Tethys Matlab Interface Cookbook

Tethys, Antioch mosaic, 3rd century from Baltimore Museum of Art

Table of Contents

1 Read me first

This document provides examples of common tasks using the Matlab interface to Tethys. In many cases, the reader will wish to provide parameters relevant to their own data, our convention is to highlight these values.

Before we get started, there are two important things that you must do:

1.1 Let Matlab know how to find the Tethys code

The Matlab client can be used to add data to the database and to use the Tethys methods for querying the database. The installer will have copied several files to the client-matlab directory which is located in the Tethys/matlab-client relative to the root folder. For “Just me” (non-administrative installs), this is relative to your account directory, which is usually c:\Users\YourLoginName. For “All users” (administrative installs), this is usually in C:\Program Files\Tethys. If you or the person who installed Tethys chose a different location, you will need to modify appropriately.
These are collected into subfolders: db, db\c and vis. The functions under db are related to accessing the database while the functions in the vis directory provide support for visualizing data.

Once Matlab has started, add the db, db\c and vis directories to your path. This can be done using Matlab’s pathtool or addpath commands. The pathtool command allows you to save the path for the next time you start Matlab. Alternatively, addpath commands can be put in the startup.m file which is executed when Matlab starts. See the Matlab documentation for details.

1.2 Set up a query handle object

All calls that interact with the Tethys database require a query handle object to be passed as the first argument. If done at a command prompt, the handle is valid for the life of the Matlab session (unless variables are cleared). In a function, the handle is valid for as long as the function executes. For all of these examples, the Tethys database should be started and you will need to know the machine on which it is running.

```
%
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
%
% for use with default server
query_h = dbInit();
```

Subsequent examples assume that a query handle has been set up and it has the name query_h, although any variable name is fine as long as it is used consistently. If you wish to use a server that does not match the name that you used during the installation process, use:

```
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
```

1.3 IMPORTANT CHANGE as of Tethys 2.5

Users of Tethys prior to 2.5 will notice a change in how results are returned. Some values in Tethys can occur more than once. For example when recording call types that occurred in a 15 minute bin, one might want to report both A and B calls for blue whales. In the past, if only one call was reported, the field name would contain the string, and if more than one call was reported, a cell array would be returned. This meant that sometimes one would reference the data as “Call”, and other times as “Call{1}” and “Call{2}”. This complicated programming logic. As a consequence, all values in the Matlab client are now returned as cell arrays.

In Tethys 2.5, the ERDDAP server is changed from coastwatch.pfeg.noaa.gov to upwell.pfeg.noaa.gov which maintains a more comprehensive NOAA wide catalog.

2 Change how species are represented.

Tethys uses Latin species names by default. Many organizations use abbreviations such as “Lo” or “Lobl” for Pacific white-sided dolphin (*Lagenorhynchus obliquidens*). The dbSeciesFmt command allows one to specify how names are written in the input to Tethys functions and how they are returned in the output. Like the dbInit function, this need only be called once per Matlab session. Before using dbSpeciesFmt, we will find out valid sets of abbreviations.

```
% We can use the query handler’s QueryTethys function to find the
% valid sets of abbreviations.  (This is an actual query to the XML
% database that does not use Matlab functions as an intermediary,
% see the full manual for details on XQuery to learn more including
% details on namespaces (the ty: which identifies that we are using
% Tethys schema:
query_h.QueryTethys(...
     'collection("SpeciesAbbreviations")/ty:Abbreviations/Name')

ans =

<Name>NOAA.NMFS.v1</Name>
<Name>SIO.SWAL.v1</Name>
```

We can use another query to see the NMFS abbreviations:

```
query_h.QueryTethys(...
```

```
    'collection("SpeciesAbbreviations")/ty:Abbreviations[Name="NOAA.N
MFS.v1"]')

ans =
<te:Abbreviations xmlns:te="http://tethys.sdsu.edu/schema/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tethys.sdsu.edu/schema/1.0 tethys.xsd">
<Name>NOAA.NMFS.v1</Name>
<Map>
  <completename>Balaenoptera borealis</completename>
  <coding>Bb</coding>
</Map>
<Map>
  <completename>Balaenoptera brydei</completename>
  <coding>Be</coding>
</Map>
<Map>
  <completename>Balaenoptera musculus</completename>
  <coding>Bm</coding>
</Map>
<Map>
  <completename>Balaenoptera physalus</completename>
  <coding>Bp</coding>
</Map>
… many more deleted …
</te:Abbreviations>
```

Here, we will set both the input (the names we provide to the system) and the output to use version 1 of the NOAA National Marine Fisheries Services abbreviations.

```
dbSpeciesFmt('Input', 'Abbrev', 'NOAA.NMFS.v1');
dbSpeciesFmt('Output', 'Abbrev', 'NOAA.NMFS.v1');
```

3 Find all projects in the database

```
%
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
%
% for use with default server
query_h = dbInit();
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
%
% Request all of the deployments
% Additional arguments could be used to restrict
% to a specific latitude range, etc.
% Note: Queries can be made using a single value ('Site', 'M') or using
% a list ('Site', {'M', 'N'}) as desired.
%
```

```
DeploymentInfo = dbDeploymentInfo(query_h);
```

RESULTS

```
% use unique in Matlab to return just the unique results of
%   your query
% the output of dbDeploymentInfo
tmp = size(unique({DeploymentInfo.Project}));
for idx = 1:tmp(2)
fprintf('%s %s\n',char(unique({DeploymentInfo(idx).Project})))
end
```

```
Aleut CINMS
```

4 List all species for which we have effort at a given site/project/etc. in our database

```
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
%
% for use with default server
query_h = dbInit();
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
%
% Request effort for a given site in the database
% Additional arguments could be used to restrict to a specific
% time, geographic location, species, etc.
% Type help dbGetEffort for details, e.g. add argument:
%   'Site', 'SiteName'
% to restrict to a specific site.
%
% A two column matrix effort is returned where each row contains
% a Matlab serial date (datenum) that represents the start and end
% of the effort.  The function datestr can be used to display the
% starts and ends in human readable format.
% For each row in effort, the details structure contains information
% about the type of effort.


[effort, details] = dbGetEffort(query_h, 'Project', 'ALEUT');
```

RESULTS

```
>> whos
  Name          Size              Bytes  Class            Attributes

  details       1x1               12464  struct
  effort        1x2                  16  double
  query_h       1x1                      dbxml.Queries
```

Using datestr converts the serial dates from effort into an easily interpretable format.

```
%
for idx=1:size(effort, 1);
   fprintf('%s %s\n', datestr(effort(idx,1)),datestr(effort(idx,2)))
end
%

27-Aug-2010 26-May-2011 08:07:00
```

The details structure array contains information from each of the XML documents that describes the effort. This includes start and end times, and an optional description. The DataSource structure contains the project, site, and deployment identifier for each deployment, which can be used to obtain additional information about a deployment with the dbDeploymentInfo() function. Algorithm contains the detection method, software, version, and parameters information. UserID is identifies the person who prepared the data. Finally, the Kind cell array contains details about what types of calls or events are being logged by this effort. This includes a species identifier a call type, and the level of detail in reporting which we call granularity. Granularity can be reported as binned (presence per specified time period), call, or acoustic encounter (beginning and end of a group of calls of the specified type).

