

CARTOGRAPHIE et LOXODROMIE

Par Nulentout le Lundi 4 Février 2025.

A partir du moment où l'on désire calculer la distance de deux entités très éloignées sur Terre au point d'avoir besoin de tenir compte de sa rotondité, nous n'échappons pas aux problèmes liés à la cartographie et à l'étude des coordonnées sphériques pour en désigner la position à la surface du globe. Nous allons passer en revue des concepts incontournables *en simplifiant au maximum*, que les spécialistes veuillent bien accepter cette vulgarisation poussée à l'extrême. *Toute la complexité de ce domaine réside dans le fait que la Terre est sphérique, et j'envie au passage celle et ceux qui sont persuadés qu'elle est plate. Les calculs de trigonométrie en seraient très simplifiés, par contre, le mouvement des satellites en serait totalement chantourlouboulés !*

1) Les coordonnées sphériques.

Vous savez toutes et tous, que pour situer un point sur notre bonne vieille Terre, on a imaginé une kyrielle de cercles théoriques qui "quadrillent" le globe et que l'on nomme les méridiens et les parallèles. Comme ce sont nos voisins Anglais qui se sont les premiers attelés à ce problème, ils ont tout naturellement pris pour origine du méridien ZÉRO, celui qui passe par la célèbre ville de Greenwich. Les méridiens sont comptés en positif vers l'Est et négatif vers l'Ouest. Une convention analogue s'impose pour les parallèles, comptés positivement vers le Nord à partir de l'Équateur et négativement vers le sud. Google Maps respecte ces conventions que la Fig.1 résume. Toujours par convention, les coordonnées d'un point sur le globe sont définies par rapport au *méridien de Greenwich* et l'*Équateur* et sont nommée *Longitude* et *Latitude*. Pour illustrer ce propos la Fig.1 propose quatre points **A**, **B**, **C** et **D** et le tableau de la Fig.2 en précise les détails des coordonnées sphériques de ces points.

	Latitude	Longitude
A	+28° ou 28°N	+32° ou 32°E
B	+35° ou 35°N	-22° ou 22°W
C	-30° ou 30°S	-18° ou 18°W
D	-32° ou 32°S	+25° ou 25°E

Fig.2

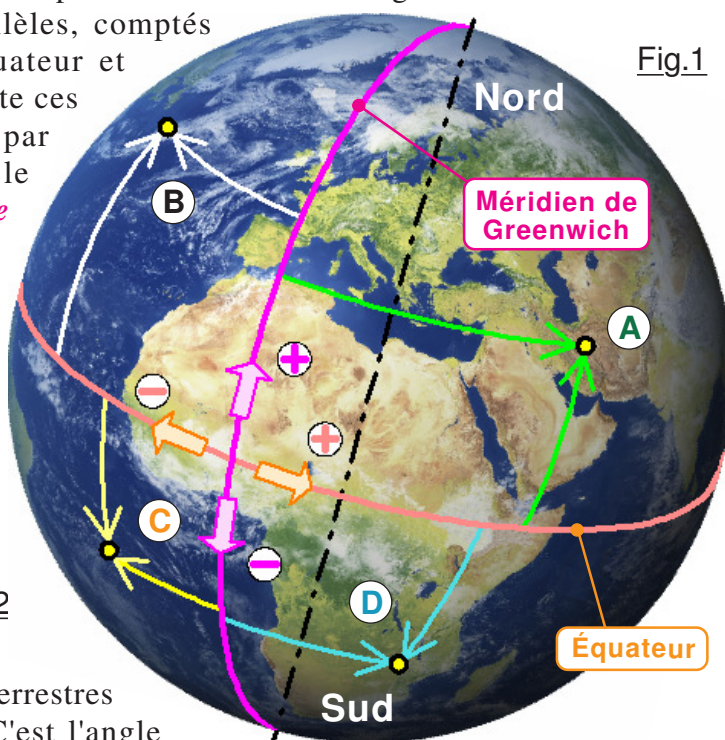


Fig.1

Remarque : On représente souvent les globes terrestres avec leurs axes Nord/Sud inclinés de 22,5°. C'est l'angle que fait l'axe de rotation de la Terre avec le plan dans lequel circule notre planète autour du soleil.

➤ Les unités de positionnement géographique.

Comme c'est le cas pour l'heure, on va utiliser des décimales qui peuvent s'exprimer en minutes et en secondes. Un *degré angulaire* contient 60 *minutes d'angle* et une minute d'angle 60 *secondes d'angle*. On peut donc utiliser plusieurs formes d'écritures toutes équivalentes. Par exemple pour la longitude de PARIS on peut écrire :

Longitude = +2.34600 (Entièrement en degrés angulaires : Google Maps.)

Longitude = 002,20.76E (En degré et en minutes : Le module GPS en norme NMEA 0183 .)

Longitude = 002° 20' 45" E (En degré, minutes et secondes : Notre programme.)

2) Latitudes, et distances sur Terre.

➤ Le système décimal pollué par la base 12 !

C'est la civilisation Babylonienne qui en utilisant les phalanges des doigts d'une seule main pour compter avec le pouce qui a naturellement utilisé la base douze. C'est la raison pour laquelle le nombre 12 avec ses multiples et ses sous-multiples est si présent dans un système actuel qui se prétend en base 10. Par exemple ils ont divisé l'année en 12 mois, vendent les œufs par six ou par 12. Il y a peu, 12 douzaines se nommait une "grosse". Quand il a fallu diviser la journée en parties égales, ils ont prévu 12 divisions quand il fait jour et 12 divisions quand il fait nuit. Puis, l'heure étant bien trop grande, il aurait été logique de diviser les heures en 12 minutes. Mais ce laps de temps n'étant pas assez fin pour les applications de l'époque, ils ont divisé en 5×12 d'où les 60 minutes. Enfin le problème s'est reposé lorsque la technique désirait des intervalles de temps encore plus petits. Du coup on a divisé encore les minutes horaires par soixante. Et au final, en dessous, on en revient au système décimal ... car leur technologie se contentant de la seconde, c'est bien plus tard que le besoin se faisant sentir le système décimal a repris ses droits. On peut être certain que si les humains à cette époque avaient eu des mains à quatre doigts, la base 9 se serait imposée, et avec six doigts c'est le 15 qui aurait dominé. Bref, *nous allons donc traiter nos calculs mathématiques avec un système décimal qui se mélange intimement avec les multiples de la base douze.*

➤ La minute d'angle pour mesurer les distances sur terre !

