The Curlometermethod and Cluster II

Christian Maszl

August 31, 2004

Contents

Chapter 1

Introduction

1.1 Abstract

In this project, the curlometer method was applied to FGM (**F**lux **G**ate **M**agnetometer) data, provided by the four CLUSTER II satellites. The primary objective was to compute the current density $\vec{J}_{av}$ and $\vec{\nabla} \cdot B$.
In order to investigate data over long periods, a ©MatLab program was created to perform such tasks. Additionally the structure of the program was designed to allow plugins for other instruments to be implemented easily (e.g. WHISER, EDI, CIS... = *multipurpose software*).
Finally, some basic model current sheets were tested to demonstrate the limitations of the curlometer method quite plainly.

Chapter 2

Summary

2.1 Development of a *Multipurpose Software*

A *multipurpose software* was developed and partially tested. Unfortunately there were some restrictions, which are described in sec(4.3) on p.(30) and in sec(4.4) on p.(31). The following instruments are supported: *FGM, CIS, EDI, ASPOC, EFW* and *WHISPER*. Furthermore it is possible to obtain position- , phase- and spin axis orientation data of the spacecrafts.
High resolution- and normal resolution mode for *FGM* magnetometer data is fully functional. The current density $\vec{J}_{av}$ and $\vec{\nabla} \cdot \vec{B}$ were calculated by using the *Curlometer method* from Sec(3.5). This can be performed for a single day or longer periods up to one year duration.
The current densities J_{av} are typically in the order of $10^{-7}\, Am^{-2}$.

2.2 Investigations on *Model Current Sheets*

Four main effects in case of a *current tube* with homogenous current density $\vec{J}_{av}$ can be observed:

1) Seemingly widening of the tube diameter by increasing the satellite separation.

2) Decrease of the current density amplitude with increasing satellite separation.

3) First appearance of backward currents if $s \approx r$.

4) $\vec{\nabla} \cdot \vec{B}$ not useable as an error estimate for thin current tubes.

In case of a flat current sheet, only the points 1) and 2) can be observed.
Due to the special character of these sheets the next step could be, to make this investigations with more artless current sheets.
It is possible that case 3) not occur in nature because of the shape of the *magnetopause*. Because of the large scales I estimate that the sheet is locally flat and therefore point 1) and 2) are the main sources of error.

Chapter 3

Background

3.1 Cluster II - The Mission

Cluster[1] is a collection of four spacecraft flying in formation around Earth. They relay the most detailed ever information about how the solar wind affects our planet in three dimensions. The solar wind (the perpetual stream of subatomic particles given out by the Sun) can damage communications satellites and power stations on Earth. The operation life-time of the Cluster mission runs from February 2001 and December 2005.

The four Cluster spacecraft have spent several years passing in and out of our planet's magnetic field. Their mission will be to complete the most detailed investigation ever made into the ways in which the Sun and Earth interact.

3.1.1 What's special?

The Sun emits the solar wind, which is a thin, hot, ionised gas that carries particles and magnetic fields outward from the Sun.

The Earth is shielded from the full blast by its magnetosphere, the region around our planet controlled by its magnetic field. Some solar wind descends into Earth's upper atmosphere through the polar cusps, funnel-like openings in the magnetosphere at the poles. These energetic particles excite atoms and molecules in the upper atmosphere to create the Northern and Southern Lights (the auroras). The part of a planetary magnetosphere that is pushed in the direction of the solar wind is known as the magnetotail.
Cluster will determine the physical processes involved in the interaction between the solar wind and the magnetosphere by visiting key regions like the polar cusps and the magnetotail. The four Cluster spacecraft map the plasma structures contained in these regions in three dimensions. The simultaneous four-point measurements also allow close studies of plasma quantities in both space and time.
During periods of high solar activity (which cycles every 11 years), the solar wind can be particularly energetic. This can have a dramatic effect on human activities, disrupting electrical power and telecommunications or causing serious problems in the operation of satellites, especially those in geostationary orbit. Subtle changes to the weather on Earth also occur during these times. Watching the effects of this increased activity during these periods is one of the main tasks of Cluster.

[1]Excerpt: www.esa.int - ESA science / *Cluster II - Overview*

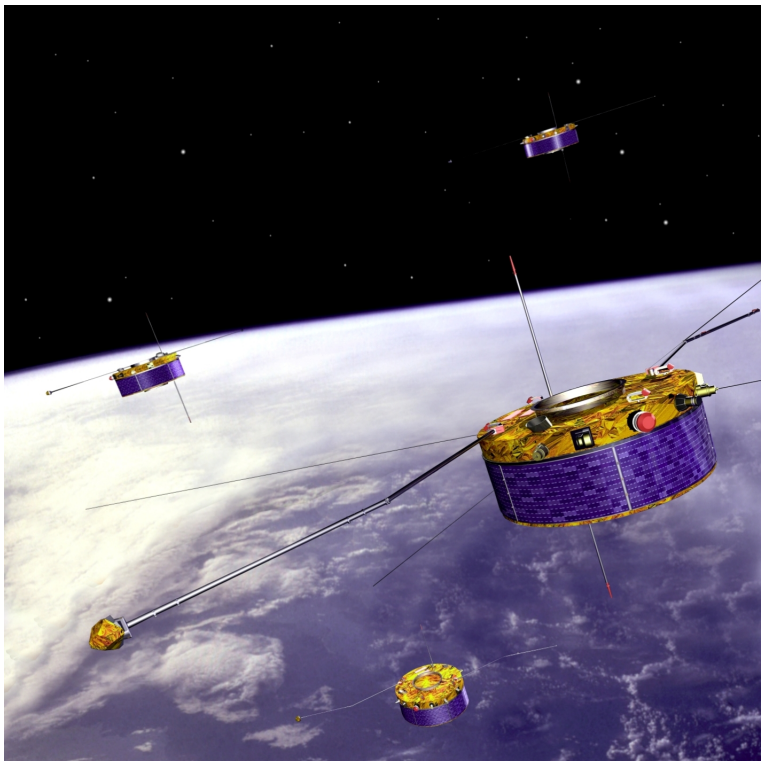

Figure 3.1: The Cluster II tetrahedron

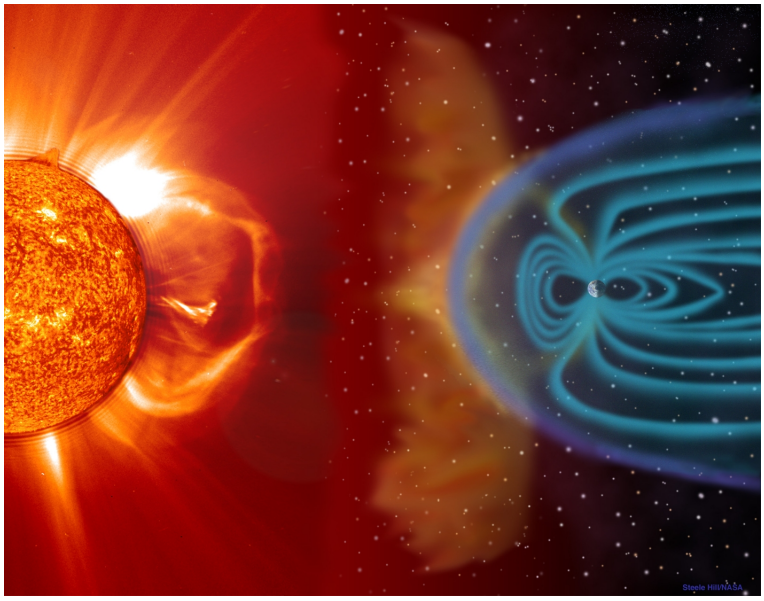

Figure 3.2: Artist's view of the solar wind and the magnetossphere

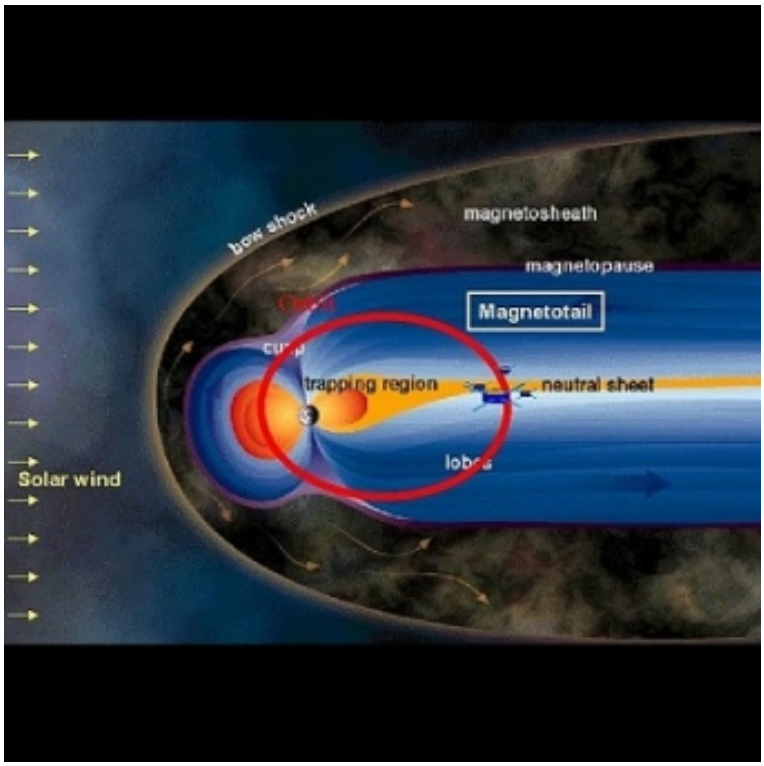

Figure 3.3: Cluster II trapping region

Understanding the interaction between the solar wind and the magnetosphere and how the plasma levels of the magnetosphere are affected is important. Cluster will help us to prepare for the effects of sudden bursts of solar energy here on Earth.

3.1.2 Spacecraft

The Cluster spacecraft resemble giant 'Lego' sets, assembled from thousands of individual blocks. Each one is shaped like a giant disc, 1.3 metres high and 2.9 metres wide, with a cylinder in the centre.
Six spherical fuel tanks are attached to the outside of this central cylinder. The fuel they carry accounts for more than half the launch weight of each spacecraft. Most of the fuel is consumed soon after launch and in complex manoeuvres to reach their operational orbits. Each spacecraft also carries eight thrusters for smaller changes of orbit.
Around the central cylinder is the main equipment platform. Electrical power comes from six curved solar panels attached around the outside of the platform. Five batteries are used for power supply during the four-hour-long eclipses when the spacecraft enter Earth's shadow.
Rod-shaped booms open out once Cluster reaches orbit. There are two antennae for communications, two sensors, and four wire booms that operate when the spacecraft begins to spin. These measure changing electrical and magnetic fields around each spacecraft.

3.1.3 Journey

At each launch, two Cluster satellites were placed in an elliptical orbit whose height varied from 200 to 18 000 kilometres above Earth. The two satellites of each launch were then released, one after the other and used their own on-board propulsion systems to reach the final operational orbit (19 000 to 119 000 kilometres from the planet).
The first pair of Cluster satellites lifted off on 16 July 2000, the second pair one month later. This gap allowed fewer people to be used for mission control in the European Space Operations Centre (ESOC) in Darmstadt (Germany).
Once the booster reached the correct altitude, after liftoff, the Fregat payload assist module and its two Cluster spacecraft were released. The Fregat main engine fired almost immediately to achieve a circular orbit of approximately 200 kilometres high. About an hour later, the Fregat engine fired again to inject the spacecraft into an elliptical orbit.
The two satellites were released, one after the other. Each Cluster spacecraft main engine performed six major manoeuvres, using the large amount of on-board fuel (about half of each satellite's launch mass).

3.1.4 History

The Cluster mission was first proposed in November 1982. The idea was developed into a proposal to study the 'cusp' and the 'magnetotail' regions of the Earth's magnetosphere with a polar orbiting mission. The Cluster idea developed into a proposal and then a mission. In 1996, Cluster was ready for launch.
Cluster was expected to benefit from a 'free' launch on the first test flight of the newly developed Ariane-5 booster. After several minor delays, Ariane-501 lifted off from Kourou, French Guiana on 4 June 1996, carrying its payload of four Cluster satellites. Unfortunately, intense aerodynamic loads resulted in its break-up and initiation of the automatic destruct system.
To recover some of the unique science from the mission, ESA decided to build a fifth Cluster satellite (named 'Phoenix'). It would be equipped with flight spares of the experiments and subsystems prepared for the Cluster mission. Phoenix was expected to be fully integrated and tested by mid-1997, opening the way for a launch later that year. However, awareness grew that the scientific objectives of the Cluster mission could not be met by a single spacecraft. There were proposals to rebuild three or four full-size Cluster spacecraft alongside Phoenix.
After a preliminary study, it was decided that a Soyuz rocket could launch a pair of Cluster spacecraft. However, the very eccentric orbit required a new upper stage. Two flights were successfully done at the beginning of 2000 and about 6 months later, Cluster was launched by a Soyuz-Fregat launcher from Baikonur Cosmodrome, Kazakhstan.

3.1.5 Partnerships

Prime contractor for the original (lost) Cluster and replacement Cluster satellites was Dornier Satellitensysteme GmbH (now Astrium), Friedrichshafen, Germany, the leader of an industrial consortium involving 35 major contractors from all of the ESA member countries and the United States.
Each spacecraft carries an identical set of 11 instruments to investigate charged particles, electrical, and magnetic fields. These were built by European and American instrument teams led by Principal Investigators.
The Cluster scientific community includes the ESA Project Scientist, 11 Principal Investigators, and more than 250 Co-Investigators from ESA Member States, the

United States, Canada, China, the Czech Republic, Hungary, India, Israel, Japan, and Russia.

3.2 Cluster II - Operation & Instruments

The four Cluster[2] spacecraft are placed in nearly identical, highly eccentric polar orbits. In the two-year working period the satellites are able to make a detailed exploration of all interesting areas of the magnetosphere.

The orbit for each spacecraft is selected so that each is located at a vertex of a pre-determined tetrahedron when crossing the regions of interest within the magnetosphere. The separation distances between the spacecraft will be adjusted during the mission depending on what will be studied and will vary from a few hundred kilometres to a few Earth radii.

In order to perform measurements the spacecraft are gyrating around their z-axis. In normal operation mode the frequency is $0.25Hz$ in high resolution "burst" mode the gyration frequency is $22Hz$.

3.2.1 FGM - Fluxgate Magnetometer

Consists of two magnetometers[3] on a five-metre long boom from the spacecraft to avoid interference. It measures the magnetic field along the orbit.
The fluxgate magnetometers are similar to many previous instruments flown in Earth-orbit and on other planetary and interplanetary missions. In order to minimise the magnetic background of the spacecraft, one of the magnetometer sensors (the outboard, or OB sensor) is located at the end of one of the two $5.2m$ radial booms of the spacecraft, the other (the inboard, or IB sensor) at $1.5m$ inboard from the end of the boom. The boom mounted sensor units are essentially passive devices, consisting of coils of wire wrapped around toroidal cores of magnetic material.
The magnetometers have eight possible operating ranges; of these, five are used on the Cluster magnetometers. These ranges were selected to provide good resolution in the solar wind (with expected field magnitudes between 3 and $30nT$), and up to the highest field values expected in the magnetosphere along the Cluster orbit (up to about $1000nT$). The highest range ($\approx 65500nT$) is used only to facilitate background testing.

3.2.2 EDI - Electron Drift Instrument

The Electron Drift Instrument (EDI) measures the drift of a weak beam of test electrons, from which the ambient electric fields and magnetic field gradients can be determined. As a by-product, the magnetic field strength is also measured.

3.2.3 ASPOC - Active Spacecraft Potential Control Experiment

Neutralises the positive charge that is built on the spacecraft which would seriously interefere with the measurements. It emits indium ions to space through a small needle.

[2]Excerpt: spaceweb.oulu.fi/projects/cluster/ - Cluster II project in Oulu
[3]Excerpt: www.sp.ph.ic.ac.uk/Cluster/ - Imperal College Cluster Group

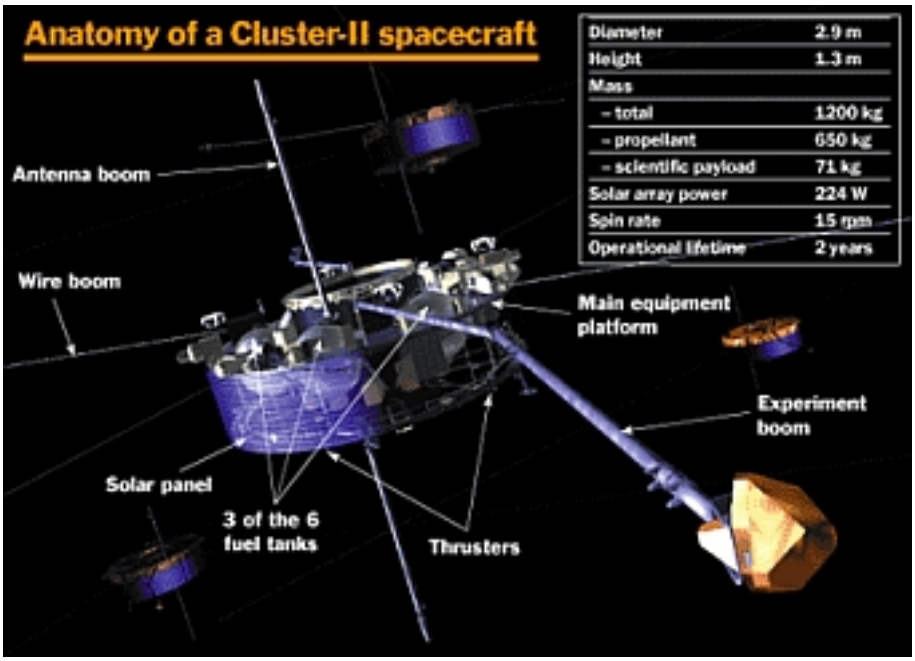

Figure 3.4: Anatomy of the Cluster Satellites

3.2.4 STAFF - Spatio-Temporal Analysis of Field Fluctuation experiment

Consists of a magnetometer on the end of a five-meter long boom that analyzes waves (rapid variations in the magnetic field).

3.2.5 EFW - Electric Field and Wave Experiment

Measures electric field to study plasma convection and waves with sensors on four 50 m long wire booms.

3.2.6 DWP - Digital Wave Processing experiment

Controls the wave experiments and performs calculations for them. Helps the WEC-instruments to make effective use of spacecraft power resources and telemetry information.

3.2.7 WHISPER - Waves of High frequency and Sounder for Probing of Electron density by Relaxation experiment

Measures the density of charged particles that fill near-Earth space with resonance sounding (radar) technique.

3.2.8 WBD - Wide Band Data instrument

Makes high-resolution measurements of both electric and magnetic fields in selected frequency bands from radio sounds from particles around Earth's magnetic poles.

3.2.9 PEACE - Plasma Electron And Current Experiment

Looks at all electrons in space which have low to medium energies, counts them and measures their direction and speed. One part of the instrument measures faster electrons, another part slower ones.

3.2.10 CIS - Cluster Ion Spectrometry experiment

Analyses the composition, mass and distribution functions of ions in the nearby space plasma and in the solar wind during each four-second spin of the spacecraft.

3.2.11 RAPID - Research with Adaptive Particle Imaging Detectors

Uses two different and independent systems (Imaging Electron Spectrometer and Imaging Mass Spectrometer) to record the highest energy electrons and ions.

3.2.12 WEC - Wave Experiment Consortium

DWP, EFW, STAFF, WBD, and WHISPER, which all measure electric and magnetic fields and waves in some form. They have been grouped together to form WEC in order to maximise resources

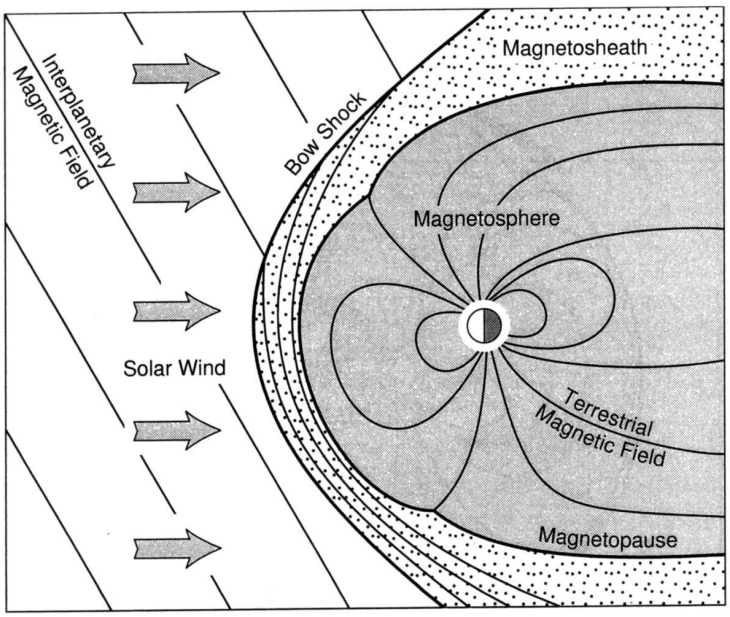

Figure 3.5: Topography of the solar-terrestrial enviroment

3.3 The Physical Background

3.3.1 Solar Wind

The sun[4] emits a highly conducting plasma at supersonic speeds of about $500km/s$ into the interplanetary space. The *solar wind* consits mainly of electrons and protons, with an admixture of 5% Helium. Because of the high conductivity, the solar magnetic field is frozen in the plasma and drawn outward by the expanding solar wind. Typical values of solar wind parameters: $n_e \approx 5cm^{-3}$, $T_e \approx 10^5 K$ $(1eV = 11600K)$ and B of the *interplanetary* magnetic field is of the order of $5nT$.

When the solar wind hits on the Earth's dipolar magnetic field, it cannot simply pentrate it bur rather is slowed down and, to a large extent, deflected around it. Since the solar wind hits the obstacle with supersonic speed, a *bow shock* wave is generated, where the plasma is slowed down and a substanial fraction of the particels' kinetic energy is converted into thermal energy. The region of thermalized subsonic plasma behind the bow shock is called *magnetosheath*. Its plasma is denser and hotter than the solar wind plasma and the magnetic field strength has higher values in this region.

3.3.2 Magnetossphere

The interplanetary magnetic field lines cannot pentrate the terrestial field lines and the solar wind particles cannot leave the interplanetary field lines due to the frozen-in characteristic of a highly conducting plasma.
The boundary separating the two different regions is called *magnetopause* and the cavity generated by the terrestrial field has been named *magnetosphere*.
The kinetic pressure of the solar wind plasma distorts the outer part of the terrestrial dipolar field. At the frontside the field becomes compressed, while the nightside

[4]Excerpt: W. Baumjohann / R. A. Treumann - *Basic Space Plasma Physics*

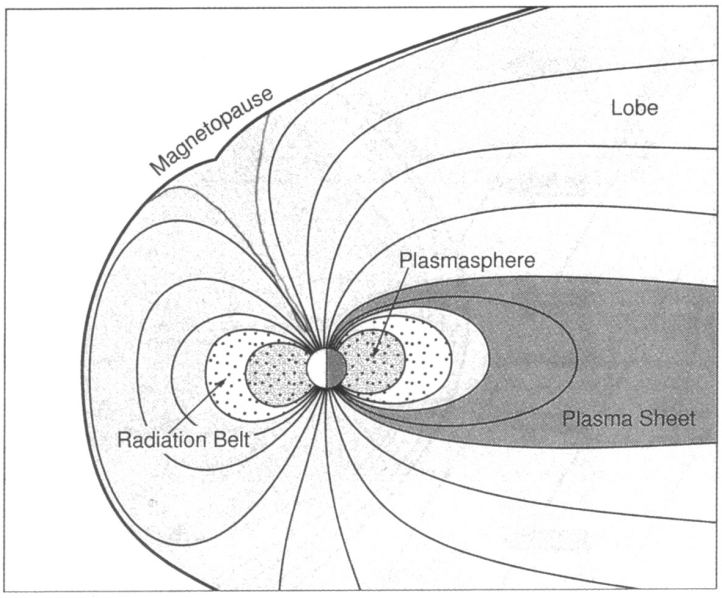

Figure 3.6: Plasma structure of Earth's magnetosphere

magnetic field is stretched out into a long *magnetotail* which reaches far beyond the lunar orbit.

3.3.3 Radiation belt

The *radiation belt* lies on dipolar field lines between about 2 and $6R_E$ ($1R_E \approx 6371km$). It consists of energetic electrons and ions which move along the field lines and oscillate back and forth between the two hemispheres.
This is due to the fact that the dipolar magnetic field acts like a *magnetic mirror*. A *dipole magnetic field* has a field strength minimum at the equator and converging field lines at the north and south pole. Since the magnetic moment is invariant and the total energy is a constant of motion, only the pitch angle can change when the magnetic field increases or decreases along the guiding center trajectory.

$$\mu = \frac{mv^2 sin^2\alpha}{2B} \qquad v_\perp = vsin\alpha \tag{3.1}$$

Eq(3.1) shows that the pitch angles of a particle at different locations are directly related to the magnetic field strengths at those locations according to:

$$\frac{sin^2\alpha_2}{sin^2\alpha_1} = \frac{B_2}{B_1} \tag{3.2}$$

If the pitch angle reaches $\alpha = 90$, the particle is reflected from this *mirror point*.
The *bounce period* τ_b is the time it takes a particle to move from the equatorial plane to one mirror point, then to the other and back to the equatorial plane.

$$\tau_b = 4 \int_0^{\lambda_m} \frac{ds}{v_\parallel} \tag{3.3}$$

The bounce period of $1keV$ electrons is in the order of a few seconds while it takes a few minutes for the heavier protons to complete a full bounce cycle.

Typical parameters: $n_e \approx 1cm^{-3}$, $T_e \approx 5 \cdot 10^7 K$ and $B \approx 100 - 1000nT$

3.3.4 Plasma sheet

Most of the magnetotail plasma is concentrated around the tail midplane in an about $10R_E$ thick *plasma sheet*. Near the Earth, it reaches down to the high-latitude *auroral ionosphere* along the field lines. The plasma sheet is also the source of the ring current.

Typical parameters: $n_e \approx 0.5cm^{-3}$, $T_e \approx 5 \cdot 10^6 K$ and $B \approx 10nT$

3.3.5 Magnetotail lobe

The *magnetotail lobe* contains a highly rarefied plasma.

Typical parameters: $n_e \approx 10^{-2}cm^{-3}$, $T_e \approx 5 \cdot 10^5 K$ and $B \approx 30nT$

3.3.6 Ionosphere

The solar ultraviolet light impinging on the Earth's atmosphere ionizes a fraction of the neutral atmosphere. At altitudes above $80km$, collisions are too infrequent to result in rapid recombination and so a permanent ionized population called the *ionosphere* is formed.

