TUTORIAL

Institute for Biodiversity and Ecosystem Dynamics

Computational Biogeography and Physical Geography

University of Amsterdam

April 2007

UvA Universiteit van Amsterdam

IBED

vl-e virtual laboratory for e-science

1 Introduction

This tutorial is about a set of MATLAB functions with which one can display spatially and temporally distributed data within Google Earth. Taken together, these functions are known as the Google Earth Toolbox.

It may be argued that converting the results of models and measurements to be able to view them in Google Earth is unnecessary since they can also be viewed within MATLAB itself. However, using Google Earth for evaluation of data can have some major advantages, a few of which are listed below:

- very intuitive tools for navigating the view
- retrieval of data from objects within the viewer
- quick rendering of large files
- interactively choosing which objects should be displayed
- dynamic representation of time-variant data

1.1 Setting up Google Earth

To avoid problems with displaying and interpreting your Google Earth files, it may be useful to have your copy of Google Earth set up according to the following specifications:

- Click 'Tools' in Google Earth's menu and choose 'Options…'. On the tab named '3D View' check that 'Graphics Mode' is set to 'OpenGL'.
- On the same tab, set the latitude/longitude format to 'Degrees' under 'Show Lat/Long'.
- Also on this tab, set the units of elevation to 'Meters' under 'Show Elevation'.

1.2 KML: Google Earth's language

The great flexibility of Google Earth stems from its use of XML-based text files, known as KML-files. Such a file typically contains a series of objects, such as polygons, lines and points. Each of these objects is represented within a KML-file by its KML-tags. For example, we may find a polygon object:

```
<Polygon>
...
</Polygon>
```

Or a line object:

```
<LinearRing>
...
</LinearRing>
```

Properties such as line width, coordinates, and polygon color may be specified using additional tags. A line may thus be assigned a line style as follows:

```
<LineStyle>
   <color>ffffffff</color>
   <width>1.00</width>
</LineStyle>
```

Which represents a fully opaque, white line of width 1.

Of course, writing KML-tags by manually typing them in a text editor is not practical when working with data sets containing more than a few objects. Fortunately though, the task of writing these files can be fully automated using the Google Earth Toolbox.

2 MATLAB to Google Earth Toolbox – tutorial

2.1 Toolbox structure

The core of the toolbox is formed by a set of about 20 MATLAB functions (*.m files) located in the toolbox root folder ('googleearth/'). The majority of these functions generate character arrays in concordance with the KML standard. For each of the m-files, a help file has been included which can be viewed from within the MATLAB help browser. The help files are located in 'googleearth/html'. In addition to the help files, demonstrations have been included ('googleearth/demo'). Many of these demos write their output to a separate folder ('googleearth/kml') in order to avoid contaminating directories with files that do not belong there. The folder 'googleearth/tutres' contains resource files used in this tutorial. Folder 'googleearth/data' contains additional files such as icon images and Collada models. Folder 'googleearth/doc' contains files pertaining to the printable documentation; e.g this document. Finally, a folder 'googleearth/tmp' has been included, which is used by some functions to write temporary data to. The contents of this folder are automatically emptied by some functions, therefore you should not save important data in it.

2.2 Adding the toolbox to the path

To be able to use the functions located in the toolbox, you first need to specify where the toolbox is located, for example:

