issn 0065-3713

INSTITUT D'AERONOMIE SPATIALE DE BELGIQUE

3 - Avenue Circulaire
B - 1180 BRUXELLES

-1010B

JL → KB

AERONOMICA ACTA

A – N° 404 – 2000

Trajectoires et altitudes minimales de particules chargées dans le champ géomagnétique

par

J. Vandenberghe

J. Lemaire

BELGISCH INSTITUUT VOOR RUIMTE-AERONOMIE

3 - Ringlaan
B - 1180 BRUSSEL

Les "Aeronomica Acta - A" constituent des documents destinés aux membres de l'Institut d'Aéronomie Spatiale de Belgique et aux services travaillant en liaison avec l'Institut. Ce sont, en général, des publications avancées.

De "Aeronomica Acta - A" zijn dokumenten bestemd voor de leden van het Belgisch Instituut voor Ruimte-Aëronomie en de diensten welke met het Instituut betrekkingen onderhouden. Meestal zijn het ver gevorderde studies en dienen zij als publikaties beschouwd worden.

The "Aeronomica Acta - A" are documents intended for the members of the Belgian Space Aeronomy Institute and for the Institutions connected with the Institute. Generally, they correspond to preprints and must be considered as a publication.

Die "Aeronomica Acta - A" sind Dokumente, die für dir Mitglieder des Belgischen Institutes für Raumaeronomie und für die mit dem Institut in Verbindung stehenden Dienste bestimmd sind. Sie sind im allgemeinen vorgeschrittene Veröffentlichungen.

Projet de recherche LIULIN (IN/BW/004) sous
les auspices du SSTC

Trajectoires et altitudes minimales de particules chargées dans le champ géomagnétique

J. Vandenberghe

J. Lemaire

Institut d'Aéronomie

Spatiale de Belgique

Mars 1998

Foreword

This work has been carried out between April 1997 and March 1998, in the framework of a joint project between the Belgian Institute of Space Aeronomy (BISA) and the Solar Terrestrial Influence Laboratory of Bulgarian Academy of Sciences (STIL-BAS).

It is a first step towards the study of the effects of small scale inhomogeneities of the geomagnetic field on the altitude of mirror points, and of the precipitation in earth atmosphere of charged particles trapped in Van Allen's belts.

The authors of this work are grateful to the belgian Minister of scientific policy, for the assignment of this project to the BISA and for the means made available to the LIULIN project.

We are also grateful to the director of the BISA for his support, Mrs M. Desmeth and Mr J.Bernard of the SSTC, Mr D. Heynderickx, Mr. M. Kruglanski, Mrs V. Pierrard, Mr L. Fedullo and Mr B. Quaghebeur of the BISA for their help or advice in the realization of this work.

Avant-propos

Ce travail a été effectué entre le premier avril 1997 et le 30 mars 1998 dans le cadre d'un projet de collaboration belgo-bulgare, entre l'Institut d'Aéronomie Spatiale de Belgique (IASB) et le Solar Terrestrial Influence Laboratory de l'Académie des Sciences de Bulgarie (STIL-BAS).

Il constitue une première étape dans l'étude des effets des inhomogénéités de faible échelle du champ géomagnétique sur l'altitude des points miroirs et de la précipitation des particules piégées dans les zones de radiation de Van Allen dans l'atmosphère terrestre.

Les auteurs de ce travail remercient le Ministre belge de la politique scientifique, pour l'attribution de ce projet à l'IASB et pour les moyens qui ont été mis à leur disposition dans le cadre du projet LIULIN.

Nous souhaitons aussi remercier le directeur de l'IASB pour l'appui qu'il nous a accordé, ainsi que Mme M. Desmeth et Mr J. Bernard du SSTC, Mr D. Heynderickx, Mr M. Kruglanski, Mme V. Pierrard, Mr L. Fedullo et Mr B. Quaghebeur de l'IASB pour leur aide ou conseil dans la réalisation de cette étude.

Voorwoord

Dit werk werd uigevoerd tussen 1 april 1997 en maart 1998, in het kader van een samenwerking tussen het Belgisch Instituut voor Ruimte-Aeronomie (BIRA) en het "Solar Terrestrial Influence Laboratory" van de Bulgaarse Academie voor Wetenschappen (STIL-BAS).

Het is een eerste stap in de studie van de effecten van kleine schaal inhomogeniteiten van het geomagnetisch veld op de hoogte van de spiegelpunten en van de precipitatie van geladen deeltjes, gevangen in Van Allen gordels.

De auteurs van dit werk wensen hun dank te betuigen aan de Minister voor Wetenschapsbeleid, voor de toekenning van dit project aan het BIRA en de middelen ter beschikking gesteld van het LIULIN project.

Wij zijn ook dankbaar: de Directeur van het BIRA voor zijn steun, Mevr. M. Desmeth en Dhr. J.Bernard van DWTC, de HH. D.Heynderickx en M. Kruglanski, Mevr. V. Pierrard en de HH. L. Fedullo en B. Quaghebeur van het BIRA voor hun hulp en raad bij de realisatie van deze studie.

Vorwort

Diese Arbeit wurde zwischen dem 1. April 1997 und dem 30. März 1998 im Rahmen eines gemeinschaftlichen belgisch-bulgarischen Projektes des Institut d'Aéronomie Spatiale de Belgique (IASB) und des Solar Terrestrial Influence Laboratory der Akademie der Wissenschaften Bulgariens (STIL-BAS) durchgeführt.

Sie ist ein erster Schritt in der Untersuchung der Auswirkungen der Schwachen Ungleichartigkeiten des erdmagnetischen Feldes auf die Höhe der Umkehrpunkte und des Niederschlages von Teilchen, die in den Van-Allen-Strahlungszonen der Erdatmosphäre eingefangen sind.

Die Autoren der vorliegenden Studie möchten dem Minister für Wissenschaftspolitik für die Zuteilung des Projektes an das IASB und für die Mittel, die ihnen im Rahmen des LIULIN-Projektes zur Verfügung gestellt wurden, an dieser Stelle herzlich danken.

Unser Dank gilt ebenfalls dem Direktor des IASB für die Unterstützung, die er uns hat angedeihen lassen, sowie Frau M. Desmeth und Herrn J.Bernard vom SSTC, Herrn D.Heynderickx, Herrn M.Kruglanski, Frau V.Pierrard, Herrn L.Fedullo und Herrn B.Quaghebeur vom IASB für ihre Hilfe und ihre Ratschläge bei der Verwirklichung dieser Studie.

Abstract

The 6-D Ordinary Differential Equation (ODE) of the motion of a charged particle in the geomagnetic field can mainly be solved by two numerical methods: the Cash-Karp method, namely a Runge-Kutta method of order five, and the Bulirsch-Stoer method, namely a combination of the so-called *modified midpoint method* and of the polynomial extrapolation. Both are based on the choice of a relative error factor $\epsilon = 10^{-n}$. But while Cash-Karp works with relatively small computation steps, Bulirsch-Stoer works with relatively big ones. Though more interesting for the computation time, the Bulirsch-Stoer method could be less efficient for the plotting of particle trajectories. However, Bulirsch-Stoer with $\epsilon = 10^{-15}$ seems to be the best choice for our purpose, because at such level of relative error the plotting isn't harmed by the relatively big steps (and the computation time of Bulirsch-Stoer is still more economic than that for Cash-Karp).

Using the computed cartesian coordinates of the points of the trajectories we interpolate to find the successive mirror points of the charged particle, defined as the points of the trajectories corresponding to minimum altitudes. An alternative definition of the mirror points is the guiding centers of the particle corresponding to minimum altitudes. There is a third definition worth to be used: the points where the scalar product $velocity * geomagnetic field$ becomes equal to zero. This latter mirror point has not been implemented in our Fortran code, but could be in the future.

In turn, the set of successive mirror points allows the computation of the minimum altitude of a magnetic drift shell. It is just a matter of sufficient length of integration time of the ODE.

The main objective of this work is the verification of the constancy of the magnetic moment which is the first adiabatic invariant of the motion of the particle in the geomagnetic field. Its invariance has been verified for protons with energies lower than 100 MeV.

Résumé

L'équation différentielle ordinaire à six dimensions du mouvement d'une particule chargée dans le champ géomagnétique peut être intégrée par deux grandes méthodes numériques: la méthode de Cash-Karp, un Runge-Kutta d'ordre 5 avec correction du pas, et la méthode de Bulirsch-Stoer, une méthode combinant la méthode dites *Modified Midpoint Method* avec l'extrapolation polynômiale. Ces deux méthodes sont basées sur le choix d'un facteur d'erreur relative $\epsilon = 10^{-n}$. Mais elles diffèrent essentiellement en ceci que celle de Cash-Karp travaille avec des pas de calcul relativement petits et celle de Bulirsch-Stoer avec des pas de calcul relativement grands. Ainsi, si la méthode de Bulirsch-Stoer est plus intéressante que celle de Cash-Karp du point de vue du temps de calcul, elle

pourrait s'avérer moins intéressante du point de vue du tracé des trajectoires de la particule chargée. Toutefois le choix de Bulirsch-Stoer avec $\varepsilon = 10^{-15}$ semble optimum, car à un tel niveau d'exigence sur l'erreur relative le tracé des trajectoires ne souffre pas de l'importance relative des pas de calcul (et le temps de calcul reste intéressant).

Les points calculés de ces trajectoires permettent d'interpoler leurs points miroirs, compris comme points d'altitude minimum ou comme *centres guides* d'altitude minimum. Il y a une troisième définition: les points où s'annule le produit scalaire de la *vitesse* et du *champ magnétique* . Cette définition n'a pas été implémentée dans notre code Fortran, mais pourrait l'être dans le futur.

Ces points miroirs permettent à leur tour d'évaluer l'altitude minimum d'une coquille magnétique. Il suffit de considérer un intervalle temporel suffisamment long pour l'intégration de l'équation du mouvement de la particule.

Mais l'objectif le plus intéressant de la présente note est l'étude du moment magnétique en tant que premier invariant adiabatique du mouvement d'une particule chargée dans le champ géomagnétique. Cette invariance du moment magnétique est vérifiée pour des protons d'énergies inférieures à une centaine de MeV.

Samenvatting

De 6-D gewone differentiaalvergelijking van de beweging van een geladen deeltje in het geomagnetisch veld kan hoofdzakelijk opgelost worden door twee numerieke methoden: de Cash-Karp methode, een vijfde orde Runge-Kutta methode, en de Bulirsch-Stoer methode, een combinatie van de zogenaamde *"Modified Midpoint Method"* en de polynomiale extrapolatie. Beide methoden zijn gebaseerd op de keuze van een relatieve foutenfactor $\varepsilon = 10^{-n}$. Het essentieel verschil tussen beide is dat de Cash-Karp methode gebruik maakt van relatief kleine stappen bij de berekening, terwijl de Bulirsch-Stoer methode met relatieve grotere stappen werkt. Hoewel interessanter qua berekeningstijd, zou de Bulirsch-Stoer methode soms minder efficiënt kunnen zijn voor het uittekenen van de banen der deeltjes. Nochtans, lijkt een Bulirsch-Stoer methode met $\varepsilon = 10^{-15}$ de beste keuze voor onze doeleinden, omdat bij een dergelijke kleine relatieve fout het tekenen der banen niet lijdt onder de relatief grotere berekeningstappen en de rekentijd interessant blijft.

Uit de berekende cartesiaanse coördinaten van de verschillende punten van de banen kunnen door interpolatie de zogenaamde *"spiegelpunten"* gevonden worden. Dit zijn punten van minimum hoogte of *"richtcentra"* van minimum hoogte. Een derde definitie is: punten waar het scalair produkt van *snelheid en het magnetisch veld* nul wordt. Hoewel de implementatie van deze definitie niet is gebeurd in onze Fortran code, kan dit wel in de toekomst.

De kennis der spiegelpunten laat toe de minimum hoogte van de magnetische schelp te bepalen. Het volstaat een voldoende lang tijdsinterval te beschouwen

voor de integratie van de bewegingsvergelijking van het deeltje.

Het belangrijkste doel van dit werk is echter de studie van het magnetisch moment als eerste adiabatische invariant van beweging van een geladen deeltje in het geomagnetisch veld. Deze invariant werd nagegaan voor protonen met een energie kleiner dan 100 MeV.

Zusammenfassung

Die einfache Differentialgleichung mit sechs Dimensionen der Bewegung eines geladenen Teilchens im erdmagnetischen Feld kann durch zwei grosse numerische Verfahren integriert werden: die Cash-Karp-Methode, ein Runge-Kutta des fünften Grades mit Schrittkorrektur, und der Bulirsch-Stoer-Methode, einer Kombination der sogenannten *Modified Midpoint Method* mit polynomischer Extrapolation. Diese beiden Methoden basieren auf der Wahl eines relativen Fehlerfaktors $\varepsilon = 10^{-n}$. Sie unterscheiden sich dahingehend, dass Cash-Karp mit verhältnismässig kleinen Rechenschritten und Burlisch-Stoer mit verhältnismässig grossen Rechenschritten vorgeht. Für das Problem, mit dem wir uns befassen, scheint die Wahl von Bulirsch-Stoer mit $\varepsilon = 10^{-15}$ optimal zu sein. Bei derartigen Anforderungen bezüglich des relativen Fehlers wird der Streckenverlauf nicht durch die verhältnismässige Bedeutung der Rechenschritte im Falle des Burlisch-Stoer-Verfahrens beeinträchtigt.

Die berechneten Punkte dieser Verläufe ermöglichen die Interpolation ihrer Umkehrpunkte, die als Mindesthöhenpunkte oder als *richtungweisende Mittelpunkte* für die Mindesthöhe verstanden werden (dritte Definition: Punkte, in denen das innere Produkt (Skalarprodukt) von Geschwindigkeit und Magnetfeld sich aufhebt).

Diese Umkehrpunkte ermöglichen ihrerseits die Einschätzung der Mindesthöhe einer magnetischen Schale. Für die Integration der Gleichung der Bewegung des Teilchens reicht es aus, ein ausreichend langes Zeitintervall in Betracht zu ziehen.

Das interessanteste Ziel der vorliegenden Arbeit ist das Studium des magnetischen Momentes als erste adiabatische Invariante der Bewegung eines geladenen Teilchens im erdmagnetischen Feld. Diese Invarianz des magnetischen Momentes wird auf Protonen mit einer Energie von unter einem Hundertstel MeV untersucht.

Introduction

L'étude comparative des méthodes d'intégration numérique pour le calcul des trajectoires d'une particule chargée dans un champ magnétique uniforme et dans un dipôle (problème de Störmer) a été décrite dans la note intitulée *Méthodes d'intégration numérique des trajectoires de particules chargées dans un champ magnétique constant et dipolaire* de J.Vandenberghe et J.Lemaire.

La présente note technique, qui fait suite à ce travail d'intérêt général, constitue une application de ces méthodes au calcul du mouvement d'une particule chargée dans un champ magnétique plus complexe comme le champ géomagnétique qui est représenté par une expansion en harmonique sphérique contenant 120 termes d'ordre multipolaire. Nous décrirons les entrées (inputs) et sorties (outputs) des différents programmes que nous avons réalisés pour:

- calculer les trajectoires des particules chargées de masse, charge et énergie données, pour différentes conditions initiales et pour différents modèles de champ géomagnétique.
- dessiner les trajectoires de ces particules dans un espace à trois dimensions avec une projection du globe terrestre en arrière-fond.
- calculer les coordonnées géographiques des points miroirs successifs dans l'hémisphère Nord et dans l'hémisphère Sud. L'altitude minimale des points miroirs est définie avec une précision de 0.5 km. Deux définitions sont utilisées pour définir les points miroirs:
  - l'altitude minimale de la trajectoire de la particule.
  - l'altitude minimale du centre-guide de la particule.

  Dans l'étude des zones de radiation de Van Allen le point-miroir est définit par l'altitude du point où la projection de la vitesse de la particule sur la direction du champ magnétique s'annule c'est-à-dire où $(\mathbf{v}|\mathbf{B}) = 0$. Les coordonnées de ce type de point miroir n'ont pas été déterminées dans ce travail, leur calcul fera l'objet d'une note ultérieure.
- calculer l'altitude minimum d'une coquille magnétique, en évaluant les points miroirs.
- évaluer le moment magnétique en tant que premier invariant adiabatique, et ce grâce aux points miroirs.

Programme UNITRAJ

Le programme UNITRAJ, rédigé en VMS Fortran, traite numériquement les équations de mouvement (relativiste) d'une particule chargée placée dans le champ géomagnétique (en l'absence de toute autre force, ce qui implique la constance de la norme de la vitesse de cette particule, et donc la constance de sa masse), et ce avec l'aide de la librairie UNILIB développée à l'IASB dans le cadre du projet TREND-3.

Il faut fournir, sous forme de namelist (dont un exemple est donné en annexe), les inputs suivants:

- la définition de la particule (charge, masse, énergie cinétique);
- la position initiale de la particule (en coordonnées géodétiques) et la direction initiale de sa vitesse (en coordonnées sphériques), cette direction ne devant pas être nécéssairement normée (cosinus directeurs facultatifs);
- le temps final d'intégration (le temps initial étant nul);
- le numéro d'identité du modèle du champ géomagnétique (le choix entre les modèles disponibles dans la librairie UNILIB: IGRF, Jensen and Cain, GSFC-1960, composante dipolaire extraite d'IGRF) et l'année de son application;
- le choix de la méthode numérique (Cash-Karp ou Bulirsch-Stoer) et de la précision (c'est-à-dire du facteur d'erreur relative ε de ces méthodes); afin d'éviter des fichiers de sortie trop importants dans le cas de la méthode de Cash-Karp, on peut également choisir de ne stocker qu'un point sur un certain nombre (*nokwrite*) de points.

Les fichiers de sortie (dont les noms commencent par CK ou BS, c'est-à-dire l'identité de la méthode numérique choisie: CK pour Cash-Karp/BS pour Bulirsch-Stoer) contiennent:

- les temps, positions géodétiques et composantes sphériques des vitesses de la particule (fichiers CKTIME.DAT/BSTIME.DAT et CKPOINTS.DAT /BSPOINTS.DAT);
- positions de la particule et de son centre guide, en coordonnées cartésiennes, le rayon terrestre étant l'unité de longueur (CKPOS.DAT/BSPOS.DAT et CKGC.DAT/BSGC.DAT);
- les erreurs relatives sur la norme (constante) de la vitesse (fichiers CKERRV.DAT/BSERRV.DAT);
- les valeurs du moment magnétique (adiabatiquement invariant) $\mu = W_\perp/B$ calculés de deux manières différentes: en évaluant l'intensité du champ

magnétique B en la position de la particule et en son centre guide (fichiers CKMU.DAT/BSMU.DAT); $W_\perp$ est l'énergie cinétique associée à la composante de la vitesse perpendiculaire à la direction du champ magnétique: $W_\perp = \frac{1}{2}mv_\perp^2$.

Représentation graphique des trajectoires dans l'espace 3D

Les fichiers créés par le programme UNITRAJ peuvent servir d'"input" à des programmes graphiques permettant de visualiser les trajectoires des particules dans l'espace de configuration. Considérons par exemple un proton de 1000 MeV aux conditions initiales suivantes:

- altitude géodétique = 1646 km;
- latitude géodétique = -44 deg;
- longitude = 228 deg;
- composante radiale de la direction de la vitesse = 0.08998;
- composante colatitudinale de la direction de la vitesse = -0.15032;
- composante longitudinale de la direction de la vitesse = 0.98453.

