Les Systèmes Géodésiques de l'Île de La Réunion
Méthodes de Conversion

Pierre Brial
Outre Mer Topographie SARL
31bis, route de l'Éperon
97435 Saint-Gilles-les-Hauts
p.brial@orange.fr

2015

1 Introduction

Les cartes topographiques sont couvertes de deux ou plusieurs quadrillages (ou d'amorces de quadrillages), appelés carroyages ou grilles.

L'une de ces grilles est graduée en degrés : il s'agit des coordonnées dites géographiques : la latitude et la longitude.

Un deuxième carroyage est en général présent, gradué en mètres ou en kilomètres. Il s'agit de la grille de la projection, c'est à dire la tentative de représenter un volume sphérique, la terre, sur une surface plane, la carte. Il existe plusieurs types de projections. Toutes se basent sur des formules mathématiques qui convertissent les degrés des coordonnées géographiques en coordonnées planes dans le système métrique. L'intérêt d'utiliser des coordonnées planes est dû au fait qu'il

est plus facile de faire des calculs et des mesures sur des données métriques que sur des degrés.

Un système géodésique se défini ainsi par la combinaison d'un système de coordonnées géographique, ou datum, établi par son point fondamental ou une détermination spatiale, par les paramètres d'un ellipsoïde représentant une forme approchée du globe terrestre, et par la projection qui lui est attachée [1].

Indiquer les coordonnées d'un point n'est donc pas suffisant. Il convient aussi de préciser dans quelle système géodésique, et avec quelle projection, elles ont été établie.

À l'île de La Réunion, les plans et cartes disponibles peuvent utiliser deux systèmes géodésiques différents :

- L'ancien système Piton des Neiges, mis en place en 1950 et encore utilisé au début des années 2000, s'appuyant sur une projection Gauss-Laborde
- Le nouveau système IGN 92, en vigueur depuis le 28 décembre 2000([2]), et s'appuyant sur une projection UTM.

Les coordonnées d'un même point géographique diffèrent suivant le système utilisé. Il importait donc de définir les moyens de convertir aisément les coordonnées d'un système à l'autre.

2 Historique

En octobre 1716, un relèvement des caps de l'île fut effectué par Augustin Panon, lors de l'expédition Criais, ce qui permis, pour la première fois, d'en dessiner un contour proche de la réalité [3].

La première triangulation de l'île fut réalisée en 1825 par l'ingénieur géographe Schneider [4]. Cette opération ne donna toutefois pas lieu à la mise en place d'un système de coordonnées planes sur La Réunion.

En 1948, une nouvelle triangulation fut réalisée par l'Institut Géographique National, par une mission astro-géodésique dirigée par M. Héraut [5]. Cette triangulation s'appuyait sur deux points astronomiques, à Saint-Denis et à Saint-Pierre, et une ligne de base à Bras-Panon. Cette opération donna lieu à la mise en place du

1. Brial, P. & Shaalan, C. (2009),Introduction à la Géodésie et au Géopositionnement par Satellites, p.12.
2. Décret d'application n°2000-1276 publié au Journal Officiel n°300 du 28/12/2000.
3. comm.pers. Auguste de Villèle
4. Germanaz, Christian (2011), « Cartographier Bourbon aux XVII^{e}-XIX^{e} Siècles », CFC n°210, p.112
5. IGN (1987), Île de La Réunion, Répertoire des coordonnées et altitudes des points géodésiques.

système géodésique IGN 1949, communément appelé « Piton des Neiges » (PDN), présentant les caractéristiques suivantes :

- Nom : IGN 1949
- Alias : Réunion 1947, Piton des Neiges, PDN
- Code EPSG : 6626
- Méridien d'origine : Greenwich
- Ellipsoïde : International Hayford 1909
  a=6 378 388 m ; f=1/297
- Projection associée : Gauss Laborde Réunion
  $\lambda_0 = 55°32'$ E
  $\phi_0 = 21°07'$ S
  $E_0 = 160\,000$ m
  $N_0 = 50\,000$ m
  $k_0 = 1$

En 1992, le réseau de stations fut remesuré par satellites, avec un canevas GPS dense, rattaché au système mondial ITRF91 époque 93.0([6]). Il s'agit d'une réalisation précise du World Geodetic System 1984 (WGS84).

L'IGN constata des discordances allant jusqu'à 30 cm sur les coordonnées issues de l'ancien réseau[7]. L'IGN publia des fiches signalétiques avec les nouvelles coordonnées des points du réseau.

Suite à ces opérations, un nouveau système géodésique fut défini, sous le nom de Réseau Géodésique de La Réunion 1992 (RGR 92). Ses caractéristiques sont les suivantes :

- Nom : Réseau Géodésique de La Réunion 1992
- Alias : RGR 92
- Code EPSG : 6627
- Méridien d'origine : Greenwich
- Ellipsoïde : GRS 1980
  a=6 378 137 m ; f=1/298.257223563
- Projection associée : UTM zone 40 Sud
  $\lambda_0 = 57°$ E
  $\phi_0 = 0°$
  $E_0 = 50\,000$ m
  $N_0 = 1\,000\,000$ m
  $k_0 = 0.9996$

6. http ://geodesie.ign.fr/contenu/fichiers/documentation/srtom/systemeReunion.pdf
7. IGN - Luzet, Claude (1993), Courrier du 2 août 1993 à l'attention des mairies de l'île de La Réunion

Par décret n°2000-1276 publié au Journal Officiel n°300 le 28 décembre 2000, le système géodésique RGR 92 devenait le système de coordonnées officiel et obligatoire pour l'île de La Réunion.

3 Problématique

Bien que le nouveau système géodésique de La Réunion ait été déterminé en 1992, les fiches publiés par l'IGN en 1993 indiquaient des coordonnées planes converties dans l'ancien système Piton des Neiges, en projection Gauss-Laborde Réunion.

L'ancien système de coordonnées a ainsi continué à être utilisé jusque vers 2010, dans la production de nombreux plans et autre documents topographiques. Il est donc important, pour pouvoir travailler sur ces documents, de savoir convertir les coordonnées PDN en coordonnées RGR 92.

Suite à la campagne GPS de 1992, l'IGN publiait des paramètres permettant, par une transformation de Helmert, d'effectuer ces conversions [8]. Toutefois, ces paramètres étaient *différents* suivant le sens de la transformation :

TABLE 1 – **du système RGR 92 vers l'ancien système P. des Neiges :**

Tx (m)	Ty (m)	Tz (m)	D (ppm)	Rx (")	Ry (")	Rz (")
-789.990	627.333	89.685	32.2083	-0.6072	-76.8019	10.5680

TABLE 2 – **de l'ancien système P. des Neiges vers le système RGR 92 :**

Tx (m)	Ty (m)	Tz (m)	D (ppm)	Rx (")	Ry (")	Rz (")
789.524	-626.486	-89.904	-32.3241	0.6006	76.7946	-10.5788

avec :

- Tx, Ty, Tz : translations sur les trois axes de l'espace
- Rx, Ry, Rz : rotations
- D : facteur d'échelle.