Note that for the text values, or strings, we use a curly bracket notation to access them. In some cases, there may be more than one value (although not in this example), so in general the strings are formed in a format that supports multiple values. These are called cell arrays, see the Matlab documentation if you wish more information about cell array structures.

```
%
% List the species and calls for which there was effort for the
% query we just executed.  As we wish to print the call subtype
% and a possible group associated with a species (or other taxonomic
% designation), the loop is a little more complicated

for eidx = 1:length(details)   % For each detection effort
      % Loop through the kinds of effort and display them.
```

```
    for kidx = 1:length(details(eidx).Kind)
         try
             % not all calls have subtypes
            subtype  =  details(eidx).Kind(kidx).Parameters.Subtype{1};
         catch
            subtype = '';
         end
         try
             % Not all species descriptors have associated groups,
             % we currently use this to distinguish between groups
             % of beaked whale echolocation signals that we can
             % distinguish but not link to a specific species.
            group  =  details(eidx).Kind(kidx).SpeciesID_attr.Group{1};
         catch
             group = '';
         end
         fprintf('%d: %s %s - call: %s %s granularity %s\n', ...
                eidx, details(eidx).Kind(kidx).SpeciesID{1},
                group, ...
                details(eidx).Kind(kidx).Call{1}, subtype, ...
                details(eidx).Kind(kidx).Granularity{1});
    end
end
%
```

Result:

```
1: Human  - call: Active Sonar MFA<5kHz granularity encounter
1: Human  - call: Active Sonar Echosounder granularity encounter
1: Killer Whale  - call: Whistles  granularity encounter
```

5 Find all deployments in a given latitude range

Often studies are limited to specific geographic areas. One must be able to search for all existing data from a specific region of the earth. In this case, querying the deployment information in the Tethys database can provide a list of deployments for a given range of Latitude and Longitude.

```
%
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
%
% for use with default server
query_h = dbInit();
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
%
```

```
deployments = dbDeploymentInfo(query_h,'DeploymentDetails/Latitude',
{'>', 45},'DeploymentDetails/Latitude',{'<', 60});
```

RESULTS

```
To see a list of the deployments that meet the query criteria without
repeating sites, use the "unique" command in Matlab.

sites = vertcat(deployments.Site);
sites_unq = unique(sites)
fprintf('%s ', sites_unq{:})

BD
```

6 What is the effort for a specific deployment?

This example is very similar to the previous one, except that we are further limiting our search.

```
%
% Set the parameters for the data search.
% This example uses project, deployment, and site. Any parameter for
% an attribute of the data can be used for the data search.
% It is important to input the parameters in the correct format.
% For example, project is a string as indicated by the single quotes.
% deployment is numeric, and has no quotes.
project = 'ALEUT';
deployment = 2;
%
% Find the effort for the data parameters using dbGetEffort.m
% Effort is a matrix of Matlab serial dates containing the start and
% end times in each row.
% An array called details contains the species in the format
% set by dbSpeciesFmt
[effort details] = dbGetEffort(query_h, 'Project', project, ...
            'Deployment', deployment);
%
```

RESULTS

Example output:

```
>> whos
  Name                Size                Bytes  Class                 Attributes

  deployment          1x1                     8  double
  details             1x1                 12420  struct
  effort              1x2                    16  double
  eidx                1x1                     8  double
```

```
  group           0x0                  0  char
  idx             1x1                  8  double
  kidx            1x1                  8  double
  project         1x5                 10  char
  query_h         1x1                     dbxml.Queries
  subtype         0x0                  0  char
```

The details structure array contains information from the XML document that describes the effort, start and end times, and an optional description. The DataSource structure contains the project, site, and deployment identifier for each deployment. Algorithm contains the detection method, software, version, and parameters information. UserID is identifies the person who prepared the data. Finally, the Kind cell array contains details about what types of calls or events are being logged by this effort. This includes a species identifier a call type, and the level of detail in reporting which we call granularity. Granularity can be reported as binned (presence per specified time period), call, or acoustic encounter (beginning and end of a group of calls of the specified type).

```
>> details(1)

    XML_Document: {'dbxml:///Detections/ALEUT02BD_MF_MFAOrca_ajc'}
           Start: {[2010 8 27 0 0 0]}
             End: {[2011 5 26 8 7 0]}
     Description: {''}
      DataSource: [1×1 struct]
       Algorithm: [1×1 struct]
          UserID: {'ACummins'}
            Kind: [1×3 struct]

% Example of examining the kinds of effort conducted
>> tmp = details(1).Kind;
   for idx = 1:length(tmp)
   fprintf('%d: %s - call: %s granularity %s\n', idx, ...
     tmp(idx).SpeciesID{1}, tmp(idx).Call{1}, tmp(idx).Granularity{1});
   end
```

```
1: Homo sapiens - call: Active Sonar granularity encounter
2: Homo sapiens - call: Active Sonar granularity encounter
3: Orcinus orca - call: Whistles granularity encounter
```

Caveats: Sometimes, there are multiple efforts for the same species. As an example, running two different detectors for the same species can result in duplicate effort. When performing analyses on data, be very careful that you don’t double count. When querying effort (or detections), you can always specify queries for a specific type of effort (see the function’s help).

7 Find detections for a given date and time range

```
%
% Set up a query handler.
% This is passed to all Tethys functions that query the database
% and lets the functions know where the server is and defines
% the communication protocol.  See the Tethys manual for details.
%
% for use with default server
query_h = dbInit();
% for use with specified server use the line below
% query_h = dbInit('Server', 'yourserverName');
%
% Request all of the detections for a given date and
% time range in the database for toothed whales
% Additional arguments could be used to restrict
% to a specific species, etc.
%


detections = dbGetDetections(query_h, 'SpeciesID', 'UO', ...
  'Effort/Start',{'>', '2001-10-17T19:09:00Z'}, ...
  'Effort/End',{'<', '2009-05-19T00:00:00Z'} );
```

RESULTS

```
dates = dbSerialDateToISO8601(detections);

dates(1:5,:)  % Show first 5 starts and ends

    '2008-10-15T02:21:15Z'    '2008-10-15T03:07:30Z'
    '2008-10-15T03:18:45Z'    '2008-10-15T03:20:00Z'
    '2008-10-15T05:52:30Z'    '2008-10-15T05:53:45Z'
    '2008-10-15T06:06:15Z'    '2008-10-15T06:07:30Z'
    '2008-10-15T06:10:00Z'    '2008-10-15T06:11:15Z'
```

8 Which time periods have calls from a particular species?

```
%
% We assume the following have been called
% See example 2
% dbSpeciesFmt('Input', 'Abbrev', 'NOAA.NMFS.v1');
% dbSpeciesFmt('Output', 'Abbrev', 'NOAA.NMFS.v1');
%
% Use dbGetDetections() to find all Killer whale detections
% in the database.  Killer whales are abbreviated Oo (Orcinus
% orca) in the NOAA.NMFS.v1 abbreviation map.
[timestamps, Endp] = dbGetDetections(query_h,'SpeciesID', 'Oo');
```

The variable timestamps contains one or two columns depending upon whether the effort is for binned intervals (single column giving some time during the interval) or finding calls or encounters (start and end columns). If a query returns information that requires both single and double columns, the optional second output variable (Endp in this example), contains zeros where no end time was specified and ones otherwise.

```
% The output, timestamps, is in Matlab serial date format.
% We will convert them to ISO8601 format:  YYYY-MM-DDTHH:MM:SSZ
%
dates = dbSerialDateToISO8601(timestamps);
```