Choisissons le modèle de champ géomagnétique IGRF pour l'année 1995. Dans ce cas les valeurs des coordonnées B, L sont $L = 2.0733$ et $B = 0.2289$. Nous adoptons la méthode numérique de Cash-Karp, avec $\varepsilon = 10^{-10}$; celle-ci est *a priori* plus indiquée que celle de Bulirsch-Stoer pour tracer des trajectoires. La figure 1 représente, pour un intervalle temporel de 3 secondes, la trajectoire obtenue par les commandes IDL du fichier "batch" IGRF3D (cfr Annexes). Le fichier CKPOS.DAT a été utilisé en "input" pour le fichier IGRF3D en vue de faire ce graphique.

Reprenons les mêmes inputs, sauf l'énergie que nous diminuons d'un facteur 10. La figure 2 représente la trajectoire résultante, beaucoup plus lisible que la précédente (on distingue beaucoup plus nettement les mouvements de gyration, d'oscillation en latitude entre des points miroirs conjugués et de dérive en longitude).

La figure 2 a été obtenue avec le programme UNITRAJ en imposant *nokwrite*=1; il en résulte que le fichier CKPOS.DAT contient dans ce cas 70000 points/lignes. Le fait est que la qualité du trait représentant la trajectoire est pratiquement la même avec *nokwrite*=10, c'est-à-dire avec seulement 7000 points/lignes dans le fichier CKPOS.DAT. Les figures 3 et 4, correspondant respectivement à *nokwrite*=50 (1400 points) et *nokwrite*=100 (700 points), sont

de MIRP et MIRPGC, nous avons pu déterminer les altitudes minimum avec une précision de l'ordre du dixième de km. De là, nous avons pu établir que ces altitudes étaient obtenues avec une précision de 0.5 km par la méthode de Cash-Karp avec $\varepsilon \leq 10^{-7}$ et par la méthode de Bulirsch-Stoer avec $\varepsilon \leq 10^{-15}$.

Le tableau suivant donne les résultats de calcul obtenus à l'aide des programmes MIRP et MIRPGC pour un proton de 100 MeV, un intervalle de 6 secondes, les conditions initiales du paragraphe *Représentation graphique des trajectoires dans l'espace 3D* et la méthode de Bulirsch-Stoer avec $\varepsilon = 10^{-15}$.

	POINTS MIROIRS NORD		POINTS MIROIRS SUD	
	MIRP	MIRPGC	MIRP	MIRPGC
Temps [secs]	0.21006	0.20951	0.43401	0.43323
Altitude [km]	1411.37	1440.71	1409.13	1434.30
Temps [secs]	0.65838	0.65803	0.88236	0.88251
Altitude [km]	1395.31	1424.13	1454.84	1481.45
Temps [secs]	1.10669	1.10786	1.33348	1.33298
Altitude [km]	1379.39	1405.79	1497.67	1524.83
Temps [secs]	1.55776	1.55896	1.78454	1.78473
Altitude [km]	1357.07	1382.61	1537.58	1565.67
Temps [secs]	2.01156	2.01120	2.23838	2.23751
Altitude [km]	1341.11	1367.75	1578.46	1606.22
Temps [secs]	2.46536	2.46442	2.69221	2.69117
Altitude [km]	1337.84	1362.65	1614.58	1642.35
Temps [secs]	2.91909	2.91819	3.14600	3.14524
Altitude [km]	1354.15	1378.30	1645.67	1674.45
Temps [secs]	3.37273	3.37202	3.59973	3.59913
Altitude [km]	1394.32	1418.10	1673.62	1702.85
Temps [secs]	3.82630	3.82543	4.05340	4.05227
Altitude [km]	1456.78	1479.36	1700.56	1728.53
Temps [secs]	4.27713	4.27789	4.50425	4.50427
Altitude [km]	1529.69	1551.81	1715.48	1745.09
Temps [secs]	4.73067	4.72925	4.95503	4.95509
Altitude [km]	1603.27	1622.87	1726.11	1755.49
Temps [secs]	5.17875	5.17970	5.40580	5.40490
Altitude [km]	1660.70	1681.09	1727.28	1755.25
Temps [secs]	5.62960	5.62921	5.85377	5.85386
Altitude [km]	1696.94	1718.02	1706.68	1735.39

Dans ce tableau on donne les temps auxquels la particule (MIRP) ou son centre guide (MIRPGC) atteignent les altitudes minimales dans les hémisphères Nord et Sud.

Notons que les altitudes des centres guides sont toujours supérieures à celles des positions des particules au point le plus bas sur leur trajectoire. On peut vérifier que la différence d'altitude est égale au rayon de gyration multiplié par le sinus de l'angle entre la direction du champ magnétique et la direction zénithale.

Programmes ALTMIN et ALTMINGC: recherche de l'altitude minimum d'une coquille magnétique

Si nous appliquons UNITRAJ à un intervalle temporel suffisamment long, nous pourrons tracer une coquille magnétique (drift shell) complète de la particule chargée. Pour illustrer ceci, prenons un proton de 100 MeV dans le modèle de champ géomagnétique IGRF pour l'année 1960, avec les conditions initiales suivantes:

- altitude géodétique = 2561.6 km;
- latitude géodétique = -25.05 deg;
- longitude géodétique = 100.9 deg;
- composante radiale de la direction de la vitesse = 0.00201;
- composante colatitudinale de la direction de la vitesse = -0.35327;
- composante longitudinale de la direction de la vitesse = -0.24197.

La figure 5 représente la trajectoire de ce proton pour un intervalle temporel de 130 secondes obtenue avec la méthode de Bulirsch-Stoer ($\varepsilon = 10^{-15}$); la figure 6 représente cette même trajectoire pour 20 secondes; la figure 7 représente les lignes du centre guide pour 20 secondes.

La figure 8, obtenue avec la partie dipolaire du champ magnétique IGRF, représente la trajectoire pour 20 secondes; on voit, par comparaison avec la figure 6, que l'Anomalie Sud-Atlantique n'est pas présente.

En appliquant ensuite le programme MIRP (resp. MIRPGC), on obtient les fichiers de sortie NORTHMP.DAT et SOUTHMP.DAT (NORTHMPGC.DAT et SOUTHMPGC.DAT) contenant les points miroirs de l'hémisphère Nord et de l'hémisphère Sud d'altitude minimum. L'objet du programme ALTMIN (resp. ALTMINGC) est de rechercher le minimum des altitudes minimales contenues dans ces deux types de fichier. Plus précisément, ce programme recherche dans ces fichiers les trois altitudes minimales permettant l'interpolation parabolique de ce minimum.

Le tableau suivant donne les coordonnées géographiques (longitude, latitude et altitude) des points miroirs d'altitude minimum dans l'hémisphère Nord (N) et dans l'hémisphère Sud (S). Ces coordonnées sont calculées à l'aide du programme ALTMIN pour un intervalle de 20 secondes et pour différents modèles

du champ géomagnétique:

MODELE	Long N	Lat N	Alt min N	Long S	Lat S	Alt min S
DIPÔLE	339.9	31.63	1830.4 km	91.62	-27.84	1828.9 km
J. & CAIN	333.9	35.00	1468.4 km	368.60	-44.60	634.5 km
GSFC	330.3	34.21	1450.9 km	332.2	-43.10	662.8 km
IGRF(1960)	333.8	35.26	1450.4 km	319.9	-42.97	661.6 km

On constate que lorsqu'on utilise le dipôle magnétique centré l'altitude des points miroirs est quasiment la même dans les hémisphères Nord et Sud. Par contre si on utilise des modèles de champ géomagnétique plus réalistes pour lesquels le champ magnétique est décentré et non symétrique, le point miroir le plus proche de la surface de la Terre se trouve dans l'hémisphère Sud, et situé à une longitude voisine de celle de l'anomalie sud-atlantique, entre 319 et 369 degrés de longitude Est. Il est alors situé à une altitude beaucoup plus faible (634–663 km) que dans le cas du modèle dipolaire (1829–1830 km). D'autre part, lorsque l'on compare la longitude du point-miroir minimum obtenue avec les modèles IGRF pour les années 1960 et 1995 on constate un déplacement vers l'Est de 10.5 degrés en 35 ans, soit 3 degrés par décade. Par contre la latitude du point miroir minimum n'a changé que de 1.7 degrés en 35 ans. Pendant cette période l'altitude du point miroir dans l'hémisphère Sud a varié de 661.6 km à 307.56 km et donc de 354.04 km.

Le tableau suivant donne les altitudes minimales des centres guides calculées à l'aide du programme ALTMINGC par interpolation parabolique des altitudes minimales de "points miroirs centre guide" contenus dans le fichier input SOUTHMPGC.DAT. On constate que les altitudes de "points miroirs centre guide" sont plus élevées que celles des positions des particules elles-mêmes. On retrouve une différence d'altitude proportionnelle à la valeur du rayon de gyration des protons. Dans le cas d'ions plus lourds de même énergie cette différence serait plus grande. Par contre pour des électrons la différence entre ces deux types de points miroirs peut être négligée vis à vis de la précision de 0.5 km.

MODELE DE B	Alt min N	Alt min S
DIPÔLE	1868.3 km	1868.3 km
JENSEN AND CAIN	1503.5 km	662.2 km
GSFC	1486.1 km	691.6 km
IGRF(1960)	1485.7 km	690.1 km

Moment magnétique

Si nous intégrons par le programme UNITRAJ les équations de mouvement d'un proton de 100 MeV avec la méthode de Cash-Karp ($\epsilon = 10^{-10}$), avec les conditions initiales du paragraphe *Représentation graphique des trajectoires dans l'espace 3D* , le champ IGRF (1995) et un intervalle temporel de 0.4 secondes, nous obtenons une oscillation complète entre les points miroirs (figure 9).

La partie gauche de la figure 10 représente le moment magnétique $\mu = W_\perp / B$ (en unités SI multipliées par 10^7) de ce proton en fonction du temps (en unités de 1/10 secondes). Il s'agit du moment magnétique obtenu en évaluant l'intensité du champ magnétique à la position du proton pour la partie supérieure de la figure, et en évaluant cette intensité au centre guide pour la partie inférieure de la figure. Les oscillations autour d'une moyenne constante ($\mu_0 = 6.5 \times 10^{-7}$) sont dûes à la variation de l'intensité magnétique B au cours du mouvement de gyration autour de la ligne de champ magnétique. L'amplitude de cette oscillation est maximum lorsque la particule passe dans le plan équatorial où le rayon de gyration de la particule est maximum. Au voisinage des points miroirs la trajectoire de la particule tend vers une circonférence et le moment magnétique est pratiquement indépendant du temps. Ceci conduit à la conclusion qu'il est préférable d'évaluer le moment magnétique d'une particule en ses points miroirs pour vérifier la constance de celui-ci au cours de ses oscillations en latitude et de sa dérivée en longitude. Enfin, on observe une très légère atténuation de l'amplitude de l'oscillation lorsque l'on utilise la valeur de l'intensité du champ magnétique au centre guide, au lieu de celle calculée à la position de la particule elle-même.

La partie droite de la figure 10 correspond à toutes les données de la partie gauche, à l'exception de l'énergie qui vaut 25 MeV. L'énergie étant réduite à son quart, la vitesse diminue de moitié et les 0.4 secondes ne correspondent plus qu'à une demi-oscillation entre deux points miroirs. On voit aussi que le moment magnétique au voisinage des points miroirs est pratiquement réduit au quart de sa valeur. Enfin, l'amplitude de l'oscillation de ce moment est fortement réduite.

Programme FSTADDIP: premier invariant adiabatique pour le dipôle terrestre

Nous avons écrit le programme FSTADDIP qui intègre les équations de mouvement d'une particule chargée dans le champ géomagnétique réduit à un dipôle (pour une plus grande rapidité d'exécution, ce dipôle est évalué par une simple sous-routine et non par appel à la librairie UNILIB), avec pour objectif principal l'évaluation du moment magnétique en tant que premier invariant adiabatique du mouvement de la particule.

Autrement dit ce programme ne s'intéresse qu'aux points miroirs, au voisinage desquels le moment magnétique tend à se confondre avec sa moyenne in-

variante (cfr figure 10). Ces points miroirs sont évalués de la même manière que dans le programme MIRP, autrement dit par interpolation parabolique sur trois points obtenus par intégration numérique et correspondant à une portion de trajectoire voisine d'une altitude minimale.

Nous avons considéré un proton de 10 MeV avec les conditions initiales du paragraphe *Représentation graphique des trajectoires dans l'espace 3D* et avons exécuté le programme FSTADDIP pour un intervalle temporel de trois heures et avec la méthode de Cash-Karp (facteur $\epsilon = 10^{-12}$). Les résultats sont convaincants: le tableau suivant montre l'excellente stabilité de l'altitude et de la latitude des points miroirs, de l'intensité magnétique au point miroir, et de la valeur du moment magnétique associé après 10800 sec (trois heures) de temps orbital; il montre aussi qu'après trois heures il reste encore six bonnes décimales sur dix pour la vitesse de la particule normalisée à l'unité à l'instant initial, ce qui prouve que l'énergie cinétique de la particule est conservée avec une excellente approximation.

CASH-KARP	Time=0.78 sec	Time=10799.5 sec
Altitude [km]	1592.17777739084	1592.15715392821
Latitude (absolute value) [deg]	44.2007979	44.2008719
B [gauss]	0.242553559	0.242555638
velocity	1.0000000000924	1.000000887477
μ [SI]	$6.640596*10^{-8}$	$6.640620*10^{-8}$

Nous avons ensuite procédé à la même exécution de FSTADDIP mais avec la méthode de Bulirsch-Stoer (facteur $\epsilon = 10^{-15}$). Pour un "CPU time" de l'ordre de deux heures, contre une quinzaine d'heures pour Cash-Karp, nous avons obtenu des résultats encore très acceptables, comme en témoigne le tableau qui suit. D'ailleurs, la précision sur la vitesse est nettement supérieure avec Bulirsch-Stoer, et si les autres résultats sont meilleurs avec Cash-Karp ce n'est qu'en raison de l'interpolation parabolique qui est à la base de leur calcul (interpolation plus précise avec la méthode à pas plus fins, Cash-Karp). Le tableau rapporte aussi les résultats, toujours acceptables, de la même intégration pour un intervalle temporel de deux jours (48 heures).

BULIRSCH-STOER	Time=0.78 sec	Time=10799.5 sec	Time=172799.58 sec
Altitude [km]	1592.18209134207	1592.24022499	1592.145921
Latitude (abs. val.) [deg]	44.20086278	44.20066769	44.20114659
B [gauss]	0.2425533346	0.242547509	0.24255738
velocity	1.00000000000006	1.00000000055431	1.00000000931
μ [SI]	$6.6406027*10^{-8}$	$6.640605087*10^{-8}$	$6.6406177*10^{-8}$

Programme FSTADIABINV: premier invariant adiabatique pour le champ géomagnétique IGRF

Ce programme généralise le précédent en utilisant pour l'évaluation du champ géomagnétique non plus une formule analytique correspondant au dipôle mais une formule numérique correspondant à un modèle (IGRF) disponible dans la librairie UNILIB.

A titre d'illustration de ce programme, nous avons repris les conditions initiales du paragraphe *Représentation graphique des trajectoires dans l'espace 3D* pour des protons de 50 MeV et 100 MeV, pour le modèle IGRF de l'année 1995 et en appliquant la méthode de Bulirsch-Stoer avec $\varepsilon = 10^{-15}$. Pour le proton de 50 MeV on obtient encore une bonne invariance du moment magnétique aux points miroirs, puisque sur un intervalle temporel de 10000 secondes ce moment oscille entre $3.14 * 10^{-7}$ et $3.16 * 10^{-7}$. Pour le proton de 100 MeV par contre l'invariance du moment magnétique est déjà plus problématique, puisque ce moment oscille sur le même intervalle temporel entre $5.9 * 10^{-7}$ et $7.5 * 10^{-7}$.

Conclusions

Ce travail nous a permis de développer des outils software permettant de calculer avec une précision de 0.5 km les altitudes des points miroirs de particules chargées pour différents modèles de champs géomagnétiques et de tracer leurs trajectoires dans l'espace 3D. L'invariance du moment magnétique de ces particules aux points miroirs a été éprouvée, et vérifiée pour des énergies inférieures à une centaine de MeV. Il reste d'une part à étudier les deux autres invariants du mouvement (l'invariant longitudinal lié à l'oscillation entre deux points miroirs, et l'invariant de flux lié à la dérive longitudinale de la particule); d'autre part à considérer la troisième définition des points miroirs (points d'annulation du produit scalaire vitesse-champ magnétique).

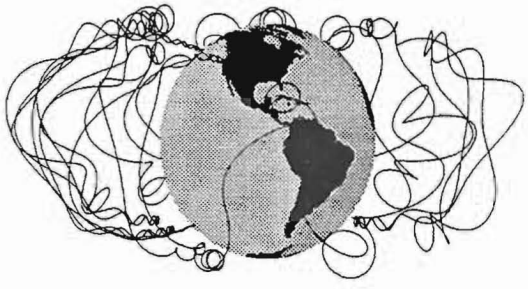

Figure 1: Trajectoire d'un proton de 1000 MeV placé dans le champ géomagnétique modélisé par IGRF(1995), pour un intervalle temporel de 3 secondes.

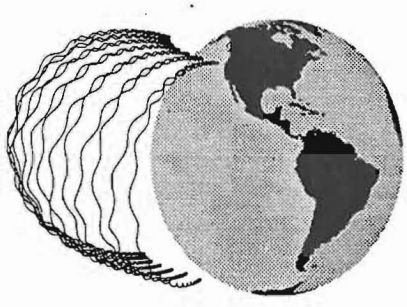

Figure 2: Trajectoire d'un proton de 100 MeV placé dans le champ géomagnétique modélisé par IGRF(1995), pour un intervalle temporel de 3 secondes.

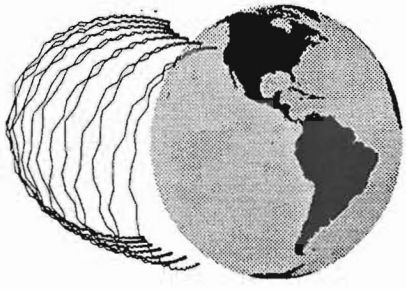

Figure 3: Trajectoire de la figure 2 redessinée avec 50 fois moins de points (paramètre d'UNITRAJ *nokurite* =50).

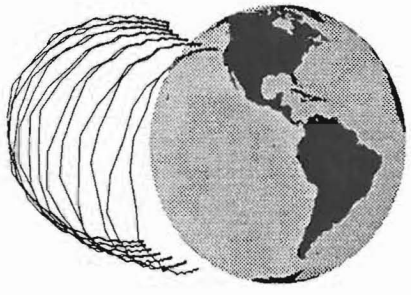

Figure 4: Trajectoire de la figure 2 redessinée avec 100 fois moins de points (*nokwrite* =100).