Typical parameters: $n_e \approx 10^5 cm^{-3}$, $T_e \approx 5 \cdot 10^3 K$ and $B \approx 10^4 nT$

The ionosphere extends to rather high altitudes and, at low- and mid-latitudes, gradually merges into the *plasmasphere*. The plasmasphere is a torusshaped volume inside the radiation belt. It contains a cool but dense plasma of ionospheric origin, which co-rotates with the Earth.

Typical parameters: $n_e \approx 5 \cdot 10^2 cm^{-3}$, $T_e \approx 5 \cdot 10^3 K$

The plasmasphere extends outwards to about $4R_E$ where the density drops down sharply to about $1cm^{-3}$. This boundary is known as the *plasmapause*.
At high latitudes plasma sheet electrons can precipitate along magnetic field lines down to ionospheric altitudes where they collide with and ionize neutral atmospheric particles. As a by-product, photons emitted by this process create the polar light known as the *aurora*. These auroras are typically observed inside the *auroral oval* which contains the footprints of those field lines which thread the plasma sheet. Inside of the auroral oval lies the *polar cap* which is threaded by field lines connecting to the tail lobe.

3.4 Magnetospheric currents

The plasmas discussed in the previous section are usually not stationary but move under the influence of external forces. Sometimes ions and electrons move together, like in the solar wind. But there are other occasions and in other plasma regions where ions and electrons move in different directions creating an electric current. Such currents are very important for the dynamics of the Earth's plasma environment. They transport *charge*, *mass*, *momentum* and *energy*. Moreover, these cur-

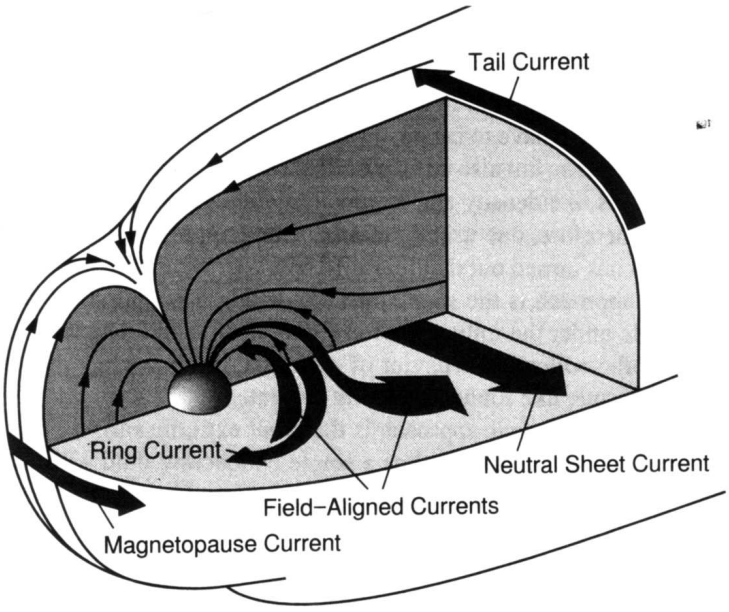

Figure 3.7: Synopsis of magnetospheric currents

rents create magnetic fields which may severely alter or distort any pre-existing fields.

In fact, the distortion of the terrestial dipole field into the typical shape of the magnetosphere is accompanied by electrical currents.
The compression of the terrestrial magnetic field on the dayside is associated with current flow across the magnetopause surface, the *magnetopause current*. The tail-like field of the nightside magnetosphere is accompanied by the *tail current* which flows on the tail surface and the *neutral sheet current* in the central plasma sheet. Both of these currents are connected and form a θ-like current system if viewed from along the Earth-Sun line.

Another large-scale current system, which influences the configuration of the inner magnetosphere is the *ring current*. The ring current flows around the Earth in a westward direction at radial distances in the order of several Earth radii and is carried by the radiation belt particles. In addition to their bounce motion these particles drift (due to the curvature of the magnetic field) slowly around the Earth. Since the protons drift westward while the electrons move in the eastward direction, this constitutes a net charge transport.

A number of current systems exist in the conducting layers of the Earth's ionosphere at altitudes of 100-$150km$. Most notable are the *auroral electrojets* located inside the auroral oval, the *Sq currents* in the dayside mid-latitude ionosphere, and the *equatorial electrojet* near the magnetic equator.

In addition to these perpendicular currents, currents also flow along magnetic field lines. The *field-aligned currents* connect the magnetospheric current systems in the magnetosphere to those flowing in the polar ionossphere. The field-aligned currents are mainly carried by electrons and are essential for the exchange of energy and momentum between these regions.

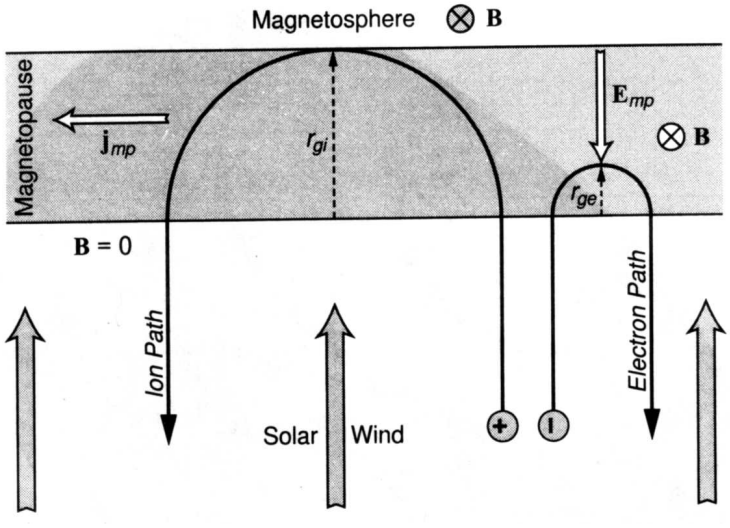

Figure 3.8: Specular reflection off the magnetopause

3.4.1 Magnetopause current

The following chapter deals with the *magnetopause* current. The fully ionized and magnetized solar wind plasma cannot mix with the terrestrial flux tubes. Instead, it will deviate from its original direction and will, by its dynamical pressure, compress the terrestrial field and confine it into a small region of space, producing the *magnetosphere*. This layer is a discontinuity which must be different from *bow shock* because the plasma flow behind the *bow shock* is subsonic. In fact, to first order, the magnetopause can be regarded as a tangential discontinuity.

Separating the solar wind from the magnetospheric magnetic field and being a surface across which the magnetic field strength jumps from low interplanetary value to high magnetospheric field strength, the magnetopause represents a surface current layer. The origin of this current is shown schematically in fig(3.8).
Specularly reflected ions and electrons hitting the magnetospheric field inside the magnetopause boundary will perform half a gyro-orbit inside the magnetic field before escaping with reversed normal velocity from the magnetopause back into the magnetosheath.
The thickness of the solar wind-magnetosphere transition layer under such idealized conditions becomes of the order of the ion gyroradius.

$$r_{gi} = \frac{v_{sw\perp}}{\omega_{gi}} \tag{3.4}$$

Where $v_{sw\perp}$ is the normal velocity component of the solar wind and ω_{gi} represents the ion *cyclotron frequency* given by:

$$\omega_{gi} = \frac{q_{i^+}B}{m_{i^+}} \tag{3.5}$$

For typical values of $B = 10nT$ and $v_{sw\perp} = 300m/s$ one gets a value of $r_{gi} = 1000km$ for the thickness.
Electrons also perform half gyro-orbits, but with much smaller gyroradii. The sense of gyration inside the boundary is opposite due to the sign of the charge for both kinds of particles leading to the generation of a narrow surface current layer.

This current provides the additional magnetic field which compresses the magne-

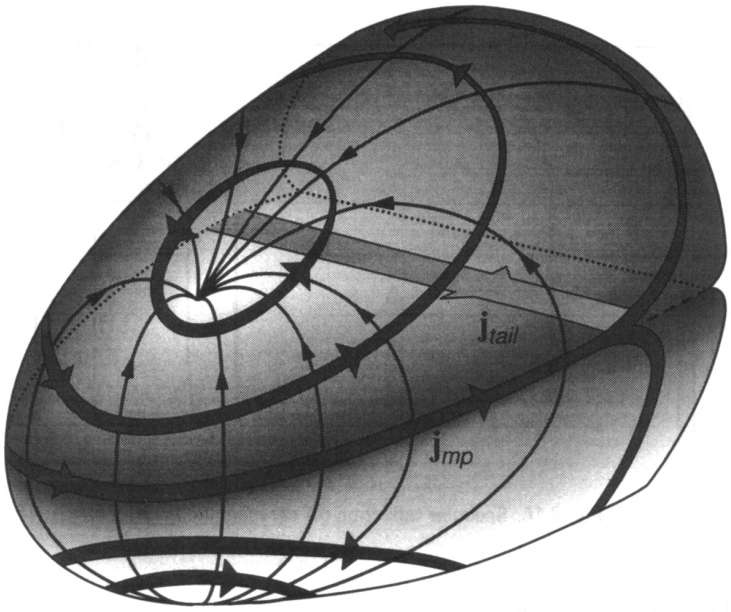

Figure 3.9: 3D geometry of magnetopause currents

tospheric field into the magnetosphere and at the same time annihilates its external part. It is a diamagnetic current caused by the perpendicular density gradient at the magnetopause.
The current density inside the magnetopause can be estimated to approximately $10^{-6}\,Am^{-2}$. The total current flowing inside the magnetopause is of the order of 10^{7}A. In the equatorial plane the magnetopause current flows from dawn to dusk, as shown schematically in fig(3.9). It closes on the *tail* magnetopause, where it splits into northern and southern parts flowing across the *lobe* magnetopause from dusk to dawn. The tail magnetopause current is additionaly fed by the cross-tail neutral sheet current which flows from dawn to dusk.

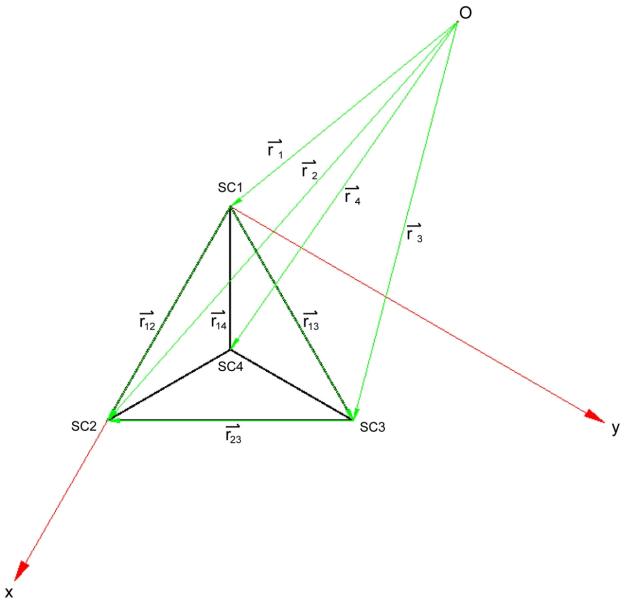

Figure 3.10: Geometrical coherences of Cluster II

3.5 The Curlomether Technique

3.5.1 Derivation of $\vec{J}_{av}$ and $\vec{\nabla} \cdot \vec{B}$

Assuming that the current density $\vec{J}_{av}$ is constant over the tetrahedal volume and the magnetic field $\vec{B}$ changes only linearly, one can use Ampere's law to compute $\vec{J}_{av}$. The difference from $\vec{\nabla} \cdot \vec{B} = 0$ gives a quality estimate of the derived quantities. Geometrical coherences can be observed in fig(3.10).

$$\mu_0 \vec{J}_{av} = \vec{\nabla} \times \vec{B} \tag{3.6}$$

Applying the theorem of Stoke's and assuming that $\vec{J}_{av}$ is constant over the tetrahedral volume one gets

$$\mu_0 \vec{J}_{av} \int_A \vec{da} = \int_L \vec{dl} \vec{B} \tag{3.7}$$

Assuming that $\vec{B}$ changes only linearly and implying that the satellite separation is small compared with the magnitude of the variation of $\vec{B}$ one can write (using SC1 as reference)

$$\vec{B}_{12} = \frac{\vec{B}_1 + \vec{B}_2}{2} \qquad \vec{r}_{12} = \vec{r}_1 - \vec{r}_2 \tag{3.8}$$

$$\vec{B}_{23} = \frac{\vec{B}_2 + \vec{B}_3}{2} \qquad \vec{r}_{23} = \vec{r}_2 - \vec{r}_3 \tag{3.9}$$

$$\vec{B}_{13} = \frac{\vec{B}_1 + \vec{B}_3}{2} \qquad \vec{r}_{13} = \vec{r}_1 - \vec{r}_3 \tag{3.10}$$

Rewritting

$$\int_A \vec{da} \qquad \textbf{as} \qquad \frac{1}{2}(\vec{r}_{13} \times \vec{r}_{12})$$

and projecting the magnetic field $\vec{B}$ on the edges of the tetrahedron one gets

$$\frac{1}{2}\mu_0\vec{J}_{av}\cdot(\vec{r}_{13}\times\vec{r}_{12}) = \vec{B}_{12}\cdot\vec{r}_{12} - \vec{B}_{23}\cdot\vec{r}_{23} - \vec{B}_{13}\cdot\vec{r}_{13}. \tag{3.11}$$

After some mathematical transformations using eq(3.8) - eq(3.10) and defining $\Delta\vec{B}_{13} = \vec{B}_1 - \vec{B}_3$, $\Delta\vec{B}_{12} = \vec{B}_1 - \vec{B}_2$ one finally achieves

$$\boxed{\mu_0\vec{J}_{av}\cdot(\vec{r}_{13}\times\vec{r}_{12}) = \Delta\vec{B}_{13}\cdot\vec{r}_{12} - \Delta\vec{B}_{12}\cdot\vec{r}_{13}.} \tag{3.12}$$

In order to obtain a quality estimate of the computed current density $\vec{J}_{av}$, one investigates the aberration of $\vec{\nabla}\cdot\vec{B}$ from 0. $\vec{J}_{av}$ and $\vec{\nabla}\cdot\vec{B}$ are subject to errors of which there are basically three types. The first concerns to the measurement uncertainties in $\vec{B}$ and the *spatial configuration* of the four spacecrafts.The second relates to the *linear interpolation* which is made between the various measurement points. The third relates to the *simultaneity of the measurements*.

Applying the theorem of Gauss on $\vec{\nabla}\cdot\vec{B}$ one gets

$$\int_V dV\,\vec{\nabla}\cdot\vec{B} = \int_A \vec{da}\cdot\vec{B}. \tag{3.13}$$

Treating $\vec{B}$ as constant and taking the average of $\vec{\nabla}\cdot\vec{B}$ yields

$$<\vec{\nabla}\cdot\vec{B}>_{av}\ V = \mid\sum_{\nu=1}^{4}\vec{A}_\nu\cdot\vec{B}_{m\nu}\mid. \tag{3.14}$$

ν represent the four areas of the tetrahedron. For example $\vec{A}_1$ one gets

$$\vec{A}_1 = \frac{1}{2}(\vec{r}_{13}\times\vec{r}_{12}). \tag{3.15}$$

For the tetrahedral volume one can write (using SC1 again as reference)

$$V = \mid\frac{1}{3!}\,\vec{r}_{14}\cdot(\vec{r}_{13}\times\vec{r}_{12})\mid. \tag{3.16}$$

At this state we use the linear approximation for $\vec{B}$. For the triangel composed of spacecrafts 1, 2 and 3 one can, for the interception point of the three heights, write

$$\vec{B}_{m1} = \frac{\vec{B}_1+\vec{B}_2+\vec{B}_3}{3}. \tag{3.17}$$

Analogous to this, one can obtain the fields for the remaining three areas. Inserting eq(3.15), eq(3.16) and eq(3.17) into eq(3.14) one obtains the result ($\alpha,\beta,\gamma = 2,3$ and 4)

$$\boxed{<\vec{\nabla}\cdot\vec{B}>_{av}\ \vec{r}_{14}\cdot(\vec{r}_{13}\times\vec{r}_{12}) = \mid\textstyle\sum_{cyclic}\Delta\vec{B}_{1\alpha}\cdot(\vec{r}_{1\beta}\times\vec{r}_{1\gamma})\mid.} \tag{3.18}$$

To get an error estimate for the computed current density one takes eq(3.18) divided by μ_0 and divides this by the norm of eq(3.12). One performs the first division to achieve the same dimension. Finally one can write for the error estimate

$$\boxed{P = \mu_0 \frac{< \vec{\nabla} \cdot \vec{B} >_{av}}{\mid \vec{J}_{av} \mid}.} \tag{3.19}$$

3.5.2 Derivation of $J_{av\perp}$ and $J_{av\parallel}$

In order to investigate the currents created by ions and electrons we split $\vec{J}_{av}$ into perpendicular and parallel parts with respect to the magnetic field strength $\vec{B}$. The field aligned currents $J_{av\parallel}$ are mainly carried by electrons and are essential for the exchange of energy and momentum between the magnetospheric current systems in the magnetosphere and in the polar ionosphere. The perpendicular currents $J_{av\parallel}$ are carried by the ions.
First we calculate the angle θ between $\vec{J}_{av}$ and $\vec{B}$,

$$\cos\,\theta = \frac{\vec{B} \cdot \vec{J}_{av}}{\mid \vec{B} \mid\mid \vec{J}_{av} \mid}. \tag{3.20}$$

Splitting $\vec{J}_{av}$ into a perpendicular and parallel part one derives

$$J_{av\parallel} = \mid \vec{J}_{av} \mid cos\,\theta \qquad J_{av\perp} = \mid \vec{J}_{av} \mid sin\,\theta. \tag{3.21}$$

Using the fact that

$$\sin\,\theta = \sqrt{1 - \cos^2\theta} \tag{3.22}$$

and taking into account that ©MatLab is more rapid with matrices, one substitutes $\cos\,\theta$ and finally achieves for $J_{av\parallel}$

$$\boxed{J_{av\parallel} = \frac{\vec{B} \cdot \vec{J}_{av}}{\mid \vec{B} \mid}.} \tag{3.23}$$

Similarly one derives for the perpendicular part $J_{av\perp}$

$$\boxed{J_{av\perp} = \mid \vec{J}_{av} \mid \sqrt{1 - (\frac{\vec{B} \cdot \vec{J}_{av}}{\mid \vec{B} \mid\mid \vec{J}_{av} \mid})^2}.} \tag{3.24}$$

Chapter 4

The Software

4.1 Preface

Originally this program was planed to apply the *curlometer method* to *FGM magnetometer data* for long periods.
The first version was capable of searching for ascii data files and performing the calculation. In order to save processing time and improve the flexibility of the program, the more flexible subroutine *getData* of Yuri Kothyaintsev (IRFU) was implemented as raw data source. A disadvantage incurred as a result of this step disallows the program to be run locally on a PC. As a consequence, it is necessary to establish a SSH connection to *SANNA* and run the program there.
Due to the high flexibility of the routine *getData* and the ability to obtain data for many measurement instruments (EFW, EDI, WHISPER ...) Stefan Buchert (IRFU) suggested that the program be extended. This would facilitrate other members of the institute which could write plugins for their special tasks. As a result the program was altered so that it now searches, obtains and sorts data for later disposal. Furthermore it provides a list with errors, gaps, ... that might occur during the operation.

Because of the programs "history" it is possible to find structures in the code that are not necessary anymore. e.g.
The first approach to fetch data from binary file was to use several c-programs of Stefan Buchert (IRFU) and the use of pipes to get the data. Therefore the subroutien *filelist* search for binary raw data files in *automatic mode*. After the switch to *getData* this isn't necessary anymore because this routine need only the date as input parameter. *GetData* fetches the data from *ISDAT* database *CDF* files.

It is possible to find more of such parts. These parts should be removed in a maybe second revision. In the following section you will find information about all used subroutines, limitations, bugs and not finished tasks. There are also a lot of coments in the source code to ease debugging and to reduce the period of vocational adjustment.

4.2 Description of Program Parts

4.2.1 The Main Program - *mainprogram.m*

This section provides a rough overview of the main program. For details relating to each subroutine see the following chapters.

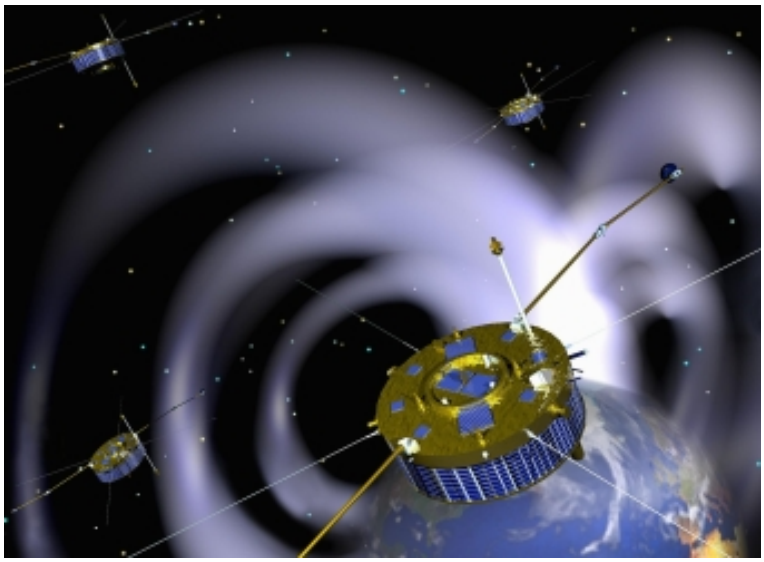

Figure 4.1: Artist's view of Cluster II and Earth

syscheck.m

The entire program is controlled by a small configuration file *configfile.m*. Initially the subroutine *syscheck.m* attempts to check the entries of *configfile.m*, next it performs several write and read tests and finally sets variables for the different operation modes.

filelist.m

The next step is the routine *filelist.m*. This searches for datafiles in *automatic mode*. In *manual mode* this routine is skipped.
Due to the fact that *getData* doesn't require a filelist anymore this subroutine can be replaced with a program that generates a date list for a whole year. If time is short or there is a lack of "motivation" to program it, it is possible to set the variable *mode* in *syscheck.m* to *fn*. This will generate a list of FGM raw data files. *getData* will take the date out of the filenames and so will work properly.

userinput.m

In *automatic mode* *userinput.m* asks for the desired operation mode. The first possibility is choice [A]. Following selection, hte program performs calculation for an entire year of data. This can require approximately $4h$ for *normal mode*-$4s$ data or a few days for *high resolution*-$45ms$ data depending on the processor speed and the available RAM (The datafiles are quite large and consume much memory. Therefore, it would be recommended to use a system with at least 512MB to avoid "swapping".)
The choice item [F] gives a filelist of found data and [M] allows to define a manual range.
In *manual mode* the desired date can be selected.

yuri.m

This subroutine provides the raw data and information concerning the "fetching" process. This subroutine is the interface to *getData.m*.

sorting.m

The four satellites do not function precisely in synchrony. Therefore *sorting.m* synchronises the four data files together depending on the time resolution of the input files.

report.m

***report.m* stores a report file to the HD in case *yuri.m* relate some errors.**

jav_fgm.m

***jav_fgm.m* applies the curlometer method to the data and saves the raw data and the computed data to the HD.**

4.2.2 The Control Center - *configfile.m*

is the main control unit of the program. In order to reduce the number of dialogues during execution of the program all necessary parameters become stored in this file.

datapath

This variable stores the path to the input data files.

Default on *SANNA*: *'/data/cluster/DDS'*

isdat_db

***ISDAT* database string.**

Default on *SANNA*: *'disco:10'*

year

defines the year to analyse for the *automatic mode*. *Manual mode* overrides this with a user entry.

op_mode

selects the instrument. At the moment only *'nr'* and *'hr'* modes (FGM - $4s$ **and** $45ms$ **data) have been tested and shown to work.**

extension

is a relic from a early development stage. Move to *syscheck.m* to obviate user errors if you like.

Default: *'*'*

datatype

defines the nature of the stored datafiles. It is possible to choose between binary *'-mat'* files and with any editor readable *'-ascii'* files. The recommendation is clear on binary files because loading and saving is approximately five times faster than in ascii format.

Default: *'-mat'*

logfilepath

is the directory where the *error.log* and *myfilelist.log* created by *logfiles.m* in *filelist.m* are stored. It is important to ensure that there is write permission for this directory!

outputspath

is the directory were the computed data and the raw data is going to be stored. It is important to ensure that there is write permission for this directory!

tempdirpath

is the directory for temporary data created by *getData.m*. It is important to ensure that there is write permission for this directory!

y0

is the magnetic permeability constant μ_0 in *SI* units.

Default: $\mu_0 = 1.256637061 \cdot 10^{-6} \frac{Vs}{Am}$

format long

is the means by which ©MatLab displays data on the screen.

advanced

The program is intended for use for whole days in *manual mode* or longer periods in *automatic mode*, although it is possible to analyse smaller periods. This handles the variable *advanced*.
The first three cells of the array represent the start time of the computation in hours, minutes and seconds. The fourth cell is the duration. Ensure that the period is not any longer than the remaining day.

Default: *[00 00 00 86400]*

compmode

switches between *manual mode* (*'m'*) and *automatic mode* (*'a'*).

nolog

set to *'n'* to supress the generation of logfiles.

Default: *'y'*

plot_mode_normal - Graphics output

is one of the control variables for the graphical output. Note that *plotdiagram.m* works only for the modes *'nr'* and *'hr'*! Set this variable to *'y'* to display Jav_x, Jav_y, Jav_z and $\frac{\vec{\nabla}\cdot\vec{B}}{|\vec{\nabla}\times\vec{B}|}$. Choose *'n'* to skip this plot.

Default: *'y'*

plot_mode_parnor - Graphics output

set to *'y'* produce the plot with $Jav_\perp$, $Jav_\parallel$ and $\frac{\vec{\nabla}\cdot\vec{B}}{|\vec{\nabla}\times\vec{B}|}$.

Default: *'y'*

legend_on - Graphics output

toggles the legend in the graph B_x, B_y, B_z and $\mid \vec{B} \mid$ on or off.

Default: *'y'*

4.2.3 The Security Unit - *syscheck.m*

This program loads *configfile.m* and examines the values to ensure they are valid. It also performs several write tests. It also converts the mode variable of the configfile to the proper input parameters for the subroutine *getData.m*.

At the beginning of the program one find several error messages that will be displayed in an error case. This part is followed by

Conversion to satellite instrument

This section contains the variables which are of no interest to the user. Prior to the insertion of a new plugin (e.g for EDI) ensure that the variables are checked and fit to your request.

Raw data file names are usually in the following shape : [date(yymmdd)][change number][mode].[satellite] - e.g 0203040000fn.01. This is a magnetometer raw datafile for 4^{th} of March 2002 for satellite nr.1 with no changes.
The variable *mode* is used in *filelist.m* to know which data to look (e.g FGM - mode='fn').

