Geodise Database Toolbox for Matlab Tutorial
Version 0.3

J. L. Wason, Z. Jiao, and S. J. Cox
{j.l.wason, z.jiao, sjc}@soton.ac.uk
Southampton Regional e-Science Centre
School of Engineering Sciences
University of Southampton
Highfield, Southampton SO17 1BJ
United Kingdom

1. Please follow the installation and configuration instructions in "Geodise Database Toolbox for Matlab Installation and Setup" (install.txt in the distribution).

2. Start Matlab.

3. Generate your proxy certificate.

   ```
   >> gd_createproxy
   ```

   A new window pops up, here you need to enter your certificate passphrase and click 'Create' button. After the proxy has been generated, click 'Cancel' and press 'Enter' at the Matlab prompt.

4. Archive a file with some metadata.

   Create a metadata structure containing some information that describes your file. This can be any combination of doubles, strings, arrays, cell arrays, complex numbers and substructures.

   ```
   >> m.model.name = 'test_design';
   >> m.model.params = [1 4.7 5.3];
   >> m.product = 25.5431;
   ```

   Add some standard information (localName, format, comment or version) about the file.

   ```
   >> m.standard.comment = 'Test design model file';
   >> m.standard.version = '1.2.0';
   ```

Archive the file and metadata.

```
>> fileID = gd_archive('/home/jlw/myfile.txt', m)
fileID =
myfile_txt_4957b83b-322c-4f06-822c-4da38791cdfa
```

In addition to the optional metadata structure, `gd_archive` takes a string representing the full path name of a local file and stores it on a remote file store (specified in the `<user_home>/.geodise/ClientConfig.xml` file). An ID is returned which is a unique handle that can be used to retrieve the file.

The metadata is stored in a database and can be later queried to help you find relevant files. Some additional metadata is automatically generated and stored in the 'standard' substructure. This consists of localName (the original name of the file), byteSize, format, archiveDate, createDate (when the original file was created/modified) and userID. You can specify your own overriding values for standard.localName and standard.format if you prefer.

5. Query file metadata.

   A call to `gd_query` returns a cell array of structures, one for each matching result.

```
>> result = gd_query('standard.version=1.2.0 & product>25.4')
result =
    [1x1 struct]

>> result{1}
ans =
    standard: [1x1 struct]
       model: [1x1 struct]
     product: 25.5431
```

6. Display query results.

   `gd_display` is a convenient way to view your query results.

```
>> gd_display(result)

*** Content of the structure result{1} (Total structures: 1) ***
  standard.ID: myfile_txt_4957b83b-322c-4f06-822c-4da38791cdfa
  standard.localName: myfile.txt
  standard.byteSize: 34
  standard.format: txt
  standard.createDate: 2003-09-18 17:08:31
  standard.archiveDate: 2003-12-09 15:09:48
  standard.userID: jlw
  standard.comment: Test design model file
  standard.version: 1.2.0
```

```
  standard.datagroups:
  model.name: test_design
  model.params:
    1.0000    4.7000    5.3000
  product: 25.5431
*** No more results. ***
```

7. Query using wildcards

   To search for some text within a metadata value use the 'like' operator together with % to specify any characters, or _ to specify one character.

   ```
   >> gd_query('standard.comment like %design m_del%');
   ```

   The * wildcard can be used to represent an anonymous subfield, or any number of subfields if it appears at the beginning.

   ```
   >> gd_query('*.name = test_design');
   ```

8. Query GUI.

   Use `gd_query` without any input arguments to use the graphical query interface. Hyperlinks are supported for downloading and browsing data. See Fig. 1.

9. Retrieve a file to the local filesystem.

   A file can be retrieved to the local filesystem by specifying its unique ID. This string is returned when the file is archived and also appears in the metadata query results as standard.ID.

   ```
   >> ID = result{1}.standard.ID;

   % Retrieve to a specified file location
   >> gd_retrieve(ID, '/home/jlw/myfile2.txt')
   ans =
   /home/jlw/myfile2.txt

   % Retrieve to a specified directory (the original file
   % name is used)
   >> gd_retrieve(ID, '/home/jlw/filesdir')
   ans =
   /home/jlw/filesdir/myfile.txt
   ```

10. Archive, query and retrieve Matlab variables.

    ```
    >> v.width = 12;
    >> v.height = 6;
    >> metadata.standard.comment = 'measurements variable';
    >> varID = gd_archive(v, metadata);
    ```