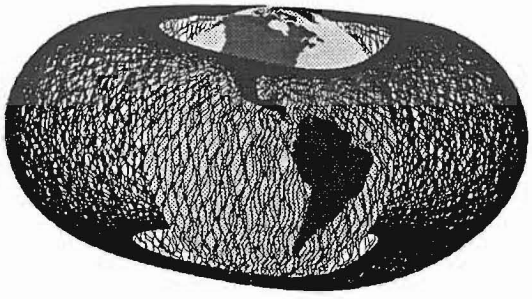

Figure 5: Trajectoire d'un proton de 100 MeV dans le champ IGRF (1960), pour un intervalle de 130 secondes.

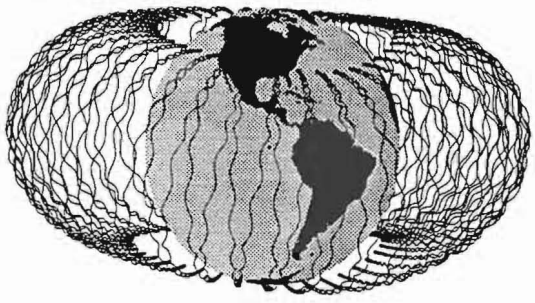

Figure 6: Trajectoire d'un proton de 100 MeV dans le champ IGRF(1960), pour 20 secondes.

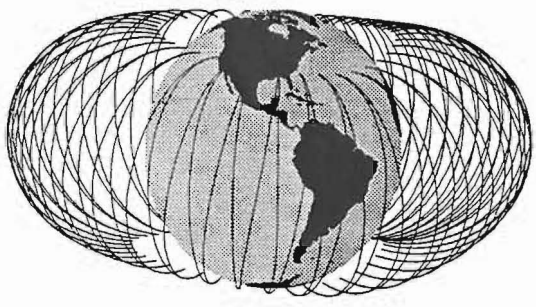

Figure 7: Lignes du centre guide d'un proton de 100 MeV dans le champ IGRF(1960), pour 20 secondes.

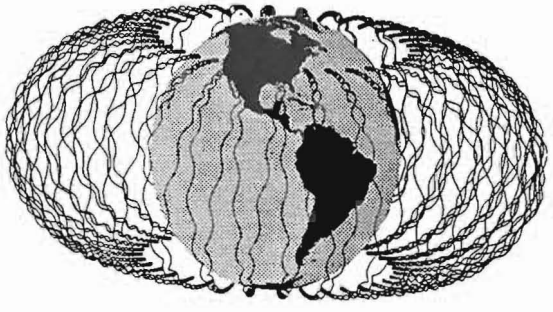

Figure 8: Trajectoire d'un proton de 100 MeV dans le dipôle géomagnétique (1960), pour 20 secondes.

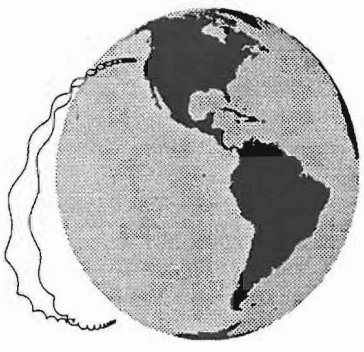

Figure 9: Trajectoire d'un proton de 100 MeV placé dans le champ géomagnétique IGRF-1995, pour un intervalle de 0.4 secondes (*nokwrite* = 1).

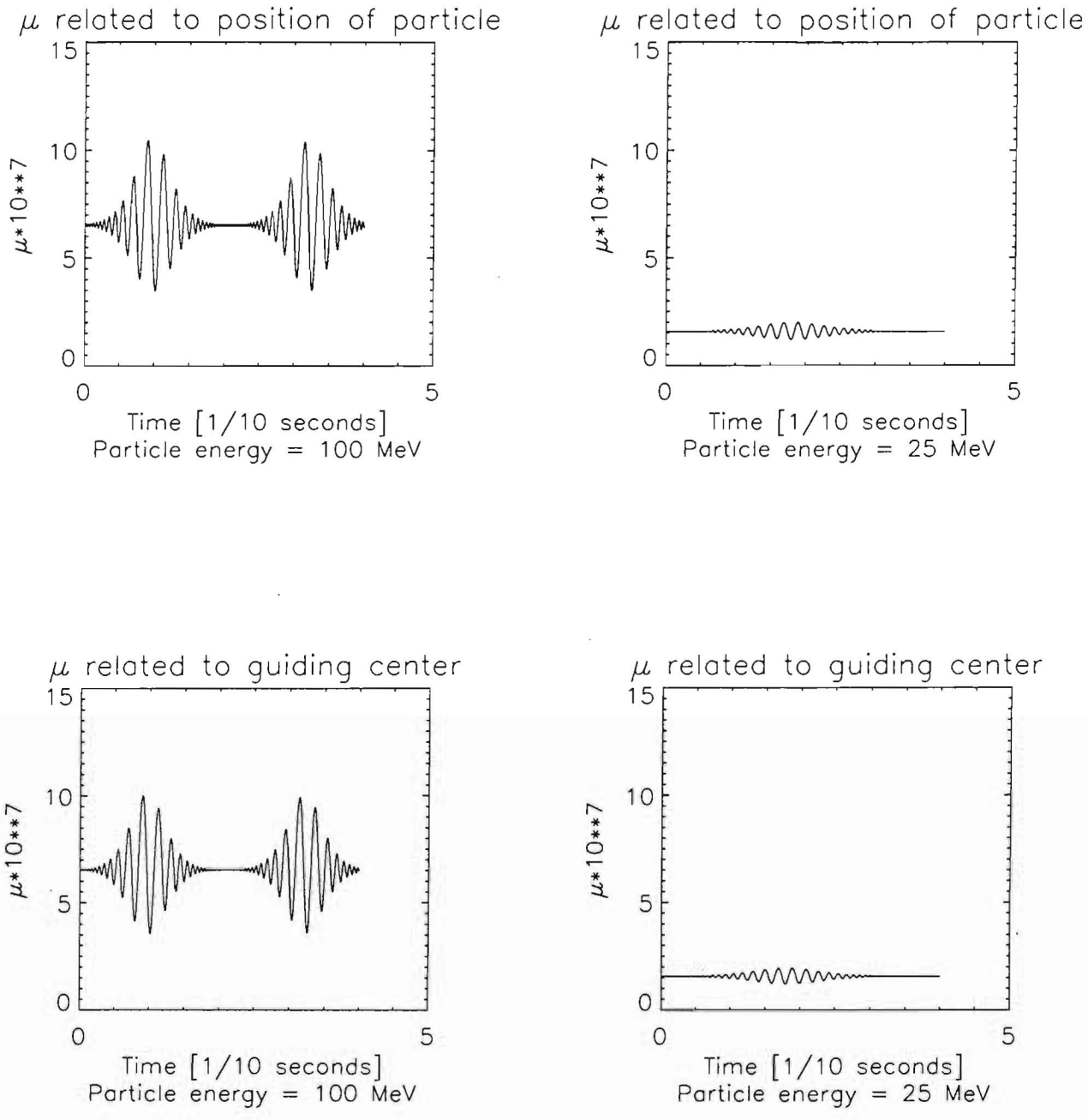

Figure 10: Moments magnétiques d'un proton de 100 MeV et d'un proton de 25 MeV placés dans le champ IGRF, pour un intervalle de 0.4 secondes (qui correspond respectivement à une oscillation complète et à une demi-oscillation entre deux points miroirs).

Annexes

- Un exemple de namelist du programme UNITRAJ
- Listings des programmes principaux (UNITRAJ, MIRP, MIRPGC, ALTMIN, ALTMINGC, FSTADDIP, FSTADIABINV)
- Listings des sous-routines (par ordre alphabétique)
- Le fichier "batch" idl IGRF3D.PRO de représentation en 3D des trajectoires et ses sous-routines

1 Exemple de namelist de UNITRAJ

```
$particle
     a = 1,
     z = 1,
     en = 100.D0,
$end
$initcond
     geodalt = 2561.6d0,
     geodlat = -25.05d0,
     geodlong = 100.9d0,
     velrho = 0.00201d0,
     veltheta = -0.35327d0,
     velphi = -0.24197d0,
$end
$magnfield
     kint = 0,
     year = 1995.d0,
$end
$meth
     imeth = 2,
$end
$integr
     fintim= 20.d0,
     hmin= 0.D0,
     eps=1.D-15,
$end
$outfile
     nokwrite=1,
$end
```

