Centre National de la Recherche Scientifique

Laboratoire de Physique des Plasmas
UMR 7648
Ecole Polytechnique, route de Saclay, 91128 PALAISEAU CEDEX

ROCOTLIB :
a Coordinate Transformation Library for Solar-Terrestrial studies

by

Patrick ROBERT

Version 3.2, February 2020

FOREWORD

The ROCOTLIB library (RObert's COordinate Transformation LIBrary) is a set of software modules to perform transformations between the various coordinate systems used in geophysical and magnetospheric studies. Most of the frames of reference are geocentric, and are thus independent of the position of the point of observation ; nevertheless, some local frames are also considered.

In addition to coordinate transformations, the library also provides a set of modules to perform format conversions and other operations associated with epoch, date and time, allowing calendar computations.

This library was originally developed in 1992 by P. Robert, CNRS/CRPE, with support from ESA within the framework of preparation of the CLUSTER mission; since then it has been regularly updated by the author. Original document is entitled: " Document de travail DT/CRPE/1231, CLUSTER Software Tools, Part 1 : Coordinate Transformation Library, Version 1.1 ", by Patrick ROBERT, RPE/TID, CNRS/CNET/CRPE, Juillet 1993. A 1.7 version of software and documentation was published in 2003 as an internal report, named RI-CETP/02/2003. The actuel 3.2 version is the last at this time, and include new transformation such as Minimum Variance Analysis, or TPN local frame, as well as use of Euler's angles.

ROCOTLIB exists in both FORTRAN 77 and in FORTRAN 90, and of course can be run on any computer where these compilers are available. Translation in C language has been done and tested. Translation in IDL programming languages is also available. Each transformation or module corresponds to a subroutine in FORTRAN or C, and to a procedure in IDL. The package delivered to the user includes sources and makefiles of the library, examples of its use, a test program and the corresponding test output file to check the validity of the installation on the user's machine. The test program has been developed and tested using the following FORTRAN compilers :

- LINUX /Intel i686 , g77 - GNU project FORTRAN Compiler (v0.5.24)
- LINUX /x86_64, Fortran Intel(R) 64 Compiler XE, for F77 and F90 codes
- LINUX /x86_64, f77 GNU Fortran (Ubuntu 5.4.0-6 ubuntu1 16.04.4) for F77 codes
- LINUX /x86_64, gfortran GNU Fortran (Ubuntu 5.4.0) for F77 and F90 codes
- LINUX /x86_64, gcc (Ubuntu 5.4.0-6 ubuntu 1 16.04.4) for C codes.

This document is the full documentation of the ROCOTLIB library and includes two parts:

- The first defines the various coordinate systems considered, and gives the mathematical formulas and matrices to pass from one system to another.

- The second part is the user's manual for the FORTRAN, C and IDL library. It give the complete list of available modules, the role for each one, and a description of the input and output variables. The example programs and the test program are also commented.
Documentation and code source are available on www.lpp.polytechnique.fr/-Patrick-Robert- or mail to : Patrick.robert@lpp.polytechnique.fr

This library is a living product, and can be completed in the future by other transformations, or derived applications. Any comments are welcome.

Contents

Part I

DEFINITIONS AND MATHEMATICAL FORMULAS

Chapter 1

DESCRIPTION OF COORDINATE SYSTEMS

Most of the coordinate systems described are geocentric, with any exceptions as the dipole meridian system and the VDH system, which are local coordinate systems and thus depend of the position of the point of observation.

1.1 Geocentric Equatorial Inertial System (GEI)

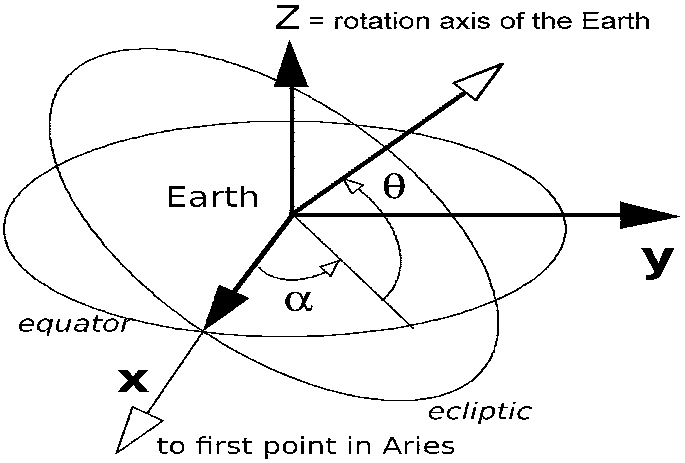


Figure 1.1: GEI system definition

The Z-axis is parallel to the rotation axis of the Earth.
The X-axis is defined by the intersection of the equator plane and the ecliptic plane, and is pointing towards the first point of Aries (Sun position at the vernal equinox).

one can define the *right ascension* α and the *declination* θ as:

right ascension $\alpha = \tan^{-1}(V_Y/V_X)$

$$\begin{array}{rlll} \text{with} & \alpha \text{ in} & [\ \ 0^\circ, & 180^\circ\ ] \quad \text{for } V_Y > 0 \\ & \alpha \text{ in} & [180^\circ, & 360^\circ\ ] \quad \text{for } V_Y < 0 \end{array}$$

declination $\theta = \sin^{-1}(V_Z/V)$

$$\text{with} \quad \theta \text{ in} \quad [\ -90^\circ, \quad +90^\circ\ ]$$

1.2 Geographic System (GEO)

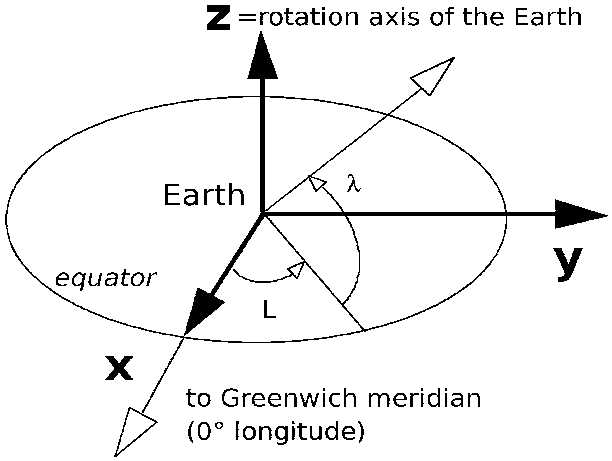


Figure 1.2: GEO system definition

The Z-axis is parallel to the rotation axis of the Earth.
The X and Y axis are included in the equator plane.
The X axis is pointing from the centre of the Earth to the Greenwich meridian (0° longitude).

The GEO system is fixed with the rotating Earth. Longitude L and latitude λ are defined in this system in the same way as right ascension and declination in GEI system.

1.3 Geomagnetic system (MAG)

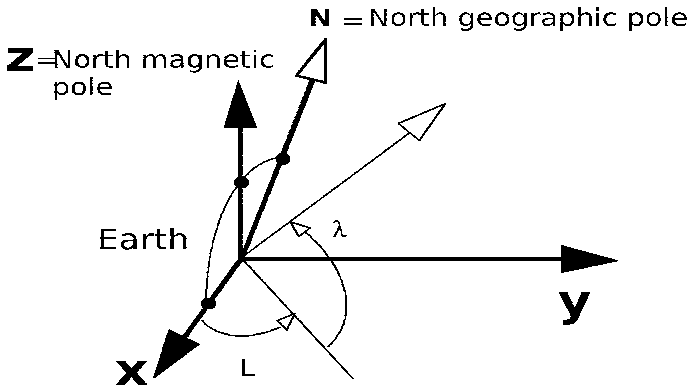


Figure 1.3: MAG system definition

The Z-axis is parallel to the magnetic dipole axis.
If $\vec{N}$ is the North geographic pole, the $\vec{N}$,$\vec{Z}$ and $\vec{X}$ vector are in the same plane.
The Y-axis is defined as $\vec{Y} = -\vec{Z} \times \vec{N}$

The MAG system is fixed with the rotating Earth. The magnetic longitude L and magnetic latitude λ are defined in this system in the same way as right ascension and declination in GEI system.

1.4 Geocentric Solar Ecliptic system (GSE)

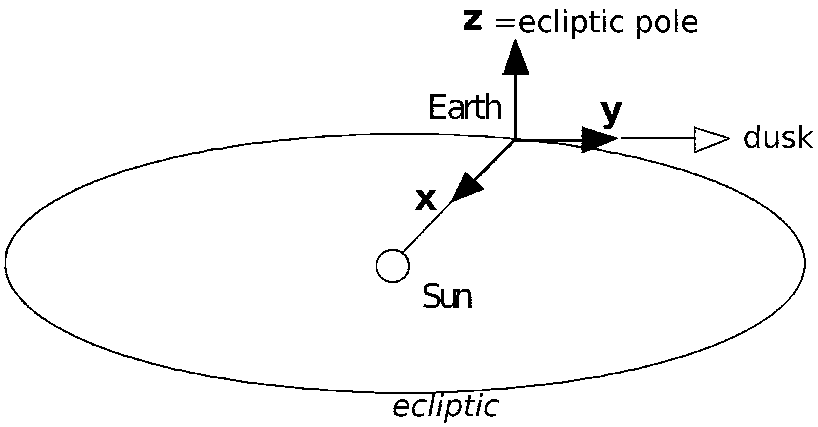


Figure 1.4: GSE system definition

The X-axis is pointing from the Earth towards the Sun.
The X-axis and the Y-axis are include in the ecliptic plane.
The Y-axis is pointing toward the dusk, opposing to the planetary motion.

The Z-axis is parallel to the ecliptic pole. The GSE system has a yearly rotation with respect to the inertial system.

1.5 Geocentric Solar Equatorial system (GSEQ)

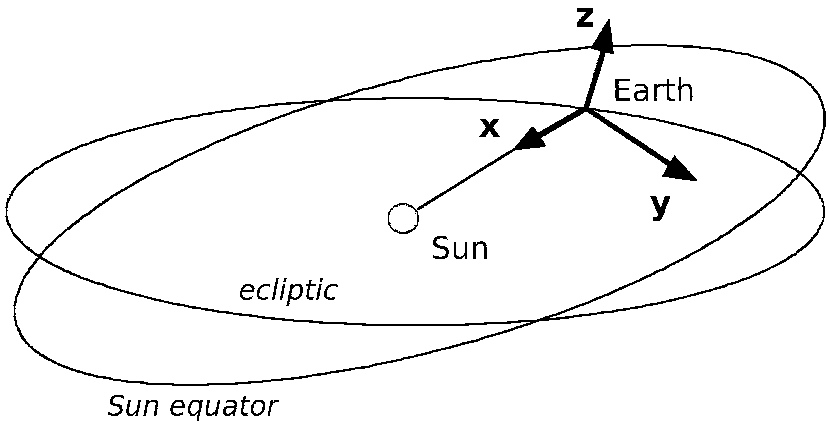


Figure 1.5: GSEQ system definition

The X-axis is pointing toward the Sun and include in the ecliptic plane.
The Y-axis is parallel to the Sun's equatorial plane (inclined to the ecliptic)

X-axis is not necessarily in the Sun's equatorial plane;
Z-axis is not necessarily be parallel to the Sun's axis of rotation (which is perpendicular to Y, and thus in the X-Z plane);
Z-axis is chosen to be in the same sense as the ecliptic pole, i.e. northwards.

1.6 Geocentric Solar Magnetospheric system (GSM)

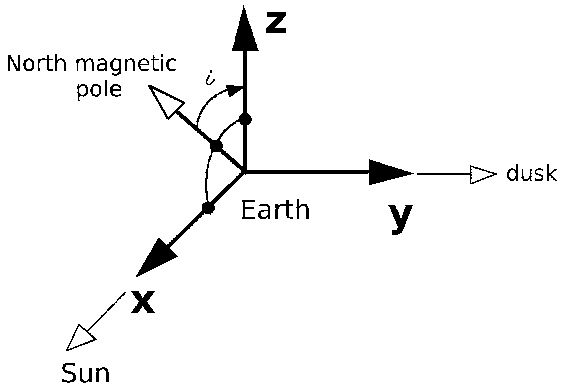


Figure 1.6: GSM system definition

The X-axis is pointing from the Earth towards the Sun.
The X-Z plane contains the dipole axis.
The Y-axis is perpendicular to the Earth's magnetic dipole, towards the dusk and include in the magnetic equator plane.

The positive Z-axis is chosen to be in the same sense as the northern magnetic pole; the dipole tilt angle i is positive when the north magnetic pole is tilted towards the Sun. In addition to a yearly period due to the motion of the Earth about the Sun, the GSM system rocks about the Solar direction with a 24 h period.

1.7 Solar Magnetic system (SM)

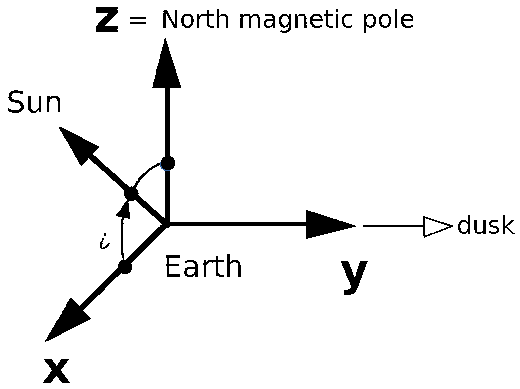


Figure 1.7: SM system definition

The Z-axis is parallel to the North magnetic dipole.
The X-Z plane contains the direction of the Sun.
The Y-axis is perpendicular to the Earth-Sun line toward dusk.

The SM system rotates with both a yearly and a daily period with respect to the inertial system.

1.8 Dipole Meridian system (DM)

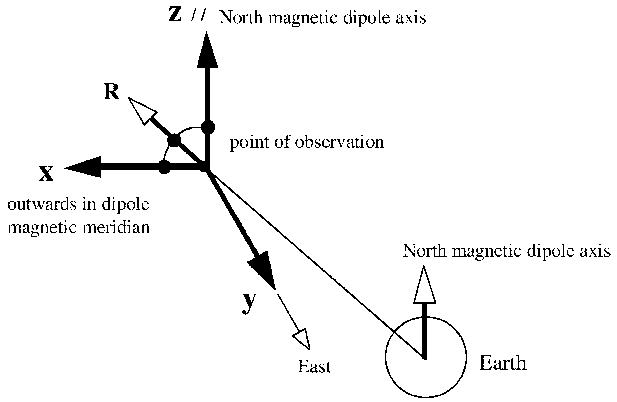


Figure 1.8: DM system definition

The Z-axis is parallel to the North magnetic dipole axis.
The X-Z plane contains the direction $\vec{R}$ of the point of observation, from the Earth, and is a dipole magnetic meridian plane.

The Y-axis is perpendicular to the $\vec{R}$ vector, eastwards.

This system is a local coordinate system, which is dependent of the position of the point of observation from the Earth.

1.9 Vertical Dusk Horizontal system (VDH)

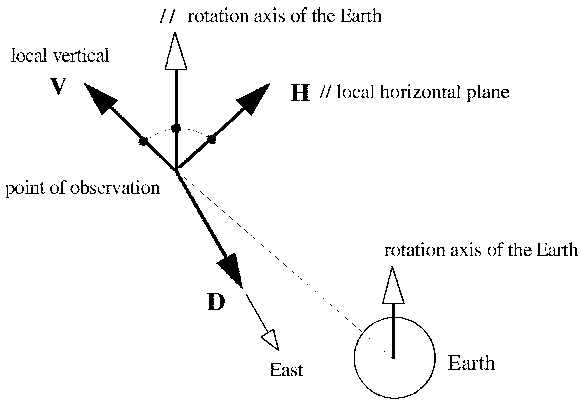


Figure 1.9: VDH system definition

The V-axis is the outwards local vertical, to the point of observation.
The H-axis is parallel to the horizontal local plane, positive to the North.
The V-H plane is a geographic meridian plane. The D-axis is azimuthal, eastwards.

As DM system, this system is a local coordinate system, which is dependent of the position of the point of observation from the Earth.

1.10 Spin Reference system (SR)

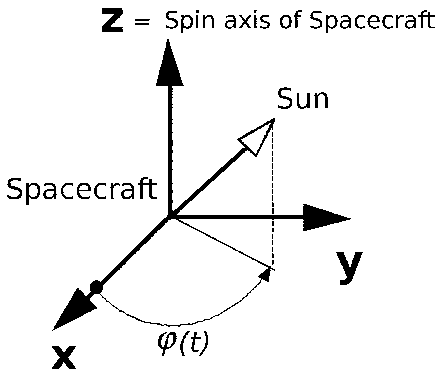


Figure 1.10: SR system definition

This is a spinning local system close to the measurement antenna of a spacecraft.
The Z-axis is the spin axis of the spacecraft.
The X-axis and Y-axis are perpendicular to the spin axis, and rotate at the spin frequency of the spacecraft.

The definition of the SR system need the knowledge of the spin axis in a fixed frame of reference as the GEI inertial system, and the value of the spin phase ϕ at a given time.

1.11 Spin Reference 2 system (SR2)

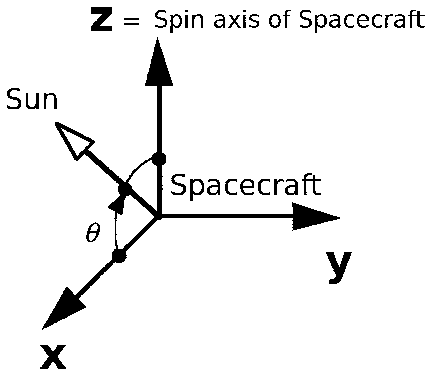


Figure 1.11: SR2 system definition

This is a fixed system usefull for the spacecraft data processing. It is also called SCS, as "Spacecraft-Sun system", or DS system (Despun Satellite).
The Z-axis is the spin axis of the spacecraft.
The X-Z plane contains the direction of the Sun.
The X-axis is towards the day side.
The Y-axis is perpendicular to the spacecraft-Sun line.

The SR2 system rotates with the same period than the orbital period of the spacecraft with respect to the inertial system, while the declination θ varies continously.

1.12 Inverse SR2 system (ISR2)

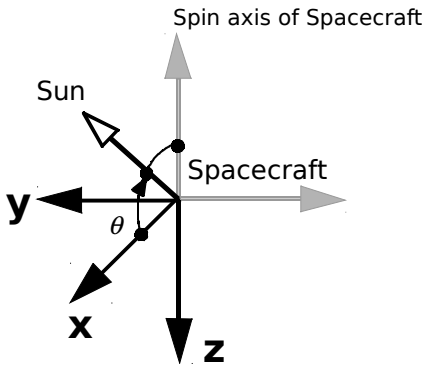


Figure 1.12: ISR2 system definition

This system is equivalent to the SR2 system where the Z and Y axis has inverse sign. This system is useful for CLUSTER, where the Z axis of ISR2 system is close to the Z axis of the GSE system, so ISR2 is a rather good approximation of the GSE system, and does not requires knowledge of spin direction in GSE system.

1.13 Magnetic Field Aligned system (MFA)

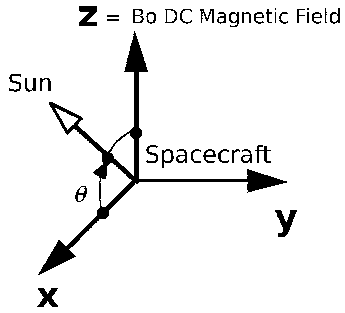


Figure 1.13: MFA system definition

This is a system useful for physic, but the meaning of the Bo DC magnetic filed must be knew, as its time variation (see ref. [2]).
The Z-axis is the DC magnetic field vector.
The X-Z plane contains the direction of the Sun.
The X-axis is towards the day side.
The Y-axis is perpendicular to the spacecraft-Sun line.

The MFA system move continuously with the time variation of the DC magnetic field (see [3]).

1.14 Tangent Paraboloid Normal system (TPN)

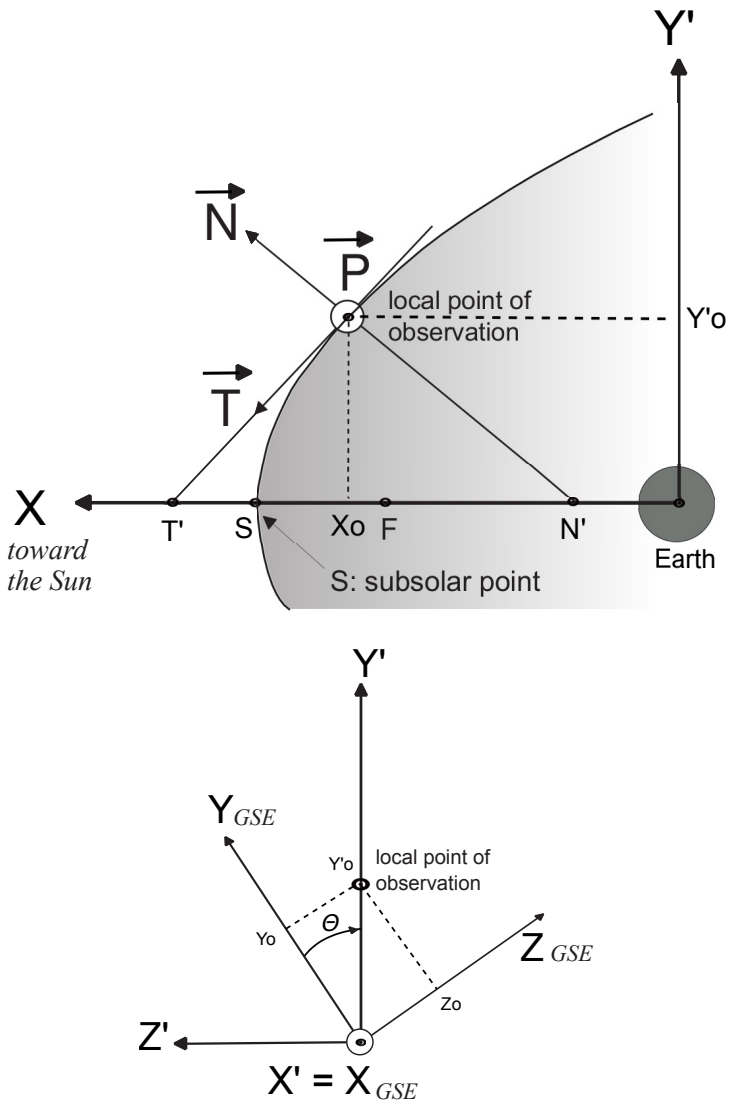


Figure 1.14: TPN system definition

The TPN system has been introduced by the author in 2005, for magnetopause studies. When the spacecraft cross the magnetopause, this one is modelized by a paraboloide around the Sun-Earth direction. The $\vec{N}$ vector is the output normal to the plane tengant to the paraboloid, at the local point of observation. The $\vec{T}$ vector is the tangent in the X-N plane, toward the submit of the parabole, while the $\vec{P}$ vector is the tangent perpendicular to the X-N or T-N plane, in the direct sens $T, \vec{P}, N$.

When magnetopause crossing data are analysed by the Minimum Variance Analysis method, the $\vec{N}$ vector of the TPN system is often close the MVA eigen vector corresponding to the minimum eigenvalue.

Chapter 2

DIAGRAM OF TRANSFORMATIONS

2.1 General remarks

Among all coordinates systems described in section I, we have single out two particular systems more frequently encountered: the Geographic system (GEO) and the Geocentric Equatorial Inertial system (GEI). From these two system, we have computed all transformations to directly convert theses coordinates into any else other. Similarly GSE system has been added in version 3.0 as a starting point for other coordinates. Furthermore, other direct "circular" transformations between the other system are also given, as explained on the schematic diagram below.

2.2 Schematic diagram of transformations

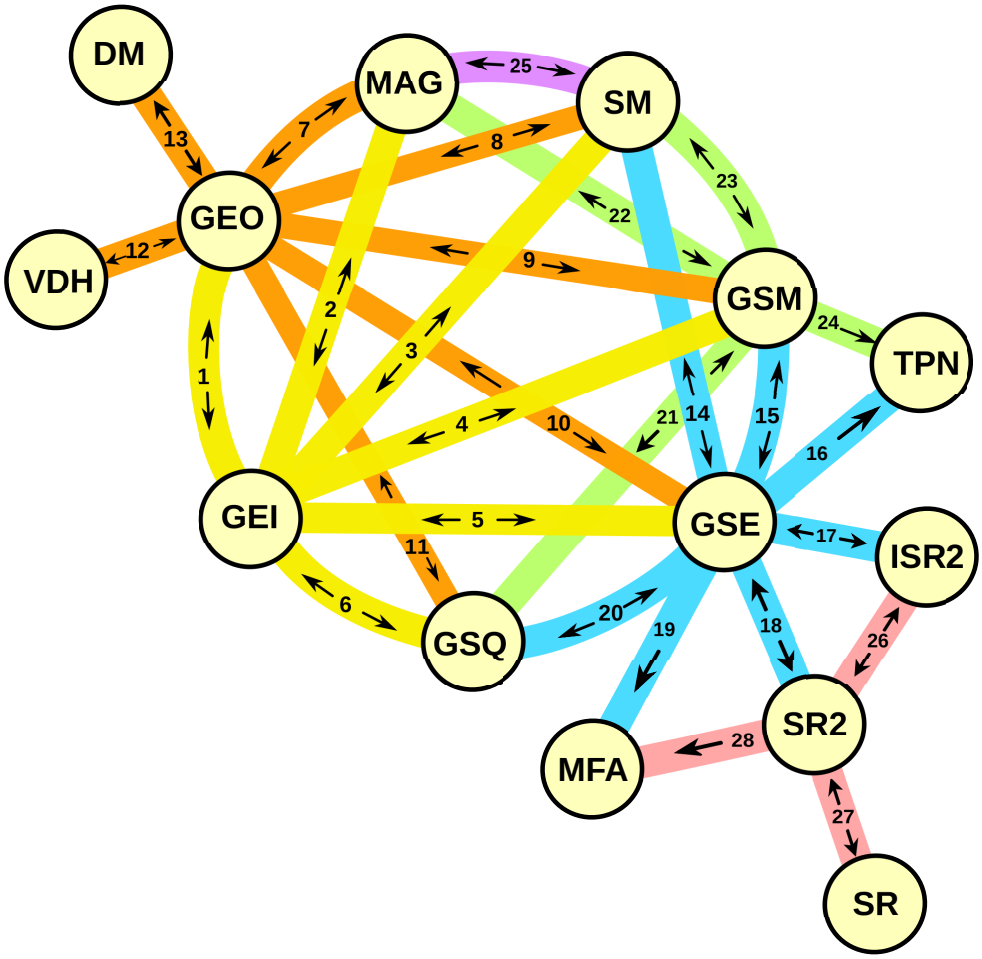


Figure 2.1: Diagram of available transformations

2.3 List of transformations

All different coordinate transformations are listed below; number correspond to number given in the above schematic diagram, and detailled in the same order in section 3. Name corresponds to the corresponding subroutines described in Part II.

AROUND GEI

1)	t_gei_geo	Geocentric Equatorial Inertial	⟶	Geographical
	t_geo_gei	Geographical	⟶	Geocentric Equatorial Inertial
2)	t_gei_mag	Geocentric Equatorial Inertial	⟶	Magnetic dipole
	t_mag_gei	Magnetic dipole	⟶	Geocentric Equatorial Inertial
3)	t_gei_sm	Geocentric Equatorial Inertial	⟶	Solar Magnetic
	t_sm_gei	Solar Magnetic	⟶	Geocentric Equatorial Inertial
4)	t_gei_gsm	Geocentric Equatorial Inertial	⟶	Geocentric Solar Magnetospheric
	t_gsm_gei	Geocentric Solar Magnetospheric	⟶	Geocentric Equatorial Inertial
5)	t_gei_gse	Geocentric Equatorial Inertial	⟶	Geocentric Solar Ecliptic
	t_gse_gei	Geocentric Solar Ecliptic	⟶	Geocentric Equatorial Inertial
6)	t_gei_gseq	Geocentric Equatorial Inertial	⟶	Geocentric Solar Equatorial
	t_gseq_gei	Geocentric Solar Equatorial	⟶	Geocentric Equatorial Inertial

AROUND GEO

7)	t_geo_mag	Geographical	⟶	Magnetic dipole
	t_mag_geo	Magnetic dipole	⟶	Geographical
8)	t_geo_sm	Geographical	⟶	Solar Magnetic
	t_sm_geo	Solar Magnetic	⟶	Geographical
9)	t_geo_gsm	Geographical	⟶	Geocentric Solar Magnetospheric
	t_gsm_geo	Geocentric Solar Magnetospheric	⟶	Geographical
10)	t_geo_gse	Geographical	⟶	Geocentric Solar Ecliptic
	t_gse_geo	Geocentric Solar Ecliptic	⟶	Geographical
11)	t_geo_gseq	Geographical	⟶	Geocentric Solar Equatorial
	t_gseq_geo	Geocentric Solar Equatorial	⟶	Geographical
12)	t_geo_vdh	Geographical	⟶	Vertical Dusk Horizontal
	t_vdh_geo	Vertical Dusk Horizontal	⟶	Geographical
13)	t_geo_dm	Geographical	⟶	Dipole Meridian
	t_dm_geo	Dipole Meridian	⟶	Geographical

AROUND GSE

14)	t_gse_sm	Geocentric Solar Ecliptic	⟶	Solar Magnetic
	t_sm_gse	Solar Magnetic	⟶	Geocentric Solar Ecliptic
15)	t_gse_gsm	Geocentric Solar Ecliptic	⟶	Geocentric Solar Magnetospheric
	t_gsm_gse	Geocentric Solar Magnetospheric	⟶	Geocentric Solar Ecliptic
16)	t_gse_tpn	Geocentric Solar Ecliptic	⟶	Tangent Paraboloid Normal
17)	t_gse_isr2	Geocentric Solar Ecliptic	⟶	Inverse SR2
	t_isr2_gse	Inverse SR2 system	⟶	Geocentric Solar Ecliptic
18)	t_gse_sr2	Geocentric Solar Ecliptic	⟶	Spin Reference 2
	t_sr2_gse	Spin Reference 2	⟶	Geocentric Solar Ecliptic
19)	t_gse_mfa	Geocentric Solar Ecliptic	⟶	Magnetic Field Aligned
20)	t_gse_gseq	Geocentric Solar Ecliptic	⟶	Geocentric Solar Equatorial
	t_gseq_gse	Geocentric Solar Equatorial	⟶	Geocentric Solar Ecliptic

AROUND GSM

21)	t_gsm_gseq	Geocentric Solar Magnetospheric	⟶	Geocentric Solar Equatorial
	t_gseq_gsm	Geocentric Solar Equatorial	⟶	Geocentric Solar Magnetospheric
22)	t_gsm_mag	Geocentric Solar Magnetospheric	⟶	Magnetic dipole
	t_mag_gsm	Magnetic dipole	⟶	Geocentric Solar Magnetospheric
23)	t_gsm_sm	Geocentric Solar Magnetospheric	⟶	Solar Magnetic
	t_sm_gsm	Solar Magnetic	⟶	Geocentric Solar Magnetospheric
24)	t_gsm_tpn	Geocentric Solar Magnetospheric	⟶	Tangent Paraboloid Normal

AROUND SM

25)	t_sm_to_mag	Solar Magnetic	⟶	Magnetic dipole
	t_mag_to_sm	Magnetic dipole	⟶	Solar Magnetic

AROUND SR2

26)	t_sr2_isr2	Spin Reference 2	⟶	Inverse SR2
	t_isr2_sr2	Inverse SR2	⟶	Spin Reference 2
27)	t_sr2_sr	Spin Reference 2	⟶	Spin Reference
	t_sr_sr2	Spin Reference	⟶	Spin Reference 2
28)	t_sr2_mfa	Spin Reference 2 (SR2)	⟶	Magnetic Field Aligned

Chapter 3

MATHEMATICAL FORM OF TRANSFORMATIONS

3.1 General remarks on transformation matrix

To obtain the matrix to transform a vector expressed in a coordinate system A into another system B, the simplest way is to express the directions of the 3 axis of system B in the coordinate system A.

Indeed, if we notes the coordinates of these 3 unit axes $\vec{X}_B$, $\vec{Y}_B$, $\vec{Z}_B$ in coordinate system A as:

$$\vec{X}_B = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}_{(A)} \quad \vec{Y}_B = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}_{(A)} \quad \vec{Z}_B = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix}_{(A)}$$

Any vector $\vec{V}$ can be expressed in system B as:

$$V_1^{(B)} = \vec{X}_B \cdot \vec{V} = X_1^{(A)} V_1^{(A)} + X_2^{(A)} V_2^{(A)} + X_3^{(A)} V_3^{(A)} \tag{3.1}$$

$$V_2^{(B)} = \vec{Y}_B \cdot \vec{V} = Y_1^{(A)} V_1^{(A)} + Y_2^{(A)} V_2^{(A)} + Y_3^{(A)} V_3^{(A)} \tag{3.2}$$

$$V_3^{(B)} = \vec{Z}_B \cdot \vec{V} = Z_1^{(A)} V_1^{(A)} + Z_2^{(A)} V_2^{(A)} + Z_3^{(A)} V_3^{(A)} \tag{3.3}$$

Thus the transform matrix of any vector $\vec{V}$ is:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(B)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(A)}}$$

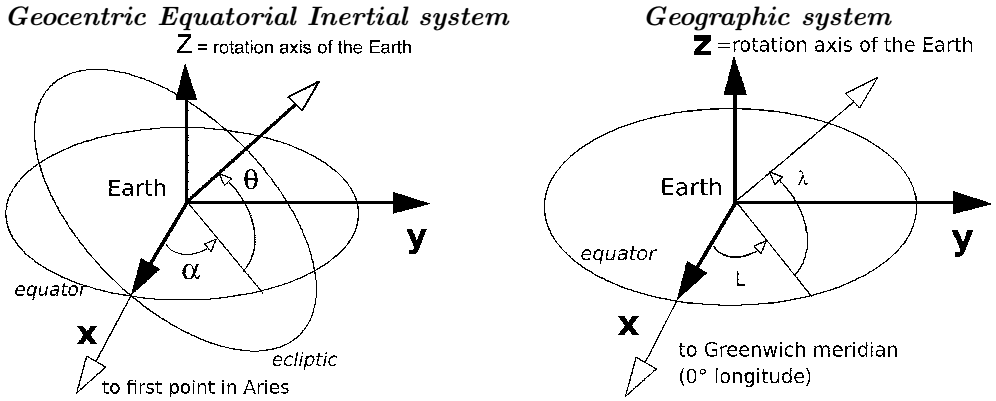
Similarly the transformation from system B to A is:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(A)} = \begin{pmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(B)}}$$

All transformation matrix have the following properties, useful for error checking:

1) Each row and column is a unit vector;
2) The dot products of any two rows or any two columns is zero;
3) The cross product of any two rows or columns equals the third row or column or its negative (Row 1 cross row 2 equals row3; row 2 cross row 1 equal minus row3).

3.2 GEI to GEO transformation



GEO and GEI system have their Z-axis in common, so the only difference is a rotation around Z-axis of θ angle, thus the matrix transformation is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

and the inverse transformation is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

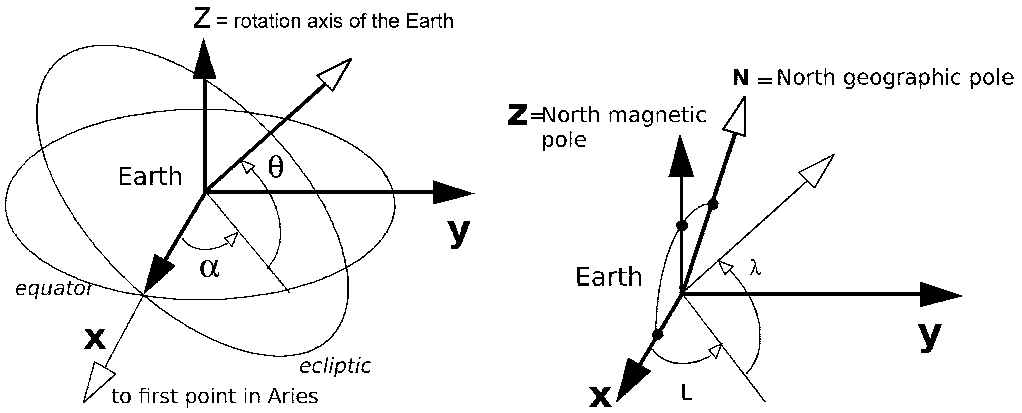
The θ angle is the angle between the Greenwich meridian and the first point in Aries, measured Eastward, in the Earth's equator, from the first point in Aries.

θ is called Greenwich Mean Sideral Time; GMST is a function of the time of the day and the time of year, since the sideral day (duration of a day relative to inertial space) is less than 24h.

Practically, GMST is computed from ***cp_gei_sun_dir*** subroutine.

3.3 GEI to MAG transformation

Geocentric Equatorial Inertial ***Geomagnetic system***



Transformation from GEI to MAG system requires a knowledge of the dipole direction in GEI system, noted as $\vec{M}$.

The geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude, thus:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

Practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine.

To know the $\vec{M} = \vec{D}$ vector in GEI system, which is the Z-axis of MAG system, we use the GEO to GEI transformation computed §3.2, so:

$$\vec{Z} = \vec{M} = \vec{D}_{GEI} = \begin{pmatrix} D_1 \cos\theta - D_2 \sin\theta \\ D_1 \sin\theta + D_2 \cos\theta \\ D_3 \end{pmatrix}$$

where θ is the Greenwich Mean Sideral Time computed from ***cp_gei_sun_dir*** subroutine.

We can deduce then the Y-axis of SM system in GEI coordinates from the cross product between North geographic pole ($\vec{N}$) and North magnetic pole ($\vec{M}$) :

$$\vec{Y} = \frac{\vec{N} \times \vec{M}}{|\vec{N} \times \vec{M}|}$$

Normalizing factors occurs because $\vec{N}$ and $\vec{M}$ are not necessarily perpendicular; since $\vec{N}$ has two components equal to zero, cross product is easy and we found:

$$\vec{Y} = \frac{1}{\left(M_1^2 + M_2^2\right)^{1/2}} \cdot \begin{pmatrix} -M_1 \\ M_2 \\ 0 \end{pmatrix}$$

X-axis is deduced from: $\vec{X} = \vec{Y} \times \vec{M}$, so:

$$\vec{X} = \frac{1}{\left(M_1^2 + M_2^2\right)^{1/2}} \cdot \begin{pmatrix} M_1 M_3 \\ M_2 M_3 \\ -\left(M_1^2 + M_2^2\right) \end{pmatrix}$$

All coordinates of X-Y-Z axis of MAG system in GEI coordinates being known, the transform matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ M_1 & M_2 & M_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

Similarly the transformation from system MAG to GEI is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} X_1 & Y_1 & M_1 \\ X_2 & Y_2 & M_2 \\ X_3 & Y_3 & M_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)}$$

By replacing corresponding values, with $Q = \left(M_1^2 + M_2^2\right)^{1/2}$, we obtain the expression of all terms of the transformation matrix:

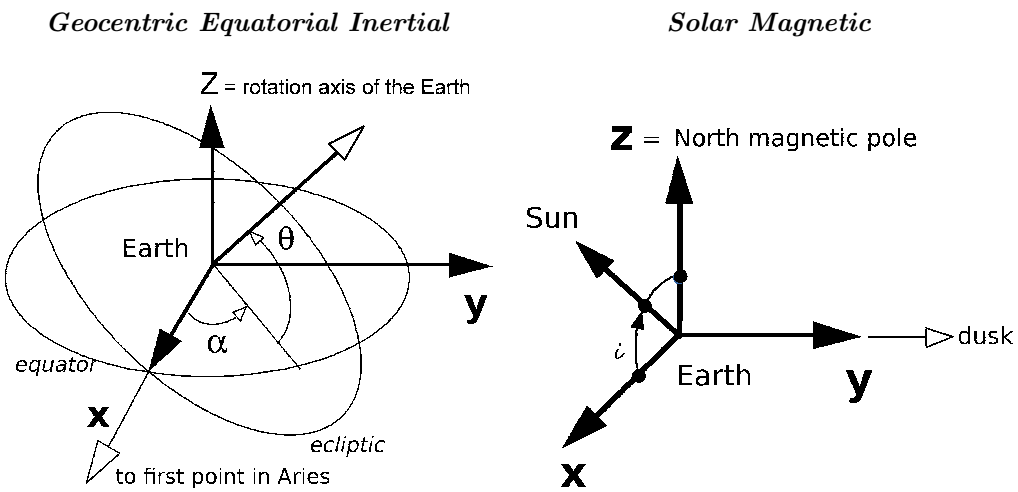
$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} (D_1 D_3 \cos\theta - D_2 D_3 \sin\theta)/Q \\ (D_1 D_3 \sin\theta + D_2 D_3 \cos\theta)/Q \\ -Q \end{pmatrix}$$

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} -(D_2 \cos\theta + D_1 \sin\theta)/Q \\ (-D_2 \sin\theta + D_1 \cos\theta)/Q \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} D_1 \cos\theta - D_2 \sin\theta \\ D_1 \sin\theta + D_2 \cos\theta \\ D_3 \end{pmatrix}$$

Remember that θ is the Greenwich Mean Sideral Time computed from ***cp_gei_sun_dir*** subroutine, and $\vec{D}$ is the direction of the dipole axis in the geographic system. $\vec{M}$ is the direction of the dipole axis in the GEI system, allowing computation of the Q factor.

3.4 GEI to SM transformation



Transformation from GEI system to SM system requires a knowledge of Sun direction and magnetic dipole direction in GEI system.

In GEI system, the direction of the Sun is computed from ***cp_gei_sun_dir*** subroutine:

$$\vec{S} = (S_1, S_2, S_3)$$

The geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude, thus:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

Practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine.

To know the $\vec{M} = \vec{D}$ vector in GEI system, which is the Z-axis of SM system, we use the GEO to GEI transformation computed §3.2, so:

$$\vec{Z} = \vec{M} = \vec{D}_{GEI} = \begin{pmatrix} D_1 \cos\theta - D_2 \sin\theta \\ D_1 \sin\theta + D_2 \cos\theta \\ D_3 \end{pmatrix}$$

where θ is the Greenwich Mean Sideral Time computed from ***cp_gei_sun_dir*** subroutine.

