Lake Routing - Navigation

In[85]:=
```
SetDirectory["~/ZHAW"]
FileNames[] // TableForm
```

Out[85]= /Users/jacques/ZHAW

Out[86]//TableForm=
```
Bodensee_OP.pdf
Bodensee_Polaire.pdf
Décision_Sphere_Commented.nb
Décision_Sphere_Commented.pdf
Dessin1.jpg
Dessin2.jpg
.DS_Store
Interpolation_commented.nb
LakRouting_ZHAW.pdf
Navigation_Commented_New.nb
Navigation_Commented_New.pdf
```

Coordinate system

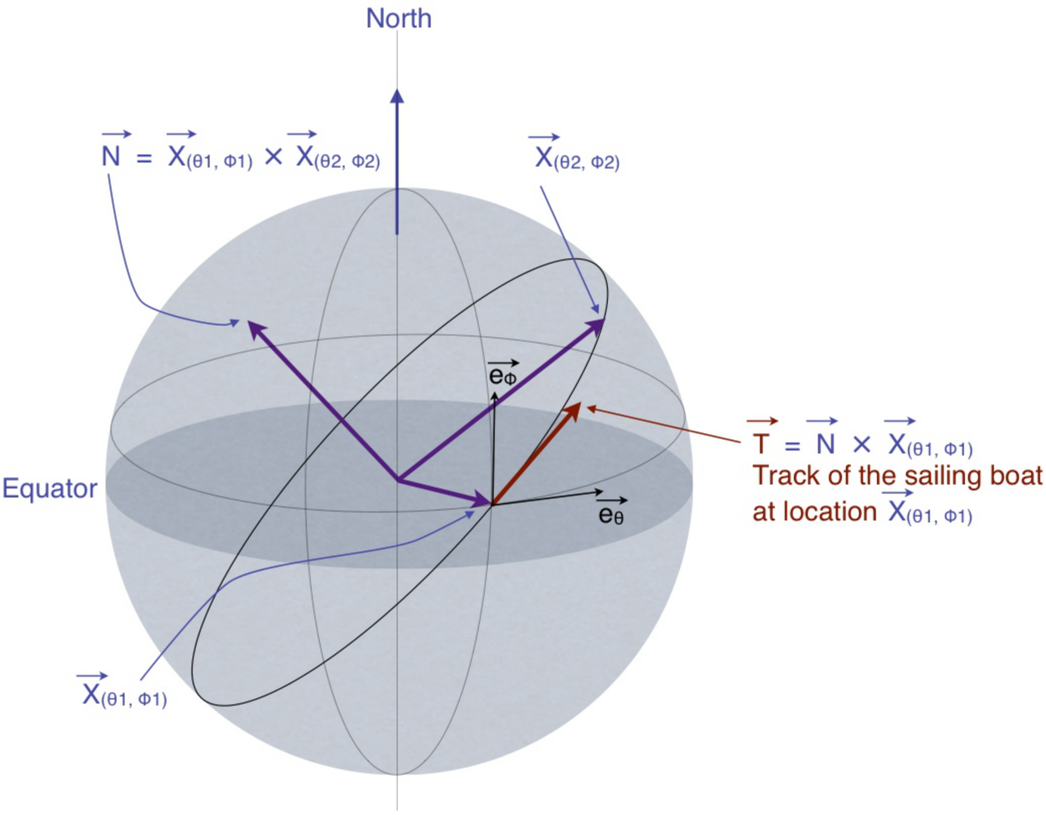
The earth is represented as the S_2 unit sphere immerged in $\mathbb{R}^3$. Coordinates are $\{\theta, \phi\}$, where $\theta \in [0, 2\pi)$ spans longitudes eastwardly and $\phi \in \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$ spans latitudes from south to north pole. The immersion follows the standard application:

$$
\begin{array}{ccc}
[0, 2\pi) \lozenge \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right] & \longrightarrow & S_2 \subset \mathbb{R}^3 \\
\{\theta, \phi\} & \longrightarrow & X_{(\theta,\phi)} = \begin{pmatrix} \cos(\theta)\cos(\phi) \\ \sin(\theta)\cos(\phi) \\ \sin(\phi) \end{pmatrix}
\end{array}
$$

with $X_{(\theta,\phi)}$ expressed in *Mathematica* as:

In[87]:=
```
X[θ_, ϕ_] := {Cos[θ] Cos[ϕ], Sin[θ] Cos[ϕ], Sin[ϕ]};
```

In[88]:=
```
Figure1 = Import["Dessin2.jpg"]
```

Out[88]=


Orthodromy

An orthodromy is a shortest route linking two locations on a geometrical surface, according to the intrisic metric of that surface. Orthodromies are usually computed following a extremal principle delivering a second order differential equation, whose solution is the orthodromy. On the sphere, othodromies are constructed as follow: consider two locations $\{\Theta_1, \phi_1\}$ and $\{\Theta_2, \phi_2\}$ on the sphere. Together with the center of the sphere, $X_C = \{0, 0, 0\}$, the points $X_{(\Theta_1,\phi_1)}$ and $X_{(\Theta_2,\phi_2)}$ define a plane whose intersection with the sphere is the circle drawn on the sphere, indeed the orthodromy. Of course, starting from $\{\Theta_1, \phi_1\}$, $\{\Theta_2, \phi_2\}$ can be reached by travelling on the circle into two, shorter or longer, directions.

Historically, orthodromies on earth are called “arcs of great circle”, whereas the arc is indeed the short path.

Making use of the scalar product between the two vectors X1 and X2 on the sphere:

$$X1 \cdot X2 = ||X1||\,||X2||\ \mathrm{Cos}[\sphericalangle(X1, X2)],$$

the distance on the sphere between two locations $\{\Theta_1, \phi_1\}$ and $\{\Theta_2, \phi_2\}$ is provided by the module **Ortho** computing $\mathrm{ArcCos}[X_{(\Theta_1,\phi_1)} . X_{(\Theta_2,\phi_2)}]$ (remember that the radius of our sphere is unity). Making use of // FullSimplify, it delivers the distance expressed in nautical miles:

In[89]:=
```
{X1, X2} = {X[θ1, ϕ1], X[θ2, ϕ2]};

Orthodromy = X1.X2;
Orform = % // FullSimplify
```

Out[91]= Cos[θ1 - θ2] Cos[ϕ1] Cos[ϕ2] + Sin[ϕ1] Sin[ϕ2]

In[92]:=
```
Ortho[{x1_, y1_}, {x2_, y2_}] := Module[{θ1, ϕ1, θ2, ϕ2},
   {θ1, ϕ1, θ2, ϕ2} = π/180 {x1, y1, x2, y2};
   N[(360 60)/(2 π) ArcCos[Cos[θ1 - θ2] Cos[ϕ1] Cos[ϕ2] + Sin[ϕ1] Sin[ϕ2]]]
  ];
```

Why nautical miles?

On earth, one Nautical Mile is the distance separating two locations $\{\theta_1, \phi_1\}$ and $\{\theta_2, \phi_2\}$ making with the centre of earth X_c an angle $\sphericalangle (X_{(\theta_1,\phi_1)}, X_c, X_{(\theta_2,\phi_2)})$ that equals to one minute of degree.

This explains the coefficient $\frac{360 \times 60}{2\pi}$ in the expression of **Ortho**. By that way, our computations are performed without any reference to the actual radius of earth, or of any planet on which a regatta would be organized. Furthemore, computing speed in knots - nautical miles per hour - makes us only dependent on the angular rotation of the planet, one full rotation of 2 π in 24 hours. This definition justifies the way the module **Ortho** is concieved. This system is valid on any other planet or spherical body. Expressed in our usual unit system, the values of the nautical mile and the knot would simply be different. And of course, I almost forgot it, on earth 1 Nautical Mile = 1.852 km, 1 Knot = 0.514 m/s.