2 Listings des programmes principaux

```
      PROGRAM UNITRAJ
C
C
C     ************************************************************************
C     ************************************************************************
C     *                                                                      *
C     * Runge-Kutta (Cash-Karp) or Bulirsch-Stoer steps for dy/dt = f(y),    *
C     * where y is the 6-dim vector                                          *
C     *                              ------------                            *
C     *                             | Position |                             *
C     *                             |----------|                             *
C     *                             | Velocity |                             *
C     *                              ------------                            *
C     * of a charged particle and f the 6-dim vector                         *
C     *                  ------------------------------------                *
C     *                 |             Velocity               |               *
C     *                 |------------------------------------|               *
C     *                 |          Massic force              |               *
C     *                 |      exerted on the particle       |               *
C     *                 |     in the geomagnetic field       |               *
C     *                 |  described by one of the models    |               *
C     *                 |  available in the UNILIB library   |               *
C     *                  ------------------------------------                *
C     *                                                                      *
C     *                                                                      *
C     * The Runge-Kutta-Cash-Karp option is aimed at plotting trajectories   *
C     * of the particle (for the chosen interval of time [0,FINTIM]) with    *
C     * high precision.                                                      *
C     *                                                                      *
C     * Bulirsch-Stoer is aimed at reaching a precise point (corresponding   *
C     * to the chosen final time FINTIM) above all, but may also be used to  *
C     * plot trajectories: it is faster than Cash-Karp, and the precision of *
C     * the plotting is high enough if the relative error factor EPS is low  *
C     * enough (adviced value: 1.D-15).                                      *
C     *                                                                      *
C     ************************************************************************
C     ************************************************************************
C
C
C
      EXTERNAL odeint2
C     (and UNILIB routines: UT990, UM510, UM536, UT541, UT542)
C
      INCLUDE 'structure.h'
C
C
      INTEGER*4 kunit, kinit, ifail, nutime, nupoints,
     >          nutraj, nugc nuerrv, numagnmom
C
      REAL*8    p, v, gamma, ystart(6), t1, t2, h1
C
      CHARACTER*32 lbint
C
      RECORD /zgeo/ mkgde, mkgeo, mpos
      RECORD /zxyz/ mkxyz, vcart, bcart
      RECORD /zvec/ vspher, mb
C
C
      COMMON/particule/ q, massp, massr
        REAL*8  q, massp, massr
      COMMON/methode/  imeth
        INTEGER*4 imeth
      COMMON/lightvel/ c
      COMMON/units/ lu, tu, vu
        REAL*8  lu, tu, vu
      COMMON/mfmodel/ kint
        INTEGER*4 kint
C
      COMMON/UC140/ mint, mext, msun
        RECORD/zimf/mint
```

```
71            RECORD/zsun/msun
72            RECORD/zemf/mext
73  C
74          COMMON /UC160/ pi, deg, re, gmagmo, eclipt, geoid, uma
75            REAL*8  pi, deg, re
76            REAL*8  gmagmo
77            REAL*8  eclipt, geoid(3), uma(30)
78  C
79  C       DATAS
80  C
81  C       proton mass [kg]
82          DATA mp /1.672610356D-27/
83            REAL*8 mp
84  C       electron charge [C]
85          DATA e /1.6021927D-19/
86            REAL*8 e
87  C       light velocity [m/s]
88          DATA c /2.997925D+8/
89            REAL*8 c
90  C       unit conversion of energy from MeV to J
91          DATA mevtoj / 1.6021927D-13/
92            REAL*8 mevtoj
93  C       proton mass/electron mass ratio
94          DATA pmonem /1836.083D0/
95            REAL*8 pmonem
96  C
97  C
98          NAMELIST/particle/ a, z, en
99            INTEGER*4 a, z
100           REAL*8    en
101         NAMELIST/initcond/ geodalt, geodlat, geodlong, velrho,
102        >                   veltheta, velphi
103           REAL*8    geodalt, geodlat, geodlong, velrho,
104        >            veltheta, velphi
105         NAMELIST/magnfield/ kint, year
106           REAL*8    year
107         NAMELIST/meth/ imeth
108         NAMELIST/integr/ fintim, hmin, eps
109           REAL*8    fintim, hmin, eps
110         NAMELIST/outfile/ nokwrite
111           INTEGER*4 nokwrite
112 C
113 C
114 C
115 C
116 C       ------------------------------------------------------------
117 C                 CONTENTS OF INPUT FILE 'UNITRAJ.NML'
118 C       ------------------------------------------------------------
119 C
120 C       a :         if a>0, the rest mass of the particle is
121 C                   a*(proton mass);
122 C                   if a=0, the particle is an electron and its
123 C                   rest mass is (proton mass)/1836.
124 C       z :         charge of the particle is z*(proton charge)
125 C       en :        kinetic energy of the particle in MeV
126 C       ------------------------------------------------------------
127 C       - INITIAL POINT POSITION :
128 C       geodalt :  geodetic altitude in km
129 C       geodlat :  geodetic latitude in deg
130 C       geodlong:  geodetic longitude in deg
131 C       - SPHERICAL VECTOR PARALLEL TO DIRECTION OF INITIAL
132 C         VELOCITY :
133 C       velrho :   rho-component
134 C       veltheta : theta-component
135 C       velphi :   phi-component
136 C       ------------------------------------------------------------
137 C       GEOMAGNETIC FIELD MODEL (see UNILIB, UM510) :
138 C       kint = 0 - DGRF/IGRF 45-95
139 C              1(-1) - Jensen & Cain 1962 (correction for SAA
140 C                      westward drift)
```

```
C            2(-2) - GSFC 12/66 (correction for SAA westward
C                    drift)
C            3 - Dipolar components of IGRF
C            4 - DGRF/IGRF 45-95 with Kluge algorithm
C      year : year to evaluate the model
C      ------------------------------------------------------------
C      imeth= 1 - CK, a fifth order runge-kutta (Fehlberg, with
C                 Cash-Karp coefficients)
C            2 - BS, Bulirsh-Stoer extrapolation method
C      ------------------------------------------------------------
C      fintim :   final time in seconds
C      hmin :     minimum value of the CK or BS step
C      eps :      For a CK or a BS step, relative error factor
C                 (estimated error of the step is kept lower than
C                 eps*[a scale factor])
C      ------------------------------------------------------------
C      nokwrite : - For BS, 1.
C                 - For CK, number aimed at avoiding too big
C                 output files: these are filled only every
C                 nokwrite steps (only the final point is written
C                 for all nokwrite).  The plotting of the
C                 trajectory of the particle is less precise
C                 for high values of nokwrite!
C
C      ------------------------------------------------------------
C      ------------------------------------------------------------
C
C
C                                  ***
C
C
C      ------------------------------------------------------------
C                        CONTENTS OF OUTPUT FILES
C      ------------------------------------------------------------
C
C                    'CKTIME.DAT' and 'BSTIME.DAT':
C             Times of the intermediate points in SECONDS
C      ------------------------------------------------------------
C                'CKPOINTS.DAT' and 'BSPOINTS.DAT':
C          Intermediate points given as: geodetic altitude in
C            km, geodetic latitude and longitude in degrees,
C           spherical components of the velocity (unit: light
C                              velocity c).
C      ------------------------------------------------------------
C                   'CKPOS.DAT' and 'BSPOS.DAT':
C                    Positions of the particle
C        - the three cartesian components being in EARTH RADIUS -
C              aimed at plotting the particle trajectory.
C        For BS, this aim is fulfilled only if EPS is low enough.
C      ------------------------------------------------------------
C                       'CKGC.DAT' and 'BSGC.DAT':
C             Locations of the guiding center (the three
C              cartesian components are in EARTH RADIUS)
C                aimed at plotting guiding center lines
C      ------------------------------------------------------------
C                    'CKERRV.DAT' and 'BSERRV.DAT':
C                  Relative errors on constant velocity
C      ------------------------------------------------------------
C                       'CKMU.DAT' and 'BSMU.DAT':
C                  Values of the adiabatically invariant
C                      magnetic moment in SI UNITS
C              (two columns, for two means to calculate it:
C              with evaluation of the magnetic field at the
C           position of the particle and at the guiding center)
C
C      ------------------------------------------------------------
C      ------------------------------------------------------------
C
C
C
```

```
211 C
212 C
213 C
214 C      *************
215 C      FILE OPENINGS
216 C      *************
217 C
218 C      input/namelist file
219        open(unit=10, file='unitraj.nml', status='old')
220        read(10, particle)
221        read(10, initcond)
222        read(10, magnfield)
223        read(10, meth)
224        read(10, integr)
225        read(10, outfile)
226 C
227 C      output files
228        if (imeth.eq.1) then
229          open(unit=210, file='cktime.dat', status='new')
230          open(unit=211, file='ckpoints.dat', status='new')
231          open(unit=212, file='ckpos.dat', status='new')
232          open(unit=213, file='ckgc.dat', status='new')
233          open(unit=220, file='ckerrv.dat', status='new')
234          open(unit=230, file='ckmu.dat', status='new')
235        else if (imeth.eq.2) then
236          open(unit=210, file='bstime.dat', status='new')
237          open(unit=211, file='bspoints.dat', status='new')
238          open(unit=212, file='bspos.dat', status='new')
239          open(unit=213, file='bsgc.dat', status='new')
240          open(unit=220, file='bserrv.dat', status='new')
241          open(unit=230, file='bsmu.dat', status='new')
242        else
243          write(6,*)'ERROR IN NAMELIST METH: IMETH MUST BE 1 OR 2'
244          STOP
245        endif
246 C
247 C      *************************
248 C      OPENING OF UNILIB LIBRARY
249 C      *************************
250 C
251        kunit = 6
252        kinit = 1
253        CALL UT990(kunit, kinit, ifail)
254 C
255 C      *********************************************
256 C      INITIALISATION OF GEOMAGNETIC FIELD IN UNILIB
257 C      *********************************************
258 C
259        kunit = 6
260        CALL UM510 (kint, year, lbint, kunit, ifail)
261 C
262 C
263 C      ***************************
264 C      THE PARTICLE AND ITS MOTION
265 C      ***************************
266 C
267 C      Kinetic energy [from MeV to J]
268        en = en * mevtoj
269 C
270 C      Charge of the particle
271        q = z*e
272        write(6,*) ' '
273        write(6,*) 'Charge of the particle            = ', q, ' C'
274
275 C      Rest mass of the particle
276        if(a.gt.0)then
277          massr = a*mp
278        else
279          massr = mp/pmonem
280        endif
```

```
281  C
282  C     Moment of the particle
283        p = DSQRT( (en+massr*c*c)**2 - (massr*c*c)**2 )
284       :                       /c
285  C
286  C     Velocity of the particle
287  C
288        v = (1/c)**2 + (massr/p)**2
289        v = 1/v
290        v = DSQRT(v)
291        write(6,*)'Velocity of the particle          = ', v/c, ' c'
292  C
293  C     Relativistic mass of the particle
294        gamma = 1/DSQRT(1-(v/c)**2)
295        massp = gamma*massr
296        write(6,*)'Relativistic mass of the particle = ', massp, ' kg'
297  C
298  C     ****************************
299  C     INITIAL VALUES OF THE MOTION
300  C     ****************************
301  C
302        mkgde.radius = re + geodalt
303        mkgde.colat  = 90.d0-geodlat
304        mkgde.elong  = geodlong
305  C     Geodetic to geocentric conversion
306        CALL UM536(mkgde,mkgeo)
307  C     Geocentric to cartesian conversion
308        CALL UT541(mkgeo,mkxyz)
309        vspher.rho   = velrho
310        vspher.theta = veltheta
311        vspher.phi   = velphi
312        vspher.dnrm  = DSQRT(vspher.rho**2 + vspher.theta**2 +
313       >                     vspher.phi**2)
314  C     Spherically coordinated vector converted into cartesian one
315        CALL UT542 (mkgeo,vspher,vcart)
316  C     conversion from km to m
317        ystart(1) = mkxyz.x*1000.D0
318        ystart(2) = mkxyz.y*1000.D0
319        ystart(3) = mkxyz.z*1000.D0
320        ystart(4) = v*(vcart.x/vspher.dnrm)
321        ystart(5) = v*(vcart.y/vspher.dnrm)
322        ystart(6) = v*(vcart.z/vspher.dnrm)
323  C
324  C     ****************************
325  C     TIME INTERVAL OF INTEGRATION
326  C     ****************************
327  C
328  C     initial time
329        t1 = 0.D0
330  C     final time
331        t2 = fintim
332  C
333  C     *****************
334  C     UNITS TO BE USED
335  C     *****************
336  C
337  C     lu : length unit, equals to mean earth radius (6371200 m)
338  C     tu : time unit, such that the constant velocity of particle
339  C          equals 1
340  C
341  C     mean earth radius is given in km as re in UNILIB
342        lu = re*1000.D0
343        vu = v
344  C     Time unit tu = length unit/velocity of the particle
345        tu = lu/vu
346  C
347  C     *****************
348  C     UNITS CONVERSION
349  C     *****************
350  C
```

```
C      from S.I. to (lu, tu) units
C
       ystart(1) = ystart(1)/lu
       ystart(2) = ystart(2)/lu
       ystart(3) = ystart(3)/lu
       ystart(4) = ystart(4)/vu
       ystart(5) = ystart(5)/vu
       ystart(6) = ystart(6)/vu
       t1 = t1/tu
       t2 = t2/tu
       v = v/vu
       c = c/vu
C
C      ***********
C      INTEGRATION
C      ***********
C
C
       nutime     = 210
       nupoints   = 211
       nutraj     = 212
       nugc       = 213
       nuerrv     = 220
       numagnmom = 230
C
       write(nutime,*) 0.D0
       write(nupoints,1110) geodalt, geodlat, geodlong,
      >                     (v/c)*velrho/vspher.dnrm,
      >                     (v/c)*veltheta/vspher.dnrm,
      >                     (v/c)*velphi/vspher.dnrm
 1110  format (F9.2, F9.2, F9.2, F9.5, F9.5, F9.5)
       write(nutraj,1111) ystart(1), ystart(2), ystart(3)
 1111  format (F9.5, F9.5, F9.5)
C
C
C
       h1=0.05D0
C
       write(6,*)' '
       if(imeth.eq.1)then
         write(6,*)'--------------------------------------'
         write(6,*)' NUMERICAL INTEGRATION OF THE MOTION'
         write(6,*)'           EQUATIONS WITH A           '
         write(6,*)' FIFTH ORDER RUNGE-KUTTA  (CASH-KARP) '
         write(6,*)'-------------------------------------- '
         write(6,*)' '
       else
         write(6,*)'--------------------------------------'
         write(6,*)' NUMERICAL INTEGRATION OF THE MOTION'
         write(6,*)'          EQUATIONS WITH AN           '
         write(6,*)'EXTRAPOLATION METHOD  (BULIRSCH-STOER) '
         write(6,*)'-------------------------------------- '
         write(6,*)' '
       endif
C
       call odeint2(ystart, t1, t2, eps, h1, hmin, nutime,
      >   nupoints, nutraj, nugc, nuerrv, numagnmom, nokwrite)
C
       write(6,*)'REACHED TIME= ', t2*tu, 'SECONDS'
C
       END
end
```

```
1         PROGRAM MIRP
2  C
3  C
4  C      ****************************************************************
5  C      ****************************************************************
6  C      *                                                              *
7  C      *          Reads files 'cktime' and 'ckpoints', or             *
8  C      *   'bstime' and 'bspoints',  in order to find mirror points   *
9  C      *          as particle positions of minimum altitude           *
10 C      *                                                              *
11 C      ****************************************************************
12 C      ****************************************************************
13 C
14 C
15        EXTERNAL INTERPOL
16 C
17        INTEGER*4 imeth, i, init, nread
18        REAL*8    t(3), h(3), lat(3), long(3), TIME(3), ALT(3),
19       >          LATIT(3), LONGIT(3), hstart, latstart, longstart,
20       >          hend, latend, tmp, hmin, latmp, longmp
21 C
22 C      ASK USER TO MAKE A CHOICE (CK or BS)
23  500   write(6,*)'If you want to work with CK files >>>> type 1'
24        write(6,*)'If you want to work with BS files >>>> type 2'
25        read(6,*) imeth
26 C
27 C
28 C      INPUT FILES
29        if (imeth.eq.1) then
30          open(unit=10, file='cktime.dat', status='old')
31          open(unit=20, file='ckpoints.dat', status='old')
32        else if (imeth.eq.2) then
33          open(unit=10, file='bstime.dat', status='old')
34          open(unit=20, file='bspoints.dat', status='old')
35        else
36          go to 500
37        endif
38 C      OUTPUT FILES
39 C      Information about mirror points
40        open(unit=30, file='mirpoints.dat', status='new')
41 C      Northern mirror points: times, altitudes, longitudes
42        open(unit=31, file='northmp.dat', status='new')
43 C      Southern mirror points: times, altitudes, longitudes
44        open(unit=32, file='southmp.dat', status='new')
45 C
46 C
47 C      FIRST CANDIDATE
48 C      (A CANDIDATE = A POINT AND ITS TWO NEIGHBOURS)
49        do i=1,3
50          read(10,*)t(i)
51          read(20,1000) h(i), lat(i), long(i)
52   1000   FORMAT(F9.2, F9.2, F9.2)
53        enddo
54        nread = 1
55        hstart = DMIN1(h(1),h(2),h(3))
56        latstart = lat(1)
57        longstart = DMAX1(long(1), long(2), long(3))
58 C      NO GOOD CANDIDATE FOUND YET FOR FIRST MIRROR POINT
59        init=0
60 C
61     1  if(h(1).ge.h(2) .and. h(3).ge.h(2))then
62 C      GOOD CANDIDATE
63          if (init.eq.0) then
64            if ( h(2).lt.hstart .or.
65       >         latstart*lat(2).le.0 ) then
66 C            FIRST GOOD CANDIDATE FOR FIRST MIRROR POINT
67              init = 1
68              do i=1,3
69                TIME(i) = t(i)
70                ALT(i)  = h(i)
```

```
71                LATIT(i)= lat(i)
72                LONGIT(i) = long(i)
73              enddo
74            endif
75          else
76  C       NEW GOOD CANDIDATE FOR CURRENT MIRROR POINT,
77  C       OR FIRST GOOD CANDIDATE FOR NEXT MIRROR POINT
78            if (lat(2)*LATIT(2) .gt. 0) then
79  C         NEW GOOD CANDIDATE FOR CURRENT MIRROR POINT
80  C         ==> SEARCH THE BEST CANDIDATE (THE GOOD CANDIDATE
81  C         WITH MINIMUM ALTITUDE)
82              if (h(2).lt.ALT(2)) then
83                do i=1,3
84                  TIME(i) = t(i)
85                  ALT(i)  = h(i)
86                  LATIT(i)= lat(i)
87                  LONGIT(i) = long(i)
88                enddo
89              endif
90            else
91  C         BEST CANDIDATE FOR CURRENT MIRROR POINT FOUND
92  C         ==> CARRY OUT A PARABOLIC INTERPOLATION
93              CALL INTERPOL(TIME, ALT, LATIT, LONGIT,
94       >                    tmp, hmin, latmp, longmp)
95              write(30,*) ' '
96              write(30,*) 'mirror point at altitude ',hmin,' km'
97              write(30,*) '              at longitude',longmp,' deg'
98              write(30,*) '              at latitude ',latmp,' deg'
99              write(30,*) '              at time     ',tmp, ' sec'
100             if (latmp.gt.0) then
101               write(31,2000) tmp, hmin, longmp, latmp
102  2000         format(F17.10,F17.10,F17.10,F17.10)
103             else
104               write(32,2000) tmp, hmin, longmp, latmp
105             endif
106 C
107 C         FIRST GOOD CANDIDATE FOR NEXT MIRROR POINT
108           do i=1,3
109             TIME(i) = t(i)
110             ALT(i)  = h(i)
111             LATIT(i)= lat(i)
112             LONGIT(i) = long(i)
113           enddo
114         endif
115       endif
116     endif
117 C
118     t(1) = t(2)
119     h(1) = h(2)
120     lat(1) = lat(2)
121     long(1) = long(2)
122     t(2) = t(3)
123     h(2) = h(3)
124     lat(2) = lat(3)
125     long(2) = long(3)
126     read(10,*,end=2001) t(3)
127     read(20,1000,end=2001) h(3), lat(3), long(3)
128     nread = nread+1
129     go to 1
130 C
131 C
132  2001 hend = DMIN1(h(1),h(2),h(3))
133     latend = lat(2)
134     if ( latend*LATIT(2).le.0.D0 .or.
135    >    hend.gt.ALT(2) ) then
136 C     LAST MIRROR POINT
137       CALL INTERPOL (TIME, ALT, LATIT, LONGIT,
138    >                 tmp, hmin, latmp, longmp)
139       write(30,*) ' '
140       write(30,*) 'mirror point at altitude ',hmin,' km'
```

```
141           write(30,*) '              at longitude',longmp,' deg'
142           write(30,*) '              at latitude ',latmp,' deg'
143           write(30,*) '              at time     ',tmp,' sec'
144           if (latmp.gt.0) then
145             write(31,2000) tmp, hmin, longmp, latmp
146           else
147             write(32,2000) tmp, hmin, longmp, latmp
148           endif
149         endif
150         write(30,*) ' '
151         write(30,*) 'Results obtained from ', nread, ' points'
152         write(30,*) ' '
153   C
154   C
155         END
end
```

```
      PROGRAM MIRPGC
C
C
C     ************************************************************************
C     ************************************************************************
C     *                                                                      *
C     *    Reads files 'cktime' and 'ckgc', or files 'bstime' and 'bsgc',    *
C     *    in order to find mirror points as guiding centers of minimum      *
C     *    altitude.                                                         *
C     *    Output files: 'mirpgc.dat' (times and altitudes of points)        *
C     *                  'northmpgc.dat'(for northern points only)           *
C     *                  'southmpgc.dat'(for southern points only)           *
C     *                                                                      *
C     ************************************************************************
C     ************************************************************************
C
C
C
      EXTERNAL parabint
C
      INTEGER*4 imeth
      REAL*8 t(4), gcx(3), gcy(3), gcz(3), alt(4)
      REAL*8 tin, y, dy
C
      REAL*8 re
      DATA re / 6371.2D0/
C
C
C     ASKS THE USER TO CHOICE BETWEEN CK AND BS
    1 write(6,*) 'If you want to work with CK files >>> type 1'
      write(6,*) 'If you want to work with BS files >>> type 2'
      read(6,*) imeth
C
C     INPUT FILES
      if (imeth.eq.1) then
        open(unit=10, file='cktime.dat', status='old')
        open(unit=20, file='ckgc.dat', status='old')
      else if (imeth.eq.2) then
        open(unit=10, file='bstime.dat', status='old')
        open(unit=20, file='bsgc.dat', status='old')
      else
        go to 1
      endif
C     OUTPUT FILE
      open(unit=30, file='mirpgc.dat', status='new')
      open(unit=31, file='northmpgc.dat', status='new')
      open(unit=32, file='southmpgc.dat', status='new')
C
C
      do i=1,3
        read(10,*)t(i)
        read(20,*) gcx(i), gcy(i), gcz(i)
        gcx(i) = gcx(i)*re
        gcy(i) = gcy(i)*re
        gcz(i) = gcz(i)*re
        alt(i) = DSQRT(gcx(i)**2+gcy(i)**2+gcz(i)**2)-re
      enddo
C
    2 if(alt(1).gt.alt(2) .and. alt(3).ge.alt(2))then
        call parabint(t, alt)
        write(30,*) ' '
        write(30,*) 'mirror point at altitude ', alt(4),' km'
        write(30,*) '              at time     ', t(4),' sec'
        if (gcz(2).gt.0) then
          write(31,2000) t(4), alt(4)
        else
          write(32,2000) t(4), alt(4)
        endif
 2000   format(F17.10, F17.10)
      endif
```

```
71          t(1) = t(2)
72          gcx(1) = gcx(2)
73          gcy(1) = gcy(2)
74          gcz(1) = gcz(2)
75          alt(1) = DSQRT(gcx(1)**2 + gcy(1)**2 + gcz(1)**2)-re
76          t(2) = t(3)
77          gcx(2) = gcx(3)
78          gcy(2) = gcy(3)
79          gcz(2) = gcz(3)
80          alt(2) = DSQRT(gcx(2)**2 + gcy(2)**2 + gcz(2)**2)-re
81          read(10,*,end=1000) t(3)
82          read(20,*,end=1000) gcx(3), gcy(3), gcz(3)
83          gcx(3) = gcx(3)*re
84          gcy(3) = gcy(3)*re
85          gcz(3) = gcz(3)*re
86          alt(3) = DSQRT(gcx(3)**2 + gcy(3)**2 + gcz(3)**2)-re
87          go to 2
88    C
89    C
90     1000 CONTINUE
91          END
end
```

```
      PROGRAM ALTMIN
C
C     ************************************************************
C     ************************************************************
C     *                                                          *
C     *      Reads files 'northmp.dat' and 'southmp.dat'         *
C     *   in order to find minimum altitude of a drift shell     *
C     *      in the Northern and the Southern hemispheres        *
C     *         (results are put in file 'altmin.dat')           *
C     *                                                          *
C     ************************************************************
C     ************************************************************
C
      EXTERNAL parabint, POLINT
C
      INTEGER*4 i, nread, imin
      REAL*8    hmin, time(4), ord(4), y, dy
C
      INTEGER*4 NMAX
      PARAMETER (NMAX=100)
C
      REAL*8    t(NMAX), hmp(NMAX), long(NMAX), lat(NMAX)
C
      open (unit=20, file='northmp.dat', status='old')
      open (unit=30, file='southmp.dat', status='old')
      open (unit=40, file='altmin.dat', status='new')
C
C
C
      nread = 0
      hmin = 10.D+15
C
C     NORTH
C
      write(40,*)'------NORTHERN HEMISPHERE------'
C
C     Search for the northern mirror point of minimum altitude
      do i=1, NMAX
        read(20,*,end=1000) t(i), hmp(i), long(i), lat(i)
        nread = nread+1
        if(nread.gt.NMAX) then
          write(6,*)'too many points!'
          STOP
        endif
        if(hmp(i).lt.hmin)then
          hmin = hmp(i)
          imin = i
        endif
      enddo
C
 1000 if (imin.gt.1) then
        time(1) = t(imin-1)
        time(2) = t(imin)
        time(3) = t(imin+1)
        ord(1)  = hmp(imin-1)
        ord(2)  = hmp(imin)
        ord(3)  = hmp(imin+1)
C       Parabolic interpolation of altitude and search for the minimum
        call parabint(time,ord)
        write(40,*) 'Minimum altitude [km] = ', ord(4)
C       Parabolic interpolation of longitude
        ord(1)=long(imin-1)
        ord(2)=long(imin)
        ord(3)=long(imin+1)
        CALL POLINT (time, ord, 3, time(4), y, dy)
        write(40,*) 'Longitude [deg] = ', y
C       Parabolic interpolation of latitude
        ord(1)=lat(imin-1)
        ord(2)=lat(imin)
        ord(3)=lat(imin+1)
```

```
71          CALL POLINT (time, ord, 3, time(4), y, dy)
72          write(40,*) 'Latitude [deg] = ', y
73          write(40,*) 'Time [sec] = ', time(4)
74        else
75          write(6,*)'imin not greater than 1 in ALTMIN'
76        endif
77  C
78  C     SOUTH
79  C
80        nread = 0
81        hmin = 10.D+15
82  C
83        write(40,*)'    '
84        write(40,*)'------SOUTHERN HEMISPHERE------'
85  C
86  C     Search for the southern mirror point of minimum altitude
87        do i=1, NMAX
88          read(30,*,end=2000) t(i), hmp(i), long(i), lat(i)
89          nread = nread+1
90          if(nread.gt.NMAX) then
91            write(6,*)'too many points!'
92            STOP
93          endif
94          if(hmp(i).lt.hmin)then
95            hmin = hmp(i)
96            imin = i
97          endif
98        enddo
99  C
100  2000 if (imin.gt.1) then
101         time(1) = t(imin-1)
102         time(2) = t(imin)
103         time(3) = t(imin+1)
104         ord(1)  = hmp(imin-1)
105         ord(2)  = hmp(imin)
106         ord(3)  = hmp(imin+1)
107 C       Parabolic interpolation of altitude and search for the minimum
108         call parabint(time,ord)
109         write(40,*) 'Minimum altitude [km] = ', ord(4)
110 C       Interpolation of the longitude
111         ord(1)=long(imin-1)
112         ord(2)=long(imin)
113         ord(3)=long(imin+1)
114         CALL POLINT (time, ord, 3, time(4), y, dy)
115         write(40,*) 'Longitude [deg]= ', y
116 C       Parabolic interpolation of latitude
117         ord(1)=lat(imin-1)
118         ord(2)=lat(imin)
119         ord(3)=lat(imin+1)
120         CALL POLINT (time, ord, 3, time(4), y, dy)