We can deduce then the Y-axis of SM system in GEI coordinates as:

$$\vec{Y} = \frac{\vec{M} \times \vec{S}}{|\vec{M} \times \vec{S}|}$$

Normalizing factors occurs because $\vec{M}$ and $\vec{S}$ are not necessarily perpendicular.

X-axis is deduced from: $\vec{X} = \vec{Y} \times \vec{M}$

All coordinates of X-Y-Z axis of SM system in GEI coordinates being known, the transform matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

Similarly the transformation from system SM to GEI is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

with respectively:

$$\begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix}$$

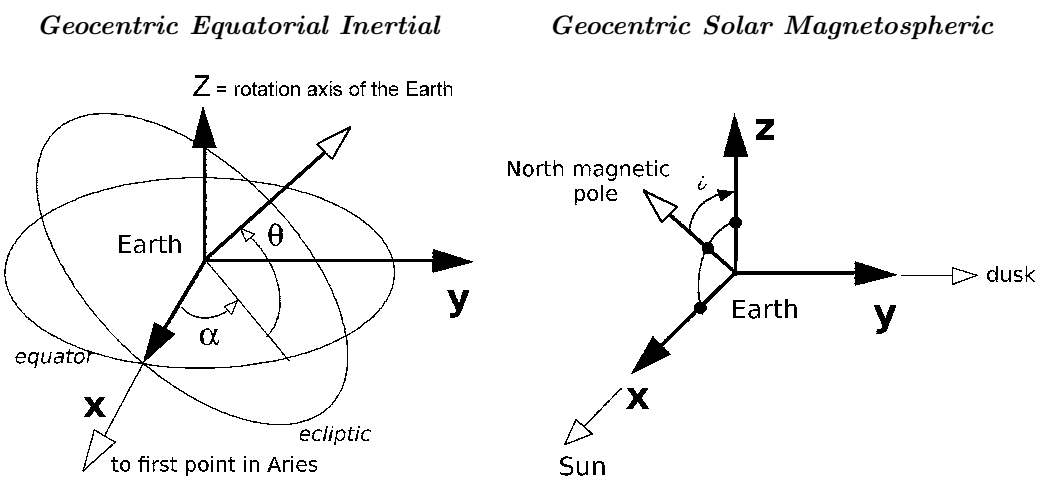
$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \frac{1}{Q} \cdot \begin{pmatrix} M_2S_3 - M_3S_2 \\ M_3S_1 - M_1S_3 \\ M_1S_2 - M_2S_1 \end{pmatrix}$$

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_2M_3 - Y_3M_2 \\ Y_3M_1 - Y_1M_3 \\ Y_1M_2 - Y_2M_1 \end{pmatrix}$$

$$\begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} D_1\cos\theta - D_2\sin\theta \\ D_1\cos\theta + D_2\sin\theta \\ D_3 \end{pmatrix}$$

with $Q = \left[(M_2S_3 - M_3S_2)^2 + (M_3S_1 - M_1S_3)^2 + (M_1S_2 - M_2S_1)^2\right]^{1/2}$

3.5 GEI to GSM transformation



Transformation from GEI system to GSM system requires a knowledge of Sun direction and magnetic dipole direction in GEI system.

In GEI system, the direction of X-axis of GSM system is the direction of the Sun computed from ***cp_gei_sun_dir*** subroutine:

$$\vec{X} = \vec{S} = (S_1, S_2, S_3)$$

The geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude, thus:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

Practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine.

To know the $\vec{M} = \vec{D}$ vector in GEI system, we use the GEO to GEI transformation computed in §3.2, so:

$$\vec{M} = \vec{D}_{GEI} = \begin{pmatrix} D_1 \cos\theta - D_2 \sin\theta \\ D_1 \sin\theta + D_2 \cos\theta \\ D_3 \end{pmatrix}$$

where θ is the Greenwich Mean Sideral Time computed from ***cp_gei_sun_dir*** subroutine.

We can deduce then the Y-axis and Z-axis of GSM system in GEI coordinates as:

$$\vec{Y} = \frac{\vec{M} \times \vec{S}}{|\vec{M} \times \vec{S}|}$$

Normalizing factors occurs because $\vec{M}$ and $\vec{S}$ are not necessarily perpendicular, and Z is obtained by:

$$\vec{Z} = \vec{S} \times \vec{Y}$$

All coordinates of X-Y-Z axis of GSM system in GEI coordinates being known, the transform matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} S_1 & S_2 & S_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

Similarly the transformation from system GSM to GEI is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} S_1 & Y_1 & Z_1 \\ S_2 & Y_2 & Z_2 \\ S_3 & Y_3 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

with respectively:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \frac{1}{Q} \cdot \begin{pmatrix} M_2S_3 - M_3S_2 \\ M_3S_1 - M_1S_3 \\ M_1S_2 - M_2S_1 \end{pmatrix}$$

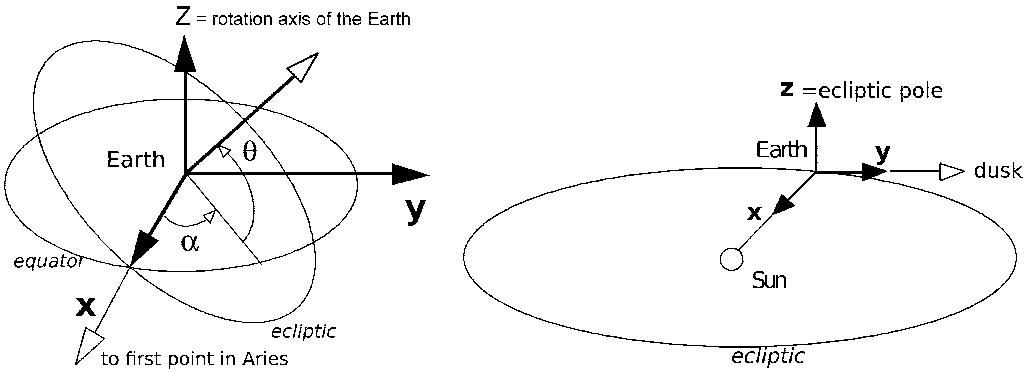
$$\begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \begin{pmatrix} S_2Y_3 - S_3Y_2 \\ S_3Y_1 - S_1Y_3 \\ S_1Y_2 - S_2Y_1 \end{pmatrix}$$

$$\begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} D_1\cos\theta - D_2\sin\theta \\ D_1\sin\theta + D_2\cos\theta \\ D_3 \end{pmatrix}$$

with $Q = \left[(M_2S_3 - M_3S_2)^2 + (M_3S_1 - M_1S_3)^2 + (M_1S_2 - M_2S_1)^2\right]^{1/2}$

3.6 GEI to GSE transformation

Geocentric Equatorial Inertial ***Geocentric Solar Ecliptic***



In GEI system, the direction of X-axis of GSE system is the direction $\vec{S}$ of the SUN , computed from ***cp_gei_sun_dir*** subroutine:

$$\vec{X} = \vec{S} = (S_1, S_2, S_3)$$

The direction of the Z-axis of GSE is the direction of ecliptic pole, which is a known constant value:

$$\vec{Z} = \vec{E} = (E_1, E_2, E_3) = (0, -0.398, 0.917)$$

The third axis, $\vec{Y}$ is deduced from $\vec{Y} = \vec{Z} \times \vec{X} = \vec{E} \times \vec{S}$, thus:

$$\vec{Y} = \vec{E} \times \vec{S} = \begin{pmatrix} E_2 S_3 - S_3 E_2 \\ E_3 S_1 - E_1 S_3 \\ E_1 S_2 - E_2 S_1 \end{pmatrix}$$

Thus the transform matrix of any vector $\vec{V}$ is:

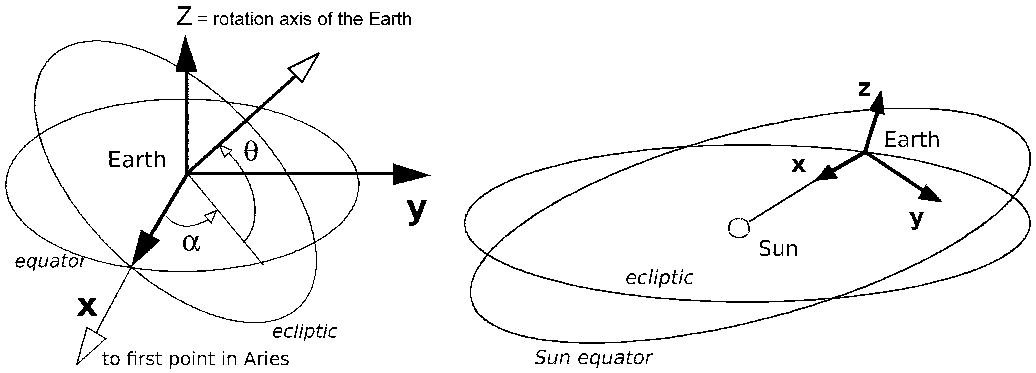
$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} S_1 & S_2 & S_3 \\ Y_1 & Y_2 & Y_3 \\ E_1 & E_2 & E_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

Similarly the transformation from system GSE to GEI is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} S_1 & Y_1 & E_1 \\ S_2 & Y_2 & E_2 \\ S_3 & Y_3 & E_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)}$$

3.7 GEI to GSEQ transformation

Geocentric Equatorial Inertial ***Geocentric Solar Equatorial***



In GEI system, the direction of X-axis of GSEQ system is the direction of the SUN computed from ***cp_gei_sun_dir*** subroutine:

$$\vec{X} = \vec{S} = (S_1, S_2, S_3)$$

The direction of the rotation axis of the SUN in GEI system is a known constant value:

$$\vec{R} = (R_1, R_2, R_3) = (0.122, -0.424, 0.899)$$

Since Y-axis of GSEQ is parallel to the Sun's equatorial plane, the direction of Y-axis in GEI is $\vec{R} \times \vec{S}$; nevertheless the cross product must be normalized to have a Y unit axis, because $\vec{R}$ and $\vec{S}$ are not necessarily perpendicular, so:

$$\vec{Y} = \frac{\vec{R} \times \vec{S}}{|\vec{R} \times \vec{S}|}$$

$$\text{and } \vec{Z} = \vec{S} \times \vec{Y}$$

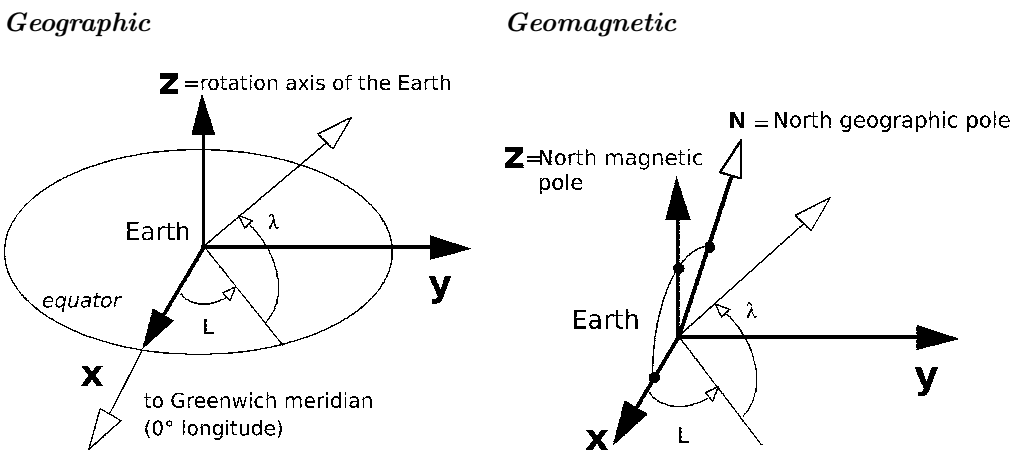
Thus the transform matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSEQ)} = \begin{pmatrix} S_1 & S_2 & S_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

Similarly the transformation from system GSEQ to GEI is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} S_1 & Y_1 & Z_1 \\ S_2 & Y_2 & Z_2 \\ S_3 & Y_3 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSEQ)}$$

3.8 GEO to MAG transformation



In geomagnetic coordinates, Z-axis is parallel to the magnetic dipole axis D, and the Y-axis is perpendicular to the North geographic pole N, then we have in geographic system:

$$\begin{cases} \vec{Z} = \vec{D} \\ \vec{Y} = (\vec{N} \times \vec{D})/|\vec{N} \times \vec{D}| \\ \vec{X} = \vec{Y} \times \vec{D} \end{cases}$$

The geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude.
Practically, geographic dipole direction $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine; value for 1965.0 is:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

$$\text{we deduce: } \vec{Y} = (-D_2, D_1, 0)/(D_1^2 + D_2^2)^{1/2}$$

$$\text{and: } \vec{X} = (Y_2 D_3, -Y_1 D_3, Y_1 D_2 - Y_2 D_1)$$

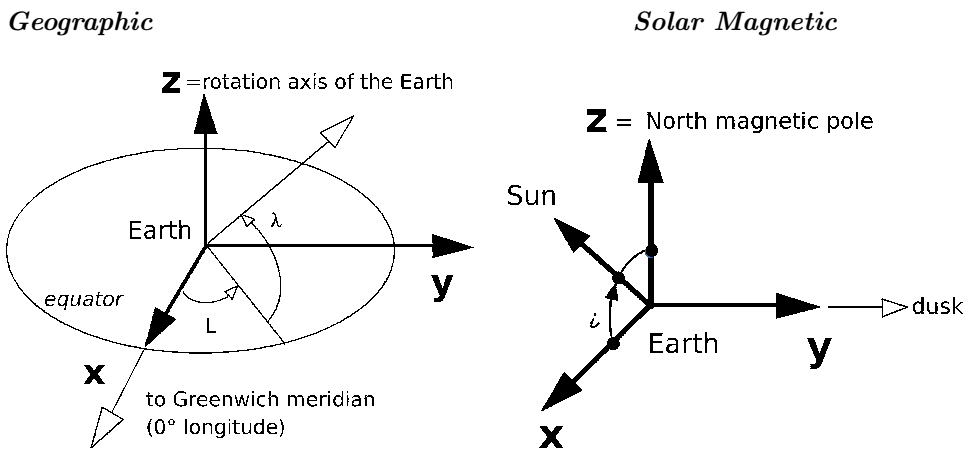
Thus the transform matrix of any vector $\vec{V}$ is:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ D_1 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}}$$

Similarly the transformation from system MAG to GEO is:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} X_1 & Y_1 & D_1 \\ X_2 & Y_2 & D_2 \\ X_3 & Y_3 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)}}$$

3.9 GEO to SM transformation



In GEO system, the direction of the Z-axis of SM system is the dipole direction $\vec{D}$. The geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude, thus:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

Practically, D is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine.

We can then deduce in GEO system the Y-axis of SM system as:

$$\vec{y} = \frac{\vec{D} \times \vec{\mathscr{S}}}{|\vec{D} \times \vec{\mathscr{S}}|}$$

where $\mathscr{S}$ is the direction of the sun in GEO system, which can be computed from GEI to GEO transformation given in §3.2, then we have:

$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

$\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ normalizing factors occurs because $\vec{D}$ and $\vec{S}$ are not necessarily perpendicular.

The third axis X is computed from $\vec{x} = \vec{y} \times \vec{D}$ and the GEO to SM transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ D_1 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

Similarly the SM to GEO transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} x_1 & y_1 & D_1 \\ x_2 & y_2 & D_2 \\ x_3 & y_3 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

with respectively:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} y_2 D_3 - y_3 D_2 \\ y_3 D_1 - y_1 D_3 \\ y_1 D_2 - y_2 D_1 \end{pmatrix}$$

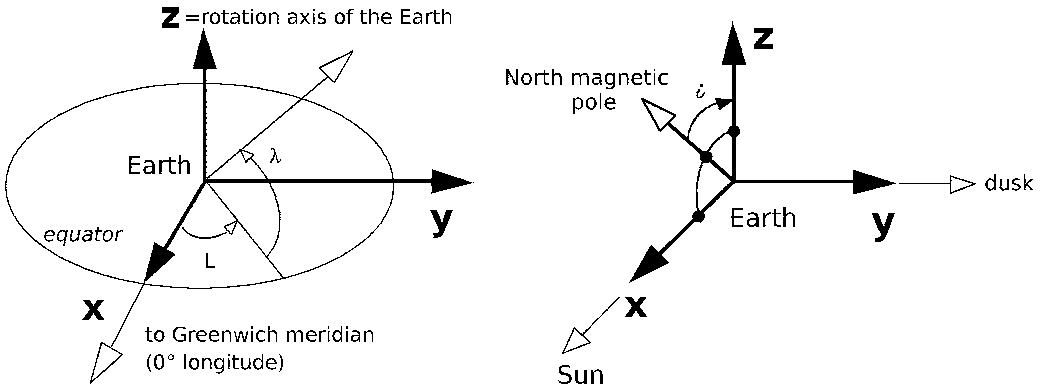
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}_{(GEO)} = \frac{1}{Q} \cdot \begin{pmatrix} D_2 \mathscr{S}_3 - D_3 \mathscr{S}2 \\ D_3 \mathscr{S}_1 - D_1 \mathscr{S}3 \\ D_1 \mathscr{S}_2 - D_2 \mathscr{S}1 \end{pmatrix}$$

$$Q = [(D_2 \mathscr{S}_3 - D_3 \mathscr{S}_2)^2 + (D_3 \mathscr{S}_1 - D_1 \mathscr{S}_3)^2 + (D_1 \mathscr{S}_2 - D_2 \mathscr{S}_1)^2]^{1/2}$$

$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

3.10 GEO to GSM transformation

Geographic ***Geocentric Solar Magnetospheric***



In GEO system, the direction of the X-axis of GSM system is the direction of the sun $\mathscr{S}$, which can be computed from GEI to GEO transformation given in §3.2; then we have:

$$\vec{x} = \vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

$\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ.

The Y axis in GEO system can be deduced from:

$$y = \frac{\vec{D} \times \vec{\mathscr{S}}}{|\vec{D} \times \vec{\mathscr{S}}|}$$

where $\vec{D}$ is the dipole direction; the geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude, thus:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

Practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine; normalizing factors in cross product occurs because $\vec{D}$ and $\vec{\mathscr{S}}$ are not necessarily perpendicular.

Finally the third axis Z is computed from $z = \vec{\mathscr{S}} \times \vec{y}$ and the GEO to GSM transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

Similarly the GSM to GEO transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

with respectively:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} \mathscr{S}_1 \\ \mathscr{S}_2 \\ \mathscr{S}_3 \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}_{(GEO)} = \frac{1}{Q} \cdot \begin{pmatrix} D_2\mathscr{S}_3 - D_3\mathscr{S}_2 \\ D_3\mathscr{S}_1 - D_1\mathscr{S}_3 \\ D_1\mathscr{S}_2 - D_2\mathscr{S}_1 \end{pmatrix}$$

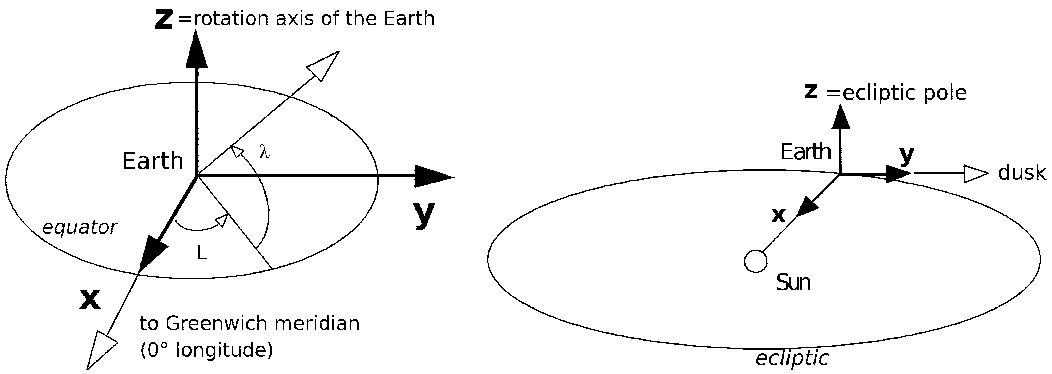
$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} x_2y_3 - x_3y_2 \\ x_3y_1 - x_1y_3 \\ x_1y_2 - x_2y_1 \end{pmatrix}$$

$$Q = [(D_2\mathscr{S}_3 - D_3\mathscr{S}_2)^2 + (D_3\mathscr{S}_1 - \mathscr{S}_1D_3)^2 + (D_1\mathscr{S}_2 - D_2\mathscr{S}_1)^2]^{1/2}$$

$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1\cos\theta + S_2\sin\theta), (-S_1\sin\theta + S_2\cos\theta), S_3)$$

3.11 GEO to GSE transformation

Geographic ***Geocentric Solar Ecliptic***



In GEO system, the direction of the X-axis of GSE system is the direction of the sun $\vec{\mathscr{S}}$, which can be computed from GEI to GEO transformation given in §3.2; then we have:

$$\vec{X} = \vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

$\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ

The Z axis is the direction of the ecliptic pole, which is a known constant value in GEI system:

$$\vec{E} = (E_1, E_2, E_3) = (0, -0.398, 0.917)$$

From GEI to GEO transformation we have in GEO system:

$$\vec{Z} = \vec{\mathscr{E}} = (\mathscr{E}_1, \mathscr{E}_2, \mathscr{E}_3) = ((E_1 \cos\theta + E_2 \sin\theta), (-E_1 \sin\theta + E_2 \cos\theta), E_3)$$

And finally the Y axis in GEO system can be deduced from:

$$\vec{Y} = \vec{\mathscr{E}} \times \vec{\mathscr{S}}$$

and the GEO to GSE transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(G\mathscr{S}E)} = \begin{pmatrix} \mathscr{S}_1 & \mathscr{S}_2 & \mathscr{S}_3 \\ Y_1 & Y_2 & Y_3 \\ \mathscr{E}_1 & \mathscr{E}_2 & \mathscr{E}_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

Similarly the GSE to GEO transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} \mathscr{S}_1 & Y_1 & \mathscr{E}_1 \\ \mathscr{S}_2 & Y_2 & \mathscr{E}_2 \\ \mathscr{S}_3 & Y_3 & \mathscr{E}_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)}$$

with respectively:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} \mathscr{E}_2\mathscr{S}_3 - \mathscr{E}_3\mathscr{S}_2 \\ \mathscr{E}_3\mathscr{S}_1 - \mathscr{E}_1\mathscr{S}_3 \\ \mathscr{E}_1\mathscr{S}_2 - \mathscr{E}_2\mathscr{S}_1 \end{pmatrix}$$

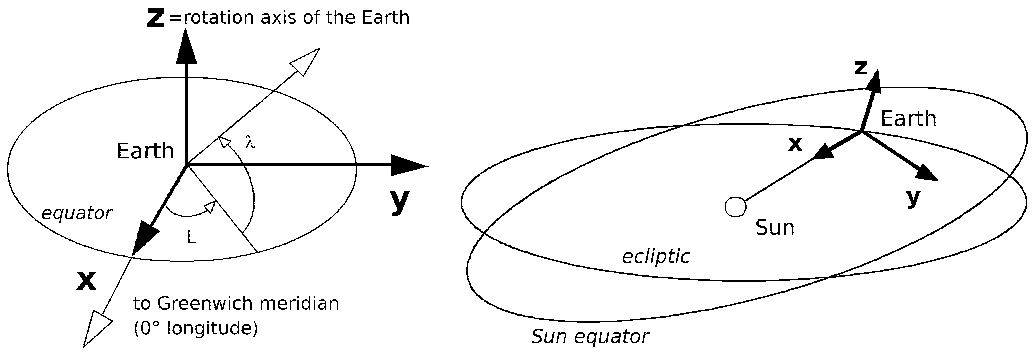
$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

$$\vec{\mathscr{E}} = (\mathscr{E}_1, \mathscr{E}_2, \mathscr{E}_3) = ((E_1 \cos\theta + E_2 \sin\theta), (-E_1 \sin\theta + E_2 \cos\theta), E_3)$$

$$\vec{E} = (E_1, E_2, E_3) = (0, -0.398, 0.917)$$

3.12 GEO to GSEQ transformation

Geographic ***Geocentric Solar Equatorial***



In GEO system, the direction of the X-axis of GSEQ system is the direction of the sun $\vec{\mathscr{S}}$, which can be computed from GEI to GEO transformation given in §3.2; then we have:

$$\vec{x} = \vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

$\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ

The Sun equator plane is defined from the direction of the rotation axis of the SUN which is in GEI system a known constant value:

$$\vec{R} = (R_1, R_2, R_3) = (0.122, -0.424, 0.899)$$

from GEI to GEO transformation we obtain this vector in GEO system as:

$$\vec{\mathscr{R}} = (\mathscr{R}_1, \mathscr{R}_2, \mathscr{R}_3) = ((R_1 \cos\theta + R_2 \sin\theta), (-R_1 \sin\theta + R_2 \cos\theta), R_3)$$

Then the Y axis in GEO system can be deduced from:

$$\vec{y} = \frac{\vec{\mathscr{R}} \times \vec{\mathscr{S}}}{|\vec{\mathscr{R}} \times \vec{\mathscr{S}}|}$$

Normalizing factors in cross product occurs because $\vec{\mathscr{R}}$ and $\vec{\mathscr{S}}$ are not necessarily perpendicular. Finally the third axis Z is computed from: $\vec{z} = \vec{\mathscr{S}} \times \vec{y}$ and the GEO to GSEQ transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSEQ)} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

Similarly the GSEQ to GEO transformation is given by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSEQ)}$$

with respectively:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} \mathscr{S}_1 \\ \mathscr{S}_2 \\ \mathscr{S}_3 \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}_{(GEO)} = \frac{1}{Q} \cdot \begin{pmatrix} \mathscr{R}_2\mathscr{S}_3 - \mathscr{R}_3\mathscr{S}_2 \\ \mathscr{R}_3\mathscr{S}_1 - \mathscr{R}_1\mathscr{S}_3 \\ \mathscr{R}_1\mathscr{S}_2 - \mathscr{R}_2\mathscr{S}_1 \end{pmatrix}$$

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} \mathscr{S}_2 y_3 - \mathscr{S}_3 y_2 \\ \mathscr{S}_3 y_1 - \mathscr{S}_1 y_3 \\ \mathscr{S}_1 y_2 - \mathscr{S}_2 y_1 \end{pmatrix}$$

$$Q = [(\mathscr{R}_2\mathscr{S}_3 - \mathscr{R}_3\mathscr{S}_2)^2 + (\mathscr{R}_3\mathscr{S}_1 - \mathscr{R}_1\mathscr{S}_3)^2 + (\mathscr{R}_1\mathscr{S}_2 - \mathscr{R}_2\mathscr{S}_1)^2]^{1/2}$$

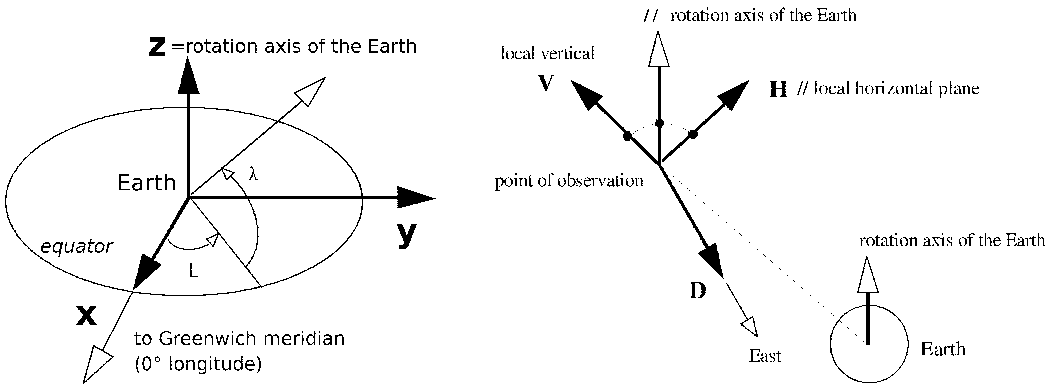
$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = ((S_1 \cos\theta + S_2 \sin\theta), (-S_1 \sin\theta + S_2 \cos\theta), S_3)$$

$$\vec{\mathscr{R}} = (\mathscr{R}_1, \mathscr{R}_2, \mathscr{R}_3) = ((R_1 \cos\theta + R_2 \sin\theta), (-R_1 \sin\theta + R_2 \cos\theta), R_3)$$

$$\vec{R} = (R_1, R_2, R_3) = (0.122, -0.424, 0.899)$$

3.13 GEO to VDH transformation

Geographic ***VDH***



VDH system is a local coordinate system, and varies with the position of the point of observation relative to the centre of the Earth; this positions is noted in GEO system as:

$$\vec{R} = (R_1, R_2, R_3)$$

and we have directly : $\vec{V} = \dfrac{1}{(R_1^2 + R_2^2 + R_3^2)^{1/2}} \cdot \vec{R}$

Since $\vec{D}$ is perpendicular to $\vec{V}$ and to the rotation axis of the Earth, we have:

$$\vec{D} = \frac{\vec{Z_E} \times \vec{R}}{|\vec{Z_E} \times \vec{R}|} \quad \text{which give:} \quad \vec{D} = \begin{pmatrix} -R_2 \\ R_1 \\ 0 \end{pmatrix} \cdot \frac{1}{(R_1^2 + R_2^2)^{1/2}}$$

The third axis H is deduced from: $\vec{H} = \vec{V} \times \vec{D}$ which give: $\vec{H} = \frac{1}{Q} \cdot \begin{pmatrix} -R_1 R_3 \\ -R_2 R_3 \\ R_1^2 + R_2^2 \end{pmatrix}$

with : $Q = [(R_1^2 + R_2^2)(R_1^2 + R_2^2 + R_3^2)]^{1/2}$

All coordinates of X-Y-Z axis of VDH system in GEO coordinates being known, the transform matrix of any vector V is:

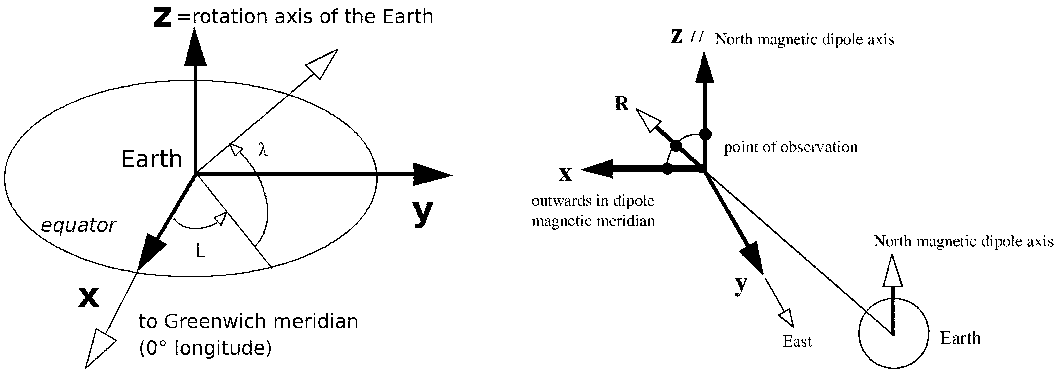
$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(VDH)} = \begin{pmatrix} V_1 & V_2 & V_3 \\ D_1 & D_2 & D_3 \\ H_1 & H_2 & H_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

Similarly the transformation from system VDH to GEO is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} V_1 & D_1 & H_1 \\ V_2 & D_2 & H_2 \\ V_3 & D_3 & H_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(VDH)}$$

3.14 GEO to DM transformation

Geographic ***Dipole Meridian***



Dipole meridian system is a local coordinate system, and varies with the position of the point of observation relative to the centre of the Earth; this positions is noted in GEO system as:

$$\vec{R} = (R_1, R_2, R_3)$$

To transform GEO coordinate to DM coordinates, we need the dipole position in GEO system which is the Z axis of DM system. The geographic coordinates of the dipole axis can be known, for instance for IGRF epoch 1965, as 11.435° colatitude and -69.761° east longitude, thus:

$$\vec{Z} = \vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

Practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine.

We can deduce then the Y-axis of DM system in GEO coordinates as:

$$\vec{Y} = \frac{\vec{D} \times \vec{R}}{|\vec{D} \times \vec{R}|}$$

Normalizing factors occurs because $\vec{D}$ and $\vec{R}$ are not necessarily perpendicular and because $\vec{R}$ is not a unit vector. The third axis X is deduced from: $\vec{X} = \vec{Y} \times \vec{D}$

All coordinates of X-Y-Z axis of DM system in GEO coordinates being known, the transform matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(DM)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ D_1 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)}$$

Similarly the transformation from system DM to GEO is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} X_1 & Y_1 & D_1 \\ X_2 & Y_2 & D_2 \\ X_3 & Y_3 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(DM)}$$

with respectively:

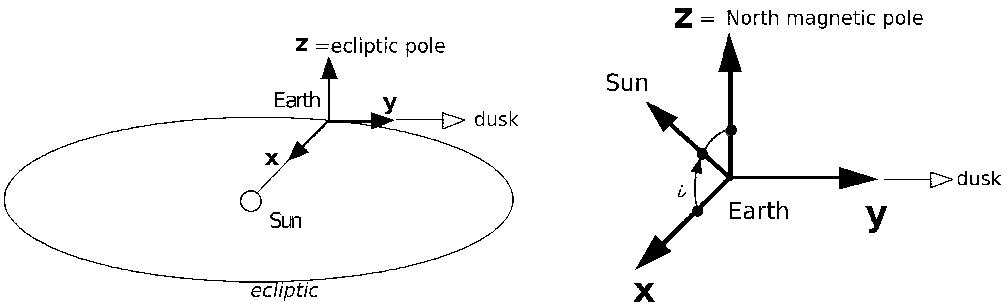
$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} Y_2 D_3 - Y_3 D_2 \\ Y_3 D_1 - Y_1 D_3 \\ Y_1 D_2 - Y_2 D_1 \end{pmatrix}$$

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = 1/Q \cdot \begin{pmatrix} D_2 R_3 - D_3 R_2 \\ D_3 R_1 - D_1 R_3 \\ D_1 R_2 - D_2 R_1 \end{pmatrix}$$

$$Q = [(D_2 R_3 - D_3 R_2)^2 + (D_3 R_1 - D_1 R_3)^2 + (D_1 R_2 - D_2 R_1)^2]^{1/2}$$

3.15 GSE to SM transformation

Geocentric Solar Ecliptic ***Solar Magnetic***



The direct computation from GSE system to SM system is a little bit difficult.
In fact, it is more easy to pass by the GSM intermediate system (see §1.6), because we have only the product of two rotation matrix: Indeed, in one hand GSE and GSM systems have their X-axis in common, so the only difference is a rotation around the X-axis of the angle ξ (see §3.16).
In a second hand, the GSM and SM system have the Y-axis in common, then the transformation matrix is a simple rotation of μ angle, which is named the dipole tilt angle (see 3.24).

So from 3.16 we have:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & \sin\xi \\ 0 & -\sin\xi & \cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)}$$

and from 3.24 we have:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} \cos\mu & 0 & -\sin\mu \\ 0 & 1 & 0 \\ \sin\mu & 0 & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

Thus the transformation from GSE to SM is:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} \cos\mu & \sin\mu\sin\xi & -\sin\mu\cos\xi \\ 0 & \cos\xi & \sin\xi \\ \sin\mu & -\cos\mu\sin\xi & \cos\mu\cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}}$$

The inverse transformation being:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} \cos\mu & 0 & \sin\mu \\ \sin\mu\sin\xi & \cos\xi & -\cos\mu\sin\xi \\ -\sin\mu\cos\xi & \sin\xi & \cos\mu\cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}}$$

Reminder

The μ angle is computed in section 3.24:

$$\boxed{\begin{array}{rcl} \sin\mu & = & \mathscr{S}_1 D_1 + \mathscr{S}_2 D_2 + \mathscr{S}_3 D_3 \\ \cos\mu & = & (1-\sin^2\mu)^{1/2} \end{array}}$$

where $\vec{\mathscr{S}}$ is the direction of the sun and $\vec{D}$ the dipole direction, both in GEO system.

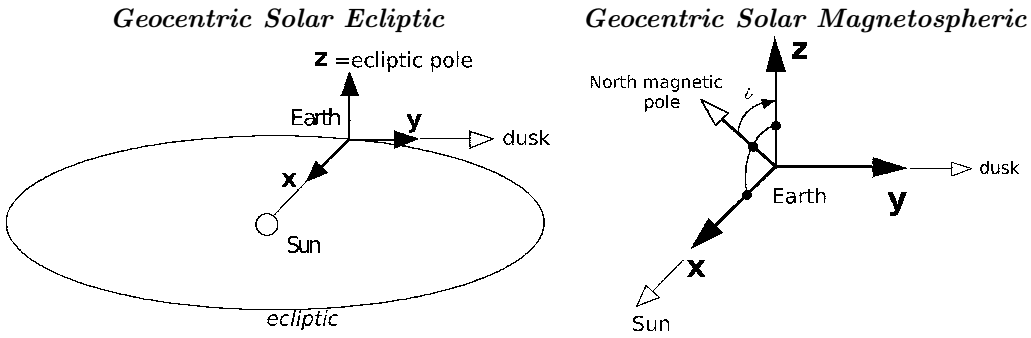
Vector $\vec{\mathscr{S}}$ can be computed from GEI to GEO transformation given in §3.2; then we have:

$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = (S_1\cos\theta + S_2\sin\theta, -S_1\sin\theta + S_2\cos\theta, S_3)$$

where $\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ.

Vector $\vec{D}$ is obtained from the International Geomagnetic Reference Field (IGRF); practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine; value for 1965.0 is $\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$

3.16 GSE to GSM transformation



GSE and GSM systems have their X-axis in common, so the only difference is a rotation around the X-axis of the angle ξ, thus the matrix transformation is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & \sin\xi \\ 0 & -\sin\xi & \cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)}$$

The inverse transformation is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & -\sin\xi \\ 0 & \sin\xi & \cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

Nevertheless the ξ angle cannot be obtained from a simple equation. To compute the rotation terms of transformation matrix, we use the GSE to GEI and the GEI to GSM previous matrix transformations, given in §3.6 and §3.5.

From §3.6 we have:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} S_1 & S_2 & S_3 \\ y_1 & y_2 & y_3 \\ E_1 & E_2 & E_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)}$$

Where $\vec{S}$ is the direction of the Sun and $\vec{E}$ the direction of ecliptic pole in GEI system, and last $\vec{y} = (\vec{E} \times \vec{S})$ still in GEI system. Note that we changed the notation $\vec{Y}$ by $\vec{y}$ to avoid confusion with the next $\vec{Y}$ vector.

From §3.5 we have:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEI)} = \begin{pmatrix} S_1 & Y_1 & Z_1 \\ S_2 & Y_2 & Z_2 \\ S_3 & Y_3 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

Where we have this time in GEI system $\vec{Y} = (\vec{M} \times \vec{S})/|(\vec{M} \times \vec{S})|$ and $\vec{Z} = (\vec{S} \times \vec{Y})$, $\vec{M}$ being the direction of dipole axis.

So we can write the GSM to GSE transformation as:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} S_1 & S_2 & S_3 \\ y_1 & y_2 & y_3 \\ E_1 & E_2 & E_3 \end{pmatrix} \begin{pmatrix} S_1 & Y_1 & Z_1 \\ S_2 & Y_2 & Z_2 \\ S_3 & Y_3 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

which give :

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} S\cdot S & S\cdot Y & S\cdot Z \\ y\cdot S & y\cdot Y & y\cdot Z \\ E\cdot S & E\cdot Y & E\cdot Z \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

which becomes, since $\vec{S}$ and $\vec{Y}$ are unit vectors perpendicular between us, as $\vec{S}$ and $\vec{Z}$:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & y\cdot Y & y\cdot Z \\ 0 & E\cdot Y & E\cdot Z \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

Of course the final matrix does not depend on the $\vec{S}$ vector.

Computation of cos ξ:

We have to equalize $\vec{y}\cdot\vec{Y}$ and $\vec{E}\cdot\vec{Z}$ terms as $\cos\xi$

Taking	$\vec{y}\cdot\vec{Y} = (\vec{E}\times\vec{S})\cdot(\vec{M}\times\vec{S})/\lvert(\vec{M}\times\vec{S})\rvert$
we compute	$\vec{E}\cdot\vec{Z} = \vec{E}\cdot(\vec{S}\times\vec{Y}) = \vec{E}\cdot[(\vec{S}\times(\vec{M}\times\vec{S})/\lvert(\vec{M}\times\vec{S})\rvert]$
since	$\vec{A}\cdot(\vec{B}\times\vec{C}) = (\vec{A}\times\vec{B})\cdot\vec{C}$
we effectively found	$\vec{E}\cdot\vec{Z} = (\vec{E}\times\vec{S})\cdot(\vec{M}\times\vec{S})/\lvert(\vec{M}\times\vec{S})\rvert = \vec{y}\cdot\vec{Y}$
then:	$\vec{y}\cdot\vec{Y} = \vec{E}\cdot\vec{Z} = \cos\xi$

By replacing the corresponding values, we set $\cos\xi = (\vec{E}\times\vec{S})\cdot\dfrac{\vec{M}\times\vec{S}}{\lvert\vec{M}\times\vec{S}\rvert}$

since $\vec{A}\cdot(\vec{B}\times\vec{C}) = (\vec{A}\times\vec{B})\cdot\vec{C}$

we found $\cos\xi = -(\vec{E}\times\vec{S})\cdot\dfrac{\vec{S}\times\vec{M}}{\lvert\vec{M}\times\vec{S}\rvert} = -\dfrac{(\vec{E}\times\vec{S})\times\vec{S}}{\lvert\vec{M}\times\vec{S}\rvert}\cdot\vec{M}$

and finally :

$$\boxed{\cos\xi = \frac{\vec{E}\cdot\vec{M}}{\lvert\vec{M}\times\vec{S}\rvert}}$$

Note that in this formula $\vec{E}$ and $\vec{M}$ are known since:

1) the direction of ecliptic pole in GEI system is a known constant value:

$$\vec{E} = (E_1, E_2, E_3) = (0, -0.398, 0.917)$$

2) $\vec{M}$ is the dipole direction in GEI system, computed §3.5 as:

$$\vec{M} = (M_1, M_2, M_3) = (D_1 \cos\theta - D_2 \sin\theta, D_1 \sin\theta + D_2 \cos\theta, D_3)$$

3) the geographic coordinates of the dipole axis $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine; for instance for IGRF epoch 1965 we have:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

4) the θ Greenwich Mean Sideral Time is computed for a given time and year from ***cp_gei_sun_dir*** subroutine.