***instrument* is the parameter for *getData.m* which selects the operation mode. All possible parameters are listed at the end of *mainprogram.m* or *yuri.m*. There the style of the "fetched" data is also found.**

***option* is also a parameter for *getData.m* and prevents storage of the fetched data to the HD. The default option is *'nosave'* because the data will be stored afterwards anyway. (The only exception is the *'ve'*-mode. Check *syscheck.m* for the details.**

***timeacc* is the accuracy parameter for *sorting.m*. This parameter should be half the time resolution of the data. Accuracy must be optimized, if it is set too low results obtained would be inaccurate. On the other hand, if accuracy is set too high, data is eliminated.**

timeres, *datasrc*, *dataname* and *varname* are used to display messages on the screen.

Checks and Tests

The remainder of the subroutine is dedicated to checks, tests and also ensures that all path details end with a /.

4.2.4 Search for Data - *filelist.m*

As explained above this subroutine is not a necessity any more. However it can be usefull when substituting *getData.m*.

In *automatic mode filelist.m* searches for datafiles defined in the directory *datapath* with the adequate "string" defined in the *mode* variable.
The data becomes checked and sorted. Later the program checks if there are four files (satellite 1 to 4) for each date. All valid input files become stored in *myfilelist* and all others in *errorlog*. The first part of *errorlog* contains files with nonmatching *mode* criteria. The second contains information of uncompleted quad sets.
filelist.m also stores *errorlog* and *myfilelist* to the harddisk (*datatodisk*) in the directory defined in *logfilepath* (*error.log* and *myfilelist.log*).

4.2.5 Only one Question - *userinput.m*

In *manual mode* the user must input the desired date in *automatic mode*. The user can choose between [A] for a whole year, [M] for a favoured range or [F] to display a list of all found data files.

4.2.6 The Heart - *yuri.m*

This is the interface to the program *getData.m*. Unfortunately the subroutine *getData.m* has not been perfected and is not reliable. Check Sec(4.3) and Sec(4.4) for known bugs and recomended workarounds.

Referring to the conversation with Yuri Kothyainsev (IRFU) I recommend to substitute getData.m with a new program!

getData.m is designed to be used for minute intervals only, not hours. The main problem is that *getData.m* creates ascii files and pipes the data through several programs. This consumes much memory and time, especially when this is performed for a whole day.

Prepare the input parameters

The first step involves taking the date out of the filenames and start time and duration out of the variable *advanced*. Due to the fact that *getData.m* cannot handle long time intervals, the duration becomes split into four pieces and four new starttimes must be calculated.
Next the operation mode will be determined. *getData.m* returns fetched data in cell arrays. This data is different in the number of double arrays and therefore it is necessary to split *yuri.m* into three parts. Part one for "one" array data, part two for "two" array data and part three for "seven" array data, all stored in one cell.

Check the end of yuri.m or mainprogram.m for a list of all supported data files. Additional information concerning the cells, variable names, shape and dimension of the arrays, ... can also be found there.

Cells with one enclosed matrix

This refers to the *only* mode which is debugged and should perform properly. There is unfortunately a problem in *automatic mode*. If calculations are performed for longer periods then thirty days, *getData.m* reports *'No Data from Server'* when position data has been fetched. See sec(4.4.1) for a possible workaround.
First *getData* fetches $\vec{B}$ field data and $\vec{r}$ positon data for all satellites. It is possible that there are gaps for several hours in the data. In this case $\vec{B}$ or $\vec{r}$ are set to zero. Success or failure of this process is reported in *data_B* and *data_r*.
If there is no data for one satellite over a whole day, one can find this information in *b_report* and *c_report*.
***a_report*, *d_report* and *e_report* contain information if there is $\vec{B}$ data and no $\vec{r}$ ...**

Interpolate the position vector - *av_interp.m*

The time resolution of the position vector $\vec{r}$ is worse than the vector $\vec{B}$. Therefore one must interpolate $\vec{r}$ to the timeline of $\vec{B}$. This task can be performed on *av_interp.m*. The default method in *yuri.m* is spline.

Make clean and convert time - *fromepoch.m*

***getData.m* stores data during the process in a temporary directory defined by *tempdirpath*. These files are will be cleared after use.**
***getData.m* provides the time in *ISDAT epoch*. The function *fromepoch.m* allows this format to be converted into hours of a day.**
***getData.m* has also tends to mix data occasionally, particularlay if fetching data for long periods. To be confident that the timeline is in the proper order, the data is sorted.**

Cells with two enclosed matrices

The mode of operation is similar to that for one enlosed matrix. Check Sec(4.3.3) on p.(30) for the differences.

Cells with seven enclosed matrices

The mode of operation is similar to that for one enlosed matrix. Check Sec(4.3.4) on p.(31) for the differences and unfinished features.

4.2.7 Synchronise the Satellites - *sorting.m*

This script sorts the data of the four satellites according to time. *timeacc* sets the maximum time difference between the measurements of one event.
The procedure is as follows:
Because of different start times of each satellite, *sorting.m* looks for the latest start entry. Afterwards it takes this value and compares it with the other three data files. In the event of a match inside the $\pm$*timeacc* borders, the entries become stored in a list.

4.2.8 The Sneak - *report.m*

uses the report variables from *yuri.m* to generate error messages and store them in a file. The filename is in the following shape [yymmss]0000[mode].report (e.g 0203040000fn.report). This file is loaded in case of execution of *plotdiagram.m*.

I recommend to display this message file also for other possible plugins!

It is possible that *getData.m* does not provide position data $\vec{r}$. However *curlB.m* performs calculations and produces output if there is $\vec{r}$ data or not.

4.2.9 The Working Horse - *jav_fgm.m*

contains the program *curlB.m* and *datatodisk.m*. *curlB.m* applies the curlometer method to the FGM data. *datatodisk* stores this data and the raw data to the HD. The filenames are in the following style.
For the raw data [yymmdd]0000[mode].D[1-4].gse.hr.igm[asc or mat], for the current density [yymmdd]0000[mode].jav.gse.hr.igm[asc or mat] and for $\vec{\nabla} \cdot \vec{B}$ [yymmdd]0000[mode].divB.gse.hr.igm[asc or mat]. The data is stored in the directory defined by *outputspath* in *configfile.m*.

4.2.10 The Graphical Output - *plotdiagram.m*

plots diagrams on the screen. Note that this program only works for the FGM modes *nr* and *hr*.
As before this program starts with *syscheck.m*. Therefore it is important to ensure that the input parameters in *configfile.m* are adjusted to your needs!
plotdiagram.m is designed to plot three types of diagrams:
The three diagram types first B_x, B_y, B_z and $\mid \vec{B} \mid$, second Jav_x, Jav_y, Jav_z and $\frac{\vec{\nabla} \cdot \vec{B}}{|\vec{\nabla} \times \vec{B}|}$ and third $Jav_\perp$, $Jav_\parallel$ and $\frac{\vec{\nabla} \cdot \vec{B}}{|\vec{\nabla} \times \vec{B}|}$. Latter can be selected by changing *plot_mode_parnor* and *plot_mode_normal* in *configfile.m*

4.3 Unfinished Features

4.3.1 No time line for *op_mode*=sa

Take the time lines from the position vectors and use them as first columns (*yuri.m*).

4.3.2 Automatic accuracy for *sorting.m*

The time accuracy manually defined in *syscheck.m* in the variable *timeacc* should be found by the program itself. This can be done after *getData.m* when the timesteps are observed. Note that the time accuracy should be half the time resolution.

4.3.3 Test *yuri.m* for cells with two enclosed matrizes.

This mode isn't fully tested yet. In order to store two arrays in one both have to have the same length. It is possible that one of these two is zero. Therefore it is necessary to fill the empty array with the timeline of the full array and zeros for the data.
This mode is tested for the ***EFW*** instrument (*op_mode*='ef' in *configfile.m*) in manual

mode for the day *030702* (yymmdd) and working. Note that we1p34 for SC1 and we4p34 for SC4 is 0.

4.3.4 Finish *yuri.m* for cells with seven enclosed matrizes.

This section has not been completed yet. The remaining work involves searching for empty variables and to fill them with a timeline and zeros. It would be recommended to insert this function at *line 473*

4.3.5 Generate a date list for *yuri.m*

As *myfilelist.m* is not necessary anymore this time consuming routine can be skipped. All other routines only take the date out of the filenames. Therefore a data list should be created which can fullfill this task in a faster way. Store the date list in *myfilelist*.

4.3.6 Easier control of *userinput.m*

If one likes to enter a manual range one has to input numbers connected to the date and therefore must always check the filelist. A superior method would be to allow the user to define the range by declaring start- and end-times. In case *myfilelist* is subsituted by a date list, the option [F] filelist can be removed.

4.3.7 Multi coordinate system support for *datatodisk.m*

The default filenames for datatodisk contain *gse* for the coordinate system and *hr* for the time base.
The program *getData.m* is able to fetch data for different coordinate systems. Therefore it is recommend to add information about the time unit and coordinate system in *syscheck.m* to facilitate adjustment of the filenames.

Note that you also have to change plotdiagram.m in this case!

4.3.8 Automatic eps-file generator for *plotdiagram.m*

It may be convenient, if a large ammount of data is observed, to create eps-files of the graphs on the fly for later use. This should avoid long loading and waiting times if the data is monitored more frequently. This feature could also be controlled with *configfile.m*.

4.3.9 Replace *getData.m*

As *getData.m* is not reliable and not intended for use in the required way, it is recommended to write your own data fetching routine. For FGM data the C programs of Stephan Buchert (IRFU) can be used for example.
But there are also signs that the errors and problems occur in the ISDAT database, sec(4.4.1) and sec(4.4.3). If you have the skills you can try and find it there.

4.4 Known bugs

In this section a list of known bugs is found. Some have been corrected. In any case they are listed to counteract of problems which may occur if program parts are added or changed.

4.4.1 No data from Server - *getData.m*

If one performs calculations for periods exceeding then thirty days, *getData.m* occasionaly report *'No Data from Server'* when fetching position data. But position data is always available! It should be noted that this error is not reproducible. This is maybe due an error in the ISDAT database.

***Workaround*: Use of *clorb.m* to get the position data but be careful with the coordinate systems!**

4.4.2 Scrambled data - *getData.m*

Mixes the time line for long intervals[1].

***Workaround*: Always use the *sort* command of ©MatLab afterwards!**

4.4.3 Fail to work - *getData.m*

In cases where *getData.m* is used in short intervals repetitively. It is possible that the program does not work anymore. e.g. It provides no data without any try to fetch it[2].This is maybe due an error in the ISDAT database.

***Workaround*: Always delete the compiled function from the memory. Use *inmem* to check for compiled functions manually. Use *clear getData* to delete the function after use!**

4.4.4 Freezes in *op_mode*='hr' - *getData.m*

Sometimes the program fail to work in *'hr'* mode. This can happen if two users execute *getData.m* at the same time.
Besides this mode is fully functional.

4.4.5 Insufficient sorting accuracy - *sorting.m*

Due to the sorting algorithm it is possible, although all measurements are in the *timeacc* boundaries with respect to the reference satellite, that the error between the other satellites is in the magnitude of the time resolution. e.g.
Suppose the time resolution is $4s$ and therefore the time accuracy for sorting $2s$.The sorting routine finds subsequent times with respect to the reference time of SC1 called $t1$. SC2 measured $1.9s$ before $t1$, SC3 $1.9s$ after $t1$ and SC4 $0.9s$ after $t1$. All are in the range of $2s$ compared with SC1, although the maximum error is $3.8s$. If one checks SC2 again, one observes that the next value is at $2.1s$ after $t1$. Therefore the maximum error is $2.1s$ instead of $3.8s$!

***Workaround*: Do not settle with one reference satellite, but also take other combinations into account.**

4.4.6 Freezes in *op_mode*='ph' - *getData.m*

The program hangs if starting it in this mode.

***Workaround*: Ask Yuri Kothyaintsev (IRFU) for support!**

[1] Solution already implemented in yuri.m
[2] Solution already implemented in yuri.m

4.4.7 Error in *op_mode*='wp' - *yuri.m*

This mode is only tested with [yymmdd]=020304. *yuri.m* displays this error message: *All rows in the bracketed expression mut have the same number of columns!*

***Workaround*: Check all variabls Dtemp* in *yuri.m* for the number of columns and adjust them.**

4.4.8 Error in *op_mode*='sp' - *yuri.m*

This mode is only tested with [yymmdd]=020304. *yuri.m* displays this error message: *All rows in the bracketed expression mut have the same number of columns!*

***Workaround*: Check all variabls Dtemp* in *yuri.m* for the number of columns and adjust them.**

4.5 HOW TO install a new Plugin

This section describes how to develop and implement new program parts in the software. It would be recommended to read Sec(4.2) to get a better insight in the mode of operation. Second I suggest to follow these steps for the mode *op_mode*='nr'. This mode is tested and working. After looking thrue this steps and running the program you should get the idea how to do it.

$1^{st} step$: **Check the end of *yuri.m* for supported data types.**

$2^{nd} step$: **Find the mode variable *op_mode* for the desired data in *configfile.m*.**

$3^{rd} step$: **In *syscheck.m* go to section *Conversion to satellite instrument* and make the proper entries for the variables *mode, timeacc, datasrc, timeres, dataname* and *varname*. The latter four concern only the output on the display. (e.g. FGM:** $D1 = [time\ B_1\ r_1] \Rightarrow$ ***varname*='B')**

$4^{th} step$: **The data for the four satellites is stored in the arrays *D1, D2, D3* and *D4*. One can also find the structure of the arrays in the list at the end of *yuri.m* or *mainprogram.m*. The format is always** $[(time)(yourdata)...(position)]$**. One should check the variable *cell_name* first. Here one finds the variables included in D1-D4. The data is stored in this order and in the *format* specified in the list of supported data types. The inner structure of** $(yourdata)$ **for example *EFW* is** $[t\ x_1\ t\ x_2\ r_x\ r_y\ r_z]$**.**

$5^{th} step$: **In *mainprogram.m* go to the "*plugin*" section. Find the proper case to your *op_mode* and insert your subroutines. It is recommended to always check the report files (**.report*) before you working with the data!**

Chapter 5

Results

5.1 Test results

This section provides information about the testing and debugging status. In the operation modes *wp* and and *sp* are still two bugs. *ve* is not supported at the moment.

op_mode	instrument	description	status
lo	r	pos. data	Tested and functional. D1-D4 contain $\vec{r}$ two times!
ph	a	phase data	*getData.m* freezes!
sa	sax	spin axis	D1-D4 contain no time information. Take it from the position vectors.
nr	b	FGM - 4s	Tested and functional.
hr	bfgm	FGM - 45ms	Tested and functional.
ni	ncis	CIS	Not fully tested. Working for [yymmdd]=020304.
vi	vcis	CIS	Not fully tested. Working for [yymmdd]=020304.
ve	vce	CIS	Not supported at the moment!
wp	whip	Whisper	*Error*: All raws in the bracketed expression must have the same number of columns!
ed	edi	EDI	Not fully tested. Working for [yymmdd]=020304. D4 always zero!
sp	p	Probe pot.	*Error*: All raws in the bracketed expression must have the same number of columns!
ef	e	EFW	Not fully tested. Working for [yymmdd]=020304.

5.2 An Example for 4s data

In order to analyse ***FGM*** data in normal mode ($4s$ data), one must alter the variables *opmode* and *compmode* in *configfile.m* to 'nr' and 'm' for manual mode. For the plot routine it is important to en sure that the variables *plot_mode_normal* and *plot_mode_parnor* are set to 'y' and all other variables are set to their default values. After starting the program by typing *mainprogram* in ©MatLab, it request the day to analyse. In this example the day of interest is the 4^{th} ***of March*** 2002.
Incidentally this is a "famous" day because the trajectory of the cluster satellites led through one of the *polar cusps*. Luckily the incoherent scatter radar ***EISCAT*** also observed the region at this time. (EISCAT is based in Tronso in Norway and one of the receiver stations is located in Kiruna, Sweden.)
After typing the date in the form *020304* the program starts with the computation.

Program messages:

Enter a date to fetch "NORMAL MODE(PP)" - 4sec FGM - data.
Please enter the date [yymmdd]: 020304

Data loaded to RAM. Proceed in execution!

1. data-package sorted. Proceed in execution!

1. data-package ready. Proceed in execution!

Data stored to harddisk. Proceed in execution!

Computation of 1 data-packages ready. End of execution!

Computation time: 0:0

When the task is ready, type in *plotdiagram* and insert the date again. After loading the variables from the HD three windows will appear. One with the magnetic field strength data $\vec{B}$, the second with the normal and the parallel current density $\vec{J}_{av}$ and the third with $\vec{J}_{av}$ in component description. The latter two also contains $\frac{\vec{\nabla}\cdot\vec{B}}{|\vec{\nabla}\times\vec{B}|}$ as an error estimate.

Program messages:

Enter a date to plot "NORMAL MODE(PP)" - 4sec FGM-data.
Please enter the desired day [yymmdd]: 020304
jav loaded to memory!
B1-B4 loaded to memory!
divB loaded to memory!

REPORT FROM THE GETDATA ROUTINE! READ THIS MESSAGE CAREFULLY!
04-Mar-2002 18:0-24:0, Mode:fn, ATTENTION: No B data for SC1 in this time interval! Position data r existent!
04-Mar-2002 18:0-24:0, Mode:fn, ATTENTION: No B data for SC2 in this time interval! Position data r existent!
04-Mar-2002 18:0-24:0, Mode:fn, ATTENTION: No B data for SC3 in this time interval! Position data r existent!
04-Mar-2002 18:0-24:0, Mode:fn, ATTENTION: No B data for SC4 in this time interval! Position data r existent!

The report is stored in the file *0203040000fn.report*.

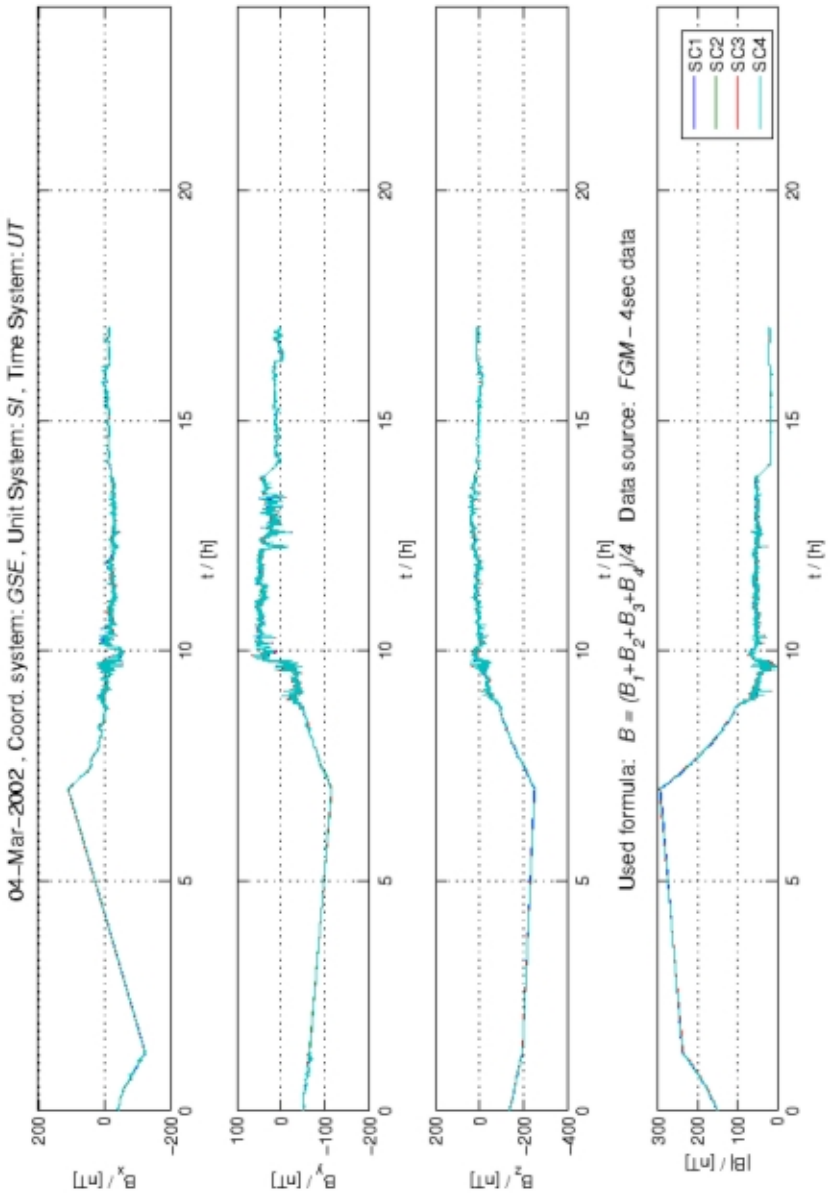

Figure 5.1: The magnetic field strength B.

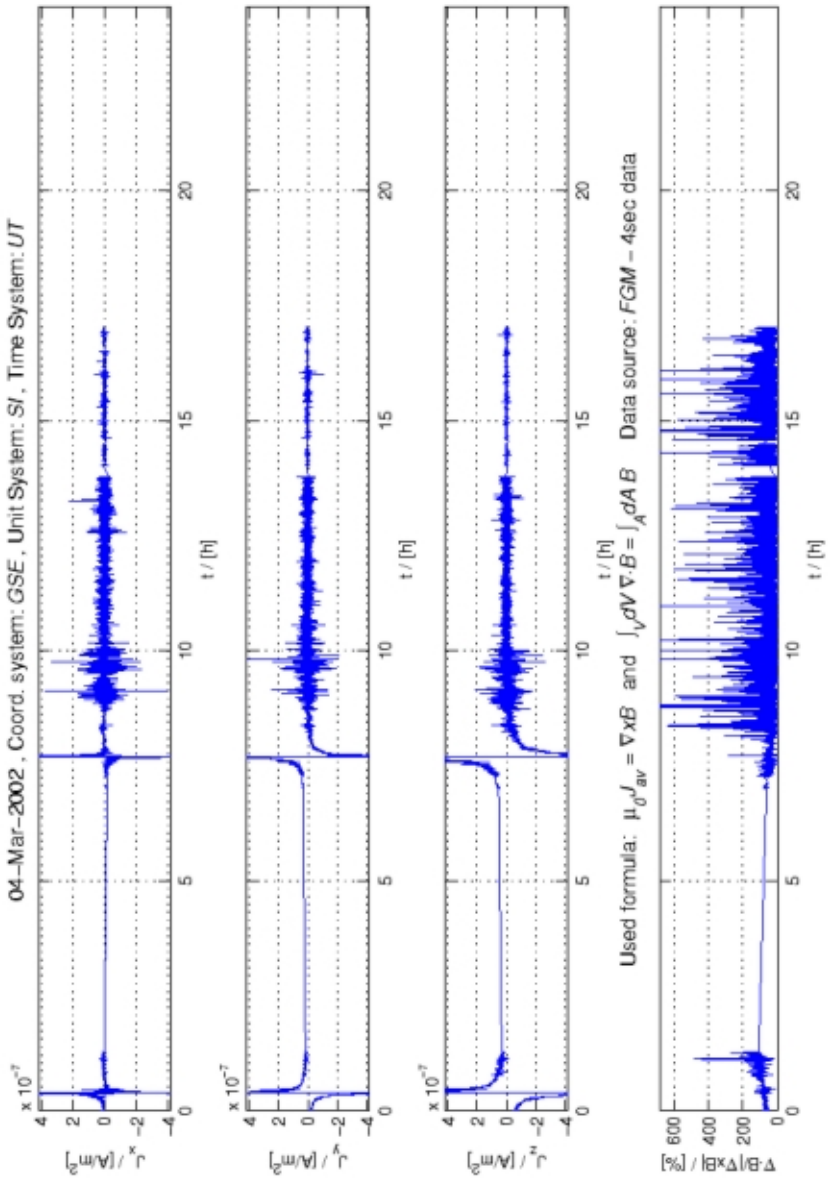

Figure 5.2: Components of the current density J_{av}.

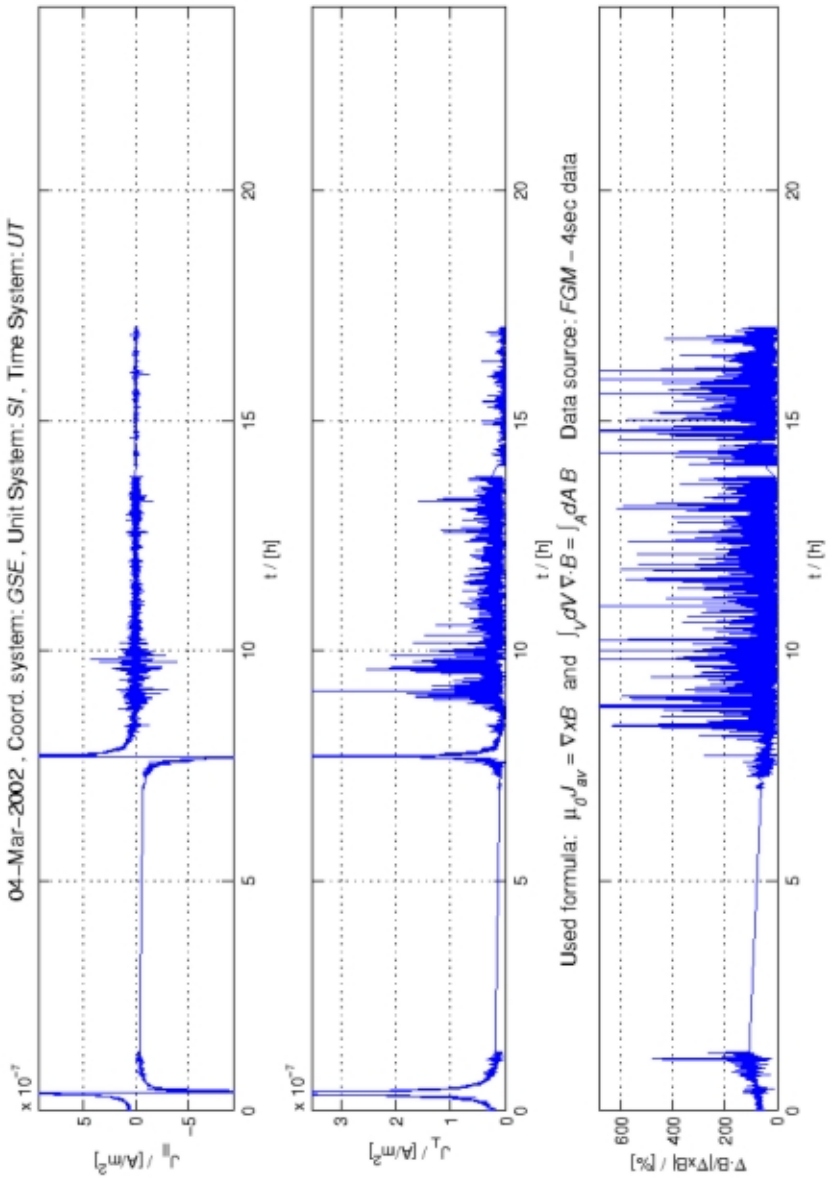

Figure 5.3: Parallel und perpendicular component of J_{av}.