Dans ce chapitre nous allons établir une relation directe entre la minute d'angle qui permet d'avoir des sous-multiples précis sur les cartes du monde et les distances que par habitude nous exprimons en kilomètres dans le système international basé sur le système décimal.

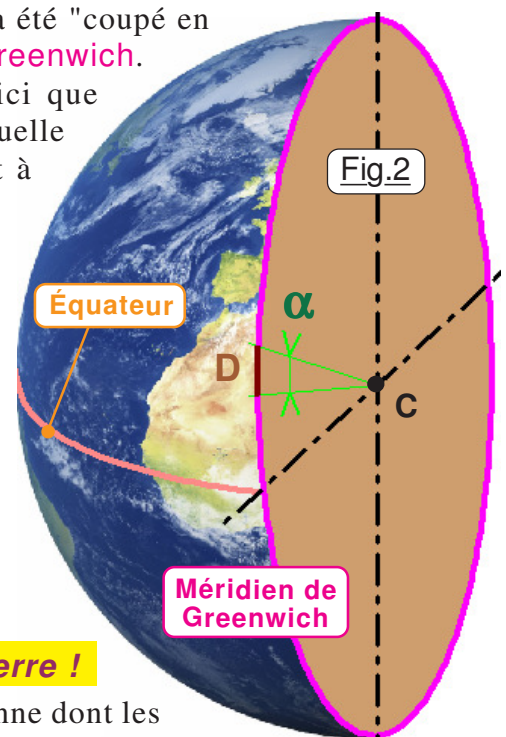
Par définition, un *mile Nautique* ou *mille marin* correspond à la distance sur Terre mesurée sur un écart angulaire d'une *minute d'angle sur un méridien*.

Considérons la Fig.2 sur laquelle le dessin du globe terrestre a été "coupé en deux". On y retrouve l'Équateur terrestre et le méridien de Greenwich.

En C se trouve le Centre de la Terre et l'on va supposer ici que l'ouverture de l'angle α fait exactement une minute d'angle. Quelle est alors la distance D à la surface de la Terre correspondant à cette ouverture angulaire ?

La réponse est facile. En simplifiant, le rayon terrestre fait en moyenne 6367km. La distance totale mesurée sur un cercle de type méridien est donc de $6367 \times 2 \times \text{PI}$ soit environ 40005km. Sur 360° angulaires nous avons $360 \times 60 = 21600$ minutes d'angle. Donc, la distance D correspondant à une minute d'angle sur un méridien est égale à : $D = 40005 / 21600 = 1,852\text{km}$.

Une minute d'angle sur un méridien correspond à un *mile Nautique* ou un *mille marin* soit environ 1852m.

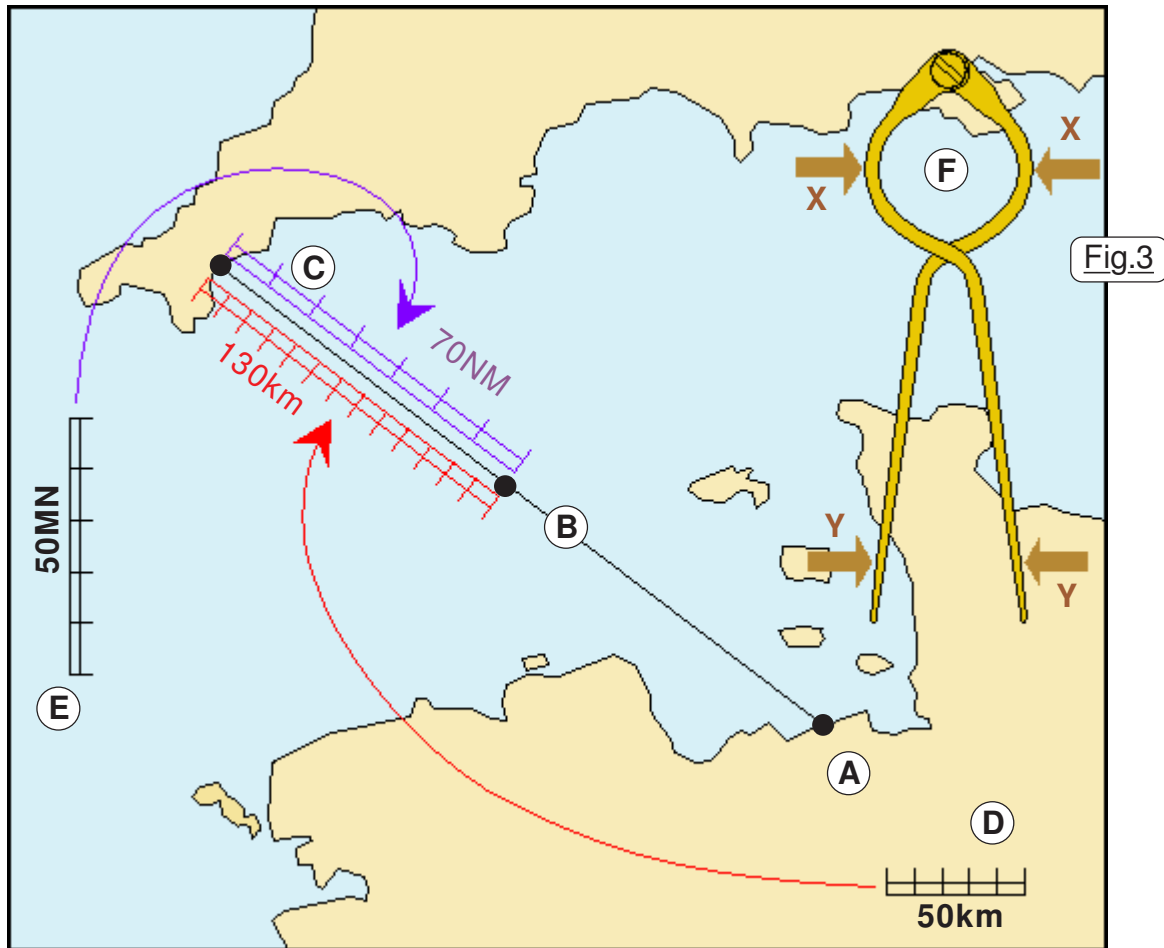


➤ Le mile Nautique pour mesurer les distances sur terre !