L'IGN ajoutait : *« Les valeurs des facteurs d'échelle et de certaines rotations sont très importantes, ce qui fait que ces paramètres ne sont pas publiables du point de vue purement géodésique. »*

8. Rapport de Claude Luzet, IGN 1992-1993, p.33

En 2001, nous constations que l'utilisation de ces paramètres dans le GPS System 500 construit par Leica Geosystems générait des erreurs de plusieurs centimètres. Interrogé sur la possibilité d'utiliser ce GPS dans le système de coordonnées Piton des Neiges, le support de Leica Geosystems nous transmettait les éléments d'une similitude 3D à seulement 3 paramètres (translations), donc encore moins précise, et ajoutait [9] :

« Ce système n'est pas satisfaisant pour l'altimétrie, mais n'est pas non plus satisfaisant pour la planimétrie (selon la dernière personne qui est venue faire des formations GPS à la Réunion et qui l'avait utilisé pour tester sa validité). Il faut donc en déduire qu'il n'y a pas de paramètres de transformation satisfaisant sur la totalité de l'île. La marche à suivre sera de calculer des transformations locales en fonction des points d'appui disponibles (c'est la méthode que les utilisateurs de GPS appliquent à la Réunion). La personne qui viendra vous former à la Réunion vous expliquera les manipulations nécessaires et les conditions requises. »

Ces affirmations sont étonnantes. En effet, les coordonnées publiées des points géodésiques dans le système Piton des Neiges après 1992 s'appuient sur une nouvelle détermination par satellite, et sont donc cohérentes sur l'ensemble de l'île. Un changement de système de coordonnées est une opération mathématique qui peut être assimilé au déplacement physique d'un repère orthonormé dans l'espace, une rotation sur ses trois axes, et éventuellement une réduction ou un agrandissement par l'application d'un facteur d'échelle. Il est donc *toujours* possible de procéder à l'opération inverse en appliquant simplement une translation, une rotation et un facteur d'échelle de signes opposés, ce qui équivaut à ramener le repère orthonormé à son emplacement de départ.

Pourquoi l'IGN a t'il donc publié, en 1993, des paramètres de translation, rotation et facteur d'échelle *différents suivant le sens de la transformation*, et pourquoi ces paramètres ne fonctionnent pas sur un GPS Leica ?

4 Solution

Un changement de coordonnées géocentrique par transformation de Helmert se calcule avec la formule matricielle suivante [10] :

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + D \cdot R \cdot \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \quad (1)$$

9. email de Franck Reffreger, Support Technique LEICA, 12 septembre 2001

10. Tran, Dinh Trong (2013) « Analyse rapide et robuste des solutions GPS pour la tectonique », Thèse de doctorat, Université de Nice - Sophia Antipolis, p.10-11

avec

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos R_x & -\sin R_x \\ 0 & \sin R_x & \cos R_x \end{bmatrix} \begin{bmatrix} \cos R_y & 0 & -\sin R_y \\ 0 & 1 & 0 \\ \sin R_y & 0 & \cos R_y \end{bmatrix} \begin{bmatrix} \cos R_z & -\sin R_z & 0 \\ -\sin R_z & \cos R_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

où :

- T : vecteur de translation
- R : matrice de rotation
- D : facteur d'échelle.

Une telle formule présentait deux difficultés, dans son application a un algorithme de calcul informatique :

1. Le développement de ces matrices donne une formule très longue, consommant de la place dans un programme informatique. Or dans les années 1980-1990, la capacité des ordinateurs pouvant être utilisés sur le terrain était limitée.
2. Le calcul de la transformation nécessite la détermination de plusieurs fonction trigonométriques successives. Or ces fonctions ne sont pas directement calculées telle qu'elle par un microprocesseur, mais estimées soit par une série convergente polynomiale, composée d'additions et de multiplications, soit en les assimilant à des rotations successives de vecteurs (technique CORDIC) [11]. Ce calcul mobilise un nombre important d'instructions du processeur, et est donc très lent. À titre d'exemple, le calcul matriciel ci-dessus, pour un seul jeu de coordonnées, mettrait environ 19 secondes à s'exécuter sur une calculatrice programmable HP-41C, appareil fréquemment utilisé sur le terrain dans les années 1980. Sur une machine plus récente tel que le Sharp PC-1262, avec un processeur cadencé à 768 kHz, un tel calcul prenait environ 6 secondes.

En géodésie, les angles de rotations entre deux systèmes de coordonnées géocentriques sont trés faibles, en général inférieurs à 10^{-5} radians (2"). dans ce cas, le sinus de l'angle peut être assimilé à la valeur de l'angle lui-même, en radian :

$$\sin 0.00001 = 0.000009999999999833333... \simeq 0.00001$$

Partant de ce principe, les formules (1) et (2) deviennent :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + D \cdot \begin{bmatrix} 1 & -R_z & R_y \\ R_z & 1 & -R_x \\ -R_y & R_x & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3)$$

11. Cochran, David S. (1972), « Algorithms and accuracy in the HP-35 », HP Journal n°7206

qui se développe en :

$$\begin{aligned} X &= x + T_x + Dx - R_z y + R_y z \\ Y &= y + T_y + Dy + R_z x - R_x z \\ Z &= z + T_z + Dz - R_y x + R_x y \end{aligned} \tag{4}$$

Cette dernière formule est simple, aisée à programmer, et, ne contenant aucune fonction transcendante, est rapide à calculer. De fait, c'est cette formule qui est utilisée dans la plupart des logiciels de topographie, de cartographie ou de SIG. L'application des paramètres publié par l'IGN pour la transformation Piton des Neiges - RGR 92, avec cette formule, donne des résultats exacts avec une précision millimétrique. Ces paramètres ont donc été calculés pour pouvoir être utilisé ainsi. C'est ce qui explique pourquoi il existe deux jeux de paramètres suivant le sens de la transformation. Les rotations sont ici trop fortes suivant les axes y et z : 77" et 11". Dès lors, la transformation simplifiée n'est plus réversible.