5) the normalisation term $|\vec{M} \times \vec{S}|$ can be computed as:

$$Q = |\vec{M} \times \vec{S}| = \sqrt{(M_2S_3 - M_3S_2)^2 + (M_3S_1 - M_1S_3)^2 + (M_1S_2 - M_2S_1)^2}$$

This leads to the formula:

$$\boxed{\cos\xi = (E_1M_1 + E_2M_2 + E_3M_3)/Q}$$

Computation of sin ξ

Similarly one has to ensure that the $\vec{E} \cdot \vec{Y}$ and $-\vec{y} \cdot \vec{Z}$ terms are equal.

Taking $\vec{y} \cdot \vec{Z} = (\vec{E} \times \vec{S}) \cdot (\vec{S} \times \vec{Y})$

since $\vec{A} \cdot (\vec{B} \times \vec{C}) = (\vec{A} \times \vec{B}) \cdot \vec{C}$

we have $\vec{y} \cdot \vec{Z} = [(\vec{E} \times \vec{S}) \times \vec{S}] \cdot \vec{Y}$

and find: $\vec{y} \cdot \vec{Z} = -\vec{E} \cdot \vec{Y} = -\sin\xi$

Similarly computation of $\sin\xi$ is made as: $\sin\xi = \vec{E} \cdot \vec{Y}$

thus :

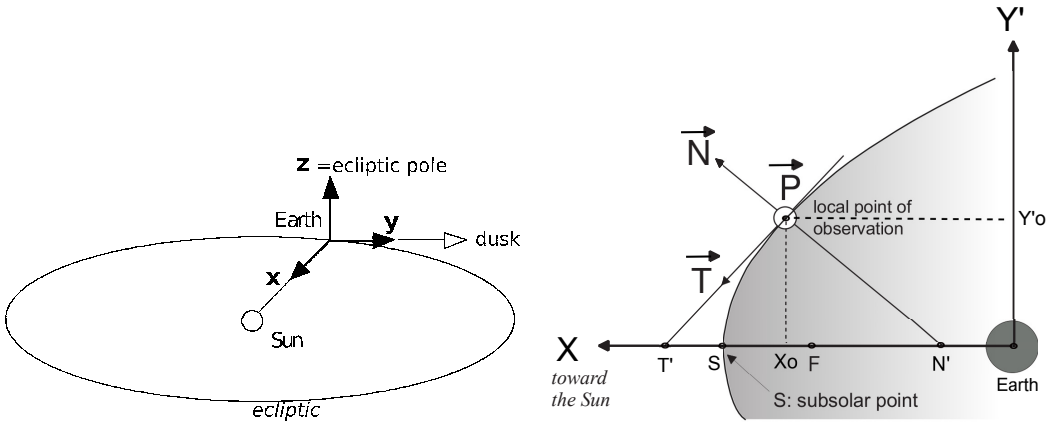
$$\boxed{\sin\xi = \vec{E} \cdot \frac{\vec{M} \times \vec{S}}{|\vec{M} \times \vec{S}|}}$$

The expended formula gives:

$$\boxed{\sin\xi = \left[\; (E_1(M_2S_3 - M_3S_2) + E_2(M_3S_1 - M_1S_3) + E_3(M_1S_2 - M_2S_1) \; \right]/Q}$$

3.17 GSE to TPN transformation

Geocentric Solar Ecliptic *Tangent Parabole Normal*



TPN system is dfined by a paraboloid aroud the X axis toward the Sun, and passing by a given point **M** x_0, y_0, z_0. Let's consider a plane (X',Y') containing the **M** point where X' is the X axis of GSE system. In this plane, the parabol equation is:

$$y'^2 = 2p(Sx - x')$$

Computation of the $\vec{T}, \vec{P}, \vec{N}$ vectors in GSE system

From parabol knowed properties (see http://homeomath2.imingo.net/parabol5.htm or other math book) the [x_0-N] distance is the p parameter, which, from the equation'parabol passing by the given point, is:

$$p = \frac{1}{2}\frac{(y_0^2 + z_0^2)}{(S_x - x_0)}$$

So in the (X',Y',Z') system, the coordinates of the nonstandard (not normalisez) $\vec{N}$ vector is:

$$\begin{pmatrix} \widetilde{N}_x \\ \widetilde{N}_y \\ \widetilde{N}_z \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} p \\ y'_0 \\ 0 \end{pmatrix}$$

Another knowed propertie is that the submit S is the middle of the $[T' - x_0]$ segment. So we have directly the coordinate of the nonstandard $\vec{T}$ vector:

$$\begin{pmatrix} \widetilde{T}_x \\ \widetilde{T}_y \\ \widetilde{T}_z \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} 2(Sx - x_0) \\ -y'_0 \\ 0 \end{pmatrix}$$

To get the tranformation between X'Y'Z' system and GSE system, we use the fact that GSE system is deduced from X'Y'Z' by a rotation around the X axis (see fig. 3.1):

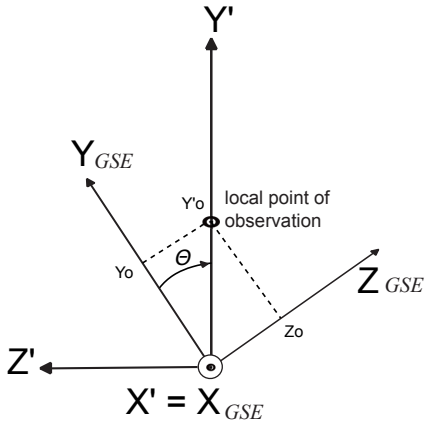


Figure 3.1: X'Y'Z' and GSE system are different by a rotation around X

Thus the transformation from $X'Y'Z'$ coordinates to GSE coordinates is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(X'Y'Z')}$$

$$\text{with } cos(\theta) = \frac{y_0}{\sqrt{(y_0^2 + z_0^2)}} \qquad \text{and } sin(\theta) = \frac{z_0}{\sqrt{(y_0^2 + z_0^2)}}$$

So we have the nonstandard vectors :

$$\begin{pmatrix} \widetilde{N}_x \\ \widetilde{N}_y \\ \widetilde{N}_z \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} p \\ y'_0 \\ 0 \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} p \\ y_o \\ z_o \end{pmatrix}$$

$$\begin{pmatrix} \widetilde{T_x} \\ \widetilde{T_y} \\ \widetilde{T_z} \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} 2(Sx - x_0) \\ -y'_0 \\ 0 \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} 2(Sx - x_0) \\ -y_o \\ -z_o \end{pmatrix}$$

After normalisation we get:

$$\boxed{\begin{array}{c} \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix}_{(GSE)} = \frac{1}{p^2 + y_0^2 + z_0^2} \begin{pmatrix} p \\ y_o \\ z_o \end{pmatrix} \\ \\ \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}_{(GSE)} = \frac{1}{4(Sx - x_0)^2 + y_0^2 + z_0^2} \begin{pmatrix} 2(Sx - x_0) \\ -y_o \\ -z_o \end{pmatrix} \end{array}}$$

$$\text{with } p = \frac{1}{2}\frac{(y_0^2 + z_0^2)}{(S_x - x_0)}$$

The $\vec{P}$ vector is obtained directly normalised by:

$$\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} 0 \\ \sin\theta \\ -\cos\theta \end{pmatrix}$$

$$\text{with } cos(\theta) = \frac{y_0}{\sqrt{(y_0^2 + z_0^2)}} \qquad \text{and } sin(\theta) = \frac{z_0}{\sqrt{(y_0^2 + z_0^2)}}$$

Transformation matrix from GSE to TPN

Knowing the coordinate of the $\vec{T}, \vec{P}, \vec{N}$ vectors in GSE system, we deduce at last the transformation matrix:

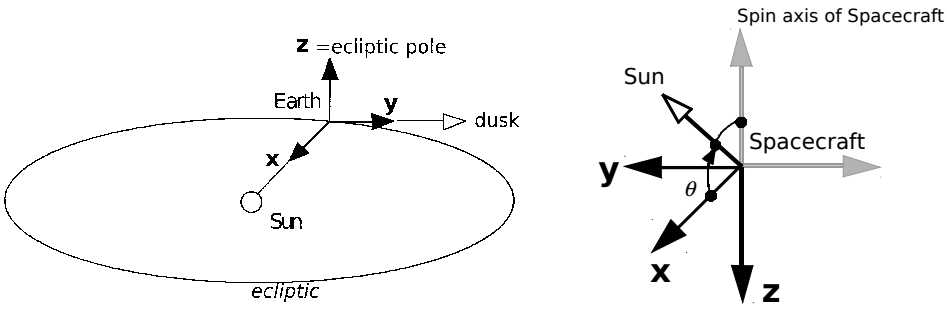
$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(TPN)} = \begin{pmatrix} T_x & T_y & T_z \\ P_x & P_y & P_z \\ N_x & N_y & N_z \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)}$$

The inverse transformation (not really usefull) is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} T_x & P_x & N_x \\ T_y & P_y & N_y \\ T_z & P_z & N_z \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(TPN)}$$

3.18 GSE to ISR2 transformation

Geocentric Solar Ecliptic ***Inverse SR2***



The best way is to use the GSE to SR2 transformation (see next section 3.19), since the tranformation from SR2 to ISR2 is very simple.

From section 3.27 we have:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SR2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(ISR2)}$$

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(ISR2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SR2)}$$

From section 3.19 we have:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2} = \begin{pmatrix} a(R_y^2 + R_z^2) & -a(R_x R_y) & -aR_x R_z \\ 0 & aR_z & -aR_y \\ R_x & R_y & R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE} = \begin{pmatrix} a(R_y^2 + R_z^2) & 0 & R_x \\ -aR_x R_y & aR_z & R_y \\ -aR_x R_z & -aR_y & R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

where $\vec{R}$ vector is the spin axis (or Rotation axis) expressed in GSE coordinate system, and extracted from auxiliary data, and $a = \dfrac{1}{\sqrt{R_y^2 + R_z^2}}$.

So the GSE to ISR2 transformation is:

$$\boxed{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{ISR2} = \begin{pmatrix} a(R_y^2 + R_z^2) & -a(R_x R_y) & -aR_x R_z \\ 0 & -aR_z & aR_y \\ -R_x & -R_y & -R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}}$$

while the inverse transformation is of course:

$$\boxed{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE} = \begin{pmatrix} a(R_y^2+R_z^2) & 0 & -R_x \\ -aR_xR_y & -aR_z & -R_y \\ -aR_xR_z & aR_y & -R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{ISR2}}$$

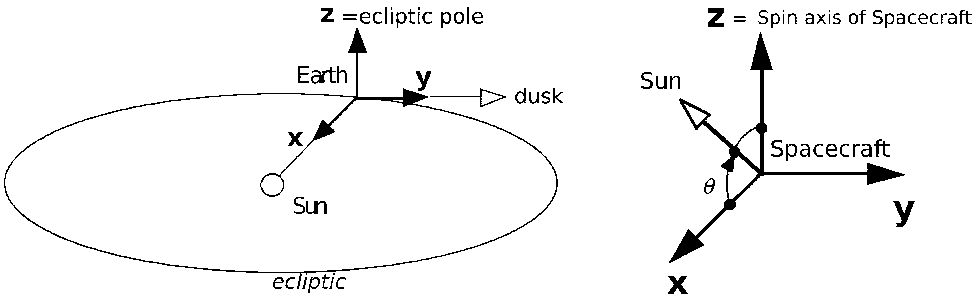
Note that for CLUSTER, $\vec{R}$ is close to the $-Z_{GSE}$ axis, so if we set $\vec{R} = (0, 0, -1)$ we can check that we found:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{ISR2}$$

This is the main interest of the ISR2 system.

3.19 GSE to SR2 transformation

Geocentric Solar Ecliptic ***Spin Reference 2***



The transformation of any vector expressed in GSE system into the SR2 system can be written as:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSE}$$

where (X_1, X_2, X_3) are the components of the $\vec{X}$ axis of the SR2 system expressed in GSE coordinates, and in the same way for $\vec{Y}$ and $\vec{Z}$ axis.

In SR2 system, $\vec{Z}$ is the rotation axis of the spacecraft, and must be known in GSE system from auxiliary data of the Spacecraft. We can write it as:

$$\vec{Z} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \vec{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix}_{GSE}$$

The components of the Y axis can be found by the relation $\vec{Y} = \dfrac{\vec{R} \times \vec{S}}{|\vec{R} \times \vec{S}|}$ which is very simple since the direction of the Sun $\vec{S}$ is the X axis of GSE system, expressed as (1,0,0). So we deduce:

$$\vec{Y} = \frac{1}{\sqrt{R_y^2 + R_z^2}} \begin{pmatrix} 0 \\ R_z \\ -R_y \end{pmatrix}_{GSE}$$

Finally we complete the system by computing $\vec{X} = \vec{Y} \times \vec{Z}$, so:

$$\vec{X} = a \begin{pmatrix} R_y^2 + R_z^2 \\ -R_x R_y \\ -R_x R_z \end{pmatrix}_{GSE} \quad \text{with} \quad a = \frac{1}{\sqrt{R_y^2 + R_z^2}}$$

Thus, the transformation of GSE coordinate into SR2 coordinates can be written as:

$$\boxed{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2} = \begin{pmatrix} a(R_y^2 + R_z^2) & -a(R_x R_y) & -aR_x R_z \\ 0 & aR_z & -aR_y \\ R_x & R_y & R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}}$$

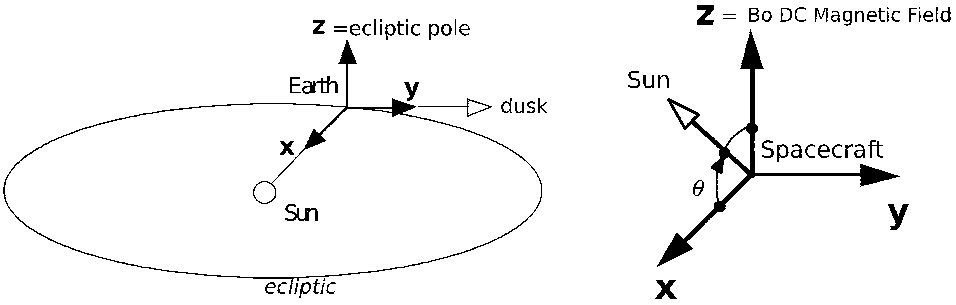
where $\vec{R}$ vector is the spin axis (or Rotation axis) expressed in GSE coordinate system, and extracted from auxiliary data.

Note that the reverse transform is simply:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE} = \begin{pmatrix} a(R_y^2 + R_z^2) & 0 & R_x \\ -aR_xR_y & aR_z & R_y \\ -aR_xR_z & {}_aR_y & R_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

3.20 GSE to MFA transformation

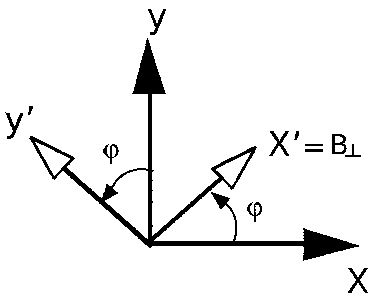
Geocentric Solar Ecliptic *Magnetic Field Aligned*



The Magnetic Field Aligned system is defined as a system where Z is along the DC magnetic field, noted as Bo, and X in the plane containing the DC magnetic field vector and the direction of the Sun. A problem of definition can appear when the DC-MF is aligned with the direction of the Sun, which is not an unlikely case. When this situation arrives, the direction of the normal to the ecliptic plane could be used instead of the direction of the Sun, but is is not still the case in this version.

The GSE to MAG transformation requires the coordinates of the $\vec{B}$ vector in GSE system. To obtain the alignment of Z with Bo, we apply two rotations.

The first rotation, in the x-y plane of GSE system, align the x axis with the perpendicular component of Bo in the x-y plane:

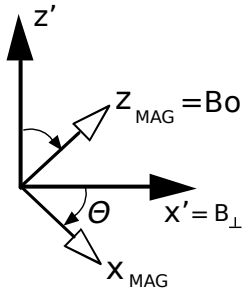


$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}$$

The values of $\sin\varphi$ and $\cos\varphi$ are simply:

$$\sin\varphi = \frac{B_y}{\sqrt{B_x^2 + B_y^2}} \qquad \cos\varphi = \frac{B_x}{\sqrt{B_x^2 + B_y^2}}$$

The second rotation, in the x'-z' plane, leads the new z axis to be aligned with Bo:



$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$

The values of $\sin\theta$ and $\cos\theta$ are simply:

$$\sin\theta = \frac{\sqrt{B_x^2 + B_y^2}}{\sqrt{B_x^2 + B_y^2 + B_z^2}} \qquad \cos\theta = \frac{B_z}{\sqrt{B_x^2 + B_y^2 + B_z^2}}$$

and the transformation from GSE to a new MAG is given by:

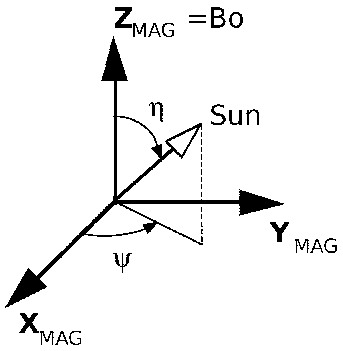
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

So:

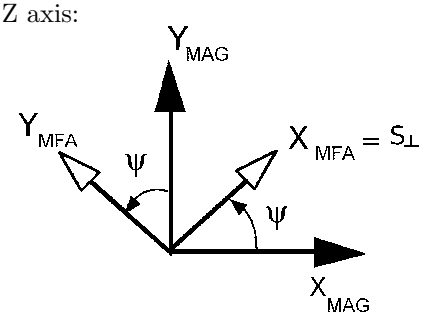
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\theta\cos\varphi & \cos\theta\sin\varphi & -\sin\theta \\ -\sin\varphi & \cos\varphi & 0 \\ \sin\theta\cos\varphi & \sin\theta\sin\varphi & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}$$

At this level, the Z axis of new MAG system is aligned with the Bo DC local magnetic field, but the MAG system is still not the MFA system, because the MFA system is defined as having the X in the plane containing the DC magnetic field vector and the direction of the Sun. So we must apply a third rotation to pass from the MAG system to the MFA system.

The third rotation in the x-y plane depends of the direction of the Sun in the MAG system, defined by its spherical angles η and ψ :



and we pass from the MAG system to the MFA system by a rotation of ψ angle around the Z axis:



$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG}$$

The ψ angle is the azimuthal angle of the direction of the Sun in MAG system. In fact, it is easy to compute, since in the GSE system the direction of the Sun is $(1, 0, 0)$.

So, the direction of the Sun in the MAG system is:

$$\vec{S_M} = \begin{pmatrix} \cos\theta\cos\varphi & \cos\theta\sin\varphi & -\sin\theta \\ -\sin\varphi & \cos\varphi & 0 \\ \sin\theta\cos\varphi & \sin\theta\sin\varphi & \cos\theta \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}_{GSE} = \begin{pmatrix} \cos\theta\cos\varphi \\ -\sin\theta \\ \sin\theta\cos\varphi \end{pmatrix}$$

and we have:

$$\sin\psi = \frac{Sy_M}{\sqrt{Sx_M^2 + Sy_M^2}} \qquad \cos\psi = \frac{Sx_M}{\sqrt{Sx_M^2 + Sy_M^2}}$$

So, the total transformation from GSE system to MFA system is:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}$$

This formula can be expanded as:

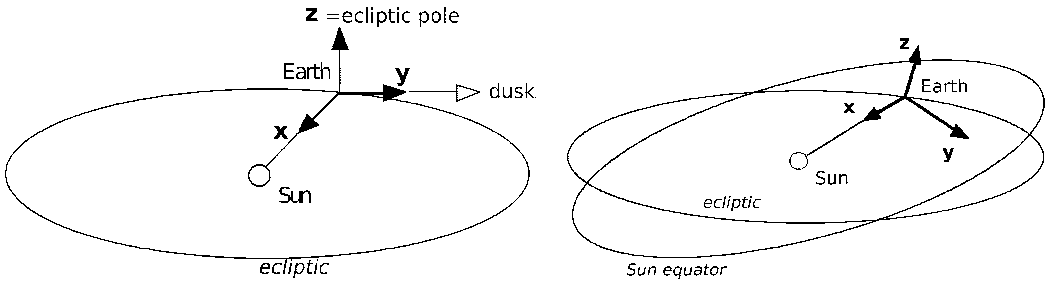
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta\cos\varphi & \cos\theta\sin\varphi & -\sin\theta \\ -\sin\varphi & \cos\varphi & 0 \\ \sin\theta\cos\varphi & \sin\theta\sin\varphi & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}$$

and finally:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi\cos\theta\cos\varphi - \sin\psi\sin\varphi & \cos\psi\cos\theta\sin\varphi + \sin\psi\cos\varphi & -\cos\psi\sin\theta \\ -\sin\psi\cos\theta\cos\varphi - \cos\psi\sin\varphi & -\sin\psi\cos\theta\sin\varphi + \cos\psi\cos\varphi & \sin\psi\sin\theta \\ \sin\theta\cos\varphi & \sin\theta\sin\varphi & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}$$

3.21 GSE to GSEQ transformation

Geocentric Solar Ecliptic ***Geocentric Solar Equatorial***



The only difference between GSE and GSEQ systems is a rotation about the common X-axis, to have the Y-GSEQ axis parallel to the Sun equator plane.

So, if θ is the rotation angle, the transformation matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSEQ)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)}$$

The inverse transformation is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSE)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSEQ)}$$

computation of θ angle:

θ is the $(\vec{Y}_{GSE}, \vec{Y}_{GSEQ})$ angle, so $\sin\theta = |\vec{Y}_{GSE} \times \vec{Y}_{GSEQ}|$

To compute θ, we use the following known vectors in GEI system:

1) the direction $\vec{S}$ of the SUN , computed from ***cp_gei_sun_dir*** subroutine:

$$\vec{S} = (S_1, S_2, S_3)$$

2) the direction of ecliptic pole, which is a known constant value:

$$\vec{E} = (E_1, E_2, E_3) = (0, -0.398, 0.917)$$

3) the direction of the rotation axis of the Sun, which is also a constant value:

$$\vec{R} = (R_1, R_2, R_3) = (0.122, -0.424, 0.899)$$

To compute $\sin\theta = |\vec{Y}_{GSE} \times \vec{Y}_{GSEQ}|$ we use the following properties:

$$\vec{Y}_{GSE} = \vec{Z}_{GSE} \times \vec{X}_{GSE} = \vec{E} \times \vec{S}$$

and since $\vec{R}$ is in the X-Z plane in the GSEQ system:

$$\vec{Y}_{GSEQ} = (\vec{R} \times \vec{S})/|\vec{R} \times \vec{S}|$$

so we have $\sin\theta = |(\vec{E} \times \vec{S}) \times (\vec{R} \times \vec{S})|/|\vec{R} \times \vec{S}|$

since $(\vec{A} \times \vec{B}) \times (\vec{C} \times \vec{D}) = (\vec{A} \times \vec{B} \cdot \vec{D})\vec{C} - (\vec{A} \times \vec{B} \cdot \vec{C})\vec{D}$

and $(\vec{A} \times \vec{B}) \cdot \vec{C} = (\vec{A} \cdot \vec{B}) \times \vec{C}$

thus $(\vec{E} \times \vec{S}) \times (\vec{R} \times \vec{S}) = (\vec{E} \times \vec{S} \cdot \vec{S})\vec{R} - (\vec{E} \times \vec{S} \cdot \vec{R})\vec{S} = (\vec{R} \times \vec{E} \cdot \vec{S})\vec{S}$

and finally, as $\vec{S}$ is a unit vector:

$$\boxed{\sin\theta = (\vec{R} \times \vec{E}) \cdot \frac{\vec{S}}{|\vec{R} \times \vec{S}|}}$$

For numerical applications, expression $\sin\theta = (\vec{R} \times \vec{E}) \cdot \dfrac{\vec{S}}{|\vec{R} \times \vec{S}|}$ can be extended as:

$$\boxed{\sin\theta = \frac{[(R_2E_3 - R_3E_2)S_1 + (R_3E_1 - R_1E_3)S_2 + (R_1E_2 - R_2E_1)S_3]}{[(R_2S_3 - R_3S_2)^2 + (R_3S_1 - R_1S_3)^2 + (R_1S_2 - R_2S_1)^2]^{1/2}}}$$

with numerical values above, this becomes:

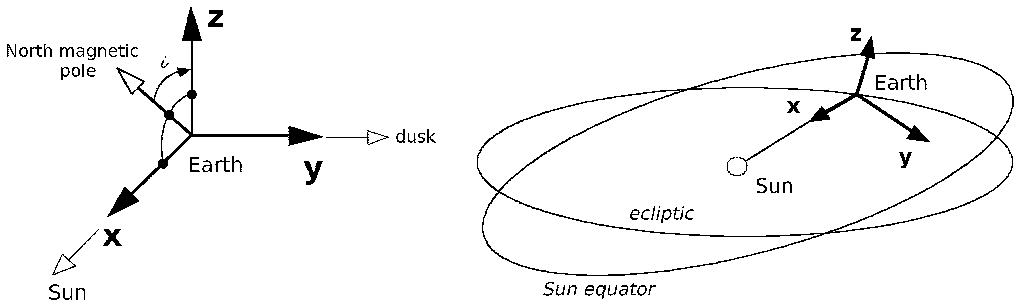
$$\boxed{\sin\theta = \frac{(-0.031, -0.112, -0.049) \cdot \vec{S}}{|(0.122, -0.424, 0.899) \cdot \vec{S}|}}$$

Since the Sun's spin axis is inclined 7.25° to the ecliptic, θ changes from -7.25 to 7.25 each year, from approximately December 5 to June 5.

The Sun's spin axis is directed most towards the Earth on approximately September 5 at which time the Earth reaches its northern most heliographic latitude. At this time $\theta = 0$.

3.22 GSM to GSEQ transformation

Geocentric Solar Magnetospheric *Geocentric Solar Equatorial*



The GSM and GSEQ systems have a common X axis, toward the Sun. The best way to compute GSM to GSEQ transformation is to use the GSM to GSE transformation, and then the GSE to GSEQ transformation.

GSE and GSM systems have their X-axis in common, so the only difference is a rotation around the X-axis of the angle ξ, and from section 3.16 we have:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSE} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & -\sin\xi \\ 0 & \sin\xi & \cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSM}$$

In other hand, the only difference between GSE and GSEQ systems is a rotation about the common X-axis, to have the Y-GSEQ axis parallel to the Sun equator plane. So, if θ is the rotation angle, the transformation matrix computed in section 3.21 is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSEQ} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSE}$$

So we get immediately the transformation from GSM to GSEQ by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSM} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & -\sin\xi \\ 0 & \sin\xi & \cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSEQ}$$

which gives:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSEQ} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta\cos\xi - \sin\theta\sin\xi & -(\cos\theta\sin\xi - \sin\theta\cos\xi) \\ 0 & \sin\theta\cos\xi + \cos\theta\sin\xi & -\sin\theta\sin\xi + \cos\theta\cos\xi \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSM}}$$

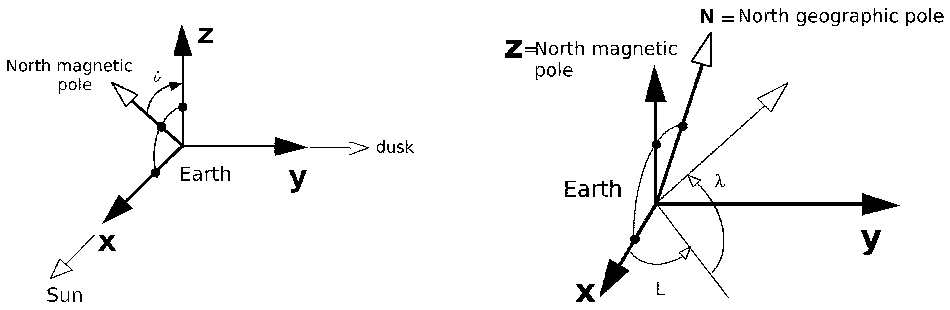
or obviously:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSEQ} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta+\xi) & -\sin(\theta+\xi) \\ 0 & \sin(\theta+\xi) & \cos(\theta+\xi) \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSM}}$$

Values of θ and ξ are given in section 3.21 and 3.16

3.23 GSM to MAG transformation

Geocentric Solar Magnetospheric ***Magnetic Dipole***



The best way to compute the GSM to MAG transformation is tu use the SM intermediate system. Indeed, the The GSM and SM system have the Y-axis in common, then the transformation matrix is a simple rotation of μ angle, which is named the dipole tilt angle. This transformation is given in section 3.24:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} \cos\mu & 0 & -\sin\mu \\ 0 & 1 & 0 \\ \sin\mu & 0 & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

In other hand, the SM and MAG system have the Z-axis in common, then the transformation matrix is again a simple rotation of ϕ angle. This transformation is given in section 3.26:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

So we get immediately the GSM to MAG transformation by:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\mu & 0 & -\sin\mu \\ 0 & 1 & 0 \\ \sin\mu & 0 & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

which give:

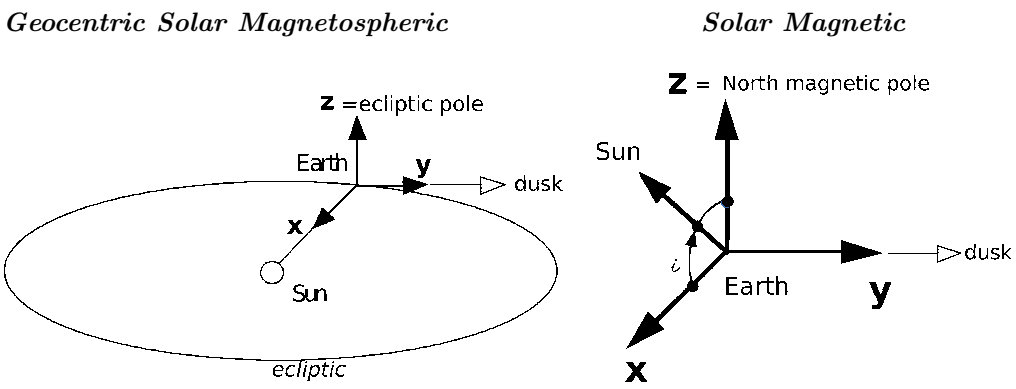
$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\phi\cos\mu & \sin\phi & -\cos\phi\sin\mu \\ -\sin\phi\cos\mu & \cos\phi & \sin\phi\sin\mu \\ \sin\mu & 0 & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSM}}$$

and conversely:

$$\boxed{\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{GSM} = \begin{pmatrix} \cos\phi\cos\mu & -\sin\phi\cos\mu & \sin\mu \\ \sin\phi & \cos\phi & 0 \\ -\cos\phi\sin\mu & \sin\phi\sin\mu & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{MAG}}$$

Values of ϕ and μ are given in section 3.26 and 3.24.

3.24 GSM to SM transformation



The GSM and SM system have the Y-axis in common, then the transformation matrix is a simple rotation of μ angle, which is named the dipole tilt angle.

Thus the transformation matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} \cos\mu & 0 & -\sin\mu \\ 0 & 1 & 0 \\ \sin\mu & 0 & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

and the inverse transformation is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} \cos\mu & 0 & \sin\mu \\ 0 & 1 & 0 \\ -\sin\mu & 0 & \cos\mu \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

The angle μ can be obtained from $\sin\mu = \vec{\mathscr{S}} \cdot \vec{D}$, where $\vec{\mathscr{S}}$ is the direction of the sun and $\vec{D}$ the dipole direction, both in GEO system.

Vector $\vec{\mathscr{S}}$ can be computed from GEI to GEO transformation given in §3.2; then we have:

$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = (S_1\cos\theta + S_2\sin\theta, -S_1\sin\theta + S_2\cos\theta, S_3)$$

where $\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ.

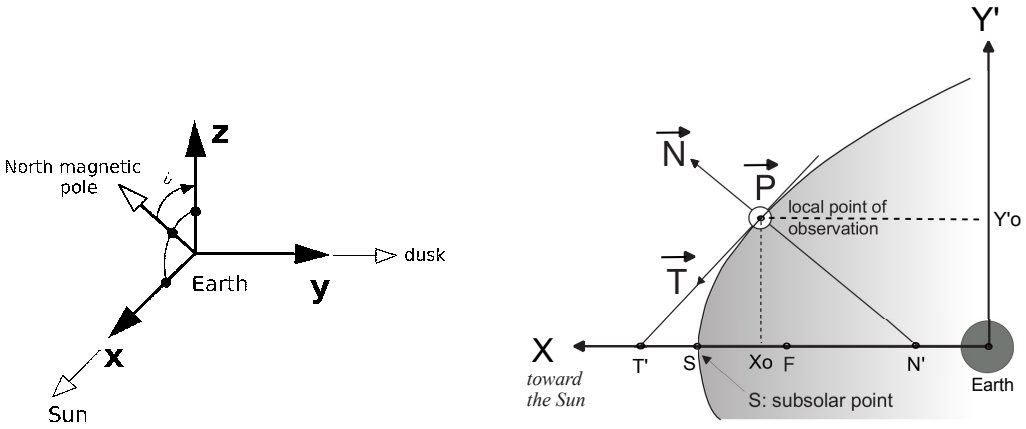
Vector $\vec{D}$ is obtained from the International Geomagnetic Reference Field (IGRF); practically, $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine; value for 1965.0 is $\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$

Finally, rotation matrix elements are:

$$\begin{aligned} \sin\mu &= \mathscr{S}_1 D_1 + \mathscr{S}_2 D_2 + \mathscr{S}_3 D_3 \\ \cos\mu &= (1 - \sin^2\mu)^{1/2} \end{aligned}$$

3.25 GSM to TPN transformation

Geocentric Solar Magnetospheric ***Tangent Paraboloid Normal***



TPN system is dfined by a paraboloid aroud the X axis toward the Sun, and passing by a given point **M** (x_0, y_0, z_0). The formulas below are the same that the GSE system, since GSM and GSE have the same X axis, which is the axe of revolution of the paraboloid.

So let's consider a plane (X',Y') containing the **M** point where X' is the X axis of GSM system. In this plane, the parabol equation is:

$$y'^2 = 2p(Sx - x')$$

Computation of the $\vec{T}, \vec{P}, \vec{N}$ vectors in GSM system

From parabol knowed properties (see http://homeomath2.imingo.net/parabol5.htm or other math book) the [x_0-N] distance is the p parameter, which, from the equation'parabol passing by the given point, is:

$$p = \frac{1}{2}\frac{(y_0^2 + z_0^2)}{(S_x - x_0)}$$

So in the (X',Y',Z') system, the coordinates of the nonstandard (not normalisez) $\vec{N}$ vector is:

$$\begin{pmatrix} \widetilde{N}_x \\ \widetilde{N}_y \\ \widetilde{N}_z \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} p \\ y'_0 \\ 0 \end{pmatrix}$$

Another knowed propertie is that the submit S is the middle of the $[T' - x_0]$ segment. So we have directly the coordinate of the nonstandard $\vec{T}$ vector:

$$\begin{pmatrix} \widetilde{T}_x \\ \widetilde{T}_y \\ \widetilde{T}_z \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} 2(Sx - x_0) \\ -y'_0 \\ 0 \end{pmatrix}$$

To get the tranformation between X'Y'Z' system and GSM system, we use the fact that GSM system is deduced from X'Y'Z' by a rotation around the X axis (see fig. 3.2):

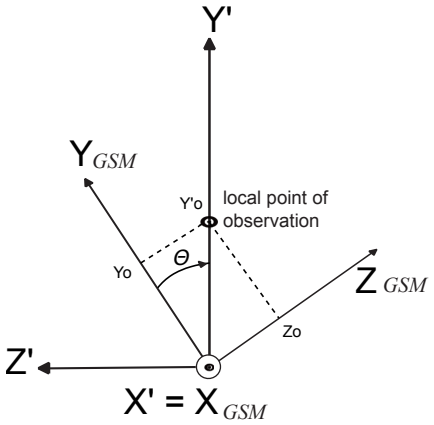


Figure 3.2: X'Y'Z' and GSM system are different by a rotation around X

Thus the transformation from $X'Y'Z'$ coordinates to GSM coordinates is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(X'Y'Z')}$$

$$\text{with } cos(\theta) = \frac{y_0}{\sqrt{(y_0^2 + z_0^2)}} \qquad \text{and } sin(\theta) = \frac{z_0}{\sqrt{(y_0^2 + z_0^2)}}$$

So we have the nonstandard vectors :

$$\begin{pmatrix} \widetilde{N}_x \\ \widetilde{N}_y \\ \widetilde{N}_z \end{pmatrix}_{(GSM)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} p \\ y_0' \\ 0 \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} p \\ y_o \\ z_o \end{pmatrix}$$

$$\begin{pmatrix} \widetilde{T_x} \\ \widetilde{T_y} \\ \widetilde{T_z} \end{pmatrix}_{(GSM)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} 2(Sx - x_0) \\ -y_0' \\ 0 \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} 2(Sx - x_0) \\ -y_o \\ -z_o \end{pmatrix}$$

After normalisation we get:

$$\boxed{\begin{array}{c} \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix}_{(GSM)} = \frac{1}{p^2 + y_0^2 + z_0^2} \begin{pmatrix} p \\ y_o \\ z_o \end{pmatrix} \\ \\ \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}_{(GSM)} = \frac{1}{4(Sx - x_0)^2 + y_0^2 + z_0^2} \begin{pmatrix} 2(Sx - x_0) \\ -y_o \\ -z_o \end{pmatrix} \end{array}}$$

$$\text{with } p = \frac{1}{2} \frac{(y_0^2 + z_0^2)}{(S_x - x_0)}$$

The $\vec{P}$ vector is obtained directly normalised by:

$$\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix}_{(GSM)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & -\cos\theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}_{(X'Y'Z')} = \begin{pmatrix} 0 \\ \sin\theta \\ -\cos\theta \end{pmatrix}$$

$$\text{with } cos(\theta) = \frac{y_0}{\sqrt{(y_0^2 + z_0^2)}} \qquad \text{and } sin(\theta) = \frac{z_0}{\sqrt{(y_0^2 + z_0^2)}}$$

Transformation matrix from GSM to TPN

Knowing the coordinate of the $\vec{T}, \vec{P}, \vec{N}$ vectors in GSM system, we deduce at last the transformation matrix:

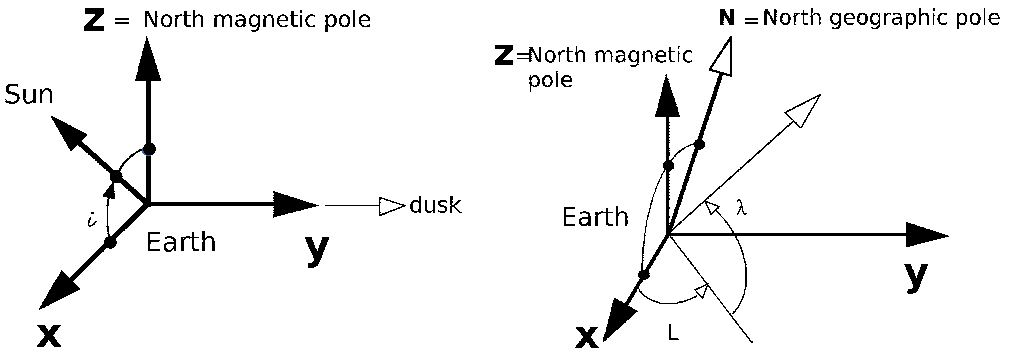
$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(TPN)} = \begin{pmatrix} T_x & T_y & T_z \\ P_x & P_y & P_z \\ N_x & N_y & N_z \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)}$$

The inverse transformation (not really usefull) is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GSM)} = \begin{pmatrix} T_x & P_x & N_x \\ T_y & P_y & N_y \\ T_z & P_z & N_z \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(TPN)}$$

3.26 SM to MAG transformation

Solar Magnetic ***Geomagnetic***



The SM and MAG system have the Z-axis in common, then the transformation matrix is a simple rotation of ϕ angle, thus the transformation matrix of any vector $\vec{V}$ is:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

and the inverse transformation is obviously:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)} = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)}$$

Nevertheless the angle ϕ is not derivable from a simple equation.

To compute the rotation terms of transformation matrix, we use the MAG to GEO and the GEO to SM matrix transformations, given in §3.8 and §3.9.

The first transformation is noted (see §3.8):

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ D_1 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)}$$

with $\vec{X} = (\vec{Y} \times \vec{D})$ in GEO System and $\vec{Y} = (\vec{N} \times \vec{D})/|\vec{N} \times \vec{D}|$

$\vec{N}$ and $\vec{D}$ vector are respectively the North geographic pole and the magnetic dipole in geographic system.

Practically, geographic dipole direction $\vec{D}$ is computed for a given time and year from ***cp_geo_dipole_dir*** subroutine; value for 1965.0 is:

$$\vec{D} = (D_1, D_2, D_3) = (0.06859, -0.18602, 0.98015)$$

we deduce from $\vec{Y} = \dfrac{\vec{N} \times \vec{D}}{|\vec{N} \times \vec{D}|}$ the result $\vec{Y} = \dfrac{(-D_2, D_1, 0)}{(D_1^2 + D_2^2)^{1/2}}$

The second transformation is noted (see §3.9):

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(GEO)} = \begin{pmatrix} x_1 & y_1 & D_1 \\ x_2 & y_2 & D_2 \\ x_3 & y_3 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

with $x = \vec{y} \times \vec{D}$ in GEO system and $\vec{y} = (\vec{D} \times \vec{\mathscr{S}})/|\vec{D} \times \vec{\mathscr{S}}|$

$\vec{\mathscr{S}}$ is the direction of the sun in GEO system, which can be computed from GEI to GEO transformation given in §3.2, then we have:

$$\vec{\mathscr{S}} = (\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3) = (S_1 \cos\theta + S_2 \sin\theta, -S_1 \sin\theta + S_2 \cos\theta, S_3)$$

$\vec{S}$ is the direction of the Sun in GEI system, computed from ***cp_gei_sun_dir*** subroutine, such as the Greenwich Mean Sideral Time θ.