Chapter 6

The Source Code

6.1 The Main Program - *mainprogram.m*

```
%*************************************************************************
% CLUSTER project: Mainprogram                    Autor: Christian Maszl
% Unit System = [SI]                  Mail: e0025754@student.tuwien.ac.at
%*************************************************************************
%
% Version 1.0 - 20.08.2004
%

% Reset Variables
clear all;
tic
%*************************************************************************
% Load configuration data like paths, constants, date, ...
%*************************************************************************

% Load configuration data and check the entries.
    syscheck;

%*************************************************************************
% Read data from datafiles / [B]=nT [r]=km / Coord. system=GSE
% [OK] Write a subroutine that detects the files. Style of the filenames:
% [yymmdd]0000[type].[satellite]
% Example: 0203040000fn.01
%*************************************************************************

% Create filelist of valid files if opmode='a' automatic
    filelist;

% Ask the user a few questions depending on opmode.
    userinput;

%*************************************************************************
% The "BIG LOOP" for all packages or a manual range entry
%*************************************************************************

for lauf=startloop:1:endloop
```

```
if (op_mode=='ve')
% Skip yuri at this place and do it in switch
else
% Get the satellite data using the program of Yuri Kothjaintsev (IRFU).
    [D1, D2, D3, D4, B_ok_r_ff,no_B,no_r,B_ff_r_ok,B_ff_r_ff,cell_name,doit]...
    =yuri(instrument,datapath,tempdirpath,myfilelist{lauf,1},...
         advanced,isdat_db,option);
% Output processing status, remove in the real program
    text1=sprintf('\n%s\n','Data loaded to RAM. Proceed in execution!');
          disp(text1);
% If there is no data in D1 or ... - D4 skip the sorting routine
% and the splitting into B1-B4 and r1-r4 store the not empty data
% to the harddisk.
    if (isempty(find(no_B)) & isempty(find(no_r)))
% [Ok] Sort the data of the four satellites according to the date
% and the satellite numbers
% [Ok] Write a function for the sort routine
        [D1, D2, D3, D4] = sorting(D1,D2,D3,D4,timeacc);
% Output processing status, remove in the real program
        message=['. data-package sorted. Proceed in execution!'];
        text1  =sprintf('%u%s\n',lauf,message);
        disp(text1);
        % Store the time information for the output-datafile.
        T = D1(:,1);
        % Remove compiled functions in yuri.m from system memory (This should
        % solve the problem of "no data" after more than 30 days.)
        clear functions;
        errorflag_nodata=0;
        if (sum(sum(B_ok_r_ff,2),1)==0 & sum(sum(B_ff_r_ok,2),1)==0)
            errorflag_message=0;
        else
            errorflag_message=1;
        end
        clear dimension filename readin i;
    else
        if (doit=='y')
            % doit='y' - Empty array filld with zeros and a time line
            [D1, D2, D3, D4] = sorting(D1,D2,D3,D4,timeacc);
            % Output processing status, remove in the real program
            message=['. data-package sorted. Proceed in execution!'];
            text1  =sprintf('%u%s\n',lauf,message);
            disp(text1);
            message=['Warning: NO DATA ' ...
                      'in at least one of the packages!'] ;
            text1  =sprintf('%s\n',message);
                    disp(text1);
            disp('Proceed in execution!');
            disp(' ');
            errorflag_message=1;
            errorflag_nodata =1;
            clear dimension filename readin i;
        else
            message=['. data-package not sorted. NO DATA ' ...
                      'in at least one of the packages!'] ;
```

```
            text1  =sprintf('%u%s\n',lauf,message);
                    disp(text1);
            disp('Proceed in execution!');
            disp(' ');
            errorflag_message=1;
            errorflag_nodata =1;
            clear dimension filename readin i;
        end
    end
end
% Create report according to the errorflags of routine yuri
if (errorflag_message==1)
    report( B_ok_r_ff,no_B,no_r,B_ff_r_ok,B_ff_r_ff,...
    myfilelist,lauf,advanced,mode,nolog,outputspath,varname);
else
end
% If there is no data in D1 or ... - D4 skip the sorting routine
% and the splitting into B1-B4 and r1-r4 store the not empty data
% to the harddisk.
if (isempty(find(no_B)) & isempty(find(no_r)))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here starts the place for the "PLUGINS". Put your subroutines
% into the desired division. SEE BOTTOM of the mainprogram for
% necessary steps!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    switch op_mode
       case 'nr'
       % FGM -  4 sec data (fluxgate magnetometer)
       % Check also: cell_name for var. included in DX
            jav_fgm;

       case 'hr'
       % FGM - 45msec data (fluxgate magnetometer)
       % Check also: cell_name for var. included in DX
            jav_fgm;

       case 'vi'
       % CIS - vcis data (cluster ion spectrometry)
       % DATAFORMAT: DX=[t VCp1(:,3) t VCh1(:,3) rXx rXy rXz]
       % Check also: cell_name for var. included in DX

       case 've'
       % CIS -  vce data (cluster ion spectrometry)
       % DATAFORMAT: DX=[t VCEp1(:,3) t VCEh1(:,3) rXx rXy rXz]
       % OPERATION ORDER: vcis -¿ b -¿ vce (option='nosave')!
       % Check also: cell_name for var. included in DX

       % Create mB.mat
            yuri('vcis',datapath,tempdirpath,myfilelist{lauf,1},...
                 advanced,isdat_db,1);
            clear functions;
```

```
    yuri('b',datapath,tempdirpath,myfilelist{lauf,1},...
         advanced,isdat_db,1);
    clear functions;
    % Use mB.mat and mCIS.mat to get D1 - D4
    [D1, D2, D3, D4, B_ok_r_ff,no_B,no_r,B_ff_r_ok,B_ff_r_ff] = ...
    yuri('vce',datapath,tempdirpath,myfilelist{lauf,1},...
         advanced,isdat_db,option);
 % Output processing status, remove in the real program
   text1=sprintf('\n%s\n','Data loaded to RAM. Proceed in execution!');
        disp(text1);
% Remove compiled functions in yuri.m from system memory (This should
% solve the problem of "no data" after more than 30 days.)
clear functions;

case 'ni'
% CIS - ncis data (cluster ion spectrometry)
% DATAFORMAT: DX=[t NCp1 t NCh1 rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'ed'
% EDI - data (electron drift)
% DATAFORMAT: DX=[t EDI1(:,3) rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'ef'
% EFW - data (Electric Field and Wave Experiment)
% DATAFORMAT: DX=[t weXp12 t weXp34 rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'lo'
% POSITION data
% DATAFORMAT: DX=[time rXx rXy rXz rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'ph'
% PHASE data
% DATAFORMAT: DX=[time p rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'sa'
% SAX - Spin Axis Orientation
% DATAFORMAT: DX=[o1 o2 o3 rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'wp'
% WHISPER - Wave of High Frequency and Sounder for Probing
%           of Electron Density by Relaxation
% DATAFORMAT: DX=[time wX rXx rXy rXz]
% Check also: cell_name for var. included in DX

case 'sp'
% PROBE potential (ASPOC)
% DATAFORMAT: DX=[t P10Hz1p1 P10Hz1p2 P10Hz1p3 P10Hz1p4
%                 P10Hz1    P1        NVps1(:,2)]
```

```
       % Check also: cell_name for var. included in DX

       otherwise
    end
else
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Report succesfull storage of data to the harddisk
if (lauf~=endloop)
    else
    counter=endloop-startloop+1;
    message=['Computation of ' int2str(counter) ' data-packages ready.'...
             ' End of execution!'];
    text1  =sprintf('\n%s \n',message);
                disp(text1)
    end
    clear error* r* B*;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The end of the "BIG LOOP" for all packages or a manual range entry
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=toc;
% Show the computation time
t=num2str(sec2hm(t));
l=size(t,2);
if (l<=2)
    h='0';
    m=t;
else
    h=t(1,1:l-2);
    m=t(1,l-1:l);
end
    text1=sprintf('\n%s%s%s%s\n','Computation time: ',h,':',m);
          disp(text1);
% Clear data in the temporary directory
delete([tempdirpath '*']);
% Free memory and unlock functions
clear a* co* d* e* h* i* m* n* l* o* s* t* y* v* plot_*;
try
munlock sorting;
munlock curlB;
munlock report;
munlock filelist;
catch
% Don't care if one of the files isn't locked.
end
clear functions;

% How to: Implement your subroutine

% 0) Check the end of yuri.m for supported datatypes. The format of
%    D1-D4 is always in the following way: time data position!
```

```
% 1) D1-D4 contain all the data (time,position, magnetic field ...)
%     DX=[time yourdata rx(position)]

%     =¿ Split the data in suitable variables (choose the proper case!)

%     NOTE: Check also the variable cell_name. It's possible that
%           there are gaps in DX! (time and r is always
%           included. time at the beginning, r at the end)

% 2) Insert the correct variable name in syscheck.m. This name is
%     used when report display errormessages on the screen. eg.
%     FGM: D1=[time B1 r1] =¿ varname='B'

%     =¿ Set varname in syscheck.m

%     (Check also if the other settings in syscheck.m suit you!)

% 3) Write a program that perform your calculations

%     =¿ Put this program in the proper case!

% 4) =¿ Store your data to the harddisk!

% 5) =¿ Write a program the plot the data on the screen

% Have a look at the FGM mode (nr & hr) as an example:
% (This mode is tested and hopefully works in a proper way)

% Steps: configfile -¿ syscheck -¿ yuri -¿ sorting -¿ report
%          jav_fgm     -¿ datatodisk

%*************************************************************************
% SUPPORTED DATA TYPES
%*************************************************************************
% POSITION data [GSE]

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  r               Cell          R1                    : x 4    t x y z        nosave
% Comment: Interpolate with av_interp to an arbitrary timeline.
%*************************************************************************
% PHASE data

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  a               Cell          A1                    : x 2     t a           nosave
% Comment: Interpolate with av_interp to an arbitrary timeline.
%*************************************************************************
% SPIN axis vector [GSE]

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  sax             Cell          SAX1                  : x 3     x y z         nosave
% Comment:
%*************************************************************************
% FGM magnetometer data [GSE+DSI]
```

```
% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  b                Cell          BPP1                    : x 4      t x y z       nosave
% Comment: 4sec data

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  bfgm             Cell          B1                      : x 4      t x y z       nosave
% Comment: 45msec data
%*************************************************************************
% CIS (Cluster Ion Spectrometry [GSE+DSI]

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  ncis             Cell          NCp1                    : x 2     t p            nosave
%                                 NCh1                     : x 2     t h

%  vcis             Cell          VCp1                   : x 4     t x y z        nosave
%                                 VCh1                     : x 4     t x y z

%  vce              Cell          VCEp1                    : x 4     t x y z        nosave
%                                 VCEh1                    : x 4     t x y z
% Comment: VCE will only work if you run VCIS and B before! Note
%          that the option 'nosave' has to be disabled in this
%          case. VCE reads the data from mB.mat and mC.mat
%*************************************************************************
% WHISPER (Wave of HIgh frequency and Sounder for Probing of Electron
%          density by Relaxation)

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  whip             Cell          WHIP1                    : x 2      t w           nosave
%*************************************************************************
% EDI (Electron Drift instrument)

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  edi              Cell          EDI1                    : x 4      t x y z       nosave
%*************************************************************************
% ASPOC (Active Spacecraft Potential Control) -¿ Probe potential

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  p                Cell          P10Hz1p1                : x 2      t x           nosave
%                                 P10Hz1p2                 : x 2      t x
%                                 P10Hz1p3                 : x 2      t x
%                                 P10Hz1p4                 : x 2      t x
%                                 P10Hz1                   : x 2      t x
%                                 P1                      : x 2      t x
%                                 NVps1                    : x 3      t x y
%*************************************************************************
% EFW (Electron Field and Wave instrument)

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  e                Cell          we[X]p12                : x 2      t x           nosave
%                                 we[X]p34                 : x 2      t x
%*************************************************************************
```

6.2 The Control Center - *configfile.m*

```
%*********************************************************************
% CLUSTER project: Config file curlometer          Autor: Christian Maszl
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at
%*********************************************************************

% Version 1.0 - 20.08.2004

%*********************************************************************
% PATH &  DEFINITIONS for INPUT DATA
%*********************************************************************

% Set path variable to the binary data files
    datapath='/data/cluster/DDS/';
% Isdat database string. Default¿ 'disco:10'
    isdat_db='disco:10';
% Select the year to analyse
    year='03';

% Select the mode to analyse: FGM :   'nr' - 4.0sec ... magnetometer
%                                     'hr' - 45msec
%*********************************************************************
% NOTE: Modes below this line are not implemted yet.
%*********************************************************************
%                           CIS:      'vi' - 4.0sec ... ion spectrometry
%                                     've' - 4.0sec
%                                     'ni' - 4.0sec

%                           EDI:      'ed' - 4.0sec ... electron drift

%                           P:        'sp'            ... probe potential

%                           EFW:      'ef'            ... E field and  wave
%                           SAX:      'sa'            ... spin axis orientat
%                           PHASE:    'ph'             ... phase data
%                           LOCATION'lo'               ... position vector
%                           WHISPER:'wp'               ... wave of high freq
%*********************************************************************
    op_mode='sp';

% * for 01,02,03 and 04 ..... DON'T EDIT
    extension='*';

%*********************************************************************
% PATH & DEFINITIONS for OUTPUT DATA
%*********************************************************************

% Type of the output datafiles: '-ascii' or '-mat' = binary data
% (Binary output is faster!)
    datatype='-mat';

% All paths are relative to the path of the m-File directory
```

```
% Logfile-directory (myfilelist, errorlog)
    logfilepath='../logfiles/';
% Output-directory (jav,divB and B in mat or ascii format)
    outputspath='../outputdata/';
% Directory for temporary data
    tempdirpath='../temp/';

%*************************************************************************
% CONSTANTS / MISCELLANEOUS
%*************************************************************************

% Magnetic permeability, Unit System=[SI]
    y0= 1.256637061E-6; %[y0]=Vs/Am=H/m
% Numberformat on the screen
    format long;

%*************************************************************************
% ADVANCED SETTINGS: Program Control
%*************************************************************************

% The default operating mode is to analyse the data of a whole day
% 00:00:00-24:00:00. With these settings you can choose an arbitray
% starttime and duration. DEFAULT: advanced=[0 0 0 86400];

% Format: [hh mm ss (duration)=sec]
    advanced=[00 00 00 86400];
% The interval from the start time must not be longer than the day itself.

% Switch the program to MANUAL mode. This mode is usefull to avoid
% long waiting times (filelist) when you only want to process one
% or two data-packages. DEFAULT: 'a', Manual: 'm'
    compmode='m';

% Disable the logfiles myfilelist.log, error.log, [yymmdd]0000.report
% DEFAULT: 'y', Disable: 'n' (no logfiles)
    nolog   ='y';

 %*************************************************************************
% ADVANCED SETTINGS: Plot control
%*************************************************************************

% Display the x y z component of jav in GSE. DEFAULT='y' (otherwise 'n'=no)
    plot_mode_normal='y';

% Display the normal and the parallel component of the current jav
% DEFAULT='y' (otherwise 'n'=no)
    plot_mode_parnor='y';

% Toggle legend for B1-B4. DEFAULT='y' (otherwise 'n;=no)
    legend_on       ='y';

%*************************************************************************
% INFORMATION ABOUT THE SOFTWARE
%*************************************************************************
```

```
% The following subroutines are necessary for the us of the program:

% AUTOR: Christian Maszl
% mainprogram.m ..... The mainprogram
% syscheck.m ..... Load configfile.m and check for proper
%                  input values.
% filelist.m ..... Creates a list of all available files.
% userinput.m ..... Ask for range to analyse .
% yuri.m ..... Interface to the program getData. Fetch
%                  data from binary files.
% sorting.m ..... Sync. the data of the four satellites
%                  together.
% curlB.m ..... Apply the curlometer method to the data.
% datatodisk.m ..... Store the computed data to the HD.
% logfiles.m ..... Store the filelist and the error log.
% report.m ..... Error messages from the routine getData.

% AUTOR: Yuri Kothyaintsev
% getData.m ..... Fetch data from various files .
% av_interp.m ..... Interpolates data to a certain timeline.
% fromepoch ..... Convert ISDAT time to standard time.
% toepoch ..... Convert to ISDAT epoch
% clusterDB .....
```

6.3 The Security Unit - *syscheck.m*

```
%**************************************************************************
% CLUSTER project: System check                    Autor: Christian Maszl
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at
%**************************************************************************
%
% Version 1.0 - 20.08.2004
%
% This program loads the configfile.m and check if the values in
% the configfile are valid and perform several write tests . It
% also  converts the mode variable of the configfile to the proper
% input parameters for the subroutine getData.
%
% Example: nr to b (normal mode), hr to bfgm (high res. mode)

errorflag=0;

% Error messages
error0=['Please restart the program!'];
error1=['Directory for outputdata not found! Please check the' ...
        ' entry in configfile.m or create it (mkdir)!'];
error2=['Directory of the inputdata not found! Please check the' ...
        ' entry in configfile.m or make sure that the disk is' ...
        ' mounted! '];
error3=['Temporary directory not found! Please check the entry in' ...
        ' configfile.m or create it (mkdir)!'];
error4=['Logfile directory not found! Please check the entry in' ...
        ' configfile.m or create it (mkdir)!'];
error5=['Please enter -mat or -ascii for the' ...
        ' variable datatype in configfile.m (between primes)!'];
error6=['Please enter * for the  variable extension' ...
        ' in configfile.m (between primes)!'];
error7=['Please enter an integer value with TWO digits for the year!' ...
        ' Examples: 01,02,03 ... '];
error8=['Please enter an INTEGER value with two digits for the year!' ...
        ' Examples: 01,02,03 ... '];
error9=['Please enter for mode in configfile.m 2 characters between' ...
        ' primes! Example: fn'];
error10=['Configfile not loaded or incomplete! There are maybe deleted' ...
        ' entries!'];
error11=['Please check if all variable assingments in configfile.m' ...
         ' are between primes! [Exceptions: y0 and advanced] and'];
error12=['Unknown mode! Please check the variable mode in' ...
         ' configfile.m!'];
error13=['Please correct the variable advanced in configfile.m!' ...
         ' Example: [hh mm ss duration]'];
error14=['Please correct the variable advanced in configfile.m! Use' ...
         ' only integer values! Example: [00 00 00 86400]'];
error15=['Please make sure that the components of advanced are only' ...
         ' integer!'];
error16=['Please correct the induction constant y0 to 1.256637061E-6' ...
         ' Vs/Am. Unit System: SI'];
error17=['Please correct the varibale opmode in configfile.m to either' ...
```

```
         ' a for AUTOMATIC or m for MANUAL!'];
error18=['Please correct the varibale nolog in configfile.m to either' ...
         ' y = YES or n = NO logfiles!'];
error19=['You do not have write permission for the directory!'];
error20=['Please change the path or use the chmod command to alter' ...
        ' the permission status!'];
error21=['Please correct the varibale plot_mode_normal in configfile.m '...
         'to either y = YES or n = NO!'];
error22=['Please correct the varibale plot_mode_parnor in configfile.m '...
         'to either y = YES or n = NO!'];
error23=['Please correct the varibale legend_on in configfile.m to either' ...
         ' y = YES or n = NO!'];

% Load the configfile
try
   configfile;
catch
   disp(error11);
   disp(error15);
   disp(error0);
   break;
end

%************************************************************************
% Conversion to satellite instrument
%************************************************************************
op_mode=lower(op_mode);
% hr = high resolution mode - 45msec data
if     (op_mode=='hr')
        % Mode is the parameter in the filenames
        mode      ='fb';
        instrument='bfgm';
        % Prevet Yuri's programm from storing data to the HD
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =8.00e-6; % in [h] - ca. 29msec
        timeres   ='45m';
        datasrc   ='fgm';
        dataname  ='"high resolution mode"';
        varname   ='B';
% nr = normal resolution mode -  4sec data
elseif (op_mode=='nr')
        mode      ='fn';
        instrument='b';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='fgm';
        dataname  ='"normal mode(pp)"';
        varname   ='B';
elseif (op_mode=='vi')
        mode='notavailable';
        instrument='vcis';
```

```
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='vcis - cis';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='ve')
        mode='notavailable';
        instrument='r';
        % Instrument use in the following order vcis - b
        % - vce. Therefore start with r. Have a look at the
        % mainprogram!
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='vce - cis';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
        % Note in order to use this parameter don't set the
        % option 'nosave' for yuri.m
elseif (op_mode=='ni')
        mode='notavailable';
        instrument='ncis';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='ncis - cis';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='ed')
        mode='notavailable';
        instrument='edi';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='edi';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='sp')
        mode='notavailable';
        instrument='p';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='aspoc';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='ef')
        mode='notavailable';
```

```
        instrument='e';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-6; % in [h] - ca.  2sec
        timeres   ='40m';
        datasrc   ='efw';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='sa')
         mode='notavailable';
        instrument='sax';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='ephemeris spin axis orientation';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='wp')
        mode='notavailable';
        instrument='whip';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='whisper';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='ph')
        mode='notavailable';
        instrument='a';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='ephemeris phase';
        dataname  ='"normal mode(pp)"';
        varname   ='X';
elseif (op_mode=='lo')
        mode='notavailable';
        instrument='r';
        option    ='nosave';
        % Set the time accuracy for the sorting routine
        timeacc   =5.50e-4; % in [h] - ca.  2sec
        timeres   ='4';
        datasrc   ='ephemeris position';
        dataname  ='"normal mode(pp)"';
        varname   ='r';
else
% If mode is unknown display error message!
    disp(error12);
    errorflag=1;
end
mode      =lower(mode);
```

```
instrument=lower(instrument);
% The time accuracy "timeacc" effect the sorting routine a lot! A to
% small accuracy lead to high peaks in jav. A to high accuracy cancels
% all the data!
%**************************************************************************

try
% Check directories
if (isdir(outputspath)==1)
else
    disp(error1);
    errorflag=1;
end
if (isdir(datapath)==1)
else
    disp(error2);
    errorflag=1;
end
if (isdir(logfilepath)==1)
else
    disp(error4);
    errorflag=1;
end
if (isdir(tempdirpath)==1)
else
    disp(error3);
    errorflag=1;
end
if (isdir(logfilepath)==1)
else
    disp(error4);
    errorflag=1;
end
catch
    disp(error10);
end
% Check for typos
datatype=lower(datatype);
if (size(datatype,2)==4)
    if (datatype=='-mat')
    else
        disp(error5);
        errorflag=1;
    end
elseif(size(datatype,2)==6)
    if (datatype=='-ascii')
    else
        disp(error5);
        errorflag=1;
    end
else
    disp(error5);
    errorflag=1;
end
```

```
try
    if (extension=='*')
    else
    disp(error6);
    errorflag=1;
    end
catch
    disp(error6);
    errorflag=1;
end
if (isempty(str2num(year))==1)
    disp(error8);
    errorflag=1;
else
    if (size(year,2)~=2)
        disp(error7);
        errorflag=1;
    else
    end
end
if (size(op_mode,2)~=2)
    disp(error9);
    errorflag=1;
else
end
if (size(advanced,2)~=4)
    disp(error13);
    errorflag=1;
else
end
try
    if (isempty(str2num(advanced))==1)
        disp(error14);
        errorflag=1;
    else
    end
catch
    disp('');
end
if (y0~=1.256637061E-6)
    disp(error16);
    errorflag=1;
else
end
compmode=lower(compmode);
if (compmode=='a'| compmode=='m')
else
    disp(error17);
    errorflag=1;
end
nolog =lower(nolog);
if (nolog =='y'| nolog =='n')
else
```

```
    disp(error18);
    errorflag=1;
end
plot_mode_normal =lower(plot_mode_normal);
if (plot_mode_normal =='y'| plot_mode_normal =='n')
else
    disp(error21);
    errorflag=1;
end
plot_mode_parnor =lower(plot_mode_parnor);
if (plot_mode_parnor =='y'| plot_mode_parnor =='n')
else
    disp(error22);
    errorflag=1;
end
legend_on =lower(legend_on);
if (legend_on =='y'| legend_on =='n')
else
    disp(error23);
    errorflag=1;
end
% Check if all path variables end with /
l=size(datapath,2);
if (datapath(1,l)~='/')
    datapath(1,l+1)='/';
else
end
l=size(logfilepath,2);
if (logfilepath(1,l)~='/')
    logfilepath(1,l+1)='/';
else
end
l=size(outputspath,2);
if (outputspath(1,l)~='/')
    outputspath(1,l+1)='/';
else
end
l=size(tempdirpath,2);
if (tempdirpath(1,l)~='/')
    tempdirpath(1,l+1)='/';
else
end
% Check for write permission in temp, log, and outputdata
command_string=['touch ' logfilepath 'write_test'];
if (unix(command_string)==0)
    delete([logfilepath 'write_test']);
else
    message=[error19 logfilepath ' !'];
    disp(message);
    disp(error20);
    errorflag=1;
end
command_string=['touch ' outputspath 'write_test'];
if (unix(command_string)==0)
```

```
    delete([outputspath 'write_test']);
else
    message=[error19 outputspath ' !'];
    disp(message);
    disp(error20);
    errorflag=1;
end
command_string=['touch ' tempdirpath 'write_test'];
if (unix(command_string)==0)
    delete([tempdirpath 'write_test']);
else
    message=[error19 tempdirpath ' !'];
    disp(message);
    disp(error20);
    errorflag=1;
end
% Display restart message
if (errorflag==1)
  disp(error0);
  break;
else
end
clear l error* message command_string;
```

