AIR ROUTES

1. INTRODUCTION

The objective of this assignment is to gain experience in the computation and representation of air routes. The concepts of great circle, rhumb lines, and waypoint-defined routes will be worked to a great extent in this exercise. The time calculation and the wind effect on time calculations will be also analyzed.

This assignment requires installing Matlab and Google Earth in your computer. Additionally you will also need the following resources:

- KML library for generating KML files (used by Google Earth) in Matlab. Download the last version from PoliformaT-> Contents -> SOFTWARE section.
- Geo library with formulas for great-circles and rhumb-lines. Download the last version from PoliformaT-> Contents -> SOFTWARE section.
- Flight plan of the route computed by simBrief (http://www.simbrief.com/home/). It is provided in *xml* and *pdf* formats attached to this PoliformaT.
- `xml2struct` Matlab function. It is provided as a zip file attached to this PoliformaT.

2. PROBLEM STATEMENT

The first task of this assignment is to compute and to represent in Google Earth different routes between Madrid airport and Santiago de Chile airport with the following ICAO codes:

ORG= LEMD

DST = SCEL

Exercises

The required routes are:

(1) Great circle route (`great_circle.m`)

(2) Rhumb line passing through ORG and DST (`rhumb_line.m`)

(3) Approximation of the great circle route using rhumb lines (`great_circle_approx.m`)

(4) Waypoint defined route through standard airways provided by simBrief (`airways.m`)

(5) Waypoint defined route through standard airways provided by simBrief in windy conditions (`airways_wind.m`).

Assume that all these routes will be flown at a constant TAS (True Airspeed) of 450 knots (except when otherwise specified). The speed is important for time calculations. The TAS (True Airspeed) and the GS (Ground Speed) are equal with no wind.

Assume also that the cruising altitude is FL380 which will be considered as 38000 ft.

Results

All required routes will be described by

a) A **map** showing the route in Google Earth or Google Maps.

b) A **table** (not necessarily in Excel) specifying: waypoint name, waypoint coordinates (longitude, latitude, altitude in ft), track angle (real, not magnetic), ground speed (GS), cumulative distance in NM to the ORG waypoint and, whenever required, cumulative time *hh:mm:sec* to ORG waypoint.

Waypoint	Longitude	Latitude	Altitude	Track	GS	Distance	Time
LEMD							
...							
...							
LAST WAYPOINT							
SCEL							

c) The Matlab **program** that is been used to generate the file (and optionally the table) will also be required. This program should be self-explicative. Use comments to explain it.

As an example, the below figure shows a cylindrical projection (or planisphere), which is a cylindrical projection of the Earth, showing different routes from Valencia to Sydney:

- **Red**: great-circle route
- **Green**: rhumb-line route obtained using a constant rhumb equal to the initial rhumb.
- **Blue**: rhumb-line route obtained using a constant rhumb which passes through the origin and destination points. Note that this is a straight line in a planisphere.
- **Pink**: approximation of the great circle using rhumb lines and correcting the rhumb every 10º.

It is important to note that Google Earth does not provide cylindrical projections, as the above map. If the distance is so long that it cannot be shown entirely in a Google Earth view, use Google Maps to represent it on a planisphere, or use Google Earth and split the route into several partial views, so the whole route can be viewed.

Remark: Google Maps require an account in Google. Once you get this account, log in, go to "My places" select "Create New Map" and then "Import" a KML file.

Figure 2: Routes from Valencia to Sydney

3. THE LIDO FLIGHT PLAN

A flight plan for this route has been generated using the *simBrief* calculator. This flight plan uses the LIDO format, defined by Lufthansa, and provides much more information that the standard flight plan defined by ICAO.