Knowing all elements for MAG to GEO and GEO to SM transformations, we can write the SM to MAG transformation as:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ D_1 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} x_1 & y_1 & D_1 \\ x_2 & y_2 & D_2 \\ x_3 & y_3 & D_3 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

which give :

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} \vec{X} \cdot \vec{x} & \vec{X} \cdot \vec{y} & \vec{X} \cdot \vec{D} \\ \vec{Y} \cdot \vec{x} & \vec{Y} \cdot \vec{y} & \vec{Y} \cdot \vec{D} \\ \vec{D} \cdot \vec{x} & \vec{D} \cdot \vec{y} & \vec{D} \cdot \vec{D} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

In other hand we can write the following equivalences:

$$\begin{cases} \vec{X} \cdot \vec{D} = (\vec{Y} \times \vec{D}) \cdot \vec{D} = \vec{Y} \cdot (\vec{D} \times \vec{D}) = 0 \\ \vec{Y} \cdot \vec{D} = (\vec{N} \times \vec{D}) \cdot \vec{D}/|\vec{N} \times \vec{D}| = \vec{N} \cdot (\vec{D} \times \vec{D})/|\vec{N} \times \vec{D}| = 0 \\ \vec{D} \cdot \vec{x} = \vec{D} \cdot (\vec{y} \times \vec{D}) = -\vec{D} \cdot (\vec{D} \times \vec{y}) = -(\vec{D} \times \vec{D}) \cdot \vec{y} = 0 \\ \vec{D} \cdot \vec{y} = \vec{D} \cdot (\vec{D} \times \vec{\mathscr{S}})/|\vec{D} \times \vec{\mathscr{S}}| = (\vec{D} \times \vec{D}) \cdot \vec{\mathscr{S}}/|\vec{D} \times \vec{\mathscr{S}}| = 0 \\ \vec{D} \cdot \vec{D} = 1 \end{cases} \tag{3.4}$$

then we have the following matrix:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(MAG)} = \begin{pmatrix} \vec{X}\cdot\vec{x} & \vec{X}\cdot\vec{y} & 0 \\ \vec{Y}\cdot\vec{x} & \vec{Y}\cdot\vec{y} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SM)}$$

Computation of cos φ

We have to equalize $\vec{X}\cdot\vec{x}$ and $\vec{Y}\cdot\vec{y}$ terms as $\cos\varphi$

$$\vec{X}\cdot\vec{x} = (\vec{Y}\times\vec{D})\cdot(\vec{y}\times\vec{D}) = -(\vec{Y}\times\vec{D})\cdot(\vec{D}\times\vec{y})$$

Taking $\vec{X}\cdot\vec{x} = -(\vec{Y}\times\vec{D})\times\vec{D}\cdot\vec{y}$

and since $\vec{Y}$ and $\vec{D}$ are perpendicular:

$$\boxed{\vec{X}\cdot\vec{x} = \vec{Y}\cdot\vec{y} = \cos\varphi}$$

Coordinates of $\vec{Y}$ axe is given in 3.8 as: $\vec{Y} = \dfrac{(-D_2, D_1, 0)}{(D_1^2+D_2^2)^{1/2}}$ and $\vec{y} = \dfrac{\vec{D}\times\vec{\mathscr{S}}}{|\vec{D}\times\vec{\mathscr{S}}|}$

Then we have: $\cos\varphi = \vec{Y}\cdot\vec{y} = \vec{Y}\cdot(\vec{D}\times\vec{\mathscr{S}})/|\vec{D}\times\vec{\mathscr{S}}| = (\vec{Y}\times\vec{D})\cdot\vec{\mathscr{S}}/|\vec{D}\times\vec{\mathscr{S}}|$

The $(\vec{Y}\times\vec{D})$ vector can be expanded as: $(\vec{Y}\times\vec{D}) = (D_1D_3, D_2D_3, -\dfrac{D_1^2+D_2^2}{(D_1^2+D_2^2)^{1/2}})$

and we deduce:

$$\boxed{\cos\varphi = \frac{1}{Q}\cdot[D_1D_3\mathscr{S}_1 + D_2D_3\mathscr{S}_2 - D_1^2 + D_2^2\mathscr{S}_3]}$$

with $Q = (D_1^2+D_2^2)^{1/2}.[(D_2\mathscr{S}3 - D_3\mathscr{S}_2)^2 + (D_3\mathscr{S}1 - D_1\mathscr{S}3)^2 + (D_1\mathscr{S}_2 - D_2\mathscr{S}_1)^2]^{1/2}$

and $(\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}3) = ((S_1\cos\theta + S_2\sin\theta), (-S_1\sin\theta + S_2\cos\theta), S_3)$

Computation of sin φ

We have to equalize $\vec{X}\cdot\vec{y}$ and $-\vec{Y}\cdot\vec{x}$ terms as $\sin\varphi$

Taking $\vec{X}\cdot\vec{y} = (\vec{Y}\times\vec{D})\cdot\vec{y}$ and $-\vec{Y}\cdot\vec{x} = -\vec{Y}\cdot(\vec{y}\times\vec{D}) = \vec{Y}\cdot(\vec{D}\times\vec{y}) = (\vec{Y}\times\vec{D})\cdot\vec{y}$

yet we have well:

$$\boxed{\vec{X}\cdot\vec{y} = -\vec{Y}\cdot\vec{x} = \sin\varphi}$$

$\sin\varphi$ is then computed from $\sin\varphi = (\vec{Y} \times \vec{D}) \cdot \vec{y}$

From above we have: $(\vec{Y} \times \vec{D}) = (D_1 D_3, D_2 D_3, -\dfrac{D_1^2 + D_2^2}{(D_1^2 + D_2^2)^{1/2}}$

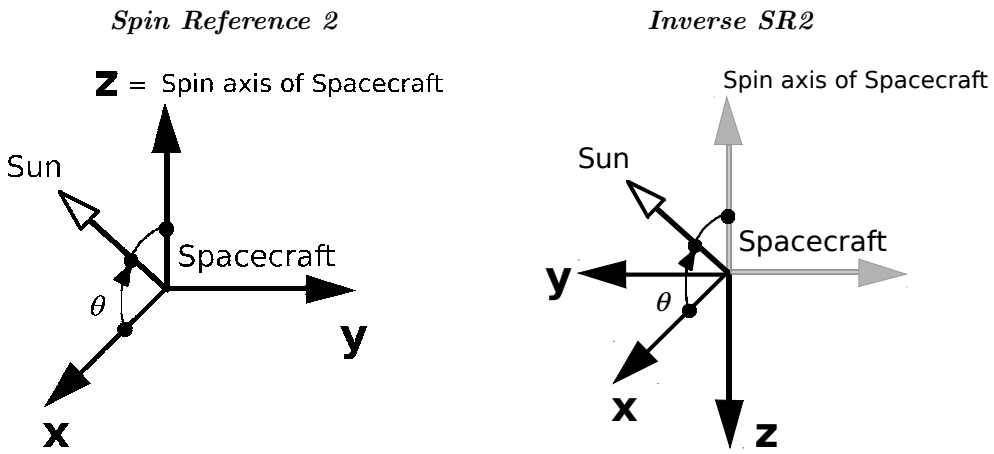
and with $\vec{y} = (\vec{D} \times \vec{\mathscr{S}})/|\vec{D} \times \vec{\mathscr{S}}|$

we deduce:

$$\boxed{\sin\varphi = \frac{1}{Q} \cdot (D_2 \mathscr{S}_1 - D_1 \mathscr{S}_2)}$$

with Q such as above.

3.27 SR2 to ISR2 transformation



ISR2 system is often used for CLUSTER data, because the spin axis of CLUSTER S/C is close to the -Z axis of GSE system. So, by simply changing the signe of Z and Y in the SR2 system, we obtain a system named ISR2 (Inverse SR2) which is close to the GSE system.

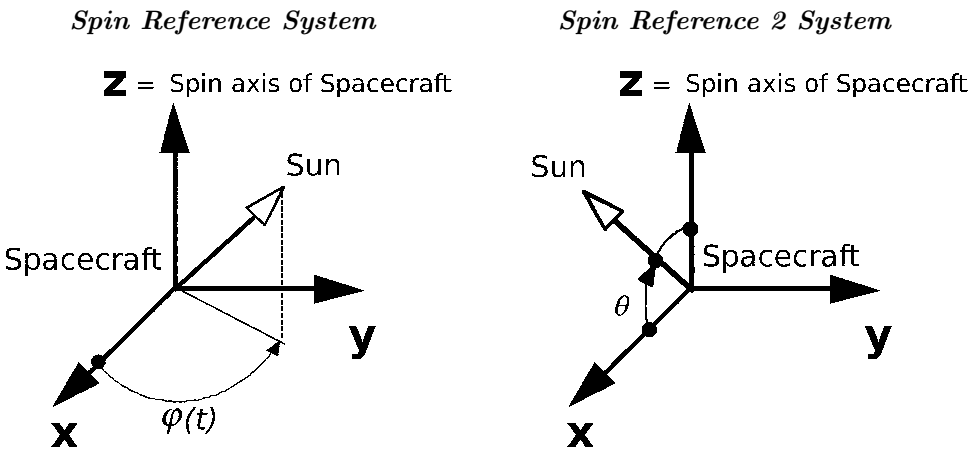
The transformation beetween SR2 to ISR2 is simply:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SR2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(ISR2)}$$

and the inverse transformation is obviously the same:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(ISR2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(SR2)}$$

3.28 SR to SR2 transformation



By the way of the Sun Reference Pulse, we know the spin phase $\varphi(t_o)$ at a given time t_o. The spin phase $\varphi(t_o)$ is the azimutal angle of the direction of the Sun in the SR system, for the knowed time to. At any time t, knowing the spin frequency f_s, the $\varphi(t)$ angle can be computed by by:

$$\varphi(t) = \varphi(t_o) - 2\pi f_s(t - t_o)$$

The spacecraft is supposed rotating in the direct x to y sens. If it rotates in the other sens, the signe of f_s must be changed. Note that at each time multiple of $t = t_o + k/f_s$ (k being any signed integer) , $\varphi(t)$ has the same value as $\varpi(t_o)$.

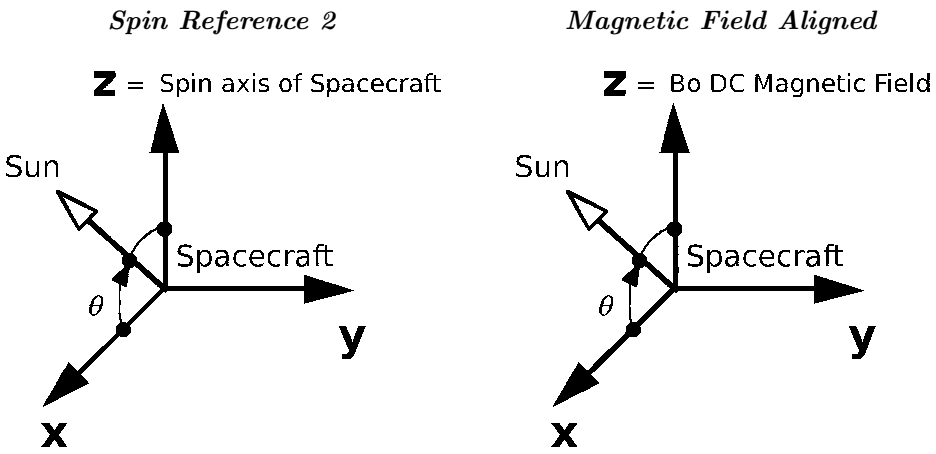
The only one difference between the SR system and the SR2 system is a rotation of the φ angle around the spin axis, so we have:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR}$$

and the inverse transformation is obviously:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

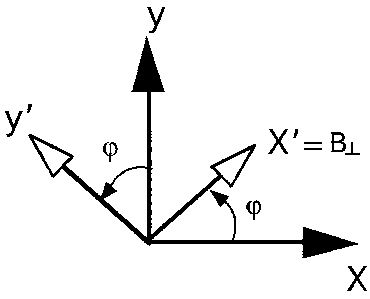
3.29 SR2 to MFA transformation



The Magnetic Field Aligned system is defined as a system where Z is along the DC magnetic field, noted as Bo, and X in the plane containing the DC magnetic field vector and the direction of the Sun. A problem of definition can appear when the DC-MF is aligned with the direction of the Sun, which is not an unlikely case. When this situation arrives, the direction of the normal to the ecliptic plane could be used instead of the direction of the Sun, but is is not still the case in this version.

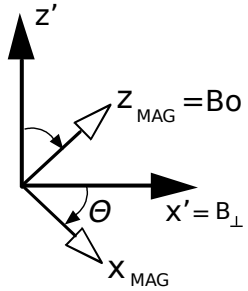
To obtain the alignment of Z with Bo, we apply two rotations.

The first rotation, in the x-y plane, align the x axis with the perpendicular component of Bo in the x-y plane:



$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

The second rotation, in the x'-z' plane, leads the new z axis to be aligned with Bo:



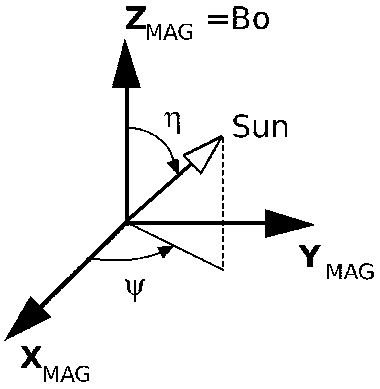
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$

and the transformation from SR2 to a new MAG is given by:

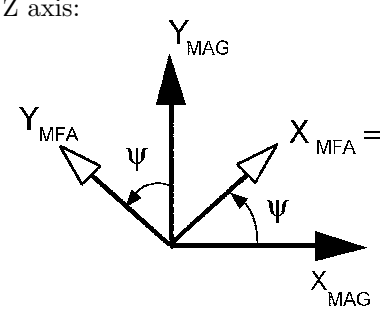
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

At this level, the Z axis of new MAG system is aligned with the Bo DC local magnetic field, but the MAG system is still not the MFA system, because the MFA system is defined as having the X in the plane containing the DC magnetic field vector and the direction of the Sun. So we must apply a third rotation to pass from the MAG system to the MFA system.

The third rotation in the x-y plane depends of the direction of the Sun in the MAG system, defined by its spherical angles η and ψ :



and we pass from the MAG system to the MFA system by a rotation of ψ angle around the Z axis:



$$S_{\perp} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MAG}$$

So, the total transformation from SR2 system to MFA system is:

$$\boxed{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}}$$

Computation of the angles θ and φ

θ and φ are respectively the polar and azimuthal angles of the Bo DC magnetic field vector in SR2 system. If we note Bx, By and Bz the components of Bo in SR2 system, we have:

$$\sin\theta = \frac{\sqrt{Bx^2+By^2}}{B_0} \quad , \quad \cos\theta = \frac{Bz}{B_0}$$

$$\sin\varphi = \frac{By}{\sqrt{Bx^2+By^2}} \quad , \quad \cos\varphi = \frac{Bx}{\sqrt{Bx^2+By^2}}$$

Computation of the angle ψ

ψ is the azimuthal angle of the direction of the Sun in MAG system . If we note Sx_M, Sy_M, Sz_M the Cartesian components of this direction in MAG system, we have:

$$\sin\psi = \frac{Sy_M}{\sqrt{Sx_M^2+Sy_M^2}}, \cos\psi = \frac{Sx_M}{\sqrt{Sx_M^2+Sy_M^2}}$$

Computation of Sx_M, Sy_M, Sz_M:

To compute the components of the direction of the Sun in MAG system, we use the transformation from SR2 to MAG, computed in the beginning of this annexe, as:

$$\begin{pmatrix} Sx_M \\ Sy_M \\ Sz_M \end{pmatrix}_{MAG} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix}_{SR2}$$

where S_x, S_y, S_z are the components of the direction of the Sun in SR2 system. Having $S_y = 0$, this leads to:

$$\begin{pmatrix} Sx_M \\ Sy_M \\ Sz_M \end{pmatrix}_{MAG} = \begin{pmatrix} S_x\cos\theta\cos\varphi - S_z\sin\theta \\ -S_x\sin\varphi \\ S_x\sin\theta\cos\varphi + S_z\cos\theta \end{pmatrix}$$

S_x, S_y, S_z are deduced by the transformation of GSE to SR2 given in annexe 1, thus:

$$\begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix}_{SR2} = \begin{pmatrix} a(R_y^2+R_z^2) & -a(R_xR_y) & -aR_xR_z \\ 0 & aR_z & -aR_y \\ R_x & R_y & R_z \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}_{GSE}$$

with $a = \dfrac{1}{\sqrt{Ry^2+Rz^2}}$, $\vec{R}$ vector being the Rotation axis expressed in GSE coordinate.

Note that in GSE, the direction of the Sun is simply (1,0,0), thus the direction of the Sun in SR2 system is:

$$\begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix}_{SR2} = \begin{pmatrix} \sqrt{R_y^2 + R_z^2} \\ 0 \\ R_x \end{pmatrix}$$

which leads to the result:

$$\boxed{\begin{pmatrix} Sx_M \\ Sy_M \\ Sz_M \end{pmatrix}_{MAG} = \begin{pmatrix} \sqrt{R_y^2 + R_z^2}\cos\theta\cos\varphi - R_x\sin\theta \\ -\sqrt{R_y^2 + R_z^2}\sin\varphi \\ \sqrt{R_y^2 + R_z^2}\sin\theta\cos\varphi + R_x\cos\theta \end{pmatrix}}$$

Knowing now the 3 angles θ,φ and ψ we can expand the previous formula

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{SR2}$$

as:

$$\boxed{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{MFA} = \begin{pmatrix} \cos\psi\cos\theta\cos\varphi - \sin\psi\sin\varphi & \cos\psi\cos\theta\sin\varphi + \sin\psi\cos\varphi & -\cos\psi\sin\theta \\ -\sin\psi\cos\theta\cos\varphi - \cos\psi\sin\varphi & -\sin\psi\cos\theta\sin\varphi + \cos\psi\cos\varphi & \sin\psi\sin\theta \\ \sin\theta\cos\varphi & \sin\theta\sin\varphi & \cos\theta \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{GSE}}$$

Chapter 4

USE OF EULER ANGLES

4.1 General remarks

Most space missions give Euler angles, allowing the knowledge of position of spinning spacecraft by respect to another coordinate system, assumed to be fixed by respect to the spin period. Following sections give definition of Euler angle, and how to use it. Example chosen is for GEOS spacecraft, where the fixed system is VDH (see [4]).

4.2 Definition of Euler angles

Euler angles are defining according the figure 4.2. In this example, Euler angle allow to pass from the spinning system XYZ (in blue) to the fixed system VDH (in red). The line corresponding to the intersection of two plane is the node line (green).

θ is the the nutation, φ the precession, and Ψ the intrinsic rotation.

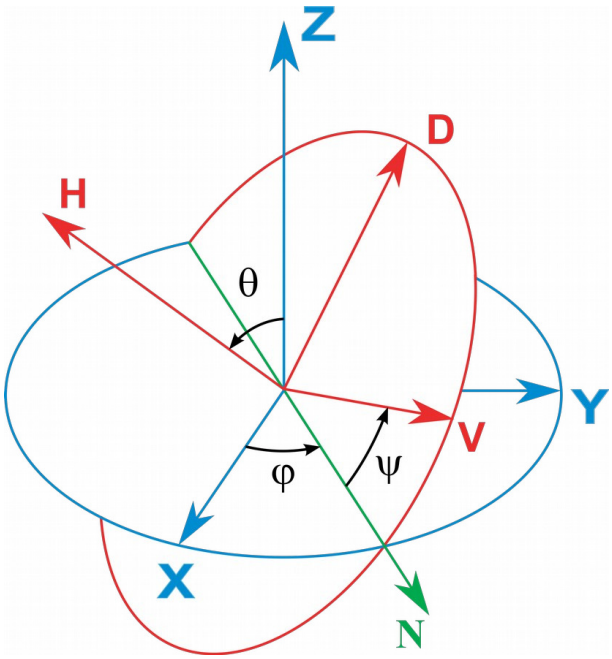


Figure 4.1: Definition of Euler angles

4.3 XYZ to VDH transformation

One need 3 rotations to pass from XYZ system to VDH one:

- a first rotation A around the Z axis (spin axis) of φ angle lead X on the node axis N, and become X'
- a second rotation B around N of θ angle lead Z on H, and become Z'
- a third rotation C around Z' of Ψ angle lead X' on V.

So the full matrix to pass from XYZ to VDH is the matrix product $R = CBA$, so be:

$$R = \begin{pmatrix} cos\Psi & -sin\Psi & 0 \\ sin\Psi & cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \end{pmatrix} \begin{pmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

After computation we get:

$$R = \begin{pmatrix} cos\Psi cos\varphi - sin\Psi cos\theta sin\varphi & -cos\Psi sin\varphi - sin\Psi cos\theta cos\varphi & sin\Psi sin\theta \\ sin\Psi cos\varphi + cos\Psi cos\theta sin\varphi & -sin\Psi sin\varphi + cos\Psi cos\theta cos\varphi & -cos\Psi sin\theta \\ sin\theta sin\varphi & sin\theta cos\varphi & cos\theta \end{pmatrix}$$

The inverse matrix is $R^{-1} = A^{-1}B^{-1}C^{-1}$ but it is more simply to take the transposed matrix.

So, to summarize:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(VDH)} = \begin{pmatrix} cos\Psi cos\varphi - sin\Psi cos\theta sin\varphi & -cos\Psi sin\varphi - sin\Psi cos\theta cos\varphi & sin\Psi sin\theta \\ sin\Psi cos\varphi + cos\Psi cos\theta sin\varphi & -sin\Psi sin\varphi + cos\Psi cos\theta cos\varphi & -cos\Psi sin\theta \\ sin\theta sin\varphi & sin\theta cos\varphi & cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(XYZ)}$$

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(XYZ)} = \begin{pmatrix} cos\Psi cos\varphi - sin\Psi cos\theta sin\varphi & sin\Psi cos\varphi + cos\Psi cos\theta sin\varphi & sin\theta sin\varphi \\ -cos\Psi sin\varphi - sin\Psi cos\theta cos\varphi & -sin\Psi sin\varphi + cos\Psi cos\theta cos\varphi & sin\theta cos\varphi \\ sin\Psi sin\theta & -cos\Psi sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}_{(VDH)}$$

Chapter 5

CALENDAR CONVERSIONS

5.1 General remarks

Most space missions deliver or use for their processing data date and time in various format, such as decimal Julian day, month-day in the month, year, decimal hours, hour-min-sec, and so on. In order to easily manipulate all theses quantities, and to get transformations between else, special calendar conversions, described below, has been added to ROCOTLIB library. Details on input/output of each subroutine are available on section 6.3

5.2 List of calendar computation utilities

5.2.1 Compute if a given year is or not a leap year (cp_leap_year)

A given year (as 1990) is a leap year if this number is divisible by 4; if the year is a secular year (as 1900 or 2000), this year is a leap year if this number is divisible by 400. For instance, 1988, 1992, 2000 are leap years, 1900, 1989, 2100 are not.

5.2.2 Compute the number of day in a month (cp_nbday_in_month)

Dependent of the year because of leap years.

5.2.3 Compute english day name from day in a week (cp_en_day_name)

Example : " Monday " for 1

5.2.4 Compute french day name from day in a week (cp_fr_day_name)

Example : " Lundi " for 1

5.2.5 Compute english month name from month number (cp_en_month_name)

Example : " January " for 1

5.2.6 Compute french month name from month number (cp_fr_month_name)

Example : " Janvier " for 1

5.3 List of calendar conversion utilities

5.3.1 Convert date to day of the year (cv_date_to_doty)

This conversion compute the day of the year, equal to 1 for January 1, from a date given as month, day in the month, and year. For instance, the day of the year of October 17, 1990 is 290. This conversion counts the number of the day in a month, taking into account leap year since February has not a constant number of days (using cp_leap_year).

5.3.2 Convert day of the year to date (cv_doty_to_date)

This is the inverse conversion of cv_date_to_doty, since one compute the corresponding month and day in the month for a given day of the year, and a given year. Year is necessary to know if the year is or not a leap year (using cp_leap_year).

5.3.3 Convert date to Julian day 1950 (cv_date_to_jul1950)

From a given date as month, day, year, this conversion compute the Julian day , i.e. day number from January 1, 1950, with Julian day equal to 1 for January 1, 1950 (use cv_date_to_doty and cp_leap_year conversions).

5.3.4 Convert date to Julian day 2000 (cv_date_to_jul2000)

From a given date as month, day, year, this conversion compute the Julian day , i.e. day number from January 1, 2000, with Julian day equal to 1 for January 1, 2000 (use cv_date_to_doty and cp_leap_year conversions).

5.3.5 Convert Julian day 1950 to date (cv_jul1950_to_date)

This is the inverse conversion of cv_date_to_jul1950, since one compute for a given Julian day the corresponding date as month, day in the month and year (use cp_leap_year and cv_doty_to_date conversions).

5.3.6 Convert Julian day 2000 to date (cv_jul2000_to_date)

This is the inverse conversion of cv_date_to_jul2000, since one compute for a given Julian day the corresponding date as month, day in the month and year (use cp_leap_year and cv_doty_to_date conversions).

5.3.7 Convert hour, min, sec to decimal hour (cv_hms_to_dech)

This conversion makes the operation hour + minute/60. + second/3600.

5.3.8 Convert decimal hour to hour, min, sec (cv_dech_to_hms)

Inverse of cv_hms_to_dech.

5.3.9 Convert day, hour, min, sec,ms. to ms of the day (cv_dhms_to_msotd)

As jour*24*360000 + hour*3600000 +min*60000 +sec*1000 +ms

5.3.10 Convert ms of the day to hour, min, sec., ms (cv_msotd_to_hmsms)

Inverse of cv_dhms_to_msotd

5.3.11 Convert date to week number in a year (cv_date_to_weekn)

A week is always beginning by a Monday.

5.3.12 Convert date to day of the week (cv_date_to_dotw)

With 1 for Monday, etc.

5.3.13 Convert the week number in a year to date (cv_weekn_to_date)

Date is computed for the first day of the week, always a Monday.

Part II

USER'S MANUAL

Chapter 6

DESCRIPTION OF AVAILABLE MODULES

6.1 General remarks

ROCOTLIB library delivers 6 kinds of subroutines:

a) "Basic computation subroutines" such as `cp_time_param, cp_gei_sun_dir` etc. It only do computations without any particular previous call. One has include in this category the particular `cp_time_param` subroutine which set for a given date and time all the time varying quantities needed for the transformations matrix (results are stored in 15 common transparent to the user). The `cp_gei_sun_dir` subroutine compute the direction of the Sun in GEI, while `cp_geo_dipole_dir` compute the direction of the magnetic dipole axis in GEO. Both are used in `cp_time_param`, but can be used independently.

b) "Calendar subroutines" contains two categories: computation and conversion. Computation subroutines are such as `cp_xxx_xxx`, for instance `cp_nbday_in_month` and does computations without any particular previous call. Conversion subroutines are such as `cv_xxx_to_xxx`, for instance `cv_date_to_doty, cv_dech_to_hms` etc. There are useful for conversion of dates and times given in various format.

c) "Give subroutines" such as `g_gei_geo_sun_dir, g_gsm_dipole_tilt_angle`, etc... give in the output arguments useful parameters as for instance direction of the Sun in GEI and GEO systems, value of the dipole tilt angle, etc... Subroutine `cp_time_param` must be called before using theses subroutines at each date/time variation.

d) "Read and check subroutines" such as `r_date, r_time`, which query and read on the standard input date and time, with automatic check of good format (no hour greater than 23, or minutes greater than 59, and so on), useful to write an interactive user program where date and time are an input.

e) "Transformations subroutines" such as `t_gei_to_geo, t_geo_to_gei` transforms input Cartesian coordinates system into an another one following mathematical expressions given in Part I ; as above, subroutine `cp_time_param` must be called before using theses subroutines at each date/time variation.

f) "Matrix operations subroutines" such as `mat_xxx` (`mat_product, mat_diff, mat_transpose, mat_diagonalise`, etc...) allowing a lot of matrix operations, and are used for MVA system (Minimum Variance Analysis).

One subroutine is out of class: a call to `print_rocot_info` subroutine just print the information relative to the library (version number, etc.).

Chapter 8 give instructions for software installation of the library on a UNIX system. The library is given in three languages: Fortran 77, Fortran 90 and C user program. All provides a test program code source to check the library validity.

Successful tests on PC/Windows 7 lab-top has been done using Msys interface and fort77 code, but are not documented here.

6.2 Description of "Basic Computation" subroutines

Subroutine cp_time_param must be called before any tranformation. It prepare all the matrix required to do coordinate transformation, depending of date and time.

Subroutines cp_time_param2 and cp_time_param3 do the same, but with input arguments different, which could be more easy to use in certain cases.

cp_time_param

```
subroutine cp_time_param(iyear,imonth,iday,ihour,imin,isec)
```

► ***compute_time_parameters**: prepare matrix for coordinate transformations*

Prepare all time varying quantities for computations of coordinate transforms of the library, and store results in 15 common statements; use cp_gei_sun_dir and cp_geo_dipole_dir subroutines. Quantities stored in commons are below:

sin and cos of GMST
ecliptic pole in GEI system
direction of the rotation axis of the sun in GEI system
dipole direction in GEI system
direction of the sun in GEO system
direction of the ecliptic in GEO system
direction of the rotation axis of the sun in GEO system
cross product MxS in GEI system
cross product ExS in GEI system
cross product RxS in GEI system
cross product RxE in GEI system
cross product DxS in GEO system
cross product ExS in GEO system
cross product RxS in GEO system
computation of gei to mag vectors
computation of gei to sm vectors
computation of gei to gsm vectors
computation of gei to gseq vectors
computation of tetq angle
computation of mu angle
computation of dzeta angle
computation of phi angle
computation of geo to mag vectors
computation of geo to sm vectors
computation of geo to gsm vectors
computation of geo to gsq vectors

```
input :  iyear, imonth, iday :  date as 1985 08 24
         iyear must be > 1901 and < 2099
         ihour,imin,isec :  time 23 57 36 (U.T.)

output :  in common statements
```

cp_time_param2

```
subroutine cp_time_param2(jd1950, houday)
```

► ***compute_time_parameters:*** *prepare matrix for coordinate transformations*
Same as cp_time_param, but with different date-time input

```
input :  Julian day from 1950, decimal hour in the day
output:  in common statements
```

---

cp_time_param3

```
subroutine cp_time_param3(jd2000, houday)
```

► ***compute_time_parameters:*** *prepare matrix for coordinate transformations*
Same as cp_time_param, but with different date-time input

```
input :  Julian day from 2000, decimal hour in the day
output:  in common statements
```

---

cp_geo_dipole_dir

```
subroutine cp_geo_dipole_dir(iyear,idoty,d1,d2,d3)
```

► ***compute_dipole_direction*** *in GEO system*

Compute geodipole axis direction from International Geomagnetic Reference Field (IGRF) models for time interval 1965 to 1990. For time out of interval, computation is made for nearest boundary.

```
input :  iyear :  year (1965 - 1990)
         idoty :  day of the year (1 for January 1)
output:  d1,d2,d3 Cartesian dipole components in GEO
```

---

cp_gei_sun_dir

```
subroutine cp_gei_sun_dir(iyear,idoty,ihour,imin,isec,gst,slong,
                          sra,sdec,obliq)
```

► ***compute_sun_direction*** *in GEI system*

Calculates four quantities in GEI system necessary for coordinate transformations dependent on sun position (and, hence, on universal time and season). From C.T. Russel71, cosmic electrodynamics ,v.2,184-196,1971, revised P. Robert 1992 & 2000.

```
input :  iyear :  year (1901-2099)
         idoty :  day of the year (1 for January 1)
         ihour,imin,isec :  hours, minutes, seconds U.T.
output:  gst          Greenwich mean sideral time (radians)
         slong        longitude along ecliptic (radians)
         sra          right ascension (radians)
         sdec         declination of the sun (radians)
```

---

cp_angle_and_ratio

```
subroutine cp_angle_and_ratio(ux,uy,uz, vx,vy,vz, angle,ratio)
```

► ***compute_angle_and_ratio*** *between 2 vectors*

```
input :  ux,uy,uz, vx,vy,vz
output:  angle (radians) and |u|/|v|
```

---

cp_sunrise_sunset

```
subroutine cp_sunrise_sunset(iyear,imon,iday,rlat,rlon,tmer,tris,
tset,durd)
```

► ***compute_Sunset*** *and other Sun parameters*

```
input :  iyear,imon,iday :  date
         rlat,rlon:  geographic latitude and longitude in rad.
output:  tmer,tris,tset,durd :  Sun meridian time, Sunrise time,
         Sunset time and duration of the day, in Cha*8 variable
```

---

cp_Euler_interpol

```
subroutine cp_Euler_interpol(a1,b1,c1,a2,b2,c2,ti,dt,ai,bi,ci)
```

► ***compute_Euler_interpolation*** *interpolation of Euler's angles*

```
input :  a1,b1,c1,a2,b2,c2, :  Values of Euler'angle at t1 and t2
         ti,dt :  time of interpolation, t1 < ti < t2, dt= t2 − t1
output:  ai,bi,ci :  values of interpolated Euler'angles at ti
```

cp_tpn_param

```
subroutine cp_tpn_param(xo,yo,zo,xs, Tx,Ty,Tz, Px,Py,Pz, Nx,Ny,Nz)
```

► ***compute_TPN_parameters***

Compute TPN vector in GSE system or any system having X axis towards the SUN.
N: Output normal to the paraboloid
T: tangent to the paraboloid, towards the summit
P: tangent to the paraboloid, P=N X T

The paraboloid is defined by its summit Xs and the local point Xo, Yo, Zo

Note: the paraboloid is close to the magnetopause if the summit is defined as the subsolar point (by T87,T89 model or other) and if the local point Xo,Yo,Zo correspond to a magnetopause crossing.

```
input :  xo,yo,zo,xs
output:  Nx,Ny,Nz,Tx,Ty,Tz,Px,Py,Pz
```

6.3 Description of "Calendar" subroutines

6.3.1 "compute" subroutines

cp_seasons

```
subroutine cp_seasons(iyear,id_sso,id_wso,id_seq,id_feq,
ct_sso,ct_wso,ct_seq,ct_feq)
```

► ***compute_seasons*** *spring and fall equinoxes, summer and winter solstices*

```
input :  iyear :  given year
output:  id_sso,id_wso :  June and december day of summer and
                          winter solstice.
         id_seq,id_feq :  same for march and september spring
                          and fall equinoxes.
         ct_sso,ct_wso :  June and december time of summer and
                          winter solstice.
         ct_seq,ct_feq :  same for march and september spring
                          and fall equinoxes; Cha*5 (hh:mm)
```

cp_nbday_in_month

```
subroutine cp_nbday_in_month(iyear,imonth,nbday)
```

► ***compute_number_of_days_in_a_month***

```
input :  iyear,imonth
output:  nbday, ex 31 for January
```

cp_leap_year

```
subroutine cp_leap_year(iyear,ileap)
```
► ***compute_leap_year*** *with ileap=1 for leap year, 0 if not*
```
input :  iyear (ex:  1980)
output:  ileap (1 or 0 if iyear is or not a leap year)
```

cp_fr_day_name

```
subroutine cp_fr_day_name(iday,cday,n)
```
► ***compute_french_day_name*** *as "Lundi" for iday= 1*
```
input :  iday, day number (1-7)
output:  cday, as ``Lundi", n= number of character
         of the word
```

cp_fr_month_name

```
subroutine cp_fr_month_name(imonth,cmonth,n)
```
► ***compute_french_month_name*** *as "Janvier" for imonth=1*
```
input :  imonth, month (1-12)
output:  cmonth, as ``Janvier", n= number of character of the word
```

cp_en_day_name

```
subroutine cp_en_day_name(iday,cday,n)
```
► ***compute_english_day_name*** *as "Monday" for iday= 1*
```
input :  iday, day number (1-7)
output:  cday, as ``Monday", n= number of character of the word
```

cp_en_month_name

```
subroutine cp_en_month_name(imonth,cmonth,n)
```
► ***compute_english_month_name*** *as "January" for imonth=1*
```
input :  imonth, month (1-12)
output:  cmonth, as ``January", n= number of character of the word
```

6.3.2 "convert" subroutines

cv_doty_to_date

```
subroutine cv_doty_to_date(idoty,iyear,imonth,iday)
```
► ***compute_date_from_day_of_the_year*** *and for a given year.*
```
input :  idoty, iyear, (idoty=1 for January 1)
output:  imonth, iday
```

cv_jul2000_to_date

```
subroutine cv_jul2000_to_date(jud00,iyear,imonth,iday)
```
► ***compute_date_from_Julian_day*** *with jud00=1 for January 1, 2000*
```
input :  jud00 Julian day (1= 1/1/2000)
output:  iyear, imonth, iday
```

cv_jul1950_to_date

```
subroutine cv_jul1950_to_date(jud50,iyear,imonth,iday)
```
► ***compute_date_from_Julian_day*** *with jud50=1 for January 1, 1950*
```
input :  jud50 Julian day (1= 1/1/1950)
output:  iyear, imonth, iday
```

cv_weekn_to_date

```
subroutine cv_weekn_to_date(iweek,iyear,imonth,iday)
```
► ***compute_date_for_first_day_of_week*** *number always a Monday*
```
input :  iweek:  week number in the year
output:  iyear, imonth, iday
```

cv_date_to_dotw

```
subroutine cv_date_to_dotw(iyear,imonth,iday,idow)
```
► ***compute_day_of_the_week*** *from date*
```
input :  iyear, imonth, iday
output:  idow :  day of the week, 1 for Monday
```

cv_date_to_doty

```
subroutine cv_date_to_doty(iyear,imonth,iday, idoty)
```
► ***compute_day_of_the_year*** *with idoty=1 for January 1*
```
input :  iyear,imonth,iday, ex:  1990,10,17
output:  idoty, ex:  290
```

cv_hms_to_dech

```
subroutine cv_hms_to_dech(ih,im,is,houday)
```
► ***compute_hour_of_the_day*** *from hours, minutes, seconds*
```
input :  ih,im,is
output:  houday decimal hour of the day
```

cv_date_to_jul1950

```
subroutine cv_date_to_jul1950(iyear,imonth,iday, jud50)
```
► ***compute_Julian_day*** *with jud50=1 for January 1, 1950*
```
input :  iyear,imonth,iday ex:  1990,10,17
output:  jud50
```

cv_date_to_jul2000

```
subroutine cv_date_to_jul2000(iyear,imonth,iday, jud00)
```
► ***compute_Julian_day*** *with jud00=1 for January 1, 2000*
```
input :  iyear,imonth,iday ex:  2001,10,17
output:  jud00
```

cv_dhms_to_msotd

```
subroutine cv_dhms_to_msotd(ih,im,is,ims,milday)
```
► ***compute_millisecond_of_the_day***
```
input :  ih,im,is,ims:  time in hour,min,sec,millisec
output:   milday, millisec.  of the day (<86400000)
```

cv_dech_to_hms

```
subroutine cv_dech_to_hms(houday,ih,im,is)
```
► ***compute_time*** *from decimal hour of the day*
```
input :  houday, decimal hour of the day
output:  ih,im,is
```

cv_msotd_to_hmsms

```
subroutine cv_msotd_to_hmsms(milday,ih,im,is,ims)
```
► ***compute_time*** *from millisecond of the day*
```
input :  houday, decimal hour of the day
output:  ih,im,is,ims
```

cv_date_to_weekn

```
subroutine cv_date_to_weekn(iyear,imonth,iday,iweek)
```
► ***compute_week_of_the_year*** *from date*
```
input :  iyear, imonth, iday
output:  iweek :  week of the year, 1 for first Monday
```

6.4 Description of "Give" subroutines

---

g_gei_geo_dipole_dir

```
subroutine g_gei_geo_dipole_dir(dxgei,dygei,dzgei,dxgeo,dygeo,dzgeo)
```

► ***give_dipole_direction*** *in GEI and GEO system*

```
input :  none
output:  dxgei,dygei,dzgei Cartesian dipole GEI coord.
         dxgeo,dygeo,dzgeo Cartesian dipole GEO coord.
```

---

g_gsm_dipole_tilt_angle

```
subroutine g_gsm_dipole_tilt_angle(diptan)
```

► ***give_dipole_tilt_angle*** *in GSM system (radians)*

```
input :  none
output:  diptan=(D,Z) angle in GSM (radians)
         (diptan>0 when the north magnetic pole
         is tilted toward the Sun)
```

---

g_gei_geo_ecliptic_dir

```
subroutine g_gei_geo_ecliptic_dir(exgei,eygei,ezgei,exgeo,eygeo,ezgeo)
```

► ***give_ecliptic_direction*** *in GEI and GEO system*

```
input :  none
output:  exgei,eygei,ezgei Cartesian eclip.  GEI coord.
         exgeo,eygeo,ezgeo Cartesian eclip.  GEO coord.
```

---

g_gei_geo_sun_rot

```
subroutine g_gei_geo_sun_rot(rxgei,rygei,rzgei,rxgeo,rygeo,rzgeo)
```

► ***give_sun_rotation_direction*** *in GEI and GEO system*

```
input :  none
output:  rxgei,rygei,rzgei Cartesian sun rot.  GEI coord.
         rxgeo,rygeo,rzgeo Cartesian sun rot.  GEO coord.
```

---

g_gei_geo_sun_dir

```
subroutine g_gei_geo_sun_dir(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)
```

► ***give_sun_direction*** *in GEI and GEO system*

```
input :  none
output:  sxgei,sygei,szgei Cartesian sun GEI coord.
         sxgeo,sygeo,szgeo Cartesian sun GEO coord.
```

---

g_gei_sun_param

```
subroutine g_gei_sun_param(gmst,slon,sras,sdec)
```

► ***give_sun_parameter*** *dependant of time in GEI system*

```
input :  none
output:  gmst  Greenwich mean sideral time(radians)
         slon  longitude along ecliptic (radians)
         sras   right ascension (radians)
         sdec   declination of the sun (radians)
```

---

g_rocot_version_number

```
subroutine g_rocot_version_number(vernum)
```

► ***give_version_number*** *of the library*

```
input :  none
output:  vernum (ex 1.0)
```

6.5 Description of "Read and check" subroutines

r_coordinate_values

```
subroutine r_coordinate_values(x,y,z,cs)
```

▶ ***read_coordinate*** *values from standard input input and check validity*

read cs character variable from standard input;
if cs is eq to "c", then read Cartesian component of a vector from standard input.
if cs is eq to "s", else read spherical r, t, j components from standard input.

```
input :   cs (char*1)
output:   x,y,z, cs Cartesian components and cs
```