Bien qu'influencé directement par la civilisation Babylonienne dont les personnes avaient cinq doigts à chaque main et par le rayon de la Terre qui ne dépend que de l'évolution du système solaire, c'est de loin le système le plus commode jamais imaginé par les voyageurs au long cours pour simplifier considérablement le travail du navigateur. Si l'on utilise le *mille Nautique* pour définir des distances sur la surface de la Terre, alors une vitesse d'un Nautique par heure porte le nom de *Nœud marin* souvent symbolisé par **Kn**. Cette technique est tellement intelligente, que ces unités un peu ésotériques pour le commun des mortels sont encore bien présentes en aviation y compris dans les avions de ligne les plus modernes.

En quoi le *mille marin* est-il si commode pour faciliter les calculs du navigateur ?

Naviguer c'est savoir en permanence où se trouve le navire ou l'avion (*Ici on oublie le vol spatial.*) C'est également savoir combien de temps on va mettre pour arriver à destination et à quelle heure locale. Pour les marins c'est primordial car ils sont tributaires de la "hauteur de la marée". En aviation c'est tout aussi important, car en traversant l'Atlantique il faut savoir quelle sera la météorologie sur la destination envisagée. Pour comprendre facilement toute l'intelligence contenue dans le système des unités "marines", nous allons nous exercer sur un petit exemple simplifié à l'extrême et je vous autorise à utiliser une calculette car on n'est pas là pour se prendre la tête.



Vous incarnez le navigateur qui à bord du navire de commerce le "fringuant" vous naviguez toutes voiles dehors par vent de travers. La mer est belle et une petite brise vous pousse à 8km/H. Parti de Saint-Malo en **A** vous vous dirigez en **C** à Falmouth pour livrer du charbon. Il est 12h et votre dernier point vous situe en **B**. En supposant pour simplifier que la vitesse va rester constante et qu'il n'y a pas de courants dans la Manche, calculez votre heure d'arrivée. (*Quand je vous dis que je simplifie exagérément, imaginer que dans le "chanel" l'eau est stagnante, c'est un scandale !*)

Alors, vous avez trouvé ?

Le navigateur de l'époque n'a pas de calculette à son service, mais il n'en a pas besoin, car il n'a aucun calcul à faire. Pour lui, la vitesse sur l'eau mesurée n'est pas de 8km/h, mais son "poison" est gradué en nœuds. Le navire file **4,3Kn**. Il saisit le compas à ouverture montré en **F**. En serrant sur les oreilles en **X** il écarte les pointes, en pressant en **Y** il les rapproche. Sur la carte marine l'échelle **D** n'existe pas, ni celle en **E**. Mais sur le côté vertical sont indiqués les parallèles toutes les minutes d'angle ce qui correspond à l'échelle **E** sur toute la hauteur de la carte. Il est alors facile à notre navigateur d'ouvrir son compas à 4,3 minutes d'angle. Ensuite il regarde sur le segment **BC** combien de fois il y a cet écart, c'est à dire $70 / 4.3 = 16,25$ heures. $12H + 16,25H = 28,25H$. Le port de Falmouth sera atteint à 4,25H du matin durant la nuit.



Ben Môa môa je n'ai que trois doigts à chaque patte, comptant avec mon orteil je suis en base six c'est à dire la moitié de la base des babylonitrus !

CONCLUSION : Nous avons vu dans le chapitre précédent que l'utilisation des minutes d'angle pour évaluer les distances sur une carte présente de nombreux avantages majeurs :

- Il devient inutile de faire figurer l'échelle de cette dernière qui est contenue dans les méridiens,
- Pour déterminer une durée de navigation il n'y a aucun calcul à faire.
- Pour positionner un mobile sur la Terre les angles sont faciles à utiliser alors que des distances par rapport à une origine cartésienne serait très difficile.
- Pour un satellite il est bien plus simple de mesurer des angles que des distances sur le géoïde.

Il ne faut donc pas s'étonner que ce système de positionnement se soit universellement imposé pour effectuer des calculs de navigation et qu'il soit impératif pour du positionnement GPS.

➤ Principe du "Pythagore angulaire".

Élémentaire dans sa mise en œuvre *sur une surface plane*, le théorème de Pythagore s'applique quelles que soient les unités utilisées, que ce soit le mètre, le kilomètre, le nautique ou avec des unités plus anciennes comme la coudée ou la brasse. Prenons un exemple Fig.4 dans lequel nous désirons déterminer la distance entre les points **A** et **B**. Le calcul de **D** est très facile depuis que le savant grec à annoncé son théorème 580 AV JC :

$$AB = \sqrt{AC^2 + CB^2}$$

Le détail technique vient du fait qu'en Fig.4 il s'agit d'une carte marine ou aviation. L'échelle des longueurs n'est pas indiquée. En revanche, sur le bord gauche et le coté du bas on a les indications des latitudes et des longitudes. On peut donc en déduire :

AC = 7' = 7NM = 7 x 1.852km = 12.964km

CB = 6' = 6NM = 6 x 1.852km = 11.112km

$$\text{Donc } AB = \sqrt{12.964^2 + 11.112^2} = 17.07\text{km}$$

Ce sera facile à coder en langage C++. Il serait également possible de calculer directement en milles marins :

$$AB = \sqrt{7^2 + 6^2} = 9.219\text{NM} \quad \text{et l'on retrouve : } 9.219 \times 1.852 = 17.07\text{km}$$

➤ Planisphère plan.

Pour représenter la surface terrestre sur une feuille de papier plane il suffit comme montré sur la Fig.5 de "décoller" du globe terrestre des tranches "verticales" et de les

placer dans un plan commun. C'est le principe de la *cartographie*

Mercator. On repère une tranche comme celle hachurée en rose.

Moins cette section est large plus la carte sera précise. Pour

cet exposé on va imaginer que l'on prévoit dix-huit sections.