It is provided in *pdf* format and *xml* format. The *pdf* format is the one used by pilots and contains the information in an easy way to read it. The *xml* format is used to enable automatic processing, for example using Matlab. Both files can be found for the proposed route attached to this PoliformaT:

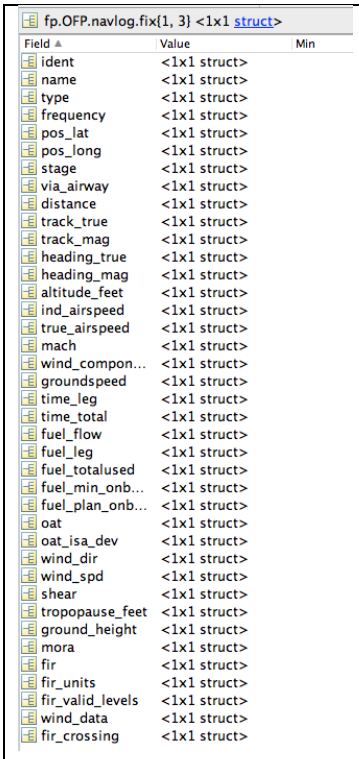
`LEMDSCEL_PDF_1474620942.pdf`

`LEMDSCEL_PDF_1474620942.xml`

The flight plan in *xml* format can be converted into a Matlab struct containing all the information. This can be done with Matlab function **`xml2struct`** contained in a zip file attached to this PoliformaT:

`xml2struct.zip`

Use this function, and display the different fields of the generated Matlab strtuct. The fields in **`OFP.navlog`** are especially interesting for solving this exercise. They are shown in the below figure **`OFP.navlog.fix{1,i}`** corresponds to the waypoints of first route alternative. Alternative routes can be specified in the flight plan for contingencies, but they have not be defined in this flight plan.



Waypoint names and coordinates:

```
OFP.navlog.fix{1, i}.ident

OFP.navlog.fix{1, i}.name

OFP.navlog.fix{1, i}.pos_lat

OFP.navlog.fix{1, i}.pos_long

OFP.navlog.fix{1, i}.alt_feet
```

Wind forecast at each waypoint:

```
OFP.navlog.fix{1, i}.wind_dir

OFP.navlog.fix{1, i}.wind_spd
```

4. EXERCISE 1: GREAT CIRCLE

Google Earth always represents the great line circle between two points, so this exercise is straightforward: draw a line in Google Earth between ORG and DST. Compute also the distance and initial and final courses.

5. EXERCISE 2: RHUMB LINE

Compute the loxodromic course *Phi* between ORG and DST. This can be done using the proper formula of the **`lib/geo`** library provided in the SOFTWARE section of the PoliformaT.

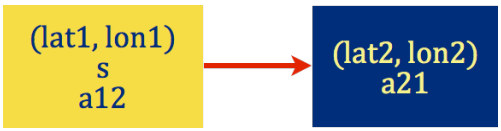
Compute also the loxodromic distance *D* between ORG and DST.

The route will be computed in an iterative way, generating a leg in each iteration. Let *N* be the number of iterations/legs and let *d=D/N*.

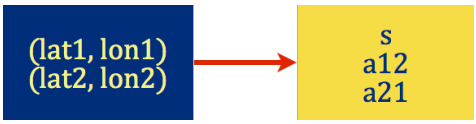
The first waypoint WP1 will be obtained by navigating a distance *d* from ORG with a constant rhumb *Phi*. Similarly, the second waypoint WP2 will be obtained by navigating a distance *d* with the same rhumb *Phi* from WP1.

It is a matter of using the right formulas of the **`lib/geo`** library to compute the right courses and distances:

Direct problem:



Inverse problem



- **Matlab library lib/geo**
  - Direct problem

```
- Spherical  orthodromic (great circle)
  function [lat2,lon2,a21] = orto_reckon(lat1,lon1,s,a1)
- Spherical  loxodromic (rhumb lines)
  function [lat2,lon2,a21] = loxo_reckon(lat1,lon1,s,a1)
- Ellipsoidal  orthodromic (great "circle")
  function [lat2,lon2,a21] = vincenty_reckon(lat1,lon1,s,a1)
```

  - Inverse problem

```
- Spherical  orthodromic (great circle)
  function [s,a12,a21] = orto_distazi(lat1,lon1,lat2,lon2)
- Spherical  loxodromic (rhumb lines)
  function [s,a12,a21] = loxo_distazi(lat1,lon1,lat2,lon2)
- Ellipsoidal  orthodromic (great "circle")
  function [s,a12,a21] = vincenty_distazi(lat1,lon1,lat2,lon2)
```

6. EXERCISE 3: APPROXIMATING THE GREAT CIRCLE USING RHUMB LINES

The suggested approximation works as follows:

1. Compute the orthodromic course *Phi* between ORG and DST.
2. Compute also the orthodromic distance *D* between ORG and DST.
3. Let *N* be the number of iterations and let *d=D/N*.
4. The first waypoint WP1 will be obtained by reckoning: navigate a distance *d* from ORG with constant course *Phi*.
5. Recompute the course *Phi* as the orthodromic course from WP1 to DST
6. The second waypoint WP2 will be obtained by navigating a distance *d* with the new course *Phi*.