RESULTS

```
  Name                     Size                  Bytes  Class
Attributes

  Endp                     0x0                       0  double
  dates                  154x2                   46816  cell
  query_h                  1x1                          dbxml.Queries
  timestamps             154x2                    2464  double
```

9 How to find day and night, and make a diel plot for a selected time period

Scenario A: The time period and coordinates are set explicitly. These are usually derived from other queries, see scenario B.

```
%
% Assume using NOAA.NMFS.v1 (see example 2)

% Set the parameters for the data search.
% Choose the start (Time1) and
% end (Time2) times and then convert them
Time1 = '10-Jan-2011 00:00:22';
Time2 = '27-Feb-2011 15:00:22';
starttime = datenum(Time1); endtime = datenum(Time2);
%
% Set the Latitude and Longitude
% as numbers, not a string (near the Aleutian islands)
Latitude = 52.7234;     % + for north, - for south
Longitude = 174.7654;   % degrees east
% Determine when the sun is down between start and end times
night = dbDiel(query_h, Latitude, Longitude, starttime, endtime);
%
```

RESULTS

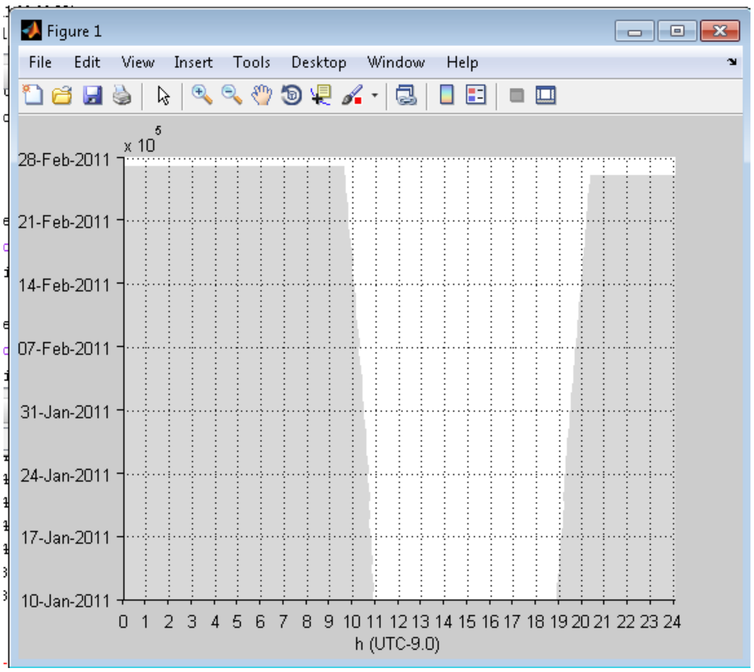
dbDiel returns the times for sunset and sunrise at the specified coordinates in matlab serial dates.

```
>> whos night
  Name        Size               Bytes  Class     Attributes
  night       48x2                  32  double

%
for idx=1:size(night, 1);
   fprintf('%s %s\n', datestr(night(idx,1)), datestr(night(idx,2)))
end
%
10-Jan-2011 03:07:22 10-Jan-2011 19:11:22
11-Jan-2011 03:09:22 11-Jan-2011 19:13:22
12-Jan-2011 03:11:22 12-Jan-2011 19:10:22
13-Jan-2011 03:13:22 13-Jan-2011 19:07:22
...
23-Feb-2011 04:30:22 23-Feb-2011 18:04:22
24-Feb-2011 04:32:22 24-Feb-2011 18:01:22
25-Feb-2011 04:34:22 25-Feb-2011 17:58:22
26-Feb-2011 04:36:22 26-Feb-2011 17:55:22
```

If a plot is desired, visPresence() is provided in the Tethys matlab functions.

```
% Plot in local time.
% All times in the database are in UTC, the offset allows
% for plots in local time
UTCOffset = -9;
%
%
% See the Tethys manual or type
% help visPresence in Matlab for more information
% on using visPresence.m
nightH = visPresence(night, 'Color', 'black', 'LineStyle',...
    'none', 'Transparency', .15,'Resolution_m', 1/60, ...
    'DateRange',[starttime, endtime],'UTCOffset', UTCOffset);
```



Scenario B:

Plot detections with day/night shown and position and time derived from the deployment(s) across which we are querying.

```
project = 'ALEUT';
deployment = 2;
species = 'Oo';    % NOAA.NMFS.v1 - Killer whale (Orcinus orca)
%
%Find the times of the detections from the query parameters using
dbGetDetections.m
detections = dbGetDetections(query_h, 'Project', project, ...
    'Deployment', deployment,'SpeciesID', species);
%
% Find first and last detection time from serial dates
starttime = min(detections(:, 1));
endtime = max(detections(:, 2));
%
% Query for deployment coordinates
%
%
```

```
% if coordinates are in your deployment information,
% these can be queried as in the line below
sensor = dbDeploymentInfo(query_h, 'Project', ...
   project,'DeploymentID', deployment);
%
% Determine when the sun is down between start and end times
% for the  Latitude and Longitude
night = dbDiel(query_h, sensor(1).DeploymentDetails.Latitude{1}, ...
   sensor(1).DeploymentDetails.Longitude{1}, starttime, endtime);
%
% Set the UTC offset for the plot.
% All times in the database are in UTC, the offset allows
% for plots in local time
UTCOffset = -9;
%

% make a plot of night times
% See the Tethys manual or type
% help visPresence in Matlab for more information
% on using visPresence.m
nightH = visPresence(night, 'Color', 'black', 'LineStyle',...
   'none', 'Transparency', .15,'Resolution_m', 1/60, ...
   'DateRange',[starttime, endtime],'UTCOffset', UTCOffset);

%
%
% add detections of selectes species to plot using visPresence.m
speciesH = visPresence(detections, 'Color', 'b', ...
    'Resolution_m', 5, 'UTCOffset', UTCOffset);
%
```