On plaque sur le globe terrestre un "fuseau" en papier et on

trace les terres et les océans sur ce support plan mais cintré.

On redresse cette "tranche" à la verticale. Puis on prend la

feuille de papier **F** et on la place un peu décalée d'un

nouveau "fuseau", toujours en orientation verticale. On

recommence comme pour la section précédente. Quand on

aura effectué ces opérations dix-huit fois, on aboutira à la

carte relativement exacte de la Fig.6 mais qui n'est pas

spécifiquement conviviale pour qu'un instituteur puisse parler

de la dérive des continents à ses élèves de primaire. Aussi,

pour aboutir à des cartes qui ressemblent à celles que nous avons

l'habitude d'utiliser, on étire exagérément les limites des "fuseaux"

comme montré sur la Fig.7 aboutissant ainsi à des cartes qui sont

de plus en plus déformées au fur et à mesure que l'on s'écarte de l'équateur. Qu'elles soient outrageusement déformées ne nous

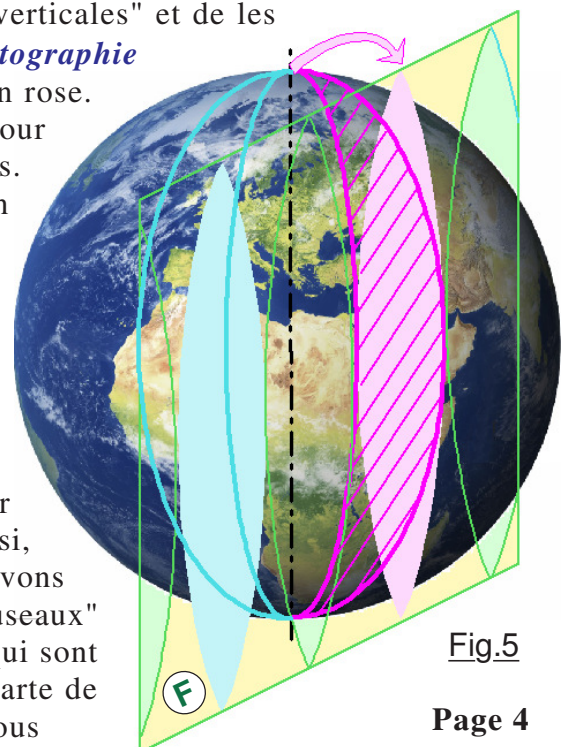
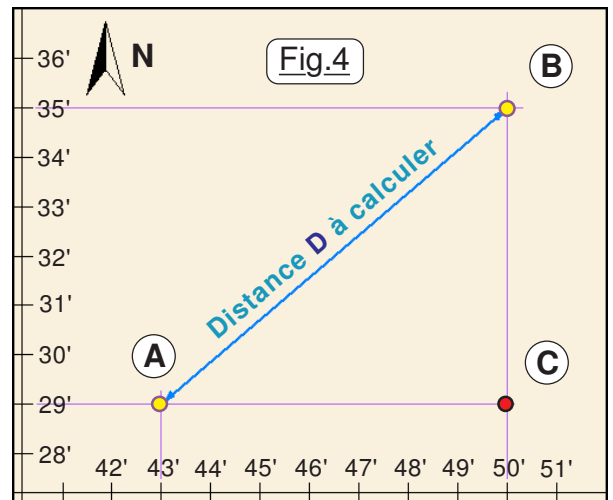
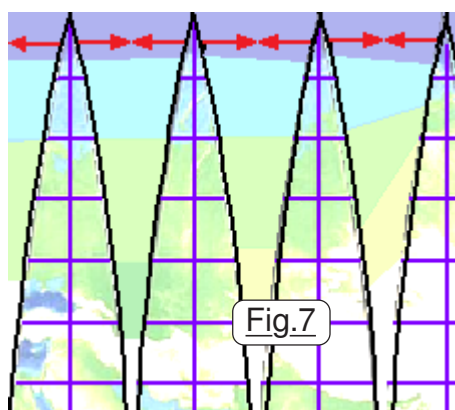
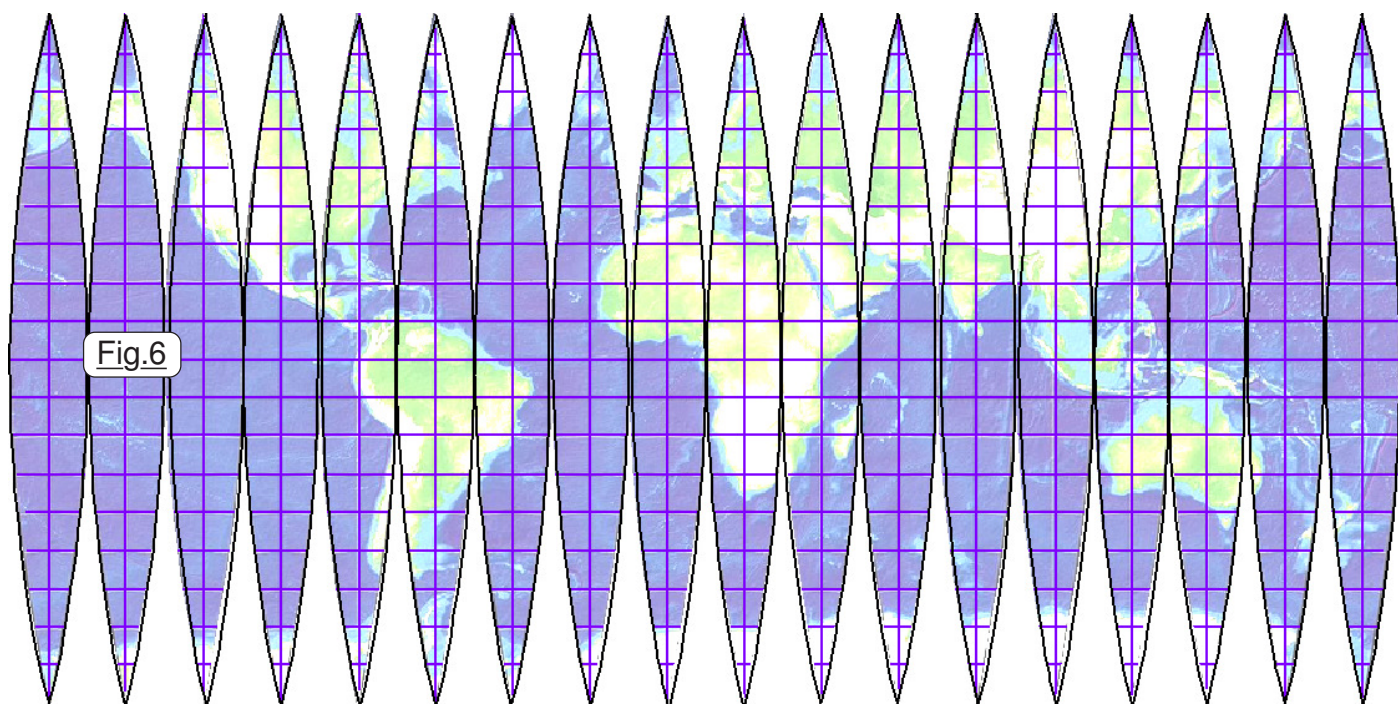
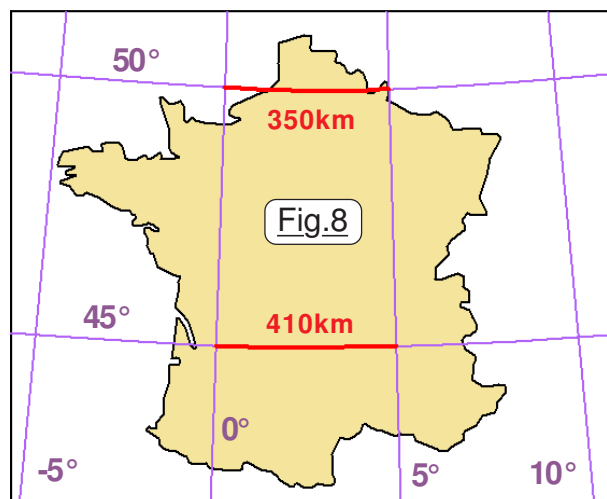