---

recys

```
subroutine recys(csys)
```

▶ ***read_coordinate_system*** *from terminal input and check validity*

csys must have only the following values: gei, geo, mag, sma, gsm, gse, gsq if not, ask again

```
input :   none
output:   csys ; print error if csys is not valid, and ask again
```

---

r_date

```
subroutine r_date(iyear,imonth,iday)
```

▶ ***read_date*** *from standard input and check validity*

test if imonth is not greater than 12, test if iday is not greater then length of the month, taking into account the leap years; year must be greater or equal to 1900.

```
input :   none
output:   iyear, imonth,iday
          print error if date is not valid, and ask again
```

---

r_time

```
subroutine r_time(ih,im,is)
```

▶ ***read_time*** *from standard input and check validity*

ih must be between 1 and 23, im and is between 1 and 59

```
input :   none
output:   ih,im,is
          print error if time is not valid, and ask again
```

6.6 Description of "Print information" subroutines

```
subroutine prinfo
```

▶ ***print_information*** *and version of the library; result is listed in section VI*

```
input :   none
output:   print information on standard output
```

6.7 Description of "Transformations" subroutines

Subroutines are sorted according with the schematic diagram of section 2.2, or the liste given in section 2.3

---

1) t_gei_to_geo

```
subroutine t_gei_to_geo(xgei,ygei,zgei,xgeo,ygeo,zgeo)
```

▶ ***transforms_gei_to_geo :*** *GEI ⟶ GEO system*

```
input :   xgei,ygei,zgei   Cartesian gei coordinates
output:   xgeo,ygeo,zgeo   Cartesian geo coordinates
```

---

t_geo_to_gei

```
subroutine t_geo_to_gei(xgeo,ygeo,zgeo,xgei,ygei,zgei)
```

▶ ***transforms_geo_to_gei :*** *GEO ⟶ GEI system*

```
input :   xgeo,ygeo,zgeo   Cartesian geo coordinates
output:   xgei,ygei,zgei   Cartesian gei coordinates
```

---

2) t_gei_to_mag

```
subroutine t_gei_to_mag(xgei,ygei,zgei,xmag,ymag,zmag)
```

▶ ***transforms_gei_to_mag :*** *GEI ⟶ MAG system*

```
input :   xgei,ygei,zgei   Cartesian gei coordinates
output:   xmag,ymag,zmag   Cartesian mag coordinates
```

---

t_mag_to_gei

```
subroutine t_mag_to_gei(xmag,ymag,zmag,xgei,ygei,zgei)
```

▶ ***transforms_mag_to_gei :*** *MAG ⟶ GEI system*

```
input :   xmag,ymag,zmag    Cartesian mag coordinates
output:   xgei,ygei,zgei   Cartesian gei coordinates
```

---

3) t_gei_to_sm

```
subroutine t_gei_to_sm(xgei,ygei,zgei,xsma,ysma,zsma)
```

▶ ***transforms_gei_to_sma :*** *GEI ⟶ SM system*

```
input :   xgei,ygei,zgei   Cartesian gei coordinates
output:   xsma,ysma,zsma   Cartesian sma coordinates
```

---

t_sm_to_gei

```
subroutine t_sm_to_gei(xsma,ysma,zsma,xgei,ygei,zgei)
```

▶ ***transforms_sma_to_gei :*** *SM ⟶ GEI system*

```
input :   xsma,ysma,zsma   Cartesian sma coordinates
output:   xgei,ygei,zgei    Cartesian gei coordinates
```

---

4) t_gei_to_gsm

```
subroutine t_gei_to_gsm(xgei,ygei,zgei,xgsm,ygsm,zgsm)
```

▶ ***transforms_gei_to_gsm :*** *GEI ⟶ GSM system*

```
input :   xgei,ygei,zgei   Cartesian gei coordinates
output:   xgsm,ygsm,zgsm   Cartesian gsm coordinates
```

---

t_gsm_to_gei

```
subroutine t_gsm_to_gei(xgsm,ygsm,zgsm,xgei,ygei,zgei)
```

▶ ***transforms_gsm_to_gei :*** *GSM ⟶ GEI system*

```
input :   xgsm,ygsm,zgsm    Cartesian gsm coordinates
output:   xgei,ygei,zgei    Cartesian gei coordinates
```

---

5) t_gei_to_gse

```
subroutine t_gei_to_gse(xgei,ygei,zgei,xgse,ygse,zgse)
```

▶ ***transforms_gei_to_gse :*** *GEI ⟶ GSE system*

```
input :   xgei,ygei,zgei   Cartesian gei coordinates
output:   xgse,ygse,zgse   Cartesian gse coordinates
```

---

t_gse_to_gei

```
subroutine t_gse_to_gei(xgse,ygse,zgse,xgei,ygei,zgei)
```

▶ ***transforms_gse_to_gei :*** *GSE ⟶ GEI system*

```
input :   xgse,ygse,zgse    Cartesian gse coordinates
output:   xgei,ygei,zgei    Cartesian gei coordinates
```

---

6) t_gei_to_gseq

```
subroutine t_gei_to_gseq(xgei,ygei,zgei,xgsq,ygsq,zgsq)
```

▶ ***transforms_gei_to_gsq :*** *GEI ⟶ GSEQ system*

```
input :   xgei,ygei,zgei   Cartesian gei coordinates
output:   xgsq,ygsq,zgsq   Cartesian gsq coordinates
```

---

t_gseq_to_gei

```
subroutine t_gseq_to_gei(xgsq,ygsq,zgsq,xgei,ygei,zgei)
```

▶ ***transforms_gsq_to_gei :*** *GSEQ ⟶ GEI system*

```
input :   xgsq,ygsq,zgsq    Cartesian gsq coordinates
output:   xgei,ygei,zgei    Cartesian gei coordinates
```

---

7) t_geo_to_sm

```
subroutine t_geo_to_sm(xgeo,ygeo,zgeo,xsma,ysma,zsma)
```

▶ ***transforms_geo_to_sma :*** *GEO ⟶ SM system*

```
input :   xgeo,ygeo,zgeo   Cartesian geo coordinates
output:   xsma,ysma,zsma   Cartesian sma coordinates
```

---

t_sm_to_geo

```
subroutine t_sm_to_geo(xsma,ysma,zsma,xgeo,ygeo,zgeo)
```

▶ ***transforms_sma_to_geo :*** *SM ⟶ GEO system*

```
input :   xsma,ysma,zsma   Cartesian sma coordinates
output:   xgeo,ygeo,zgeo   Cartesian geo coordinates
```

---

8) t_geo_to_gsm

```
subroutine t_geo_to_gsm(xgeo,ygeo,zgeo,xgsm,ygsm,zgsm)
```

▶ ***transforms_geo_to_gsm :*** *GEO ⟶ GSM system*

```
input :   xgeo,ygeo,zgeo   Cartesian geo coordinates
output:   xgsm,ygsm,zgsm   Cartesian gsm coordinates
```

---

t_gsm_to_geo

```
subroutine t_gsm_to_geo(xgsm,ygsm,zgsm,xgeo,ygeo,zgeo)
```

▶ ***transforms_gsm_to_geo :*** *GSM ⟶ GEO system*

```
input :   xgsm,ygsm,zgsm    Cartesian gsm coordinates
output:   xgeo,ygeo,zgeo    Cartesian geo coordinates
```

---

9) t_geo_to_gse

```
subroutine t_geo_to_gse(xgeo,ygeo,zgeo,xgse,ygse,zgse)
```

▶ ***transforms_geo_to_gse :*** *GEO ⟶ GSE system*

```
input :   xgeo,ygeo,zgeo   Cartesian geo coordinates
output:   xgse,ygse,zgse   Cartesian gse coordinates
```

---

t_gse_to_geo

```
subroutine t_gse_to_geo(xgse,ygse,zgse,xgeo,ygeo,zgeo)
```

▶ ***transforms_gse_to_geo :*** *GSE ⟶ GEO system*

```
input :   xgse,ygse,zgse   Cartesian gse coordinates
output:   xgeo,ygeo,zgeo   Cartesian geo coordinates
```

---

10) t_geo_to_gseq

```
subroutine t_geo_to_gseq(xgeo,ygeo,zgeo,xgsq,ygsq,zgsq)
```

▶ ***transforms_geo_to_gsq :*** *GEO ⟶ GSEQ system*

```
input :   xgeo,ygeo,zgeo   Cartesian geo coordinates
output:   xgsq,ygsq,zgsq   Cartesian gsq coordinates
```

t_gseq_to_geo

```
subroutine t_gseq_to_geo(xgsq,ygsq,zgsq,xgeo,ygeo,zgeo)
```

▶ ***transforms_gsq_to_geo :*** *GSEQ ⟶ GEO system*

```
input :   xgsq,ygsq,zgsq    Cartesian gsq coordinates
output:   xgeo,ygeo,zgeo    Cartesian geo coordinates
```

11) t_gse_to_gseq

```
subroutine t_gse_to_gseq(xgse,ygse,zgse,xgsq,ygsq,zgsq)
```

▶ ***transforms_gse_to_gsq :*** *GSE ⟶ GSEQ system*

```
input :   xgse,ygse,zgse   Cartesian gse coordinates
output:   xgsq,ygsq,zgsq   Cartesian gsq coordinates
```

t_gseq_to_gse

```
subroutine t_gseq_to_gse(xgsq,ygsq,zgsq,xgse,ygse,zgse)
```

▶ ***transforms_gsq_to_gse :*** *GSEQ ⟶ GSE system*

```
input :   xgsq,ygsq,zgsq   Cartesian gsq coordinates
output:   xgse,ygse,zgse   Cartesian gse coordinates
```

12) t_gse_to_gsm

```
subroutine t_gse_to_gsm(xgse,ygse,zgse,xgsm,ygsm,zgsm)
```

▶ ***transforms_gse_to_gsm :*** *GSE ⟶ GSM system*

```
input :   xgse,ygse,zgse    Cartesian gse coordinates
output:   xgsm,ygsm,zgsm    Cartesian gsm coordinates
```

t_gsm_to_gse

```
subroutine t_gsm_to_gse(xgsm,ygsm,zgsm,xgse,ygse,zgse)
```

▶ ***transforms_gsm_to_gse :*** *GSM ⟶ GSE system*

```
input :   xgsm,ygsm,zgsm   Cartesian gsm coordinates
output:   xgse,ygse,zgse   Cartesian gse coordinates
```

13) t_gsm_to_sm

```
subroutine t_gsm_to_sm(xgsm,ygsm,zgsm,xsma,ysma,zsma)
```

▶ ***transforms_gsm_to_sma :*** *GSM ⟶ SM system*

```
input :   xgsm,ygsm,zgsm   Cartesian gsm coordinates
output:   xsma,ysma,zsma   Cartesian sma coordinates
```

t_sm_to_gsm

```
subroutine t_sm_to_gsm(xsma,ysma,zsma,xgsm,ygsm,zgsm)
```

▶ ***transforms_sma_to_gsm :*** *SM ⟶ GSM system*

```
input :   xsma,ysma,zsma   Cartesian sma coordinates
output:   xgsm,ygsm,zgsm   Cartesian gsm coordinates
```

14) t_sm_to_mag

```
subroutine t_sm_to_mag(xsma,ysma,zsma,xmag,ymag,zmag)
```

▶ ***transforms_sma_to_mag :*** *SM ⟶ MAG system*

```
input :   xsma,ysma,zsma   Cartesian sma coordinates
output:   xmag,ymag,zmag   Cartesian mag coordinates
```

t_mag_to_sm

```
subroutine t_mag_to_sm(xmag,ymag,zmag,xsma,ysma,zsma)
```

▶ ***transforms_mag_to_sma :*** *MAG ⟶ SM system*

```
input :   xmag,ymag,zmag   Cartesian mag coordinates
output:   xsma,ysma,zsma   Cartesian sma coordinates
```

15) t_geo_to_mag

```
subroutine t_geo_to_mag(xgeo,ygeo,zgeo,xmag,ymag,zmag)
```

▶ ***transforms_geo_to_mag :*** *GEO ⟶ MAG system*

```
input :   xgeo,ygeo,zgeo   Cartesian geo coordinates
output:   xmag,ymag,zmag   Cartesian mag coordinates
```

t_mag_to_geo

```
subroutine t_mag_to_geo(xmag,ymag,zmag,xgeo,ygeo,zgeo)
```

▶ ***transforms_mag_to_geo :*** *MAG ⟶ GEO system*

```
input :   xmag,ymag,zmag   Cartesian mag coordinates
output:   xgeo,ygeo,zgeo   Cartesian geo coordinates
```

---

16) tgsesma

```
subroutine tgsesma(xgse,ygse,zgse,xsma,ysma,zsma)
```

▶ ***transforms_gse_to_sma :*** *GSE ⟶ SMA system*

```
input :   xgse,ygse,zgse   Cartesian gse coordinates
output:   xsma,ysma,zsma   Cartesian sma coordinates
```

---

tsmagse

```
subroutine tsmagse(xsma,ysma,zsma,xgse,ygse,zgse)
```

▶ ***transforms_sma_to_gse :*** *SM ⟶ GSE system*

```
input :   xsma,ysma,zsma   Cartesian sma coordinates
output:   xgse,ygse,zgse   Cartesian gse coordinates
```

---

17) t_gsm_to_mag

```
subroutine t_gsm_to_mag(xgsm,ygsm,zgsm,xmag,ymag,zmag)
```

▶ ***transforms_gsm_to_mag :*** *GSM ⟶ MAG system (via GSM⟶SM⟶MAG)*

```
input :   xgsm,ygsm,zgsm   Cartesian gsm coordinates
output:   xmag,ymag,zmag   Cartesian mag coordinates
```

---

t_mag_to_gsm

```
subroutine t_mag_to_gsm(xmag,ymag,zmag,xgsm,ygsm,zgsm)
```

▶ ***transforms_mag_to_gsm :*** *MAG ⟶ GSM system (via MAG⟶SM⟶GSM)*

```
input :   xmag,ymag,zmag   Cartesian mag coordinates
output:   xgsm,ygsm,zgsm   Cartesian gsm coordinates
```

---

18) t_gsm_to_gseq

```
subroutine t_gsm_to_gseq(xgsm,ygsm,zgsm,xgsq,ygsq,zgsq)
```

▶ ***transforms_gsm_to_gsq :*** *GSM ⟶ GSQ system (via GSM⟶GSE⟶GSQ)*

```
input :   xgsm,ygsm,zgsm   Cartesian gsm coordinates
output:   xgsq,ygsq,zgsq   Cartesian gsq coordinates
```

---

t_gseq_to_gsm

```
subroutine t_gseq_to_gsm(xgsq,ygsq,zgsq,xgsm,ygsm,zgsm)
```

▶ ***transforms_gsq_to_gsm :*** *GSQ ⟶ GSM system (via GSQ⟶GSE⟶GSM)*

```
input :   xgsq,ygsq,zgsq   Cartesian gsq coordinates
output:   xgsm,ygsm,zgsm   Cartesian gsm coordinates
```

---

19) t_geo_to_dm

```
subroutine t_geo_to_dm(xgeo,ygeo,zgeo,rlat,rlong,xdme,ydme,zdme)
```

▶ ***transforms_geo_to_dme :*** *GEO ⟶ DM system*

```
input :   xgeo,ygeo,zgeo    Cartesian geo coordinates
          rlat,rlong        latitude and longitude of the
                            point of observation (radians)
output:   xdme,ydme,zdme    Cartesian dme coordinates
```

---

t_dm_to_geo

```
subroutine t_dm_to_geo(xdme,ydme,zdme,rlat,rlong,xgeo,ygeo,zgeo)
```

▶ ***transforms_dme_to_geo :*** *DM ⟶ GEO system*

```
 input:   xdme,ydme,zdme   Cartesian dme coordinates
          rlat,rlong       latitude and longitude of the
                           point of observation (radians)
output:   xgeo,ygeo,zgeo   Cartesian geo coordinates
```

---

20) t_geo_to_vdh

```
subroutine t_geo_to_vdh(xgeo,ygeo,zgeo,rlat,rlong,xvdh,yvdh,zvdh)
```

► ***transforms_geo_to_vdh :*** *GEO ⟶ VDH system*

input :	xgeo,ygeo,zgeo	Cartesian geo coordinates
	rlat,rlong	latitude and longitude of the point of observation (radians)
output:	xvdh,yvdh,zvdh	Cartesian vdh coordinates

t_vdh_to_geo

```
subroutine t_vdh_to_geo(xvdh,yvdh,zvdh,rlat,rlong,xgeo,ygeo,zgeo)
```

► ***transforms_vdh_to_geo :*** *VDH ⟶ GEO system*

input:	xvdh,yvdh,zvdh	Cartesian vdh coordinates
	rlat,rlong	latitude and longitude of the point of observation (radians)
output:	xgeo,ygeo,zgeo	Cartesian geo coordinates

21) t_sr2_to_sr

```
subroutine t_sr2_to_sr(xsr2,ysr2,spifre,spipha,deltaT,xsre,ysre)
```

► ***transforms_sr2_to_sre :*** *SR2 ⟶ SR system*

input :	xsr2, ysr2	cartesian sr2 coordinates
	spifre	spin frequency in Hz
	spipha	spin phase in radians, growing with time spipha= positive angle between the xsr axis and the component of the direction of the Sun in the xsr-ysr plane.
	deltaT	(T -To) (sec.), between the current time & the time where is measured the spin phase
output:	xsre,ysre	cartesian sr coordinates
Comment:	Z component	is unchanged (spin axis)

t_sr_to_sr2

```
subroutine t_sr_to_sr2(xsre,ysre,spifre,spipha,deltaT,xsr2,ysr2)
```

► ***transforms_sre_to_sr2 :*** *SR ⟶ SR2 system*

input :	xsre,ysre	cartesian sr coordinates
	spifre	spin frequency in Hz
	spipha	spin phase in radians, growing with time spipha= positive angle between the xsr axis and the component of the direction of the Sun in the xsr-ysr plane.
	deltaT	(T -To) (sec.), between the current time & the time where is measured the spin phase
output:	xsr2, ysr2	cartesian sr2 coordinates
Comment:	Z component	is unchanged (spin axis)

22) t_gse_to_sr2

```
subroutine t_gse_to_sr2(xgse,ygse,zgse,rotx,roty,rotz,xsr2,ysr2,zsr2)
```

► ***transforms_gse_to_sr2 :*** *GSE ⟶ SR2 system*

input :	xgse,ygse,zgse	Cartesian gse coordinates
	rotx,roty,rotz	Cartesian gse coordinates of the rotation axis of the S/C
output:	xsr2,ysr2,zsr2	Cartesian sr2 coordinates

t_sr2_to_gse

```
subroutine t_sr2_to_gse(xsr2,ysr2,zsr2,rotx,roty,rotz,xgse,ygse,zgse)
```

► ***transforms_sr2_to_gse :*** *VDH ⟶ GEO system*

input :	xsr2,ysr2,zsr2	Cartesian sr2 coordinates
	rotx,rot,y,rotz	Cartesian gse coordinates of the rotation axis of the S/C
output:	xgse,ygse,zgse	Cartesian gse coordinates

23) t_sr2_to_mfa

```
subroutine t_sr2_to_mfa(xsr2,ysr2,zsr2,bx,by,bz,rotx,roty,rotz,
xmfa,ymfa,zmfa)
```

▶ ***transforms_sr2_to_mfa :*** *SR2 ⟶ MFA system*

input :	xsr2,ysr2,zsr2	Cartesian SR2 coordinates
	bx,by,bz	Cartesian SR2 coordinates of the DC magnetic field
	rotx,roty,rotz	Cartesian GSE coordinates of the rotation axis of the S/C
output:	xmfa,ymfa,zmfa	Cartesian MFA coordinates

24) t_gse_to_mfa

```
subroutine t_gse_to_mfa(xgse,ygse,zgse,bx,by,bz,xmfa,ymfa,zmfa)
```

▶ ***transforms_gse_to_mfa :*** *GSE ⟶ MFA system*

input :	xgse,ygse,zgse	Cartesian SR2 coordinates
	bx,by,bz	Cartesian GSE coordinates of the DC magnetic field
output:	xmfa,ymfa,zmfa	Cartesian MFA coordinates

24) t_gse_to_tpn

```
subroutine t_gse_to_tpn(xgse,ygse,zgse,xo,yo,zo,xs,xtpn,ytpn,ztpn)
```

▶ ***transforms_gse_to_tpn :*** *GSE ⟶ TPN system*

input :	xgse,ygse,zgse	Cartesian GSE coordinates
	xo,yo,zo	position of the S/C in gse
	xs	subsolar point, submit of the paraboloid from Earth to Sun
output:	xtpn,ytpn,ztpn	Cartesian TPN coordinates

25) t_gsm_to_tpn

```
subroutine t_gsm_to_tpn(xgsm,ygsm,zgsm,xo,yo,zo,xs,xtpn,ytpn,ztpn)
```

▶ ***transforms_gsm_to_tpn :*** *GSM ⟶ TPN system*

input :	xgsm,ygsm,zgsm	Cartesian GSM coordinates
	xo,yo,zo	position of the S/C in gsm
	xs	subsolar point, submit of the paraboloid from Earth to Sun
output:	xtpn,ytpn,ztpn	Cartesian TPN coordinates

• Transformation with Euler'angles:

t_xyz_to_vdh

```
subroutine t_xyz_to_vdh(,y,z,a1,a2,a3,v,d,h)
```

▶ ***transforms_xyz_to_vdh :*** *spinning XYZ ⟶ fixed VDH system*

input :	x,y,z	Cartesian XYZ coordinates
	a1,a2,a3	Euler's angles in degree
output:	v,d,h	Cartesian VDH coordinates

t_vdh_to_xyz

```
subroutine t_vdh_to_xyz(,y,z,a1,a2,a3,v,d,h)
```

▶ ***transforms_vdh_to_xyz :*** *fixed VDH ⟶ spinning XYZ system*

input :	v,d,h	Cartesian VDH coordinates
	a1,a2,a3	Euler's angles in degree
output:	x,y,z	Cartesian XYZ coordinates

• Conversion spherical to cartesian and vice versa:

---

t_car_to_sph

```
subroutine t_car_to_sph(x,y,z,r,teta,phi)
```

► ***transforms_car_to_sph :*** *CAR* $\longrightarrow$ *SPH system*

```
input :   x,y,z        Cartesian coordinates
output:   r,teta,phi   spherical coordinates (radians)
```

---

t_sph_to_car

```
subroutine t_sph_to_car(r,teta,phi,x,y,z)
```

► ***transforms_sph_to_car :*** *SPH* $\longrightarrow$ *CAR system*

```
input :   r,teta,phi   spherical coordinates (radians)
output:   x,y,z        Cartesian coordinates
```

6.8 Description of "Matrix operations" subroutines

This part concern matrix operations allowing the transformation of an input 3D signal into the minimum variance coordinate (MVA Analysis). All codes in this part have been extracted of Roproc Software, V 4.5 and rewrited properly in f77, with Rocotlib V2.2 conventions in april 2016.

---

mat_covarmin

```
subroutine mat_covarmin(ifc,Vx,Vy,Vz,n,irep,covar,lambda,eigvec)
```

▶ ***compute_var_min :*** *compute variance minimum coordinates of a signal Vx, Vy, Vz*

```
input :   ifc                 unit for writing results
          Vx(n),Vy(n),Vz(n)   input signal
          n                   number of point of the signal
output:   covar(3,3)          covariance matrix
          lamda(3)            eigen values
          eigvev(3,3)         eigen vectors
```

---

mat_covariance

```
subroutine mat_covariance(Vx,Vy,Vz,n,covar)
```

▶ ***compute_covariance :*** *compute covariance matrix for a vector series V(n)*

```
input :   Vx(n),Vy(n),Vz(n)   vector series
          n                   number of point of the signal
output:   covar(3,3)          covariance matrix
```

---

mat_diagonalise

```
subroutine mat_diagonalise(mat,lambda,eigvec)
```

▶ ***diagonalise*** *the given matrix mat(3,3)*

```
input :   mat(3,3)      matrix to diagonalise
output:   lambda(3)     eigen values
          eigvec(3,3)   eigen vectors normalised to 1.
```

---

mat_checkortho

```
subroutine mat_checkortho(ifc,mat)
```

▶ ***check orthogonality*** *of matrix components*

```
input :   ifc           unit for writing results
          mat(3,3)      matrix to check
output:   on unit ifc
```

---

mat_determin

```
subroutine mat_determin(mat,det)
```

▶ ***compute determinant*** *of the given matrix*

```
input :   mat(3,3)   input matrix
output:   det        determinant value
```

---

mat_eigenvec

```
subroutine mat_eigenvec(mat,lambda,eigvec)
```

▶ ***compute_eigen vectors*** *and eigen values of real mat(3,3)*

```
input :   mat(3,3)      must be real and symmetrical
                        Method used is Householder
output:   lamda(3)      eigen values
          eigvec(3,3)   eigen vectors
```

---

mat_fpythag

```
subroutine mat_fpythag(a,b,fpyth)
```

▶ ***Pythagore function*** *of two real (used by mat_eigenvec)*

```
input :   a,b     arguments of the function
output:   fpyth   Pythagore function
```

---

mat_normavec

```
subroutine mat_normavec(mat)
```
► ***normalize_vectors:*** *normalise to 1 the vectors of the input matrix*
```
input :   mat(3,3)   input matrix
output:   mat(3,3)   matrix with vectors normalized to 1.
```

mat_product

```
subroutine mat_product(mat1,mat2,mat3)
```
► ***product*** *of two given matrix of dim. 3x3*
```
input :   mat1(3,3)
          mat2(3,3)
output:   mat3(3,3)   Mat3=Mat1*Mat2
```

mat_somme

```
subroutine mat_somme(mat1,mat2,mat3)
```
► ***somme*** *of two given matrix of dim. 3x3*
```
input :   mat1(3,3)
          mat2(3,3)
output:   mat3(3,3)   Mat3=Mat1 + Mat2
```

mat_diff

```
subroutine mat_diff(mat1,mat2,mat3)
```
► ***difference*** *of two given matrix*
```
input :   mat1(3,3)
          mat2(3,3)
output:   mat3(3,3)   Mat3=Mat1 - Mat2
```

mat_transpose

```
subroutine mat_transpose(mat)
```
► ***transpose*** *input matrix*
```
input :   mat(3,3)   matrix to transpose
output:   mat(3,3)   transposed matrix
```

mat_changecoord

```
subroutine mat_changecoord(mat,Vx,Vy,Vz,n)
```
► ***change_coordinate*** *of a vector series with a given matrix*
```
input :   mat(3,3)            coordinate transform matrix
          Vx(n),Vy(n),Vz(n)   vector series to transform
          n                   dimension of
```
$\vec{V}$
```
output:   Vx(n),Vy(n),Vz(n)   vector series transformed as
```
$\vec{V} = mat * \vec{V}$

mat_write

```
subroutine mat_write(ifc,com,mat)
```
► ***write*** *on ifc unit the mat(3,3) with a comment*
```
input :   ifc          file unit
          com          coment string
          mat(3,3)     matrix to be writed
output:   on unit ifc
```

mat_writeigen

```
subroutine mat_writeigen(ifc,lambda,mat)
```
► ***write*** *on ifc unit the eigen values and eigen vectors of mat(3,3)*
```
input :   ifc          file unit
          lamda(3)     eigen values to be writed
          mat(3,3)     matrix containing eigen vectors
output:   on unit ifc
```

6.9 Summary of available subroutine

6.9.1 "Basic computation" subroutines

subroutine	arguments	object
cp_angle_and_ratio	(ux,uy,uz, vx,vy,vz, angle,ratio)	compute_angle_and_ratio beetween U and V vectors
cp_Euler_interpol	(a1,b1,c1,a2,b2,c2,ti,dt,ai,bi,ci)	compute_Euler_angles_interpolation
cp_geo_dipole_dir	(iyear,idoy, d1,d2,d3)	compute_dipole_direction in GEO system
cp_gei_sun_dir	(iyear,idoy,ih,im,is, gst,slong, sra,sdec,obliq)	compute_sun_direction in GEI system
cp_sunrise_sunset	(iyear,imon,iday,rlat,rlon,tmer, tris,tset,durd)	compute_sunset_time and others parameters
cp_tpn_param	(xo,yo,zo,xs, Tx,Ty,Tz, Px,Py,Pz, Nx,Ny,Nz)	compute_TPN_parameters in GSE system or any system having X axis towards the SUN.
cp_time_param	(iyear,imonth,iday,ih,im,is)	compute_time_parameters and time-dependent matrix
cp_time_param2	(jd1950, houday)	compute_time_parameters and time-dependent matrix
cp_time_param3	(jd2000, houday)	compute_time_parameters and time-dependent matrix

6.9.2 "Calendar" subroutines

subroutine	arguments	object
cp_nbday_in_month	(iyear,imonth,nbday)	compute_number_of_day_of_the_month
cp_fr_day_name	(iday,cday,nbcha)	compute_french_day_name, ex: 'Lundi' for iday=1
cp_fr_month_name	(imonth,cmonth,nchar)	compute_French_month_name
cp_en_day_name	(iday,cday,nbcha)	compute_US_day_name, ex: 'Monday' for iday=1
cp_en_month_name	(imonth,cmonth,nchar)	compute_US_month_name
cp_leap_year	(iyear,ileap)	compute_seasons, i.e. solstice & equinox
cp_seasons	(iyear,id_sso,id_wso,id_seq,id_feq) ct_sso,ct_wso, ct_seq,ct_feq)	compute_leap_year with ileap=1 for leap year, 0 if not

cv_doty_to_date	(idoy,iyear,imonth,iday)	convert_day_of_year_to_date and for a given year
cv_jul2000_to_date	(jd00,iyear,imonth,iday)	convert_julian_day_2000_to_date with jd00=0 for jan. 1
cv_jul1950_to_date	(jd50,iyear,imonth,iday)	convert_julian_day_1950_to_date with jd50=0 for jan. 1
cv_weekn_to_date	(iweek,iyear,imonth,iday)	convert_week_number_to_date for first day of the week
cv_date_to_dotw	(iyear,imonth,iday,idow)	convert_date_to_day_of_the_week
cv_date_to_doty	(iyear,imonth,iday,idoy)	convert_date_to_day_of_year with idoy=1 for january 1
cv_hms_to_dech	(ih,im,is,houday)	convert_hours_minutes_seconds_to_decimal_hour_of_day
cv_date_to_jul1950	(iyear,imonth,iday,jd50)	convert_date_to_julian_day_1950 jd50=0 for january 1, 1950
cv_date_to_jul2000	(iyear,imonth,iday,jd00)	convert_date_to_julian_day_2000 jd00=0 for january 1, 2000
cv_dhms_to_msotd	(ih,im,is,ims,milday)	convert_decimal_hours_minutes_seconds_to_millisec_of_day
cv_dech_to_hms	(houday,ih,im,is)	convert_decimal_hour_to_hours_minutes_seconds
cv_msotd_to_hmsms	(milday,ih,im,is,ims)	convert_millisec._of_the_day_to_hours_minutes_seconds
cv_date_to_weekn	(iyear,imonth,iday,iweek)	convert_date_to_week_number

6.9.3 "Read and check" subroutines

subroutine	arguments	object
r_coordinate_values	(x,y,z,cs)	read_coordinate values from input
r_coordinate_system	(csys)	read_coordinate system from input & check validity
r_date	(iyear,imonth,iday)	read_date from input and check validity
r_time	(ih,im,is)	read_time from input and check validity

6.9.4 "Give" subroutines

subroutine	arguments	object
g_gei_geo_dipole_dir	(dxgei,dygei,dzgei,dxgeo,dygeo,dzgeo)	give_dipole_direction in GEI and GEO system
g_gsm_dipole_tilt_angle	(diptan)	give_dipole_tilt_angle in radians
g_gei_geo_ecliptic_dir	(exgei,eygei,ezgei,exgeo,eygeo,ezgeo)	give_ecliptic_direction in GEI and GEO system
g_gei_geo_sun_rot	(rxgei,rygei,rzgei,rxgeo,rygeo,rzgeo)	give_sun_rotation_direction in GEI and GEO system
g_gei_geo_sun_dir	(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)	give_sun_direction in GEI and GEO system
g_gei_sun_param	(gmst,slon,sras,sdec,obli)	give_sun_parameter dependent of time in GEI system
g_rocot_version_number	(vernum,verdat)	give_version_number and last update of the library

6.9.5 "Matrix operations" subroutines

mat_cp_varmin	(ifc,Vx,Vy,Vz,n,irep, covar,lambda,eigvec)	compute variance minimum coordinate of a signal Vx,Vy,Vz of dimension n
mat_cp_covariance	(Vx,Vy,Vz,n,covar)	compute covariance matrix for a vector series V(n)
mat_diagonalise	(mat,lambda,eigvec)	diagonalise the given matrix mat(3,3)
mat_check_ortho	(ifc,mat)	check orthogonality of matrix components
mat_cp_determin	(mat,det)	compute determinant of the given matrix
mat_cp_eigen_vec	(mat,lambda,eigvec)	compute eigen vectors and eigen values of real mat(3,3)
mat_cp_pythag_func	(a,b,fpyth)	Pythagore function of two real (used by mat_cp_eigen_vec)
mat_normalize_vec	(mat)	normalize to 1. the vectors of the input matrix
mat_product	(mat1,mat2,mat3)	matrix product of two given matrix of dim. 3
mat_somme	(mat1,mat2,mat3)	matrix somme of two given matrix of dim. 3
mat_diff	(mat1,mat2,mat3)	matrix difference of two given matrix
mat_transpose	(mat)	transpose input matrix
mat_change_coord	(mat,Vx,Vy,Vz,n)	change coordinate of a vector serie with a given matrix
mat_write	(ifc,com,mat)	print on ifc unit mat(3,3) with a comment
mat_write_eigen_vec	(ifc,lambda,mat)	print on ifc unit eigen values & vectors of mat(3,3)

6.9.6 "Print informations" subroutine

subroutine	arguments	object
print_rocot_info		print_library_informations

6.9.7 "Transform" subroutines

subroutine	arguments	object
t_car_to_sph	(x,y,z,r,teta,phi)	transforms_car_to_sph: CAR ⟶ SPH system
t_dm_to_geo	(xdme,ydme,zdme,rlat,rlong,xgeo,ygeo,zgeo)	transforms_dme_to_geo: DM ⟶ GEO system
t_gei_to_geo	(xgei,ygei,zgei,xgeo,ygeo,zgeo)	transforms_gei_to_geo: GEI ⟶ GEO system
t_gei_to_gse	(xgei,ygei,zgei,xgse,ygse,zgse)	transforms_gei_to_gse: GEI ⟶ GSE system
t_gei_to_gsm	(xgei,ygei,zgei,xgsm,ygsm,zgsm)	transforms_gei_to_gsm: GEI ⟶ GSM system
t_gei_to_gseq	(xgei,ygei,zgei,xgsq,ygsq,zgsq)	transforms_gei_to_gsq: GEI ⟶ GSEQ system
t_gei_to_mag	(xgei,ygei,zgei,xmag,ymag,zmag)	transforms_gei_to_mag: GEI ⟶ MAG system
t_gei_to_sm	(xgei,ygei,zgei,xsma,ysma,zsma)	transforms_gei_to_sma: GEI ⟶ SM system
t_geo_to_dm	(xgeo,ygeo,zgeo,rlat,rlong,xdme,ydme,zdme)	transforms_geo_to_dme: GEO ⟶ DM system
t_geo_to_gei	(xgeo,ygeo,zgeo,xgei,ygei,zgei)	transforms_geo_to_gei: GEO ⟶ GEI system
t_geo_to_gse	(xgeo,ygeo,zgeo,xgse,ygse,zgse)	transforms_geo_to_gse: GEO ⟶ GSE system
t_geo_to_gsm	(xgeo,ygeo,zgeo,xgsm,ygsm,zgsm)	transforms_geo_to_gsm: GEO ⟶ GSM system
t_geo_to_gseq	(xgeo,ygeo,zgeo,xgsq,ygsq,zgsq)	transforms_geo_to_gsq: GEO ⟶ GSEQ system
t_geo_to_mag	(xgeo,ygeo,zgeo,xmag,ymag,zmag)	transforms_geo_to_mag: GEO ⟶ MAG system
t_geo_to_sm	(xgeo,ygeo,zgeo,xsma,ysma,zsma)	transforms_geo_to_sma: GEO ⟶ SM system
t_geo_to_vdh	(xgeo,ygeo,zgeo,rlat,rlong,xvdh,yvdh,zvdh)	transforms_geo_to_vdh: GEO ⟶ VDH system
t_gse_to_gei	(xgse,ygse,zgse,xgei,ygei,zgei)	transforms_gse_to_gei: GSE ⟶ GEI system
t_gse_to_geo	(xgse,ygse,zgse,xgeo,ygeo,zgeo)	transforms_gse_to_geo: GSE ⟶ GEO system
t_gse_to_gsm	(xgse,ygse,zgse,xgsm,ygsm,zgsm)	transforms_gse_to_gsm: GSE ⟶ GSM system
t_gse_to_gseq	(xgse,ygse,zgse,xgsq,ygsq,zgsq)	transforms_gse_to_gsq: GSE ⟶ GSEQ system
t_gse_to_mfa	(xgse,ygse,zgse,bx,by,bz,xmfa,ymfa,zmfa)	transforms_gse_to_mfa: GSE ⟶ MFA system
t_gse_to_sr2	(xgse,ygse,zgse,rotx,roty,rotz,xsr2,ysr2,zsr2)	transforms_gse_to_sr2: GSE ⟶ SR2 system
t_gse_to_tpn	(xgse,ygse,zgse,xo,yo,zo,xs,xtpn,ytpn,ztpn)	transforms_gse_to_tpn: GSE ⟶ TPN system
t_gsm_to_gei	(xgsm,ygsm,zgsm,xgei,ygei,zgei)	transforms_gsm_to_gei: GSM ⟶ GEI system
t_gsm_to_geo	(xgsm,ygsm,zgsm,xgeo,ygeo,zgeo)	transforms_gsm_to_geo: GSM ⟶ GEO system
t_gsm_to_gse	(xgsm,ygsm,zgsm,xgse,ygse,zgse)	transforms_gsm_to_gse: GSM ⟶ GSE system
t_gsm_to_gseq	(xgsm,ygsm,zgsm,xgsq,ygsq,zgsq)	transforms_gsm_to_gsq: GSM ⟶ GSQ system
t_gsm_to_mag	(xgsm,ygsm,zgsm,xmag,ymag,zmag)	transforms_gsm_to_mag: GSM ⟶ MAG system
t_gsm_to_sm	(xgsm,ygsm,zgsm,xsma,ysma,zsma)	transforms_gsm_to_sma: GSM ⟶ SM system
t_gsm_to_tpn	(xgsm,ygsm,zgsm,xo,yo,zo,xs,xtpn,ytpn,ztpn)	transforms_gsm_to_tpn: GSM ⟶ TPN system
t_gseq_to_gei	(xgsq,ygsq,zgsq,xgei,ygei,zgei)	transforms_gsq_to_gei: GSEQ ⟶ GEI system
t_gseq_to_geo	(xgsq,ygsq,zgsq,xgeo,ygeo,zgeo)	transforms_gsq_to_geo: GSEQ ⟶ GEO system
t_gseq_to_gse	(xgsq,ygsq,zgsq,xgse,ygse,zgse)	transforms_gsq_to_gse: GSEQ ⟶ GSE system
t_gseq_to_gsm	(xgsq,ygsq,zgsq,xgsm,ygsm,zgsm)	transforms_gsq_to_gsm: GSQ ⟶ GSM system
t_mag_to_gei	(xmag,ymag,zmag,xgei,ygei,zgei)	transforms_mag_to_gei: MAG ⟶ GEI system
t_mag_to_geo	(xmag,ymag,zmag,xgeo,ygeo,zgeo)	transforms_mag_to_geo: MAG ⟶ GEO system
t_mag_to_gsm	(xmag,ymag,zmag,xgsm,ygsm,zgsm)	transforms_mag_to_gsm: MAG ⟶ GSM system
t_mag_to_sm	(xmag,ymag,zmag,xsma,ysma,zsma)	transforms_mag_to_sma: MAG ⟶ SM system
t_sm_to_gei	(xsma,ysma,zsma,xgei,ygei,zgei)	transforms_sma_to_gei: SM ⟶ GEI system
t_sm_to_geo	(xsma,ysma,zsma,xgeo,ygeo,zgeo)	transforms_sma_to_geo: SM ⟶ GEO system
t_sm_to_gsm	(xsma,ysma,zsma,xgsm,ygsm,zgsm)	transforms_sma_to_gsm: SM ⟶ GSM system
t_sm_to_mag	(xsma,ysma,zsma,xmag,ymag,zmag)	transforms_sma_to_mag: SM ⟶ MAG system
t_sph_to_car	(r,teta,phi,x,y,z)	transforms_sph_to_car: SPH ⟶ CAR system
t_sr2_to_gse	(xsr2,ysr2,zsr2,rotx,roty,rotz,xgse,ygse,zgse)	transforms_sr2_to_gse: SR2 ⟶ GSE system
t_sr2_to_mfa	(xsr2,ysr2,zsr2,bx,by,bz,rox,roy,roz,xm,ym,zm)	transforms_sr2_to_mfa: SR2 ⟶ MFA system
t_sr2_to_sr	(xsr2,ysr2,spifre,spipha,deltaT,xsre,ysre)	transforms_sr2_to_sr: SR2 ⟶ SRef.system
t_sr_to_sr2	(xsre,ysre,spifre,spipha,deltaT,xsr2,ysr2)	transforms_sre_to_sr2: SR ⟶ SR2 system
t_vdh_to_geo	(xvdh,yvdh,zvdh,rlat,rlong,xgeo,ygeo,zgeo)	transforms_vdh_to_geo: VDH ⟶ GEO system
t_xyz_to_vdh	(x,y,z,a1,a2,a3,v,d,h)	transforms_xyz_to_vdh: XYZ ⟶ VDH system
t_vdh_to_xyz	(v,d,h,a1,a2,a3,x,y,z)	transforms_vdh_to_xyz: VDH ⟶ XYZ system

Chapter 7

USEFUL PROGRAMS

7.1 General remarks

A library is nothing if it is not used to make useful programs. The Rocotlib package is delivered with a few executable programs, which can be useful for a user which do not want write himself its own programs, or which can be used as complementary test of the library.