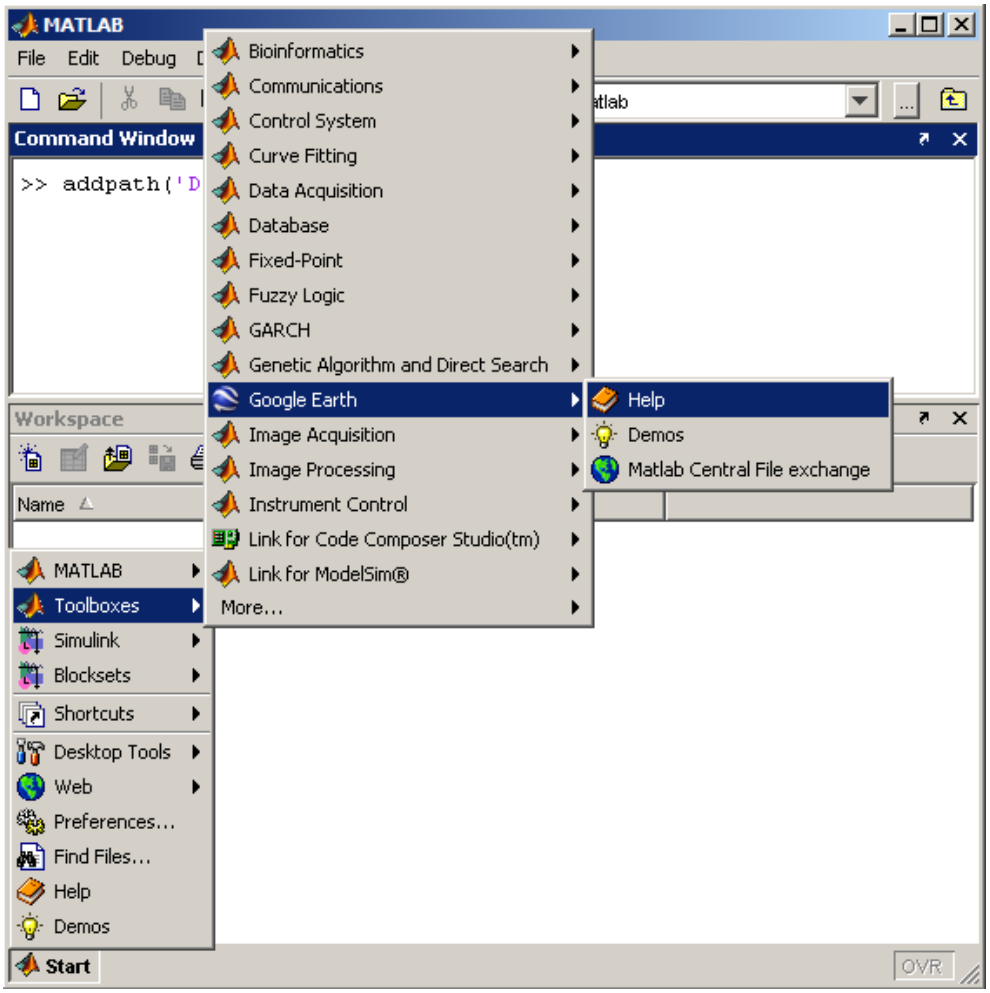
```
>>addpath('D:\matlab\googleearth')
```

This will add the indicated directory to the MATLAB search path (see `doc path`).

2.3 Accessing the documentation

The toolbox has been designed to be as intuitive as possible. For this, the syntax of the functions is very similar to their MATLAB counterparts. In addition to that, a help file is available for all functions within the toolbox. In these files, you can find information on what a particular function does, what variables to pass it, the function's options, links to other functions that perform related tasks, and an example of how the function can be used.

Click on the "Start" button that is located in the lower left corner of the MATLAB application and select "Toolboxes". The Google Earth Toolbox should be present there (see image below).



You can access the documentation by entering the command below at the prompt:

```
>>help googleearth
```

Click on the link '`Google Earth Toolbox Contents`' to be taken to the alphabetical list of functions that together constitute the Google Earth Toolbox.

General remark

Please note that it is strongly recommended to start a new script m-file for each Chapter.

2.4 Drawing points

★ Navigate to the documentation on `ge_point` or enter the command below at the prompt:

```
>>doc ge_point
```

This should invoke the documentation about `ge_point`. Read through it carefully.

★ Now try to place a point at a location of your choice, for example the city that you are in. Assign the result of `ge_point` to a new variable `kmlStr`.

★ Consult the documentation on how to embed the output of `ge_point` in a kml-file using the function `ge_output`.

★ By default, the point is labeled with the full filename of its kml-file when opened in the Google Earth Viewer (see left-hand pane 'Places'). You might find it useful to assign a different name to the object when viewed in Google Earth using `ge_output`'s parameter `'name'`.