Fig.5

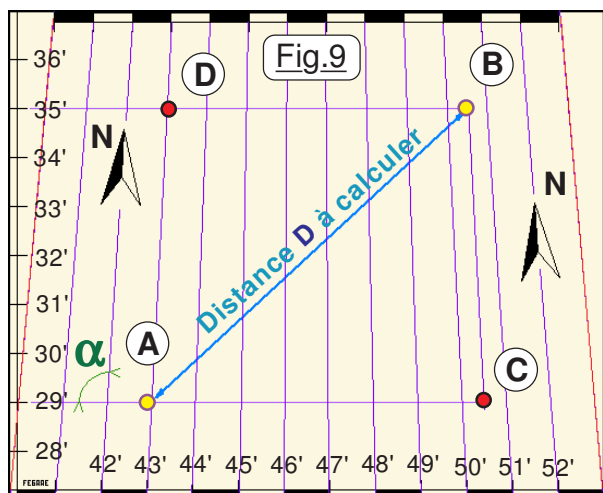


choque absolument pas puisque nous les voyons ainsi depuis notre plus tendre enfance. Pour bien comprendre à quel point les planisphères du type de celui de la Fig.7 sont déformés, il faut remarquer que toutes les pointes des "fuseaux" en haut du dessin correspondent en réalité à un seul et même endroit : Le pôle Nord. Sur la Fig.7 tout le bord en haut de la carte représente en réalité



un seul et même point, ce qui mathématiquement est inacceptable. Comme le montre la Fig.8 déjà à l'échelle de la France le phénomène est très marqué. En France un degré de longitude mesure environ 82 km au sud et seulement 70 km au nord. Donc, pour calculer dans notre programme la distance qui sépare notre GPS d'une cible, il faudra impérativement tenir compte du rétrécissement des longueurs en longitude. Une minute d'angle en longitude va alors représenter 1,852km sur l'Équateur et 0km au pôle Nord et au pôle sud et varie entre les deux.

➤ Principe du "Pythagore angulaire sphérique".



Rassurez-vous, tous les calculs proposés dans le chapitre précédent restent valides sur le principe. La seule différence dans la Fig.9 réside dans le fait que pour évaluer l'écart de position en longitude du point B il ne faut plus prendre en compte la distance angulaire AC, mais celle à la latitude de B, c'est à dire DB. On remarque qu'effectivement une minute d'angle y est plus "courte" à 35' de latitude qu'à 29' pour A et C. N'oublions pas au passage que l'angle α entre un méridien et un parallèle est un angle droit. Par ailleurs on remarque que la direction vers les pôles est "matérialisée" par les méridiens. Du coup la direction du Nord sur la carte est "convergente" comme le montrent les deux flèches N.

Dans notre exemple :

~~$AC = 7,5' = 7,5NM = 7,5 \times 1,852km = 12.964km$~~ car une minute d'angle = 1,852km.

~~$DB = 7,5' = 7,5NM = 7,5 \times 1,852km = 12.964km$~~

Cette façon de calculer est erronée, car en longitude une minute d'angle est directement fonction de la latitude à laquelle on mesure.

$AC = 7,5' = 7,5 \times k1NM = 7,5 \times k1 \times 1,852km = 11,85km.$

$DB = 7,5' = 7,5 \times k2NM = 7,5 \times k2 \times 1,852km = 10,37km.$

$D = \sqrt{(6 \times 1,852)^2 + (10,37)^2}$
soit environ 18km

Le dernier problème qui nous reste à résoudre avant de pouvoir coder en C++ consiste à trouver comment prendre en compte la variation du "coefficient de longitude" **k** en fonction de la latitude, sachant qu'un "fuseau" présente l'allure globale d'une ogive pointue. Du coup la variation n'est absolument pas linéaire partant de l'Équateur jusqu'au pôle considéré.

➤ La longueur d'un arc de cercle.