For instance, the **rocot_example.exe** described in next section or the **rocot_check.exe** program described in section 8.2 and 9.1 and are used to check the portability of the library on user processor and/or compiler.

Other programs in the next sections can also be used for test, as we saw in section 9.4 but a part of them are utility programs, for instance the **rocot_utility.exe** which allow user to do coordinate transformation without write any else programs.

Note that all programs can be run interactively, or in batch mode. For instance, the user can run **bin/co_mag_time.exe** and enter date, time and position and get the result on current display. He can also run the batch **sh/co_mag_time.sh** which read the input parameters from **in/co_mag_time.in** and write the results in **out/co_mag_time.exe**.

A few program does not have corresponding *.in file, such as :

- **co_julday_table.exe**,
- **rocot_info.exe**,
- **test_day_in_week.exe**,
- **test_format.exe**,
- **test_week.exe**.

For theses program, parametres are defined inside the code. Nevertheless, all *.exe program has a corresponding *.sh shell file for batch execution, such as example below:

```
#!/bin/sh

prog=co_mag_time

# ------------------------------------------------------------------------
# run $prog.exe, read parameters in $prog.in,
# put results in $prog.out
# ------------------------------------------------------------------------

dirsh=`dirname $0`
direxe=$dirsh/../bin
dirin=$dirsh/../in
dirout=$dirsh/../out

echo "------------------------------------------------------------------------"
echo "$prog.sh :"
echo "Running $prog.exe with parameters taken from $prog.in"
echo "results will be in $prog.out"
echo "Please wait...."

$direxe/$prog.exe <$dirin/$prog.in >$dirout/$prog.out

echo "$prog.sh  terminated"
echo "$prog.out is located in $dirout/$prog.out"
echo "------------------------------------------------------------------------"
echo
```

7.2 Example of simple programs

7.2.1 rocot_example

This is a very simple program showing how using the library. The source of rocot_example.f (in src directory) is given hereafter in table 7.1.

```
      program rocot_example
c
c     ****************************************************************0**
c *   Rocot_Example: show example of Rocotlib use.
c     P. Robert, July 2000
c     ****************************************************************0**
c
c
      print 100
      print 100, '*************************************************'
      print 100
      print 100, 'ROCOTEXP:'
      print 100
c
      call r_date(iyear,imonth,iday)
      call r_time(ih,im,is)
      call cv_date_to_doty(iyear,imonth,iday,idoy)
c
      print 100
      print 100, 'date: MM/JJ/YY =', imonth,iday,iyear
      print 100, 'time: HH/MM/SS =', ih,im,is
      print 100, 'day of the year=', idoy
c
      call cp_time_param(iyear,imonth,iday,ih,im,is)
      call g_gei_geo_sun_dir(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)
      call t_gei_to_gsm(sxgei,sygei,szgei,sxgsm,sygsm,szgsm)
c
      print 100
      print 200, 'Sun in GEI:',sxgei,sygei,szgei
      print 200, 'Sun in GEO:',sxgeo,sygeo,szgeo
      print 200, 'Sun in GSM:',sxgsm,sygsm,szgsm
c
      call g_gei_geo_dipole_dir(dxgei,dygei,dzgei,dxgeo,dygeo,dzgeo)
      call t_gei_to_gsm(dxgei,dygei,dzgei,dxgsm,dygsm,dzgsm)
      call t_gei_to_mag(dxgei,dygei,dzgei,dxmag,dymag,dzmag)
      call t_gei_to_sm(dxgei,dygei,dzgei,dxsma,dysma,dzsma)
c
      print 100
      print 200, 'Dip in GEI:',dxgei,dygei,dzgei
      print 200, 'Dip in GEO:',dxgeo,dygeo,dzgeo
      print 200, 'Dip in GSM:',dxgsm,dygsm,dzgsm
      print 200, 'Dip in MAG:',dxmag,dymag,dzmag
      print 200, 'Dip in SM :',dxsma,dysma,dzsma
c
      print 100
      print 200, 'xyz in GEO=', -5.5, -12.5, -7.10
c
      call t_geo_to_gsm(-5.5, -12.5, -7.10,xgsm,ygsm,zgsm)
      call t_geo_to_gse(-5.5, -12.5, -7.10,xgse,ygse,zgse)
c
      print 200, 'xyz in GSM:', xgsm, ygsm, zgsm
      print 200, 'xyz in GSE:', xgse, ygse, zgse
c
      print 100
      print 100, '*************************************************'
      print 100
c
  100 format(a,3i6)
  200 format(a,3f14.6)

      stop 'rocot_example.exe: normal termination'
      end
```

Table 7.1: rocot_example.f source code

By running "**sh/rocot_example.sh**" you will get on the terminal:

```
-----------------------------------------------------------------------
rocot_example.sh :
Running rocot_example.exe with parameters taken from rocot_example.in
results will be in rocot_example.out
Please wait....
STOP rocot_example.exe: normal termination statement executed
rocot_example.sh  terminated
rocot_example.out is located in sh/../out/rocot_example.out
-----------------------------------------------------------------------
```

Then the command **cat ./out/rocot_example.out** should display:

```
*****************************************************

ROCOTEXP:

iyear,imonth,iday ? (ex: 1990,10,17)
2007  2  7
hour, minute, second ? (ex: 10,45,50)
7 30 33

date: MM/JJ/YY =     2     7  2007
time: HH/MM/SS =     7    30    33
day of the year=    38

Sun in GEI:        .744703       -.612328       -.265465
Sun in GEO:        .315648        .910985       -.265465
Sun in GSM:       1.000000        .000000        .000000

Dip in GEI:       -.174649        .007003        .984606
Dip in GEO:        .054076       -.166214        .984606
Dip in GSM:       -.395728        .000000        .918368
Dip in MAG:        .000000        .000000       1.000000
Dip in SM :        .000000        .000000       1.000000

xyz in GEO=     -5.500000    -12.500000     -7.100000
xyz in GSM:    -11.238584      -.104493    -10.516336
xyz in GSE:    -11.238584     -2.162290    -10.292170

*****************************************************
```

Table 7.2: rocot_example.out file

and you get the requested calculations.

7.2.2 latlon_to_xyz.exe

Another very short program to convert latitude, longitude and distance in cartesian XYZ coordinates. Source is given in table 7.3 while example of .out file is given below in table 7.4.

```
      program latlon_to_xyz
c
c ----------------------------------------------------------------------
c *   Conversion lat, longitude to x y z
c ----------------------------------------------------------------------
c
      print*, 'lat, long, dist ? (deg. and any units)'
      read *, rlat, rlon, rdis
      print*, rlat, rlon, rdis
c
      rlat=rlat*3.1415927/180.
      rlon=rlon*3.1415927/180.
c
      call t_sph_to_car(rdis,rlat,rlon,x,y,z)
c
      print*, x,y,z
c
      stop 'latlon_to_xyz: normal termination'
      end
c
c ----------------------------------------------------------------------
c
```

Table 7.3: latlon_to_xyz.f source code

```
lat, long, dist ? (deg. and any units)
50.  20.  6000.
4319.07764  1572.01587  3856.72559
```

Table 7.4: latlon_to_xyz.out file

7.2.3 vdh_to_gse.exe

This is a small program to show an example of coordinate transformation. User can clone it to make is own program to run other transformation. Table 7.5 show the code, while table 7.6 show the result with choose input parameters.

```
      program vdh_to_gse
c
c     ------------------------------------------------------------------
c
c *   Rocot_Example of using Rocotlib
c
c     ------------------------------------------------------------------
c
      print 100
      print 100, '*************************************************'
      print 100
      print 100, 'vdh_to_gse:'
      print 100

  100 format(a, 3i5)
  200 format(a, 3f10.5)

c
      call r_date(iyear,imonth,iday)
      call r_time(ih,im,is)

      call cp_time_param(iyear,imonth,iday,ih,im,is)
c
      print 100, 'x,y,z in VDH ?'
      read *, xvdh,yvdh,zvdh
c
      print 100, 'lat et long en degre?'
      read *, rlat, rlon
      rlat=rlat*3.1415927/180.
      rlon=rlon*3.1415927/180.
c
      call t_vdh_to_geo(xvdh,yvdh,zvdh,rlat,rlon,xgeo,ygeo,zgeo)
      call t_geo_to_gse(xgeo,ygeo,zgeo,xgse,ygse,zgse)

      print 100
      print 100, 'date: MM/JJ/YY=', imonth,iday,iyear
      print 100, 'time: HH/MM/SS=', ih,im,is
      print 100
      print 200, 'xyz in VDH:', xvdh,yvdh,zvdh
      print 200, 'xyz in GEO:', xgeo,ygeo,zgeo
      print 200, 'xyz in GSE:', xgse,ygse,zgse
      print 200
      print 200, '*************************************************'
      print 200

      stop 'vdh_to_gse.exe: normal termination'
      end
```

Table 7.5: vdh_to_gse.f source code

```
*************************************************

vdh_to_gse:

iyear,imonth,iday ? (ex: 1990,10,17)
1978  8 28
hour, minute, second ? (ex: 10,45,50)
9 14  0
x,y,z in VDH ?
lat et long en degre?

date: MM/JJ/YY=    8   28 1978
time: HH/MM/SS=    9   14    0

xyz in VDH:    .00000   -.00450    .99990
xyz in GEO:    .00956    .00141    .99986
xyz in GSE:    .17784   -.36412    .91412

*************************************************
```

Table 7.6: vdh_to_gse.out file

7.3 test programs

7.3.1 rocot_check program

This is a complete checking program, described in details in **chapter 9**, used to check the portability of the library, that is to say checking that we obtain the same results on difference machine and compilers.

7.3.2 rocot_check_mva.exe

This program check the validity of the MVA computation. The input signal is a current tube, with an arbitrary direction in the measurement system. The output signal is in MVA system, where the Z axis is the direction of the tube. This implies that the third eigen value is zero, and the third eigen vector is the direction of the tube in the initial coordinate system. Other checks are done on the properties of the covariance matrix. Exemple is given on table 7.7.

7.3.3 visu_check_mva.pro

It exists an IDL procedure in src directory to visualize the content of rocot_check_mva.resu file. By running visu_check_mva.pro within IDL, you can get the Postscript file corresponding to the figure 7.3.3. You can cvheck that the MVA system reduce the 3D initial signal into a 2D one.

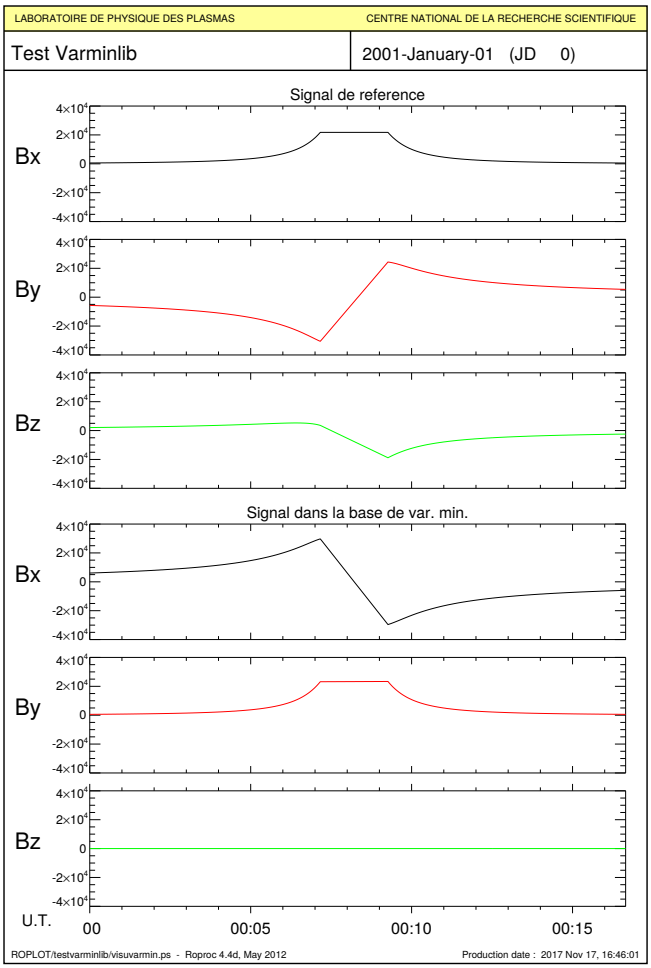


Figure 7.1: Visualisation du fichier test_varminlib.resu par la procedure IDL visu_check_mva.pro

```
=======================
rocot_check_mva program
=======================


Current tube characteristics:
-----------------------------

Radius of the current tube, in km
6000.000

impact parameter (min dist to the curr. tube), in km
4000.000

Current density, in A/km2
0.100

Theta angle of the current direction/ given frame (d)
teta ?
45.000

Phi   angle of the current direction/ given frame (d)
30.000

Current density model ("u" or "g"/ uniform or gauss.)
u


Computation of B vector along the trajectory
--------------------------------------------

nb. of points=     1000
R=  6000.0 km  IP=  4000.0 km J= 0.100 A/km2  tet= 45.0 d.  phi= 30.0 d. mod=u

-------------------------------------------------
Compute Minimum Variance Analaysis coord. system
-------------------------------------------------

covariance matrix of input signal:

 0.35125E+04   -0.17062E+04   -0.21888E+04
-0.17062E+04    0.17030E+05   -0.70373E+04
-0.21888E+04   -0.70373E+04    0.54142E+04

covariance matrix diagonalized:

 0.55875E+04    0.00000E+00    0.00000E+00
 0.00000E+00   -0.63610E-03   -0.30518E-04
 0.00000E+00   -0.30518E-04    0.20369E+05

eigen vectors matrix:

 0.37064E-01    0.78970E+00    0.61237E+00
-0.90628E+00   -0.23163E+00    0.35355E+00
 0.42104E+00   -0.56809E+00    0.70711E+00

Check orthogonality of given matrix:
------------------------------------

1) dot product of any two rows or any two colums must be equal to zero:

V1.V2 =  -0.44703E-07
V2.V3 =  -0.11921E-06
V3.V1 =   0.00000E+00

2) cross product of any two rows or colums must be equal to the third
row or column or its negative):

V1XV2 =     0.61237E+00     0.35355E+00     0.70711E+00
V3    =     0.61237E+00     0.35355E+00     0.70711E+00

V2XV3 =     0.37064E-01    -0.90628E+00     0.42104E+00
V1    =     0.37064E-01    -0.90628E+00     0.42104E+00

V3XV1 =     0.78970E+00    -0.23163E+00    -0.56809E+00
V2    =     0.78970E+00    -0.23163E+00    -0.56809E+00

Eigen vectors and eigen values
------------------------------

V1             V2             V3
x             0.03706        0.78970        0.61237
y            -0.90628       -0.23163        0.35355
z             0.42104       -0.56809        0.70711

r             1.00000        1.00000        1.00000
theta        65.09956      124.61702       45.00004
phi         -87.65820      -16.34700       30.00003

Lambda    0.20369E+05    0.55875E+04   -0.63610E-03

input signal is passed into MVA coordinates

Input signal and MVA coord. syst. signal are in rocot_check_mva.resu file

=======================
end of rocot_check_mva
=======================
```

Table 7.7: rocot_check_mva.out file

7.3.4 test_format.exe

Program **test_format** is just a little program to test how the used compiler write formatted values. Indeed, when your change your compiler, you can be surprised by the fact that the output could change, and it can be very unpleasant in such cases (see section 9.4) Code is given in table 7.8 and results in table 7.9.

```
      program test_format

c     ****************************************************************0**
c *   Test_Format: test how is writen a float, according format g or e
c     P. Robert, Jan 2003
c     ****************************************************************0**
c
      real v(22)
c
      v( 1)= -      0.1234567e-7
      v( 2)= -      0.1234567e-6
      v( 3)= -      0.1234567e-5
      v( 4)= -      0.1234567e-4
      v( 5)= -      0.1234567e-3
      v( 6)= -      0.1234567e-2
      v( 7)= -      0.1234567e-1
      v( 8)= -      0.1234567
      v( 9)= -      7.123456
      v(10)= -     67.12345
      v(11)= -    567.1234
      v(12)= -   4567.123
      v(13)= -  34567.12
      v(14)= - 234567.1
      v(15)= -1234567.
      v(16)= -1234567.e1
      v(17)= -1234567.e2
      v(18)= -1234567.e3
      v(19)= -1234567.e4
      v(20)= -1234567.e5
      v(21)= -1234567.e6
      v(22)= -1234567.e7
c
      do i=1,22
        v(i)=v(i)*13.
        v(i)=v(i)/13.
        print 100, i,v(i), v(i)
      enddo
c
  100 format(i3,5x,g14.7,5x,e14.7)
      stop 'test_format.exe: normal termination'
      end
```

Table 7.8: test_format.f source code

```
  1      -.1234567E-07      -.1234567E-07      -1.234567E-08
  2      -.1234567E-06      -.1234567E-06      -1.234567E-07
  3      -.1234567E-05      -.1234567E-05      -1.234567E-06
  4      -.1234567E-04      -.1234567E-04      -1.234567E-05
  5      -.1234567E-03      -.1234567E-03      -1.234567E-04
  6      -.1234567E-02      -.1234567E-02      -1.234567E-03
  7      -.1234567E-01      -.1234567E-01      -1.234567E-02
  8      -.1234567          -.1234567E+00      -1.234567E-01
  9      -7.123456          -.7123456E+01      -7.123456E+00
 10      -67.12345          -.6712345E+02      -6.712345E+01
 11      -567.1234          -.5671234E+03      -5.671234E+02
 12      -4567.123          -.4567123E+04      -4.567123E+03
 13      -34567.12          -.3456712E+05      -3.456712E+04
 14      -234567.1          -.2345671E+06      -2.345671E+05
 15      -1234567.          -.1234567E+07      -1.234567E+06
 16      -.1234567E+08      -.1234567E+08      -1.234567E+07
 17      -.1234567E+09      -.1234567E+09      -1.234567E+08
 18      -.1234567E+10      -.1234567E+10      -1.234567E+09
 19      -.1234567E+11      -.1234567E+11      -1.234567E+10
 20      -.1234567E+12      -.1234567E+12      -1.234567E+11
 21      -.1234567E+13      -.1234567E+13      -1.234567E+12
 22      -.1234567E+14      -.1234567E+14      -1.234567E+13
```

Table 7.9: test_format.out file

7.3.5 test_week.exe

The **test_week program** check calendar computations done by the library. It is also an example of program to use calendar subroutines. Source file is given in table 7.10 while results are in table 7.11. We apologize for the French output language.

```
      program test_week

c     *****************************************************************0**
c *   Test_Week: test procedures concerning date and week in the year
c     P. Robert, Dec 2006
c     *****************************************************************0**

      character*9 cday,cday1,cday2

      do iy=1998,2018

      iyear=iy

      print 200, '------------'
      print 200, 'annee ',iyear
      print 200, '------------'

      call cv_date_to_dotw(iyear,1,1,idow1)
      call cp_fr_day_name(idow1,cday,nbcha)
      call cv_date_to_weekn(iyear,1,1,iweek1)
      call cv_date_to_weekn(iyear,2,1,iweek2)
c
      print 100, 'Le premier jour de l''annee est un ',cday(1:nbcha)
      print 200, 'Le premier janvier est la semaine ',iweek1
      print 200, 'Le premier fevrier est la semaine ',iweek2

  100 FORMAT(2a)
  200 format(a,i4,a,i5,i3,i3)
  300 format(a,i5,i3,i3,2a)
  400 format(a,i5,i3,i3,a,i3)

c
      iyear1=iyear
      iyear2=iyear

      call cv_weekn_to_date(iweek1,iyear1,imon1,iday1)
      call cv_weekn_to_date(iweek2,iyear2,imon2,iday2)

      print 200,'la semaine ',iweek1,' correspondant a',
     &           iyear1,imon1,iday1
      print 200,'la semaine ',iweek2,' correspondant a',
     &           iyear2,imon2,iday2

      call cv_date_to_doty(iyear,1,1,idoty1)
      call cv_date_to_doty(iyear,2,1,idoty2)

      print 200, 'Le premier janvier est le jour num. ',idoty1
      print 200, 'Le premier fevrier est le jour num. ',idoty2

      call cv_doty_to_date(idoty1,iyear,imon1,iday1)
      call cv_doty_to_date(idoty2,iyear,imon2,iday2)

      print 200, 'et le jour ',idoty1,' est le ',iyear,imon1,iday1
      print 200, 'et le jour ',idoty2,' est le ',iyear,imon2,iday2

      call cv_date_to_dotw(iyear,imon1,iday1,idow1)
      call cv_date_to_dotw(iyear,imon2,iday2,idow2)
      call cp_fr_day_name(idow1,cday1,nbcha)
      call cp_fr_day_name(idow2,cday2,nbcha)

      print 300, 'et ',iyear,imon1,iday1,' est un ',cday1
      print 300, 'et ',iyear,imon2,iday2,' est un ',cday2

      call cv_date_to_weekn(iyear,imon1,iday1,iweek1)
      call cv_date_to_weekn(iyear,imon2,iday2,iweek2)

      print 400, 'et le ',iyear,imon1,iday1,' donne la semaine ',iweek1
      print 400, 'et le ',iyear,imon2,iday2,' donne la semaine ',iweek2

      enddo

      stop 'test_week.exe: normal termination'
      end
```

Table 7.10: test_week.f source code

```
------------
annee 1999
------------
Le premier jour de l'annee est un Vendredi
Le premier janvier est la semaine    0
Le premier fevrier est la semaine    5
la semaine    0 correspondant a 1998 12 28
la semaine    5 correspondant a 1999  2  1
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  1999  1  1
et le jour   32 est le  1999  2  1
et  1999  1  1 est un Vendredi
et  1999  2  1 est un Lundi
et le  1999  1  1 donne la semaine   0
et le  1999  2  1 donne la semaine   5
------------
annee 2000
------------
Le premier jour de l'annee est un Samedi
Le premier janvier est la semaine    0
Le premier fevrier est la semaine    5
la semaine    0 correspondant a 1999 12 27
la semaine    5 correspondant a 2000  1 31
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2000  1  1
et le jour   32 est le  2000  2  1
et  2000  1  1 est un Samedi
et  2000  2  1 est un Mardi
et le  2000  1  1 donne la semaine   0
et le  2000  2  1 donne la semaine   5
------------
annee 2001
------------
Le premier jour de l'annee est un Lundi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2001  1  1
la semaine    5 correspondant a 2001  1 29
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2001  1  1
et le jour   32 est le  2001  2  1
et  2001  1  1 est un Lundi
et  2001  2  1 est un Jeudi
et le  2001  1  1 donne la semaine   1
et le  2001  2  1 donne la semaine   5
------------
annee 2002
------------
Le premier jour de l'annee est un Mardi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2001 12 31
la semaine    5 correspondant a 2002  1 28
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2002  1  1
et le jour   32 est le  2002  2  1
et  2002  1  1 est un Mardi
et  2002  2  1 est un Vendredi
et le  2002  1  1 donne la semaine   1
et le  2002  2  1 donne la semaine   5
------------
annee 2003
------------
Le premier jour de l'annee est un Mercredi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2002 12 30
la semaine    5 correspondant a 2003  1 27
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2003  1  1
et le jour   32 est le  2003  2  1
et  2003  1  1 est un Mercredi
et  2003  2  1 est un Samedi
et le  2003  1  1 donne la semaine   1
et le  2003  2  1 donne la semaine   5
------------
annee 2004
------------
Le premier jour de l'annee est un Jeudi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2003 12 29
la semaine    5 correspondant a 2004  1 26
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2004  1  1
et le jour   32 est le  2004  2  1
et  2004  1  1 est un Jeudi
et  2004  2  1 est un Dimanche
et le  2004  1  1 donne la semaine   1
et le  2004  2  1 donne la semaine   5
------------
annee 2013
------------
Le premier jour de l'annee est un Mardi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2012 12 31
la semaine    5 correspondant a 2013  1 28
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2013  1  1
et le jour   32 est le  2013  2  1
et  2013  1  1 est un Mardi
et  2013  2  1 est un Vendredi
et le  2013  1  1 donne la semaine   1
et le  2013  2  1 donne la semaine   5
------------
annee 2014
------------
Le premier jour de l'annee est un Mercredi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2013 12 30
la semaine    5 correspondant a 2014  1 27
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2014  1  1
et le jour   32 est le  2014  2  1
et  2014  1  1 est un Mercredi
et  2014  2  1 est un Samedi
et le  2014  1  1 donne la semaine   1
et le  2014  2  1 donne la semaine   5
------------
annee 2015
------------
Le premier jour de l'annee est un Jeudi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2014 12 29
la semaine    5 correspondant a 2015  1 26
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2015  1  1
et le jour   32 est le  2015  2  1
et  2015  1  1 est un Jeudi
et  2015  2  1 est un Dimanche
et le  2015  1  1 donne la semaine   1
et le  2015  2  1 donne la semaine   5
------------
annee 2016
------------
Le premier jour de l'annee est un Vendredi
Le premier janvier est la semaine    0
Le premier fevrier est la semaine    5
la semaine    0 correspondant a 2015 12 28
la semaine    5 correspondant a 2016  2  1
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2016  1  1
et le jour   32 est le  2016  2  1
et  2016  1  1 est un Vendredi
et  2016  2  1 est un Lundi
et le  2016  1  1 donne la semaine   0
et le  2016  2  1 donne la semaine   5
------------
annee 2017
------------
Le premier jour de l'annee est un Dimanche
Le premier janvier est la semaine    0
Le premier fevrier est la semaine    5
la semaine    0 correspondant a 2016 12 26
la semaine    5 correspondant a 2017  1 30
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2017  1  1
et le jour   32 est le  2017  2  1
et  2017  1  1 est un Dimanche
et  2017  2  1 est un Mercredi
et le  2017  1  1 donne la semaine   0
et le  2017  2  1 donne la semaine   5
------------
annee 2018
------------
Le premier jour de l'annee est un Lundi
Le premier janvier est la semaine    1
Le premier fevrier est la semaine    5
la semaine    1 correspondant a 2018  1  1
la semaine    5 correspondant a 2018  1 29
Le premier janvier est le jour num.    1
Le premier fevrier est le jour num.   32
et le jour    1 est le  2018  1  1
et le jour   32 est le  2018  2  1
et  2018  1  1 est un Lundi
et  2018  2  1 est un Jeudi
et le  2018  1  1 donne la semaine   1
et le  2018  2  1 donne la semaine   5
```

Table 7.11: test_week.out file

7.3.6 test_day_in_week.exe

Program **test_day_in_week** is like a calendar. It produce a list of day in the month, with the name of each day (i.e. Monday, Tuesday etc.). Useful to produce a past or a future calendar, and to check the library. Source code is given in table 7.12 and example of output in table 7.13.

```
      program test_day_in_week
c
c     ****************************************************************0**
c *   Test_Day_in_Week: compute day in a week, to check with calendars.
c     P. Robert, Dec 2006
c     ****************************************************************0**
c
      character*9 cmonth,cday
c
      print*
      print 100, '************'
      print*
      print 100, 'doweek: compute day in the week for given year'
      print*
c
      do 10 iyear=1999,2018
c
      do 10 imonth=1,12
c
      call cp_en_month_name(imonth,cmonth,nbcha)
      call cp_nbday_in_month(iyear,imonth,nbday)
c
      print 100, '--------------'
      print 200, cmonth,iyear
      print 100, '--------------'

 200 format(a,1x,i4)
c
      do 10 iday=1,nbday
c
      call cv_date_to_dotw(iyear,imonth,iday,idow)
      call cp_en_day_name(idow,cday,nbcha)
c
      print 100, cday,iday
      if(cday(1:6).eq.'Sunday') print 100, ' '
c
   10 continue
c
  100 format(a,i3)
c
      print*
      print 100, '************'
      print*

      stop 'test_day_in_week.exe: normal termination'
      end
c
c     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX0XX
```

Table 7.12: test_day_in_week.f source code

January 1999	February 1999	March 1999	April 1999	May 1999	June 1999
Friday1	Monday1	Monday 1	Thurday 1	Saturday 1	Tuesday 1
Saturday 2	Tuesday 2	Tuesday 2	Friday 2	Sunday 2	Wednesday 2
Sunday3	Wednesday 3	Wednesday 3	Saturday 3		Thurday 3
	Thurday 4	Thurday 4	Sunday 4	Monday 3	Friday 4
Monday4	Friday5	Friday 5		Tuesday 4	Saturday 5
Tuesday 5	Saturday 6	Saturday 6	Monday 5	Wednesday 5	Sunday 6
Wednesday 6	Sunday 7	Sunday 7	Tuesday 6	Thurday 6	
Thurday 7			Wednesday 7	Friday 7	Monday 7
Friday8	Monday 8	Monday 8	Thurday 8	Saturday 8	Tuesday 8
Saturday 9	Tuesday 9	Tuesday 9	Friday 9	Sunday 9	Wednesday 9
Sunday 10	Wednesday 10	Wednesday 10	Saturday 10		Thurday 10
	Thurday 11	Thurday 11	Sunday 11	Monday 10	Friday 11
Monday 11	Friday 12	Friday 12		Tuesday 11	Saturday 12
Tuesday 12	Saturday 13	Saturday 13	Monday 12	Wednesday 12	Sunday 13
Wednesday 13	Sunday 14	Sunday 14	Tuesday 13	Thurday 13	
Thurday 14			Wednesday 14	Friday 14	Monday 14
Friday 15	Monday 15	Monday 15	Thurday 15	Saturday 15	Tuesday 15
Saturday 16	Tuesday 16	Tuesday 16	Friday 16	Sunday 16	Wednesday 16
Sunday 17	Wednesday 17	Wednesday 17	Saturday 17		Thurday 17
	Thurday 18	Thurday 18	Sunday 18	Monday 17	Friday 18
Monday 18	Friday 19	Friday 19		Tuesday 18	Saturday 19
Tuesday 19	Saturday 20	Saturday 20	Monday 19	Wednesday 19	Sunday 20
Wednesday 20	Sunday 21	Sunday 21	Tuesday 20	Thurday 20	
Thurday 21			Wednesday 21	Friday 21	Monday 21
Friday 22	Monday 22	Monday 22	Thurday 22	Saturday 22	Tuesday 22
Saturday 23	Tuesday 23	Tuesday 23	Friday 23	Sunday 23	Wednesday 23
Sunday 24	Wednesday 24	Wednesday 24	Saturday 24		Thurday 24
	Thurday 25	Thurday 25	Sunday 25	Monday 24	Friday 25
Monday 25	Friday 26	Friday 26		Tuesday 25	Saturday 26
Tuesday 26	Saturday 27	Saturday 27	Monday 26	Wednesday 26	Sunday 27
Wednesday 27	Sunday 28	Sunday 28	Tuesday 27	Thurday 27	
Thurday 28			Wednesday 28	Friday 28	Monday 28
Friday 29		Monday 29	Thurday 29	Saturday 29	Tuesday 29
Saturday 30		Tuesday 30	Friday 30	Sunday 30	Wednesday 30
Sunday 31		Wednesday 31			
				Monday 31	

July 1999	August 1999	September 1999	October 1999	November 1999	December 1999
Thurday 1	Sunday 1	Wednesday 1	Friday 1	Monday 1	Wednesday 1
Friday 2		Thurday 2	Saturday 2	Tuesday 2	Thurday 2
Saturday 3	Monday 2	Friday 3	Sunday 3	Wednesday 3	Friday 3
Sunday 4	Tuesday 3	Saturday 4		Thurday 4	Saturday 4
	Wednesday 4	Sunday 5	Monday 4	Friday 5	Sunday 5
Monday 5	Thurday 5		Tuesday 5	Saturday 6	
Tuesday 6	Friday 6	Monday 6	Wednesday 6	Sunday 7	Monday 6
Wednesday 7	Saturday 7	Tuesday 7	Thurday 7		Tuesday 7
Thurday 8	Sunday 8	Wednesday 8	Friday 8	Monday 8	Wednesday 8
Friday 9		Thurday 9	Saturday 9	Tuesday 9	Thurday 9
Saturday 10	Monday 9	Friday 10	Sunday 10	Wednesday 10	Friday 10
Sunday 11	Tuesday 10	Saturday 11		Thurday 11	Saturday 11
	Wednesday 11	Sunday 12	Monday 11	Friday 12	Sunday 12
Monday 12	Thurday 12		Tuesday 12	Saturday 13	
Tuesday 13	Friday 13	Monday 13	Wednesday 13	Sunday 14	Monday 13
Wednesday 14	Saturday 14	Tuesday 14	Thurday 14		Tuesday 14
Thurday 15	Sunday 15	Wednesday 15	Friday 15	Monday 15	Wednesday 15
Friday 16		Thurday 16	Saturday 16	Tuesday 16	Thurday 16
Saturday 17	Monday 16	Friday 17	Sunday 17	Wednesday 17	Friday 17
Sunday 18	Tuesday 17	Saturday 18		Thurday 18	Saturday 18
	Wednesday 18	Sunday 19	Monday 18	Friday 19	Sunday 19
Monday 19	Thurday 19		Tuesday 19	Saturday 20	
Tuesday 20	Friday 20	Monday 20	Wednesday 20	Sunday 21	Monday 20
Wednesday 21	Saturday 21	Tuesday 21	Thurday 21		Tuesday 21
Thurday 22	Sunday 22	Wednesday 22	Friday 22	Monday 22	Wednesday 22
Friday 23		Thurday 23	Saturday 23	Tuesday 23	Thurday 23
Saturday 24	Monday 23	Friday 24	Sunday 24	Wednesday 24	Friday 24
Sunday 25	Tuesday 24	Saturday 25		Thurday 25	Saturday 25
	Wednesday 25	Sunday 26	Monday 25	Friday 26	Sunday 26
Monday 26	Thurday 26		Tuesday 26	Saturday 27	
Tuesday 27	Friday 27	Monday 27	Wednesday 27	Sunday 28	Monday 27
Wednesday 28	Saturday 28	Tuesday 28	Thurday 28		Tuesday 28
Thurday 29	Sunday 29	Wednesday 29	Friday 29	Monday 29	Wednesday 29
Friday 30		Thurday 30	Saturday 30	Tuesday 30	Thurday 30
Saturday 31	Monday 30		Sunday 31		Friday 31
	Tuesday 31				

Table 7.13: test_day_in_week.out file

7.3.7 test_dir_sun.exe

Program **test_dir_sun** allows checking fundamental component of the library: direction of the Sun in various coordinate system. Code is given in table 7.14 and results in table 7.15.

Note that the Sun direction in GSE or GSM frame is only along the X axes (by definition, so it is another test).

```
      program test_dir_sun
c
c     ****************************************************************0**
c *   Test_Dir_Sun: Sun direction in various coordinate systems
c     P. Robert, CETP, June 2001 ; use Rocotlib library
c     ****************************************************************0**
c
      print*
      print 200, '*****************************************************'
      print*
      print 200, 'Sun direction in various coordinate systems'
      print*
c
c *   reading date and time & compute general matrix
c
      call r_date(iyear,imonth,iday)
      call r_time(ih,im,is)
      call cp_time_param(iyear,imonth,iday,ih,im,is)
c
c *   reading position of point of observation in GEO
c
      print 200, 'latitude, longitude, distance in GEO ? (deg & any u.)'
      read *, rlageo, rlogeo,rdis
      print 200, ' ',rlageo, rlogeo,rdis
c
      rlagr=rlageo*3.1415927/180.
      rlogr=rlogeo*3.1415927/180.
      teta=(90.-rlageo)*3.1415927/180.
c
c *   geographic cartesian coordinates
c
      call t_sph_to_car(rdis,teta,rlogr,xgeo,ygeo,zgeo)
c
c *   same in other systems
c
      call t_geo_to_gei(xgeo,ygeo,zgeo,xgei,ygei,zgei)
      call t_geo_to_gse(xgeo,ygeo,zgeo,xgse,ygse,zgse)
      call t_geo_to_gsm(xgeo,ygeo,zgeo,xgsm,ygsm,zgsm)
      call t_geo_to_vdh(xgeo,ygeo,zgeo,rlagr,rlogr,xvdh,yvdh,zvdh)
c
c *   latitude, longitude in other systems
c
      call t_car_to_sph(xgeo,ygeo,zgeo,r,tetgeo,phigeo)
      call t_car_to_sph(xgei,ygei,zgei,r,tetgei,phigei)
      call t_car_to_sph(xgse,ygse,zgse,r,tetgse,phigse)
      call t_car_to_sph(xgsm,ygsm,zgsm,r,tetgsm,phigsm)
      call t_car_to_sph(xvdh,yvdh,zvdh,r,tetvdh,phivdh)
c
      call colalon(tetgeo,phigeo,rlageo,rlogeo)
      call colalon(tetgei,phigei,rlagei,rlogei)
      call colalon(tetgse,phigse,rlagse,rlogse)
      call colalon(tetgsm,phigsm,rlagsm,rlogsm)
      call colalon(tetvdh,phivdh,rlavdh,rlovdh)
c
      print*,'_____________________________________________________________'
      print*
      print 300, 'date: YY/MM/JJ=', iyear,imonth,iday
      print 300, 'time: HH/MM/SS=', ih,im,is
      print*
c
      print 100, '            ','x','y','z','R','Lat','Long'
      print 100
c
      print 200, 'Pos. in GEO :',xgeo,ygeo,zgeo,  r,rlageo,rlogeo
      print 200, 'Pos. in GEI :',xgei,ygei,zgei,  r,rlagei,rlogei
      print 200, 'Pos. in GSE :',xgse,ygse,zgse,  r,rlagse,rlogse
      print 200, 'Pos. in GSM :',xgsm,ygsm,zgsm,  r,rlagsm,rlogsm
      print 200, 'Pos. in VDH :',xvdh,yvdh,zvdh,  r,rlavdh,rlovdh
c
c *   Sun direction
c
      call g_gei_geo_sun_dir(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)
      call t_gei_to_gse(sxgei,sygei,szgei,sxgse,sygse,szgse)
      call t_gei_to_gsm(sxgei,sygei,szgei,sxgsm,sygsm,szgsm)
      call t_geo_to_vdh(sxgeo,sygeo,szgeo,rlagr,rlogr,sxvdh,syvdh,szvdh)
c
      call t_car_to_sph(sxgeo,sygeo,szgeo,rgeo,tetgeo,phigeo)
      call t_car_to_sph(sxgei,sygei,szgei,rgei,tetgei,phigei)
      call t_car_to_sph(sxgse,sygse,szgse,rgse,tetgse,phigse)
      call t_car_to_sph(sxgsm,sygsm,szgsm,rgsm,tetgsm,phigsm)
      call t_car_to_sph(sxvdh,syvdh,szvdh,rvdh,tetvdh,phivdh)
c
      call convdeg(tetgeo,phigeo)
      call convdeg(tetgei,phigei)
      call convdeg(tetgse,phigse)
      call convdeg(tetgsm,phigsm)
      call convdeg(tetvdh,phivdh)
c
```

```
c
      print*
      print 200, 'Sun  in GEO :',sxgeo,sygeo,szgeo,  rgeo,tetgeo,phigeo
      print 200, 'Sun  in GEI :',sxgei,sygei,szgei,  rgei,tetgei,phigei
      print 200, 'Sun  in GSE :',sxgse,sygse,szgse,  rgse,tetgse,phigse
      print 200, 'Sun  in GSM :',sxgsm,sygsm,szgsm,  rgsm,tetgsm,phigsm
      print 200, 'Sun  in VDH :',sxvdh,syvdh,szvdh,  rvdh,tetvdh,phivdh
      print*
      print*
      print*, '*******************************************************'
      print*
c
 100 format(a,4a10  ,2a10  )
 200 format(a,4f10.3,2f10.3)
 300 format(a,3I5)

     stop 'test_dir_sun.exe: normal termination'
     end


     subroutine convdeg(x,y)
c
     pisd=3.1415927/180.
     x=x/pisd
     y=y/pisd
     return
     end

     subroutine colalon(teta,phi,rlat,rlon)
c
     pisd=3.1415927/180.
     rlat= 90. -teta/pisd
     rlon= phi/pisd
     return
     end
```

Table 7.14: test_dir_sun.f source code

```
*******************************************************

Sun direction in various coordinate systems

iyear,imonth,iday ? (ex: 1990,10,17)
2001  6 13
hour, minute, second ? (ex: 10,45,50)
3 40  0
latitude, longitude, distance in GEO ? (deg & any u.)
-65.380   157.940  6500.000
_______________________________________________________

date: YY/MM/JJ= 2001    6   13
time: HH/MM/SS=    3   40    0

                    x         y         z         R       Lat      Long

Pos. in GEO : -2509.646  1017.021 -5909.090  6500.000   -65.380   157.940
Pos. in GEI : -1121.443  2464.756 -5909.090  6500.000   -65.380   114.465
Pos. in GSE :  -240.266  1099.021 -6401.908  6500.000   -80.033   102.332
Pos. in GSM :  -240.266  1804.240 -6239.951  6500.000   -73.738    97.585
Pos. in VDH :  6500.000     0.000     0.000  6500.000     0.000     0.000

Sun  in GEO :    -0.527     0.753     0.394     1.000    66.790   124.995
Sun  in GEI :     0.136     0.909     0.394     1.000    66.790    81.520
Sun  in GSE :     1.000     0.000     0.000     1.000    90.000     0.000
Sun  in GSM :     1.000     0.000     0.000     1.000    90.000     0.000
Sun  in VDH :    -0.037    -0.500     0.865     1.000    30.078   -94.230

*******************************************************
```

Table 7.15: test_dir_sun.out file

7.4 Usefull programs

These programs can be used to do useful computation without writing any new codes. It can be also used as examples to write similar other programs.

If the user do not want to modify the makefile, we remember that you can directly compile and make executable file by the simple commands hereafter (for instance for fort77 compiler):

`fort77 -c toto.f` create object file toto.o

`fort77 toto.o rocotlib_V3p0.o -o toto.exe` create executable file toto.exe

Program can be run interactively by lauching *.exe, or in batch mode with *.sh (as explained in section 7.1.).

7.4.1 rocot_info.exe

This program give information and historic of the library and the version number. Output is given in table 7.16.

7.4.2 rocot_utility.exe

This program allow to make geocentric coordinate transformation without write any program. It take in entry a file as the one given in table 7.17. The program ask a new desired coordinate, do the computation and put the results in an output new file with same format, in the given new coordinates.

The data must be formatted as following:

- date/time of the vector, given as ISO format
- values of the 3 components of the vector, in any format.