6.4 Search for Data - *filelist.m*

```
%*************************************************************************
% CLUSTER project: Filelist creator                    Autor: Christian Maszl
% Unit System = [SI]                          Mail: e0025754@student.tuwien.ac.at
%*************************************************************************
%
% Version 1.0 - 20.8.2004
%
% This script searches for datafiles defined in the path variable
% datapath and store the filenames in an array. Afterwards the program
% curlB use this files for the calculation of Jav and div(B)av. The
% filenames have to be in the following shape:
%
% [yymmdd]0000[type].[satellite]
% Example: 0203040000fn.01.
%
% These datafiles contains the measuring time, the components of the B
% field and the position vector. It's important that the B field data
% is spin (satellite) compensated!
%
% The valid files (variable: myfilelist) are stored in myfilelist.log
%
% The program also creates the file error.log (variable errorlog). In this
% file will be missing and invalid files reported.

% Lock m-File in memory
mlock;

% Check for operation mode 'a'==automatic / 'm'==manual

if (compmode=='a')
% Plot some infos from configfile.m
dataname=upper(dataname);
datasrc =upper(datasrc);
message =['Searching ' dataname ' - ' timeres 'sec ' datasrc ...
          ' - data for the year 20' year '.'];
         disp(message);
% Read all files with the extension *
filenam1=dir([datapath,'*.',extension]);
% Preallocate array
status      =cell (size(filenam1,1),2);
indexlist1  =zeros(size(filenam1,1),1);
invalidlist1=zeros(size(filenam1,1),1);
check1      =zeros(1,4);
check2      =zeros(1,4);
invalidlist2=zeros(1,1);
indexlist   =zeros(1,1);
datalist    =zeros(1,4);
% Check data for mode, coord.system, timeformat, datatype
for i=1:size(filenam1,1)
    status{i,1}=findstr(filenam1(i,1).name,mode);
    status{i,2}=findstr(filenam1(i,1).name,year);
    % Transformation from cell to matrix
```

```
    check1=status{i,1};
    check2=status{i,2};
    if (isempty(check1)==1 | isempty(check2)==1)
        invalidlist1(i,1)=i;
    else
        if (check1(1,1)==11 & check2(1,1)==1)
           indexlist(i,1)=i;
        else
           invalidlist1(i,1)=i;
        end
    end
end
% Find all nonzero elements of the lists
indexlist   =find(indexlist);
invalidlist1=find(invalidlist1);
% Preallocate arrays
filenam2 =cell(size(indexlist,1),3);
filenam3 =cell(size(indexlist,1),1);
% Write valid filenames into filenam2
for i=1:size(indexlist,1)
    % Valid lines of the input file list
    j=indexlist(i);
    % Extract date, mode, satellite
    filenam2{i,1}=sscanf(filenam1(j,1).name,'%6c',1);
    filenam2{i,2}=sscanf(filenam1(j,1).name,'%*u%2c',1);
         modeselect=['%*u' mode '.%2u' ];
    filenam2{i,3}=sscanf(filenam1(j,1).name,modeselect,1);
         clear modeselect;
    % Store complete filename
    filenam3{i,1}=filenam1(j,1).name;
end
% Sort the datafiles with respect to the date satellite number
dateinfo=zeros(size(filenam2,1),6);
filenam4=cell (size(filenam2,1),3);
filenam5=cell (size(filenam2,1),1);
for i=1:size(filenam2,1)
    % Store year
    dateinfo(i,1)=str2double(sscanf(filenam2{i,1},'%2c',1));
    % Store month
    dateinfo(i,2)=str2double(sscanf(filenam2{i,1},'%*2c%2c',1));
    % Store day
    dateinfo(i,3)=str2double(sscanf(filenam2{i,1},'%*4c%2c',1));
    % Satellite number
    dateinfo(i,4)=filenam2{i,3};
    % Store index
    dateinfo(i,6)=i;
end
% Sort the rows according to the following rule:
% 1) year(1) -¿ 2) month(2) -¿ 3) day(3) -¿ 4)  Satellite number(4)
dateinfo=sortrows(dateinfo,[1 2 3 4]);
% Write sorted data in filenam2 and filenam3
for i=1:size(dateinfo,1)
        filenam4{i,1}=filenam2{dateinfo(i,6),1};
        filenam4{i,2}=filenam2{dateinfo(i,6),2};
```

```
        filenam4{i,3}=filenam2{dateinfo(i,6),3};
        % Store complete filenames
        filenam5{i,1}=filenam3{dateinfo(i,6),1};
end
filenam2=filenam4;
filenam3=filenam5;
clear filenam4 dateinfo;
% Check for the same date and start and end time
lauf = 1;
indexlist2 =zeros(size(filenam2,1),1);
for i=1:size(filenam2,1)-3
    if (filenam2{i,1}  == filenam2{i+1,1} & filenam2{i,1}== ...
        filenam2{i+2,1} & filenam2{i,1} == filenam2{i+3,1})
        % Write match to row, no zero lines in datalist due to lauf
        datalist(lauf,:) =  [i i+1 i+2 i+3];
        lauf=lauf+1;
    else
    end
end
% Create filelist for use in the main program
myfilelist=cell(lauf-1,4);
for i=1:lauf-1
        myfilelist{i,1}=filenam3{datalist(i,1),1};
        myfilelist{i,2}=filenam3{datalist(i,2),1};
        myfilelist{i,3}=filenam3{datalist(i,3),1};
        myfilelist{i,4}=filenam3{datalist(i,4),1};
end
% Compare filenam1(:,1).name with filenam3. Write nonmatching results to
% invalidlist2
lauf    =1;
X       =size(datalist);
elements=X(1,1)*X(1,2);
datalist=reshape(datalist,elements,1);
foundone=0;
for i=1:size(filenam3,1)
    for j=1:length(datalist)
        if (i==datalist(j,1))
            foundone=1;
        else
        end
    end
    if (foundone==0)
        invalidlist2(lauf,1)=i;
        lauf=lauf+1;
    else
    end
    foundone=0;
end
% Use invalidlist1 and 2 to create filelist.log
% Find maximum length of the log file + 2 lines for comments
l=size(invalidlist1,1)+size(invalidlist2,1)+2;
errorlog=cell(l,1);
errorlog{1,1}=['Error in the file name. Files not included' ...
               ' in the calculation!'];
```

```
for i=1:size(invalidlist1,1)
    errorlog{i+1,1}=filenam1(invalidlist1(i),1).name;
end
errorlog{size(invalidlist1,1)+2,1}=['Set of four datafiles' ...
                                    ' not complete!'];
if (invalidlist2(1)~=0)
    for i=1:size(invalidlist2,1)
        errorlog{size(invalidlist1,1)+2+i,1}= ...
        cell2mat(filenam3(invalidlist2(i),:));
    end
else
end
% Delete unnecessary variables
clear  filenam1 filenam2 filenam3 filenam5 foundone i j l lauf;
clear indexlist1 indexlist2 invalidlist1 invalidlist2 datalist;
clear indexlist X elements check1 check2 message status;
% Alert if no data can be found in myfilelist
if (size(myfilelist,1)<1)
    disp(['Error: myfilelist is empty! Check variable mode in' ...
          ' configfile.m! Examples: fn']);
    break;
else
end
% myfilelist contains the filenames for the main program.
% errorlog contains the unused files due to several errors.
% Use for a possible GUI
    if (nolog=='y')
% Store myfilelist and errorlog in files
        logfiles;
    else
    end
% Output processing status
    text1=sprintf('%s \n','File list ready. Proceed in execution!');
          disp(text1);
else
% Plot some infos from configfile.m for the manual mode
dataname=upper(dataname);
datasrc =upper(datasrc);
message =['Enter a date to fetch ' dataname ' - ' timeres 'sec '...
          datasrc ' - data.'];
          disp(message);
end
```

6.4.1 The Logfiles - *logfiles.m*

```
%*************************************************************************
% CLUSTER project: Store logfiles                    Autor: Christian Maszl
% Unit System = [SI]                  Mail: e0025754@student.tuwien.ac.at
%*************************************************************************
%
% Version 1.0 - 20.8.2004
%
```

```
% This subroutine stores the errorlog and myfilelist to the harddisk.

% Create myfilelist.log and store the package names
    savedata=[logfilepath 'myfilelist.log'];
fid=fopen(savedata,'w');
for i=1:size(myfilelist,1)
    place1=cell2mat(myfilelist(i,1));
    place2=cell2mat(myfilelist(i,2));
    place3=cell2mat(myfilelist(i,3));
    place4=cell2mat(myfilelist(i,4));
    fprintf(fid,'%s\t%s\t%s\t%s\n',place1,place2,place3,place4);
end
fclose(fid);
clear place1 place2 place3 place4 fid i;
% Create error.log and store the unvalid package names
    savedata=[logfilepath 'error.log'];
fid2=fopen(savedata,'w');
for i=1:size(errorlog,1)
    place1=cell2mat(errorlog(i,1));
    fprintf(fid2,'%s\r',place1);
end
fclose(fid2);
clear place1 place2 place3 place4 fid2 i errorlog savedata;
```

6.5 Only one Question - *userinput.m*

```
%*************************************************************************
% CLUSTER project: User input                    Autor: Christian Maszl
% Unit System = [SI]                  Mail: e0025754@student.tuwien.ac.at
%*************************************************************************
%
% Version 1.0 - 20.8.2004

errorflag=0;

%*************************************************************************
% Automatic mode user input
%*************************************************************************
if (compmode=='a')
l=size(myfilelist,1);
date1=sscanf(myfilelist{1,1},'%6c',1);
date2=sscanf(myfilelist{l,1},'%6c',1);
% Output number of datapackages (1 package = four satellites)
text1=sprintf('%s %u %s\n','The filelist contains',l,...
              ' datapackages. 1 package = 4 files (satellite 1-4)');
% Output the date of the earliest and the latest measurement
text2=sprintf('%s %6s %s %6s\n','Start date [yymmdd]:',date1,...
              ' -> End date [yymmdd]:',date2);
text3=sprintf('%s %u%s','Please enter a [M]anual range from: 1 to',l,...
              ', view the [F]ilelist or choose [A]ll.');
disp(text1);disp(text2);disp(text3);
user_entry = input('','s') ;
% Show the filelist
if (lower(user_entry)=='f')
   j=1;
   for i=1:size(myfilelist,1)/4
       ausgabe=[  '[' int2str(j  ) '] ' myfilelist{j  ,1} ...
                '  [' int2str(j+1) '] ' myfilelist{j+1,1} ...
                '  [' int2str(j+2) '] ' myfilelist{j+2,1} ...
                '  [' int2str(j+3) '] ' myfilelist{j+3,1}];
       disp(ausgabe);
       j=j+4;
   end
   rest=mod(size(myfilelist,1),4);
   switch rest
     case 1
          ausgabe=[  '[' int2str(j  ) '] ' myfilelist{j  ,1}];
          disp(ausgabe);
     case 2
          ausgabe=[  '[' int2str(j  ) '] ' myfilelist{j  ,1} ...
                   '  [' int2str(j+1) '] ' myfilelist{j+1,1}];
          disp(ausgabe);
     case 3
          ausgabe=[  '[' int2str(j  ) '] ' myfilelist{j  ,1} ...
                   '  [' int2str(j+1) '] ' myfilelist{j+1,1} ...
                   '  [' int2str(j+2) '] ' myfilelist{j+2,1}];
          disp(ausgabe);
     otherwise
```

```
    end
    % Read the userinput from the keyboard
    user_entry = input('Please select [A]ll or [M]anual: ','s') ;
else
end
% Check user input for desired operation mode
for i=1:1:3
    switch lower(user_entry)
        case 'a'
            startloop=1;
            endloop  =l;
            break;
        case 'm'
             sl=input('Start the loop at position: ');
             el=input('End the loop at position:   ');
             if (isstr(sl)==1 | isstr(el)==1)
                 disp('Invalid input arguments!')
             else
                 if (sl>=1 & el<=l & sl<=el)
                     startloop=sl;
                     endloop  =el;
                     break;
                 else
                 disp('Invalid input arguments!')
                 end
             end
        otherwise
            user_entry = input('Please select [A]ll or [M]anual: ','s');
    end
    if (i==3)
        disp('4 times invalid input. Please check the manual!');
        errorflag=1;
    else
    end
end
    if (errorflag==1)
        break;
    else
    end
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Manual mode user input
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
startloop=1;
endloop  =1;
lauf     =1;
errorflag=0;
datum    =input('Please enter the date [yymmdd]: ','s');
for i=1:1:3
    if (isempty(str2num(datum))==1)
        disp('Please enter only integer values! [yymmdd]');
        errorflag=1;
    else
        errorflag=0;
```

```
    end
    if (size(datum,2)~=6)
        disp('Please enter the date in this form [yymmdd] !');
        errorflag=1;
    else
        if (errorflag==0)
            break;
        else
        end
        errorflag=0;
    end
    datum=input('Please enter the date [yymmdd]: ','s');
    if (i==3)
        errorflag=1;
    else
    end
end
if (errorflag==0)
    myfilelist{1,1}=[datum '0000' mode '.01'];
else
    disp('4 times invalid input. Please check the manual!');
    break;
end
end
clear text1 text2 text3 date1 date2 l i j sl el user_entry ausgabe rest;
clear errorflag datum;
```

6.6 The Heart - *yuri.m*

```
function [D1, D2, D3, D4, a_report,b_report,c_report,d_report,e_report,...
          names,doit]=yuri(instrument,datapath,tempdirpath,filename,...
                           advanced,isdat_db,option)
%*********************************************************************
% CLUSTER project: BIN-Files to MatLab              Autor: Christian Maszl
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at
%*********************************************************************
%
% Version 1.0 - 20.8.2004
%
% This script is the interface to the program of Yuri
% Kothjaintsev (IRFU).

%*********************************************************************
% Computation interval in sec (default=86400 - 1day)
t_interval  = advanced(1,4);
% Interpolation method for the position vectors
interpolmeth= 'spline';
% Quantity r for position vectors
quantity    = 'r';
% Store path of the m-Files
path        = pwd;
%*********************************************************************

% datapath MUST point to the /data/cluster directory and NOT to
% /data/cluster/DDS/. getData will also look for files in some other
% directories. Take care that at the end of /data/cluster is NO / !
datapath=strrep(datapath,'/DDS/','');

% Extract the date from the filename and set the starttime of the
% computation (default=0:0:1 -¿ 1sec after midnight)
% Store year
datum(1,1)=2000+str2double(sscanf(filename,'%2c',1));
% Store month
datum(1,2)=str2double(sscanf(filename,'%*2c%2c',1));
% Store day
datum(1,3)=str2double(sscanf(filename,'%*4c%2c',1));
% Store hour     (0=default)
datum(1,4)=advanced(1,1);
% Store minute   (0=default)
datum(1,5)=advanced(1,2);
% Store seconds (1 after midnight=default)
datum(1,6)=advanced(1,3)+1;
%*********************************************************************

% The subroutine getData can't handle large intervals of t_interval
% - Split intervall in 4 pieces (trial).
% [] Rework getData to support large ranges.
t_interval=t_interval/4;
% Find starttime in seconds
starttime(1,1)=datum(1,4)*3600+datum(1,5)*60+datum(1,6);
starttime(1,2)=starttime(1,1)+t_interval+1;
```

```
starttime(1,3)=starttime(1,2)+t_interval+1;
starttime(1,4)=starttime(1,3)+t_interval+1;
% Define error variables for the error report
% 4 satellites, 4 time intervals
a_report      =zeros(4,4);
b_report      =zeros(4,1);
c_report      =zeros(4,1);
d_report      =zeros(4,4);
e_report      =zeros(4,4);
errorcount_B  =0;
errorcount_r  =0;
% Enables sorting no matter if there are empty arrays
doit='n';
% Disables filling with zeros and a time line of empty arrays
skip='n';
% Check for operation mode
if     (size(instrument,2)==4)
    if (instrument=='whip' | instrument=='bfgm')
        modus=1;
    else
    end
    if (instrument=='ncis' | instrument=='vcis')
        modus=2;
    else
    end
    % Set the width of ncis ... to find the time t
    if (instrument=='ncis')
        width=2;
    else
    end
    if (instrument=='vcis')
        width=4;
    else
    end
elseif (size(instrument,2)==3)
    if (instrument=='edi'  | instrument=='sax' )
        modus=1;
        if (instrument=='sax')
        % Don't try to fill empty arrays with time
            skip ='y';
        else
        end
    else
    end
    if (instrument=='vce')
        modus=2;
        % Set the width of vce ... to find the time t
        width=4;
    else
    end
 elseif (size(instrument,2)==1)
    if (instrument=='r'    | instrument=='b'   )
        modus=1;
    else
```

```
    end
    if (instrument=='e')
        modus=2;
    else
    end
    if (instrument=='p')
        modus=3;
    else
    end
    % Set the width of e ... to find the time t
    if (instrument=='e')
        width=2;
    else
    end
    if (instrument=='a')
        modus=1;
    else
    end
else
    modus=0;
end
%*************************************************************************
% Deal with cells with one enclosed matrix
%*************************************************************************
if (modus==1)
% Use Yuri's subroutine to get B1-B4. i choose the satellite
for i=1:1:4
    % The following loop is necessary due to the splitting of the data
    for j=1:1:4
        if (j==4)
        % -5 makes sure that the range stay one second smaller than
        % a whole day.
            t_interval=t_interval-5;
        end
        % Convert the new starttimes in hh mm ss
        hours     =timedim(starttime(1,j),'seconds','hours');
        datum(1,4)=hours-rem(hours  ,1);
        minutes=rem(hours,1)*60;
        datum(1,5)=minutes-rem(minutes,1);
        seconds=rem(minutes,1)*60;
        datum(1,6)=seconds-rem(seconds,1);
        % Use try - catch to intercept errors of GetData
        try
        % Use Yuri's subroutine to get B1-B4.
            eval(['B',int2str(i),'=getData(ClusterDB(isdat_db,datapath,'...
                  'tempdirpath),toepoch(datum),t_interval,',int2str(i), ...
                  ',instrument,option);']);
        % Convert cell to array
            eval(['names(i,:)=B',int2str(i),'{:,1};']);
            eval(['B',int2str(i),'=B',int2str(i),'{:,2};']);
            data_B='y';
        catch
            eval(['B',int2str(i),'=[0 0 0 0];']);
            data_B='n';
```

```
        errorcount_B=errorcount_B+1;
        if (errorcount_B==4)
            names(i,:)='No Data!';
            b_report(i) =1;
        else
        end
    end
    try
    % Use Yuri's subroutine to get r1-r4 for every ~five
    % minutes - interpolate.
        eval(['r',int2str(i),'=getData(ClusterDB(isdat_db,datapath,'...
              'tempdirpath),toepoch(datum),t_interval,',int2str(i), ...
              ',quantity,option);']);
        eval(['r',int2str(i),'=r',int2str(i),'{:,2};']);
        data_r='y';
    catch
        eval(['r',int2str(i),'=[0 0 0 0];']);
        data_r    ='n';
        errorcount_r=errorcount_r+1;
        if (errorcount_r==4)
            c_report(i)=1;
        else
        end
    end
    % Perform error check
    if (data_B=='y' & data_r=='y')
    else
        if     (data_B=='y' & data_r=='n')
                a_report(i,j)=1;
        elseif (data_B=='n' & data_r=='y')
                d_report(i,j)=1;
        elseif (data_B=='n' & data_r=='n')
                e_report(i,j)=1;
        else
        end
    end

    % Interpolate the position vector
    % [z]=av_interp(x,y,method) interpolates x to the time line of y
        eval(['r',int2str(i),'=av_interp(r',int2str(i), ...
             ',B',int2str(i),',interpolmeth);']);
    % Store variables in temporary variable Dtemp
        eval(['Dtemp',int2str(i),int2str(j),'=[B',int2str(i), ...
             ' r',int2str(i),'(:,2:4)];']);
    % Free memory
        clear B* r*;
end
% Construct DX for the sorting routine
eval(['D',int2str(i),'=[Dtemp',int2str(i),int2str(1),'; Dtemp',...
      int2str(i),int2str(2),';Dtemp',int2str(i),int2str(3),    ...
      '; Dtemp',int2str(i),int2str(4),'];']);
% Delete lines with no data
eval(['indexlist=find(D',int2str(i),'(:,1));']);
eval(['D',int2str(i),'=D',int2str(i),'(indexlist,:);']);
```

```
    % Resert for the next satellite
    errorcount_B=0;
    errorcount_r=0;
    clear Dtemp*;
    % Kick getData out of the memory. This should solve the problem
    % with log periods (more than 30 days).
    clear functions;
end
% Clear files in the temp directory
cd(tempdirpath);
delete('tB_*');
% Restore the path to the m-Files
cd(path);
query1=isempty(D1);
query2=isempty(D2);
query3=isempty(D3);
query4=isempty(D4);
% Skip if instrument=sax
if (skip=='y')
else
  % EDI only delivers three arrays, fill the empty one with time
  % and zeros to you the sorting routine.
  if(query1==1 | query4==1)
     if(query1==1 & query4==0)
        D1=zeros(size(D4,1),size(D4,2));
        % Fill with timeline of D4
        D1(:,1)=D4(:,1);
     else
     end
     if(query1==0 & query4==1)
        D4=zeros(size(D1,1),size(D1,2));
        % Fill with timeline of D4
        D4(:,1)=D1(:,1);
     else
     end
  else
  end
  if(query1==2 | query3==1)
     if(query2==1 & query3==0)
        D2=zeros(size(D3,1),size(D3,2));
        % Fill with timeline of D4
        D2(:,1)=D3(:,1);
     else
     end
     if(query2==0 & query3==1)
        D3=zeros(size(D2,1),size(D2,2));
        % Fill with timeline of D4
        D3(:,1)=D2(:,1);
     else
     end
  else
  end
end
if (isempty(D1)==0 | isempty(D2)==0 | isempty(D3)==0 | ...
```

```
    isempty(D4)==0 )
    % Enable sorting again!
    doit='y';
else
    doit='n';
end
else
end
%*************************************************************************
% Deal with cells with two enclosed matrices
%*************************************************************************
if (modus==2)
% Use Yuri's subroutine to get B1-B4. i choose the satellite
for i=1:1:4
    % The following loop is necessary due to the splitting of the data
    for j=1:1:4
        if (j==4)
        % -4 makes sure that the range stay one second smaller than
        % a whole day.
            t_interval=t_interval-5;
        end
        % Convert the new starttimes in hh mm ss
        hours     =timedim(starttime(1,j),'seconds','hours');
        datum(1,4)=hours-rem(hours  ,1);
        minutes=rem(hours,1)*60;
        datum(1,5)=minutes-rem(minutes,1);
        seconds=rem(minutes,1)*60;
        datum(1,6)=seconds-rem(seconds,1);
        % Use try - catch to intercept erros of GetData
        try
        % Use Yuri's subroutine to get B1-B4.
            eval(['B',int2str(i),'=getData(ClusterDB(isdat_db,datapath,'...
                  'tempdirpath),toepoch(datum),t_interval,',int2str(i), ...
                  ',instrument,option);']);
            data_B='y';
        catch
            disp('Error in getData.m!');
            eval(['B',int2str(i),'=[];']);
        end
        % Fill with zeros if empty
        query1=eval(['isempty(B',int2str(i),')']);
        if (query1==1)
            if     (size(instrument,2)==1)
                    if (instrument=='e')
                        eval(['A',int2str(i),'=[0 0];']);
                        eval(['C',int2str(i),'=[0 0];']);
                    else
                    end
            elseif (size(instrument,2)==3)
                    if (instrument=='vce')
                        eval(['A',int2str(i),'=[0 0 0 0];']);
                        eval(['C',int2str(i),'=[0 0 0 0];']);
                    else
                    end
```

```
    elseif (size(instrument,2)==4)
            if     (instrument=='ncis')
                eval(['A',int2str(i),'=[0 0];']);
                eval(['C',int2str(i),'=[0 0];']);
            elseif (instrument=='vcis')
                eval(['A',int2str(i),'=[0 0 0 0];']);
                eval(['C',int2str(i),'=[0 0 0 0];']);
            else
            end
    else
    end
    data_B='n';
    errorcount_B=errorcount_B+1;
    if (errorcount_B==4)
        names{i,1}='No Data!';
        names{i,2}='No Data!';
        b_report(i) =1;
    else
    end
else
    % Convert cell to array
    try
       eval(['names(i,:)=B',int2str(i),'{:,1};']);
    catch
       names{i,1}='No Data!';
       names{i,2}='No Data!';
    end
    try
       eval(['A',int2str(i),'=B',int2str(i),'{:,2};']);
    catch
       eval(['A',int2str(i),'=[];']);
    end
    try
       eval(['C',int2str(i),'=B',int2str(i),'{:,3};']);
    catch
       eval(['C',int2str(i),'=[];']);
    end
end
% Check for empty cell.
query1=eval(['isempty(A',int2str(i),')']);
query2=eval(['isempty(C',int2str(i),')']);
if(query1==1 | query2==1)
   if(query1==1 & query2==0)
      eval(['A',int2str(i),'=zeros(size(C',int2str(i),',1),size(C',int2str(i),',2));'])
      % Write T (time) into the empty cell
      eval(['A',int2str(i),'(:,1)=C',int2str(i),'(:,1);']);
   else
   end
   if(query1==0 & query2==1)
      eval(['C',int2str(i),'=zeros(size(A',int2str(i),',1),size(A',int2str(i),',2));'])
      % Write T (time) into the empty cell
      eval(['C',int2str(i),'(:,1)=A',int2str(i),'(:,1);']);
   else
   end
```

```
    else
     % Find minimum length of A and C and bring both to the same length
       eval(['l=min(size(A',int2str(i),',1),size(C',int2str(i),',1));']);
       eval(['A',int2str(i),'=A',int2str(i),'(1:l,:);']);
       eval(['C',int2str(i),'=C',int2str(i),'(1:l,:);']);
    end
    try
    % Use Yuri's subroutine to get r1-r4 for every five
    % minutes - interpolate.
        eval(['r',int2str(i),'=getData(ClusterDB(isdat_db,datapath,'...
              'tempdirpath),toepoch(datum),t_interval,',int2str(i), ...
              ',quantity,option);']);
        eval(['r',int2str(i),'=r',int2str(i),'{:,2};']);
        data_r='y';
    catch
        eval(['r',int2str(i),'(1,:)=0;']);
        data_r    ='n';
        errorcount_r=errorcount_r+1;
        if (errorcount_r==4)
            c_report(i)=1;
        else
        end
    end
    % Perform error check
    if (data_B=='y' & data_r=='y')
    else
        if     (data_B=='y' & data_r=='n')
                a_report(i,j)=1;
        elseif (data_B=='n' & data_r=='y')
                d_report(i,j)=1;
        elseif (data_B=='n' & data_r=='n')
                e_report(i,j)=1;
        else
        end
    end
    % Interpolate the position vector
    % [z]=av_interp(x,y,method) interpolates x to the time line of y
        eval(['r',int2str(i),'=av_interp(r',int2str(i), ...
             ',A',int2str(i),',interpolmeth);']);
    % Store variables in temporary variable Dtemp
        eval(['Dtemp',int2str(i),int2str(j),'=[A',int2str(i), ...
             ' C',int2str(i), ' r',int2str(i),'(:,2:4)];']);
    % Free memory
        clear B* r*;
end
% Construct DX for the sorting routine
eval(['D',int2str(i),'=[Dtemp',int2str(i),int2str(1),'; Dtemp',...
      int2str(i),int2str(2),';Dtemp',int2str(i),int2str(3),    ...
      '; Dtemp',int2str(i),int2str(4),'];']);
% Delete lines with no data
eval(['indexlist=find(D',int2str(i),'(:,1));']);
eval(['D',int2str(i),'=D',int2str(i),'(indexlist,:);']);
% Resert for the next satellite
errorcount_B=0;
```

```
    errorcount_r=0;
    clear Dtemp*;
    % Kick getData out of the memory. This should solve the problem
    % with log periods (more than 30 days).
    clear functions;
 end
% Clear files in the temp directory
cd(tempdirpath);
delete('tB_*');
% Restore the path to the m-Files
cd(path);
query1=isempty(D1);
query2=isempty(D2);
query3=isempty(D3);
query4=isempty(D4);
% NCIS sometimes delivers three arrays, fill the empty one with time
% and zeros to you the sorting routine.
if(query1==1 | query4==1)
   if(query1==1 & query4==0)
      D1=zeros(size(D4,1),size(D4,2));
      % Fill with timeline of D4
      D1(:,1)=D4(:,1);
   else
   end
   if(query1==0 & query4==1)
      D4=zeros(size(D1,1),size(D1,2));
      % Fill with timeline of D4
      D4(:,1)=D1(:,1);
   else
   end
else
end
if(query2==1 | query3==1)
   if(query2==1 & query3==0)
      D2=zeros(size(D3,1),size(D3,2));
      % Fill with timeline of D4
      D2(:,1)=D3(:,1);
   else
   end
   if(query2==0 & query3==1)
      D3=zeros(size(D2,1),size(D2,2));
      % Fill with timeline of D4
      D3(:,1)=D2(:,1);
   else
   end
else
end
if (isempty(D1)==0 | isempty(D2)==0 | isempty(D3)==0 | ...
    isempty(D4)==0 )
    % Enable sorting again!
    doit='y';
else
    doit='n';
end
```

```
else
end
%**************************************************************************
% Deal with cells with seven enclosed matrices
%**************************************************************************
if (modus==3)
% Use Yuri's subroutine to get B1-B4. i choose the satellite
for i=1:1:4
    % The following loop is necessary due to the splitting of the data
    for j=1:1:4
        if (j==4)
        % -4 makes sure that the range stay one second smaller than
        % a whole day.
            t_interval=t_interval-5;
        end
        % Convert the new starttimes in hh mm ss
        hours     =timedim(starttime(1,j),'seconds','hours');
        datum(1,4)=hours-rem(hours  ,1);
        minutes=rem(hours,1)*60;
        datum(1,5)=minutes-rem(minutes,1);
        seconds=rem(minutes,1)*60;
        datum(1,6)=seconds-rem(seconds,1);
        % Use try -¿ catch to intercept erros of GetData
        try
        % Use Yuri's subroutine to get B1-B4.
            eval(['B',int2str(i),'=getData(ClusterDB(isdat_db,datapath,'...
                  'tempdirpath),toepoch(datum),t_interval,',int2str(i), ...
                  ',instrument,option);']);
            data_B='y';
        catch
            disp('Error in getData.m!');
            eval(['B',int2str(i),'=[];']);
        end
        query1=eval(['isempty(B',int2str(i),')']);
        if (query1==1)
            eval(['A',int2str(i),'=[0 0];']);
            eval(['C',int2str(i),'=[0 0];']);
            eval(['D',int2str(i),'=[0 0];']);
            eval(['E',int2str(i),'=[0 0];']);
            eval(['F',int2str(i),'=[0 0];']);
            eval(['G',int2str(i),'=[0 0];']);
            eval(['H',int2str(i),'=[0 0 0];']);
            data_B='n';
            errorcount_B=errorcount_B+1;
            if (errorcount_B==4)
                for k=1:1:7
                    names{i,k}='No Data!';
                end
                b_report(i) =1;
            else
            end
        else
        % Convert cell to array
            try
```

```
        eval(['names(i,:)=B',int2str(i),'{:,1};']);
    catch
        for k=1:1:7
            names{i,k}='No Data!';
        end
    end
    try
        eval(['A',int2str(i),'=B',int2str(i),'{:,2};']);
    catch
        eval(['A',int2str(i),'=[];']);
    end
    try
        eval(['H',int2str(i),'=B',int2str(i),'{:,3};']);
    catch
        eval(['H',int2str(i),'=[];']);
    end
    try
        eval(['C',int2str(i),'=B',int2str(i),'{:,4};']);
    catch
        eval(['C',int2str(i),'=[];']);
    end
    try
        eval(['D',int2str(i),'=B',int2str(i),'{:,5};']);
    catch
        eval(['D',int2str(i),'=[];']);
    end
    try
        eval(['E',int2str(i),'=B',int2str(i),'{:,6};']);
    catch
        eval(['E',int2str(i),'=[];']);
    end
    try
        eval(['F',int2str(i),'=B',int2str(i),'{:,7};']);
    catch
        eval(['F',int2str(i),'=[];']);
    end
    try
        eval(['G',int2str(i),'=B',int2str(i),'{:,8};']);
    catch
        eval(['G',int2str(i),'=[];']);
    end
end
% Check for empty cells
query1=eval(['isempty(A',int2str(i),')']);
query2=eval(['isempty(H',int2str(i),')']);
query3=eval(['isempty(C',int2str(i),')']);
query4=eval(['isempty(D',int2str(i),')']);
query5=eval(['isempty(E',int2str(i),')']);
query6=eval(['isempty(F',int2str(i),')']);
query7=eval(['isempty(G',int2str(i),')']);
if ( query1==1 | query2==1 |query3==1 | query4==1 | query5==1 ...
   | query6==1 | query7==1)
% [] Find the empty arrays and fill them with a timeline
```

```
else
% Find minimum length of A-G and bring all to the same size
   eval(['l=min([size(A',int2str(i),',1) size(H',int2str(i),',1) '...
               ' size(C',int2str(i),',1) size(D',int2str(i),',1) '...
               ' size(E',int2str(i),',1) size(F',int2str(i),',1) '...
               ' size(G',int2str(i),',1)]);']);
   eval(['A',int2str(i),'=A',int2str(i),'(1:l,:);']);
   eval(['H',int2str(i),'=H',int2str(i),'(1:l,:);']);
   eval(['C',int2str(i),'=C',int2str(i),'(1:l,:);']);
   eval(['D',int2str(i),'=D',int2str(i),'(1:l,:);']);
   eval(['E',int2str(i),'=E',int2str(i),'(1:l,:);']);
   eval(['F',int2str(i),'=F',int2str(i),'(1:l,:);']);
   eval(['G',int2str(i),'=G',int2str(i),'(1:l,:);']);
end
try
% Use Yuri's subroutine to get r1-r4 for every five
% minutes - interpolate.
    eval(['r',int2str(i),'=getData(ClusterDB(isdat_db,datapath,'...
          'tempdirpath),toepoch(datum),t_interval,',int2str(i), ...
          ',quantity,option);']);
    eval(['r',int2str(i),'=r',int2str(i),'{:,2};']);
    data_r='y';
catch
    eval(['r',int2str(i),'(1,:)=0;']);
    data_r    ='n';
    errorcount_r=errorcount_r+1;
    if (errorcount_r==4)
        c_report(i)=1;
    else
    end
end
% Perform error check
if (data_B=='y' & data_r=='y')
else
    if     (data_B=='y' & data_r=='n')
            a_report(i,j)=1;
    elseif (data_B=='n' & data_r=='y')
            d_report(i,j)=1;
    elseif (data_B=='n' & data_r=='n')
            e_report(i,j)=1;
    else
    end
end

% Interpolate the position vector
% [z]=av_interp(x,y,method) interpolates x to the time line of y
    eval(['r',int2str(i),'=av_interp(r',int2str(i), ...
         ',A',int2str(i),',interpolmeth);']);
% Store variables in temporary variable Dtemp
    eval(['Dtemp',int2str(i),int2str(j),'=[A',int2str(i), ...
         ' H',int2str(i), ' C',int2str(i),...
         ' D',int2str(i), ' E',int2str(i),...
         ' F',int2str(i), ' G',int2str(i),...
          ' r',int2str(i),'(:,2:4)];']);
```

```
        % Free memory
            clear B* r*;
    end
    % Construct DX for the sorting routine
    eval(['D',int2str(i),'=[Dtemp',int2str(i),int2str(1),'; Dtemp',...
          int2str(i),int2str(2),';Dtemp',int2str(i),int2str(3),    ...
          '; Dtemp',int2str(i),int2str(4),'];']);
    % Delete lines with no data
    eval(['indexlist=find(D',int2str(i),'(:,1));']);
    eval(['D',int2str(i),'=D',int2str(i),'(indexlist,:);']);
    % Resert for the next satellite
    errorcount_B=0;
    errorcount_r=0;
    clear Dtemp*;
    % Kick getData out of the memory. This should solve the problem
    % with log periods (more than 30 days).
    clear functions;
 end
% Clear files in the temp directory
cd(tempdirpath);
delete('tB_*');
% Restore the path to the m-Files
cd(path);
query1=isempty(D1);
query2=isempty(D2);
query3=isempty(D3);
query4=isempty(D4);
% Sometimes delivers three arrays, fill the empty one with time
% and zeros to you the sorting routine.
if(query1==1 | query4==1)
   if(query1==1 & query4==0)
      D1=zeros(size(D4,1),size(D4,2));
      % Fill with timeline of D4
      D1(:,1)=D4(:,1);
   else
   end
   if(query1==0 & query4==1)
      D4=zeros(size(D1,1),size(D1,2));
      % Fill with timeline of D4
      D4(:,1)=D1(:,1);
   else
   end
else
end
if(query2==1 | query3==1)
   if(query2==1 & query3==0)
      D2=zeros(size(D3,1),size(D3,2));
      % Fill with timeline of D4
      D2(:,1)=D3(:,1);
   else
   end
   if(query2==0 & query3==1)
      D3=zeros(size(D2,1),size(D2,2));
      % Fill with timeline of D4
```

```
      D3(:,1)=D2(:,1);
   else
   end
else
end
if (isempty(D1)==0 | isempty(D2)==0 | isempty(D3)==0 | ...
    isempty(D4)==0 )
    % Enable sorting again!
    doit='y';
else
    doit='n';
end
else
end
%**************************************************************************
% Here start the part that is equal for all datatypes
%**************************************************************************
% Convert the ISDAT epoch in hours of the specific day
for i=1:1:4
  eval(['T',int2str(i),'=fromepoch(D',int2str(i),'(:,1));']);
  eval(['dayhour=T',int2str(i),'(:,4)+T',int2str(i),'(:,5)/60+T', ...
        int2str(i),'(:,6)/3600;']);
  eval(['D',int2str(i),'(:,1)=dayhour;']);

% Yuri's program mix the times when you calculate over a whole day
% -¿ sort the rows
   eval(['D',int2str(i),'=sortrows(D',int2str(i),',1);']);
end
if (modus==2)
    for i=1:1:4
        eval(['T',int2str(i),'=fromepoch(D',int2str(i),'(:,width+1));']);
        eval(['dayhour=T',int2str(i),'(:,4)+T',int2str(i),'(:,5)/60+T', ...
        int2str(i),'(:,6)/3600;']);
        eval(['D',int2str(i),'(:,width+1)=dayhour;']);
    end
else
end

%**************************************************************************
% SUPPORTED DATA TYPES
%**************************************************************************
% POSITION data [GSE]

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  r               Cell         R1                    : x 4    t x y z       nosave
% Comment: Interpolate with av_interp to an arbitrary timeline.
%**************************************************************************
% PHASE data

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  a               Cell         A1                    : x 2     t a           nosave
% Comment: Interpolate with av_interp to an arbitrary timeline.
%**************************************************************************
% SPIN axis vector [GSE]
```

```
% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  sax             Cell        SAX1                  : x 3     x y z         nosave
% Comment:
%*************************************************************************
% FGM magnetometer data [GSE+DSI]

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  b               Cell        BPP1                  : x 4     t x y z       nosave
% Comment: 4sec data

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  bfgm            Cell        B1                    : x 4     t x y z       nosave
% Comment: 45msec data
%*************************************************************************
% CIS (Cluster Ion Spectrometry [GSE+DSI]

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  ncis            Cell        NCp1                  : x 2    t p            nosave
%                              NCh1                   : x 2    t h

%  vcis            Cell        VCp1                 : x 4    t x y z        nosave
%                              VCh1                   : x 4    t x y z

%  vce             Cell        VCEp1                 : x 4    t x y z        nosave
%                              VCEh1                  : x 4    t x y z
% Comment: VCE will only work if you run VCIS and B before! Note
%          that the option 'nosave' has to be disabled in this
%          case. VCE reads the data from mB.mat and mC.mat
%*************************************************************************
% WHISPER (Wave of HIgh frequency and Sounder for Probing of Electron
%          density by Relaxation)

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  whip            Cell        WHIP1                  : x 2     t w           nosave
%*************************************************************************
% EDI (Electron Drift instrument)

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  edi             Cell        EDI1                  : x 4     t x y z       nosave
%*************************************************************************
% ASPOC (Active Spacecraft Potential Control) -¿ Probe potential

% [parameter]    [output]    [names in {1,1}]    [size]    [format]      [option]
%  p               Cell        P10Hz1p1              : x 2     t x           nosave
%                              P10Hz1p2               : x 2     t x
%                              P10Hz1p3               : x 2     t x
%                              P10Hz1p4               : x 2     t x
%                              P10Hz1                 : x 2     t x
%                              P1                    : x 2     t x
%                              NVps1                  : x 3     t x y
%*************************************************************************
% EFW (Electron Field and Wave instrument)
```

```
% [parameter]     [output]     [names in {1,1}]     [size]     [format]       [option]
%   e                Cell          we[X]p12                 : x 2       t x             nosave
%                                  we[X]p34                 : x 2       t x
%*************************************************************************
```