Élever des nombres au carré ou en prendre la racine carrée sera très facile puisque les fonctions qui permettent ces types de calculs sont natives dans le langage C++ d'Arduino. Il nous reste toutefois à trouver l'algorithme qui à partir de la Latitude sera capable de déterminer la valeur de **k**. Le premier concept sur lequel on va s'appuyer réside dans la proportionnalité qui existe entre la longueur **L** d'un arc de cercle d'ouverture angulaire α et de rayon **R**. La proportionnalité s'écrit :

$$L = R \times \alpha$$

Dans cette formule le rayon peut être exprimé dans les unités de votre choix. **L** sera alors dans les mêmes unités. Par contre, l'angle α sera obligatoirement exprimé en radians sachant que $360^\circ = 2 \times \text{PI}$ radians.

Exemple : **R** = 6367km $\alpha = 25^\circ$ soit $2 \times \text{PI} \times 25 / 360 = 0,43$ radians. On en déduit que la longueur **L** à la surface de la Terre sera de $6367 \times 0,43 = 2738km$ environ.

➤ La trigonométrie à notre secours.

C'est le deuxième concept qui vient à notre secours pour résoudre notre épineux problème. Pour déterminer l'algorithme qui sera intégré à notre programme d'application on va s'appuyer sur la Fig.11 sur laquelle l'axe N/S est représenté verticalement. Sur cette Terre que l'on a découpée dans tous les sens on suppose que le "fuseau" hachuré croisé en jaune encadre deux méridiens qui sont écartés angulairement d'une minute d'angle. **RT** est le rayon de la Terre et **RP** le rayon du parallèle de latitude α . En considérant la Fig.12 représentant le plan de coupe vertical, on peut affirmer que **CA** = **RP**. La trigonométrie nous indique que par définition **CA** = **RP** = **RT** cosinus(α). Comme l'angle β présente la même ouverture sur l'Équateur et sur le

Parallèle, il y a proportionnalité directe entre **RP**, **RT** ainsi que **LE** et **L**. On peut donc écrire :

$$\frac{RP}{RT} = \frac{L}{LE} = k$$

On remplace dans cette équation **RP** par sa valeur :

$$RT \cosinus(\alpha)$$

$$\frac{RT \cosinus(\alpha)}{RT} = k$$

On divise le numérateur et le dénominateur par **RT** et l'on arrive à :

$$k = \cosinus(\alpha)$$

OUF !

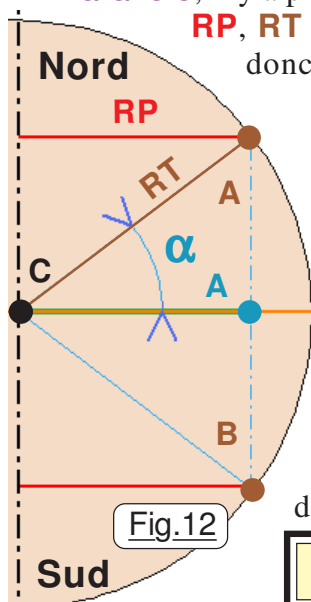


Fig.12

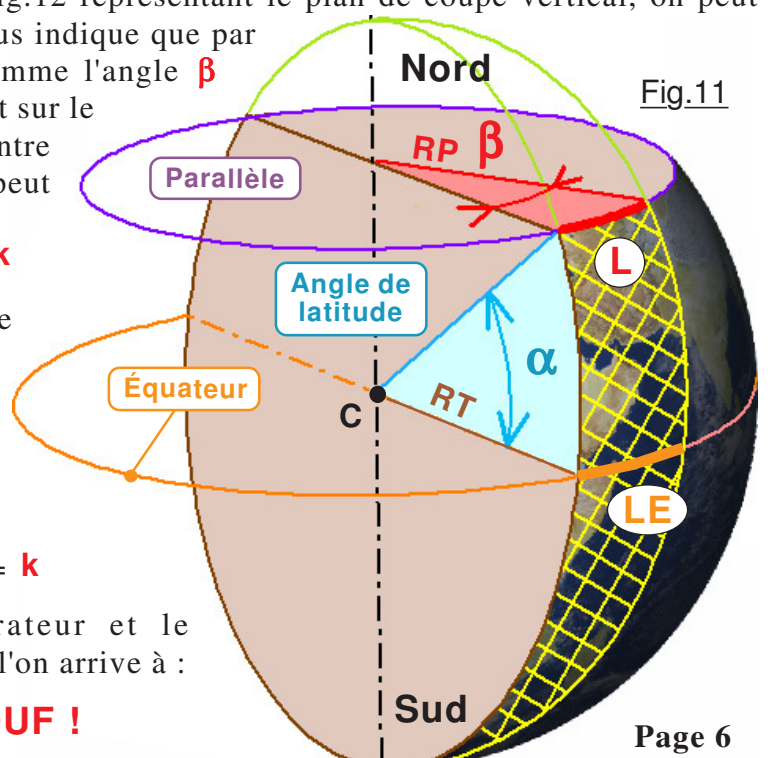


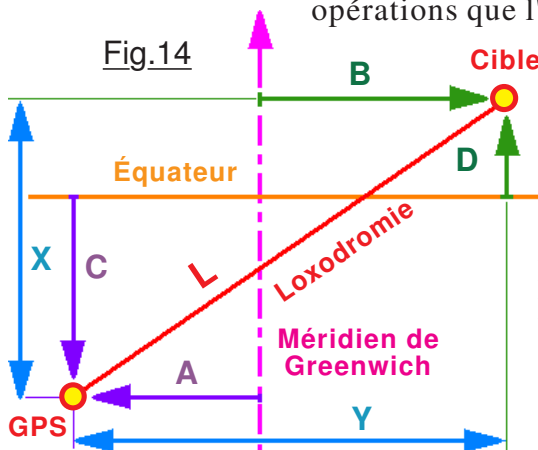
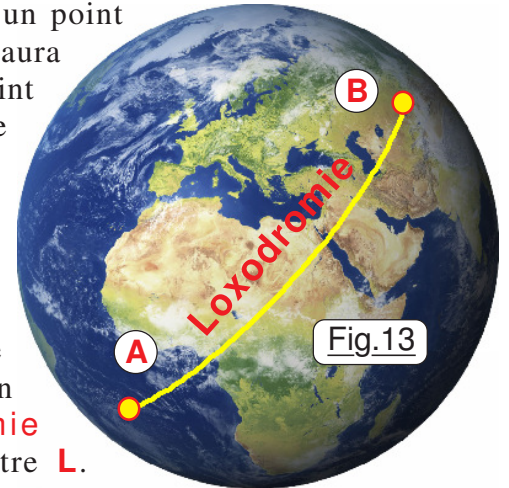
Fig.11

3) On retrouve le sourire et surtout un algorithme.