★ Instead of the default placemarker icon, use this `'iconURL'` (or any other icon of your choice): `'http://maps.google.com/mapfiles/kml/pal3/icon35.png'`. On Windows machines, you can usually find *.png files at `C:\Program Files\Google\Google Earth\res\` .

2.5 Drawing lines

Drawing lines in Google Earth can be done using the function `ge_plot`.

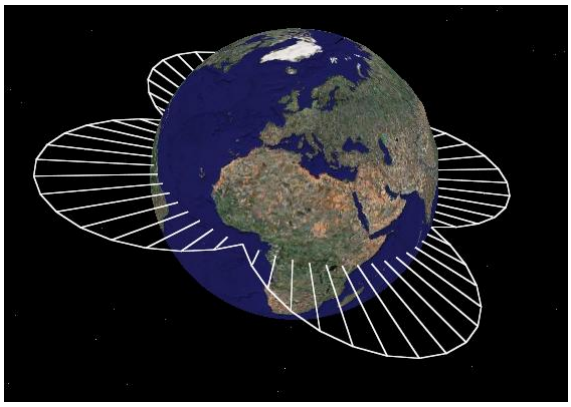
★ Draw a fully opaque, red line over the surface of the Earth connecting Johannesburg, South Africa and Buenos Aires, Argentina. Consult the documentation to find out how to use `ge_plot`, or just try the same approach as you would with MATLAB's `plot` counterpart.

In the previous exercise, altitude is irrelevant since the lince is drawn over the surface of the globe. However, many spatial problems also have an elevation component, e.g. movement of animals, plane and satellite tracks, dispersion of sediment, etc. In analogy to MATLAB's `plot` and `plot3` functions, the Google Earth Toolbox therefore includes a `ge_plot` as well as a `ge_plot3` function.

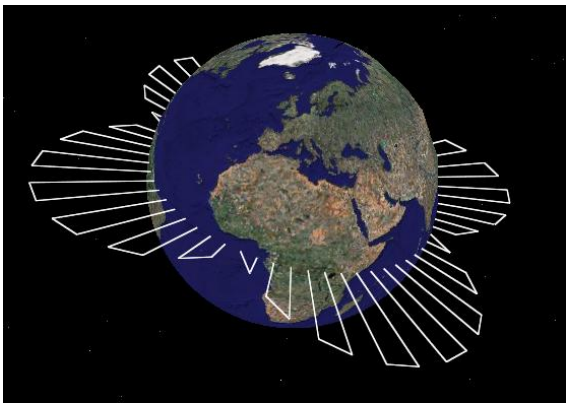
★ Visualize the line defined by `X`, `Y`, and `Z` if

```
Amp = 5e6;                  %amplitude of Z
a = linspace(-pi,pi,61)';   %angle
X = rad2deg(a);             %longitude
Y = zeros(size(a));         %latitude
Z = abs(sin(a*2)*Amp);      %elevation
```

If you did this correctly, the view in Google Earth should be more or less like the image below (shown with the '`extrude`' parameter set to 1 in order to indicate more clearly where line segments start and finish).



Now read the 'Remarks' section in the documentation, and subsequently manipulate your data in order to display the line from the previous exercise as a dashed line (see image below). The dashes should connect point 1 with 2, 2 with 3, and so on.



2.6 Drawing polygons

Now that you know how to draw lines using `plot` and `plot3`, visualization of polygons is easy. A polygon is simply a surface, the edge of which is defined by the coordinates in vectors `X` and `Y` (when using `ge_poly`) or `X`, `Y`, and `Z` (when using `ge_poly3`).

★ A vector map of observed rock outcrops on Antartica is available as `'googleearth/tutres/outcrops.mat'`. Load the data from the file using the command line below:

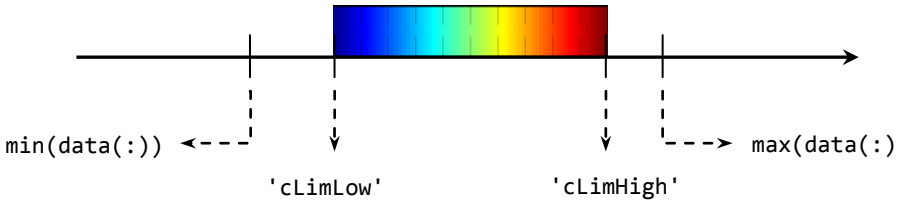
```
>>load(url_from_folder('tutres','outcrops.mat')
```

Consult the documentation to see how `ge_poly` should be used to visualize the data contained in `'outcrops.mat'`. Again use `ge_output` to write a kml file and view the result in Google Earth.