All the lines beginning by the # character is a comment. Nevertheless the line as:

```
# CHA : COORDINATE_SYSTEM : GSE
```

is mandatory, and must contain the name of the current coordinate system. Line format must be the same as given example.

Allowed coordinate systems are: `geo,gei,gsq,gse,gsm,sma,mag`.

Allowed transformations are:

- `geo→gei, geo→gsq, geo→gse, geo→gsm, geo→sma geo→mag`
- `gei→geo, gsq→geo, gse→geo, gsm→geo, sma→geo mag→geo`
- `gei→gsq, gei→gse, gei→gsm, gei→sm , gei→mag`
- `gsq→gei, gse→gei, gsm→gei, sm →gei, mag→gei`
- `gsq→gse, gsq→gsm, gsq→sm , gsq→mag`
- `gse→gsq, gsm→gsq, sm →gsq, mag→gsq`
- `gse→gsm, gse→sm , gse→mag`
- `gsm→gse, sm →gse, mag→gse`
- `gsm→sm , gsm→mag`
- `sm →gsm, mag→gsm`
- `sm →mag`
- `mag→sm`

```
Coordinates Transformation Library  ROCOTLIB

Revised Version  3.2 - Feb 2020

____________________

initially supported by

EUROPEAN  SPACE  AGENCY

Study of the Cluster Mission
Planning Related Aspects
within the Numerical Simulations Network

Patrick ROBERT, CRPE, November 1992
____________________

version 1.0, November 1992
version 1.1, July     1993
version 1.2, January  1995
version 1.3, July     2000 (Jul. day 2000/sun dir)
version 1.4, June     2001 (for automatic docum.)
version 1.5, December 2001 (add cp_sunrise_sunset
version 1.6, Juin     2002 (upgrade IGRF -> 2005)
version 1.7, December 2002 (Version for CDPP)
version 1.8, November 2003 (add t_sr2_to_sr)
version 1.9, March    2004 (compatibility with IDL)
version 2.0, November 2006 (Update  IGRF -> 2010)
version 2.1, November 2006 (Update  IGRF -> 2015)
version 2.2, December 2011 (cp_sunrise_sunset polar)
version 3.0, May      2017 (IGRF->table geomag. pole
                            -> 2020 +some other coord. sys.)
version 3.1, January  2019 (add name compatibility with V2.2)
version 3.2, February 2020 (add trans. with Euler angles)

Copyright 1992, Patrick ROBERT, CNRS-ESA,
All Rights reserved
____________________

For details, see the orginal document untitled:

CLUSTER Software Tools
Part I: Coordinate Transformation Library
Document de travail DT/CRPE/1231
Patrick Robert, CRPE/TID, Juillet 1993

Available at CDPP:

ROCOTLIB: a coordinate Transformation Library
for Solar-Terrestrial studies
Patrick ROBERT, version 1.7 - January 2003,
Rapport Interne no RI-CETP/02/2003
____________________

The previous version 3.0 contains new transformations
(TPN, MVA), and the cp_sunrise_sunset sub. allowing
computation of sunrise and sunset anywhere on Earth,
including polar zones.

The determination of the dipole axis direction
has been replaced by a table available from years
1900 to 2020, in order to have an easy way to code
maintenance in the coming years.

Operations on matrix (somme, product, inversion,
computation of eigen vectors, diagonalization...)
required for Minimum Variance Analysis coordinates
have been added with prefix "mat_".

The previous and temporary V3.1 is the same as 3.0
but is compatible with V2.2 version concerning
subroutine name (ex: tmaggsm -> t_mag_to_gsm)

The present V3.2 version contains transformations
with Euler angles and their interpolation.

Original code developped in Fortran 77.
Fortran 90, C, and IDL codes are also available.
```

Table 7.16: rocot_info.out file

```
# ------------------------------------------------------------
# CHA : MAIN_TITLE        : CLUSTER / FGM / Rumba (#1)
# CHA : NATURE            : B DC magnetic field
# CHA : UNIT              : nT
# CHA : COORDINATE_SYSTEM : GSE
# ------------------------------------------------------------
2001-01-26T11:30:30.025Z   0.156690E+02  -0.188044E+02   0.110600E+00
2001-01-26T11:30:30.070Z   0.155405E+02  -0.189873E+02   0.812000E-01
2001-01-26T11:30:30.114Z   0.154918E+02  -0.189895E+02   0.507000E-01
2001-01-26T11:30:30.159Z   0.153755E+02  -0.190919E+02  -0.244000E-01
2001-01-26T11:30:30.204Z   0.153097E+02  -0.193107E+02  -0.643000E-01
2001-01-26T11:30:30.248Z   0.151778E+02  -0.195427E+02  -0.298000E-01
2001-01-26T11:30:30.293Z   0.151333E+02  -0.196737E+02  -0.501000E-01
2001-01-26T11:30:30.337Z   0.151979E+02  -0.198302E+02  -0.873000E-01
2001-01-26T11:30:30.382Z   0.151976E+02  -0.199562E+02  -0.771000E-01
2001-01-26T11:30:30.427Z   0.152318E+02  -0.200417E+02  -0.628000E-01
2001-01-26T11:30:30.471Z   0.152323E+02  -0.201160E+02  -0.774000E-01

...

2001-01-26T11:30:59.640Z   0.334878E+02  -0.229664E+02   0.585970E+01
2001-01-26T11:30:59.684Z   0.331938E+02  -0.231660E+02   0.592940E+01
2001-01-26T11:30:59.729Z   0.326701E+02  -0.237049E+02   0.623020E+01
2001-01-26T11:30:59.774Z   0.323481E+02  -0.241811E+02   0.640490E+01
2001-01-26T11:30:59.818Z   0.319619E+02  -0.245427E+02   0.664440E+01
```

Table 7.17: Content of input file CLUFGM1-GSE.txt

```
*****************************************************

rocot_utility:

input  file ? (ex: CLUFGM1-gse.txt)
in/CLUFGM1-gse.txt
input  coordinate system :gse

output coordinate system ? (ex: gsm)
gsm
output file ? (ex: CLUFGM1-gsm.txt)
out/CLUFGM1-gsm.txt

*****************************************************
```

Table 7.18: rocot_utility.out file

```
# ------------------------------------------------------------
# CHA : MAIN_TITLE        : CLUSTER / FGM / Rumba (#1)
# CHA : NATURE            : B DC magnetic field
# CHA : UNIT              : nT
# CHA : COORDINATE_SYSTEM : gsm
# ------------------------------------------------------------
2001-01-26T11:30:30.025Z   1.566900E+01  -1.877091E+01  -1.127175E+00
2001-01-26T11:30:30.070Z   1.554050E+01  -1.895148E+01  -1.168548E+00
2001-01-26T11:30:30.114Z   1.549180E+01  -1.895167E+01  -1.199127E+00
2001-01-26T11:30:30.159Z   1.537550E+01  -1.904890E+01  -1.280803E+00
2001-01-26T11:30:30.204Z   1.530970E+01  -1.926460E+01  -1.335016E+00
2001-01-26T11:30:30.248Z   1.517780E+01  -1.949837E+01  -1.315859E+00
2001-01-26T11:30:30.293Z   1.513330E+01  -1.962775E+01  -1.344736E+00
2001-01-26T11:30:30.337Z   1.519790E+01  -1.978147E+01  -1.392155E+00
2001-01-26T11:30:30.382Z   1.519760E+01  -1.990786E+01  -1.390269E+00
2001-01-26T11:30:30.427Z   1.523180E+01  -1.999412E+01  -1.381627E+00
2001-01-26T11:30:30.471Z   1.523230E+01  -2.006730E+01  -1.401085E+00
...
2001-01-26T11:30:59.640Z   3.348780E+01  -2.330259E+01   4.333719E+00
2001-01-26T11:30:59.684Z   3.319380E+01  -2.350635E+01   4.390115E+00
2001-01-26T11:30:59.729Z   3.267010E+01  -2.406389E+01   4.654754E+00
2001-01-26T11:30:59.774Z   3.234810E+01  -2.455057E+01   4.797698E+00
2001-01-26T11:30:59.818Z   3.196190E+01  -2.492716E+01   5.012852E+00
```

Table 7.19: Content of output file CLUFGM1-GSM.txt

7.4.3 co_julday_table.exe

Produce a correspondance table between Julian day and date. Example is given in table 7.20

```
JULIAN DAY 1950  - DATE CORRESPONDENCE

0 1950-01-01       1 1950-01-02       2 1950-01-03       3 1950-01-04
4 1950-01-05       5 1950-01-06       6 1950-01-07       7 1950-01-08
8 1950-01-09       9 1950-01-10       10 1950-01-11      11 1950-01-12
12 1950-01-13      13 1950-01-14      14 1950-01-15      15 1950-01-16
16 1950-01-17      17 1950-01-18      18 1950-01-19      19 1950-01-20
20 1950-01-21      21 1950-01-22      22 1950-01-23      23 1950-01-24
24 1950-01-25      25 1950-01-26      26 1950-01-27      27 1950-01-28
28 1950-01-29      29 1950-01-30      30 1950-01-31      31 1950-02-01
32 1950-02-02      33 1950-02-03      34 1950-02-04      35 1950-02-05
36 1950-02-06      37 1950-02-07      38 1950-02-08      39 1950-02-09
40 1950-02-10      41 1950-02-11      42 1950-02-12      43 1950-02-13
44 1950-02-14      45 1950-02-15      46 1950-02-16      47 1950-02-17
48 1950-02-18      49 1950-02-19      50 1950-02-20      51 1950-02-21
52 1950-02-22      53 1950-02-23      54 1950-02-24      55 1950-02-25
56 1950-02-26      57 1950-02-27      58 1950-02-28      59 1950-03-01
60 1950-03-02      61 1950-03-03      62 1950-03-04      63 1950-03-05
64 1950-03-06      65 1950-03-07      66 1950-03-08      67 1950-03-09
68 1950-03-10      69 1950-03-11      70 1950-03-12      71 1950-03-13
72 1950-03-14      73 1950-03-15      74 1950-03-16      75 1950-03-17
76 1950-03-18      77 1950-03-19      78 1950-03-20      79 1950-03-21
80 1950-03-22      81 1950-03-23      82 1950-03-24      83 1950-03-25
84 1950-03-26      85 1950-03-27      86 1950-03-28      87 1950-03-29
88 1950-03-30      89 1950-03-31      90 1950-04-01      91 1950-04-02
92 1950-04-03      93 1950-04-04      94 1950-04-05      95 1950-04-06
96 1950-04-07      97 1950-04-08      98 1950-04-09      99 1950-04-10

... etc...

18612 2000-12-16   18613 2000-12-17   18614 2000-12-18   18615 2000-12-19
18616 2000-12-20   18617 2000-12-21   18618 2000-12-22   18619 2000-12-23
18620 2000-12-24   18621 2000-12-25   18622 2000-12-26   18623 2000-12-27
18624 2000-12-28   18625 2000-12-29   18626 2000-12-30   18627 2000-12-31


JULIAN DAY 2000  - DATE CORRESPONDENCE

 0 2000-01-01       1 2000-01-02       2 2000-01-03       3 2000-01-04
 4 2000-01-05       5 2000-01-06       6 2000-01-07       7 2000-01-08
 8 2000-01-09       9 2000-01-10       10 2000-01-11      11 2000-01-12
12 2000-01-13      13 2000-01-14      14 2000-01-15      15 2000-01-16
16 2000-01-17      17 2000-01-18      18 2000-01-19      19 2000-01-20
20 2000-01-21      21 2000-01-22      22 2000-01-23      23 2000-01-24
24 2000-01-25      25 2000-01-26      26 2000-01-27      27 2000-01-28
28 2000-01-29      29 2000-01-30      30 2000-01-31      31 2000-02-01
32 2000-02-02      33 2000-02-03      34 2000-02-04      35 2000-02-05

... etc...

9984 2027-05-03    9985 2027-05-04    9986 2027-05-05    9987 2027-05-06
9988 2027-05-07    9989 2027-05-08    9990 2027-05-09    9991 2027-05-10
9992 2027-05-11    9993 2027-05-12    9994 2027-05-13    9995 2027-05-14
9996 2027-05-15    9997 2027-05-16    9998 2027-05-17    9999 2027-05-18
10000 2027-05-19   10001 2027-05-20   10002 2027-05-21   10003 2027-05-22
```

Table 7.20: co_julday_table.out file

7.4.4 co_mag_time.exe

According input parameters (date, time, geographic position) the program compute magnetic local time. Example is given in table 7.21.

```
**************************************************

Test of cmagtim:

iyear,imonth,iday ? (ex: 1990,10,17)
2017  1 26
hour, minute, second ? (ex: 10,45,50)
13 15  0
lat et long en degre?
lat, long=    48.85700     2.35000

date: YY/MM/JJ= 2017    1   26
time: HH/MM/SS=   13   15    0

Dipole in GEO:   .048729  -.157943   .986245  1.000000    9.5140  -72.8537
Point  in GEO:   .657387   .026978   .753070  1.000000   41.1430    2.3500
Point  in MAG:   .041242   .636123   .770484  1.000000   39.6026   86.2905
Sun    in MAG:   .558368   .797235  -.229438  1.000000  103.2640   54.9934

Mag. Local Time:   9.91352 soit  09:54:48

**************************************************
```

Table 7.21: co_mag_time.out file

7.4.5 co_sunset.exe

For a given date and location on the Earth, the program compute crossing meridian time, sunrise and sunset time, and the day duration. Example is given in table 7.22.

```
sunset: computation of sunrise and sunset time
assuming spherical Earth

------------------------------------------------------
iyear,imonth,iday ? (ex: 1990,10,17)
2017  1 26
lat et long en degre, et lieu ? (blank separator, 0 or q to quit)
48.857     2.350   Paris

merid. time (UT)= 12:03:20 sunrise= 07:34:00 sunset= 16:33:00 durd=  8:59:00
------------------------------------------------------
iyear,imonth,iday ? (ex: 1990,10,17)
2017  1 26
lat et long en degre, et lieu ? (blank separator, 0 or q to quit)
43.620     1.450   Toulouse

merid. time (UT)= 12:06:50 sunrise= 07:21:50 sunset= 16:52:20 durd=  9:30:30
------------------------------------------------------
iyear,imonth,iday ? (ex: 1990,10,17)
2017  1 26
lat et long en degre, et lieu ? (blank separator, 0 or q to quit)
51.500      .170   Londres

merid. time (UT)= 12:12:00 sunrise= 07:52:20 sunset= 16:32:20 durd=  8:40:00
------------------------------------------------------
iyear,imonth,iday ? (ex: 1990,10,17)
1950  1 26
lat et long en degre, et lieu ? (blank separator, 0 or q to quit)
51.500      .170   Londres

merid. time (UT)= 12:12:00 sunrise= 07:53:30 sunset= 16:31:00 durd=  8:37:30
------------------------------------------------------
```

Table 7.22: co_sunset.out file

7.4.6 co_sunset_year.exe

For a given year and location on the Earth, this program compute a calendar table giving time, elevation and azimut of sunrise, meridian crossing, sunset and day duration for each day of the given months.

Note that elevation for sunrise and sunset is always close to zero (sun at horizon). Elevation at meridian crossing correspond to the maximum elevation of the sun in the current day.

Example is given in table 7.23.

March

date	tris	eleris	aziris	tmer	elemer	azimer	tset	eleset	aziset	durd
2018-03-01	06:33	359d11'	100d37'	12:03	33d40'	180d01'	17:34	359d09'	259d40'	11:0
2018-03-02	06:31	359d11'	100d02'	12:03	34d03'	180d01'	17:36	359d08'	260d16'	11:0
2018-03-03	06:29	359d10'	99d27'	12:03	34d26'	180d02'	17:37	359d09'	260d50'	11:0
2018-03-04	06:27	359d10'	98d51'	12:02	34d49'	180d00'	17:39	359d08'	261d27'	11:1
2018-03-05	06:25	359d11'	98d16'	12:02	35d12'	180d01'	17:40	359d09'	262d01'	11:1
2018-03-06	06:23	359d11'	97d41'	12:02	35d35'	180d02'	17:42	359d08'	262d38'	11:1
2018-03-07	06:21	359d11'	97d06'	12:02	35d58'	180d00'	17:43	359d09'	263d13'	11:2
2018-03-08	06:19	359d10'	96d29'	12:02	36d22'	180d02'	17:45	359d09'	263d48'	11:2
2018-03-09	06:17	359d10'	95d54'	12:01	36d45'	180d00'	17:47	359d08'	264d25'	11:3
2018-03-10	06:15	359d11'	95d18'	12:01	37d09'	180d02'	17:48	359d09'	265d00'	11:3
2018-03-11	06:13	359d10'	94d42'	12:01	37d32'	180d00'	17:50	359d09'	265d36'	11:3
2018-03-12	06:11	359d10'	94d06'	12:01	37d56'	180d02'	17:51	359d08'	266d13'	11:4
2018-03-13	06:09	359d11'	93d31'	12:00	38d19'	180d01'	17:53	359d08'	266d48'	11:4
2018-03-14	06:07	359d10'	92d54'	12:00	38d43'	180d00'	17:54	359d09'	267d24'	11:4
2018-03-15	06:05	359d11'	92d19'	12:00	39d07'	180d02'	17:56	359d09'	268d00'	11:5
2018-03-16	06:02	359d10'	91d43'	11:59	39d30'	180d01'	17:57	359d09'	268d36'	11:5
2018-03-17	06:00	359d10'	91d06'	11:59	39d54'	180d00'	17:59	359d09'	269d11'	11:5
2018-03-18	05:58	359d11'	90d31'	11:59	40d18'	180d03'	18:00	359d08'	269d49'	12:0
2018-03-19	05:56	359d10'	89d54'	11:59	40d41'	180d02'	18:02	359d08'	270d25'	12:0
2018-03-20	05:54	359d11'	89d19'	11:58	41d05'	180d01'	18:03	359d08'	271d01'	12:0
2018-03-21	05:52	359d10'	88d42'	11:58	41d29'	180d00'	18:05	359d09'	271d37'	12:1
2018-03-22	05:50	359d10'	88d06'	11:58	41d52'	180d00'	18:06	359d09'	272d13'	12:1
2018-03-23	05:48	359d11'	87d31'	11:57	42d16'	180d02'	18:08	359d09'	272d49'	12:2
2018-03-24	05:46	359d10'	86d54'	11:57	42d40'	180d02'	18:09	359d09'	273d25'	12:2
2018-03-25	05:44	359d11'	86d20'	11:57	43d03'	180d01'	18:11	359d09'	274d00'	12:2
2018-03-26	05:41	359d10'	85d43'	11:56	43d27'	180d01'	18:12	359d09'	274d36'	12:3
2018-03-27	05:39	359d10'	85d07'	11:56	43d50'	180d00'	18:14	359d09'	275d12'	12:3
2018-03-28	05:37	359d11'	84d32'	11:56	44d14'	180d03'	18:15	359d09'	275d48'	12:3
2018-03-29	05:35	359d10'	83d56'	11:56	44d37'	180d02'	18:17	359d09'	276d24'	12:4
2018-03-30	05:33	359d11'	83d21'	11:55	45d00'	180d01'	18:18	359d09'	276d59'	12:4
2018-03-31	05:31	359d10'	82d45'	11:55	45d24'	180d01'	18:20	359d09'	277d35'	12:4

April

date	tris	eleris	aziris	tmer	elemer	azimer	tset	eleset	aziset	durd
2018-04-01	05:29	359d10'	82d09'	11:55	45d47'	180d00'	18:21	359d09'	278d10'	12:5
2018-04-02	05:27	359d10'	81d34'	11:54	46d10'	180d03'	18:23	359d09'	278d46'	12:5
2018-04-03	05:25	359d10'	80d58'	11:54	46d33'	180d02'	18:24	359d09'	279d21'	13:0
2018-04-04	05:23	359d10'	80d24'	11:54	46d56'	180d01'	18:26	359d09'	279d56'	13:0
2018-04-05	05:21	359d11'	79d49'	11:53	47d19'	180d00'	18:27	359d09'	280d31'	13:0
2018-04-06	05:18	359d10'	79d13'	11:53	47d41'	180d03'	18:29	359d09'	281d06'	13:1
2018-04-07	05:16	359d10'	78d39'	11:53	48d04'	180d01'	18:30	359d09'	281d41'	13:1
2018-04-08	05:14	359d11'	78d05'	11:53	48d26'	180d00'	18:32	359d08'	282d16'	13:1
2018-04-09	05:12	359d10'	77d29'	11:52	48d49'	180d03'	18:33	359d08'	282d50'	13:2
2018-04-10	05:10	359d10'	76d55'	11:52	49d11'	180d01'	18:35	359d08'	283d25'	13:2
2018-04-11	05:08	359d10'	76d21'	11:52	49d33'	180d00'	18:36	359d08'	283d59'	13:2
2018-04-12	05:06	359d10'	75d47'	11:52	49d55'	180d02'	18:38	359d08'	284d33'	13:3
2018-04-13	05:04	359d10'	75d14'	11:51	50d17'	180d00'	18:39	359d08'	285d07'	13:3
2018-04-14	05:02	359d10'	74d40'	11:51	50d38'	180d02'	18:41	359d08'	285d41'	13:3
2018-04-15	05:00	359d10'	74d06'	11:51	51d00'	180d00'	18:42	359d08'	286d14'	13:4
2018-04-16	04:58	359d10'	73d32'	11:51	51d21'	180d02'	18:44	359d08'	286d48'	13:4
2018-04-17	04:56	359d11'	73d01'	11:50	51d42'	180d03'	18:45	359d09'	287d19'	13:4

...

Table 7.23: co_sunset_year.out file (part)

7.4.7 co_seasons.exe

This program compute the starting date of each seasons, i.e. the date and time of solstices and equinoxes. Time accuracy is about one hour. Example is given in table 7.24.

```
***********************************************
co_seasons: computation of Solstices & equinoxes
***********************************************

year_1, year_2 ? ex: 1990 2020
1990 2020


=====================================================================
June solstice  dec. solstice  march equinoxe   sept. equinoxe
Year LY      DD hh:mm       DD hh:mm       DD hh:mm       DD hh:mm
=====================================================================
1990 0       21 15:20       22 02:50       20 21:20       23 07:00
1991 0       21 21:10       22 08:40       21 03:10       23 12:50
1992 1       21 03:00       21 14:30       20 09:00       22 18:40
---------------------------------------------------------------------
1993 0       21 08:40       21 20:20       20 14:50       23 00:30
1994 0       21 14:30       22 02:00       20 20:40       23 06:20
1995 0       21 20:20       22 08:00       21 02:30       23 12:10
1996 1       21 02:10       21 13:50       20 08:20       22 17:50
---------------------------------------------------------------------
1997 0       21 08:00       21 19:40       20 14:00       22 23:40
1998 0       21 13:40       22 01:30       20 19:50       23 05:30
1999 0       21 19:30       22 07:20       21 01:40       23 11:20
2000 1       21 01:20       21 13:10       20 07:30       22 17:10
---------------------------------------------------------------------
2001 0       21 07:10       21 19:00       20 13:20       22 23:00
2002 0       21 13:00       22 00:50       20 19:10       23 04:50
2003 0       21 18:30       22 06:20       21 01:00       23 10:30
2004 1       21 00:10       21 12:10       20 06:50       22 16:20
---------------------------------------------------------------------
2005 0       21 06:20       21 18:20       20 12:40       22 22:10
2006 0       21 12:10       22 00:10       20 18:30       23 04:00
2007 0       21 17:40       22 06:00       21 00:10       23 09:50
2008 1       20 23:40       21 11:50       20 06:00       22 15:40
---------------------------------------------------------------------
2009 0       21 05:30       21 17:40       20 11:50       22 21:30
2010 0       21 11:20       21 23:20       20 17:40       23 03:10
2011 0       21 17:10       22 05:10       20 23:30       23 09:00
2012 1       20 23:00       21 11:00       20 05:20       22 14:50
---------------------------------------------------------------------
2013 0       21 04:40       21 16:50       20 11:10       22 20:40
2014 0       21 10:30       21 22:40       20 17:00       23 02:30
2015 0       21 16:20       22 04:30       20 22:50       23 08:20
2016 1       20 22:10       21 10:20       20 04:40       22 14:10
---------------------------------------------------------------------
2017 0       21 04:00       21 16:10       20 10:20       22 19:50
2018 0       21 09:40       21 22:00       20 16:10       23 01:40
2019 0       21 15:30       22 03:50       20 22:00       23 07:30
2020 1       20 21:20       21 09:20       20 03:50       22 13:20
---------------------------------------------------------------------
2021 0       21 03:10       21 15:30       20 09:40       22 19:10
2022 0       21 09:00       21 21:20       20 15:30       23 01:00
2023 0       21 14:40       22 03:10       20 21:20       23 06:40
2024 1       20 20:30       21 09:00       20 03:10       22 12:30
---------------------------------------------------------------------
```

Table 7.24: co_seasons.out file

Chapter 8

SOFTWARE INSTALLATION

8.1 Available packages

8.1.1 F77 package

This is the language used to write the first version of the original code created in July 1993 in the frame of an ESA contract to develop CLUSTER software tools [2].
As calendar computation and rotation matrix used in software are very easy to write, This very simple language has been kept and is the reference for other version in F90, C, and IDL. This is the example taken for installation procedure.

8.1.2 F90 package

This is the translation of the F77 version, with the F90 style, not only a simple compatibility.
Of course all the programs test has been converted, to check that the results with F90 are the same that those obtained with f77 (see section 9.2).

8.1.3 C package

This is the translation of the F77 version done by the f2c program (see https://en.wikipedia.org/wiki/F2c)
A simple retouching of the comments in the code has been added, to make it more clear, as reordering of locale variable declarations, etc.
Of course, as previous, all the programs test has been converted, to check that the result with C is the same that those - obtained with f77 or F90 (see section 9.2).

8.2 Installation on linux system

- *step 1: Copy the directory Rocotlib_V3p0*

Chose the desired package on the server: F77, F90 or C and copy the entire folder anywhere you want. You must have the following subdirectories and the Makefile file:

```
bin          : contains the .exe files
doc          : contains this documentation (pdf) and others useful doc files
in           : contains the input files for the exe programs in bin directory
obj          : contains the .o files and the rocotlib_V3p0.o object library
out          : contains the output  files of the .exe programs in bin directory
sh           : contains the shell script to run the .exe files
               (typically instructions as .exe < .in > .out)
src          : source of the programs and library
test         : test programs to check the library

Makefile     : file to compile and create .o and .exe files from the source files

out_GNU_Linux_x86_64_fort77 : same as out file but created on GNU_Linux_x86_64
      processor with fort77 compiler, for comparison
                            with other compilers
```

Table 8.1: The Rocotlib directory

- *step 2: Edit and choose compiler option in Makefile*

According the chosen package, F77, F90 or C, you have to indicate in the Makefile the used compilor availaible on your machine.
This is done at the beginning of the Makefile. For instance, if you choosed the F77 or F90 distrubution, and have the Intel iFort compilor on your machine, you must uncomment the corresponding line as below (table 8.2). Note you can choose the mode debug (**D**) or optimized (**O**) for the compilor.

```
# Makefile for rocotlib and associated programs
# P. Robert, LPP, 2010 november 16

# ====================
# Select used compiler
# ====================

 FC = ifort
#FC = f90
#FC = f95
#FC = fort77
#FC = gfortran
#FC = gfortran-4.4
#FC = g95

# ======================
# optimize or debug mode
# ======================

 MODE=O
#MODE=D
```

Table 8.2: Compiler choice

If the compilor is not present in the Makefile, you have to add it, and define yourself the compiler options in the corresponding part just after, as it is done for instance for iFort (see table 8.3).

```
# ================================
# Compilator options
# ================================

# Intel compiler

ifeq ($(FC),ifort)
ifeq ($(MODE),D)
# replace check all by all check except check arg_temp_created
FCOMP = -c -CB -g -check format -check uninit -check output_conversion \
                  -traceback -align all -module mod
FLINK =    -CB -g -check format -check uninit -check output_conversion \
                  -traceback -align all -module mod
else
FCOMP = -c -CB -fast -module mod -diag-disable 8290
FLINK =    -CB -fast -module mod -diag-disable 8290
endif
endif
```

Table 8.3: Compiler options

- *step 3: Compile source files and create executable*

Do "**make clean**" to remove all objects and executables files from the old system ; you get something like:

```
*** Removing *.exe *.o, .tmp, * , *.out & core files...
*** Done.
```

Then do "**make all**" to create object and exe file, including the Rocotlib_V3.0.o object file. With fort77 compiler, you get something like shown on table 8.4.

```
fort77 -c src/rocot_check.f -o obj/rocot_check.o
MAIN rocot_check:
strlen:

fort77 -c src/rocotlib_V3p0.f -o obj/rocotlib_V3p0.o
cp_angle_and_ratio:
cv_doty_to_date:
cv_jul2000_to_date:
cv_jul1950_to_date:
cv_weekn_to_date:
cp_geo_dipole_dir:
cv_date_to_dotw:
cv_date_to_doty:
cp_fr_day_name:
cp_fr_month_name:
cv_hms_to_dech:
cv_date_to_jul1950:
cv_date_to_jul2000:
cv_dhms_to_msotd:
cp_nbday_in_month:
cp_leap_year:
cp_gei_sun_dir:
cp_sunrise_sunset:
cv_dech_to_hms:
cv_msotd_to_hmsms:
cp_time_param2:
cp_time_param3:
cp_time_param:
cp_tpn_param:
cp_en_day_name:
cp_en_month_name:
cv_date_to_weekn:
g_gei_geo_dipole_dir:
g_gsm_dipole_tilt_angle:
g_gei_geo_ecliptic_dir:
g_gei_geo_sun_rot:
g_gei_geo_sun_dir:
g_gei_sun_param:
g_rocot_version_number:
print_rocot_info:
r_coordinate_values:
r_coordinate_system:
r_date:
r_time:
t_car_to_sph:
t_dm_to_geo:
t_gei_to_geo:
t_gei_to_gse:
t_gei_to_gsm:
t_gei_to_gseq:
t_gei_to_mag:
t_gei_to_sm:
t_geo_to_dm:
t_geo_to_gei:
t_geo_to_gse:
t_geo_to_gsm:
t_geo_to_gseq:
t_geo_to_mag:
t_geo_to_sm:
t_geo_to_vdh:
t_gse_to_gei:
t_gse_to_geo:
t_gse_to_gsm:
t_gse_to_gseq:
t_gse_to_mfa:
t_gse_to_sr2:
t_gse_to_tpn:
t_gsm_to_gei:
t_gsm_to_geo:
t_gsm_to_gse:
t_gsm_to_gseq:
t_gsm_to_mag:
t_gsm_to_sm:
t_gsm_to_tpn:
t_gseq_to_gei:
t_gseq_to_geo:
t_gseq_to_gse:
t_gseq_to_gsm:
```

Continuation of table 8.4 →

```
t_mag_to_gei:
t_mag_to_geo:
t_mag_to_gsm:
t_mag_to_sm:
t_sm_to_gei:
t_sm_to_geo:
t_sm_to_gsm:
t_sm_to_mag:
t_sph_to_car:
t_sr2_to_gse:
t_sr2_to_mfa:
t_sr2_to_sr:
t_sr_to_sr2:
t_vdh_to_geo:
mat_cp_varmin:
mat_cp_covariance:
mat_diagonalise:
mat_check_ortho:
mat_cp_determin:
mat_cp_eigen_vec:
mat_cp_pythag_func:
mat_normalize_vec:
mat_product:
mat_somme:
mat_diff:
mat_transpose:
mat_change_coord:
mat_write:
mat_write_eigen_vec:

fort77 -static obj/rocot_check.o obj/rocotlib_V3p0.o -o bin/rocot_check.exe
fort77 -c src/rocot_example.f -o obj/rocot_example.o
MAIN rocot_example:
fort77 -static obj/rocot_example.o obj/rocotlib_V3p0.o -o bin/rocot_example.exe

fort77 -c src/rocot_info.f -o obj/rocot_info.o
MAIN rocot_info:
fort77 -static obj/rocot_info.o obj/rocotlib_V3p0.o -o bin/rocot_info.exe

fort77 -c src/rocot_utility.f -o obj/rocot_utility.o
MAIN rocot_utility:
test:
t_sm_to_gse:
t_gse_to_sm:
t_mg_to_gse:
t_gse_to_mag:
t_sm_togsq:
t_gsq_to_sm:
t_mag_to_gsq:
t_gsq_to_mag:
cchalen_:
fort77 -static obj/rocot_utility.o obj/rocotlib_V3p0.o -o bin/rocot_utility.exe

fort77 -c src/rocot_check_mva.f -o obj/rocot_check_mva.o
MAIN rocot_check_mva:
sigtest:
read_para_tube:
writ_para_tube:
cal_b_tube:
cal_j_tube:
carsphe:
sincosp:
vdhxyb:
xybvdh:
fort77 -static obj/rocot_check_mva.o obj/rocotlib_V3p0.o -o bin/rocot_check_mva.exe

fort77 -c src/test_dir_sun.f -o obj/test_dir_sun.o
MAIN test_dir_sun:
convdeg:
colalon:
fort77 -static obj/test_dir_sun.o obj/rocotlib_V3p0.o -o bin/test_dir_sun.exe

fort77 -c src/test_format.f -o obj/test_format.o
MAIN test_format:
fort77 -static obj/test_format.o -o bin/test_format.exe
```

Continuation of table 8.4 →

```
fort77 -c src/test_day_in_week.f -o obj/test_day_in_week.o
MAIN test_day_in_week:
fort77 -static obj/test_day_in_week.o obj/rocotlib_V3p0.o -o bin/test_day_in_week.exe

fort77 -c src/test_week.f -o obj/test_week.o
MAIN test_week:
fort77 -static obj/test_week.o obj/rocotlib_V3p0.o -o bin/test_week.exe

fort77 -c src/co_mag_time.f -o obj/co_mag_time.o
MAIN co_mag_time:
fort77 -static obj/co_mag_time.o obj/rocotlib_V3p0.o -o bin/co_mag_time.exe

fort77 -c src/latlon_to_xyz.f -o obj/latlon_to_xyz.o
MAIN latlon_to_xyz:
fort77 -static obj/latlon_to_xyz.o obj/rocotlib_V3p0.o -o bin/latlon_to_xyz.exe

fort77 -c src/vdh_to_gse.f -o obj/vdh_to_gse.o
MAIN vdh_to_gse:
fort77 -static obj/vdh_to_gse.o obj/rocotlib_V3p0.o -o bin/vdh_to_gse.exe

fort77 -c src/co_julday_table.f -o obj/co_julday_table.o
MAIN co_julday_table:
fort77 -static obj/co_julday_table.o obj/rocotlib_V3p0.o -o bin/co_julday_table.exe

fort77 -c src/co_sunset.f -o obj/co_sunset.o
MAIN co_sunset:
cchapos_:
fort77 -static -O0 obj/co_sunset.o obj/rocotlib_V3p0.o -o bin/co_sunset.exe
fort77 -c   -O0 src/co_seasons.f -o obj/co_seasons.o
MAIN co_seasons:
fort77 -static -O0 obj/co_seasons.o obj/rocotlib_V3p0.o -o bin/co_seasons.exe
fort77 -c   -O0 src/co_sunset_year.f -o obj/co_sunset_year.o
MAIN co_sunset_year:
conv:
arondi:
cchatim_:
ctimcha_:
cchapos_:
fort77 -static -O0 obj/co_sunset_year.o obj/rocotlib_V3p0.o -o bin/co_sunset_year.exe
-> Compilation done.
-> Read lines above to check success.


-> Compilation done.
```

Table 8.4: Result of make all command using fort77 compiler

- *step 4: Check program executions*

If "**`make all`**" has gone correctly, you can try to run "**`sh/rocot_example.sh`**" which should give:

```
-------------------------------------------------------------------
rocot_example.sh :
Running rocot_example.exe with parameters taken from rocot_example.in
results will be in rocot_example.out
Please wait....
STOP rocot_example.exe: normal termination statement executed
rocot_example.sh  terminated
rocot_example.out is located in sh/../out/rocot_example.out
-------------------------------------------------------------------
```

The source of rocot_example.f (in src) is given hereafter in table 8.5.

```
      program rocot_example
c
c     ****************************************************************0**
c *   Rocot_Example: show example of Rocotlib use.
c     P. Robert, July 2000
c     ****************************************************************0**
c
c
      print 100
      print 100, '*****************************************************'
      print 100
      print 100, 'ROCOTEXP:'
      print 100
c
      call r_date(iyear,imonth,iday)
      call r_time(ih,im,is)
      call cv_date_to_doty(iyear,imonth,iday,idoy)
c
      print 100
      print 100, 'date: MM/JJ/YY =', imonth,iday,iyear
      print 100, 'time: HH/MM/SS =', ih,im,is
      print 100, 'day of the year=', idoy
c
      call cp_time_param(iyear,imonth,iday,ih,im,is)
      call g_gei_geo_sun_dir(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)
      call t_gei_to_gsm(sxgei,sygei,szgei,sxgsm,sygsm,szgsm)
c
      print 100
      print 200, 'Sun in GEI:',sxgei,sygei,szgei
      print 200, 'Sun in GEO:',sxgeo,sygeo,szgeo
      print 200, 'Sun in GSM:',sxgsm,sygsm,szgsm
c
      call g_gei_geo_dipole_dir(dxgei,dygei,dzgei,dxgeo,dygeo,dzgeo)
      call t_gei_to_gsm(dxgei,dygei,dzgei,dxgsm,dygsm,dzgsm)
      call t_gei_to_mag(dxgei,dygei,dzgei,dxmag,dymag,dzmag)
      call t_gei_to_sm(dxgei,dygei,dzgei,dxsma,dysma,dzsma)
c
      print 100
      print 200, 'Dip in GEI:',dxgei,dygei,dzgei
      print 200, 'Dip in GEO:',dxgeo,dygeo,dzgeo
      print 200, 'Dip in GSM:',dxgsm,dygsm,dzgsm
      print 200, 'Dip in MAG:',dxmag,dymag,dzmag
      print 200, 'Dip in SM :',dxsma,dysma,dzsma
c
      print 100
      print 200, 'xyz in GEO=', -5.5, -12.5, -7.10
c
      call t_geo_to_gsm(-5.5, -12.5, -7.10,xgsm,ygsm,zgsm)
      call t_geo_to_gse(-5.5, -12.5, -7.10,xgse,ygse,zgse)
c
      print 200, 'xyz in GSM:', xgsm, ygsm, zgsm
      print 200, 'xyz in GSE:', xgse, ygse, zgse
c
      print 100
      print 100, '*****************************************************'
      print 100
c
      100 format(a,3i6)
      200 format(a,3f14.6)

      stop 'rocot_example.exe: normal termination'
      end
```

Table 8.5: rocot_example.f source code

Then the command **cat /out/rocot_example.out** should display:

```
*****************************************************

ROCOTEXP:

iyear,imonth,iday ? (ex: 1990,10,17)
2007  2  7
hour, minute, second ? (ex: 10,45,50)
7 30 33

date: MM/JJ/YY =     2     7  2007
time: HH/MM/SS =     7    30    33
day of the year=    38

Sun in GEI:        .744703       -.612328       -.265465
Sun in GEO:        .315648        .910985       -.265465
Sun in GSM:       1.000000        .000000        .000000

Dip in GEI:       -.174649        .007003        .984606
Dip in GEO:        .054076       -.166214        .984606
Dip in GSM:       -.395728        .000000        .918368
Dip in MAG:        .000000        .000000       1.000000
Dip in SM :        .000000        .000000       1.000000

xyz in GEO=     -5.500000     -12.500000      -7.100000
xyz in GSM:    -11.238584       -.104493     -10.516336
xyz in GSE:    -11.238584      -2.162290     -10.292170

*****************************************************
```

Table 8.6: rocot_example.out file

- *step 5: Check result with your own computer*

The results you find on your computer must be the same as in table 8.6 taking account the accuracy of the computer. Only 6 or 7 digits must be significant.

8.3 Running all test programs

If Step 4has gone correctly, you can run all program of the **sh** or **bin** directories by launch "**sh/RUN_ALL.sh**"
All programs are run with input parameters corresponding to the **in** directory, and all output are written in the **out** directory.

Chapter 9

CHECK PROGRAM AND PORTABILITY

9.1 Check program

9.1.1 General remarks

A coordinate transform library must have a check program, in one hand to check the validity of the calculations, and in other hand to be sure that we find the same results from one computer to another one, or from one compiler to another one.

The rocot_check.exe programs delivered in the standard package allows the user to check the validity of the library for a given machine and compiler. It run from an input date and time, an input vector test, and with an input direction of observation. Next paragraph show examples of input and output data file (rocot_check.in and rocot_check.out files in standard package). Accuracy of the computation is one deduce of the Sun direction computation, which is 0.006 degrees [1].

There is 4 main parts in the rocot_check.out file:

1. **check calendar conversions** of the library;
2. **check basic time parameters** as Sun direction, Dipole direction, Ecliptic pole, Sun equator pole;
3. **check basic transformation**, for instance verify that direction of the Sun in GSM system is well the X axis, that Dipole direction in MAG system is well the Z axis, and so on;
4. **check the usual transformations**, by changing the input vector across all available paths showed on the schematic diagram of section 2.2.

Mainly, the vector is transformed:

1. as a "star" centered on GE0 system (see schematic diagram figure 2.1 of section 2.2);
2. as a "star" centered on GEI system;
3. as a "star" centered on GSE system;
4. as a "clockwise ring cumulative transformation", from GEO to GEO, in the other sense;
5. some specific systems from GSE (SR2, ISR2, SR, MFA)
6. Finally, input direction of the point of observation is used to check local coordinate systems, i.e. DM and VDH.

At each "go and back", one check that the original vector is returned to the same value, taking into account accuracy of the computation : 32 bits simple precision, 6 significant digits, and input Sun direction within 0.006 degrees [1].

So do not be worry if you get values as 0.999999 rather than 1.0, or other things like that.

9.1.2 Input rocot_check.in data file for running check program

Input parameters for rocot_check program has been choosen as below:

```
1990 10 17     # iyear, imonth, iday
12 30 1        # ihour, imin, isec
 5. 30. 60.    # R, Theta, Phi (deg.) of test vector
45. 30.        # Geographic latitude and longitude (deg.) of point of observation
2.  170. 10.   # R, Theta, Phi (deg.) of spin axis in GSE
0.25 30.       # Spin frequency (Hz), Spin phase (Deg.)
1.2345         # delta T from Spin phase time measurement
```

9.1.3 Output rocot_check.out file from check program execution

This file is displayed on several pages as table 9.1.