121         write(40,*) 'Latitude [deg] = ', y
122         write(40,*) 'Time [sec] = ', time(4)
123       else
124         write(6,*)'imin not greater than 1 in ALTMIN'
125       endif
126 C
127 C
128       END
end
```

```
      PROGRAM ALTMINGC
C
C     ************************************************************
C     ************************************************************
C     *                                                          *
C     *    Reads files 'northmpgc.dat' and 'southmpgc.dat'       *
C     *   in order to find minimum altitude of a drift shell     *
C     *      in the Northern and the Southern hemispheres        *
C     *        (results are put in file 'altmingc.dat')          *
C     *                                                          *
C     ************************************************************
C     ************************************************************
C
      EXTERNAL parabint
C
      INTEGER*4 i, nread, imin
      REAL*8    hmin, time(4), ord(4)
C
      INTEGER*4 NMAX
      PARAMETER (NMAX=100)
C
      REAL*8    t(NMAX), hmp(NMAX), long(NMAX), lat(NMAX)
C
      open (unit=20, file='northmpgc.dat', status='old')
      open (unit=30, file='southmpgc.dat', status='old')
      open (unit=40, file='altmingc.dat', status='new')
C
C
C
      nread = 0
      hmin = 10.D+15
C
C     NORTH
C
      write(40,*)'------NORTHERN HEMISPHERE------'
C
C     Search for the northern mirror point of minimum altitude
      do i=1, NMAX
        read(20,*,end=1000) t(i), hmp(i)
        nread = nread+1
        if(nread.gt.NMAX) then
          write(6,*)'too many points!'
          STOP
        endif
        if(hmp(i).lt.hmin)then
          hmin = hmp(i)
          imin = i
        endif
      enddo
C
 1000 if (imin.gt.1) then
        time(1) = t(imin-1)
        time(2) = t(imin)
        time(3) = t(imin+1)
        ord(1)  = hmp(imin-1)
        ord(2)  = hmp(imin)
        ord(3)  = hmp(imin+1)
C       Parabolic interpolation of altitude and search for the minimum
        call parabint(time,ord)
        write(40,*) 'Minimum altitude [km] = ', ord(4)
        write(40,*) 'Time [sec] = ', time(4)
      else
        write(6,*)'imin not greater than 1 in ALTMIN'
      endif
C
C     SOUTH
C
      nread = 0
      hmin = 10.D+15
C
```

```
71          write(40,*)'    '
72          write(40,*)'------SOUTHERN HEMISPHERE------'
73    C
74    C     Search for the southern mirror point of minimum altitude
75          do i=1, NMAX
76            read(30,*,end=2000) t(i), hmp(i)
77            nread = nread+1
78            if(nread.gt.NMAX) then
79              write(6,*)'too many points!'
80              STOP
81            endif
82            if(hmp(i).lt.hmin)then
83              hmin = hmp(i)
84              imin = i
85            endif
86          enddo
87    C
88     2000 if (imin.gt.1) then
89            time(1) = t(imin-1)
90            time(2) = t(imin)
91            time(3) = t(imin+1)
92            ord(1)  = hmp(imin-1)
93            ord(2)  = hmp(imin)
94            ord(3)  = hmp(imin+1)
95    C       Parabolic interpolation of altitude and search for the minimum
96            call parabint(time,ord)
97            write(40,*) 'Minimum altitude [km] = ', ord(4)
98            write(40,*) 'Time [sec] = ', time(4)
99          else
100           write(6,*)'imin not greater than 1 in ALTMIN'
101         endif
102   C
103   C
104         END
end
```

```
      PROGRAM FSTADDIP
C
C
C     ************************************************************************
C     ************************************************************************
C     *                                                                      *
C     *  The main goal of this program is to study the first adiabatic       *
C     *  invariant of the trajectory of a charged particle in the            *
C     *  Earth dipole field, that si to say the magnetic moment of that      *
C     *  particle, for a long interval of time.                              *
C     *                                                                      *
C     *  The information on the particle and on the numerical method         *
C     *  used to integrate the ordinary differential equation of the         *
C     *  particle (two main possibilities: the Runge-Kutta-Cash-Karp         *
C     *  method and the Bulirsch-Stoer method) must be given in a namelist   *
C     *  file.                                                               *
C     *                                                                      *
C     *  The output file CKMUDIPAN.DAT or BSMUDIPAN.DAT is produced by the   *
C     *  subroutine MULGTIME and contains information about the mirror       *
C     *  points of the particle: their integration time, their altitude      *
C     *  in kilometers, their latitude and longitude in degrees, their       *
C     *  magnetic intensity in Gauss, their magnetic moment in SI,           *
C     *  the related bouncing period (time between the mirror point and      *
C     *  its second predecessor); the velocity of the particle at last       *
C     *  computed point is also written to witness the quality of the        *
C     *  computation (this absolute invariant is 1, according to the         *
C     *  chosen units).                                                      *
C     *                                                                      *
C     ************************************************************************
C     ************************************************************************
C
C
C
C      ------------------------------------------------------------
C                CONTENTS OF INPUT FILE 'FSTADIABINV.NML'
C      ------------------------------------------------------------
C
C      a :          if a>0, the rest mass of the particle is
C                   a*(proton mass);
C                   if a=0, the particle is an electron and its
C                   rest mass is (proton mass)/1836.
C      z :          charge of the particle is z*(proton charge)
C      en :         kinetic energy of the particle in MeV
C      ------------------------------------------------------------
C      - INITIAL POINT POSITION :
C      geodalt :  geodetic altitude in km
C      geodlat :  geodetic latitude in deg
C      geodlong:  geodetic longitude in deg
C      - SPHERICAL VECTOR PARALLEL TO DIRECTION OF INITIAL
C        VELOCITY :
C      velrho :   rho-component
C      veltheta : theta-component
C      velphi :   phi-component
C      ------------------------------------------------------------
C      GEOMAGNETIC FIELD MODEL (see UNILIB, UM510) :
C      kint = 0 - DGRF/IGRF 45-95
C             1(-1) - Jensen & Cain 1962 (correction for SAA
C                     westward drift)
C             2(-2) - GSFC 12/66 (correction for SAA westward
C                     drift)
C             3 - Dipolar components of IGRF
C             4 - DGRF/IGRF 45-95 with Kluge algorithm
C      year : year to evaluate the model
C      ------------------------------------------------------------
C      imeth= 1 - CK, a fifth order runge-kutta (Fehlberg, with
C                 Cash-Karp coefficients)
C             2 - BS, Bulirsh-Stoer extrapolation method
C      ------------------------------------------------------------
C      fintim :   final time in seconds
C      hmin :     minimum value of the CK or BS step
```

```
C     eps :       For a CK or a BS step, relative error factor
C                 (estimated error of the step is kept lower than
C                 eps*[a scale factor])
C     ------------------------------------------------------------
C
C
C                                  ***
C
C
C     ------------------------------------------------------------
C                       CONTENTS OF OUTPUT FILES
C     ------------------------------------------------------------
C
C                 'CKMUDIPAN.DAT' and 'BSMUDIPAN.DAT':
C
C     Sets of values related to the mirror points:
C      Time of mirror point in seconds
C      Altitude of mirror point in kilometers
C      Latitude of mirror point
C      Longitude of mirror point
C      Magnetic intensity at mirror point in Gauss
C      Magnetic moment at mirror point in SI units
C      bouncing period in seconds
C      velocity
C
C     ------------------------------------------------------------
C     ------------------------------------------------------------
C
C
C
      EXTERNAL mulgtime
C     (and UNILIB routines: UT990, UM510, UM536, UT541, UT542)
C
      INCLUDE 'structure.h'
C
      INTEGER*4 kunit, kinit, ifail, numagnmom
C
      REAL*8    p, v, gamma, ystart(6), t1, t2, h1, b(3), bnorm,
     >          bmk
C
      CHARACTER*32 lbint
C
      RECORD /zgeo/ mkgde, mkgeo, mpos
      RECORD /zxyz/ mkxyz, vcart, bcart
      RECORD /zvec/ vspher, mb
C
C
      COMMON/particule/ q, massp, massr
        REAL*8  q, massp, massr
      COMMON/methode/  imeth
        INTEGER*4 imeth
      COMMON/lightvel/ c
      COMMON/units/ lu, tu, vu
        REAL*8  lu, tu, vu
      COMMON/mfmodel/ kint
        INTEGER*4 kint
C
      COMMON/UC140/ mint, mext, msun
        RECORD/zimf/mint
        RECORD/zsun/msun
        RECORD/zemf/mext
C
      COMMON /UC160/ pi, deg, re, gmagmo, eclipt, geoid, uma
        REAL*8  pi, deg, re
        REAL*8  gmagmo
        REAL*8  eclipt, geoid(3), uma(30)
C
C     ***************************
C      mom: earth dipolar moment
C     ***************************
```

```
141 C
142       COMMON/mom/mom
143         real*8 mom
144 C
145 C     DATAS
146 C
147 C     proton mass [kg]
148       DATA mp /1.672610356D-27/
149         REAL*8 mp
150 C     electron charge [C]
151       DATA e /1.6021927D-19/
152         REAL*8 e
153 C     light velocity [m/s]
154       DATA c /2.997925D+8/
155         REAL*8 c
156 C     unit conversion of energy from MeV to J
157       DATA mevtoj / 1.6021927D-13/
158         REAL*8 mevtoj
159 C     proton mass/electron mass ratio
160       DATA pmonem /1836.083D0/
161         REAL*8 pmonem
162 C
163 C
164       NAMELIST/particle/ a, z, en
165         INTEGER*4 a, z
166         REAL*8    en
167       NAMELIST/initcond/ geodalt, geodlat, geodlong, velrho,
168      >                   veltheta, velphi
169         REAL*8    geodalt, geodlat, geodlong, velrho,
170      >            veltheta, velphi
171       NAMELIST/magnfield/ kint, year
172         REAL*8    year
173       NAMELIST/meth/ imeth
174       NAMELIST/integr/ fintim, hmin, eps
175         REAL*8    fintim, hmin, eps
176 C
177 C
178 C     *************
179 C     FILE OPENINGS
180 C     *************
181 C
182 C     input/namelist file
183       open(unit=10, file='fstaddip.nml', status='old')
184       read(10, particle)
185       read(10, initcond)
186       read(10, magnfield)
187       read(10, meth)
188       read(10, integr)
189 C
190 C     output files
191       if (imeth.eq.1) then
192         open(unit=230, file='ckmudipan.dat', status='new')
193       else if (imeth.eq.2) then
194         open(unit=230, file='bsmudipan.dat', status='new')
195       else
196         write(6,*) 'ERROR IN NAMELIST METH: IMETH MUST BE 1 OR 2'
197         STOP
198       endif
199 C
200 C     *************************
201 C     OPENING OF UNILIB LIBRARY
202 C     *************************
203 C
204       kunit = 6
205       kinit = 1
206       CALL UT990(kunit, kinit, ifail)
207 C
208 C     ***************************
209 C     THE PARTICLE AND ITS MOTION
210 C     ***************************
```

```
211   C
212   C     Kinetic energy [from MeV to J]
213         en = en * mevtoj
214   C
215   C     Charge of the particle
216         q = z*e
217         write(6,*) ' '
218         write(6,*) 'Charge of the particle            = ', q, ' C'
219
220   C     Rest mass of the particle
221         if(a.gt.0)then
222           massr = a*mp
223         else
224           massr = mp/pmonem
225         endif
226   C
227   C     Moment of the particle
228         p = DSQRT( (en+massr*c*c)**2 - (massr*c*c)**2 )
229        :                       /c
230   C
231   C     Velocity of the particle
232   C
233         v = (1/c)**2 + (massr/p)**2
234         v = 1/v
235         v = DSQRT(v)
236         write(6,*)'Velocity of the particle          = ', v/c, ' c'
237   C
238   C     Relativistic mass of the particle
239         gamma = 1/DSQRT(1-(v/c)**2)
240         massp = gamma*massr
241         write(6,*)'Relativistic mass of the particle = ', massp, ' kg'
242   C
243   C     ****************************
244   C     INITIAL VALUES OF THE MOTION
245   C     ****************************
246   C
247         mkgde.radius = re + geodalt
248         mkgde.colat  = 90.d0-geodlat
249         mkgde.elong  = geodlong
250   C     Geodetic to geocentric conversion
251         CALL UM536(mkgde,mkgeo)
252   C     Geocentric to cartesian conversion
253         CALL UT541(mkgeo,mkxyz)
254         vspher.rho   = velrho
255         vspher.theta = veltheta
256         vspher.phi   = velphi
257         vspher.dnrm  = DSQRT(vspher.rho**2 + vspher.theta**2 +
258        >                     vspher.phi**2)
259   C     Spherically coordinated vector converted into cartesian one
260         CALL UT542 (mkgeo,vspher,vcart)
261   C     conversion from km to m
262         ystart(1) = mkxyz.x*1000.D0
263         ystart(2) = mkxyz.y*1000.D0
264         ystart(3) = mkxyz.z*1000.D0
265         ystart(4) = v*(vcart.x/vspher.dnrm)
266         ystart(5) = v*(vcart.y/vspher.dnrm)
267         ystart(6) = v*(vcart.z/vspher.dnrm)
268   C
269   C     ****************************
270   C     TIME INTERVAL OF INTEGRATION
271   C     ****************************
272   C
273   C     initial time
274         t1 = 0.D0
275   C     final time
276         t2 = fintim
277   C
278   C     ****************
279   C     UNITS TO BE USED
280   C     ****************
```

```
281  C
282  C      lu : length unit, equals to mean earth radius (6371200 m)
283  C      tu : time unit, such that the constant velocity of particle
284  C           equals 1
285  C
286  C      mean earth radius is given in km as re in UNILIB
287          lu = re*1000.D0
288          vu = v
289  C       Time unit tu = length unit/velocity of the particle
290          tu = lu/vu
291  C
292  C      ****************
293  C      UNITS CONVERSION
294  C      ****************
295  C
296  C      from S.I. to (lu, tu) units
297  C
298          ystart(1) = ystart(1)/lu
299          ystart(2) = ystart(2)/lu
300          ystart(3) = ystart(3)/lu
301          ystart(4) = ystart(4)/vu
302          ystart(5) = ystart(5)/vu
303          ystart(6) = ystart(6)/vu
304  C
305          t1 = t1/tu
306          t2 = t2/tu
307          v = v/vu
308          c = c/vu
309  C
310  C      ***************************
311  C       Earth dipolar moment
312  C       in SI: about 10**17/4*pi
313  C      ***************************
314  C
315          mom=7.815400593142153D+15
316  C
317  C
318  C      ***********
319  C      INTEGRATION
320  C      ***********
321  C
322          numagnmom = 230
323  C
324  C
325          h1=0.05D0
326  C
327          write(6,*)' '
328          if(imeth.eq.1)then
329            write(6,*)'-------------------------------------'
330            write(6,*)' NUMERICAL INTEGRATION OF THE MOTION'
331            write(6,*)'           EQUATIONS WITH A          '
332            write(6,*)' FIFTH ORDER RUNGE-KUTTA  (CASH-KARP) '
333            write(6,*)'-------------------------------------- '
334            write(6,*)' '
335          else
336            write(6,*)'-------------------------------------'
337            write(6,*)' NUMERICAL INTEGRATION OF THE MOTION'
338            write(6,*)'          EQUATIONS WITH AN          '
339            write(6,*)'EXTRAPOLATION METHOD  (BULIRSCH-STOER) '
340            write(6,*)'-------------------------------------- '
341            write(6,*)' '
342          endif
343  C
344          call mulgtime(ystart, t1, t2, eps, h1, hmin, numagnmom)
345  C
346          write(6,*)'REACHED TIME= ', t2*tu, 'SECONDS'
347  C
348          END
end
```

```
      PROGRAM FSTADIABINV
C
C
C     ************************************************************************
C     ************************************************************************
C     *                                                                      *
C     *  The main goal of this program is to study the first adiabatic       *
C     *  invariant of the trajectory of a charged particle in the            *
C     *  geomagnetic field, that si to say the magnetic moment of that       *
C     *  particle, for a long interval of time.                              *
C     *                                                                      *
C     *  The information on the particle, on the model of the geomagnetic    *
C     *  field (given by the UNILIB library) and on the numerical            *
C     *  method used to integrate the ordinary differential equation of      *
C     *  the particle (two main possibilities: the Runge-Kutta-Cash-Karp     *
C     *  method and the Bulirsch-Stoer method) must be given in a namelist   *
C     *  file.                                                               *
C     *                                                                      *
C     *  The output file CKMU.DAT or BSMU.DAT is produced by the             *
C     *  subroutine MULGTIME and contains information about the mirror       *
C     *  points of the particle: their integration time, their altitude      *
C     *  in kilometers, their latitude and longitude in degrees, their       *
C     *  magnetic intensity in Gauss, their magnetic moment in SI,           *
C     *  the related bouncing period (time between the mirror point and      *
C     *  its second predecessor); the velocity of the particle at last       *
C     *  computed point is also written to witness the quality of the        *
C     *  computation (this absolute invariant is 1, according to the         *
C     *  chosen units).                                                      *
C     *                                                                      *
C     ************************************************************************
C     ************************************************************************
C
C
C
C     ------------------------------------------------------------
C               CONTENTS OF INPUT FILE 'FSTADIABINV.NML'
C     ------------------------------------------------------------
C
C     a :         if a>0, the rest mass of the particle is
C                 a*(proton mass);
C                 if a=0, the particle is an electron and its
C                 rest mass is (proton mass)/1836.
C     z :         charge of the particle is z*(proton charge)
C     en :        kinetic energy of the particle in MeV
C     ------------------------------------------------------------
C     - INITIAL POINT POSITION :
C     geodalt :  geodetic altitude in km
C     geodlat :  geodetic latitude in deg
C     geodlong:  geodetic longitude in deg
C     - SPHERICAL VECTOR PARALLEL TO DIRECTION OF INITIAL
C       VELOCITY :
C     velrho :    rho-component
C     veltheta :  theta-component
C     velphi :    phi-component
C     ------------------------------------------------------------
C     GEOMAGNETIC FIELD MODEL (see UNILIB, UM510) :
C     kint = 0 - DGRF/IGRF 45-95
C            1(-1) - Jensen & Cain 1962 (correction for SAA
C                    westward drift)
C            2(-2) - GSFC 12/66 (correction for SAA westward
C                    drift)
C            3 - Dipolar components of IGRF
C            4 - DGRF/IGRF 45-95 with Kluge algorithm
C     year : year to evaluate the model
C     ------------------------------------------------------------
C     imeth= 1 - CK, a fifth order runge-kutta (Fehlberg, with
C                Cash-Karp coefficients)
C            2 - BS, Bulirsh-Stoer extrapolation method
C     ------------------------------------------------------------
C     fintim :    final time in seconds
```

```
71  C      hmin :      minimum value of the CK or BS step
72  C      eps :       For a CK or a BS step, relative error factor
73  C                  (estimated error of the step is kept lower than
74  C                  eps*[a scale factor])
75  C      -------------------------------------------------------------
76  C
77  C
78  C                                  ***
79  C
80  C
81  C      -------------------------------------------------------------
82  C                        CONTENTS OF OUTPUT FILES
83  C      -------------------------------------------------------------
84  C
85  C                        'CKMU.DAT' and 'BSMU.DAT':
86  C
87  C      Sets of values related to the mirror points:
88  C       Time of mirror point in seconds
89  C       Altitude of mirror point in kilometers
90  C       Latitude of mirror point
91  C       Longitude of mirror point
92  C       Magnetic intensity at mirror point in Gauss
93  C       Magnetic moment at mirror point in SI units
94  C       bouncing period in seconds
95  C       velocity
96  C
97  C      -------------------------------------------------------------
98  C      -------------------------------------------------------------
99  C
100 C
101 C
102        EXTERNAL mulgtime
103 C      (and UNILIB routines: UT990, UM510, UM536, UT541, UT542)
104 C
105        INCLUDE 'structure.h'
106 C
107 C
108        INTEGER*4 kunit, kinit, ifail, numagnmom, kext
109 C
110        REAL*8    p, v, gamma, ystart(6), t1, t2, h1,
111       $          param(10), amjd
112 C
113        CHARACTER*32 lbint, lbext
114 C
115        RECORD /zgeo/ mkgde, mkgeo, mpos
116        RECORD /zxyz/ mkxyz, vcart, bcart
117        RECORD /zvec/ vspher, mb
118 C
119        COMMON/particule/ q, massp, massr
120          REAL*8  q, massp, massr
121        COMMON/methode/  imeth
122          INTEGER*4 imeth
123        COMMON/lightvel/ c
124        COMMON/units/ lu, tu, vu
125          REAL*8  lu, tu, vu
126        COMMON/mfmodel/ kint
127          INTEGER*4 kint
128 C
129        COMMON/UC140/ mint, mext, msun
130          RECORD/zimf/mint
131          RECORD/zsun/msun
132          RECORD/zemf/mext
133 C
134        COMMON /UC160/ pi, deg, re, gmagmo, eclipt, geoid, uma
135          REAL*8  pi, deg, re
136          REAL*8  gmagmo
137          REAL*8  eclipt, geoid(3), uma(30)
138 C
139 C      DATAS
140 C
```

```
141 C     proton mass [kg]
142       DATA mp /1.672610356D-27/
143         REAL*8 mp
144 C     electron charge [C]
145       DATA e /1.6021927D-19/
146         REAL*8 e
147 C     light velocity [m/s]
148       DATA c /2.997925D+8/
149         REAL*8 c
150 C     unit conversion of energy from MeV to J
151       DATA mevtoj / 1.6021927D-13/
152         REAL*8 mevtoj
153 C     proton mass/electron mass ratio
154       DATA pmonem /1836.083D0/
155         REAL*8 pmonem
156 C
157 C
158       NAMELIST/particle/ a, z, en
159         INTEGER*4 a, z
160         REAL*8    en
161       NAMELIST/initcond/ geodalt, geodlat, geodlong, velrho,
162      >                   veltheta, velphi
163         REAL*8    geodalt, geodlat, geodlong, velrho,
164      >            veltheta, velphi
165       NAMELIST/magnfield/ kint, year
166         REAL*8    year
167       NAMELIST/meth/ imeth
168       NAMELIST/integr/ fintim, hmin, eps
169         REAL*8    fintim, hmin, eps
170 C
171 C
172 C
173 C     *************
174 C     FILE OPENINGS
175 C     *************
176 C
177 C     input/namelist file
178       open(unit=10, file='fstadiabinv.nml', status='old')
179       read(10, particle)
180       read(10, initcond)
181       read(10, magnfield)
182       read(10, meth)
183       read(10, integr)
184 C
185 C     output files
186       if (imeth.eq.1) then
187         open(unit=230, file='ckmu.dat', status='new')
188       else if (imeth.eq.2) then
189         open(unit=230, file='bsmu.dat', status='new')
190       else
191         write(6,*)'ERROR IN NAMELIST METH: IMETH MUST BE 1 OR 2'
192         STOP
193       endif
194 C
195 C     *************************
196 C     OPENING OF UNILIB LIBRARY
197 C     *************************
198 C
199       kunit = 6
200       kinit = 1
201       CALL UT990(kunit, kinit, ifail)
202 C
203 C     ********************************************
204 C     INITIALISATION OF GEOMAGNETIC FIELD IN UNILIB
205 C     ********************************************
206 C
207       kext=0
208       do i=1,10
209         param(i)=0.0
210       enddo
```

```
211         CALL UM510 (kint, year, lbint, kunit, ifail)
212         CALL UM520 (kext, amjd, param, lbext, kunit, ifail)
213         IF( ifail .LT. 0 )STOP
214   C
215   C
216   C     ****************************
217   C     THE PARTICLE AND ITS MOTION
218   C     ****************************
219   C
220   C     Kinetic energy [from MeV to J]
221         en = en * mevtoj
222   C
223   C     Charge of the particle
224         q = z*e
225         write(6,*) ' '
226         write(6,*) 'Charge of the particle            = ', q, ' C'
227   C
228   C     Rest mass of the particle
229         if(a.gt.0)then
230           massr = a*mp
231         else
232           massr = mp/pmonem
233         endif
234   C
235   C     Moment of the particle
236         p = DSQRT( (en+massr*c*c)**2 - (massr*c*c)**2 )
237        :                    /c
238   C
239   C     Velocity of the particle
240   C
241         v = (1/c)**2 + (massr/p)**2
242         v = 1/v
243         v = DSQRT(v)
244         write(6,*)'Velocity of the particle          = ', v/c, ' c'
245   C
246   C     Relativistic mass of the particle
247         gamma = 1/DSQRT(1-(v/c)**2)
248         massp = gamma*massr
249         write(6,*)'Relativistic mass of the particle = ', massp, ' kg'
250   C
251   C     ****************************
252   C     INITIAL VALUES OF THE MOTION
253   C     ****************************
254   C
255         mkgde.radius = re + geodalt
256         mkgde.colat  = 90.d0-geodlat
257         mkgde.elong  = geodlong
258   C
259         CALL UM530(mkgde,mb,ifail)
260         write (6,*) 'norme de Bmk= ', mb.dnrm
261   C     Geodetic to geocentric conversion
262         CALL UM536(mkgde,mkgeo)
263   C     Geocentric to cartesian conversion
264         CALL UT541(mkgeo,mkxyz)
265         vspher.rho   = velrho
266         vspher.theta = veltheta
267         vspher.phi   = velphi
268         vspher.dnrm  = DSQRT(vspher.rho**2 + vspher.theta**2 +
269        >                     vspher.phi**2)
270   C     Spherically coordinated vector converted into cartesian one
271         CALL UT542 (mkgeo,vspher,vcart)
272   C     conversion from km to m
273         ystart(1) = mkxyz.x*1000.D0
274         ystart(2) = mkxyz.y*1000.D0
275         ystart(3) = mkxyz.z*1000.D0
276         ystart(4) = v*(vcart.x/vspher.dnrm)
277         ystart(5) = v*(vcart.y/vspher.dnrm)
278         ystart(6) = v*(vcart.z/vspher.dnrm)
279   C
280   C     ****************************
```

```
281 C      TIME INTERVAL OF INTEGRATION
282 C      ****************************
283 C
284 C      initial time
285        t1 = 0.D0
286 C      final time
287        t2 = fintim
288 C
289 C      *****************
290 C      UNITS TO BE USED
291 C      *****************
292 C
293 C      lu : length unit, equals to mean earth radius (6371200 m)
294 C      tu : time unit, such that the constant velocity of particle
295 C           equals 1
296 C
297 C      mean earth radius is given in km as re in UNILIB
298        lu = re*1000.D0
299        vu = v
300 C      Time unit tu = length unit/velocity of the particle
301        tu = lu/vu
302 C
303 C      *****************
304 C      UNITS CONVERSION
305 C      *****************
306 C
307 C      from S.I. to (lu, tu) units
308 C
309        ystart(1) = ystart(1)/lu
310        ystart(2) = ystart(2)/lu
311        ystart(3) = ystart(3)/lu
312        ystart(4) = ystart(4)/vu
313        ystart(5) = ystart(5)/vu
314        ystart(6) = ystart(6)/vu
315        t1 = t1/tu
316        t2 = t2/tu
317        v = v/vu
318        c = c/vu
319 C
320 C      ************
321 C      INTEGRATION
322 C      ************
323 C
324 C
325        numagnmom = 230
326 C
327 C
328        h1=0.05D0
329 C
330        write(6,*)' '
331        if(imeth.eq.1)then
332          write(6,*)'-------------------------------------'
333          write(6,*)' NUMERICAL INTEGRATION OF THE MOTION'
334          write(6,*)'           EQUATIONS WITH A          '
335          write(6,*)' FIFTH ORDER RUNGE-KUTTA  (CASH-KARP) '
336          write(6,*)'-------------------------------------'
337          write(6,*)' '
338        else
339          write(6,*)'-------------------------------------'
340          write(6,*)' NUMERICAL INTEGRATION OF THE MOTION'
341          write(6,*)'          EQUATIONS WITH AN          '
342          write(6,*)'EXTRAPOLATION METHOD  (BULIRSCH-STOER) '
343          write(6,*)'-------------------------------------'
344          write(6,*)' '
345        endif
346 C
347        call mulgtime(ystart, t1, t2, eps, h1, hmin, numagnmom)
348 C
349        write(6,*)'REACHED TIME= ', t2*tu, 'SECONDS'
350 C
```

```
351         END
end
```