On pourra donc se demander pourquoi l'utilisation de ces paramètres dans le GPS System 500 génère des erreurs. Ceci est du au mode d'utilisation de l'appareil. Le GPS effectue ses calculs et détermine sa position en coordonnées géocentriques. Or l'utilisateur travaille dans les coordonnées de son chantier, qui ne sont pas nécessairement rattachées à un système géodésique, ni orientées vers le nord. Le GPS doit donc utiliser des algorithmes de transformation permettant des rotations très importantes, et de ce fait, utilise les formules non simplifiées (1) et (2).

Il convenait dès lors de déterminer les paramètres applicables pour ce GPS.

Le GPS 500 dispose d'une fonction embarquée permettant le calcul automatique des paramètres d'une similitude 3D, à partir des données des ellipsoïdes et de la projection éventuellement utilisés, et d'un jeu de points [12]. Pour celà, nous avons utilisé les points du réseau géodésique calculé en 1992, dont les coordonnées sont connues dans les deux systèmes.

Les paramètres calculés sont les suivants :

Tx (m)	Ty (m)	Tz (m)	D (ppm)	Rx (")	Ry (")	Rz (")
-789.7754	626.8602	89.673	32.2706	0.60653	76.79827	-10.57522

L'utilisation de ces paramètres dans le GPS System 500, ainsi que dans le GPS System 1200, plus récent, donne des résultats identiques, au millimètre prés, aux jeux de coordonnées publiées par l'IGN pour les stations du réseau géodésique RGR 92.

12. Leica Geosystems (2001), « GPS System 500 - Manuel technique de référence », version 3.0, pp.244-252

5 Applications pratiques

Nous disposons désormais de deux jeux de paramètres de transformations, dont le choix dépendra de l'algorithme utilisé. Les connaissances acquises au cours de ces recherches ont permis de réaliser notre propre application de conversion de coordonnées, appelée « Mimee ».

5.1 Algorithmes

Les algorithmes à programmer doivent permettre les opérations suivantes :

— changement de datum : passage d'un système de coordonnées géocentriques à un autre

— conversion de coordonnées géographiques en géocentriques, et inversement

— projections : passage de coordonnées géographiques en coordonnées planes, et inversement

— conversions entre unités angulaires : radians, grades, degrés décimaux, degrés sexagésimaux.

La documentation utilisée provient principalement des fiches géodésiques de l'Institut Géographique National, et de la publication de l'IOGP « Coordinate Conversions and Transformations including Formulas » [13]. Les paramètres par systèmes ont été obtenus sur la base de donnée de l'EPSG (*EPSG Geodetic Parameter Dataset*), sur le site internet *http ://www.crs-geo.eu* (Coordinate Reference Systems in Europe), ou sur les sites des agences géodésiques nationales.

5.2 En Basic

Une toute première version a été réalisée sur des ordinateurs de poche de la marque Sharp, programmables en Basic. Bien que ces appareils soient considérés comme obsolètes et ne sont plus commercialisés depuis longtemps, ils sont encore fréquemment utilisés tant sur le terrain qu'au bureau grâce à leur robustesse, leur simplicité, et leur autonomie de plusieurs mois. Le même programme a ainsi été porté sur les modèles PC-1262, PC-1403, et PC-E500S. La syntaxe en est similaire d'un modèle à l'autre. Toutefois, l'affichage du choix des systèmes géodésiques et projections par des listes déroulantes était trop lent sur les deux premiers modèles. Ces fonctions ont donc été programmées dans le langage machine du processeur SC61860. Le code source du programme pour le SHARP PC-E500S est reproduit en annexe 1. Ce programme en Basic a ensuite été adapté pour pouvoir être utilisé comme macro dans un tableur Libreoffice. Seule la projection Mercator Transverse était gérée par ces versions.

13. Geomatics Guidance Note number 7, part 2 – April 2015

5.3 En C

Afin de pouvoir étendre l'application Mimee à d'autre types de projection, une librairie géodésique a été développée en C, contenant les fonctions suivantes :

- ivf2e : calcul de l'excentricité à partir de applatissement de l'ellipse
- geocentric : conversion de coordonnées geographiques vers geocentriques
- geographic : conversion de coordonnées geocentriques vers geographiques
- datumshift : changement de système géodésiques
- LongueurMeridien : longueur d'un méridien entre l'équateur et une latitude donnée
- ProjectionTM : projection Transverse Mercator
- InvProjectionTM : inverse de la projection Transverse Mercator
- ProjectionConique
- InvProjectionConique
- ProjectionObliqueMercator
- InvProjectionObliqueMercator
- ProjectionCassini
- deg : degrés-minutes-secondes vers degrés décimaux
- dms : degrés décimaux vers degrés-minutes-secondes
- dmm2deg : degrés et minutes décimales vers degrés décimaux
- deg2dmm : degrés décimaux vers degrés et minutes décimales
- ZoneUTM : zone UTM d'après la longitude
- LongitudeOrigineUTM : longitude d'origine d'une zone UTM.

Le code source de cette librairie est reproduit en annexe 2. Elle a permis le développement de plusieurs versions du logiciel en C sur différentes plate-formes. La librairie et les programmes sont des logiciels libres publiés sous la licence GNU GPL V2 (GNU General Public License). Ils sont disponibles sur le site `http://80calcs.pagesperso-orange.fr/Navigation/Mimee.html`

5.4 En Ajax

La disponibilité d'un code en C a permis tout naturellement de traduire Mimee en javascript, langage très similaire, et de réaliser une page web HTML permettant d'effectuer des conversions de coordonnées à partir d'un navigateur web, en ligne, ou hors ligne si le site est enregistré. Cette version est ainsi accessible à une multitude de terminaux fixes ou mobiles, tels que les tablettes et smartphones, quel que soit le système d'exploitation. Elle est disponible ici :

- Version complète :
  http://80calcs.pagesperso-orange.fr/Downloads/Mimee_online.html
- Version spécifique à l'Île de La Réunion :
  http://omt-geometres.pagesperso-orange.fr/omt_fichiers/Mimee_Reunion.html

