Frankfurt Beijing

Peter Hertel

University of Osnabrück, Germany

Lecture presented at APS, Nankai University, China

`http://www.home.uni-osnabrueck.de/phertel`

Spring 2012

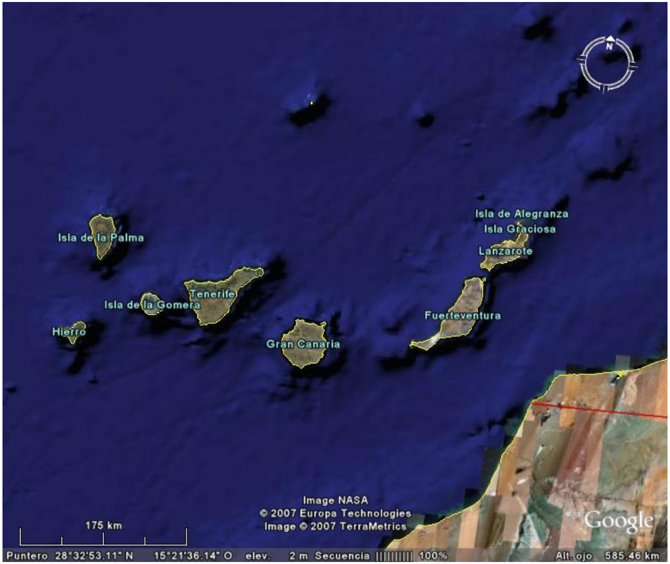
Overview

- MATLAB is tailored to the needs of physicists
- It is a high level computer language
- but it is often misused as an extended calculator
- Structure your program!
- Structure your data!
- Example: calculate distance between Frankfurt and Beijing
- Be friendly to users
- Document your programs *in loco*

Geographical coordinates

- earth's rotational axis z
- equator in x, y plane
- Greenwich in x, z plane
- longitude $-\pi < \phi \le +\pi$
- latitude $-\pi/2 \le \theta \le +\pi/2$
- location vector
$$\boldsymbol{x} = R \begin{pmatrix} \cos\theta\cos\phi \\ \cos\theta\sin\phi \\ \sin\theta \end{pmatrix}$$
- R is earth's radius

Britannia rules the waves: the Royal Observatory defines zero longitude by the location of its telescope.



Before, the Spanish navy was dominant. They defined zero longitude by the westernmost part of their territory, the island of El Hierro.

A script

- assign a good name
- write a help comment
- define constants
- ask user for input
- do the calculation
- show result

```
% calculate shortest distance between locations on globe
% angles in degrees (decimal)
% longitude: positive if east of Greenwich
% latitude : positive if northern hemisphere
globe_radius=6371; % kilometers
from.lat=input('latitude of start        : ')*pi/180;
from.lon=input('longitude of start       : ')*pi/180;
dest.lat=input('latitude of destination  : ')*pi/180;
dest.lon=input('longitude of destination : ')*pi/180;
slat=sin(from.lat);
clat=cos(from.lat);
slon=sin(from.lon);
clon=cos(from.lon);
from.vec=[clat*clon;clat*slon; slat];
slat=sin(dest.lat);
clat=cos(dest.lat);
slon=sin(dest.lon);
clon=cos(dest.lon);
dest.vec=[clat*clon;clat*slon; slat];
angle=acos(from.vec'*dest.vec);
dist=angle*globe_radius;
fprintf(1, 'distance is %i km\n', round(dist));
```

Frankfurt is among the three largest European airports besides London and Paris. Its coordinates are 50.0439 north, 8.5492 west.

Beijing is the second busiest airport worldwide, after Atlanta. Its coordinates are 39.9937 north, 116.4992 west.

```
>> dist_scr
latitude of start          : 50.0439
longitude of start         : 8.5492
latitude of destination    : 39.9937
longitude of destination   : 116.4992
distance is 7790 km
>>
```

Functions instead of scripts

- Scripts leave traces in the workspace
- *you may inspect the variables*
- *these variables may be overwritten by other scripts*
- My advice: begin with a script, test it, and convert it to a function
- function files may contain helper functions
- Here: `d=distance(FRA,PEK)`
- FRA and PEK describe the corresponding airports
- They may be selected from a list of airports
- separate input and calculation!

```
% calculate shortest distance between two locations
% usage: dist_fun(loc1,loc2)
% location [latitude,longitude], angles in degrees
% longitude: positive if east of Greenwich
% latitude : positive if northern hemisphere

function dist=distance(loc1,loc2)
R=6371; % earth's radius in km
loc1=loc1*pi/180;
loc2=loc2*pi/180;
vec1=vector(loc1);
vec2=vector(loc2);
dist=R*acos(vec1'*vec2);
end % dist_fun

function vec=vector(loc)
clat=cos(loc(1));
clon=cos(loc(2));
slat=sin(loc(1));
slon=sin(loc(2));
vec=[clat*slon;clat*clon;slat];
end % vector
```

```
>> FRA=[50.0439,8.5492];
>> PEK=[39.9937,116.4992];
>> d=distance(FRA,PEK)

d =

  7.7904e+003

>> whos
  Name       Size                Bytes  Class

  FRA        1x2                    16  double
  PEK        1x2                    16  double
  d          1x1                     8  double

>>
```

functions do not litter the workspace!

Summary

- structure your program by functions
- a script behaves like a function without input and without output
- ... except that the workspace is littered
- with functions, you may use the same variable name again and again
- e.g. `ndx` for an index or `dummy` for a dummy variable
- note that function name and file name must coincide
- If a task is too difficult, split it into two or more tasks ...
- ... until the task is a function of only a few lines
- *Structured programming !!*