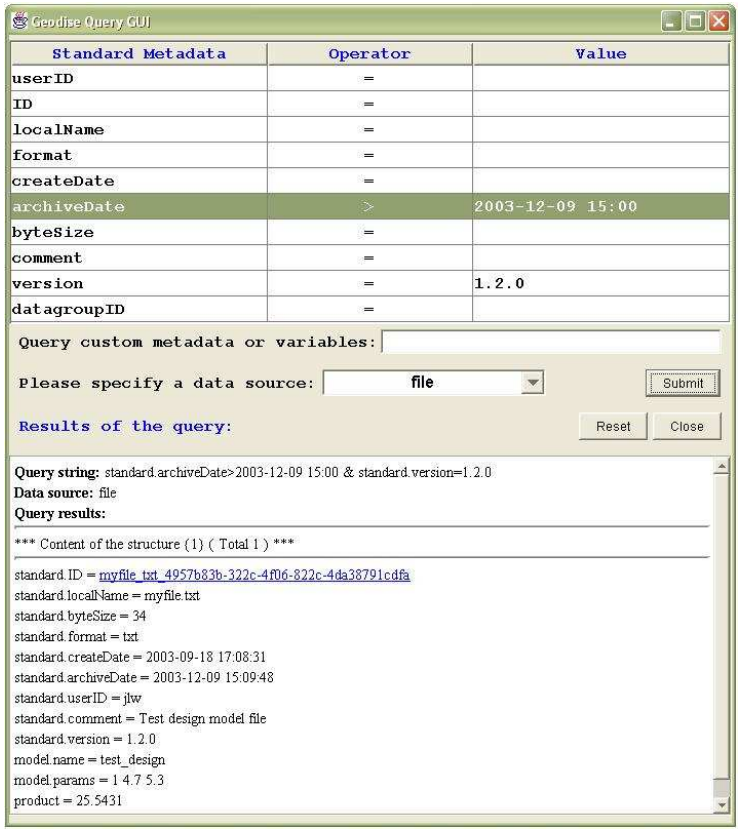

Figure 1: Query GUI.

```
% If the archived variable is a structure it can be
% queried
>> gd_query('height=6', 'var');

% You can also query a variable's metadata
>> gd_query('standard.comment=measurements variable', ...
>> 'varmeta');

>> x = gd_retrieve(varID)
x =
     width: 12
    height: 6
```

11. Grouping data.

    Related data can be logically grouped together using a datagroup.

```
% Specify metadata that applies to the whole group
>> group_metadata.standard.comment = ...
```

```
>> 'Group of data for experiment 123';

% Create a datagroup, giving it a name
>> groupID=gd_datagroup('Experiment 123',group_metadata);

% Add archived files or variables to the datagroup
>> gd_datagroupadd(groupID,fileID);
>> gd_datagroupadd(groupID,varID);

% Archive a new file (with no metadata this time) and
% add it to the datagroup.
>> gd_archive('/home/jlw/anotherfile.txt',[],groupID);

% The datagroup metadata now contains references to the
% files and variables it contains.
>> result=gd_query('standard.datagroupname=Experiment 123',...
>> 'datagroup');
>> gd_display(result)


*** Content of the structure result{1} (Total structures: 1) ***
  standard.ID: dg_1ef2641f-928f-40d9-a575-7f229856493f
  standard.datagroupname: Experiment 123
  standard.archiveDate: 2003-12-09 15:44:57
  standard.userID: jlw
  standard.comment: Group of data for experiment 123
  standard.datagroups:
  standard.subdatagroups:
  standard.files.fileID:
     myfile_txt_4957b83b-322c-4f06-822c-4da38791cdfa
  standard.files.fileID:
     anotherfile_txt_06e8cd18-974s-49ff-ba3b-d9da7ca6e057
  standard.vars.varID:
     var_b373b432-127b-4a13-9f6e-a91c8f33f5b0
*** No more results. ***
```

Metadata for the files and variables also contain references to the datagroup(s) they belong to.

Datagroups can be added to other datagroups to create a hierarchy.

```
>> parentDatagroupID = groupID;
>> childDatagroupID = gd_datagroup('child datagroup');

% Add child datagroup to parent datagroup
>> gd_datagroupadd (parentDatagroupID, childDatagroupID);

% Find all the datagroups that are in parent datagroup
>> r1 = gd_query (['standard.datagroups.datagroupID=' ...
```

```
>> parentDatagroupID], 'datagroup')

% Find all the datagroups that contain child datagroup
>> r2 = gd_query (['standard.subdatagroups.datagroupID=' ...
>> childDatagroupID], 'datagroup')
```

12. Granting access to data.

    The `gd_addusers` function allows you to grant other users permission to query particular files, variables and datagroups that you own. These users may also retrieve the variables to their local Matlab workspace and the files to their local filesystem (providing they have read permission for the appropriate directory on the Globus file server, e.g. utp-10).

    In the following examples replace user 'bob' with an existing user.

    ```
    >> users = {'bob'};
    >> gd_addusers(ID, users);
    ```

    Access may also be granted as part of the metadata when a file or variable is archived, or when a datagroup is created.

    ```
    >> m.access.users = {'bob'};
    >> gd_archive('/home/jlw/myfile.txt', m);
    ```

13. Further information.

    All of these functions have help information which can be viewed by using the help command in Matlab.

    ```
    >> help gd_datagroupadd

       gd_datagroupadd Adds an archived file, variable or
       subdatagroup to a datagroup.

       gd_datagroupadd (datagroupID, dataID)
       The datagroup is specified with the datagroupID handle
       and the data or subdatagroup is specified with the dataID
       handle. The datagroup(s) must have been created with
       gd_datagroup and the data must be file or a variable that
       was archived using gd_archive.
    ```

    Further descriptions and examples for each function are available in the `<DatabaseToolbox_install>/doc` directory.