RESULTS

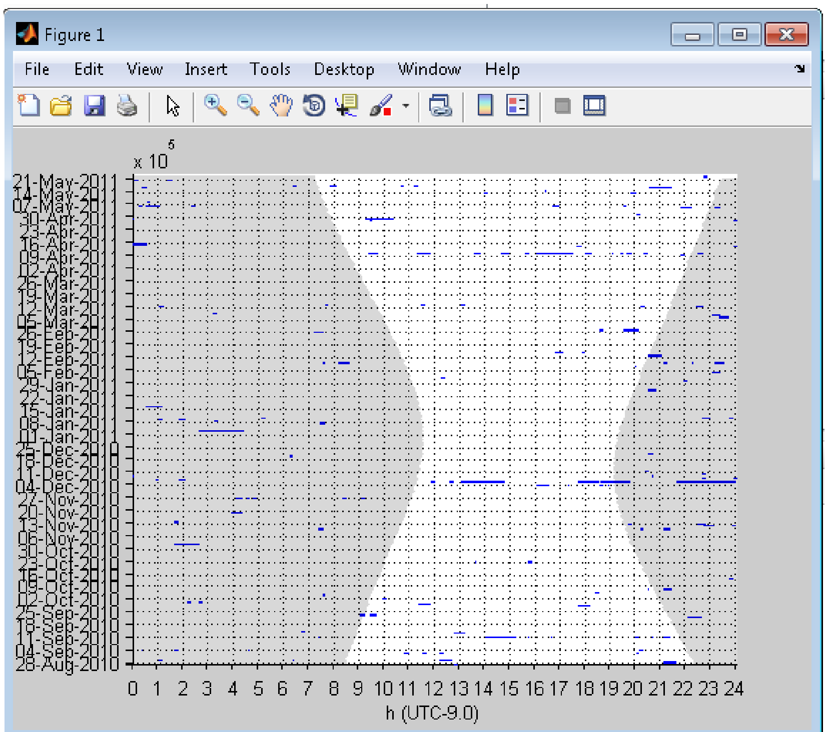


Figure 1
File Edit View Insert Tools Desktop Window Help
x 10^5
21-May-2011
14-May-2011
07-May-2011
30-Apr-2011
23-Apr-2011
16-Apr-2011
09-Apr-2011
02-Apr-2011
26-Mar-2011
19-Mar-2011
12-Mar-2011
05-Mar-2011
26-Feb-2011
19-Feb-2011
12-Feb-2011
05-Feb-2011
29-Jan-2011
22-Jan-2011
15-Jan-2011
08-Jan-2011
01-Jan-2011
25-Dec-2010
18-Dec-2010
11-Dec-2010
04-Dec-2010
27-Nov-2010
20-Nov-2010
13-Nov-2010
06-Nov-2010
30-Oct-2010
23-Oct-2010
16-Oct-2010
09-Oct-2010
02-Oct-2010
25-Sep-2010
18-Sep-2010
11-Sep-2010
04-Sep-2010
28-Aug-2010
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
h (UTC-9.0)

10 How to find lunar illumination, and make a plot for a selected time period

Scenario A: The time period and coordinates are set explicitly

```
%
% Set the parameters for the data search.
% Choose the start (Time1) and
% end (Time2) times and then convert them
Time1 = '10-Jan-2011 00:00:22';
Time2 = '27-Feb-2011 15:00:22';
starttime = datenum(Time1); endtime = datenum(Time2);
%
% Set the Latitude and Longitude
% as numbers, not string
Latitude = 52.7234;
Longitude = 174.7654;
%
% Set the time interval over which we will check.
% Interval minutes must evenly divide 24 hours, and must be no
% more than 30 m.
interval = 30;
%
% Set the getDaylight flag; false for no returns in daylight
getDaylight = false;
%
% Use dbGetLunarIllumination to get moon illumination
% returns serial date in column 1, and percent lunar illumination in 2
illu = dbGetLunarIllumination(query_h, Latitude, Longitude,...
   starttime,endtime, interval, 'getDaylight', getDaylight);
%
```

RESULTS

```
%
for idx=1:size(illu, 1);
   fprintf('%s percent illumination: %f\n',...
     datestr(illu(idx,1)),(illu(idx,2)))
end
%
10-Jan-2011 04:30:22 percent illumination: 28.883000
10-Jan-2011 05:30:22 percent illumination: 29.140000
10-Jan-2011 06:00:22 percent illumination: 29.271000
10-Jan-2011 06:30:22 percent illumination: 29.403000
10-Jan-2011 07:00:22 percent illumination: 29.539000
10-Jan-2011 07:30:22 percent illumination: 29.679000
10-Jan-2011 08:00:22 percent illumination: 29.824000
...
26-Feb-2011 17:00:22 percent illumination: 31.945000
26-Feb-2011 17:30:22 percent illumination: 31.792000
26-Feb-2011 18:00:22 percent illumination: 31.644000
26-Feb-2011 18:30:22 percent illumination: 31.500000
```

Scenario B:

Plot detections with day/night and lunar illumination shown and position and time derived from the deployment(s) across which we are querying.

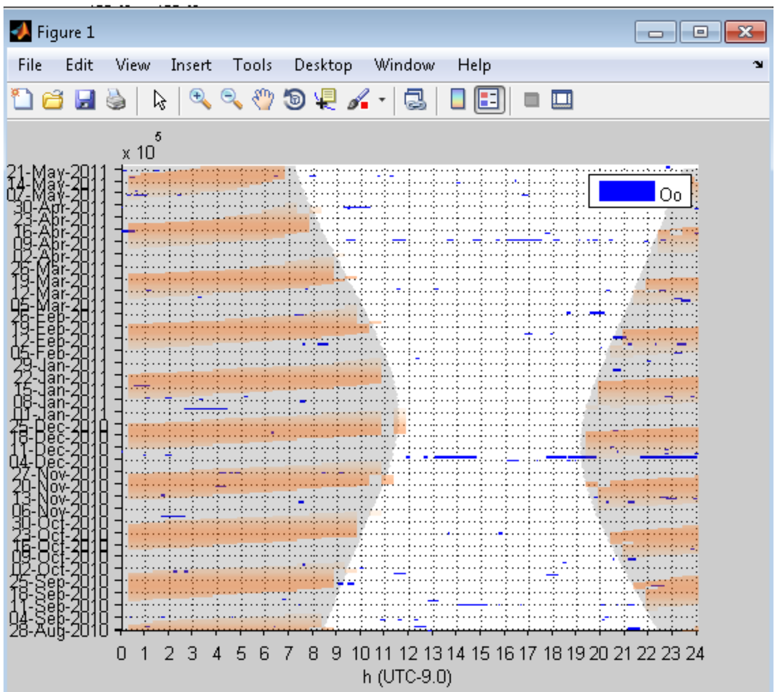
```
% Set the parameters for the data search using the output of a query
% Query parameters
project = 'ALEUT';
deployment = 2;
species = 'Oo';
%
% Find the times of the detections from the query parameters
% using dbGetDetections.m
detections = dbGetDetections(query_h, 'Project', project, ...
   'Deployment', deployment,'SpeciesID', species);
%
% Find start and end times
%  times should be in serial dates
starttime = min(detections(:, 1));
endtime = max(detections(:, 2));
%

% Pull in coordinates from deployment information
sensor = dbDeploymentInfo(query_h, 'Project', ...
   project,'DeploymentID', deployment);
Lat = sensor.DeploymentDetails.Latitude;
Long = sensor.DeploymentDetails.Longitude;

% Set the time interval over which we will check.
% Interval minutes must evenly divide 24 hours, and must be no
% more than 30 m.
interval = 30;
%
% Set the getDaylight flag; false for no returns in daylight
getDaylight = false;
%
% Use dbGetLunarIllumination to get moon illumination
% returns serial date in column 1, and percent lunar illumination in 2
illu = dbGetLunarIllumination(query_h, Lat, Long,...
   starttime,endtime, interval, 'getDaylight', getDaylight);
%
% Set to zero for GMT, we'll plot in local time
UtcOffset = -9;
% Determine when the sun is down between start and end times
% for the  Latitude and Longitude
night = dbDiel(query_h, Lat, Long, starttime, endtime);
%
% make a plot of night times
% See the Tethys manual or type
% help visPresence in Matlab for more information
% on using visPresence.m
nightH = visPresence(night, 'Color', 'black', 'LineStyle',...
   'none', 'Transparency', .15,'Resolution_m', 1/60, ...
   'DateRange',[starttime, endtime],'UTCOffset', UTCOffset);

%
```

```
% add detections of selectes species to plot using visPresence.m
speciesH = visPresence(detections, 'Color', 'b','Resolution_m', 5,
'UTCOffset', UTCOffset);
%
% add in the amount of lunar illumination to the plot
lunarH = visLunarIllumination(illu, 'UTCOffset', UtcOffset);
%
% add a legend for the species
legendH = legend(speciesH(1), species);
%
```