A Annexes

A.1 Mimee en Basic pour Sharp PC-E500S

```
10 "MIMEE":CLS :GOTO 5000
1840 "W84"DATA 6378137,8.181919084E-2,0,0,0,0,0,0
1841 "RUN"DATA 6378388,8.199188998E
   -2,-789.99,627.333,89.685,3.2208E-5
1842 DATA -2.943788672E-6,-3.723461185E-4,5.123510983E-5
1850 "MAD"DATA 6378388,8.199188998E-2,189,242,91,0,0,0,0
1865 "MUR"DATA 6378249.145,8.248340005E
   -2,770,-158,498,0,0,0,0
1870 "RUV"DATA 6378388,8.199188998E
   -2,-789.524,626.486,89.904,3.23241E-5
1871 DATA -2.911790969E-6,-3.723107271E-4,5.12874697E-5
1875 "W72"DATA 6378135,8.181881066E-2,0,0,-4.5,-2.19E
   -7,0,0,-2.68586779E-6
1877 "EGY"DATA 6378200,8.181333401E
   -2,130,-110,13,0,0,0,0
1879 "IFA"DATA 6378200,8.181333401E
   -2,121.8,-98.1,10.70,-.2263E-6,0,0
1880 DATA -2.685867793E-6
1881 "ALE"DATA 6378200,8.181333401E
   -2,223.505,-196.43,31.601,-7.719541E-6,0,0
1882 DATA 2.197545031E-5
1883 "USD"DATA 6378249.145,8.248340005E
   -2,770,-158,498,0,0,0,0
3010 "U"ZN=INT ((LO+180)/6)+1
3020 L0=(ZN-1)*6-177:RETURN
5000 CLEAR :WAIT 0:PRINT "SYSTEMES GEODESIQUES"
5010 RESTORE "DATUM":READ D,G:DIM D$(-(D>=G)*D-(D<G)*G):
   FOR I=1TO D:READ D$(I):NEXT I
5015 I=D,M=D,A=1:PRINT "DATUM DEPART:":GOSUB "MENU":K$=
   LEFT$ (D$(I),3)
5016 A=2:PRINT "DATUM ARRIVEE:":GOSUB "MENU":F$=LEFT$ (D
   $(I),3)
5020 FOR I=1TO G:READ D$(I):NEXT I
5061 I=2,M=G:A=3:PRINT "GRILLE DEPART:":GOSUB "MENU":C$=
   LEFT$ (D$(I),3)
5062 PRINT "GRILLE ARRIVEE:":GOSUB "MENU":O$=LEFT$ (D$(I
   ),3)
```

```
5063 WAIT 0:PRINT "Chargement...":DEGREE
5064 DEGREE :RESTORE K$:READ AA,EA,BA,CA,DA,KA,FA,CC,OA:
   RESTORE F$:READ AB,EB,B,I,D,X,Z,NB,Y
5065 KA=X-KA:OA=Y-OA:CC=NB-CC:FA=Z-FA:BA=B-BA:CA=I-CA:DA
   =D-DA:NB=180/PI
5072 IF LEFT$ (C$,1)="D"GOTO 5081
5075 RESTORE C$:READ LG,L,KG,XG,YG:A=AA,E=EA:GOSUB "BX":
   EC=E2,ED=E4,EF=E6,BG=B
5077 ER=(1-SQR (1-EC))/(1+SQR (1-EC))
5081 IF LEFT$ (O$,1)="D"GOTO 5085
5083 RESTORE O$:READ LH,L,KH,XH,YH:E=EB,A=AB:GOSUB "BX":
   BO=B
5085 EA=EA*EA:EB=EB*EB
5090 CLS :INPUT "Est=";X1,"Nord=";Y1,"He=";Z1
5100 IF LEFT$ (C$,1)<>"D"GOTO "GR"
5110 IF C$<>"DDD"GOSUB C$
5115 IF K$=F$ LET X2=X1,Y2=Y1,H2=Z1:GOTO 5230
5120 PRINT "Changement Datum...":GOSUB 7140
5150 XB=XA+BA+KA*XA-OA*YA+CC*ZA,YB=YA+CA+KA*YA+OA*XA-FA*
   ZA
5170 ZB=ZA+DA+KA*ZA-CC*XA+FA*YA
5180 P=SQR (XB*XB+YB*YB),X2=ATN (YB/XB),P0=ATN (ZB/P):IF
    XB<0 LET X2=X2+180*SGN YB
5190 J=SIN P0,W=SQR (1-EB*J*J),N=AB/W,Y2=ATN (ZB/P/(1-N*
   EB/(N+P*COS P0+ZB*J-AB*W)))
5210 IF ABS (Y2-P0)>0.00001LET P0=Y2:GOTO 5190
5220 G=SIN Y2,H2=P/COS Y2-AB/SQR (1-EB*G*G)
5230 IF LEFT$ (O$,1)<>"D"GOTO "SG"
5231 IF O$<>"DDD"GOSUB O$+"I"
5240 WAIT :CLS :PRINT X2,Y2,H2,"  Zone "+STR$ ZN:WAIT 0:
   GOTO 5090
7010 "MENU": WAIT 4:LOCATE 15,A:PRINT RIGHT$ (D$(I),LEN
   D$(I)-3);"       ":P=ASC INKEY$ :IF P=13 RETURN