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

     Coordinates Transformation Library  ROCOTLIB
                 ____________________

 Rocotche program: Check transformations of Rocotlib

          Patrick ROBERT, CRPE, November 1992

          Revised Version 3.0 - Feb 2017

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


SUMMARY OF CHECK OPERATIONS :
****************************

  I) TEST OF ROCOTLIB CALENDAR CONVERSIONS

 II) CHECK BASIC TIME PARAMETERS

III) CHECK BASIC TRANSFORMATIONS

 IV) TEST OF ROCOTLIB TRANSFORMATION SUBROUTINES

    A) STAR TRANSFORMATIONS AROUND  GEI  SYSTEM

       GEI ------------------------------> MAG
       GEI ------------------------> SM
       GEI -------------------> GSM
       GEI -------------> GSE

    B) COUNTERCLOCKWISE RING CUMULATIVE TRANSFORM.

       GEO ->GEI ->GSQ ->GSE ->GSM ->SM ->MAG ->GEO

    C) CLOCKWISE RING CUMULATIVE TRANSFORMATIONS

       GEO ->MAG ->SM ->GSM ->GSE ->GSQ ->GEI ->GEO

    D) STAR TRANSFORMATIONS AROUND  GEO  SYSTEM

       GEO ------> SM
       GEO ------------> GSM
       GEO ------------------> GSE
       GEO ------------------------> GSQ

    E) LOCAL SYSTEMS

       GEO ----> DM
       GEO ----> VDH

    F) SPACECRAFT SYSTEMS

       GEO -> GSE -> SR2 -> SR -> SR2 -> GSE -> GEO

    G) PSEUDO-MAGNETOPAUSE SYSTEMS

       GSE ----> TPN
       GSM ----> TPN
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
INPUT DATA TEST :
***************

INPUT DATE AND TIME:
 iyear,imonth,iday ? (ex: 1990,10,17)
 1990 10 17
 hour, minute, second ? (ex: 10,45,50)
 12 30  1

iyear,imonth,iday:  1990 10 17
ih, im, is       :  12 30 01

INPUT VECTOR TEST:
r, teta,phi of any vector in GEO system ? (deg.)
     5.00000   30.000   60.000

INPUT GEOG. LAT AND LONG. FOR LOCAL SYSTEMS TEST:
lat. and long. of local direct. of observation ? (d.)
  45.000   30.000

INPUT SPIN DIRECTION for SR and SR2 SYSTEMS:
r, teta,phi of spin axis in GSE system ? (deg.)
     2.00000  170.000   10.000

SPIN FREQUENCY, SPIN PHASE & Deltat T :
 Spin frequency, Spin phase ? (Hz, deg.)
     0.25000   30.000
delta T ? (sec. from time of spin phase)
     1.23450



RUN OF CHECK PROGRAM
********************

Please wait...
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
            T E S T   O F   R O C O T L I B  Version 3.0


I) TEST OF ROCOTLIB CALENDAR CONVERSIONS
   =====================================

INPUT DATE AND TIME:

year, month,  day   :       1990    10    17
hour, minute, second:         12    30     1

computed values:

decimal hour of the day:   12.5003
day of the year:             290
Julian day from 1-1-1950:  14899
Julian day from 1-1-2000:  -3363
leap year (1=yes,0=no):        0

recompute date from year and day of the year:  1990    10    17
recompute date from Julian day 1950         :  1990    10    17
recompute date from Julian day 2000         :  1990    10    17
recompute time from decimal hour:    12    30     1


II) CHECK BASIC TIME PARAMETERS
    ===========================

SUN PARAMETERS IN GEI SYSTEM

Greenwich Sideral Time (deg.):   213.253
ecliptic longitude     (deg.):   203.879
right ascention        (deg.):   202.100
declination            (deg.):    -9.265

DIPOLE TILT ANGLE      (deg.):    -3.789

SUN DIRECTION
                     in GEI:
                                X=   -.91444     r   =     1.00000
                                Y=   -.37132     teta=    99.265
                                Z=   -.16100     phi =  -157.900
                     in GEO:
                                X=    .96832     r   =     1.00000
                                Y=   -.19090     teta=    99.265
                                Z=   -.16100     phi =   -11.152

DIPOLE DIRECTION
                     in GEI:
                                X=   -.14729     r   =     1.00000
                                Y=    .11476     teta=    10.761
                                Z=    .98241     phi =   142.077
                     in GEO:
                                X=    .06025     r   =     1.00000
                                Y=   -.17673     teta=    10.761
                                Z=    .98241     phi =   -71.176

ECLIPTIC DIRECTION
                     in GEI:
                                X=    .00000     r   =     1.00000
                                Y=   -.39780     teta=    23.440
                                Z=    .91747     phi =   -90.000
                     in GEO:
                                X=    .21812     r   =     1.00000
                                Y=    .33266     teta=    23.440
                                Z=    .91747     phi =    56.747

SUN EQUATOR DIRECTION
                     in GEI:
                                X=    .12170     r   =     1.00000
                                Y=   -.42440     teta=    26.200
                                Z=    .89726     phi =   -74.000
                     in GEO:
                                X=    .13094     r   =     1.00000
                                Y=    .42164     teta=    26.200
                                Z=    .89726     phi =    72.747
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
III) CHECK BASIC TRANSFORMATIONS
     ===========================

     Check (S,E) angle is equal to 90 deg. : angle=  90.000    (deg.)

1) input vector: Sun direction in GEI system
   ----------------------------------------

                               X=   -.91444     r   =     1.00000
                               Y=   -.37132     teta=    99.265
                               Z=   -.16100     phi =  -157.900

                              in GEO system

                               X=    .96832     r   =     1.00000
                               Y=   -.19090     teta=    99.265
                               Z=   -.16100     phi =   -11.152


   check Y=0 in SM  system with t_gei_to_sm:

                               X=    .99781     r   =     1.00000
                               Y=    .00000     teta=    93.789
                               Z=   -.06609     phi =      .000

                          with t_geo_to_sm:

                               X=    .99781     r   =     1.00000
                               Y=    .00000     teta=    93.789
                               Z=   -.06609     phi =      .000

   check Y=0 and Z=0 in GSM system with t_gei_to_gsm:

                               X=   1.00000     r   =     1.00000
                               Y=    .00000     teta=    90.000
                               Z=    .00000     phi =      .000

                          with t_geo_to_gsm:

                               X=   1.00000     r   =     1.00000
                               Y=    .00000     teta=    90.000
                               Z=    .00000     phi =      .000

   check Y=0 and Z=0 in GSE system with t_gei_to_gse:

                               X=   1.00000     r   =     1.00000
                               Y=    .00000     teta=    90.000
                               Z=    .00000     phi =      .000

                          with t_geo_to_gse:

                               X=   1.00000     r   =     1.00000
                               Y=    .00000     teta=    90.000
                               Z=    .00000     phi =      .000

   check Y=0 and Z=0 in GSQ system with t_gei_to_gseq:

                               X=   1.00000     r   =     1.00000
                               Y=    .00000     teta=    90.000
                               Z=    .00000     phi =      .000

                          with t_geo_to_gseq:

                               X=   1.00000     r   =     1.00000
                               Y=    .00000     teta=    90.000
                               Z=    .00000     phi =      .000
```

Continuation of table 9.1 →

```
2) input vector: Dipole direction in GEI system
   --------------------------------------------

                                X=   -.14729     r   =     1.00000
                                Y=    .11476     teta=    10.761
                                Z=    .98241     phi =   142.077

                                  in GEO system

                                X=    .06025     r   =     1.00000
                                Y=   -.17673     teta=    10.761
                                Z=    .98241     phi =   -71.176

   check X=0 and Y=0 in MAG system with t_gei_to_mag:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

                           with t_geo_to_mag:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

   check X=0 and Y=0 in SM  system with t_gei_to_sm:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

                           with t_geo_to_sm:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

   check Y=0 in GSM system with t_gei_to_gsm:

                                X=   -.06609     r   =     1.00000
                                Y=    .00000     teta=     3.789
                                Z=    .99781     phi =  -180.000

                           with t_geo_to_gsm:

                                X=   -.06609     r   =     1.00000
                                Y=    .00000     teta=     3.789
                                Z=    .99781     phi =   180.000

3) input vector: Ecliptic direction in GEI system
   ----------------------------------------------

                                X=    .00000     r   =     1.00000
                                Y=   -.39780     teta=    23.440
                                Z=    .91747     phi =   -90.000

                                    in GEO system

                                X=    .21812     r   =     1.00000
                                Y=    .33266     teta=    23.440
                                Z=    .91747     phi =    56.747

   check X=0 and Y=0 in GSE system with t_gei_to_gse:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

                           with t_geo_to_gse:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

   check X=0 in GSQ system with t_gei_to_gseq:

                                X=    .00000     r   =     1.00000
                                Y=   -.07931     teta=     4.549
                                Z=    .99685     phi =   -90.000

                           with t_geo_to_gseq:

                                X=    .00000     r   =     1.00000
                                Y=   -.07931     teta=     4.549
                                Z=    .99685     phi =   -90.000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
4) input vector: Sun equator in GEI system
   ---------------------------------------

                                X=    .12170     r   =     1.00000
                                Y=   -.42440     teta=    26.200
                                Z=    .89726     phi =   -74.000

                              in GEO system

                                X=    .13094     r   =     1.00000
                                Y=    .42164     teta=    26.200
                                Z=    .89726     phi =    72.747


   check and Y=0 in GSQ system with t_gei_to_gseq:

                                X=   -.09815     r   =     1.00000
                                Y=    .00000     teta=     5.633
                                Z=    .99517     phi =  -180.000

                             with t_geo_to_gseq:

                                X=   -.09815     r   =     1.00000
                                Y=    .00000     teta=     5.633
                                Z=    .99517     phi =   180.000


5) input vector: North geographic in GEO system
   --------------------------------------------

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000


   check X=0 and Y=0 in GEI system with t_geo_to_gei:

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

   check Y=0 in MAG system with t_gei_to_mag:

                                X=   -.18672     r   =     1.00000
                                Y=    .00000     teta=    10.761
                                Z=    .98241     phi =   180.000

                             with t_geo_to_mag:

                                X=   -.18672     r   =     1.00000
                                Y=    .00000     teta=    10.761
                                Z=    .98241     phi =   180.000


IV) TEST OF ROCOTLIB TRANSFORMATION SUBROUTINES
    ===========================================

1) input choosed test vector in GEO system :

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

2) converts GEO system to GEI system by t_geo_to_gei

                                X=    .14185     r   =     5.00000
                                Y=  -2.49597     teta=    30.000
                                Z=   4.33013     phi =   -86.747
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
A) STAR TRANSFORMATIONS AROUND  GEI  SYSTEM
   ========================================

          # 3
      GEI --------> MAG
          # 4
      GEI --------> SM
          # 5
      GEI --------> GSM
          # 6
      GEI --------> GSE


1) t# 3 converts GEI system to MAG system byt_gei_to_mag

                               X=  -2.42551     r   =     5.00000
                               Y=   1.88173     teta=    37.877
                               Z=   3.94664     phi =   142.195

            come back to GEI system by t_mag_to_gei

                               X=    .14185     r   =     5.00000
                               Y=  -2.49597     teta=    30.000
                               Z=   4.33013     phi =   -86.747

         difference beetween first GEI vector: angle=      .000    (deg.)
                                               ratio=     1.00000


2) t# 4 converts GEI system to SM  system byt_gei_to_sm

                               X=    .36158     r   =     5.00000
                               Y=   3.04848     teta=    37.877
                               Z=   3.94664     phi =    83.236

            come back to GEI system by t_sm_to_gei

                               X=    .14185     r   =     5.00000
                               Y=  -2.49597     teta=    30.000
                               Z=   4.33013     phi =   -86.747

         difference beetween first GEI vector: angle=      .000    (deg.)
                                               ratio=     1.00000


3) t# 5 converts GEI system to GSM system byt_gei_to_gsm

                               X=    .09996     r   =     5.00000
                               Y=   3.04848     teta=    37.591
                               Z=   3.96191     phi =    88.122

            come back to GEI system by t_gsm_to_gei

                               X=    .14185     r   =     5.00000
                               Y=  -2.49597     teta=    30.000
                               Z=   4.33013     phi =   -86.747

         difference beetween first GEI vector: angle=      .000    (deg.)
                                               ratio=     1.00000


4) t# 6 converts GEI system to GSE system by t_gei_to_gse

                               X=    .09996     r   =     5.00000
                               Y=    .57634     teta=     6.718
                               Z=   4.96567     phi =    80.160

            come back to GEI system by t_gse_to_gei

                               X=    .14185     r   =     5.00000
                               Y=  -2.49597     teta=    30.000
                               Z=   4.33013     phi =   -86.747

         difference beetween first GEI vector: angle=      .000    (deg.)
                                               ratio=     1.00000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
B) COUNTERCLOCKWISE RING CUMULATIVE TRANSFORMATIONS
   ================================================

    # 2      7      8      9     10     11     12
  GEO -> GEI -> GSQ -> GSE -> GSM -> SM -> MAG -> GEO


   input choosed test vector in GEO system :

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

1) t# 2 converts GEO system to GEI system by t_geo_to_gei

                                X=    .14185     r   =     5.00000
                                Y=  -2.49597     teta=    30.000
                                Z=   4.33013     phi =   -86.747

2) t# 7 converts GEI system to GSQ system by t_gei_to_gseq

                                X=    .09996     r   =     5.00000
                                Y=    .18069     teta=     2.367
                                Z=   4.99573     phi =    61.047

3) t# 8 converts GSQ system to GSE system by t_gseq_to_gse

                                X=    .09996     r   =     5.00000
                                Y=    .57634     teta=     6.718
                                Z=   4.96567     phi =    80.160

4) # 9 converts GSE system to GSM system by t_gse_to_gsm

                                X=    .09996     r   =     5.00000
                                Y=   3.04848     teta=    37.591
                                Z=   3.96191     phi =    88.122

5) t# 10 converts GSM system to SM  system by t_gsm_to_sm

                                X=    .36158     r   =     5.00000
                                Y=   3.04848     teta=    37.877
                                Z=   3.94664     phi =    83.236

6) t# 11 converts SM  system to MAG system by t_sm_to_mag

                                X=  -2.42551     r   =     5.00000
                                Y=   1.88172     teta=    37.877
                                Z=   3.94664     phi =   142.195

7) t# 12 converts MAG system to GEO system by t_mag_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

         difference beetween first GEO vector: angle=      .000    (deg.)
                                               ratio=     1.00000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
C) CLOCKWISE RING CUMULATIVE TRANSFORMATIONS
   ========================================

    # 12     11    10     9      8      7      2
  GEO -> MAG -> SM -> GSM -> GSE -> GSQ -> GEI -> GEO


   input choosed test vector in GEO system :

                               X=   1.25000     r   =     5.00000
                               Y=   2.16506     teta=    30.000
                               Z=   4.33013     phi =    60.000

1) t# 12 converts GEO system to MAG system by t_geo_to_mag

                               X=  -2.42551     r   =     5.00000
                               Y=   1.88173     teta=    37.877
                               Z=   3.94664     phi =   142.195

2) t# 11 converts MAG system to SM  system by t_mag_to_sm

                               X=    .36158     r   =     5.00000
                               Y=   3.04848     teta=    37.877
                               Z=   3.94664     phi =    83.236

3) t# 10 converts SM  system to GSM system by t_sm_to_gsm

                               X=    .09996     r   =     5.00000
                               Y=   3.04848     teta=    37.591
                               Z=   3.96191     phi =    88.122

4) t# 9 converts GSM system to GSE system by t_gsm_to_gse

                               X=    .09996     r   =     5.00000
                               Y=    .57634     teta=     6.718
                               Z=   4.96567     phi =    80.160

5) t# 8 converts GSE system to GSQ system by t_gse_to_gseq

                               X=    .09996     r   =     5.00000
                               Y=    .18069     teta=     2.367
                               Z=   4.99573     phi =    61.047

6) t# 7 converts GSQ system to GEI system by t_gseq_to_gei

                               X=    .14185     r   =     5.00000
                               Y=  -2.49597     teta=    30.000
                               Z=   4.33013     phi =   -86.747

         difference beetween first GEI vector: angle=      .000    (deg.)
                                               ratio=     1.00000


7) t# 2 converts GEI system to GEO system by t_gei_to_geo

                               X=   1.25000     r   =     5.00000
                               Y=   2.16506     teta=    30.000
                               Z=   4.33013     phi =    60.000

         difference beetween first GEO vector: angle=      .000    (deg.)
                                               ratio=     1.00000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
D) STAR TRANSFORMATIONS AROUND  GEO  SYSTEM
   ========================================

         # 13
     GEO --------> SM
         # 14
     GEO --------> GSM
         # 15
     GEO --------> GSE
         # 16
     GEO --------> GSQ

   input choosed test vector in GEO system :

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

1) t# 13 converts GEO system to SM  system by t_geo_to_sm

                                X=    .36158     r   =     5.00000
                                Y=   3.04848     teta=    37.877
                                Z=   3.94664     phi =    83.236

             come back to GEO system by t_sm_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000


2) t# 14 converts GEO system to GSM system by t_geo_to_gsm

                                X=    .09996     r   =     5.00000
                                Y=   3.04848     teta=    37.591
                                Z=   3.96191     phi =    88.122

             come back to GEO system by t_gsm_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000


3) t# 15 converts GEO system to GSE system by t_geo_to_gse

                                X=    .09996     r   =     5.00000
                                Y=    .57634     teta=     6.718
                                Z=   4.96567     phi =    80.160

             come back to GEO system by t_gse_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000


4) t# 16 converts GEO system to GSQ system by t_geo_to_gseq

                                X=    .09996     r   =     5.00000
                                Y=    .18069     teta=     2.367
                                Z=   4.99573     phi =    61.047

             come back to GEO system by t_gseq_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
E) LOCAL SYSTEMS TRANSFORMATIONS
   ==============================

         # 17
     GEO --------> DM
         # 18
     GEO --------> VDH

   first direction of observation in GEO system:
        geog. lat., long. =    45.000    30.000

1) t# 17 converts GEO system to DM  system by t_geo_to_dm

                                X=   2.62675     r   =     5.00000
                                Y=   1.58877     teta=    37.877
                                Z=   3.94664     phi =    31.167

             come back to GEO system by t_dm_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000


2) t# 18 converts GEO system to VDH system by t_geo_to_vdh

                                X=   4.59279     r   =     5.00000
                                Y=   1.25000     teta=    72.170
                                Z=   1.53093     phi =    15.225

             come back to GEO system by t_vdh_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000


   second direction of observation in GEO system:
        geog. lat., long. =    45.000    30.000

3) t# 17 converts GEO system to DM  system by t_geo_to_dm

                                X=   2.62675     r   =     5.00000
                                Y=   1.58877     teta=    37.877
                                Z=   3.94664     phi =    31.167

             come back to GEO system by t_dm_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000


4) t# 18 converts GEO system to VDH system by t_geo_to_vdh

                                X=   4.59279     r   =     5.00000
                                Y=   1.25000     teta=    72.170
                                Z=   1.53093     phi =    15.225

             come back to GEO system by t_vdh_to_geo

                                X=   1.25000     r   =     5.00000
                                Y=   2.16506     teta=    30.000
                                Z=   4.33013     phi =    60.000

          difference beetween first GEO vector: angle=      .000    (deg.)
                                                ratio=     1.00000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
F) LOCAL SPACECRAFT SYSTEMS TRANSFORMATIONS
   ======================================

    # 15    19     21    -21    -19    -15
  GEO -> GSE -> SR2 -> SR -> SR2 -> GSE -> GEO


   input choosed test vector in GEO system :

                               X=   1.25000     r   =     5.00000
                               Y=   2.16506     teta=    30.000
                               Z=   4.33013     phi =    60.000

   input choosed spin axis in GSE system :

                               X=    .34202     r   =     2.00000
                               Y=    .06031     teta=   170.000
                               Z=  -1.96962     phi =    10.000

   input choosed Spin frequency, Spin phase, dt:

                 Spin frequency=    .2500 Hz
                 Spin phase    =  30.0000 deg.
                 delta T       =   1.2345 sec.


1) t# 15 converts GEO system to GSE system by t_geo_to_gse

                               X=    .09996     r   =     5.00000
                               Y=    .57634     teta=     6.718
                               Z=   4.96567     phi =    80.160

2) t# 19 converts GSE system to SR2 system by t_gse_to_sr2

                               X=    .94425     r   =     5.00000
                               Y=   -.72804     teta=   166.204
                               Z=  -4.85575     phi =   -37.633

3) t#-21 converts SR2 system to SR  system by t_sr2_to_sr

                               X=   -.57328     r   =     5.00000
                               Y=  -1.04547     teta=   166.204
                               Z=  -4.85575     phi =  -118.738

4) t# 21 converts SR  system to SR2 system by t_sr_to_sr2

                               X=    .94425     r   =     5.00000
                               Y=   -.72804     teta=   166.204
                               Z=  -4.85575     phi =   -37.633

5) t#-19 converts SR2 system to GSE system by t_sr2_to_gse

                               X=    .09996     r   =     5.00000
                               Y=    .57634     teta=     6.718
                               Z=   4.96567     phi =    80.160

6) t#-15 converts GSE system to GEO system by t_gse_to_geo

                               X=   1.25000     r   =     5.00000
                               Y=   2.16506     teta=    30.000
                               Z=   4.33013     phi =    60.000

         difference beetveen first GEO wector: angle=      .000    (deg.)
                                               ratio=     1.00000
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
G) MAGNETIC FIELD ALIGNED TRANSFORMATION
   ====================================

  GSE -> MFA
  SR2 -> MFA

   input choosed test vector in GSE system :

                               X=    .09996     r   =     5.00000
                               Y=    .57634     teta=     6.718
                               Z=   4.96567     phi =    80.160

   input choosed DC mag. field in GSE system :

                               X=  30.00000     r   =   208.32666
                               Y=  50.00000     teta=    16.254
                               Z= 200.00000     phi =    59.036


1) t# 22 converts GSE system to MFA system by t_gse_to_mfa

                               X=   -.61494     r   =     5.00000
                               Y=   -.64522     teta=    10.269
                               Z=   4.91991     phi =  -133.623

   input choosed test vector in SR2 system :

                               X=    .94425     r   =     5.00000
                               Y=   -.72804     teta=   166.204
                               Z=  -4.85575     phi =   -37.633

   input choosed DC mag. field in SR2 system :

                               X=  63.48236     r   =   208.32666
                               Y= -56.09747     teta=   156.005
                               Z=-190.32358     phi =   -41.466

2) t# 20 converts SR2 system to MFA system by t_sr2_to_mfa

                               X=   -.61494     r   =     5.00000
                               Y=   -.64522     teta=    10.269
                               Z=   4.91991     phi =  -133.623
```

Continuation of table 9.1 →

Continuation of rocot_check.out file...

```
H) PSEUDO-MAGNETOPAUSE SYSTEMS
   ============================

       GSE ----> TPN
       GSM ----> TPN

1) Computation of the TPN system

   Input position in GSE system, in RE:
   (supposed to be on the magnetopause)

   Xo, Yo, Zo in GSE =     5.720     8.230     9.220

   abcisse of subsolar point for paraboloid model:    12.000

   TPN vectors in GSE:

   Tx,Ty,Tz=       .713      -.467      -.523
   Px,Py,Pz=       .000       .746      -.666
   Nx,Ny,Nz=       .701       .475       .532


   input choosed test vector in GSE system :

                                X=    .70138     r   =     1.00000
                                Y=    .47466     teta=    57.876
                                Z=    .53176     phi =    34.088

2) t# 23 convert GSE system to TPN system by t_gse_to_tpn

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

3) t# 9 convert GSE system to GSM system by t_gse_to_gsm

                                X=    .70138     r   =     1.00000
                                Y=    .68058     teta=    77.769
                                Z=    .21186     phi =    44.138

4) t# 9 convert GSE position of S/C to GSM position

   Xo, Yo, Zo in GSM =     5.720    11.800     3.673

5) t# 24 convert GSM system to TPN system by t_gsm_to_tpn

                                X=    .00000     r   =     1.00000
                                Y=    .00000     teta=      .000
                                Z=   1.00000     phi =      .000

================================
END OF rocot_check test program
================================
```

Table 9.1: rocot_check.out file

9.2 Compare rocot_check.out files

The rocot_check.out file given in table 9.1 of section 9.1 has been computed on a GNU/Linux operating system with a processor type x86_64 with fort77 compiler.

Note that The fort77 compiler invokes the f2c command transparently, so it can be used like a real Fortran compiler. It can be used to compile Fortran and C, and to link it with the f2c libraries. So we will see in section 9.5.3 that it is not surprising that the two codes, f77 and C, lead to exactly the same result.

When you use another machines, with another compiler, the first thing to do is to compare the rocot_example.out file with the one given in section 8.6.
So we remind you have to compile with:

```
make clean
make all
```

and run all programs by

```
sh/RUN_ALL.sh
```

Note that the package is delivered with a **out_GNU_Linux_x86_64_fort77_f_corr** directory, containing all the .out files corresponding to all the .exe files. So a convenient way to do the comparison is to run all the available programs as explained in section 8.3 and then to compare the .out files.

One recommend to do first:

```
diff ./out/rocot_example.out
./out_GNU_Linux_x86_64_fort77_f_corr/rocot_example.out
```

If discrepancies are on 6^{th} or 7^{th} digits, there is no problem. So you can then check the rocot_check.out files by doing:

```
diff ./out/rocot_check.out
./out_GNU_Linux_x86_64_fort77_f_corr/rocot_check.out
```

If the output is too long, redirect the output to a file, as:

```
diff ./out/rocot_check.out ...../rocot_check.out > rocot_check.txt
```

Then take time to examine carefully the **rocot_check.txt** file.

9.3 Running all available programs

Before doing anything, verify that you have defined your compiler in the Makefile (see section 8.2) and running **make clean** and **make all**. So the directory **bin** will contains all executables files such as ***.exe** .
Then all available programs can be executed by running **sh/RUN_ALL.sh**. Details of all given programs are in chapter 7.

9.4 Compare other .out files

If there is a good agreement between the **rocot_check.out** files, you can be quiet on the validity of the library. Nevertheless, you can also check as the same manner other .out files, but you must use first the **correct_out.sh** shell script before doing comparison. Indeed, according compiler

used, the output can be different.

For example, with **fort77** or **gcc** we get result as:

X= -.91444

while with **gfortran**, with the same format, we get:

X= -0.91444

So it you make a linux **diff** command on the two output files, you will find a lot of insignificant difference. That is why we use a small shell script to convert the output file with the same convention (see table 9.2 below).

If you use a new compiler, for instance **ifort**, on f77 package, you first identify your machine/system by enter:

uname -a

you get something like:

Linux lx-robert 4.4.0-64-generic #85-Ubuntu SMP Mon Feb 20 11:50:30 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux

Then you have to run the command:

correct_out.sh out_GNU_Linux_x86_64_ifort_f_corr

and you will get in the directory **out_GNU_Linux_x86_64_ifort_f_corr** containing all the .out files, with the corrections of format making it possible to make comparisons without the previous disadvantages.

Then, by running a shell-script as, for instance:

compare_out_GNU_Linux_x86_64_fort77_gfortran.sh

you get all the difference between the results of **fort77** compiler and **gfortran**. Example of compare shell scrip is given below (see table 9.3).

```
#!/bin/sh

# ------------------------------------------------------
# utility for compare .out files
# numbers as  ".123"    will be changed as " 0.123"
# numbers as "-.123"    will be changed as "-0.123"
# numbers as "-.00000"  will be changed as " 0.00000"
# numbers as "-180.000" will be changed as " 180.000"
# numbers as "-.000"    will be changed as " 0.000"

# Example of use:
# correct_out.sh out_GNU_Linux_x86_64_gfortran_f_corr
# ------------------------------------------------------

out_corr=$1

list1=`ls -1 out/*.out`
list2=`ls -1 out/*.resu`

list="$list1 $list2"

for name in $list
do
echo $name ...
nameout=`basename $name`
sed -e s"/  / 0/g" -e s"/ -/-0/g"  -e s"/-000000/ 000000/g"
    -e s"/-180000/ 180000/g" -e s"/ -0000 /  0000 /g" $name > $out_corr/$nameout
done
```

Table 9.2: correct_out.sh shell script file

```
#!/bin/sh

dir1=out_GNU_Linux_x86_64_fort77_f_corr
dir2=out_GNU_Linux_x86_64_gfortran_f_corr

ls -1  $dir1/*.out | sed s"%.*/%%" > toto1.tmp
 ; diff $dir1/& $dir2/&%" toto1.tmp > toto2.tmp------------
sh toto2.tmp
rm toto*.tmp
```

Table 9.3: compare_out_GNU_Linux_x86_64_fort77_gfortran.sh shell script file

9.5 Differences between codes and used compilers

Whatever the program used, it is reliable only if it give the same results with various compilers on various machines and OS.
Tests done here concern only various compiler, on the same machine (see 9.4) but between the 3 delivered language: F77, F90 and C. Results being satisfactory, no other test has been done.

9.5.1 Results obtained with two different compilers on the same code

The checked code is the F77 code, with **ifort** and **gfortran**. Procedure used if the shell script `compare_out_GNU_Linux_x86_64_fort77_gfortran.sh` given in table 9.3.

As we can see, this shell compare all the .out files present in the two directories:
out_GNU_Linux_x86_64_fort77_f_corr
and **out_GNU_Linux_x86_64_gfortran_f_corr**.

Results are given in table 9.4 below. We can see that there is no differences for following programs:

- `co_julday_table.out`
- `co_mag_time.out`
- `rocot_check.out`
- `rocot_info.out`
- `test_day_in_week.out`
- `test_format.out`
- `test_week.out`
- `vdh_to_gse.out`
- `co_sunset.out`

Some minors differences exists for following programs:

- **co_seasons.out** : Only a difference of about 10 mn is observed on solstices time computation. Since the time accuracy of solstices and equinoxes is estimated about one hour, difference is negligible.
- **co_sunset_year.out** : Only one minute angle on the azimuth of the direction of solar meridian and sunset, which is negligible.
- **latlon_to_xyz.out** : Problem of output format: results are good until 7^{th} digit, which is the single precision of the computer.
- **rocot_check_mva.out** : After matrix diagonalisation, some eigen values are not strictly equal to zero, both with f77 and gfortran. Nevertheless, the maximum eigen value being 5000, the minimum found is 6.10^{-4} , that is to say a ratio of 10^{-7}, close to zero if we consider 7 digits for single precision.
  This is the same thing for dot product of vectors which should be equal to zero, observed value being 10^{-7}.
  At last, teta and phi angle differs on the 7^{th} digit, which is no significant.
- **rocot_example.out.out** : difference observed at the 6^{th} digit, so, once again, effect of single precision.
- **test_dir_sun.out.out** : difference of 10^{-3} degree on the last angle, corresponding to the 6^{th} digit of a single precision number.

9.5.2 Results obtained with the same compiler on F77 and F90 codes

Comparison was done on **F77** codes (*.f) and **F90** codes (*.f90) with the same **gfortran** compiler. To do this comparison, we used the following shell
compare_with_out_GNU_Linux_x86_64_gfortran_f.sh given in the F90 package.

Absolutely no difference for all programs tested.

9.5.3 Results obtained with gcc on C code and fort77 on F77 codes

Comparison was done on **C** codes (*.c) and **F77** codes (*.f) with **gcc** compiler for C codes and **F77** compiler for F77 codes. To do this comparison, we used the following shell
compare_with_out_GNU_Linux_x86_64_fort77_f.sh given in the F90 package.

Absolutely no difference for all programs tested.

But this is obvious in fact, since the C codes has been derived from F77 codes via the f2c converter, even if the C codes has then be Has been improved, mainly for its visibility. Furthermore the Makefile for C codes use the f2c library. So, no surprising that we found exactly the same result.

9.5.4 Results obtained with gcc on C code and gfortran on F77 or F90 codes

Comparison has been done both on F77 codes (*.f) and F90 codes (*.f90) with **gfortran** compiler.

Results founds are exactly the same that those of table 9.4

That is to say the results are the same that comparison between **Fort77** and **gfortran**.

9.5.5 Summary

Previous results show that only compiler choice is significant, and not code translation.
In fact, **f77**, **f95** or **gfortran** *are the same compiler*. They give same results, whatever *.f or *.f90 used Fortran files.

In other hand, **fort77** use **f2c** compiler, as **gcc**. So **fort77** and **gcc** give same results on *.f or *.c files.

At least small differences observed between use of **fort77** and **gfortran** are in the accuracy of the computer. If more important differences appears in the future, the user will check the possibility of using double precision computations.

```
compare_out_GNU_Linux_x86_64_fort77_gfortran.sh
----------------------------------------
co_julday_table.out :
----------------------------------------
co_mag_time.out :
----------------------------------------
co_sunset.out :
13c13
<  merid. time (UT)= 12:03:20 sunrise= 07:34:00 sunset= 16:33:00 durd=  8:59:00
---
>  merid. time (UT)= 12:03:20 sunrise= 07:28:20 sunset= 16:38:50 durd=  9:10:30
20c20
<  merid. time (UT)= 12:06:50 sunrise= 07:21:50 sunset= 16:52:20 durd=  9:30:30
---
>  merid. time (UT)= 12:06:50 sunrise= 07:16:40 sunset= 16:57:30 durd=  9:40:50
27c27
<  merid. time (UT)= 12:12:00 sunrise= 07:52:20 sunset= 16:32:20 durd=  8:40:00
---
>  merid. time (UT)= 12:12:00 sunrise= 07:46:00 sunset= 16:38:30 durd=  8:52:30
34c34
<  merid. time (UT)= 12:12:00 sunrise= 07:53:30 sunset= 16:31:00 durd=  8:37:30
---
>  merid. time (UT)= 12:12:00 sunrise= 07:47:10 sunset= 16:37:20 durd=  8:50:10
----------------------------------------
co_sunset_year.out :
162c162
< 2017-05-11  04:16 359d24'  61d20'  11:47  59d08' 180d01'  19:19 359d22' 298d57'  15:0
---
> 2017-05-11  04:16 359d24'  61d20'  11:47  59d08' 180d00'  19:19 359d22' 298d57'  15:0
401c401
< 2017-12-08  07:33 359d23' 125d08'  11:43  18d22' 180d01'  15:52 359d23' 234d49'   8:1
---
> 2017-12-08  07:33 359d23' 125d08'  11:43  18d22' 180d01'  15:52 359d23' 234d48'   8:1
----------------------------------------
latlon_to_xyz.out :
2,3c2,3
<   50.  20.  6000.
<   4319.07764  1572.01587  3856.72559
---
>    50.0000000       20.0000000       6000.00000
>    4319.07764       1572.01587       3856.72583
----------------------------------------
rocot_check_mva.out :
48,50c48,50
<                0.55875E+04    0.00000E+00    0.00000E+00
<                0.00000E+00   -0.63610E-03   -0.30518E-04
<                0.00000E+00   -0.30518E-04    0.20369E+05
---
>                0.55875E+04    0.11176E-07    0.00000E+00
>                0.11176E-07   -0.23651E-03    0.00000E+00
>                0.00000E+00    0.00000E+00    0.20369E+05
63,65c63,65
<  V1.V2 =  -0.44703E-07
<  V2.V3 =  -0.11921E-06
<  V3.V1 =   0.00000E+00
---
>  V1.V2 =  -0.14901E-07
>  V2.V3 =  -0.89407E-07
>  V3.V1 =   0.14901E-06
88,89c88,89
<  theta        65.09956      124.61702       45.00004
<  phi         -87.65820      -16.34700       30.00003
---
>  theta        65.09957      124.61701       45.00003
>  phi         -87.65820      -16.34699       30.00002
91c91
<  Lambda    0.20369E+05    0.55875E+04   -0.63610E-03
---
>  Lambda    0.20369E+05    0.55875E+04   -0.23651E-03
----------------------------------------
rocot_check.out :
----------------------------------------
rocot_example.out :
15,16c15,16
< Sun in GEI:      0.744703     -0.612328     -0.265465
< Sun in GEO:      0.315648      0.910985     -0.265465
---
> Sun in GEI:      0.744702     -0.612329     -0.265465
> Sun in GEO:      0.315649      0.910985     -0.265465
----------------------------------------
rocot_info.out :
----------------------------------------
test_day_in_week.out :
----------------------------------------
test_dir_sun.out :
26c26
< Sun  in GEI :     0.136     0.909     0.394     1.000    66.790    81.520
---
> Sun  in GEI :     0.136     0.909     0.394     1.000    66.790    81.521
----------------------------------------
test_format.out :
----------------------------------------
test_week.out :
----------------------------------------
vdh_to_gse.out :
```

Table 9.4: Result of compare_out_GNU_Linux_x86_64_fort77_gfortran.sh

Annexe: compatibility with previous versions

Rocotlib exist since version 1.0 in november 1992. Until the 2.2 version of december 2011, the subroutines names was short and low speaking.

From the version 3, all the subroutine has been renamed with a long and more descriptive name. The correspondence with old names and new names is given in table 9.5. The V3.0 version contains a significant number of new subroutines not given here.

	tcarsph : t_car_to_sph
	tdmegeo : t_dm_to_geo
cangrat : cp_angle_and_ratio	tgeigeo : t_gei_to_geo
cdatdoy : cv_doty_to_date	tgeigse : t_gei_to_gse
cdatj00 : cv_jul2000_to_date	tgeigsm : t_gei_to_gsm
cdatj50 : cv_jul1950_to_date	tgeigsq : t_gei_to_gseq
cdatwee : cv_weekn_to_date	tgeimag : t_gei_to_mag
cdipdir : cp_geo_dipole_dir	tgeisma : t_gei_to_sm
cdoweek : cv_date_to_dotw	tgeodme : t_geo_to_dm
cdoyear : cv_date_to_doty	tgeogei : t_geo_to_gei
cfrdayn : cp_fr_day_name	tgeogse : t_geo_to_gse
cfrmonn : cp_fr_month_name	tgeogsm : t_geo_to_gsm
chouday : cv_hms_to_dech	tgeogsq : t_geo_to_gseq
cjd1950 : cv_date_to_jul1950	tgeomag : t_geo_to_mag
cjd2000 : cv_date_to_jul2000	tgeosma : t_geo_to_sm
cmilday : cv_dhms_to_msotd	tgeovdh : t_geo_to_vdh
cnbdmon : cp_nbday_in_month	tgsegei : t_gse_to_gei
coleapy : cp_leap_year	tgsegeo : t_gse_to_geo
csundir : cp_gei_sun_dir	tgsegsm : t_gse_to_gsm
csunset : cp_sunrise_sunset	tgsegsq : t_gse_to_gseq
ctimhou : cv_dech_to_hms	tgsemfa : t_gse_to_mfa
ctimmil : cv_msotd_to_hmsms	tgsesr2 : t_gse_to_sr2
ctimpa2 : cp_time_param2	tgsetpn : t_gse_to_tpn
ctimpa3 : cp_time_param3	tgsmgei : t_gsm_to_gei
ctimpar : cp_time_param	tgsmgeo : t_gsm_to_geo
ctpnsys : cp_tpn_param	tgsmgse : t_gsm_to_gse
cusdayn : cp_en_day_name	tgsmgsq : t_gsm_to_gseq
cusmonn : cp_en_month_name	tgsmmag : t_gsm_to_mag
cweedoy : cv_date_to_weekn	tgsmsma : t_gsm_to_sm
	tgsmtpn : t_gsm_to_tpn
gdipdir : g_gei_geo_dipole_dir	tgsqgei : t_gseq_to_gei
gdiptan : g_gsm_dipole_tilt_angle	tgsqgeo : t_gseq_to_geo
gecldir : g_gei_geo_ecliptic_dir	tgsqgse : t_gseq_to_gse
gsrodir : g_gei_geo_sun_rot	tgsqgsm : t_gseq_to_gsm
gsundir : g_gei_geo_sun_dir	tmaggei : t_mag_to_gei
gsunpar : g_gei_sun_param	tmaggeo : t_mag_to_geo
gvernum : g_rocot_version_number	tmaggsm : t_mag_to_gsm
	tmagsma : t_mag_to_sm
plibinf : print_rocot_info	tsmagei : t_sm_to_gei
	tsmageo : t_sm_to_geo
recoor : r_coordinate_values	tsmagsm : t_sm_to_gsm
recsys : r_coordinate_system	tsmamag : t_sm_to_mag
redate : r_date	tsphcar : t_sph_to_car
retime : r_time	tsr2gse : t_sr2_to_gse
	tsr2mfa : t_sr2_to_mfa
	tsr2sre : t_sr2_to_sr
	tsresr2 : t_sr_to_sr2
	tvdhgeo : t_vdh_to_geo

Table 9.5: Correspondence between names of old Rocotlib versions with the one of version 3.0.

List of Figures

List of Tables

Acknowledgments

The author is deeply grateful to Ronan Modolo who made a first Latex translation of the formulas initially written in Word in the 1.7 version of december 2002.

Bibliography

[1] Geophysical coordinate transformations, C.T. Russell, cosmic electrodynamics, v.2, 184-196, 1971.

[2] CLUSTER Software Tools, Part 1 : Coordinate Transformation Library, Version 1.1, by Patrick Robert, CNRS-CNET/CRPE, Document de travail DT/CRPE/1231, Juillet 1993.

[3] CLUSTER DATA PROCESSING, Transformation of a STAFF waveform into a Magnetic Field Aligned coordinate system, by Patrick Robert and C. de Villedary, Rapport interne CNRS-UVSQ/CETP n_o RI-CETP/6/2000, Octobre 2000.

[4] ROCOTLIB: a Coordinate Transformation Library for Solar-Terrestrial studies, Version 1.7, by Patrick Robert, Rapport interne CNRS-UVSQ/CETP n_o RI-CETP/02/2003, January 2003.

[4] Livraison des donnes S300/UBF des satellites GEOS au CDPP, comparaison avec le magnétomètre, Patrick Robert, Rapport interne pour le CDPP, ScientiDev, Mai 2019.

The results of sunset and sunrise computation done in the library have been checked with the one given by the service of the "Institut de Mécanique céleste et de calcul des éphémérides (IMCCE)" an particularly the web page https://promenade.imcce.fr/fr/pages3/370.html

NOTES

Write here your personal notes.