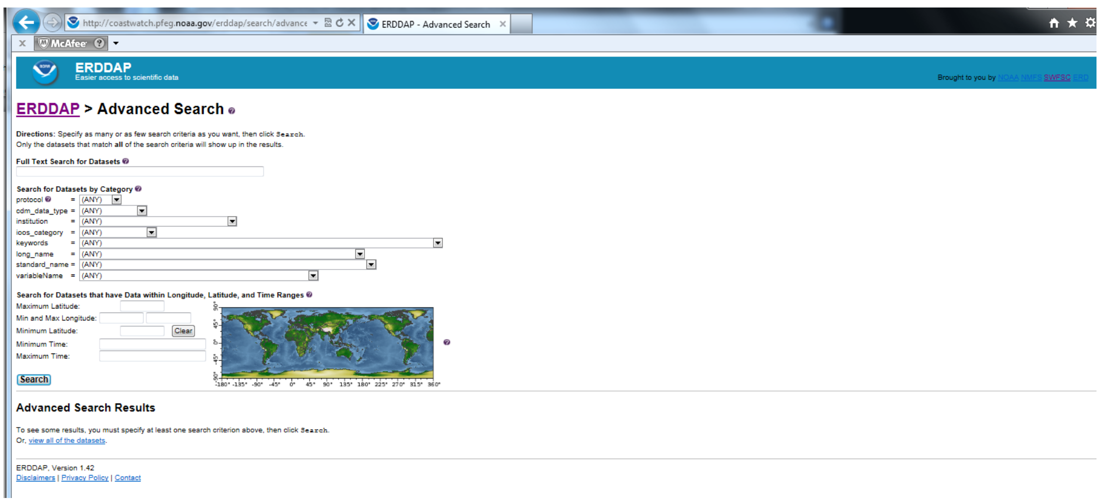
11 Find an environmental data set

Tethys is designed to interface with the NOAA Environmental Research Division Data Access Program (ERDDAP). This allows users to choose any of the data sets hosted through ERDDAP and bring the data into Matlab based on a Tethys query.

For more information on ERDDAP, see http://coastwatch.pfeg.noaa.gov/erddap/index.html and the Tethys Manual section 3.4.6.3.

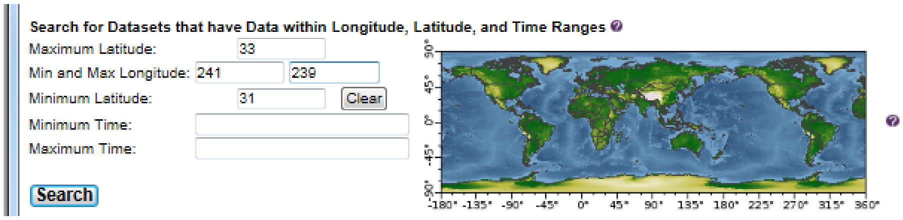
To explore available data, an Advanced Search is suggested. The ERDAP search page can be found by going to the ERDAP server directly: https://upwell.pfeg.noaa.gov/erddap/search/advanced.html, or by asking the Tethys Matlab client to open the search page for you:

```
dbERDDAPSearch(query_h)
```



Users can search for available data using space and/or time limits. To search spatially, users can input Latitude and Longitude limits, or click a box on a map of the earth. To search for data collected within a specified time frame, the minumum and maximum time can be added to the search.

For example, to find all of the available data for a region of the Pacific, the spatial limits are set to the Latitude between 31 and 33 degrees, and Longitude between 239 and 241 degrees.



The results include 298 matching data sets. For more information about a specific data set, there are columns with a summary of the metadata and complete background information.

Advanced Search Results

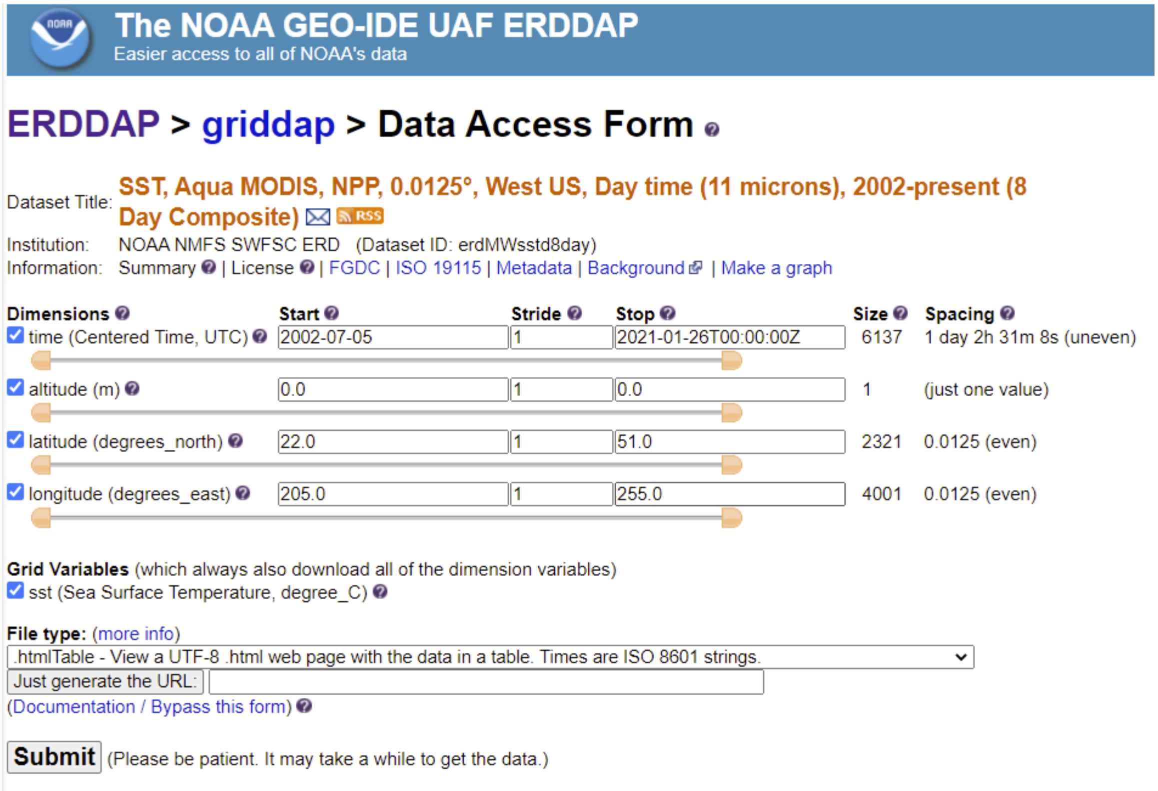
298 matching datasets, listed in alphabetical order.