3 Listings des sous-routines

```
      SUBROUTINE bsstep(y,dydx,nv,x,htry,eps,yscal,hdid,hnext)
C
C     ******************************************************************
C
C     Burlisch-Stoer step with monitoring of local truncation error to
C     ensure accuracy and adjust stepsize.
C
C     Input are the dependent variable vector y(1:nv) and its derivative
C     dydx(1:nv) at the starting value of the independent variable x.
C     Also input are the stepsize the stepsize to be attempted htry,
C     the required accuracy eps, and the vector yscal(1:nv) against which
C     the error is scaled.
C
C     On output, y and x are replaced by their new values, hdid is the
C     stepsize that was actually accomplished, and hnext is the estimated
C     next stepsize.
C
C     Parameters:
C     NMAX: maximum value of nv;
C     KMAXX: maximum row number used in the extrapolation;
C     IMAX: next row number;
C     SAFE1, SAFE2: safety factors;
C     REDMIN, REDMAX: minimum and maximum factors used when a stepsize is
C                     reduced;
C     TINY: just to prevent division by zero;
C     1/SCALMX: maximum factor by which a stepsize can be increased.
C
C     ******************************************************************
C
      EXTERNAL mmid, pzextr
C
      INTEGER*4 nv
      REAL*8    eps, hdid, hnext, htry, x, dydx(nv), y(nv),
     >          yscal(nv)
C
      INTEGER*4 NMAX,KMAXX,IMAX
      REAL*8    SAFE1,SAFE2,REDMAX,REDMIN,TINY,SCALMX
      PARAMETER (NMAX=50,KMAXX=8,IMAX=KMAXX+1,SAFE1=.25Q0,SAFE2=.7Q0,
     >           REDMAX=1.Q-5,REDMIN=.7Q0,TINY=1.Q-30,SCALMX=.1Q0)
C
      INTEGER*4 i,iq,k,kk,km,kmax,kopt,nseq(IMAX)
      REAL*8 eps1,epsold,errmax,fact,h,red,scale,work,wrkmin,xest,
     >       xnew, a(IMAX),alf(KMAXX,KMAXX),err(KMAXX),yerr(NMAX),
     >       ysav(NMAX), yseq(NMAX)
      LOGICAL first,reduct
C
      SAVE a,alf,epsold,first,kmax,kopt,nseq,xnew
C
      DATA first/.true./,epsold/-1.Q0/
C
      DATA nseq /2,4,6,8,10,12,14,16,18/
C
C
C     Initialization for a new tolerance (eps)
C
      if(eps.ne.epsold)then
C
C       impossible values
        hnext=-1.D29
        xnew=-1.D29
C
        eps1=SAFE1*eps
C
C       work coefficients
        a(1)=nseq(1)+1
        do 11 k=1,KMAXX
          a(k+1)=a(k)+nseq(k+1)
   11   continue
C
        do 13 iq=2,KMAXX
```

```
71            do 12 k=1,iq-1
72              alf(k,iq)=eps1**((a(k+1)-a(iq+1))/((a(iq+1)-a(1)+1.D0)*
73         >                  (2*k+1)))
74      12      continue
75      13    continue
76    C
77          epsold=eps
78    C
79    C       optimal row number for convergence, kmax
80          do 14 kopt=2,KMAXX-1
81            if(a(kopt+1).gt.a(kopt)*alf(kopt-1,kopt))goto 1
82      14    continue
83      1     kmax=kopt
84    C
85        endif
86    C
87        h=htry
88    C     save the starting values
89        do 15 i=1,nv
90          ysav(i)=y(i)
91      15  continue
92    C
93    C     a new stepsize or a new integration: re-establish the order window
94    C
95        if(h.ne.hnext.or.x.ne.xnew)then
96          first=.true.
97          kopt=kmax
98        endif
99        reduct=.false.
100   C
101   C
102     2   do 17 k=1,kmax
103   C
104           xnew=x+h
105           if (h.eq.0.0) then
106             write(*,*) ' step size nul'
107             STOP
108           endif
109   C
110   C       modified midpoint step
111           call mmid(ysav,dydx,nv,x,h,nseq(k),yseq)
112   C       squared, since error series is even:
113           xest=(h/nseq(k))**2
114   C       polynomial extrapolation
115           call pzextr(k,xest,yseq,y,yerr,nv)
116   C
117           if(k.ne.1)then
118             errmax=TINY
119             do 16 i=1,nv
120               if(yscal(i).ne.0.)then
121                 errmax=dmax1(errmax,DABS(yerr(i)/yscal(i)))
122               endif
123     16        continue
124               errmax=errmax/eps
125               km=k-1
126               err(km)=(errmax/SAFE1)**(1.D0/(2*km+1))
127             endif
128   C
129           if(k.ne.1.and.(k.ge.kopt-1.or.first))then
130             if(errmax.lt.1.D0)goto 4
131             if(k.eq.kmax.or.k.eq.kopt+1)then
132               red=SAFE2/err(km)
133               goto 3
134             else if(k.eq.kopt)then
135               if(alf(kopt-1,kopt).lt.err(km))then
136                 red=1.D0/err(km)
137                 goto 3
138               endif
139             else if(kopt.eq.kmax)then
140               if(alf(km,kmax-1).lt.err(km))then
```

```
141                 red=alf(km,kmax-1)*SAFE2/err(km)
142                 goto 3
143               endif
144             else if(alf(km,kopt).lt.err(km))then
145               red=alf(km,kopt-1)/err(km)
146               goto 3
147             endif
148           endif
149    17   continue
150  C
151  C      reduce stepsize
152  C
153    3    red=dmin1(red,REDMIN)
154         red=dmax1(red,REDMAX)
155         h=h*red
156         reduct=.true.
157         goto 2
158  C
159  C      convergence is reached
160  C
161    4    x=xnew
162         hdid=h
163         first=.false.
164  C
165         wrkmin=1.D35
166         do 18 kk=1,km
167           fact=dmax1(err(kk),SCALMX)
168           work=fact*a(kk+1)
169           if(work.lt.wrkmin)then
170             scale=fact
171             wrkmin=work
172             kopt=kk+1
173           endif
174    18   continue
175  C
176         hnext=h/scale
177  C
178         if(kopt.ge.k.and.kopt.ne.kmax.and..not.reduct)then
179           fact=dmax1(scale/alf(kopt-1,kopt),SCALMX)
180           if(a(kopt+1)*fact.le.wrkmin)then
181             hnext=h/fact
182             kopt=kopt+1
183           endif
184         endif
185  C
186         RETURN
187         END
end
```

```
1         subroutine derivs(x, y, dydx)
2
3   C     ****************************************************************
4   C     y being the 6-dim vector (position, velocity of the particle),
5   C     dydx(y) is dy/dt for a constant magnetic field.
6   C     UNITS OF INPUTS AND OUTPUT : (rl,v) SYSTEM
7   C     ****************************************************************
8
9         real*8    x, y(6), dydx(6)
10        integer i
11        real*8    qm, conv
12        real*8 q, massp, bz
13        real*8 rl, vorth
14        real*8 lu, tu, vu
15
16        common/prtcle/q, massp
17        common/magnfld/ bz
18        common/unitsys/ lu, tu, vu
19
20  C     code
21
22        qm = q/massp
23
24        do i=1,3
25          dydx(i) = y(i+3)
26        enddo
27
28  C     SI
29        dydx(4) = qm * (y(5)*vu*bz)
30        dydx(5) = qm * (- y(4)*vu*bz)
31        dydx(6) = 0.D0
32  C     (rl,v)
33        conv = tu/vu
34        dydx(4) = dydx(4)*conv
35        dydx(5) = dydx(5)*conv
36        dydx(6) = dydx(6)*conv
37
38        return
39        end
end
```

```
      SUBROUTINE INTERPOL(TIME, ALT, LATIT, LONGIT,
     >                    tmp, hmin, latmp, longmp)
C
C     ******************************************************
C           Parabolic interpolation based on points
C                 (TIME(i), ALT(i))(i=1,3),
C        aimed at finding a point of minimum altitude.
C     ******************************************************
C
C
      REAL*8    TIME(3), ALT(3), LATIT(3), LONGIT(3),
     >          tmp, hmin, latmp, longmp
C
      REAL*8    delta, A, B
C
C
      tmp = 0.D0
      hmin = 0.D0
      latmp = 0.D0
      longmp = 0.D0
C
      delta = (TIME(3)-TIME(2))*(TIME(1)-TIME(2))**2
     >      - (TIME(1)-TIME(2))*(TIME(3)-TIME(2))**2
C
      if (delta.eq.0.D0) then
        write(6,*) 'ERROR (DELTA IN SUBR INTERPOL IS 0)'
        RETURN
      endif
C
C     *****************
C     TIME AND ALTITUDE
C     *****************
C
      A = (  (TIME(3)-TIME(2))*(ALT(1)-ALT(2))
     >     - (TIME(1)-TIME(2))*(ALT(3)-ALT(2))  )
     >    / delta
      if(A.eq.0.D0)then
        if(ALT(1).eq.ALT(2).and.ALT(1).eq.ALT(3))then
          hmin = ALT(1)
          tmp = TIME(2)
        else
          write(6,*) 'ERROR A=0 IN INTERPOL'
          RETURN
        endif
      endif
      B = 0.5D0*( (ALT(1)-ALT(2))/(TIME(1)-TIME(2))
     >          - A*(TIME(1)-TIME(2))
     >          + (ALT(3)-ALT(2))/(TIME(3)-TIME(2))
     >          - A*(TIME(3)-TIME(2)) )
C
      if(tmp.eq.0) tmp = TIME(2)-(B/(2*A))
      if(hmin.eq.0) hmin =  - B*B/(4*A) + ALT(2)
C
C     ********
C     LATITUDE
C     ********
C
      A = (  (TIME(3)-TIME(2))*(LATIT(1)-LATIT(2))
     >     - (TIME(1)-TIME(2))*(LATIT(3)-LATIT(2))  )
     >    / delta
      if(A.eq.0.D0)then
        if(LATIT(1).eq.LATIT(2).and.LATIT(1).eq.LATIT(3))then
          latmp = LATIT(1)
        else
          write(6,*) 'ERROR A=0 IN INTERPOL'
          RETURN
        endif
      endif
      B = 0.5D0*( (LATIT(1)-LATIT(2))/(TIME(1)-TIME(2))
     >          - A*(TIME(1)-TIME(2))
```

```
71          >             + (LATIT(3)-LATIT(2))/(TIME(3)-TIME(2))
72          >             - A*(TIME(3)-TIME(2)) )
73    C
74           if(latmp.eq.0)
75          >  latmp = A*(tmp-TIME(2))**2 + B*(tmp-TIME(2)) + LATIT(2)
76    C
77    C      *********
78    C      LONGITUDE
79    C      *********
80    C
81           A = (  (TIME(3)-TIME(2))*(LONGIT(1)-LONGIT(2))
82          >       - (TIME(1)-TIME(2))*(LONGIT(3)-LONGIT(2))  )
83          >    / delta
84           if(A.eq.0.D0)then
85             if(LONGIT(1).eq.LONGIT(2).and.LONGIT(1).eq.LONGIT(3))then
86               latmp = LONGIT(1)
87             else
88               write(6,*) 'ERROR A=0 IN INTERPOL'
89               RETURN
90             endif
91           endif
92           B = 0.5D0*( (LONGIT(1)-LONGIT(2))/(TIME(1)-TIME(2))
93          >             - A*(TIME(1)-TIME(2))
94          >             + (LONGIT(3)-LONGIT(2))/(TIME(3)-TIME(2))
95          >             - A*(TIME(3)-TIME(2)) )
96    C
97           if(longmp.eq.0)
98          >  longmp = A*(tmp-TIME(2))**2 + B*(tmp-TIME(2)) + LONGIT(2)
99    C
100   C
101          RETURN
102          END
end
```

```
      SUBROUTINE INTERPOLMU (TIME, ALT, LATIT, LONGIT, BM3, MU3,
     >                       tmp, hmin, latmp, longmp, bm, mu)
C
C     ******************************************************
C           Parabolic interpolation based on points
C                  (TIME(i), ALT(i))(i=1,3),
C         aimed at finding a point of minimum altitude.
C         The interpolation also concerns the latitude
C         and longitude, the magnetic intensity and the
C                       magnetic moment.
C     ******************************************************
C
C
      REAL*8    TIME(3), ALT(3), LATIT(3), LONGIT(3), MU3(3),
     >          BM3(3), tmp, hmin, latmp, longmp, mu, bm
C
      REAL*8    delta, A, B
C
C
      tmp = 0.D0
      hmin = 0.D0
      latmp = 0.D0
      longmp = 0.D0
      mu = 0.D0
      bm=0.D0
C
      delta = (TIME(3)-TIME(2))*(TIME(1)-TIME(2))**2
     >        - (TIME(1)-TIME(2))*(TIME(3)-TIME(2))**2
C
      if (delta.eq.0.D0) then
        write(6,*) 'ERROR (DELTA IN SUBR INTERPOL IS 0)'
        RETURN
      endif
C
C     *****************
C     TIME AND ALTITUDE
C     *****************
C
      A = (  (TIME(3)-TIME(2))*(ALT(1)-ALT(2))
     >       - (TIME(1)-TIME(2))*(ALT(3)-ALT(2))  )
     >    / delta
      if(A.eq.0.D0)then
        if(ALT(1).eq.ALT(2).and.ALT(1).eq.ALT(3))then
          hmin = ALT(1)
          tmp = TIME(2)
        else
          write(6,*) 'ERROR A=0 IN INTERPOL'
          RETURN
        endif
      endif
      B = 0.5D0*( (ALT(1)-ALT(2))/(TIME(1)-TIME(2))
     >            - A*(TIME(1)-TIME(2))
     >            + (ALT(3)-ALT(2))/(TIME(3)-TIME(2))
     >            - A*(TIME(3)-TIME(2)) )
C
      if(tmp.eq.0) tmp = TIME(2)-(B/(2*A))
      if(hmin.eq.0) hmin =  - B*B/(4*A) + ALT(2)
C
C     ********
C     LATITUDE
C     ********
C
      A = (  (TIME(3)-TIME(2))*(LATIT(1)-LATIT(2))
     >       - (TIME(1)-TIME(2))*(LATIT(3)-LATIT(2))  )
     >    / delta
      if(A.eq.0.D0)then
        if(LATIT(1).eq.LATIT(2).and.LATIT(1).eq.LATIT(3))then
          latmp = LATIT(1)
        else
          write(6,*) 'ERROR A=0 IN INTERPOL'
```

```
71            RETURN
72          endif
73        endif
74        B = 0.5D0*( (LATIT(1)-LATIT(2))/(TIME(1)-TIME(2))
75       >            - A*(TIME(1)-TIME(2))
76       >            + (LATIT(3)-LATIT(2))/(TIME(3)-TIME(2))
77       >            - A*(TIME(3)-TIME(2)) )
78  C
79        if(latmp.eq.0)
80       >  latmp = A*(tmp-TIME(2))**2 + B*(tmp-TIME(2)) + LATIT(2)
81  C
82  C     *********
83  C     LONGITUDE
84  C     *********
85  C
86        A = (  (TIME(3)-TIME(2))*(LONGIT(1)-LONGIT(2))
87       >       - (TIME(1)-TIME(2))*(LONGIT(3)-LONGIT(2))  )
88       >    / delta
89        if(A.eq.0.D0)then
90          if(LONGIT(1).eq.LONGIT(2).and.LONGIT(1).eq.LONGIT(3))then
91            latmp = LONGIT(1)
92          else
93            write(6,*) 'ERROR A=0 IN INTERPOL'
94            RETURN
95          endif
96        endif
97        B = 0.5D0*( (LONGIT(1)-LONGIT(2))/(TIME(1)-TIME(2))
98       >            - A*(TIME(1)-TIME(2))
99       >            + (LONGIT(3)-LONGIT(2))/(TIME(3)-TIME(2))
100      >            - A*(TIME(3)-TIME(2)) )
101 C
102       if(longmp.eq.0)
103      >  longmp = A*(tmp-TIME(2))**2 + B*(tmp-TIME(2)) + LONGIT(2)
104 C
105 C     ***************
106 C     MAGNETIC MOMENT
107 C     ***************
108 C
109       A = (  (TIME(3)-TIME(2))*(MU3(1)-MU3(2))
110      >       - (TIME(1)-TIME(2))*(MU3(3)-MU3(2))  )
111      >    / delta
112       if(A.eq.0.D0)then
113         if(MU3(1).eq.MU3(2).and.MU3(1).eq.MU3(3))then
114           mu = MU3(1)
115         else
116           write(6,*) 'ERROR A=0 IN INTERPOL'
117           RETURN
118         endif
119       endif
120       B = 0.5D0*( (MU3(1)-MU3(2))/(TIME(1)-TIME(2))
121      >            - A*(TIME(1)-TIME(2))
122      >            + (MU3(3)-MU3(2))/(TIME(3)-TIME(2))
123      >            - A*(TIME(3)-TIME(2)) )
124 C
125       if(mu.eq.0)
126      >  mu = A*(tmp-TIME(2))**2 + B*(tmp-TIME(2)) + MU3(2)
127 C
128 C     ************************
129 C     MAGNETIC FIELD INTENSITY
130 C     ************************
131 C
132       A = (  (TIME(3)-TIME(2))*(BM3(1)-BM3(2))
133      >       - (TIME(1)-TIME(2))*(BM3(3)-BM3(2))  )
134      >    / delta
135       if(A.eq.0.D0)then
136         if(BM3(1).eq.BM3(2).and.BM3(1).eq.BM3(3))then
137           bm = BM3(1)
138         else
139           write(6,*) 'ERROR A=0 IN INTERPOL'
140           RETURN
```

```
141           endif
142         endif
143         B = 0.5D0*( (BM3(1)-BM3(2))/(TIME(1)-TIME(2))
144        >            - A*(TIME(1)-TIME(2))
145        >            + (BM3(3)-BM3(2))/(TIME(3)-TIME(2))
146        >            - A*(TIME(3)-TIME(2)) )
147   C
148         if(bm.eq.0)
149        >  bm = A*(tmp-TIME(2))**2 + B*(tmp-TIME(2)) + BM3(2)
150   C
151   C
152         RETURN
153         END
end
```

```
      SUBROUTINE mmid(y,dydx,nvar,xs,htot,nstep,yout)
C
C     ***********************************************************
C     Modified midpoint step.
C     Dependent variable vector yvar(1:nvar) and its derivative
C     vector dydx(1:nvar) are input at xs.  Also input is htot,
C     the total step to be made, and nstep, the number of substeps
C     to be used.
C     The output is returned as yout(1:nvar).
C     ***********************************************************
C
      EXTERNAL  derivs
C
      INTEGER*4 nstep,nvar
      REAL*8    htot,xs,dydx(nvar),y(nvar),yout(nvar)
C
      INTEGER*4 NMAX
      PARAMETER (NMAX=50)

      INTEGER*4 i,n
      REAL*8    h,h2,swap,x,ym(NMAX),yn(NMAX)
C
C
      h=htot/nstep
      do 11 i=1,nvar
        ym(i)=y(i)
        yn(i)=y(i)+h*dydx(i)
   11 continue
C
      x=xs+h
      call derivs(yn,yout)
      h2=2.D0*h
      do 13 n=2,nstep
        do 12 i=1,nvar
          swap=ym(i)+h2*yout(i)
          ym(i)=yn(i)
          yn(i)=swap
   12   continue
        x=x+h
        call derivs(yn,yout)
   13 continue
      do 14 i=1,nvar
        yout(i)=0.5D0*(ym(i)+yn(i)+h*yout(i))
   14 continue
C
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software )?".
end
```

```
      SUBROUTINE MULGTIME (ystart, t1, t2, eps, h1, hmin,
     >                     numagnmom)
C
C
C     ****************************************************************
C
C      Called by main programs FSTADDIP and FSTADIABINV, this
C      subroutine carries out ODE integration from time t1 to time
C      t2, for initial  conditions ystart and with a Runge-Kutta
C      or an extrapolation method: RK-Fehlberg-Cash-Karp (CK)
C      or Bulirsch-Stoer (BS), with a precision defined by eps.
C      The main goal is to evaluate the invariant magnetic moment at
C      mirror points for a long interval of time.
C      The output file ('ckmu.dat' or 'bsmu.dat'), of unit
C      numagnmom, contains the definition of the mirror points
C      (integration times, altitudes, latitudes, longitudes,
C      magnetic intensities) and the related values of magnetic
C      moment and bouncing period.
C
C     ****************************************************************
C
C
      EXTERNAL derivs, rkqs, bsstep, testhmin
C     (and routines of UNILIB)
C
      INCLUDE 'structure.h'
C
      INTEGER*4 numagnmom
      REAL*8    ystart(6), t1, t2, eps, h1, hmin
C
      INTEGER*4 i, ifail, mpok, init, nmp
      REAL*8    x, h, y(6), dydx(6), yscal(6), hdid, hnext,
     >          geodalt, geodlat, geodlong,
     >          vi2, sa2, mu, yminus1(6), yminus2(6), TIME(3),
     >          ALT(3),LATIT(3), LONGIT(3), VEL(3,3), tmp, hmp,
     >          latmp, longmp, tminus1, tminus2, bm, tb(2),
     >          veloc
C
      COMMON/particule/ q, massp, massr
        REAL*8  q, massp, massr
      COMMON/methode/imeth
        INTEGER*4 imeth
      COMMON/lightvel/ c
        REAL*8  c
      COMMON/units/ lu, tu, vu
        REAL*8  lu, tu, vu
      COMMON/mfmodel/ kint
        INTEGER*4 kint
C
      COMMON /UC160/ pi, deg, re, gmagmo, eclipt, geoid, uma
        REAL*8  pi, deg, re
        REAL*8  gmagmo
        REAL*8  eclipt, geoid(3), uma(30)
C
C
      x = t1
      h = h1
C
C
C     ******************
C     INITIAL CONDITIONS
C     ******************
C
C
      do i=1,6
        y(i) = ystart(i)
        yminus2(i) = y(i)
      enddo
      tminus2 = x
C
```

```
C
C     ***************
C     ODE INTEGRATION
C     ***************
C
C
C     dy/dx = f(x,y)
      CALL derivs (y, dydx)
C     scaling of the error
      do i=1,6
        yscal(i) = DABS(y(i)) + DABS(h*dydx(i))
      enddo
C     Don't overtake the final time !
      if((x+h-t1)*(x+h-t2).gt.0) h = t2-x
C
C     ================================================
C     First step, aimed at getting two starting points:
C     yminus2 and yminus1
C     ================================================
C
      if (imeth.eq.1)then
C
C         =======
C         CK step
C         =======
C
        call rkqs(y, dydx, x, h, eps, yscal, hdid, hnext)
C
      else if (imeth.eq.2) then
C
C         =======
C         BS step
C         =======
C
        call bsstep(y, dydx, 6, x, h, eps, yscal, hdid, hnext)
C
      endif
C
      tminus1 = x
      do i=1,6
        yminus1(i)=y(i)
      enddo
C
C     nmp: aimed at avoiding a false first mirror point
      nmp=0
      init=0
C     tb: aimed at computing the bouncing period
      tb(1) = 0.D0
      tb(2) = 0.D0
C
C     ==========
C     Next steps
C     ==========
C
    1 if ((x-t2)*(t2-t1) .lt. 0.D0) then
C       dy/dx = f(x,y)
        CALL derivs (y, dydx)
C       scaling of the error
        do i=1,6
          yscal(i) = DABS(y(i)) + DABS(h*dydx(i))
        enddo
C       Don't overtake the final time !
        if((x+h-t1)*(x+h-t2).gt.0) h = t2-x
C
        if (imeth.eq.1)then
C
C         =======
C         CK step
C         =======
C
```

```
141             call rkqs(y, dydx, x, h, eps, yscal, hdid, hnext)
142   C
143           else if (imeth.eq.2) then
144   C
145   C         =======
146   C         BS step
147   C         =======
148   C
149             call bsstep(y, dydx, 6, x, h, eps, yscal, hdid, hnext)
150   C
151           endif
152   C
153   C         -------------------------
154   C           Mirror points research
155   C         -------------------------
156   C
157           call testhmin (x, y, tminus1, yminus1, tminus2,
158          >                yminus2, init, TIME, ALT, LATIT,
159          >                LONGIT, VEL, mpok, tmp, hmp, latmp,
160          >                longmp, bm, mu)
161   C
162           if (mpok .eq. 1) then
163   C
164   C         -------------------------------
165   C          A mirror point has been found
166   C         -------------------------------
167   C
168   C           WRITES IN OUTPUT FILE ('CKMU.DAT' OR 'BSMU.DAT'):
169   C            - TIME OF MIRROR POINT [SECONDS](tmp)
170   C            - ALTITUDE OF MIRROR POINT [KILOMETERS] (converted hmp)
171   C            - LATITUDE OF MIRROR POINT (latmp)
172   C            - LONGITUDE OF MIRROR POINT (longmp)
173   C            - MAGNETIC FIELD INTENSITY AT MIRROR POINT [GAUSS] (bm)