Local tangent frame

On the sphere, the local tangent frame located at $\{\theta, \phi\}$ is the plane tangent to the sphere at that location. It can be given the structure of a 2-dimensional vector space (on $\mathbb{R}$) with basis $\{e_\theta, e_\phi\}$. e_θ and e_ϕ are the vectors tangent to the sphere al location $\{\theta, \phi\}$ and tangent to the coordinates lines defined by θ and ϕ. Pictured as the two thin black vectors on previons figure, they are provided by:

$$\{e_\theta, e_\phi\} = \left\{\frac{1}{\mathrm{Cos}(\phi)} \partial_\theta X_{(\theta,\phi)}, \partial_\phi X_{(\theta,\phi)}\right\}$$

The division by Cos(ϕ) in the first term makes the frame orthonormal: $|e_\theta| = |e_\phi| = 1$ at each location $\{\theta, \phi\}$ of the sphere, with exception of both poles. Computed hereafter at location $\{\theta_1, \phi_1\}$, the local tangent frame will be used later for the calculation of the course angle from location $\{\theta_1, \phi_1\}$ to location $\{\theta_2, \phi_2\}$.

In[93]:=
```
{eθ1, eϕ1} = {∂θ1 X[θ1, ϕ1] / Cos[ϕ1], ∂ϕ1 X[θ1, ϕ1]};
```

Computation of the Track (German: Kurs)

Everything is performed in three spatial dimensions, as pictured on figure 1. Firstly, the cross product of the two vectors $X_{(\theta_1,\phi_1)}$ and $X_{(\theta_2,\phi_2)}$ delivers the vector $N = X_{(\theta_1,\phi_1)} \Diamond X_{(\theta_2,\phi_2)}$ normal to the plan containing the great circle. A further cross product delivers the Track at location $\{\theta_1, \phi_1\}$: $T_{(\theta_1,\phi_1)} = N \times X_{(\theta_1,\phi_1)}$. Both cross products are combined in the following expression, making N implicit:

In[94]:=
```
TrackVect1 = Cross[Cross[X1, X2], X1] // FullSimplify;
TV1 = % // MatrixForm
```

Out[95]//MatrixForm=
```
( Cos[ϕ1]^2 Cos[ϕ2] Sin[θ1] Sin[θ1 - θ2] + Cos[θ2] Cos[ϕ2] Sin[ϕ1]^2 - Cos[θ1] Cos[ϕ1] Sin[ϕ1] Si
 -Cos[θ1] Cos[ϕ1]^2 Cos[ϕ2] Sin[θ1 - θ2] + Sin[ϕ1] (Cos[ϕ2] Sin[θ2] Sin[ϕ1] - Cos[ϕ1] Sin[θ1] Si
              Cos[ϕ1] (-Cos[θ1 - θ2] Cos[ϕ2] Sin[ϕ1] + Cos[ϕ1] Sin[ϕ2]) )
```

Finally, the Track vector at location $\{\theta_1, \phi_1\}$ is projected onto the local tangent basis $\{e_\theta, e_\phi\}$ at that location: $\{e_\theta . T_{(\theta_1,\phi_1)}, e_\phi . T_{(\theta_1,\phi_1)}\}$. The two numbers thus obtained are the coordnates of the Track vector in this basis:

In[96]:=
```
{TrackLong1, TrackLat1} = {eθ1. TrackVect1, eϕ1 . TrackVect1} // FullSimplify;
TP1 = % // MatrixForm
```