Grid DAP Data	Sub-set	Table DAP Data	Make A Graph	W M S	Title	Sum-mary	FGDC, ISO, Metadata	Back-ground Info	RSS	E mail	Institution	Dataset ID
data			graph	M	AMSRE Model Output, obs4MIPs NASA-JPL, Global, 1 Degree, Monthly		F I M	background	RSS		Remote Sensing ...	jplAmsreSstMon
		data	graph		Argo Float Data from the APDRC DAPPER Server		F I M	background	RSS		NOAA PMEL	apdrcArgoAll
data			graph	M	AVISO Altimetry and Niiler Climatology, Global, SSH, Monthly, Historical		F I M	background	RSS		NOAA/NESDIS/Oce ...	noaa_pifsc_dc75_b265_fef6
data			graph	M	AVISO Altimetry and Niiler Climatology, Global, SSH, Weekly, Historical		F I M	background	RSS		NOAA/NESDIS/Oce ...	noaa_pifsc_9c36_df47_3dd4
data			graph	M	AVISO Altimetry and Niiler Climatology, Global, SSH, Weekly, Near Real Time		F I M	background	RSS		NOAA/NESDIS/Oce ...	noaa_pifsc_1988_b28f_a82d
data			graph	M	AVISO Model Output, obs4MIPs NASA-JPL, Global, 1 Degree, Monthly		F I M	background	RSS		Centre National ...	jplAvisoSshMon
data			graph	M	CCMP Winds, Atlas FLK v1.1 Derived Surface Wi ... el 3.5a), Global, 0.25 Degree, 5-Day Averages		F I M	background	RSS		NASA JPL	jplCcmp35aWindPentad
data			graph	M	CCMP Winds, Atlas FLK v1.1 Derived Surface Winds (Level 3.5a), Global, 0.25 Degree, Monthly		F I M	background	RSS		NASA JPL	jplCcmp35aWindMonthly
data			graph	M	Chlorophyll-a Deviation, Orbview-2 SeaWiFS, West US (8 Day Composite)		F I M	background	RSS		NOAA CoastWatch ...	erdSHcdev8day
data			graph	M	Chlorophyll-a, Aqua MODIS, NPP, Global, Science Quality (1 Day Composite)		F I M	background	RSS		NOAA CoastWatch ...	erdMHchla1day
data			graph	M	Chlorophyll-a, Aqua MODIS, NPP, Global, Science Quality (8 Day Composite)		F I M	background	RSS		NOAA CoastWatch ...	erdMHchla8day
data			graph	M	Chlorophyll-a, Aqua MODIS, NPP, Global, Science Quality (Monthly Composite)		F I M	background	RSS		NOAA CoastWatch ...	erdMHchlamday
data			graph	M	Chlorophyll-a, Aqua MODIS, NPP, Pacific Ocean (1 Day Composite)		F I M	background	RSS		NOAA CoastWatch ...	erdMBchla1day

If the same geographic limits are used along with a minimum time of 2010-03-01T00:00:00Z and a maximum time of 2011-01-01T00:00:00Z, 183 data sets are returned.

The search can also be narrowed by keyword. To find a list of the keywords, see the ERDDAP website to use the pull down menu in the advanced search. By typing “sst” in the keyword drop-down, the return is 65 data sets.

Once a dataset is chosen, the DatasetID (from the last column on the right of the search return) is used with dbERDDAP.m to download the selected data. In the next example, the DatasetID is **erdMWsstd8day**. The variables and attributes for each dataset are described in the first column of the returns. By clicking on the “data” under the first column, a complete list of variables and of the dimensions needed for a query can be viewed.


The NOAA GEO-IDE UAF ERDDAP
Easier access to all of NOAA's data

ERDDAP > griddap > Data Access Form

Dataset Title: SST, Aqua MODIS, NPP, 0.0125°, West US, Day time (11 microns), 2002-present (8 Day Composite) RSS
Institution: NOAA NMFS SWFSC ERD (Dataset ID: erdMWsstd8day)
Information: Summary | License | FGDC | ISO 19115 | Metadata | Background | Make a graph

Dimensions	Start	Stride	Stop	Size	Spacing
time (Centered Time, UTC)	2002-07-05	1	2021-01-26T00:00:00Z	6137	1 day 2h 31m 8s (uneven)
altitude (m)	0.0	1	0.0	1	(just one value)
latitude (degrees_north)	22.0	1	51.0	2321	0.0125 (even)
longitude (degrees_east)	205.0	1	255.0	4001	0.0125 (even)

Grid Variables (which always also download all of the dimension variables)
sst (Sea Surface Temperature, degree_C)

File type: (more info)
.htmlTable - View a UTF-8 .html web page with the data in a table. Times are ISO 8601 strings.
Just generate the URL:
(Documentation / Bypass this form)

Submit (Please be patient. It may take a while to get the data.)

We see that the data are indexed by time, altitude, latitude, and longitude. The limits for each index variable are provided, for example, at the time of this wriing these data are available between July 2002 and January 2021. The spacing tells us that the data measurements are taken about 26.5 h apart (due to the satellite's orbital path) and there are 6,137 of these nearly daily observations. There is data for exactly one altitude (sea level), and we can see the geographic extent of the data as well as the spacing between measurements (0.0125 degrees).

For the erdMWsstd8day data set, sst is the name of the variable. The required dimensions are time, altitude, latitude, and longitude. When used with Matlab function **dbERDDAP**, the DatasetID is followed by a question mark and then the variable.
In the next example, erdMWsstd8day?sst is used to download data.

Once you are familiar with ERDDAP search terms, you can specify them in dbERDDAPSearch, separating each term by an ampersand (&). For example, the Integrated Ocean Observing System (IOOS) maintains a set of data categories that include terms such as bathymetry, co2, currents, dissolved_o2, ice_distribution, etc. We can use ioos_category=bathmetry to search for bathymetry. ERDDAP's standard_name provides a wide set of variable names where spaces between words are replaced with underscores. These are reasonably intuitive, e.g. sea_surface_temperature. More details on these search terms can be found at any ERDDAP server, e.g. the [NOAA GEO-IDE UAF ERDDAP]() server; follow the search for dataset by category links.

As an example query, suppose we wished to search for sea surface temperature provided by the National Centers for Environmental Information (NCEI). We would run the query:

We would run the query:

```
dbERDDAPSearch(query_h, ...
   'keywords=sea_surface_temperature&institution=ncei')
```

For most users however, the easiest way to find data is to simply open ERDAPs search site:

```
dbERDDAPSearch(query_h)
```

12 Pull in data from ERDDAP for a specific spatial location and/or time

Suppose we wished to access a subset of the sea surface temperature (SST) dataset identified in the previous section: **erdMWsstd8day?sst** . From the previous example, we know that the SST data are indexed by time, altitude, latitude (degrees North) and longitude (degrees East). We can use function **dbERDDAP** to pull in the data.