174   C            - MAGNETIC MOMENT AT MIRROR POINT [SI] (mu)
175   C            - BOUNCING PERIOD [SECONDS]
176   C            - VELOCITY AT LAST POINT
177   C
178             Veloc=DSQRT(y(4)**2+y(5)**2+y(6)**2)
179             if(nmp.gt.0) then
180               if (tb(1).gt.0) then
181                 write(numagnmom,*)  tmp, hmp*lu/1000.D0, latmp,
182          >                          longmp, bm, mu, tmp-tb(1), veloc
183                 write(numagnmom,*) ' '
184               else
185                 write(numagnmom,*)  tmp, hmp*lu/1000.D0, latmp,
186          >                          longmp, bm, mu, veloc
187                 write(numagnmom,*) ' '
188               endif
189               tb(1) = tb(2)
190               tb(2) = tmp
191             endif
192             nmp=nmp+1
193   C
194           endif
195   C
196           tminus2=tminus1
197           tminus1=x
198           do i=1,6
199             yminus2(i)=yminus1(i)
200             yminus1(i)=y(i)
201           enddo
202   C
203           if(DABS(hnext).lt.hmin)then
204             write(6,*)'stepsize smaller than minimum in odeint'
205             return
206           endif
207   C
208   C       Next step size
209   C
210           h = hnext
```

```
211             go to 1
212   C
213           endif
214   C
215           RETURN
216           END
end
```

```
      SUBROUTINE odeint2 (ystart, x1, x2, eps, h1, hmin, hmax, nu, nut)
C
C     ****************************************************
C     ODE integration from time x1 to time x2, with a RK
C     or an extrapolation method (RK-Fehlberg-Cash-Karp or
C     Bulirsch-Stoer, noted CK and BS resp.)
C     ****************************************************
C
C
      EXTERNAL derivs, rkqs, bsstep
C
      INTEGER*4 nu, nut
      REAL*8    eps, h1, hmin, hmax, x1, x2, ystart(6)
C
      INTEGER*4 i, j, nb, nto, nstep, cpu
      REAL*8    h, hdid, hnext, x, dydx(6), y(6), yscal(6),
     >          xref, rnt, second, sum1, sum2
C
      COMMON /methode/imeth, nbgraph
        INTEGER*4 imeth, nbgraph
      COMMON /snt/subnt
        INTEGER*4 subnt
      COMMON /gyrvit/ rl, vorth
        REAL*8 rl, vorth
      COMMON /unitsys/ lu, tu, vu
        REAL*8 lu, tu, vu
      COMMON /pi/ pi
        REAL*8 pi
C
C
C
      x = x1
      h = DSIGN(h1, x2-x1)
      do i=1,6
        y(i) = ystart(i)
      enddo
      nstep=0
C
C
      xref = subnt*2.D0*pi*rl/vorth
      xref = DABS(xref)
C     unity conversion
      xref = xref/tu
C
      nto=subnt
C
C     Initialize computation time
      second=0.D0
C
    1 call lib$init_timer
C
      if((x-x2)*(x2-x1).lt.0.)then
        nstep=nstep+1
C       dy/dx = f(x,y)
        call derivs(x, y, dydx)
C       scaling of the error
        do i=1,6
          yscal(i) = DABS(y(i)) + DABS(h*dydx(i))
        enddo
C       Don't overtake the final time !
        if((x+h-x1)*(x+h-x2).gt.0) h = x2-x
C
        if (imeth.eq.2)then
C
C         -------
C         CK step
C         -------
C
          call rkqs(y, dydx, x, h, eps, yscal, hdid, hnext)
C
```

```
71          else if (imeth.eq.3) then
72  C
73  C         -------
74  C         BS step
75  C         -------
76  C
77            call bsstep(y, dydx, 6, x, h, eps, yscal, hdid, hnext)
78  C
79          endif
80  C
81  C       Update computation time
82          call lib$stat_timer(2,cpu)
83          second= second + cpu*1.D-2
84  C
85          if(DABS(hnext).lt.hmin)then
86            write(6,*)'stepsize smaller than minimum in odeint'
87            return
88          endif
89  C
90          if (x.ge.xref) then
91            nb=nu-1
92            nb=nb/10
93            nb=nb-1
94            write(6,*) nb, '   nb tours= ',nto
95            rnt = nto
96            write(100,*)DLOG10(rnt)
97            nto = nto+subnt
98            xref = xref+(DABS(2.D0*pi*subnt*r1/vorth)/tu)
99  C
100 C         As subnt more gyrations were carried out, record
101 C
102 C         ------------------------------------------------
103 C         (1) the current relative precisions on r1 and v
104 C         ------------------------------------------------
105 C
106           sum1=0.D0
107           sum2=0.D0
108           do k=1,6
109             if (k.le.2) then
110               sum1= sum1+(y(k)*y(k))
111             else if (k.ge.4) then
112               sum2= sum2+(y(k)*y(k))
113             endif
114           enddo
115           sum1=DSQRT(sum1)
116           sum2=DSQRT(sum2)
117           sum1=DABS(sum1-1.D0)
118           sum2=DABS(sum2-1.D0)
119           write(nu,*)DLOG10(sum1)
120           write(nu+1,*)DLOG10(sum2)
121 C
122 C         --------------------------------
123 C         (2) the current computation time
124 C         --------------------------------
125 C
126           write(nut,*)DLOG10(second)
127 C
128         endif
129 C
130 C       Next step size
131         h = hnext
132         go to 1
133       endif
134 C
135 C
136       RETURN
137       END
end
```

```
1         SUBROUTINE parabint(time, ord)
2   C
3   C
4   C     ************************************************************
5   C      Parabolic interpolation and evaluation of the minimum
6   C     ************************************************************
7   C
8   C
9         REAL*8 time(4), ord(4)
10  C
11        REAL*8 delta, A, B, hmin, tmp
12  C
13  C
14  C
15        hmin=0.D0
16        tmp = 0.D0
17  C
18        delta = (TIME(3)-TIME(2))*(TIME(1)-TIME(2))**2
19       >        - (TIME(1)-TIME(2))*(TIME(3)-TIME(2))**2
20  C
21        if (delta.eq.0.D0) then
22          write(6,*) 'ERROR (DELTA IN SUBR INTERPOL IS 0)'
23          RETURN
24        endif
25  C
26        A = (  (TIME(3)-TIME(2))*(ORD(1)-ORD(2))
27       >       - (TIME(1)-TIME(2))*(ORD(3)-ORD(2))  )
28       >    / delta
29        if(A.eq.0.D0)then
30          if(ORD(1).eq.ORD(2).and.ORD(1).eq.ORD(3))then
31            hmin = ORD(1)
32            tmp = TIME(2)
33          else
34            write(6,*) 'ERROR A=0 IN INTERPOL'
35            RETURN
36          endif
37        endif
38        B = 0.5D0*( (ORD(1)-ORD(2))/(TIME(1)-TIME(2))
39       >            - A*(TIME(1)-TIME(2))
40       >            + (ORD(3)-ORD(2))/(TIME(3)-TIME(2))
41       >            - A*(TIME(3)-TIME(2)) )
42  C
43        if(tmp.eq.0) tmp = TIME(2)-(B/(2*A))
44        if(hmin.eq.0) hmin =  - B*B/(4*A) + ORD(2)
45  C
46        time(4) = tmp
47        ord(4)  = hmin
48  C
49  C
50  C
51        RETURN
52        END
end
```

```
      SUBROUTINE polint(xa,ya,n,x,y,dy)
C
C     **************************
C      Polynomial interpolation
C     **************************
C
      INTEGER*4 n,NMAX
      REAL*8 dy,x,y,xa(n),ya(n)
      PARAMETER (NMAX=10)
      INTEGER*4 i,m,ns
      REAL*8 den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)
      ns=1
      dif=abs(x-xa(1))
      do 11 i=1,n
        dift=abs(x-xa(i))
        if (dift.lt.dif) then
          ns=i
          dif=dift
        endif
        c(i)=ya(i)
        d(i)=ya(i)
11    continue
      y=ya(ns)
      ns=ns-1
      do 13 m=1,n-1
        do 12 i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
          w=c(i+1)-d(i)
          den=ho-hp
          if(den.eq.0.)pause 'failure in polint'
          den=w/den
          d(i)=hp*den
          c(i)=ho*den
12      continue
        if (2*ns.lt.n-m)then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
        y=y+dy
13    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software
end
```

```
1         SUBROUTINE pzextr(iest,xest,yest,yz,dy,nv)
2   C
3   C     ****************************************************************
4   C     Polynomial extrapolation, in order to evaluate nv functions
5   C     at x=0 by fitting a polynomial to a sequence of estimates with
6   C     progressively smaller values x=xest, and corresponding function
7   C     vectors yest(1:nv).  This call is number iest in the sequence of
8   C     calls.
9   C     Extrapolated function values are output as yz(1:nv), and their
10  C     estimated error is output as dy(1:nv).
11  C
12  C     Parameters:
13  C     IMAX: maximum expected value of iest;
14  C     NMAX: maximum expected value of nv.
15  C     ****************************************************************
16  C
17        INTEGER*4 iest,nv
18        REAL*8    xest, dy(nv), yest(nv), yz(nv)
19  C
20        INTEGER*4 IMAX,NMAX
21        PARAMETER (IMAX=13,NMAX=50)
22  C
23        INTEGER*4 j,k1
24        REAL*8 delta,f1,f2,q,d(NMAX),qcol(NMAX,IMAX),x(IMAX)
25  C
26        SAVE qcol,x
27  C
28        x(iest)=xest
29        do 11 j=1,nv
30          dy(j)=yest(j)
31          yz(j)=yest(j)
32     11 continue
33        if(iest.eq.1) then
34          do 12 j=1,nv
35            qcol(j,1)=yest(j)
36     12     continue
37        else
38          do 13 j=1,nv
39            d(j)=yest(j)
40     13     continue
41          do 15 k1=1,iest-1
42            delta=1.D0/(x(iest-k1)-xest)
43            f1=xest*delta
44            f2=x(iest-k1)*delta
45            do 14 j=1,nv
46              q=qcol(j,k1)
47              qcol(j,k1)=dy(j)
48              delta=d(j)-q
49              dy(j)=f1*delta
50              d(j)=f2*delta
51              yz(j)=yz(j)+dy(j)
52     14       continue
53     15     continue
54          do 16 j=1,nv
55            qcol(j,iest)=dy(j)
56     16     continue
57        endif
58        return
59        END
60  C   (C) Copr. 1986-92 Numerical Recipes Software )?".
end
```

```
      subroutine rkck(y,dydx,x,h,yout,yerr)
C
C     ***************************************************************
C     Given values for 6 variables y and their derivatives dydx
C     known at x, uses the fifth-order Cash-Karp Runge-Kutta method
C     to advance the solution over an interval and returns the
C     incremented variables as yout.
C     Also returns an estimate of the local truncation error in
C     yout using the embedded fouth-order method.  The user
C     supplies the subroutine derivs(x, y, dydx) which returns
C     derivatives dydx at x.
C     ***************************************************************
C
      EXTERNAL derivs
C
      REAL*8 h, x, dydx(6), y(6), yerr(6), yout(6)
C
      INTEGER*4 i
      REAL*8 ak2(6), ak3(6), ak4(6), ak5(6), ak6(6),
     :     ytemp(6), a2, a3, a4, a5, a6, b21, b31,
     :     b32, b41, b42, b43, b51, b52, b53, b54,
     :     b61, b62, b63, b64, b65, c1, c3, c4, c6,
     :     dc1, dc3, dc4, dc5, dc6
      PARAMETER(a2=.2D0, a3=.3D0, a4=.6D0, a5=1.D0, a6=.875D0,
     :     b21=.2D0, b31=3.D0/40.D0, b32=9.D0/40.D0, b41=.3D0,
     :     b42=-.9D0, b43=1.2D0, b51=-11.D0/54.D0, b52=2.5D0,
     :     b53=-70.D0/27.D0, b54=35.D0/27.D0, b61=1631.D0/55296.D0,
     :     b62=175.D0/512.D0, b63=575.D0/13824.D0,
     :     b64=44275.D0/110592.D0,
     :     b65=253.D0/4096.D0, c1=37.D0/378.D0, c3=250.D0/621.D0,
     :     c4=125.D0/594.D0, c6=512.D0/1771.D0,
     :     dc1=c1-2825.D0/27648.D0,
     :     dc3=c3-18575.D0/48384.D0, dc4=c4-13525.D0/55296.D0,
     :     dc5=-277.D0/14336.D0, dc6=c6-.25D0)
C
C
C     first step
      do i=1,6
        ytemp(i) = y(i) + b21*h*dydx(i)
      enddo
C     second step
      call derivs (ytemp, ak2)
      do i=1,6
        ytemp(i) = y(i) + h*(b31*dydx(i) + b32*ak2(i))
      enddo
C     third step
      call derivs(ytemp, ak3)
      do i=1,6
        ytemp(i) = y(i) + h*( b41*dydx(i) + b42*ak2(i) +
     :                        b43*ak3(i) )
      enddo
C     fourth step
      call derivs(ytemp, ak4)
      do i=1,6
        ytemp(i) = y(i) + h*( b51*dydx(i) + b52*ak2(i) +
     :                        b53*ak3(i) + b54*ak4(i) )
      enddo
C     fifth step
      call derivs(ytemp, ak5)
      do i=1,6
        ytemp(i) = y(i) + h*( b61*dydx(i) + b62*ak2(i) +
     :                        b63*ak3(i) + b64*ak4(i) +
     :                        b65*ak5(i) )
      enddo
C     sixth step
      call derivs(ytemp, ak6)
      do i=1,6
        yout(i) = y(i) + h*( c1*dydx(i) + c3*ak3(i) +
     :                       c4*ak4(i) + c6*ak6(i) )
      enddo
```

```
71  C
72  C     estimate error as difference between fourth and fifth order methods
73        do i=1,6
74          yerr(i)= h*( dc1*dydx(i) + dc3*ak3(i) + dc4*ak4(i) +
75       :               dc5*ak5(i) + dc6*ak6(i) )
76        enddo
77  C
78        return
79        end
end
```

```
      subroutine rkqs(y,dydx,x,htry,eps,yscal,hdid,hnext)
C
C     ***************************************************
C     Runge-Kutta-Cash-Karp step with monitoring of local
C     truncation error to ensure accuracy and adjust stepsize.
C     ***************************************************
C
      EXTERNAL rkck
C
      real*8 y(6), dydx(6), x, htry, eps, yscal(6), hdid, hnext
      integer*4 i
      real*8 errmax, h, yerr(6), ytemp(6), safety, pgrow, pshrnk
C
      parameter (safety=0.9D0, pgrow=-0.2D0, pshrnk=-0.25D0)
C
      h = htry
 1    call rkck(y, dydx, x, h, ytemp, yerr)
C
      errmax = 0.0D0
      do i=1,6
        if(yscal(i).ne.0)then
          errmax = DMAX1(errmax, DABS(yerr(i)/yscal(i)))
        endif
      enddo
      errmax = errmax/eps
      if(errmax.gt.1)then
        h = safety*h*(errmax**pshrnk)
        go to 1
      else
        hnext = safety*h*(errmax**pgrow)
        hdid = h
        x = x+h
        do i=1,6
          y(i) = ytemp(i)
        enddo
      endif
C
      return
      end
end
```

```
      SUBROUTINE TESTHMIN (x, y, tminus1, yminus1, tminus2,
     >                     yminus2, init, TIME, ALT, LATIT,
     >                     LONGIT, VEL, mpok, tmp, hmp, latmp,
     >                     longmp, bm, mu)
C
C     Related to FSTADIABINV
C
C     ************************************************************
C
C     Called at each numerical integration step by subroutine
C     MULGTIME (called itself by the program FSTADIABINV),
C     the subroutine TESTHMIN tests if the three points y
C     (at integration time x), yminus1 (at integration time
C     tminus1) and yminus2 (at integration time tminus2)
C     complies with the necessary condition for the parabolic
C     interpolation (related to the i.t.) of a mirror point.
C
C     The input/output variables TIME (i.t.), ALT (altitude),
C     LATIT (latitude), LONGIT (longitude), VEL (components of
C     velocities) keep in memory the best complying triplet
C     (i.e., the one with the minimum altitude), until
C     (yminus2, yminus1, y) corresponds to a first complying
C     triplet related to a next mirror point.
C
C     Then the parabolic interpolation of the current mirror
C     point is carried out (by calling subroutine INTERPOLMU),
C     giving the outputs defining this point: tmp (interpolated
C     time), hmp (interpolated altitude), latmp (interpolated
C     latitude), longmp (interpolated longitude), bm
C     (interpolated magnetic intensity) and mu (interpolated
C     magnetic moment).  The output variable mpok is put to 1
C     in order to inform the subroutine MULGTIME of this
C     achievement.
C
C    ************************************************************
C
C
      INCLUDE 'structure.h'
C
      EXTERNAL INTERPOLMU
C
      REAL*8    x, y(6), tminus1, yminus1(6), tminus2,
     >          yminus2(6), ymp(6),
     >          TIME(3), ALT(3), LATIT(3), LONGIT(3),
     >          VEL(3,3), tmp, hmp, latmp, longmp, bm, mu
      INTEGER*4 mpok
C
      REAL*8 ti(3), h(3), lat(3), long(3), mu3(3), b(3),
     >       vi2, sa2, bnorm, bm3(3)
      RECORD/zxyz/ mxyz, bcart
      RECORD/zgeo/ mrtp, mgde, mpos
      RECORD/zvec/ mb
C
C
      COMMON/UC160/ pi, deg, re, gmagmo, eclipt, geoid, uma
        REAL*8  pi, deg, re
        REAL*8  gmagmo
        REAL*8  eclipt, geoid(3), uma(30)
      COMMON/units/ lu, tu, vu
        REAL*8 lu, tu, vu
      COMMON/particule/ q, massp, massr
        REAL*8 q, massp, massr
      COMMON/lightvel/ c
        REAL*8 c
C
C
      mpok=0
C
      mxyz.x = yminus2(1)*(lu/1000.D0)
      mxyz.y = yminus2(2)*(lu/1000.D0)
```

```
71          mxyz.z = yminus2(3)*(lu/1000.D0)
72          call UT546 (mxyz, mrtp)
73          call um535 (mrtp, mgde)
74          ti(1)=tminus2
75          h(1)= (mgde.radius-re)*(1000.D0/lu)
76          lat(1)= 90.D0 - mgde.colat
77          long(1)= mgde.elong
78    C
79          mxyz.x = yminus1(1)*(lu/1000.D0)
80          mxyz.y = yminus1(2)*(lu/1000.D0)
81          mxyz.z = yminus1(3)*(lu/1000.D0)
82          call UT546 (mxyz, mrtp)
83          call um535 (mrtp, mgde)
84          ti(2)=tminus1
85          h(2)= (mgde.radius-re)*(1000.D0/lu)
86          lat(2)= 90.D0 - mgde.colat
87          long(2)= mgde.elong
88    C
89          if (h(1).eq.0 .or. h(2).eq.0) return
90    C
91          mxyz.x = y(1)*(lu/1000.D0)
92          mxyz.y = y(2)*(lu/1000.D0)
93          mxyz.z = y(3)*(lu/1000.D0)
94          call UT546 (mxyz, mrtp)
95          call um535 (mrtp, mgde)
96          ti(3) = x
97          h(3)= (mgde.radius-re)*(1000.D0/lu)
98          lat(3)= 90.D0 - mgde.colat
99          long(3)= mgde.elong
100   C
101         if(h(1).ge.h(2) .and. h(3).ge.h(2))then
102   C
103   C       GOOD CANDIDATE (A CANDIDATE = A TRIPLET)
104   C
105           if (init.eq.0) then
106   C
107   C         FIRST GOOD CANDIDATE FOR FIRST MIRROR POINT
108   C
109             init = 1
110             do i=1,3
111               VEL(1,i) = yminus2(3+i)
112               VEL(2,i) = yminus1(3+i)
113               VEL(3,i) = y(3+i)
114               TIME(i) = ti(i)*tu
115               ALT(i)  = h(i)
116               LATIT(i)= lat(i)
117               LONGIT(i) = long(i)
118             enddo
119           else
120   C
121   C         NEW GOOD CANDIDATE FOR CURRENT MIRROR POINT,
122   C         OR FIRST GOOD CANDIDATE FOR NEXT MIRROR POINT
123   C
124             if (lat(2)*LATIT(2) .gt. 0) then
125   C
126   C           NEW GOOD CANDIDATE FOR CURRENT MIRROR POINT
127   C           ==> SEARCH THE BEST CANDIDATE (THE GOOD CANDIDATE
128   C           WITH MINIMUM ALTITUDE)
129   C
130               if (h(2).lt.ALT(2)) then
131                 do i=1,3
132                   VEL(1,i) = yminus2(3+i)
133                   VEL(2,i) = yminus1(3+i)
134                   VEL(3,i) = y(3+i)
135                   TIME(i) = ti(i)*tu
136                   ALT(i)  = h(i)
137                   LATIT(i)= lat(i)
138                   LONGIT(i) = long(i)
139                 enddo
140               endif
```

```
141             else
142 C
143 C             THE BEST CANDIDATE (TRIPLET) TO INTERPOLATE CURRENT
144 C             MIRROR POINT IS FOUND
145 C
146               mpok=1
147 C
148 C             MAGNETIC INTENSITIES AND MOMENTS OF THE BEST TRIPLET
149 C
150               do i=1,3
151 C
152 C               ------------------------------------
153 C               Magnetic field evaluation by UNILIB
154 C               ------------------------------------
155 C
156 C               position in UNILIB shape and units
157                 mpos.radius = re + ALT(i)*(lu/1000.D0)
158                 mpos.colat = 90.D0-LATIT(i)
159                 mpos.elong = LONGIT(i)
160 C               Magnetic field as spherical vector
161                 CALL UM530 (mpos, mb, ifail)
162 C               Conversion as cartesian vector
163                 CALL UT542 (mpos, mb, bcart)
164                 b(1) = bcart.x
165                 b(2) = bcart.y
166                 b(3) = bcart.z
167 C               Conversion to (lu, tu) units
168                 bnorm = mb.dnrm*(tu/10000.D0)
169                 bm3(i) = mb.dnrm
170                 do j=1,3
171                   b(j) = b(j)*(tu/10000.D0)
172                 enddo
173 C
174                 vi2 = VEL(i,1)**2 + VEL(i,2)**2 + VEL(i,3)**2
175 C
176 C               ------------------------------------------------
177 C               Adiabatically invariant magnetic moment (in SI)
178 C               ------------------------------------------------
179 C
180                 sa2 = 1 -
181       >            ((VEL(i,1)*b(1) + VEL(i,2)*b(2)
182       >              + VEL(i,3)*b(3))**2/(vi2*bnorm**2))
183                 massp = massr/DSQRT((1-(vi2/(c*c))))
184 C               Relativistic magnetic moment
185                 mu3(i) = vi2*sa2*(massp)**2/(2.d0*massr*bnorm)
186 C               Conversion in SI
187                 mu3(i) = mu3(i)*((lu**2)/tu)
188 C
189               enddo
190 C
191 C             PARABOLIC INTERPOLATION OF CURRENT MIRROR POINT
192 C
193               CALL INTERPOLMU (TIME, ALT, LATIT, LONGIT, BM3, MU3,
194       >                        tmp, hmp, latmp, longmp, bm, mu)
195 C
196 C             FIRST GOOD CANDIDATE FOR NEXT MIRROR POINT
197 C
198               do i=1,3
199                 VEL(1,i) = yminus2(i+3)
200                 VEL(2,i) = yminus1(3+i)
201                 VEL(3,i) = y(3+i)
202                 TIME(i) = ti(i)*tu
203                 ALT(i)  = h(i)
204                 LATIT(i)= lat(i)
205                 LONGIT(i) = long(i)
206               enddo
207             endif
208           endif
209         endif
210 C
```

```
211   C
212         RETURN
213         END
end
```