7020 I=I+(P=4)-(P=5):I=I+((I>M)-(I<1))*M:GOTO 7010
7070 "DMS"X1=DEG X1,Y1=DEG Y1:RETURN
7080 "DMSI"X2=DMS X2,Y2=DMS Y2:RETURN
7090 "DMM"X=SGN X1*INT ABS X1,X1=X+(X1-X)/0.6
7100 Y=SGN Y1*INT ABS Y1,Y1=Y+(Y1-Y)/0.6:RETURN
7110 "DMMI"X=SGN X2*INT ABS X2,X2=X+(X2-X)*0.6
7120 Y=SGN Y2*INT ABS Y2,Y2=Y+(Y2-Y)*0.6:RETURN
```

```
7140 Q=COS Y1:S=SIN Y1
7141 N=AA/SQR (1-EA*S*S),XA=(N+Z1)*Q*COS X1,YA=(N+Z1)*Q*
   SIN X1,ZA=(N*(1-EA)+Z1)*S:RETURN
7160 "GR"X=X1-XG,M=(Y1-YG)/KG+BG
7170 IF C$<>"UTM"GOTO 7190
7175 PRINT "Hemisphere(N/S):":H$=INPUT $(1):IF H$="N"
   LET M=Y1/KG
7178 INPUT "Zone=";ZN:GOSUB 3020:LG=L0
7190 PRINT "Projection Inverse...":EP=EC/(1-EC)
7200 U=M/AA/(1-EC/4-3*ED/64-EF/51.2)*NB
7201 P1=(1.5*ER-0.84375*ER^3)*SIN (2*U)+(1.3125*ER*ER
   -1.71875*ER^4)*SIN (4*U)
7202 P1=(P1+151*ER^3/96*SIN (6*U))*NB+U,G=1-EC*SIN P1*
   SIN P1,T=TAN P1,H=COS P1,N1=AA/SQR G
7205 T1=T*T,C1=EP*H*H,R1=AA*(1-EC)/G^1.5,D=X/N1/KG
7206 G1=5+3*T1+10*C1-4*C1*C1-9*EP
7207 G2=61+90*T1+298*C1+45*T1*T1-252*EP-3*C1*C1
7208 Y1=P1-(N1*T/R1*(D*D/2-G1*D^4/24+G2*D^6/720))*NB
7209 X1=(D-(1+2*T1+C1)*D^3/6+(5-2*C1+28*T1-3*C1*C1+8*EP
   +24*T1*T1)*D^5/120)/H
7210 X1=LG+X1*NB:GOTO 5115
7215 "BX"E2=E*E,E4=E2*E2,E6=E4*E2,E8=E4*E4,E1=E8*E2
7220 B0=1+0.75*E2+0.703125*E4+0.68359375*E
   6+0.6729125977*E8+0.6661834717*E1
7221 B2=0.375*E2+0.46875*E4+0.5013020833*E
   6+0.5126953125*E8+0.516027832*E1
7222 B4=0.1171875*E4+0.205078125*E6+0.259552002*E
   8+0.2907943726*E1
7223 B6=35/768*E6+0.1025390625*E8+0.1522705078*E1
7224 B8=315/16384*E8+3465/65536*E1
7225 B1=693/81920*E1:X=COS (2*L)
7242 X=COS (2*L),B=A*(1-E2)*(B0*L/NB-SIN (2*L)*((((B1*X-
   B8)*X+B6)*X-B4)*X+B2)):RETURN
7250 "SG"PRINT "Projection...":H=COS Y2:IF O$<>"UTM"GOTO
    7290
7260 YH=10000000*(1-SGN (Y2))/2,LO=X2:GOSUB "U":LH=L0
7280 IF X2>-6AND X2<=0LET LH=-3
7281 IF X2>0AND X2<=6LET LH=3
7290 L=Y2:GOSUB 7242:U=TAN Y2,T=U*U
7301 EP=E2/(1-E2)*H*H,R=A/SQR (1-(E*SIN Y2)^2),Z=(X2-LH)
```

```
   *H/NB,Z2=Z*Z
7302 A3=EP-T+1,A4=EP*(4*EP+9)+5-T
7303 A5=2*EP*(7-29*T)+T*(T-18)+5
7304 A6=30*EP*(9-11*T)+T*(T-58)+61
7305 X2=KH*R*Z*(Z2*(A5*Z2/20+A3)/6+1)+XH
7306 Y2=KH*(B-BO+R*U*Z2*(Z2*(A6*Z2/30+A4)/12+1)/2)+YH:
   GOTO 5240
7500 "DATUM"DATA 10,9,"ALEAlexandrie","EGYEgypte1907","
   IFAIFAO"
7503 DATA "MADMadagascar","MURMaurice","RUN->Reunion"
7505 DATA "RUVReunion->","USDUser","W72WGS72"
7510 DATA "W84WGS84"
7600 "GRILLE"DATA "AGAAgalega","DDDD.ddd","DMMD.MMmm","
   DMSD.MMSSss","KILEgypte","MUGMaurice"
7610 DATA "PDNGL.Reunion","USEUser","UTMUTM"
7710 "AGA"DATA 56.58644444,-10.34788889,.9996,1E4,2E4
7715 "PDN"DATA 55.53333333,-21.11666667,1,16E4,5E4
7720 "UTM"DATA 0,0,.9996,5E5,1E7
7725 "KIL"DATA 31,30,1,615E3,81E4
7730 "USE"DATA
   57.52182778,-20.19506944,1,1000000,1000000
7731 "MUG"DATA
   57.52182778,-20.19506944,1,1000000,1000000
```