In this example, we will search for data on a small grid of the coast of southern California on November 13th, 2012. We need to specify each axis. ERDAP requires a set of array indices indicating the portion of the data set to retrieve. As there are four index variables, there will be four sets of array indices [ ]. Each array index must have the form

[start:stride:end]

where start is either an index number into the data or is specified in the units of measure, e.g. a timestamp for a time axis. When referencing by unit, you must surround the value by parentheses ( ). We indicated that we wanted to retrieve data from November 13th, 2012. We would specify this as follows using a standard time notation: YYY-MM-DDTHH:MM:SSZ where Z indicates that the time is in UTC.

```
[(2012-11-13T00:00:00Z):1:(2012-11-13T00:00:00Z)]
```

Subsequent indices are handled similarly.

```
data = dbERDDAP(query_h, 'erdMWsstd8day?sst[(2012-11-
13T00:00:00Z):1:(2012-11-
13T00:00:00Z)][(0.0):1:(0.0)][(33.47):1:(33.59)][(240.7):1:(240.80)]');
```

RESULTS

The returned data is a structure that contains three fields:
data =

  struct with fields:

    Axes: [1×1 struct]
    Data: [1×1 struct]
    dims: [9 10 1 1]

- Axes – Description of the axes
- Data – A structure with the returned data.
- dims – Provides the dimensions of the data

The Axes structure contains fields that describe the data axes:

- names – An ordered cell array of the axes names indicating how the returned data are organized, e.g. data.Axes.names{1} is 'longitude' with the remaining values being latitude, altitude, and time.
- units – Cell array of measurement units for each axis. In this case: degrees_east, degrees_north, m, and UTC.
- types – Cell array of data types for the axes units. Here, all units are doubles except for the time measurements which are coded as serial dates (datenum).
- values – The value that corresponds to the axes. For example, to see the latitudes, we would examine the 2nd cell entry:

```
>> data.Axes.values{2}

ans =

    33.4750    % first index into data along axis 2 corresponds to this latitude
    33.4875    % second index into data along axis 2 corresponds to this latitude
    33.5000    % and so on…
    33.5125
    33.5250
    33.5375
    33.5500
    33.5625
    33.5750
    33.5875
```

The dims field simply lists the dimensions of the axes.

The Data field contains the actual data and contains the following information:

- names – Cell array of variables returned. As we only requested SST, data.Data.names{1} is 'sst'.
- units – Cell array indicating the unit of measurement for each variable name (degree_C in this case).

- types – Cell array describing the data type for each value. Here, the data were returned as type 'float'. Even though Matlab stores these as double precision numbers, ERDAP stored them as single precision numbers. If numerical precision to many digits is important to your research question, this may be important to you.
- values – A cell array with one entry per variable returned. As we only requested SST, values{1} contains a 9 x 10 matrix of doubles that corresponds to the temperatures we requested.

The sst data can now be plotted using the mapping toolbox, or saved for use in other software packages.

Here's a more complex example that finds bathymetry in 400 km^2 in the Southern California Bight.

```
% dbERDDAP example
% Find bathymetry about a point.

% Center point degrees north, east
% This is in the Southern California Bight
center = [33.515  240.753];

range_km = 20; % defines square with width/height 2*range_km
try
    % Use mapping toolbox to convert to degrees
    delta_deg = km2deg(range_km);
catch e
    % mapping toobox unavailable, hardcode it
    % Note that this try catch block is only required to make
    % this example work even when someone does not have the
mapping
    % toolbox.  We are simply setting delta_deg
    delta_deg = 0.1799;
    fprintf('No mapping toolbox, assuming %f km = %f deg', ...
        range_km, delta_deg);
end

% Compute extent around center
box = [center - delta_deg; center + delta_deg];

% Determine search criteria
geospec = sprintf('minLat=%f&maxLat=%f&minLon=%f&maxLon=%f',
box(:))];
criteria = ['ioos_category=bathymetry', '&', geospec];

% Running this query, we see that there are at least a half dozen
% bathymetry data sets.  In this case, wee
dbERDDAPSearch(query_h, criteria);
```

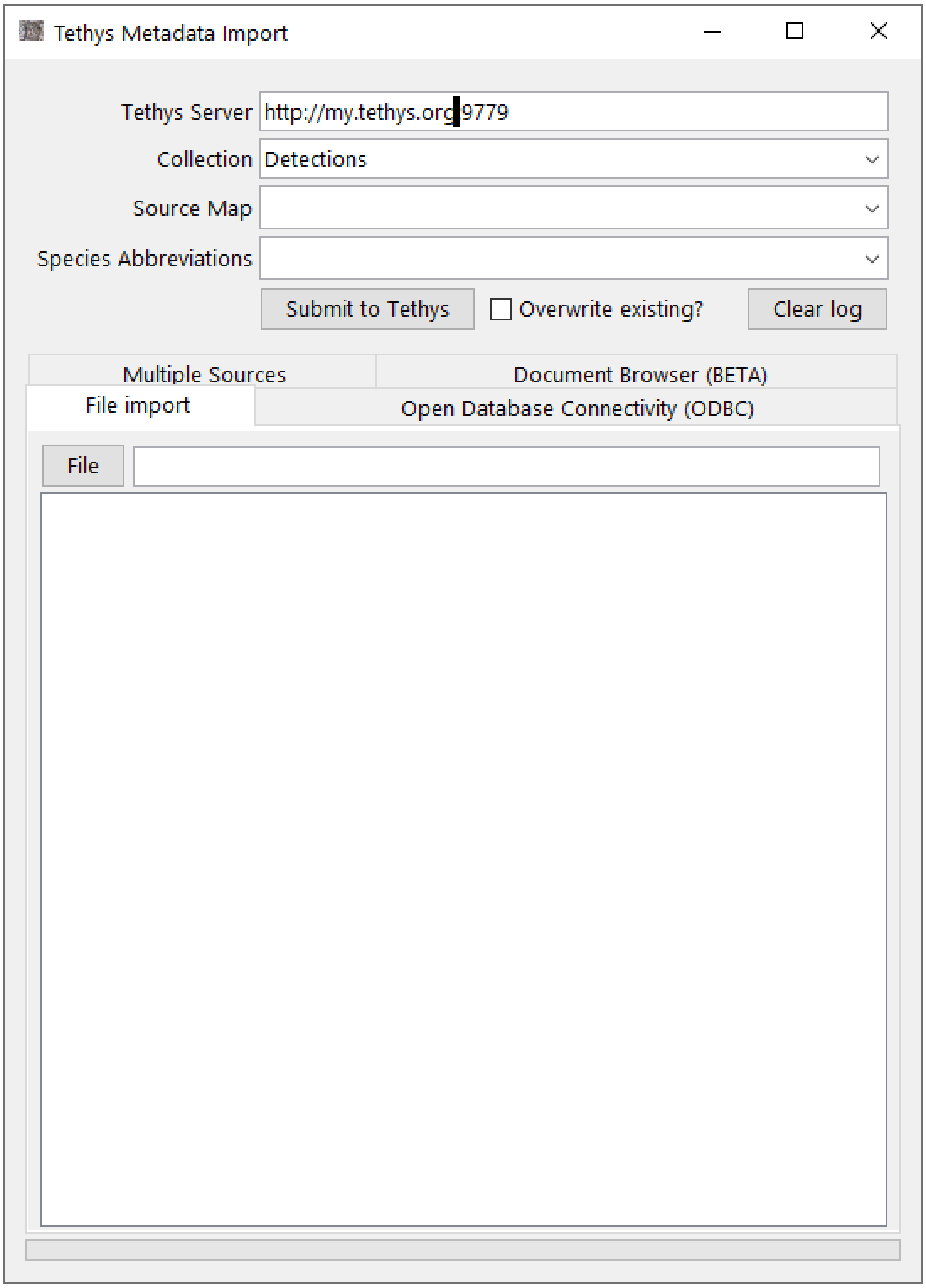
```
% Looking at the results of the search, we see that there are
% a number of datasets that might meet our purposes.  We settle
% on the San Diego, California Tsunami Forecast Grids for
% MOST Model:
% noaa_ngdc_ec9d_8632_6ca3 which has unevenly spaced data sampled
% approximately 0.017 degrees apart.
dataset = 'noaa_ngdc_ec9d_8632_6ca3';
geoind = sprintf('[(%f):1:(%f)][(%f):1:(%f)]', box(:));
data = dbERDDAP(query_h, sprintf('%s?bathy%s', dataset, geoind));

% The bathymetry data are in data.Data.values{1}
```

13 How to add a new file to the Tethys database

```
% Start a GUI for uploading files to the database
% for use with default server
dbSubmit();
% for use with specified server use the line below
% dbSubmit ('Server', 'yourserverName');
%
```

RESULTS



The first input is your server address, in many cases this will be your local host address which can be written as http://127.0.0.1:9779.
Next choose the appropriate collection from the drop down. This includes Detections, Deployments, Ensembles, Localizations, Source Maps and Species Abbreviations.