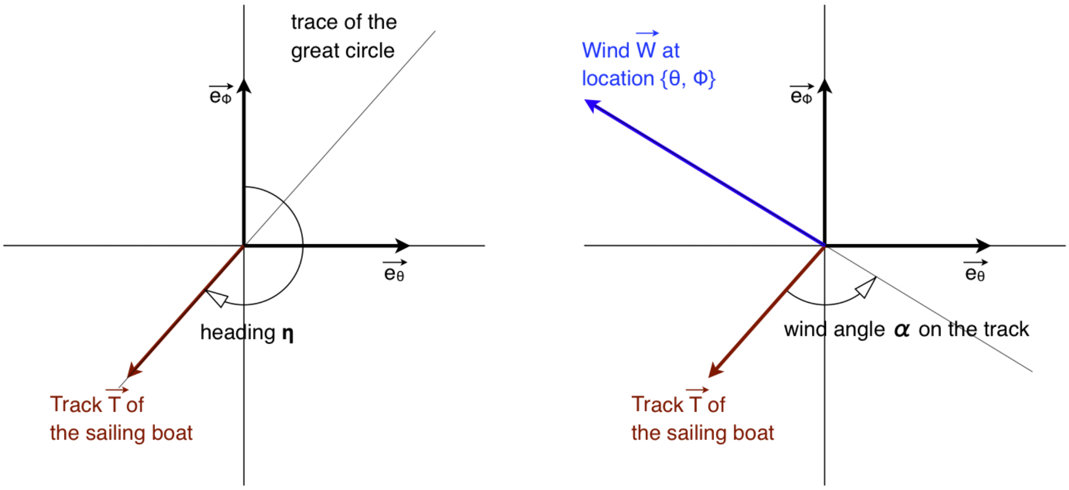
Out[97]//MatrixForm=
```
(            -Cos[ϕ2] Sin[θ1 - θ2]
 (-Cos[θ1 - θ2] Cos[ϕ2] Sin[ϕ1] + Cos[ϕ1] Sin[ϕ2]) )
```

Computation of the Track and the heading, Module Track12

Making use of the previous considerations, following Module **Track12** computes the heading η at location $\{\Theta_1, \phi_1\}$ of a mobile sailing from there to location $\{\Theta_2, \phi_2\}$. The Track vector at location $\{\Theta_1, \phi_1\}$ is firstly projected onto the local tangent basis $\{e_\Theta, e_\phi\}$, then a classical Arctan computation delivers the heading. Internal module **Arctangente**, carefully designed in order to catch all possible configurations, performs this computation. The last instruction of Track12 transforms the mathematical angle given on the trigonometric circle into the heading used in navigation with 90° towards east, 180° towards south, 270° towards west and 360° or 0° towards north.

In[98]:=
```
Figure2 = Import["Dessin1.jpg"]
```

Out[98]=



In[99]:=
```
Track12[{x1_, y1_}, {x2_, y2_}] :=
  Module[{ArcTangente, TrackLong1, TrackLat1, NTrack1, θ1, ϕ1, θ2, ϕ2},

   ArcTangente[x_, y_] := N[
     If[(x > 0) && (y > 0), ArcTan[y/x],
      If[(x < 0) && (y > 0), π - ArcTan[y/Abs[x]],
       If[(x < 0) && (y < 0), π + ArcTan[y/x],
        If[(x > 0) && (y < 0), 2 π - ArcTan[Abs[y]/x],
         If[(x == 0) && (y > 0), π/2,
          If[(x == 0) && (y < 0), 3 π/2,
           If[(x < 0) && (y == 0), π,
            If[(x > 0) && (y == 0), 0
             ]]]]]]]]];

   {θ1, ϕ1, θ2, ϕ2} = π/180 {x1, y1, x2, y2};
   {TrackLong1, TrackLat1} =
    N[{- Cos[ϕ2] Sin[θ1 - θ2], Cos[ϕ1] Sin[ϕ2] - Sin[ϕ1] Cos[ϕ2] Cos[θ1 - θ2]}];
   NTrack1 = 180/π ArcTangente[TrackLong1, TrackLat1];
   If[NTrack1 > 90.0, 450.0 - NTrack1, 90.0 - NTrack1]
  ]; (* Track12 *)
```

Example: computation of distance and heading between two locations:

In[100]:=
```
Departure = {8, 47};
Arrival = {7, 46};
Print[
  "Orthodromic Distance between Departure and Arrival expressed in Nautical Miles: ",
  Ortho[Departure, Arrival]];
Print["Heading in degree between these two locations: ", Track12[Departure, Arrival]];
```

```
Orthodromic Distance between Departure and Arrival expressed in Nautical Miles:
 72.8394

Heading in degree between these two locations: 214.905
```