A.2 Librairie Géodésique

```
/*-----------------------------------------------------
ivf2e : ellipse flattening to excentricity
geocentric : geographic to geocentric coordinates
   conversion
geographic : geocentric to geographic coordinates
   conversion
datumshift : datum1 to datum2 geocentric coordinates
   conversion
LongueurMeridien : length of a meridien between Equator
   and given latitude
ProjectionTM : Geographic to Transverse Mercator plane
   coordinates
InvProjectionTM : Transverse Mercator plane to
   Geographic coordinates
ProjectionConique : Geographic to Conic plane
   coordinates
InvProjectionConique : Conic plane to Geographic
   coordinates
ProjectionObliqueMercator : Geographical to Oblique
   Mercator coordinates
InvProjectionObliqueMercator : Oblique Mercator to
   Geographic
ProjectionCassini : Geographic to Cassini plane
   coordinates : *Not Yet Tested*
deg : degrees-minutes-seconds to decimal degree
dms : decimal degree to degrees-minutes-seconds
dmm2deg : degrees-decimal minutes to decimal degree
deg2dmm : decimal degree to degrees-decimal minutes
ZoneUTM : UTM zone given longitude
LongitudeOrigineUTM : longitude of origin given UTM zone

Copyright :

This library is free software; you can redistribute it
   and/or modify
it under the terms of the GNU General Public License as
   published by
the Free Software Foundation; either version 2 of the
   License, or
```

```
(at your option) any later version.

This library is distributed in the hope that it will be
   useful,
but WITHOUT ANY WARRANTY; without even the implied
   warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the
GNU General Public License for more details.

You should have received a copy of the GNU General
   Public License
along with this program; if not, write to the Free
   Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston,
   MA  02110-1301  USA

-------------------------------------------------*/

#include <math.h>
#include <stdlib.h>

#define radian 0.017453292519943
#define PI     3.141592653589793

struct coord{double x;double y;double z;};

struct ellipsoide
{
double a; // semi-major axis
double e; // excentricity
};

struct grille
{
// all angles in radian !
double Lon0; // Longitude of origin
double Lat0; // Latitude of origin
double k;    // Scale factor
double E0;   // False easting
```

```
double N0;    // False northing
double p1;    // Conic proj.: 1st standard parallel
// Oblique proj.: azimuth of the center line
//                passing through the centre
//                of the projection
double p2;    // Conic proj.: 2nd standard parallel
// Oblique proj.: rectified bearing of the
//                center line (usually p1=p2)
double p3;    // Conic proj.: correction on spherical
   longitude (rarely used)
// Oblique proj.: p3 = 1 : Hotine Oblique Mercator
//                p3 = 0 : Standard Oblique Merctor
};

struct Datum
{
double dx; // geocentric X translation
double dy; // geocentric Y translation
double dz; // geocentric Z translation
double d;  // correction of scale
double rx; // rotation around X axis (radian)
double ry; // rotation around Y axis (radian)
double rz; // rotation around Z axis (radian)
};

struct ConstOM{double B;double A;double H;double g0;
   double l0;double uc;};



/*-------------------------------------------------
Procedure:      ivf2e
Purpose:        Compute ellipse's excentricity given
   inverse flattening
Input:          ellipse's inverse flatening (1/f) (double
   )
Output:         ellipse's excentricity (e) (double)
-------------------------------------------------*/
double ivf2e(double ivf) {return sqrt((2-1/ivf)/ivf);}
```

```
/*-------------------------------------------------
Procedure:      geocentric
Purpose:        geographical to geocentric coordinates
   conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
geographic coordinates in radians and meters (in
   structure coord)
Output:         geocentric coordinates (in structure
   coord)
-------------------------------------------------*/
struct coord geocentric(struct ellipsoide ellips,struct
   coord crd)
{
struct coord crdout;
double n=ellips.a/sqrt(1-ellips.e*ellips.e*pow(sin(crd.y
   ),2));
crdout.x=(n+crd.z)*cos(crd.y)*cos(crd.x);
crdout.y=(n+crd.z)*cos(crd.y)*sin(crd.x);
crdout.z=(n*(1-ellips.e*ellips.e)+crd.z)*sin(crd.y);
return crdout;
}

/*-------------------------------------------------
Procedure:      geographic
Purpose:        geocentric to geographical coordinates
   conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
geocentric coordinates (in structure coord)
Output:         geographic coordinates in radians and
   meters(in structure coord)
-------------------------------------------------*/
struct coord geographic(struct ellipsoide ellips,struct
   coord crd)
{
struct coord crdout;
double j,w,n,p0;
double p = sqrt(crd.x*crd.x+crd.y*crd.y);
```

```
crdout.y = atan(crd.z / p);
crdout.x = atan(crd.y / crd.x);
if (crd.x<0.0)
{
if (crd.y<0.0) crdout.x-=PI;
else crdout.x+=PI;
}
boucle:
{
p0=crdout.y;
j = sin(p0);
w = sqrt(1 - ellips.e * ellips.e * j * j);
n = ellips.a / w;
crdout.y = atan(crd.z / p /
(1 - n * ellips.e*ellips.e / (n + p * cos(p0) + crd.z *
   j - ellips.a * w)));
}
if (fabs(crdout.y-p0) > 0.000001) goto boucle;
crdout.z = p / cos(crdout.y) - ellips.a / sqrt(1 -
   ellips.e * ellips.e * pow(sin(crdout.y),2));
return crdout;
}

/*----------------------------------------------------
Procedure:      datumshift
Purpose:        Convert geocentric coordinates from datum
    1 to datum 2
Input:          Datum 1 data, Datum 2 data (in structures
    datum),
geocentric coordinates in datum 1 (in structure coord)
Output:         geocentric coordinates in datum 2 (in
   structure coord)
----------------------------------------------------*/
struct coord datumshift(struct Datum datum1,struct Datum
    datum2,struct coord crd)
{
struct coord crdout;
double delta_dx=datum1.dx-datum2.dx;
double delta_dy=datum1.dy-datum2.dy;
double delta_dz=datum1.dz-datum2.dz;
```

```
double delta_d=datum1.d-datum2.d;
double delta_rx=datum1.rx-datum2.rx;
double delta_ry=datum1.ry-datum2.ry;
double delta_rz=datum1.rz-datum2.rz;
crdout.x= crd.x + delta_dx + delta_d * crd.x - delta_rz
   * crd.y + delta_ry * crd.z;