4 Fichier batch IDL pour représenter les trajectoires 3D

```
; Plotting of trajectories in 3D space, with a plotting of the Earth

.r spenvis:[developer.idl]draw_map
.r spenvis:[developer.idl]world
.r spenvis:[developer.idl]set_plot_3d
.r spenvis:[developer.idl]errorhandler
.r spenvis:[developer.idl]errormessage

window, 0
loadct, 12

re = 6371.2

; Length of opened file
n=7198
; opening of the file
openr,1,'ckgc.dat'
title= ' '
readf,1,title
table=replicate({alt:0.0, lat:0.0, lon:0.0}, n-1L)
readf,1,table
close,1

z=table.lon*re
x =table.alt*re
y = table.lat*re
r=0.5*[-1,1]*max(abs([x,y,z]))

pb = !d.n_colors-1
!p.background=pb
!p.color=0
map=world(cont=!p.color*0.5+0.5*pb,$
          background=pb*0.9+0.1*!p.color)
set_plot,'ps'
device, file='guidcent.ps',xsize=10,ysize=10,/color,bit=4
!p.background=!d.n_colors-1
!p.color=0

;view along the geomagnetic axis of Earth
;set_plot_3d,range=r[1]*1.8,ax=79.3,az=0,rotation=-71.41
set_plot_3d,range=r[1]*1.8,ax=0,az=0,rotation=-90

draw_map, map

plots,x,y,z,thick=1.

close_plot_3d

device,/close
set_plot,'x'
end
```