★ Change the default polygon color and polygon outline color to match your taste.

2.7 Drawing raster-based maps

Raster-based maps are basically just a set of square polygons, the color of which is determined according to the schematic below.



The raster data to be visualized using `ge_imagesc` has values that are positioned along the number line (see solid black arrow above). To differentiate between values, grid cells that represent different values of `data` must be assigned different colors. For example, a grid cell having a value equal to the minimum of input variable `data` must be assigned the color from the bottom of the colormap `'cMap'`. By default, `ge_imagesc`'s parameter `'cLimLow'` is set to `min(data(:))` and `'cLimHigh'` is set to `max(data(:))`.

In this exercise, you must visualize a map of measured zinc concentrations in the river Meuse valley near Stein, The Netherlands. An array with spatial predictions of zinc concentrations has already been derived from point measurements using ordinary kriging.

★ Load the zinc concentration data into MATLAB's workspace using:

```
>>load(url_from_folder('tutres','ok-predictions.mat')
```

★ First, use MATLAB's `imagesc` to visualize the data. Include a colorbar.

★ Now consult the documentation on how to visualize raster-based images using `ge_imagesc` and generate a character array with kml tags. The zinc map boundaries are defined by:

North:	50°59'31.20'' N
East:	5°46'08.4'' E
South:	50°57'18.91'' N
West:	5°43'03.36'' E

★ Use `ge_output` to generate a *.kml file and display it in the Google Earth Viewer.

★ Grid cells having a value of -1 have not been included in the spatial prediction during the kriging procedure. Therefore, these should not be visible when the data is displayed in Google Earth. Use one of the optional parameters to accomplish this.

★ Include a colorbar (see `>>doc ge_colorbar`).

★ Set the minimum and maximum color limits to 500 and 1500 ppm, respectively.

★ Set the number of classes in the pop-up of `ge_colorbar` to 20. Assign an appropiate name to the `ge_colorbar` object.

2.8 Collada models and quiver plots

★ Read the documentation on Collada models and `ge_quiver3`.

★ During this exercise, you will draw a `ge_quiver3` object with multiple instances of the same Collada model. The origin `[xo,yo,zo]` of the object is defined as:

```
xo = 0;
yo = 0;
zo = 1e6;
```

...and the locations of the arrows are defined by:

```
n = 16;
a = linspace(0,n*pi,501);
x = xo + 10 * sin(a/n).*sin(a);
y = yo + 10 * sin(a/n).*cos(a);
z = zo + 1e6 * cos(a/n);
```

★ Generate a kml file using `ge_quiver3` when given that the 3D vector components are defined by:

```
u = x-xo;
v = y-yo;
w = (z-zo)/1e5;
```

Set the arrow scaling parameter to `1e5` and make sure that the *.kml file and *.dae file are located in the same directory when displaying the file within the Google Earth Viewer.

2.9 Dynamic visualization

Beyond displaying static objects, the Google Earth Viewer is also a great tool for analyzing time-variant spatial data.

★ Study the documentation on `ge_windbarb`, `ge_quiver3` and 'Dynamic visualization'. Pay special attention to the examples provided therein.

★ The file 'tutres/denhelder.mat' contains a daily record of wind speed and direction for the De Kooy airfield located at 52.921329° N, 4.778582° E. Variable `serDate` contains dates in units of number of days since January 1, 0000 A.D.

★ Load the data into the MATLAB workspace using the command below:

```
>>load(url_from_folder('tutres','denhelder.mat')
```

★ Use `ge_windbarb` to create a kml file with which the entire record of wind speed and direction at the airfield is visualized using conventional wind barb symbols.

3 Ackowledgements

The Google Earth toolbox is supported by the Virtual Laboratory for e-Science project (http://www.vl-e.org).