crdout.y= crd.y + delta_dy + delta_d * crd.y + delta_rz
   * crd.x - delta_rx * crd.z;
crdout.z= crd.z + delta_dz + delta_d * crd.z - delta_ry
   * crd.x + delta_rx * crd.y;
return crdout;
}


/*-------------------------------------------------
Procedure:      LongueurMeridien
Purpose:        Meridian lenght between Equator and given
    latitude
Input:          ellipsoide's data (in structure
   ellipsoide)
Latitude (in radians)
Output:         Meridian Lenght (in meters)
-------------------------------------------------*/
double LongueurMeridien(struct ellipsoide ellips, double
    lat)
{
double E2 = ellips.e * ellips.e;
double E4 = E2*E2;
double E6 = E4*E2;
double E8 = E4*E4;
double E10 = E8*E2;
double B0 = 1+.75*E2+.703125*E4+.68359375*E6
   +.6729125977*E8+.6661834717*E10;
double B2 = .375*E2+.46875*E4+.5013020833*E6
   +.5126953125*E8+.516027832*E10;
double B4 = .1171875*E4+.205078125*E6+.259552002*E8
   +.2907943726*E10;
double B6 = .04557291667*E6+.1025390625*E8+.1522705078*
   E10;
double B8 = .01922607421875*E8+.0528717041015625*E10;
```

```
double B10 = .00845947265625*E10;
double x = cos(2 * lat);
return ellips.a*(1-E2)*(B0*lat-sin(2*lat)*((((B10*x-B8)*
   x+B6)*x-B4)*x+B2));
}

/*-------------------------------------------------
Procedure:      ProjectionTM
Purpose:        geographical to Transverse Mercator plane
    coordinates conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
geographic coordinates in radians and meters (in
   structure coord)
Output:         plane coordinates in meters (in structure
    coord)
-------------------------------------------------*/
struct coord ProjectionTM(struct ellipsoide ellips,
   struct grille gri,struct coord crd)
{
struct coord crdout;
double h = cos(crd.y);
double B0 = LongueurMeridien(ellips,gri.Lat0);
double B = LongueurMeridien(ellips,crd.y);
double U = tan(crd.y);
double T = U * U;
double e2 = ellips.e*ellips.e;
double EP = e2 / (1 - e2) * h * h;
double R = ellips.a / sqrt(1 - pow((ellips.e * sin(crd.y
   )),2));
double Z = (crd.x - gri.Lon0) * h;
double z2 = Z * Z;
double A3 = EP - T + 1;
double A4 = EP * (4 * EP + 9) + 5 - T;
double A5 = 2 * EP * (7 - 29 * T) + T * (T - 18) + 5;
double A6 = 30 * EP * (9 - 11 * T) + T * (T - 58) + 61;
crdout.x = gri.k * R * Z * (z2 * (A5 * z2 / 20 + A3) / 6
    + 1) + gri.E0;
crdout.y = gri.k * (B - B0 + R * U * z2 * (z2 * (A6 * z2
```

```
    / 30 + A4) / 12 + 1) / 2) + gri.N0;
crdout.z = crd.z;
return crdout;
}

/*-------------------------------------------------
Procedure:      InvProjectionTM
Purpose:        Transverse Mercator plane to geographical
    coordinates conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
plane coordinates in meters (in structure coord)
Output:         geographic coordinates in radians and
   meters (in structure coord)
-------------------------------------------------*/
struct coord InvProjectionTM(struct ellipsoide ellips,
   struct grille gri,struct coord crd)
{
struct coord crdout;
double EC = ellips.e * ellips.e;
double ED = EC * EC;
double EF = ED * EC;
double EP = EC / (1 - EC);
double x = crd.x-gri.E0;
double m = (crd.y - gri.N0) / gri.k + LongueurMeridien (
   ellips,gri.Lat0);
double U = m / ellips.a / (1 - EC / 4 - 0.046875 * ED -
   EF / 51.2);
double ER = (1 - sqrt(1 - EC)) / (1 + sqrt(1 - EC));
double P1 = (1.5 * ER - 0.84375 * pow(ER,3)) * sin(2 * U
   ) + (1.3125 * ER * ER - 1.71875
* pow(ER,4)) * sin(4 * U)+ 151.0 * pow(ER,3) / 96.0 *
   sin(6 * U) + U;
double G = 1 - EC * sin(P1) * sin(P1);
double T = tan(P1);
double h = cos(P1);
double N1 = ellips.a / sqrt(G);
double T1 = T * T;
double C1 = EP * h * h;
```

```
double R1 = ellips.a * (1 - EC) / pow(G,1.5);
double D = (x / N1) / gri.k;
double G1 = 5 + 3 * T1 + 10 * C1 - 4 * C1 * C1 - 9 * EP;
double G2 = 61 + 90 * T1 + 298 * C1 + 45 * T1 * T1 - 252
    * EP - 3 * C1 * C1;
crdout.y = P1 - (N1 * T / R1 * (D * D / 2 - G1 * pow(D
   ,4) / 24 + G2 * pow(D,6) / 720));
crdout.x = (D - (1 + 2 * T1 + C1) * pow(D,3) / 6 + (5 -
   2 * C1 + 28 * T1 - 3 * C1 * C1
+ 8 * EP + 24 * T1 * T1) * pow(D,5) / 120) / h + gri.
   Lon0;
crdout.z = crd.z;
return crdout;
}

/*-------------------------------------------------
Procedure:      tc
Purpose:        Function used by ProjectionConique and
   InvProjectionConique
-------------------------------------------------*/
double tc(double e, double lat)
{
double ep=e*sin(lat);
double t=tan(PI/4-lat/2)/pow((1-ep)/(1+ep),e/2);
if (t<0) t=0;
return t;
}

/*-------------------------------------------------
Procedure:      ProjectionConique
Purpose:        geographical to Conic plane coordinates
   conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
geographic coordinates in radians and meters (in
   structure coord)
Output:         plane coordinates in meters (in structure
    coord)
-------------------------------------------------*/
```

```
struct coord ProjectionConique(struct ellipsoide ellips,
   struct grille gri,struct coord crd)
{
struct coord crdout;
double m1,m2,t1,t2,tF,t,n,F,r,rF,theta;

if (gri.p1==gri.p2) gri.p1=gri.p2=gri.Lat0;
else gri.k=1;

m1=cos(gri.p1)/sqrt(1-pow(ellips.e*sin(gri.p1),2));
m2=cos(gri.p2)/sqrt(1-pow(ellips.e*sin(gri.p2),2));
t1=tc(ellips.e,gri.p1);
t2=tc(ellips.e,gri.p2);
tF=tc(ellips.e,gri.Lat0);
t=tc(ellips.e,crd.y);

if (gri.p1==gri.p2) n=sin(gri.p1);          // conic 1
   Standard Parallel
else n=(log(m1)-log(m2))/(log(t1)-log(t2)); // conic 2
   Standard Parallel

F=m1/(n*pow(t1,n));
r=gri.k*ellips.a*F*pow(t,n);
rF=gri.k*ellips.a*F*pow(tF,n);
theta=n*(crd.x-gri.Lon0);

crdout.x = gri.E0+r*sin(theta-gri.p3);
crdout.y = gri.N0+rF-r*cos(theta-gri.p3);
crdout.z = crd.z;
return crdout;
}

/*-----------------------------------------------
Procedure:      InvProjectionConique
Purpose:        Conic plane to geographical coordinates
   conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
plane coordinates in meters (in structure coord)
```

```
Output:         geographic coordinates in radians and
   meters (in structure coord)
-------------------------------------------------*/
struct coord InvProjectionConique(struct ellipsoide
   ellips,struct grille gri,struct coord crd)
{
struct coord crdout;
double n,F,rF,rp,tp,thetap,p0,ep,m1,m2,t1,t2,tF;
if (gri.p1==gri.p2) gri.p1=gri.p2=gri.Lat0;