6.7 Synchronise the Satellites - *sorting.m*

```
function [D1, D2, D3, D4] = sorting(D1, D2, D3, D4, accuracy)
%*************************************************************************
% CLUSTER project: Magnetometerdata sorter          Autor: Christian Maszl *
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at *
%*************************************************************************
%
% Version 1.0 - 20.8.2004
%
% 31.8.2004: Enhanced the sorting accuracy (line 139)
%
% This script sort the magnetometer data of the four satellites according
% to the time. "Accuracy" set the maximum time difference between the
% measurements of one "event".

% Lock function in memory
% mlock;

searched1 = 1; searched2 = 1; searched3 = 1;
k = 1; l = 0; setmatch1=0; setmatch2=0;setmatch3=0; startnext=1;
a = 0; b = 0; c = 0; d = 0; n=0; j1=1; j2=1; j3=1;
%Find the minimum length of the input data
mindim=min([size(D1,1) size(D2,1) size(D3,1) size(D4,1)]);
% Preallocate output arrays
    X1 =zeros(mindim,7);
    X2 =zeros(mindim,7);
    X3 =zeros(mindim,7);
    X4 =zeros(mindim,7);
    X  =zeros(mindim,4);
% [Ok] Find the data list with the earliest start value
% Permutation of indices 1 to 4
P = perms([1 2 3 4]);
for i=1:24
    if (eval(['D',int2str(P(i,1)),'(1,1)']) <= ...
        eval(['D',int2str(P(i,2)),'(1,1)']) &  ...
        eval(['D',int2str(P(i,1)),'(1,1)']) <= ...
        eval(['D',int2str(P(i,3)),'(1,1)']) &  ...
        eval(['D',int2str(P(i,1)),'(1,1)']) <= ...
        eval(['D',int2str(P(i,4)),'(1,1)']) )
        % Name of the data file with the earliest start value = a
        a = P(i,1); b = P(i,2); c = P(i,3); d = P(i,4); break;
    else
    end
end
P2 = perms([b c d]);
for j=1:6
    if (eval(['D',int2str(P2(j,1)),'(1,1)']) <= ...
        eval(['D',int2str(P2(j,2)),'(1,1)']) &  ...
        eval(['D',int2str(P2(j,1)),'(1,1)']) <= ...
        eval(['D',int2str(P2(j,3)),'(1,1)']) )
        b = P2(j,1); c = P2(j,2); d = P2(j,3); break;
    else
    end
```

```
end
if (eval(['D',int2str(c),'(1,1)']) <= ...
    eval(['D',int2str(d),'(1,1)']) )
else
    x = c; c = d; d = x;
end
% Construct arrays for search
% Array with the latest start time X4
% Row X3smallerX1smallerX2smallerX4
% Write data into X1-4
    eval(['X4=D',int2str(d),';']);
    eval(['X3=D',int2str(a),';']);
    eval(['X2=D',int2str(c),';']);
    eval(['X1=D',int2str(b),';']);
% Length of the input list's
    enda=eval(['size(D',int2str(a),',1);']);
    endb=eval(['size(D',int2str(b),',1);']);
    endc=eval(['size(D',int2str(c),',1);']);
    endd=eval(['size(D',int2str(d),',1);']);
    clear P P2 x D1 D2 D3 D4;
% Perform search with the latest search time
for i=startnext:endd
     for j1=searched1:endc
         if (abs(X4(i,1)-X2(j1,1))<accuracy)
            % First hit in the if statement
                firsthit=n+1;
            % Set index to which all others will be compared
                if (firsthit==1 & startnext==1)
                    startnext=i;
                else
                end
            % Decrease search length
                searched1 = j1;
            % Found a match
                setmatch1 = 1;
            % Break match found
                break;
         else
            % Found no match
                setmatch1 = 0;
         end
         % Prevent scanning the whole list when no match can be found
         if (X4(i,1)+accuracy<X2(j1,1))
             break;
         else
         end
     end
     for j2=searched2:endb
         if (abs(X4(i,1)-X1(j2,1))<accuracy)
            % Decrease search length
                searched2=j2;
            % Found a match
                setmatch2=1;
            % Break match found
```

```
            break;
    else
        % Found no match
            setmatch2=0;
    end
    % Prevent scanning the whole list when no match can be found
    if (X4(i,1)+accuracy<X1(j2,1))
         break;
    else
    end
end
for j3=searched3:enda
     if (abs(X4(i,1)-X3(j3,1))<accuracy)
        % Decrease search lengt
            searched3=j3;
        % Found a match
            setmatch3=1;
        % Break match found
            break;
    else
        % Found no match
            setmatch3=0;
    end
    % Prevent scanning the whole list when no match can be found
    if (X4(i,1)+accuracy<X3(j3,1))
        break;
    else
    end
end
if (setmatch1 & setmatch2 & setmatch3)
  try
  %*****************************
  % Enhance the sorting accuracy
  %*****************************
  X41 =X4(i ,1)-X1(j1,1);
  X42 =X4(i ,1)-X2(j2,1);
  X43 =X4(i ,1)-X3(j3,1);
  X31 =X3(j3,1)-X1(j1,1);
  X32 =X3(j3,1)-X2(j2,1);
  X21 =X2(j2,1)-X1(j1,1);
  % Find maximum error
  pp1=max(abs([X31 X32 X21 X41 X42 X43]));
  if (pp1>accuracy)
    % Check for signs of the differences
    sX41=sign(X41);
    sX42=sign(X42);
    sX43=sign(X43);
    stot=sX41+sX42+sX43;
    if (sign(stot)==-1)
      % Change data with + sign to -
      if (abs(X41)>(accuracy/2))
         if (sX41==-1)
         else
             j1=j1+1;
```

```
            end
        else
        % This is always a "good" value
        end
        if (abs(X42)>(accuracy/2))
            if (sX42==-1)
            else
                j2=j2+1;
            end
        else
        % This is always a "good" value
        end
        if (abs(X43)>(accuracy/2))
            if (sX43==-1)
            else
                j3=j3+1;
            end
        else
        % This is always a "good" value
        end
    else
        % Change data with - sign to +
        if (abs(X41)>(accuracy/2))
            if (sX41==1)
            else
                j1=j1-1;
            end
        else
        % This is always a "good" value
        end
        if (abs(X42)>(accuracy/2))
            if (sX42==1)
            else
                j2=j2-1;
            end
        else
        % This is always a "good" value
        end
        if (abs(X43)>(accuracy/2))
            if (sX43==1)
            else
                j3=j3-1;
            end
        else
        % This is always a "good" value
        end
     end
     else
     end
     catch
     end
     % Store matching lines
          X(k,:)=[i j1 j2 j3];
        % Increment line pointer
```

```
            k = k+1;
    elseif (j1==endc |  j2==endb | j3==enda)
          break;
    else
    end
end
% Delete dispensable variables
    clear j1 j2 j3 i setmatch1 setmatch2 setmatch3;
    clear searched1 searched2 searched3;
    clear con1 con2 con3 con4;
    clear D1 D2 D3 D4 j k l mindim;
    clear k1 k2 k3 firsthit startnext;
    clear enda endb endc endd;
% Find "nonzero" length of the X array
    n=size(find(X(:,1)),1);
% Store these values in an data array
    DATA(:,:)=X(1:n,:);
% Use data array to extract correct measurements
% Row X3¡X1¡X2¡X4
% Write data into D1-4
    eval(['D',int2str(d),'=X4(DATA(:,1),:);']);
    eval(['D',int2str(a),'=X3(DATA(:,4),:);']);
    eval(['D',int2str(c),'=X2(DATA(:,2),:);']);
    eval(['D',int2str(b),'=X1(DATA(:,3),:);']);
% Clear all except the sortet data for the main programm
    clear DATA a b c d con1 con2 con3 con4 ;
    clear X X1 X2 X3 X4 n accuracy;
```

6.8 The Sneak - *report.m*

```
function report( B_ok_r_ff,no_B,no_r,B_ff_r_ok,B_ff_r_ff,myfilelist,...
                 lauf,advanced,mode,nolog,outputspath,varname)
%************************************************************************
% CLUSTER project: Execution reports                Autor: Christian Maszl
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at
%************************************************************************
%
% Version 1.0 - 20.8.2004
%
% This subroutine stores the errors during the execution of yuri.m
% in the output directory (*.report) . Check this file always
% before you start to work with the data! It might happen that
% there exists B field data and no position etc.

% Create logfiles YES='y' / NO='n'
if (nolog=='y')
% Get date from the filename
datum=myfilelist{lauf,1};
datum=datum(1,1:6);
% Create date string. Example: 25-Jan-2002
year =str2num(datum(1,1:2));
month=str2num(datum(1,3:4));
day  =str2num(datum(1,5:6));
datum=datestr(datenum([2000+year month day]));
%************************************************************************
% Find time intervals (computation is splited)
%************************************************************************
t_interval=advanced(1,4);
% Store hours in sec     (0=default)
zeit(1,1) =advanced(1,1)*3600;
% Store minutes in sec   (0=default)
zeit(1,2) =advanced(1,2)*60;;
% Store seconds (1sec after midnight=default)
zeit(1,3) =advanced(1,3);
% Data is splited into four 6 hour parts.
t_interval=t_interval/4;
% Find starttime in seconds
starttime(1,1)=zeit(1,1);
starttime(1,2)=starttime(1,1)+t_interval+1;
starttime(1,3)=starttime(1,2)+t_interval+1;
starttime(1,4)=starttime(1,3)+t_interval+1;
starttime(1,5)=starttime(1,4)+t_interval-3;
% Convert into hm time format
starttime=sec2hm(starttime);
% Extract hours and minutes
for i=1:1:5
t=num2str(starttime(1,i));
l=size(t,2);
    if (l<=2)
        hour(i)  =0;
        minute(i)=str2num(t);
    else
```

```
        hour(i)  =str2num(t(1,1:l-2));
        minute(i)=str2num(t(1,l-1:l));
    end
end
%*****************************************************************************
% Check for no data in intervals
%*****************************************************************************
linefeed=1;
for i=1:1:4
    if (no_B(i)==1)
    % Dispay message for the whole day if true
message=[datum  ' ' int2str(hour(1)) ':' int2str(minute(1)) '->' ...
         int2str(hour(5)) ':' int2str(minute(5))  ', Mode: ' ...
         mode ', MISSING data ' varname int2str(i) ' for SC' int2str(i)...
         ' for the whole day!'];
         error_report(linefeed,1:size(message,2))=message;
             linefeed=linefeed+1;
    else
    for j=1:1:4
        % Make sure that r and B are not both missing
        if (B_ff_r_ff(i,j)==0)
            % Check for each time interval
            if (B_ff_r_ok(i,j)==1)
            message=[datum ' ' int2str(hour(j)) ':' int2str(minute(j)) ...
                     '->' int2str(hour(j+1)) ':' int2str(minute(j+1))  ...
                     ', Mode: ' mode ', ATTENTION: No ' varname ...
                     ' data for SC' int2str(i) ...
                     ' in this time interval!  Position data r existent!'];
            error_report(linefeed,1:size(message,2))=message;
                        linefeed=linefeed+1;
            else
            end
        else

        end

    end
    end
    % Display message for the whole day if true
    if (no_r(i)==1)
message=[datum  ' ' int2str(hour(1)) ':' int2str(minute(1)) '->' ...
         int2str(hour(5)) ':' int2str(minute(5))  ', Mode: ' ...
         mode ', MISSING position data r' int2str(i) ' for SC' ...
         int2str(i) ' for the whole day!'];
         error_report(linefeed,1:size(message,2))=message;
             linefeed=linefeed+1;
    else
    for j=1:1:4
        % Make sure that r and B are not both missing
        if (B_ff_r_ff(i,j)==0)
            % Check for each time interval
            if (B_ok_r_ff(i,j)==1)
            message=[datum ' ' int2str(hour(j)) ':' int2str(minute(j)) '->' ...
                     int2str(hour(j+1)) ':' int2str(minute(j+1))  ', Mode: '...
```

```
                    mode ', ATTENTION: No position data for SC' int2str(i) ...
                    ' in this time interval! ' varname ' data existent!'];
            error_report(linefeed,1:size(message,2))=message;
                        linefeed=linefeed+1;
            else
            end
        else
        end
    end
    end
end
% Take a name of the datapackage and replace the satellite number with
% .report
    satz    =strrep(myfilelist{lauf,1},'.01','.report');
    filename=[outputspath satz];
% Save error_report to HD
    save(filename,'error_report','-mat');
else
end
```

6.9 The Working Horse - *jav_fgm.m*

```
%************************************************************************
% CLUSTER project: Curlometer method                Autor: Christian Maszl
% Formulas: y0*Jav*(r1a x r1b) = dB1a*r1b-dB1b*r1a
%           div B*—r1a*(r1b x r1c)— = —Sum(cyclic) dB1a*(r1b x r1c)—
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at
%************************************************************************
%
% Version 1.0 - 20.8.2004
%
% [Ok] Split the input data into B1,B2,B3,B4 and r1,r2,r3,r4;
for i=1:1:4
    eval(['B',int2str(i),'=D',int2str(i),'(:,2:4);']);
    eval(['r',int2str(i),'=D',int2str(i),'(:,5:7);']);
end
clear D1 D2 D3 D4;
%************************************************************************
% Perform calculations (see tetrahedron.jpg for details about
% geometrical coherences).
%************************************************************************

% [Ok] Write a function faster execution
[jav,divB] = curlB(B1,B2,B3,B4,r1,r2,r3,r4,y0);
% Output processing status
message=['. data-package ready. Proceed in execution!'];
text1  =sprintf('%u%s\n',lauf,message);
        disp(text1);
% Store data to harddisk
try
     datatodisk;
     disp('Data stored to harddisk. Proceed in execution!')
catch
     text1=['Could not write data to HD.The disk quota is either'...
           ' exceeded or you do not have write permission for the'...
           ' directory!'];
     disp(text1);
     disp('Please restart the program!');
     break;
end
```