The third drop down is to indicate the appropriate source map. Source maps provide directions on how data contained in your documents to be submitted are mapped to Tethys when your data are not already in Tethys ready XML format.

If there is a source map listed in your input file (for example, a Detections Excel Sheet under the Metadata tab would list the parser) you can choose "Embedded in data". The Source Map needs to be part of the Tethys server, if it is new, you will need to import the Source Map first. When species identifiers are in the input data, select the species abbreviation set when using your local set of abbreviations.

To select the file to be added to Tethys, there are several tabs available on the GUI. To add an individual file from a network location, click on the "File import " button and navigate to the file to be added to Tethys. Click the "Submit to Tethys" button and your document will be submitted, with confirmation or errors displayed in the message areas. If you wish to overwrite an existing Tethys document, click the overwrite existing checkbox. Otherwise, trying to submit a document twice will fail.

More details on the other tabs can be found in the Tethys manual. Briefly multiple sources allows one to combine data from multiple files or data bases into one document. The ODBC tab allows one to import data from databases and requires that the source map contain database queries. ODBC allows for one to treat many types of data as if they were a database. As an example, one can import Excel documents using this interface.

14 View attached images in a web browser

Many Detections include an audio file or image file which are attached when adding the Detections to the Tethys database. One way to view an attached image is using the REST server component of Tethys. First, the Tethys server needs to be running. In this example, the server is running as localhost with port 9779.

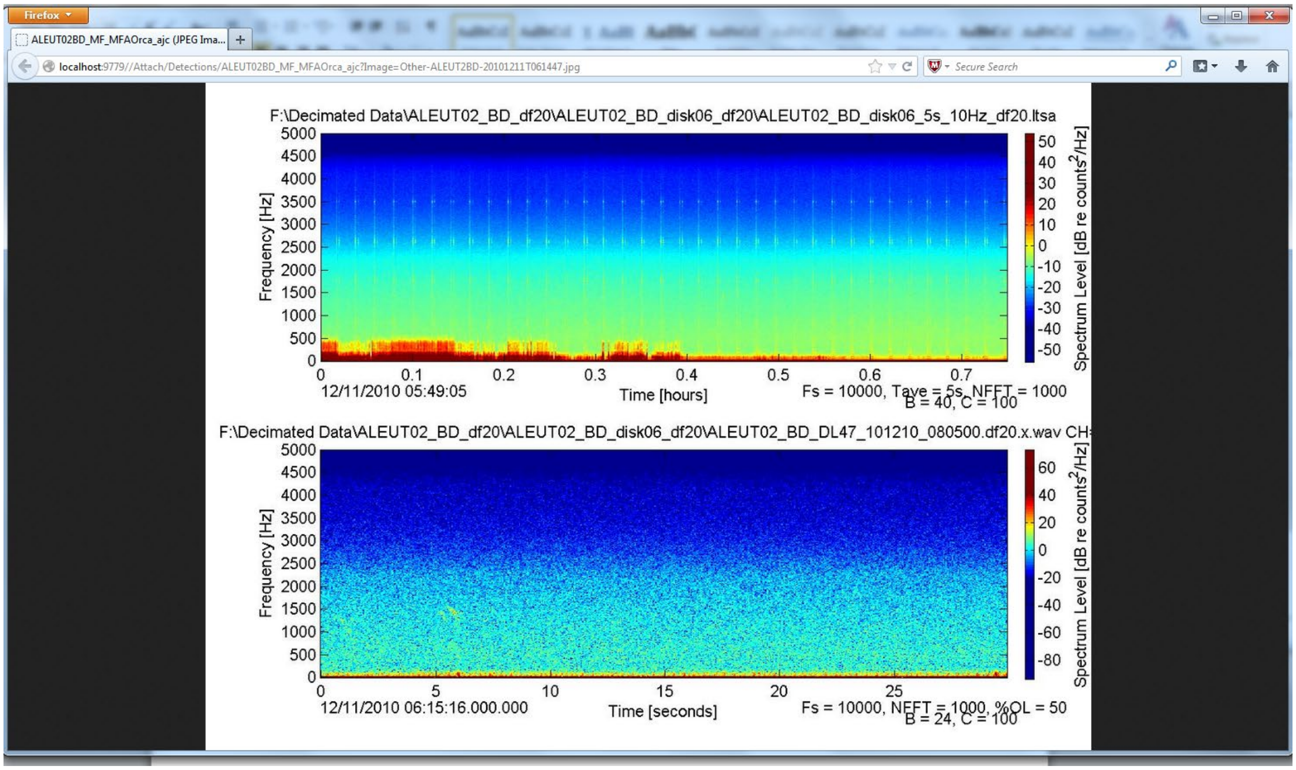
Next, open a web browser such as Firefox. To view an image, type in the address using the server location. Note that the document ID does not have the file type suffix (such as .xls) but that the name of the image file does have the file type.

For attached images you will need the following information:

Server	localhost
Port	9779
Collection	Detections
Detections doc ID	ALEUT02BD_MF_MFAOrca_ajc
Image name	Other-ALEUT2BD-20101211T061447.jpg

http://localhost:9779//Attach/Detections/ALEUT02BD_MF_MFAOrca_ajc?Image=Other-ALEUT2BD-20101211T061447.jpg

Results:



ALEUT02BD_MF_MFAOrca_ajc (JPEG Ima...
localhost:9779//Attach/Detections/ALEUT02BD_MF_MFAOrca_ajc?Image=Other-ALEUT2BD-20101211T061447.jpg
F:\Decimated Data\ALEUT02_BD_df20\ALEUT02_BD_disk06_df20\ALEUT02_BD_disk06_5s_10Hz_df20.ltsa
Frequency [Hz]
Spectrum Level [dB re counts²/Hz]
12/11/2010 05:49:05
Time [hours]
Fs = 10000, Tave = 5s, NFFT = 1000
B = 40, C = 100
F:\Decimated Data\ALEUT02_BD_df20\ALEUT02_BD_disk06_df20\ALEUT02_BD_DL47_101210_080500.df20.x.wav CH
Frequency [Hz]
Spectrum Level [dB re counts²/Hz]
12/11/2010 06:15:16.000.000
Time [seconds]
Fs = 10000, NFFT = 1000, %OL = 50
B = 24, C = 100