else gri.k=1;
m1=cos(gri.p1)/sqrt(1-pow(ellips.e*sin(gri.p1),2));
m2=cos(gri.p2)/sqrt(1-pow(ellips.e*sin(gri.p2),2));
t1=tc(ellips.e,gri.p1);
t2=tc(ellips.e,gri.p2);
tF=tc(ellips.e,gri.Lat0);
if (gri.p1==gri.p2) n=sin(gri.p1);          // conic 1
   Standard Parallel
else n=(log(m1)-log(m2))/(log(t1)-log(t2)); // conic 2
   Standard Parallel
F=m1/(n*pow(t1,n));
rF=ellips.a*F*pow(tF,n);
rp=sqrt(pow(crd.x-gri.E0,2)+pow(rF-crd.y+gri.N0,2));
tp=pow(rp/(gri.k*ellips.a*F),1.0/n);
thetap=atan((crd.x-gri.E0)/(rF-crd.y+gri.N0));
crdout.x=(thetap+gri.p3)/n+gri.Lon0;
crdout.y=gri.Lat0;
do
{
p0=crdout.y;
ep=ellips.e*sin(p0);
crdout.y = PI/2-2*atan(tp*pow((1-ep)/(1+ep),ellips.e/2))
   ;
}
while (fabs(crdout.y-p0)>0.0000000001);
crdout.z = crd.z;
return crdout;
}

/*-------------------------------------------------
Procedure:      ConstantesObliqueMercator
```

```
Purpose:        Compute various constants used by
   ProjectionObliqueMercator
and InvProjectionObliqueMercator
-----------------------------------------------*/
struct ConstOM ConstantesObliqueMercator(struct
   ellipsoide ellips,struct grille gri)
{
struct ConstOM c;
double D2,F,t0,D;
double e2=pow(ellips.e,2);
double esp=ellips.e*sin(gri.Lat0);
double esp2=pow(esp,2);
c.B=sqrt(1+e2*pow(cos(gri.Lat0),4)/(1-e2));
c.A=ellips.a*c.B*gri.k*sqrt(1-e2)/(1-esp2);
t0=tc(ellips.e,gri.Lat0);
D=c.B*sqrt(1-e2)/cos(gri.Lat0)/sqrt(1-esp2);
if (D<1) D2=1;
else D2=D*D;
F=D+copysign(sqrt(D2-1),gri.Lat0);
c.H=F*pow(t0,c.B);
c.g0=asin(sin(gri.p1)/D);
c.l0=(F-1.0/F)/2.0*tan(c.g0);
if (fabs(c.l0-1)<0.000000000001) c.l0=1;
c.l0=gri.Lon0-asin(c.l0)/c.B;
if (fabs(gri.p1-PI/2)<0.0000000001) c.uc=c.A*(gri.Lon0-c
   .l0);
else c.uc=c.A/c.B*atan2(sqrt(D2-1),cos(gri.p1))*copysign
   (1,gri.Lat0);
return c;
}

/*-----------------------------------------------
Procedure:      ProjectionObliqueMercator
Purpose:        geographical to Oblique Mercator plane
   coordinates conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
geographic coordinates in radians and meters (in
   structure coord)
```

```
Output:         plane coordinates in meters (in structure
    coord)
-------------------------------------------------*/
struct coord ProjectionObliqueMercator(struct ellipsoide
    ellips,struct grille gri,struct coord crd)
{
struct coord crdout;
struct ConstOM c=ConstantesObliqueMercator(ellips,gri);
double Q,S,U,V,v,u;
//double u0;
Q=c.H/pow(tc(ellips.e,crd.y),c.B);
S=(Q-1/Q)/2;
V=sin(c.B*(crd.x-c.l0));
U=(S*sin(c.g0)-V*cos(c.g0))/(Q+1/Q)*2;
v=c.A*log((1-U)/(1+U))/2/c.B;
u=c.A/c.B*atan2(S*cos(c.g0)+V*sin(c.g0),cos(c.B*(crd.x-c
   .l0)));//Hotine Oblique Mercator
if (gri.p3==0) u-=c.uc;//Oblique Mercator
crdout.x=v*cos(gri.p2)+u*sin(gri.p2)+gri.E0;
crdout.y=u*cos(gri.p2)-v*sin(gri.p2)+gri.N0;
crdout.z=crd.z;
return crdout;
}

/*-------------------------------------------------
Procedure:      InvProjectionObliqueMercator
Purpose:        Oblique Mercator plane to geographical
   coordinates conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
plane coordinates in meters (in structure coord)
Output:         geographic coordinates in radians and
   meters (in structure coord)
-------------------------------------------------*/
struct coord InvProjectionObliqueMercator(struct
   ellipsoide ellips,struct grille gri,struct coord crd)
{
struct coord crdout;
struct ConstOM c=ConstantesObliqueMercator(ellips,gri);
```

```
double vp,up,Qp,Sp,Vp,Up,tp,x,e2,e4,e6,e8;

vp=(crd.x-gri.E0)*cos(gri.p2)-(crd.y-gri.N0)*sin(gri.p2)
   ;
up=(crd.y-gri.N0)*cos(gri.p2)+(crd.x-gri.E0)*sin(gri.p2)
   ;
if (gri.p3==0) up+=c.uc; //Oblique Mercator
Qp=exp(-c.B*vp/c.A);
Sp=(Qp-1/Qp)/2;
Vp=sin(c.B*up/c.A);
Up=(Vp*cos(c.g0)+Sp*sin(c.g0))/(Qp+1/Qp)*2;

tp=pow(c.H/sqrt((1+Up)/(1-Up)),1/c.B);

x=PI/2-2*atan(tp);
e2=ellips.e*ellips.e;
e4=e2*e2;
e6=e4*e2;
e8=e4*e4;
crdout.y=x+sin(2*x)*(e2/2+5*e4/24+e6/12+13*e8/360)
+sin(4*x)*(7*e4/48+29*e6/240+811*e8/11520)
+sin(6*x)*(7*e6/120+81*e8/1120)
+sin(8*x)*(4279*e8/161280);

crdout.x=c.l0-atan2(Sp*cos(c.g0)-Vp*sin(c.g0),cos(c.B*up
   /c.A))/c.B;

crdout.z=crd.z;
return crdout;
}


/*-----------------------------------------------
Procedure:      ProjectionCassini
Warning :       *** Not Tested !!!***
Purpose:        geographical to Cassini plane coordinates
    conversion
Input:          ellipsoid's data (in structure ellipsoide
   )
grid's data (in structure grille)
```

```
geographic coordinates in radians and meters (in
   structure coord)
Output:         plane coordinates in meters (in structure
    coord)
-------------------------------------------------*/
/*
struct coord ProjectionCassini(struct ellipsoide ellips,
   struct grille gri,struct coord crd)
{
struct coord crdout;
double h = cos(crd.y);
double B0=LongueurMeridien(ellips,gri.Lat0);
double B=LongueurMeridien(ellips,crd.y);
double U = tan(crd.y);
double T = U * U;
double e2=ellips.e * ellips.e;
double EP = e2 / (1 - e2) * h * h;
double R = ellips.a / sqrt(1 - pow((ellips.e * sin(crd.y
   )),2));
double Z = (crd.x - gri.Lon0) * h;
double z2 = Z * Z;
crdout.x = R * (Z-T*pow(Z,3)/6 - (8-T+8*EP) * T* pow(Z
   ,5)/120) + gri.E0;
crdout.y = B - B0 + R * U * (z2/2 + (5-T+6*EP)*pow(z2,2)
   /24) + gri.N0;
crdout.z=crd.z;
return crdout;
}
*/


/*** Various Sexagesimal conversion functions ***/

double deg(double x) // D.MMSSss to D.DDD
{
double ax,s,h,m;
ax = fabs(x) + 1e-12;
h = floor(ax);
m = floor((ax - h) * 100);
s = (ax - h) * 10000 - m * 100;
```

```
return copysign(h+m/60+s/3600,x);
}

double dms(double x) // D.DDD to D.MM.SSss
{
double ax,h,m,s;
ax = fabs(x) + 1e-12;
h = floor(ax);
m = (ax - h) * 60;
s = (m - floor(m)) * 0.006;

return copysign(h + floor(m) / 100 + s,x);
}

double dmm2deg(double x) // D.MMmm to D.DDD
{
double d = copysign(floor(abs(x)),x);
return d + (x - d) / 0.6;
}

double deg2dmm(double x) // D.DDD to D.MMmm
{
double d = copysign(floor(abs(x)),x);
return d + (x - d) * 0.6;
}


/*** UTM Zones functions ***/

int ZoneUTM(double longitude) // Compute UTM Zone number
    given longitude in degree
{return floor((longitude/radian + 180) / 6) + 1;}


double LongitudeOrigineUTM(int zone) // compute UTM zone
    longitude of origin (in radian)
// given UTM zone number
{return (double)((zone - 1) * 6 - 177)*radian;}
```

Table des matières