Cette laborieuse et absconde démonstration aura peut être perdu certaines et certains. Rassurez-vous, ce n'est pas très important. Je sais que les développements mathématiques ne sont pas forcément votre tasse de thé, que la trigonométrie n'est pas forcément connue de tous, d'autant plus qu'ici on est en trigonométrie sphérique. Peu importe, on est en loisir et personnellement, lorsque mes études avec Arduino m'enlissent dans un domaine qui me fait transpirer ... je fais confiance à la démonstration et je me contente d'en utiliser le résultat. C'est exactement ce que l'on va faire ici. Si j'ai alourdi le propos dans le chapitre précédent, c'est uniquement pour satisfaire toutes celles et ceux qui désirent savoir d'où vient le codage de notre programme.

➤ **Résumé des concepts utilisés pour calculer une loxodromie.**

Pour mémoire, le but de toutes les études qui précèdent consistent à vous proposer le cheminement qui a été nécessaire pour pouvoir avec notre Arduino calculer la distance "à vol d'oiseau" qui sépare notre GPS d'une Cible correspondant à un point d'intérêt sur Terre. Par le vocable "à vol d'oiseau" tout le monde aura compris qu'il s'agit du plus court chemin séparant sur Terre un point A d'un point B. Quand on est à la surface du géoïde, comme montré sur la Fig.13 cette plus courte distance n'est pas une droite, mais un arc de cercle qui par définition est nommé **Loxodromie**. C'est précisément ce que l'on cherche à calculer avec Arduino. La fig.14 va nous permettre de résumer les opérations que l'on va devoir effectuer en



langage C++. On désire calculer la longueur en km de la **Loxodromie** repérée avec la lettre **L**.

Par application du théorème de

Pythagore on peut affirmer que $L = \sqrt{X^2 + Y^2}$

Nous avons $X = C + D$ en tenant compte des signes.

Et également $Y = A + B$ toujours avec les signes. Enfin :

$A = \text{Delta_LON_GPS.}$

$B = \text{Delta_LON_cible.}$

$C = \text{Delta_LAT_GPS.}$

$D = \text{Delta_LAT_cible.}$

$X = \text{Delta_latitude.}$

$Y = \text{Delta_longitude.}$

On aboutit au calcul final à :

$$L = \sqrt{(C + D)^2 + ((A + B) \times k)^2}$$

① $\text{Delta_latitude} = \text{Delta_LAT_GPS} + \text{Delta_LAT_cible}$

② $k1 = \cos(\text{Delta_LAT_GPS})$

③ $k2 = \cos(\text{Delta_LAT_cible})$

④ $\text{Delta_longitude} = (\text{Delta_LON_GPS} \times k1) + (\text{Delta_LON_cible} \times k2)$

⑤ $L = \sqrt{(\text{Delta_latitude})^2 + (\text{Delta_longitude})^2}$

Il ne reste plus qu'à traduire ces calculs en langage C++ et à les effectuer dans l'ordre de cette liste qui résume l'algorithme pour déterminer la valeur de la **loxodromie** recherchée..

➤ **Il ne faut pas mélanger les torchons et les serviettes !**

Une petite difficulté vient se greffer dans notre démarche. Les angles qui interviennent dans ces équations ne sont pas tous exprimés en minutes d'angle, il faut impérativement utiliser des unités homogènes sans compter que le GPS nous fournit les valeurs non pas en nombres numériques mais en chaînes de caractères. Examinons comment **Experience_22.ino** est agencé pour effectuer ces transformations. On va analyser le **traitement pour la Longitude GPS** extraite du tableau **Tampon_GGA[]** dont la Fig.15 résume le contenu entièrement détaillé en page 9 du didacticiel. C'est l'algorithme pour traiter l'étape ①.

- **001** : Latitude du GPS exprimée en degrés.
- **31.31495** : Décimales de la Latitude en minutes d'angle.
- **E** : Si **E** le signe sera positif, si **W** le signe sera négatif.

Fig.15

\$GPGGA,173120.00,4427.32958,N,00131.31495,E,1,06,1.62,155.5,M,48.1,M,,*59

Pour pouvoir transformer une chaîne de caractères en un nombre pouvant servir à des calculs, il faut l'isoler dans un tableau de `char`. Ici la plus grande chaîne à extraire de `GGA` compte huit caractères. On réserve donc un tableau `char Tampon_calculs[9]` car il faut la place du délimiteur de fin de chaîne `'\0'`. *On commence par extraire la Longitude en degrés :*

```
Tampon_calculs[0] = Tampon_GGA[23]; Tampon_calculs[1] = Tampon_GGA[24];
Tampon_calculs[2] = Tampon_GGA[25]; Tampon_calculs[3] = '\0';
```

Tampon_calculs	0	0	1	\0				
----------------	---	---	---	----	--	--	--	--

On transforme la Longitude en un nombre exprimé en minutes d'angle :

```
Delta_LON_GPS = atoi(Tampon_calculs) * 60.0; // Angle en minutes.
```

On va extraire les décimales de la Longitude exprimées en minutes d'angle :

```
for (byte l = 0; l < 8; l++) Tampon_calculs[l] = Tampon_GGA[l + 26]; Tampon_calculs[8] = '\0';
```

Tampon_calculs	3	1	.	3	1	4	9	5	\0
----------------	---	---	---	---	---	---	---	---	----

On transforme les décimales de Longitude en un nombre que l'on ajoute à Delta_LON_GPS :

```
Delta_LON_GPS = Delta_LON_GPS + atof(Tampon_calculs);
```

On affecte le signe positif ou le signe négatif :

```
if (Tampon_GGA[35] == 'W') Delta_LON_GPS = Delta_LON_GPS * -1.0;
```

Multiplier un nombre par **-1** revient à en changer le signe.

➤ Calculer la Longitude de la Cible.