6.9.1 The Curlometer Method - *curlB.m*

```
function [jav,divB] = curlB(Ba,Bb,Bc,Bd,ra,rb,rc,rd,y0)
%************************************************************************
% CLUSTER project: CurlB                            Autor: Christian Maszl *
% Formulas: y0*Jav*(rab x rac) = dBab*rac-dBac*rab                         *
%           ¡div B¿*—rab*(rac x rad)— = —Sum(cyclic) dBab*(rac x rad)—     *
%           a,b,c,d -¿ 1,2,3,4                                             *
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at *
%************************************************************************
%
% This function computes the current density jav and ¡divB¿ from
```

```
% magnetometer data.

% Calculate average of B and convert units to SI
    dBab = (Ba-Bb)*1e-9; %[B]=T
    dBac = (Ba-Bc)*1e-9;
    dBad = (Ba-Bd)*1e-9;
    dBcb = (Bc-Bb)*1e-9;
    dBcd = (Bc-Bd)*1e-9;
% Calculate the differences and convert to SI
    rab  = (rb-ra)*1e+3; %[r]=m
    rac  = (rc-ra)*1e+3;
    rad  = (rd-ra)*1e+3;
    rcb  = (rb-rc)*1e+3;
    rcd  = (rd-rc)*1e+3;
    clear Ba Bb Bc Bd ra rb rc rd;
% Calculate the matrix components of f(B)
    fB1 = dot(dBab,rac,2)-dot(dBac,rab,2);
    fB2 = dot(dBac,rad,2)-dot(dBad,rac,2);
    fB3 = dot(dBad,rab,2)-dot(dBab,rad,2);
    fB4 = dot(dBcb,rcd,2)-dot(dBcd,rcb,2);
% Calculate the matrix components of A (area)
    A1  = -cross(rab,rac,2)*y0;
    A2  = -cross(rac,rad,2)*y0;
    A3  = -cross(rad,rab,2)*y0;
    A4  = -cross(rcb,rcd,2)*y0;
% Calculate the matrix components of V (volume)
    V   = abs(dot(rab,A2,2));
% Calculate the matrix components of fBV
    fBV=abs(dot(dBab,A2,2)+dot(dBad,A1,2)+dot(dBac,A3,2));
% Clear dispensable variables
    clear dBab dBac dBad dBcd dBcb;
    clear rab rac rad rcd rcb;
% Calculate ¡div(B)¿
    divB=fBV./V;
% Preallocate the array jav
    jav =zeros(size(A1,1),3);
% Calculate the current density
for i=1:size(A1,1)
% Solve the overdetermined system of equations
jav(i,:) =([ A1(i,:);  A2(i,:);  A3(i,:); A4(i,:)]\ ...
           [fB1(i,:); fB2(i,:); fB3(i,:);fB4(i,:)])';
end
```

6.9.2 Store data to HD - *datatodisk.m*

```
%*************************************************************************
% CLUSTER project: Store jav and divB              Autor: Christian Maszl
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at
%*************************************************************************
%
% Version 1.0 - 20.8.2004
%
% This subroutine store the computed data to the harddisk.
```

```
%
% Filename format for jav:
% [yymmdd]0000[type].jav.[coordsys].[timeformat].[igmmat]
%                                                    [igmasc]
% Example: 0203040000fn.jav.gse.hr.igmmat
%
% Format of the data in the file: [time jav]
%
% Filename format for ¡divB¿:
% [yymmdd]0000[type].divB.[coordsys].[timeformat].[igmmat]
%                                                    [igmasc]
% Example: 0203040000fn.divB.gse.hr.igmmat
%
% Format of the data in the file: [time divB]
%
% Format of D: (only in the error case!)
% [yymmdd]0000[type].DX.[coordsys].[timeformat].[igmmat]
%                                                    [igmasc]
% Example: 0203040000fn.D1.gse.hr.igmmat
%
% Format of the data in the file: [time B1 r1]

if (errorflag_nodata==1)
    if (size(datatype,2)==4)
        ende = ['gse.hr.igmmat'];
    else
        ende = ['gse.hr.igmasc'];
    end
% Try to save D1-D4
    if  (isempty(D1)~=1)
        satz = strrep(myfilelist{lauf,1},'.01','.D1.');
        filename=[outputspath satz ende];
        save(filename,'D1',datatype);
    else
    end
    if(isempty(D2)~=1)
        satz = strrep(myfilelist{lauf,1},'.01','.D2.');
        filename=[outputspath satz ende];
        save(filename,'D2',datatype);
    else
    end
    if(isempty(D3)~=1)
        satz = strrep(myfilelist{lauf,1},'.01','.D3.');
        filename=[outputspath satz ende];
        save(filename,'D3',datatype);
    else
    end
    if(isempty(D4)~=1)
        satz = strrep(myfilelist{lauf,1},'.01','.D4.');
        filename=[outputspath satz ende];
        save(filename,'D4',datatype);
    else
    end
else
```

```
% Take a name of the datapackage and replace the satellite number with
% jav (current density)
    satz = strrep(myfilelist{lauf,1},'.01','.jav.');
% Check for datatype and fix the extension
    if (size(datatype,2)==4)
        filename = [outputspath satz 'gse.hr.igmmat'];
    else
        filename = [outputspath satz 'gse.hr.igmasc'];
    end
% Save data (binary=5x faster) to disk
    save(filename,'T','jav',datatype);
% Take a name of the datapackage and replace the satellite number with
% divB
    filename=strrep(filename,'.jav.','.divB.');
% Save data (binary=5x faster) to disk
    save(filename,'divB',datatype);

% Store also T, B, r for a possible seperate usage.
% Save data (binary=5x faster) to disk
    filename=strrep(filename,'.divB.','.D1.');
    save(filename,'T','B1','r1',datatype);
% Save data (binary=5x faster) to disk
    filename=strrep(filename,'.D1.','.D2.');
    save(filename,'T','B2','r2',datatype);
% Save data (binary=5x faster) to disk
    filename=strrep(filename,'.D2.','.D3.');
    save(filename,'T','B3','r3',datatype);
% Save data (binary=5x faster) to disk
    filename=strrep(filename,'.D3.','.D4.');
    save(filename,'T','B4','r4',datatype);
% Clear dispensable variables
    clear filename satz;
end
```

6.10 The Graphical Output - *plotdiagram.m*

```
%*************************************************************************
% CLUSTER project: Plot divB & curlB to screen      Autor: Christian Maszl
% Unit System = [SI]                          Mail: e0025754@student.tuwien.ac.at
%*************************************************************************
%
% Version 1.0 - 20.8.2004
%
% This subroutine plot jav and B on the screen. The program looks
% up the datafiles in the "outputdata" directrory defined in
% config.m . Please make sure that you selected the right mode!

clear variables;
% Load configfile and check the path's ...
syscheck;
% Plot some infos from configfile.m for the manual mode
dataname=upper(dataname);
datasrc =upper(datasrc);
message =['Enter a date to plot ' dataname ' - ' timeres 'sec '...
          datasrc '-data.'];
          disp(message);
date=input('Please enter the desired day [yymmdd]: ','s');
% Extract extension
l=size(datatype,2);
if (l==4)
    exten=datatype(2:4);
else
    exten=datatype(2:4);
end
javname =[outputspath date '0000' mode '.jav.gse.hr.igm' exten];
D1name  =[outputspath date '0000' mode '.D1.gse.hr.igm' exten];
D2name  =[outputspath date '0000' mode '.D2.gse.hr.igm' exten];
D3name  =[outputspath date '0000' mode '.D3.gse.hr.igm' exten];
D4name  =[outputspath date '0000' mode '.D4.gse.hr.igm' exten];
divBname=[outputspath date '0000' mode '.divB.gse.hr.igm' exten];
reportna=[outputspath date '0000' mode '.report'];
% Load the variabels to the memory
try
load(javname,datatype);
    disp('jav loaded to memory!');
load(D1name,datatype);
load(D2name,datatype);
load(D3name,datatype);
load(D4name,datatype);
    disp('B1-B4 loaded to memory!');
    clear r1 r2 r3 r4;
load(divBname,datatype);
    disp('<divB> loaded to memory!');
catch
    disp(['Error: Filename not found! Please check your input'...
          ' or the available files!']);
    disp('Please restart the program!');
    try
```

```
       load(reportna,'-mat');
       disp(error_report);
    catch
       reportfile='n';
    end
    break;
end
try
   load(reportna,'-mat');
   reportfile='y';
catch
   reportfile='n';
end
clear javname B14name divBname l exten year month day;
try
year(1,1) =str2num(date(1,1:2));
month(1,1)=str2num(date(1,3:4));
day(1,1)  =str2num(date(1,5:6));
datum=datestr(datenum([2000+year month day]));
timelabel=['{t / [h]}'];
text1=[datum ' , Coord. system: {\itGSE} , Unit System: {\itSI} , '...
            'Time System: {\itUT}'];
% Find plot range for jav and check for data
if (isempty(jav)~=1)
    if (plot_mode_normal=='y')
    range =(sum(abs(jav) ,1)/size(jav,1))*10;
% Define styles, labels for axis, ...
    jav_axisr=[0 24.0 -range(1,2) range(1,2)];
figure(1);
textfig1=['Jav & divB - ' datum];
set(1,'Name',textfig1);
% Plot the x component of the current density versus the time.
    subplot(4,1,1)
    plot(T,jav(:,1))
        axis(jav_axisr)
        ylabel('{J}_{x}{ / [A/m]}^{2}{]}',...
        'FontSize',10)
        xlabel(timelabel    ,'FontSize',10)
        title(text1,'FontSize',12)
        grid on
% Plot the y component of the current density versus the time.
    subplot(4,1,2)
    plot(T,jav(:,2))
        axis(jav_axisr)
        ylabel('{J}_{y}{ / [A/m]}^{2}{]}',...
        'FontSize',10)
        xlabel(timelabel    ,'FontSize',10)
        grid on
% Plot the z component of the current density versus the time.
        subplot(4,1,3)
        plot(T,jav(:,3))
        axis(jav_axisr)
        ylabel('{J}_{z}{ / [A/m]}^{2}{]}',...
        'FontSize',10)
        xlabel(timelabel    ,'FontSize',10)
        grid on
% Plot ¡divB¿/abs(curlB) versus the time.
```

```
    divBocurlB=(divB./sqrt(dot(jav,jav,2)))*100;
    range     =(sum(abs(divBocurlB) ,1)/size(T,1))*10;
    subplot(4,1,4)
    plot(T,divBocurlB)
        axis([0.0 24.0 0 range])
        ylabel('{\nabla\cdotB/|\nablaxB| / [%]}' ,'FontSize',10)
        xlabel(timelabel            ,'FontSize',10)
        formula1=['{\it\mu}_{\it0}{\itJ}_{\itav}{ = }'...
                  '{\it\nablaxB}'];
        formula2=['{\it\int_V}{\itdV \nabla\cdotB} = '...
                  '{\it\int_A}{\itdA B}'];
        text=['Used formula:   ' formula1 '   and   ' formula2...
              '     Data source: {\it' upper(datasrc) '} - '  ...
              timeres   'sec data'];
        title(text,'Fontsize',12)
        grid on
    else
    end
    if (plot_mode_parnor=='y')
% Calculate the normal and parallel component of Jav
    Bm      =(B1+B2+B3+B4)/4;
    norm_jav=sqrt(dot(jav,jav,2));
    norm_B  =sqrt(dot(Bm ,Bm ,2));
    cosphi  =(dot(Bm,jav,2)./(norm_B.*norm_jav));
    jav_p   =dot(Bm,jav,2)./norm_B;
    jav_n   =norm_jav.* sqrt(1-dot(cosphi,cosphi,2));
    % Find plot ranges
    range =(sum(abs(jav_p) ,1)/size(jav_p,1))*10;
    % Define styles, labels for axis, ...
    jav_axisr=[0 24.0 -range(1,1) range(1,1)];
    timelabel=['{t / [h]}'];
figure(3);
textfig1=['Jav(para), Jav(perp) & divB - ' datum];
set(3,'Name',textfig1);
% Plot the Jav_para versus the time.
    subplot(3,1,1)
    plot(T,jav_p)
        axis(jav_axisr)
        ylabel('{J}_{||} { / [A/m]^{2}{J}}','FontSize',10)
        xlabel(timelabel    ,'FontSize',10)
        text1=[datum ' , Coord. system: {\itGSE} , Unit System:'...
               ' {\itSI} , Time System: {\itUT}'];
        title(text1,'FontSize',12)
        grid on
% Plot Jav_perp versus the time.
    range     =(sum(abs(jav_n) ,1)/size(T,1))*10;
    subplot(3,1,2)
    plot(T,jav_n)
        axis([0.0 24.0 0 range])
        ylabel('{J}_\perp{ / [A/m]^{2}{J}}','FontSize',10)
        xlabel(timelabel            ,'FontSize',10)
        grid on
% Plot ¡divB¿/abs(curlB) versus the time.
    divBocurlB=(divB./sqrt(dot(jav,jav,2)))*100;
```

```
    range     =(sum(abs(divBocurlB) ,1)/size(T,1))*10;
    subplot(3,1,3)
    plot(T,divBocurlB)
        axis([0.0 24.0 0 range])
        ylabel('{\nabla\cdotB/|\nablaxB| / [%]}' ,'FontSize',10)
        xlabel(timelabel            ,'FontSize',10)
        formula1=['{\it\mu}_{\it0}{\itJ}_{\itav}{ = }'...
                  '{\it\nablaxB}'];
        formula2=['{\it\int_V}{\itdV \nabla\cdotB} = '...
                  '{\it\int_A}{\itdA B}'];
        text=['Used formula:   ' formula1 '   and   ' formula2...
              '     Data source: {\it' upper(datasrc) '} - '  ...
              timeres   'sec data'];
        title(text,'Fontsize',12)
        grid on

    else
    end
else
end
% Open a new plot window
figure(2);
textfig2=['B-field - ' datum];
set(2,'Name',textfig2)
% Plot the x component of B versus the time
subplot(4,1,1)
plot(T,B1(:,1),T,B2(:,1),T,B3(:,1),T,B4(:,1))
    axis([0.0 24.0 -1000 1000])
    axis 'auto y'
    ylabel('{B}_{x}{ / [nT]}','FontSize',10)
    xlabel(timelabel  ,'FontSize',10)
    title(text1,'FontSize',12)
    grid on
% Plot the y component of B versus the time.
subplot(4,1,2)
plot(T,B1(:,2),T,B2(:,2),T,B3(:,2),T,B4(:,2))
    axis([0.0 24.0 -1000 1000])
    axis 'auto y'
    ylabel('{B}_{y}{ / [nT]}','FontSize',10)
    xlabel(timelabel  ,'FontSize',10)
    grid on
% Plot the z component of B versus the time.
subplot(4,1,3)
plot(T,B1(:,3),T,B2(:,3),T,B3(:,3),T,B4(:,3))
    axis([0.0 24.0 -1000 1000])
    axis 'auto y'
    ylabel('{B}_{z}{ / [nT]}','FontSize',10)
    xlabel(timelabel  ,'FontSize',10)
    grid on
% Calculate —B—
B1m=sqrt(dot(B1,B1,2));
B2m=sqrt(dot(B2,B2,2));
B3m=sqrt(dot(B3,B3,2));
B4m=sqrt(dot(B4,B4,2));
```

```
% Plot —B— versus the time.
subplot(4,1,4)
plot(T,B1m,T,B2m,T,B3m,T,B4m)
    axis([0.0 24.0 -1000 1000])
    axis 'auto y'
    ylabel('{|B| / [nT]}' ,'FontSize',10)
    xlabel(timelabel    ,'FontSize',10)
    text=['Used formula:   ' '{\itB = (B}_{\it1}{+\itB}_{\it2}+'...
          '{\itB}_{\it3}{+\itB}_{\it4}{\it)/4}' '    Data source:  '...
          '{\it' upper(datasrc) '} - ' timeres 'sec data'];
    title(text,'FontSize',12)
    grid on
    if (legend_on=='y')
        legend('SC1','SC2','SC3','SC4',4);
    else
    end
catch
   disp('Error: Files contain no data!')
end
% Check if there is a report file
if (reportfile=='n')
else
% Display error messages on the screen
disp(' ');
disp('REPORT FROM THE GETDATA ROUTINE! READ THIS MESSAGE CAREFULLY!');
disp(error_report);
end
clear advanced datapath datasrc datatype date datum day extension;
clear formula1 formula2 instrument jav_axisr logfilepath mode;
clear month nolog opmode outputspath range reportfile reportna;
clear tempdirpath text* timeacc timelabel timeres y0 year compmode;
clear B*m D*name varname option op_mode message isdat_db dataname;
clear plot_* legend_* cos* norm_*
% Unlock plotdiagram
munlock plotdiagram;
```

Chapter 7

Curlometer on Model Current Sheets

This chapter deals with the limitations and imperfections of the *curlometer method*. In order to investigate effects that may occur because of different enviromental conditions, two model current sheets have been chosen for what the magnetic field strength $\vec{B}$, predetermined by a specific current density J_{av}, is known.

Therefore a program called *Cluster Flight* was written.

7.1 Helper Application - *Cluster Flight*

***Cluster Flight* is a very simple program that allows simulation of flights of the *Cluster II* tetrahedron outside, inside and through different current sheets.**
The program leads you trough a few simple dialogues where you have to insert the *radius* or *diameter* of *the current sheet*, *the satellite separation* and the *current density*. Furthermore one has to enter the time-steps (*1 time-step*=$1m$ and the direction of movement.
To display the results the program *plotdiagram.m* is used. Note, do not mistake this program with the plotfunction in sec(4.2.10) at p.(30).

7.1.1 The Source Code

The Main Program - *cluster_flight.m*

is the mainprogram and call the following subroutines.

```
%*************************************************************************
% CLUSTER project: Cluster Flight                    Autor: Christian Maszl *
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at *
%*************************************************************************
%
% This program simulates flights of the four cluster spacecrafts
% trough various current sheets.

% Define constants
y0 = 1.256637061e-6; %[y0]=Vs/Am
disp ('Please choose the type of the current sheet: ');
message=['[1] Cylindrical current sheet - [2] Flat current sheet'];
disp (message);
sheetform=input('Please type [1] or [2]: ');
disp(' ');
if (sheetform==1)
   r =input('Radius of the current tube [m]: r= ');
   s =input('Satellite separation       [m]: s= ');
   jy=input('Current density          [A/m]: j= ');
   disp(' ');
   s=s/2;
% Compute the x,y and z values depending on the flight path.
        [x,y,z,T,stepanz]=bewegungsrichtung(r,s);
% Compute the position vectors of the space crafts.
        [r1,r2,r3,r4]=sat_pos(x,y,z,s);
% Report when the spacecrafts enter the current sheet
%                        enter_sheet(r1,r2,r3,r4,r);
% Compute the B-field vectors at the tetrahedron vertices.
        [B1]=cylsheet_B(r1,r,y0,jy,stepanz);
        [B2]=cylsheet_B(r2,r,y0,jy,stepanz);
        [B3]=cylsheet_B(r3,r,y0,jy,stepanz);
        [B4]=cylsheet_B(r4,r,y0,jy,stepanz);
% Use the curlometer method to compute jy again.
        r1=r1*1e-3;r2=r2*1e-3;r3=r3*1e-3;r4=r4*1e-3;
        [jav,divB]=curlB(B1,B2,B3,B4,r1,r2,r3,r4,y0);
% Plot the results on the screen
        jav=[T jav];
        plotdiagram(r1,r2,r3,r4,B1,B2,B3,B4,jav,divB,stepanz);
elseif (sheetform==2)
   d =input('Thickness of the flat current sheet [m]: d= ');
   s =input('Satellite separation                [m]: s= ');
   jy=input('Current density                   [A/m]: j= ');
   s=s/2; d=d/2;
% Compute the x,y and z values depending on the flight path.
        [x,y,z,T,stepanz]=bewegungsrichtung(d,s);
% Compute the position vectors of the space crafts.
        [r1,r2,r3,r4]=sat_pos(x,y,z,s);
% Compute the B-field vectors at the tetrahedron vertices.
        [B1]=flatsheet_B(r1,d,y0,jy,stepanz);
```

```
        [B2]=flatsheet_B(r2,d,y0,jy,stepanz);
        [B3]=flatsheet_B(r3,d,y0,jy,stepanz);
        [B4]=flatsheet_B(r4,d,y0,jy,stepanz);
% Use the curlometer method to compute jy again.
        r1=r1*1e-3;r2=r2*1e-3;r3=r3*1e-3;r4=r4*1e-3;
        [jav,divB]=curlB(B1,B2,B3,B4,r1,r2,r3,r4,y0);
% Plot the results on the screen
        r1=r1;r2=r2;r3=r3;r4=r4;
        jav=[T jav];
        plotdiagram(r1,r2,r3,r4,B1,B2,B3,B4,jav,divB,stepanz);
        r1=r1*1e3;r2=r2*1e3;r3=r3*1e3;r4=r4*1e3;
else
end
```

Check for Operation Mode - ***bewegungsrichtung.m***

Check for the operation and ask for the number of time-steps.

```
function [x,y,z,T,stepanz]=bewegungsrichtung(u,s);
%*************************************************************************
% CLUSTER project: Cluster Flight                    Autor: Christian Maszl *
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at *
%*************************************************************************
%
% This subroutine calculate the the path of the midpoint between
% satellite S1 and S3.

disp('Please choose the motion type: ');
disp('[a] In the current sheet (along y)');
disp('[b] Outside the current sheet (along y)');
disp('[c] Penetrate the current sheet (along x)');
howtomove=input('Please choose [a],[b] or [c]: ','s');
stepanz=input('How many time steps do you wish?: ');
% Use T as time base
T=(1:0.1:stepanz+1)';
%[a] In the current sheet (along y)
if (howtomove=='a')
   x(1:(10*stepanz)+1,1)=0;
   z(1:(10*stepanz)+1,1)=0;
   if (s>=u)
     disp('Warning: At least one statellite outside the current sheet!');
   end
   y=(-stepanz/2:0.1:stepanz/2)';
%[b] Outside the current sheet (along y)
elseif (howtomove=='b')
   z(1:(10*stepanz)+1,1)=0;
   y=(-stepanz/2:0.1:stepanz/2)';
   x=input('Distance to the x-axis [m]: x=');
   if (x<=u+s)
     disp('Warning: At least one satellite inside the current sheet');
   end
   x(1:(10*stepanz)+1,1)=x;
%[c] Penetrate the current sheet (along x)
elseif (howtomove=='c')
   y(1:(10*stepanz)+1,1)=0;
   z(1:(10*stepanz)+1,1)=0;
   x=(-stepanz/2:0.1:stepanz/2)';
else
end
```

Find the position - *sat_pos.m*

find the position of the tetrahedron vertices according to the flight direction and the time-steps.

```
function [r1,r2,r3,r4]=sat_pos(x,y,z,s)
%*************************************************************************
% CLUSTER project: Cluster Flight                    Autor: Christian Maszl *
% Unit System = [SI]                        Mail: e0025754@student.tuwien.ac.at *
%*************************************************************************
%
% This subroutine calculate the position vectors for all time
% steps.

% Convert to km.
r1=[x(:)-(sqrt(6)/3)*s                  y(:)  -s+z(:)];
r2=[x(:)-(sqrt(6)/3)*s        sqrt(3)*s+y(:)     z(:)];
r3=[x(:)-(sqrt(6)/3)*s                  y(:)   s+z(:)];
r4=[x(:)+(sqrt(6)/3)*s    (sqrt(3)/3)*s+y(:)     z(:)];
```

Find $\vec{B}$ for a Cylindrical Sheet - *cylsheet_B.m*

calculates the magnetic field strength $\vec{B}$ at the vertices of the tetrahedron for each time-step for a cylindrical current tube.

```
function [B]=cylsheet_B(r,rc,y0,j,stepanz)
%*************************************************************************
% CLUSTER project: Cluster Flight                    Autor: Christian Maszl *
% Unit System = [SI]                    Mail: e0025754@student.tuwien.ac.at *
%*************************************************************************
%
% This subroutine calculate the B field vectors for every
% position. The middle axis of the current tube with infinite
% length ist the z-axis. The current points in the positive z-direction.

B=zeros(10*stepanz,3);

for i=1:(10*stepanz)+1
% Compute the distance to the middle-axes.
abstand= sqrt(r(i,1)^2+r(i,2)^2);
% Compute the angle with respect to the x-axis.
phi    = acos(r(i,1)/abstand);
    if (abstand<=rc)
       % Innen
       Bl    =(y0/2)*j*abstand*1e9;
       B(i,1)=Bl*cos(phi+pi/2);
       B(i,2)=Bl*sin(phi+pi/2);
    elseif (abstand>rc)
       % Aussen
       Bl    =(y0/2)*j*rc^2/abstand*1e9;
       B(i,1)=Bl*cos(phi+pi/2);
       B(i,2)=Bl*sin(phi+pi/2);
    end
end
```

Find $\vec{B}$ for a Flat Sheet - *flatsheet_B.m*

calculates the magnetic field strength $\vec{B}$ at the vertices of the tetrahedron for each time-step for a flat current sheet.

```
function [B]=flatsheet_B(r,d,y0,j,stepanz)
%************************************************************************
% CLUSTER project: Cluster Flight                    Autor: Christian Maszl *
% Unit System = [SI]                        Mail: e0025754@student.tuwien.ac.at *
%************************************************************************
%
% This subroutine calculate the B field vectors for every
% position. The infinite current sheet with thickness of 2d
% lay in the y/z-plane. The current vector point in the positive
% y-direction.

B=zeros(10*stepanz,3);
% Convert B to nT
for i=1:(10*stepanz)+1
    if (abs(r(i,1))<=d)
       B(i,3)=-y0*j*r(i,1)*1e9;
    elseif (r(i,1)>d)
       B(i,3)=-y0*j*d*1e9;
    elseif (r(i,1)<d)
       B(i,3)=+y0*j*d*1e9;
    end
end
```