7. Repeat the process for all waypoints.

Alternative proposals that improve the distance and can be navigated using a compass (loxodromically) can also be used. Explain the new proposal, if it is used.

Errors and number of iterations

The final waypoint computed with this approximation will not usually match the DST waypoint, so there will be an error that you should clearly specify. The maximum allowed error will be, in principle, 50 NM. If you cannot accomplish this value, then specify your error and the reason it could not be accomplished.

The error mostly depends of the number of iterations. It also depends on the chosen distance for each leg of the iteration. Take into account that the distance of the approximation is usually longer than the big circle. You can make some variations in the number of legs and distance of each leg in order to get the specified error in the minimum number of iterations.

Explaining the criterion for choosing the number of legs and the leg's distance in order to minimize the error is an important part of the assignment.

7. EXERCISE 4: WAYPOINT DEFINED ROUTE THROUGH STANDARD AIRWAYS

The route for Matlab processing will be obtained from the flight plan provided by simBrief in *xml* format. Start by reading the xml file into a Matlab variable which is a struct:

```
fp=xml2struct('LEMDSCEL_XML_1474620942');
```

Generate the information required by Google Earth in a `wp` struct:

```
wp(i).name  = fp.OFP.navlog.fix{1, i}.id
wp(i).lat   = fp.OFP.navlog.fix{1, i}.pos_lat
wp(i).lon   = fp.OFP.navlog.fix{1, i}.pos_long
wp(i).alt   = fp.OFP.navlog.fix{1, i}.alt_feet
wp(i).desc  = fp.OFP.navlog.fix{1, i}.name
              % or any information you like…
```

Representing the route in Google Earth with this information is straightforward.

For time calculations assume that the route will be flown at a constant TAS (True Airspeed) of 450 knots and the cruising altitude is 38000 ft.

8. EXERCISE 4: WAYPOINT DEFINED ROUTE IN WINDY CONDITIONS

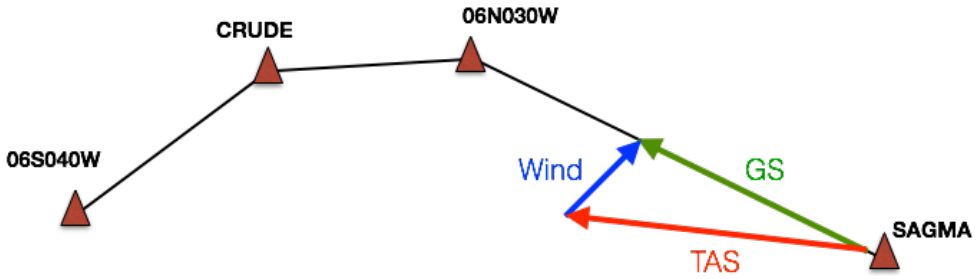
The only difference with the previous routes is how the time calculations are performed. The wind forecast at each waypoint is provided by the simBrief flight plan:

```
fp.OFP.navlog.fix{1, i}.wind_dir

fp.OFP.navlog.fix{1, i}.wind_spd
```

This afects the plane speed and thus, the flight times. You need to use now the Ground Speed (GS) which does no longer matches the True Airspeed (TAS).

For computing the GS at each waypoint asume that the TAS is 450 knots and the autopilot performs the necessary heading corrections, so the GS direction is always alligned with the route, as shown in the wind triangle of the figure.



Compute the GS at each waypoint by adding the TAS the tailwind or component of the wind or subtracting the frontwind component.

Calculate the new times estimates assuming that the GS remains constant along each leg of the route.

Making the time estimates more accurate needs to take into account the climb and descent phases. The climb phase lasts until the cruising altitude is reached at the TOC (Top Of Climb) and the descent phase is initiated at the TOD (Top of Descent), which is the point where the descent should start to reach the airport with a given vertical speed. The TOC and TOD will be computed assuming that the TAS during these phases is TAS=250 knots and the Vertical Speed V/S=750 ft/min.