Lorsque l'on désire obtenir la position géographique d'un point d'intérêt, personnellement j'utilise **Google Maps** qui présente l'avantage de couvrir l'intégralité du globe. Ce n'est pas forcément que je compte me rendre au pôle sud par exemple, mais pour valider le programme il faut utiliser des points assez éloignés de la France et disséminés un peu partout sur la Terre. Il se prouve que **Google Maps** précise les coordonnées en degrés. Ces informations seront fournies manuellement au programme sous forme d'une chaîne de huit caractères correspondant aux sept chiffres que précise **Google Maps** auquel on ajoutera manuellement le **N**, le **S**, le **E** ou le **W** pour le signe. Par exemple on envisage pour **Cible** l'Île de Pâques :

```
char Latitude_cible[9] = "2718624S"; // Format DDmmmmmm.
```

```
char Longitude_cible[10] = "10943481W"; // Format DDDmmmmmm.
```

La taille des tableaux tient compte de celle de la chaîne de caractère plus une cellule pour le `'\0'`. Il aurait été plus simple de ne pas ajouter **S** et **W** et compléter en tête par le signe indiqué par **Google Maps** mais je trouve que c'est plus "parlant". On va retrouver des étapes analogues à celle pour le GPS :

On commence par extraire la Longitude en degrés :

```
Tampon_calculs[0] = Longitude_cible[0]; Tampon_calculs[1] = Longitude_cible[1];
Tampon_calculs[2] = Longitude_cible[2]; Tampon_calculs[3] = '\0';
```

On transforme la Longitude en un nombre exprimé en minutes d'angle :

```
Delta_LON_cible = atoi(Tampon_calculs) * 60.0; // Angle en minutes.
```

On va extraire les décimales de la Longitude exprimées en degrés angulaires :

```
for (byte l = 0; l < 5; l++) Tampon_calculs[l] = Longitude_cible[l + 3]; Tampon_calculs[5] = '\0';
```

On transforme les décimales de Longitude en un nombre que l'on ajoute à Delta_LON_cible :

```
Delta_LON_cible = Delta_LON_cible + (atof(Tampon_calculs) * 0.0006);
```

La valeur étant en degrés on multiplie par **60** mais il faut diviser par **100000** car ce sont des décimales.

On affecte le signe positif ou le signe négatif :

```
if (Longitude_cible[7] == 'W') Delta_LON_cible = Delta_LON_cible * -1.0;}
```

➤ Calculer la longueur de la Loxodromie.

Il n'est pas utile de détailler l'intégralité des traitements. Pour calculer `Latitude_GPS` et `Latitude_cible` on procède exactement comme explicité en page 8 sauf que ce sont les références des variables qui changent. Passons à la suite des calculs qui au final est plus simple que l'extraction des données et le tri entre "les torchons et les serviettes". La procédure qui affichera la distance de la cible en kilomètres (*Ou en MN à notre guise.*) commence par déterminer les longitudes et les latitudes :

```
void Affiche_la_PAGE_7() { // Affiche la distance à la cible.
```

```
  Determine_les_parametres_angulaires();
```

Étape ① : Calculer l'angle total en latitude :

```
if (Delta_LAT_cible > Delta_LAT_GPS) Delta_Latitude = Delta_LAT_cible - Delta_LAT_GPS;  
else Delta_Latitude = Delta_LAT_GPS - Delta_LAT_cible;
```

Commencer par enlever la plus grande à la plus petite tiendra compte du signe. Si les signes sont différents le calcul se résumera à une somme.

Étape ② et ③ : Calculer et prendre en compte **k1** et **k2** :

```
// La distance sur une minute d'angle en Longitude est fonction de la Latitude.
```

```
Delta_LON_cible = Delta_LON_cible * cos(radians(Delta_LAT_cible / 60));
```

```
Delta_LON_GPS = Delta_LON_GPS * cos(radians(Delta_LAT_GPS / 60));
```

Comme les angles `Delta_LAT_cible` et `Delta_LAT_GPS` sont exprimés en minutes d'angle, pour convertir en radians il faut **diviser par 60 pour les convertir en degrés**. Ensuite, sachant que $360^\circ = 2 \times \text{PI}$ radians, une simple proportion permet de convertir les degrés en radians. Le langage C++ d'Arduino nous fournit la fonction `radians` qui effectue directement cette transformation.

Étape ④ : Calculer l'angle total en longitude :

```
if (Delta_LON_cible > Delta_LON_GPS)  
  Delta_Longitude = Delta_LON_cible - Delta_LON_GPS;  
else Delta_Longitude = Delta_LON_GPS - Delta_LON_cible;
```

Ici aussi, commencer par enlever la valeur la plus grande à la donnée la plus petite tiendra compte du signe. Si les signes sont différents le calcul se résumera ainsi à une somme.

Étape ⑤ : Calculer **L** en utilisant le théorème de Pythagore :

```
Distance_cible = sqrt(sq(Delta_Longitude) + sq(Delta_Latitude));
```

Attention : Les deux opérateurs `sq` et `sqrt` ont une écriture qui se ressemble mais sont très différents :

- `sq` retourne le carré de l'argument. (*Pour square.*)
- `sqrt` retourne la racine carrée de l'argument. (*Pour square root.*)

Remarque : Mathématiquement, on ne peut extraire la racine carrée d'un nombre que s'il est de signe positif. Hors `Delta_Longitude` et `Delta_Latitude` résultant d'une différence peuvent parfaitement présenter un signe négatif. **Avant le calcul d'une racine carrée il faut toujours vérifier que l'argument sera positif.** Dans notre application ce n'est pas nécessaire car les deux variables étant élevées au carré sont forcément positives.

Conclusion : certaines et certains vont probablement considérer que le contenu de ce document est pour le moins indigeste, voir un peu hors sujet dans le cadre d'une activité purement ludique. Vous pouvez totalement l'ignorer et "prendre le programme comme il vient". Toutefois, sans ce type d'analyse, la programmation est illusoire. C'est précisément dans nos limites que réside la restriction de l'éventail des chemins qui s'offrent à nous ... et on doit faire avec ...

Ben Môa môa, j'ai toute la Latitude de cheminer
comme je le désire car je ne suis pas obligée de calculer
des cosinustrucmachin pour me promener !