Computation of the speed of the sailing boat on a given track

Let $W_{(t)} = \{u, v\}_{(t)}$ the wind blowing at location $\{\theta, \phi\}$ when the sailing boat crosses it at time t. (u and v are called zonal, respectively meridional components of the wind vector. The former is directed west- eastwards, along a latitude circle, the latter south- northwards, along a longitude circle). The speed of the sailing boat firstly depends on the intensity of the wind $|W_{(t)}|$, secondly on the angle of incidence α of the wind vector onto the sails represented as $\boldsymbol{\alpha}$ on the right panel of figure 2. Technically, the speed of the sailing boat is available as a polar diagramme delivering for each couple $\{|W_{(t)}|, \alpha\}$ the speed $\boldsymbol{\rho}$ expressed in knots (KT)

Practically, only a few couples of $\{|W|, \alpha\}$ values being known, an interpolation process has to be implemented in order to enable the computation of an good estimation of the sailing boat performance for each wind intensity and attack angle. It is encapsulated in the Module **Yacht** proposed below. This interpolation is presented in section xxx. Formally, the incidence angle α is computed with a scalar product: $\alpha = 180\left(1 - \frac{1}{\pi}\text{ArcCos}\left[\frac{T.W}{|T|\,|W|}\right]\right)$.

In[104]:=
```
Yacht[W_, A_] := Module[{MinA, MinW, MaxW, InterpolationVoilier, MinimumSpeed},
    {MinA, MinW, MaxW, MinimumSpeed} = {30, 0.3, 30, 0.01};
   If[(A < MinA) || (W < MinW) , MinimumSpeed,
    If[(W > MaxW) , InterpolationVoilier[MaxW, A],
     InterpolationVoilier[W, A]]]];
```

In[105]:=
```
SpeadBoat[{x1_, y1_}, {x2_, y2_}, {u_, v_}] :=
  Module[{W, θ1, ϕ1, θ2, ϕ2, TrackLong1, TrackLat1, T, α},
   W = Sqrt[u^2 + v^2];
   {θ1, ϕ1, θ2, ϕ2} = π/180 {x1, y1, x2, y2};
   {TrackLong1, TrackLat1} =
    N[{- Cos[ϕ2] Sin[θ1 - θ2], Cos[ϕ1] Sin[ϕ2] - Sin[ϕ1] Cos[ϕ2] Cos[θ1 - θ2]}];
   T = Sqrt[TrackLong1^2 + TrackLat1^2];
   α = 180 (1 - 1/Pi ArcCos[1/(T W) {TrackLong1, TrackLat1}.{u, v}]);
   Yacht[W, α]];
```

Remarks:

1. The Module **Yacht** given hereafter is incomplete. MinA, MinW, MaxW have been set at arbitrary values. They have to be provided as information related to the polar diagramme by the procedure **InterpolationVoilier**. The tests implemented in **Yacht** on MinA, MinW, MaxW provide minumum speed for the sailing boat if the incidence angle is lower than 30°, or if the wind is weaker than 0.3 KT, and consider the maximum wind speed in the table if the true wind is stronger than that value.

2. arcs being very short for lake routing, track vectors and headings remain unchanged at departure and arrival nodes. Therefore, the same procedure can be used to compute the speed of the sailing boat at both nodes, of course with the related winds. This property is applied hereafter for the computation of the sailing duration betwen two nodes.

Sailing Duration

The time elapsed on the track separating two nodes located at {θ1, ϕ1} and {θ2, ϕ2} is given by the following procedure. Winds {ui, vi}, i = 1, 2, have to provided for the corresponding node at the time at which the sailing boat crosses this node. Expecting that this condition will be satisfied, the sailing duration is simply given by the length of the track divided by the average speed on the it:

In[106]:=
```
SailingDuration[{x1_, y1_}, {x2_, y2_}, {u1_, v1_}, {u2_, v2_}] :=
   2 Ortho[{x1, y1}, {x2, y2}] /
   (SpeadBoat[{x1, y1}, {x2, y2}, {u1, v1}] + SpeadBoat[{x1, y1}, {x2, y2}, {u2, v2}]);
```