Use Curlometer to get J_{av} - *curlB.m*

applies the *Curlometer Method* at the magnetic field data.

```
function [jav,divB] = curlB(Ba,Bb,Bc,Bd,ra,rb,rc,rd,y0)
%*************************************************************************
% CLUSTER project: CurlB                          Autor: Christian Maszl *
% Formulas: y0*Jav*(rab x rac) = dBab*rac-dBac*rab                       *
%           ¡div B¿*—rab*(rac x rad)— = —Sum(cyclic) dBab*(rac x rad)—    *
%           a,b,c,d -¿ 1,2,3,4                                           *
% Unit System = [SI]                  Mail: e0025754@student.tuwien.ac.at *
%*************************************************************************
%
% This function computes the current density jav and ¡divB¿ from
% magnetometer data.

% Calculate average of B and convert units to SI
    dBab = (Ba-Bb)*1e-9; %[B]=T
    dBac = (Ba-Bc)*1e-9;
    dBad = (Ba-Bd)*1e-9;
    dBcb = (Bc-Bb)*1e-9;
    dBcd = (Bc-Bd)*1e-9;
% Calculate the differences and convert to SI
    rab  = (rb-ra)*1e+3; %[r]=m
    rac  = (rc-ra)*1e+3;
    rad  = (rd-ra)*1e+3;
    rcb  = (rb-rc)*1e+3;
    rcd  = (rd-rc)*1e+3;
    clear Ba Bb Bc Bd ra rb rc rd;
% Calculate the matrix components of f(B)
    fB1 = dot(dBab,rac,2)-dot(dBac,rab,2);
    fB2 = dot(dBac,rad,2)-dot(dBad,rac,2);
    fB3 = dot(dBad,rab,2)-dot(dBab,rad,2);
    fB4 = dot(dBcb,rcd,2)-dot(dBcd,rcb,2);
% Calculate the matrix components of A (area)
    A1  = -cross(rab,rac,2)*y0;
    A2  = -cross(rac,rad,2)*y0;
    A3  = -cross(rad,rab,2)*y0;
    A4  = -cross(rcb,rcd,2)*y0;
% Calculate the matrix components of V (volume)
    V   = abs(dot(rab,A2,2));
% Calculate the matrix components of fBV
    fBV=abs(dot(dBab,A2,2)+dot(dBad,A1,2)+dot(dBac,A3,2));
% Clear dispensable variables
    clear dBab dBac dBad dBcd dBcb;
    clear rab rac rad rcd rcb;
% Calculate ¡div(B)¿
    divB=fBV./V;
% Preallocate the array jav
    jav =zeros(size(A1,1),3);
% Calculate the current density
for i=1:size(A1,1)
% Solve the overdetermined system of equations
jav(i,:) =([ A1(i,:);  A2(i,:);  A3(i,:); A4(i,:)]\ ...
           [fB1(i,:); fB2(i,:); fB3(i,:);fB4(i,:)])';
```

```
end
```

Display results to screen -*plotdiagram.m*

displays the results on the screen.

```
function plotdiagram(r1,r2,r3,r4,BA,BB,BC,BD,jav,divB,stepanz)
%************************************************************************
% CLUSTER project: Plot divB & curlB to screen      Autor: Christian Maszl *
% Unit System = [SI]                          Mail: e0025754@student.tuwien.ac.at *
%************************************************************************
l=size(BA,1);
BA  = BA (1:l-1,:);
BB  = BB (1:l-1,:);
BC  = BC (1:l-1,:);
BD  = BD (1:l-1,:);
r1  = r1 (1:l-1,:)*1000;
r2  = r2 (1:l-1,:)*1000;
r3  = r3 (1:l-1,:)*1000;
r4  = r4 (1:l-1,:)*1000;
jav = jav(1:l-1,:);
divB=divB(1:l-1,:);
B1=BA;
B2=BB;
B3=BC;
B4=BD;
%[] Load data from a mat-file
%[] Find plot ranges
timelabel=['{t / [s]}'];
T=jav(:,1);
jav=[jav(:,2) jav(:,3) jav(:,4)];
% Plot the x component of the current density versus the time.
figure(1)
subplot(4,1,1)
plot(T,jav(:,1))
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{J}_{x}{ / [A/m]^{2}{J}}','FontSize',10)
    xlabel(timelabel    ,'FontSize',10)
    grid on
% Plot the y component of the current density versus the time.
subplot(4,1,2)
plot(T,jav(:,2))
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{J}_{y}{ / [A/m]^{2}{J}}','FontSize',10)
    xlabel(timelabel    ,'FontSize',10)
    grid on
% Plot the z component of the current density versus the time.
subplot(4,1,3)
plot(T,jav(:,3))
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{J}_{z}{ / [A/m]^{2}{J}}','FontSize',10)
    xlabel(timelabel    ,'FontSize',10)
    grid on
% Plot ¡divB¿/abs(curlB) versus the time.
```

```
divBocurlB=(divB./sqrt(dot(jav,jav,2)))*100;
subplot(4,1,4)
plot(T,divBocurlB)
     axis([0.0 stepanz -1000 1000])
     axis 'auto y'
     ylabel('{\nabla\cdotB/|\nablaxB| / [%]}' ,'FontSize',10)
     xlabel(timelabel            ,'FontSize',10)
     grid on
% Plot the x component of B versus the time
figure(2)
subplot(4,1,1)
plot(T,BA(:,1),T,BB(:,1),T,BC(:,1),T,BD(:,1))
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{B}_{x}{ / [nT]}','FontSize',10)
    xlabel(timelabel  ,'FontSize',10)
    grid on
% Plot the y component of B versus the time.
subplot(4,1,2)
plot(T,BA(:,2),T,BB(:,2),T,BC(:,2),T,BD(:,2))
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{B}_{y}{ / [nT]}','FontSize',10)
    xlabel(timelabel  ,'FontSize',10)
    grid on
% Plot the z component of B versus the time.
subplot(4,1,3)
plot(T,BA(:,3),T,BB(:,3),T,BC(:,3),T,BD(:,3))
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{B}_{z}{ / [nT]}','FontSize',10)
    xlabel(timelabel  ,'FontSize',10)
    grid on
% Calculate —B—
B1n=sqrt(dot(BA,BA,2));
B2n=sqrt(dot(BB,BB,2));
B3n=sqrt(dot(BC,BC,2));
B4n=sqrt(dot(BD,BD,2));
% Plot —B— versus the time.
subplot(4,1,4)
plot(T,B1n,T,B2n,T,B3n,T,B4n)
    axis([0.0 stepanz -1000 1000])
    axis 'auto y'
    ylabel('{|B| / [nT]}' ,'FontSize',10)
    xlabel(timelabel    ,'FontSize',10)
    legend('{SC1}','{SC2}','{SC3}','{SC4}',4);
    grid on
Bm      =(B1+B2+B3+B4)/4;
norm_jav=sqrt(dot(jav,jav,2));
norm_B  =sqrt(dot(Bm ,Bm ,2));
cosphi  =(dot(Bm,jav,2)./(norm_B.*norm_jav));
jav_p   =dot(Bm,jav,2)./norm_B;
jav_n   =norm_jav.* sqrt(1-dot(cosphi,cosphi,2));
figure(3);
```

```
% Plot the Jav_para versus the time.
    subplot(3,1,1)
    plot(T,jav_p)
        axis([0.0 stepanz -1000 1000])
        axis 'auto y'
        ylabel('{J}_{||} { / [A/m}^{2}{]}','FontSize',10)
        xlabel(timelabel    ,'FontSize',10)
        grid on
% Plot Jav_perp versus the time.
     subplot(3,1,2)
     plot(T,jav_n)
         axis([0.0 stepanz -1000 1000])
         axis 'auto y'
         ylabel('{J}_\perp{ / [A/m}^{2}{]}','FontSize',10)
         xlabel(timelabel            ,'FontSize',10)
         grid on
% Plot ¡divB¿/abs(curlB) versus the time.
    divBocurlB=(divB./sqrt(dot(jav,jav,2)))*100;
    subplot(3,1,3)
    plot(T,divBocurlB)
        axis([0.0 stepanz -1000 1000])
        axis 'auto y'
        ylabel('{\nabla\cdotB/|\nablaxB| / [%]}' ,'FontSize',10)
        xlabel(timelabel            ,'FontSize',10)
        grid on
figure(4)
    subplot(4,1,1)
    plot(T,r1)
        axis([0.0 stepanz -1000 1000])
        axis 'auto y'
        ylabel('{SC1 - r_1 / [m]}','FontSize',10)
        xlabel(timelabel          ,'FontSize',10)
        grid on
        legend('{x_1}','{y_1}','{z_1}',3);
    subplot(4,1,2)
    plot(T,r2)
        axis([0.0 stepanz -1000 1000])
        axis 'auto y'
        ylabel('{SC2 - r_2 / [m]}','FontSize',10)
        xlabel(timelabel          ,'FontSize',10)
        grid on
        legend('{x_2}','{y_2}','{z_2}',3);
    subplot(4,1,3)
    plot(T,r3)
        axis([0.0 stepanz -1000 1000])
        axis 'auto y'
        ylabel('{SC3 - r_3 / [m]}','FontSize',10)
        xlabel(timelabel          ,'FontSize',10)
        grid on
        legend('{x_3}','{y_3}','{z_3}',3);
    subplot(4,1,4)
    plot(T,r4)
         axis([0.0 stepanz -1000 1000])
         axis 'auto y'
```

```
ylabel('{SC4 - r_4 / [m]}','FontSize',10)
    xlabel(timelabel          ,'FontSize',10)
    grid on
    legend('{x_4}','{y_4}','{z_4}',3);
```

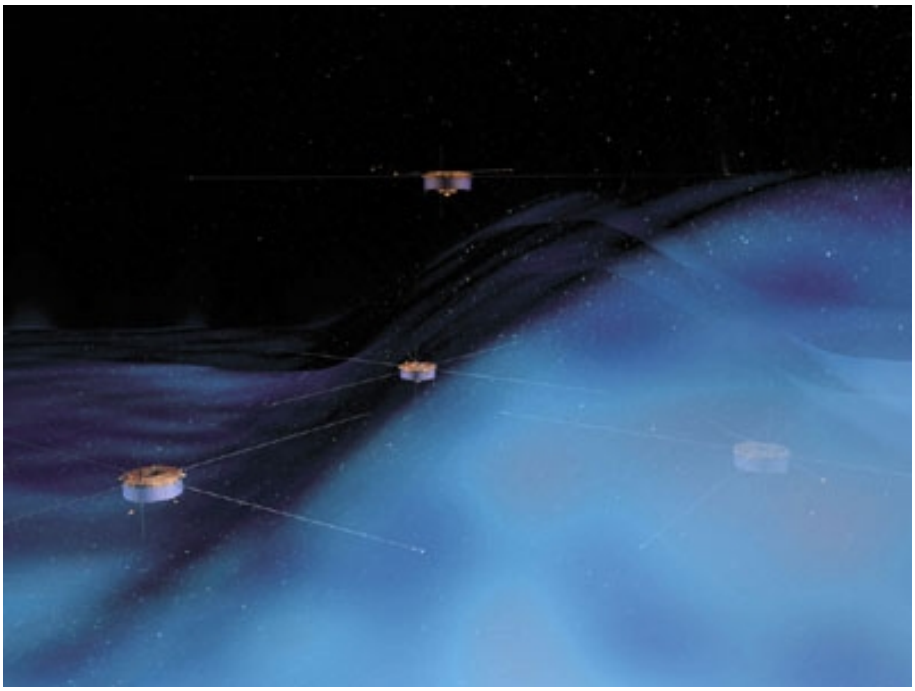

Figure 7.1: Artist's view of the magnetosphere

7.2 Flat current sheet with homogenous $\vec{j}$

7.2.1 Derivation of the magnetic field

The infinite expanded current sheet is in the yz-plane with a finite thickness $\pm d$. The current density $\vec{j}$ point is in the positive y-direction.

First the magnetic field $\vec{B}$ in the current sheet $(b < d)$ is calculated.

$$\int_C \vec{ds}\ \vec{B} = \mu_0 \int_A \vec{da}\ \vec{j}. \tag{7.1}$$

Due to the high symmetry where is only $\vec{B}_z$ and therefore one can write

$$\int_a^{-a} dz\ B_z - \int_{-a}^{a} dz B_z = \mu_0\, 4ab\, j_y. \tag{7.2}$$

Integrating and rewritting b as x leads to eq(7.3) for the inner field

$$\boxed{B_z = -\mu_0\, j_y\, x \qquad x < d.} \tag{7.3}$$

Using the same steps for the outer field $(x > d)$ one writes

$$\boxed{B_z = -\mu_0\, j_y\, d \qquad x > d.} \tag{7.4}$$

And similarly for the outer field in the negative direction $(x < -d)$

$$\boxed{B_z = +\mu_0\, j_y\, d \qquad x < -d.} \tag{7.5}$$

The field in the outerspace is constant as observable from eq(7.4) and eq(7.5) and changes linearly inside the current sheet eq(7.3).

7.2.2 Plots and results

In this section are the plots for different configurations of the current sheet and the satellites.

Note that the values of the different parameters are far from values in nature!

This is only a phenomenological approach to show the limitations in two easy cases. The errors due to deviations from the tetrahedron shape and currents changing with time are not investigated!

Geometrical coherences of the Current Sheet

The infinite current sheet is parallel to the yz-plane. The thickness of the tube is given by d **and the current density** $\vec{j}$ **points in the positve y-direction.**
The satellite separation is given by s**. The tetrahedron is moving along the x-axis from negative to positive values. The plane with the satellites SC1, SC2 and SC3 on its vertices is parallel to the yz-plane. SC4 penetrates the current tube first, followed by SC1-SC3. For SC1 and SC3** $y = 0$**. Therefore these satellites measures the highest values of** $\vec{B}$**.**
For each plot, 3000 **time steps have been used (**$1s = 1m$**, 1 time step=10 "measurements"). The means that the x-axis values vary from** -1500 **to** $+1500$**. Check *sat_pos.m* for more details.**

Large current sheets - $s \ll r$

$d = 1000m$**,** $s = 1m$**,** $j = 10^{-6} \frac{A}{m^2}$

The thickness of the sheet and the amplitude of the current density are well preserved. $\vec{\nabla} \cdot \vec{B}$ **is alway** 0**. That is because** $\vec{B}$ **is constant outside the current sheet.**

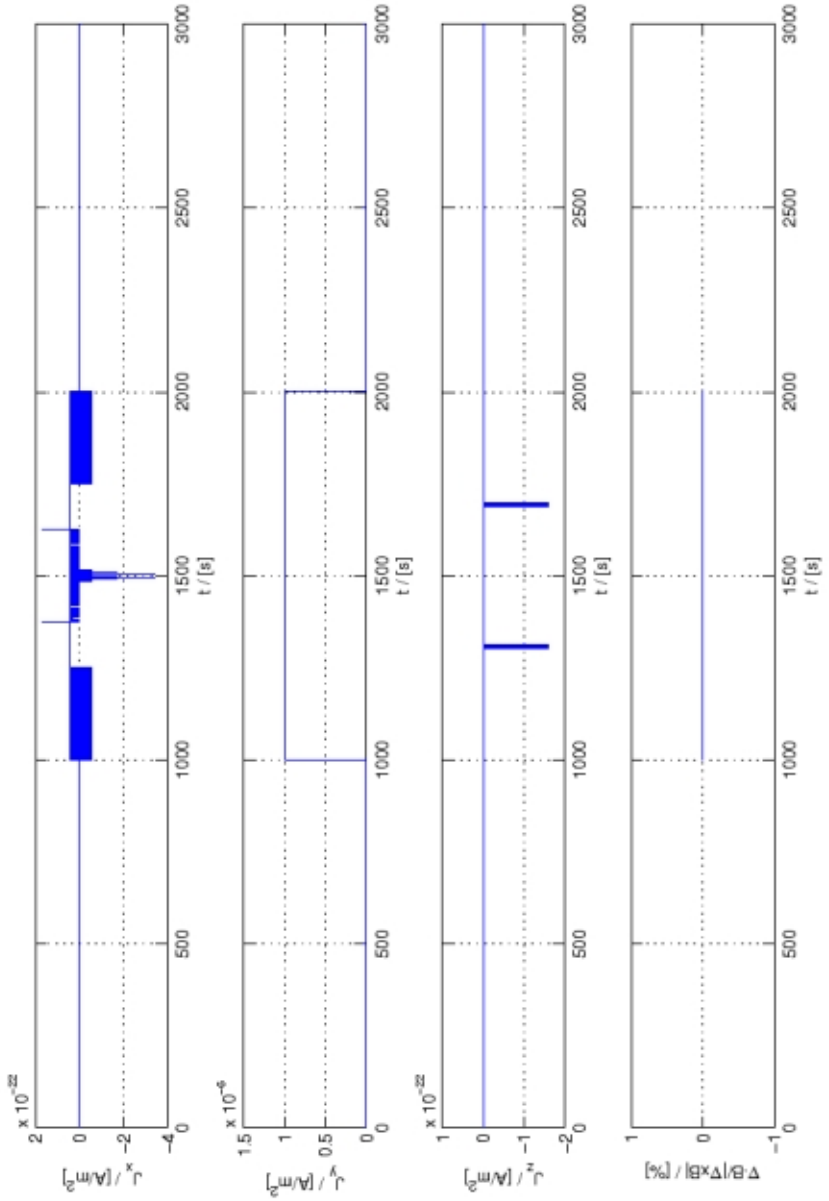

Figure 7.2: Large current sheets: $\vec{j}$ - $d = 1000m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

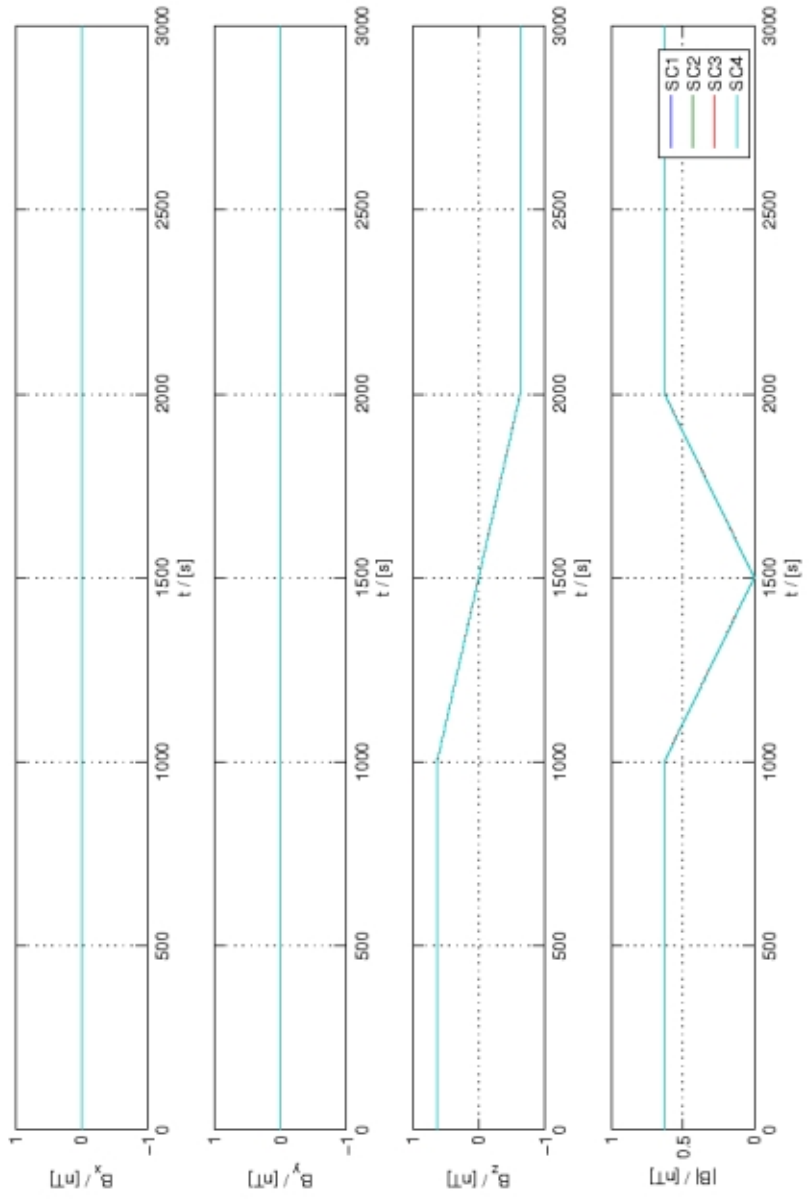

Figure 7.3: Large current sheets: $\vec{B}$ - $d = 1000m$, $s = 1m$, $j = 10^{-6} \frac{A}{m^2}$

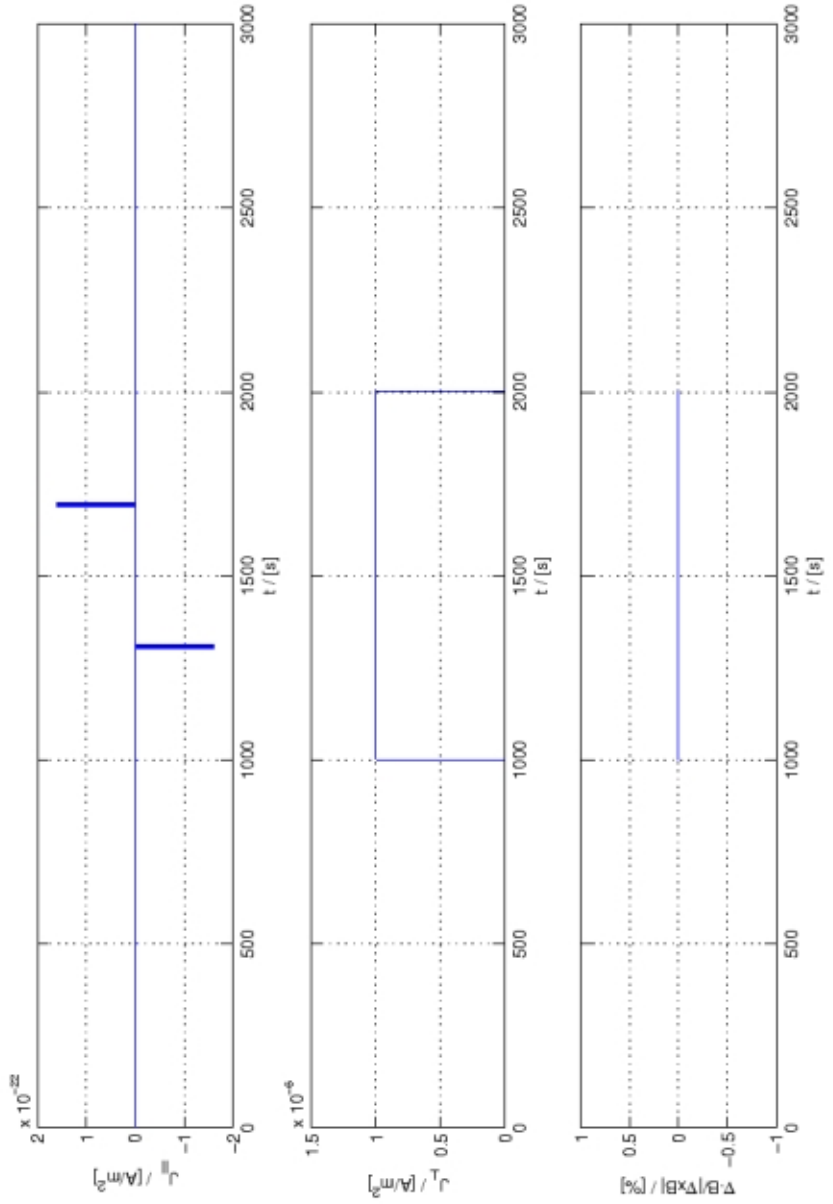

Figure 7.4: Large current sheets: $\vec{j}_{\perp/\parallel}$ - $d = 1000m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

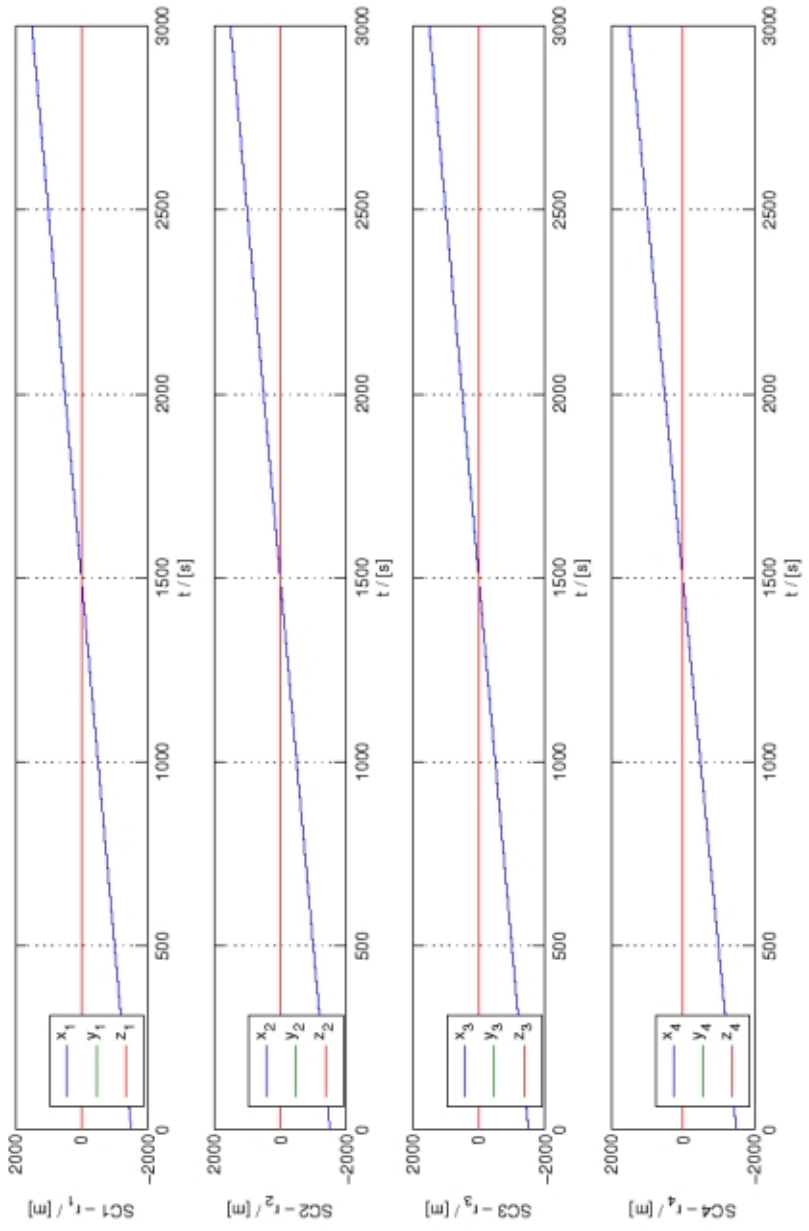

Figure 7.5: Large current sheets: $\vec{r}$ - $d = 1000m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

Small current sheets - $s \gg r$

$d = 1m$**,** $s = 1000m$**,** $j = 10^{-6}\frac{A}{m^2}$

The thickness of the sheet appears bigger then it should be. The amplitude of the current density becomes damped. $\vec{\nabla} \cdot \vec{B}$ **is alway** 0**. That is because** $\vec{B}$ **is constant outside the current sheet.**

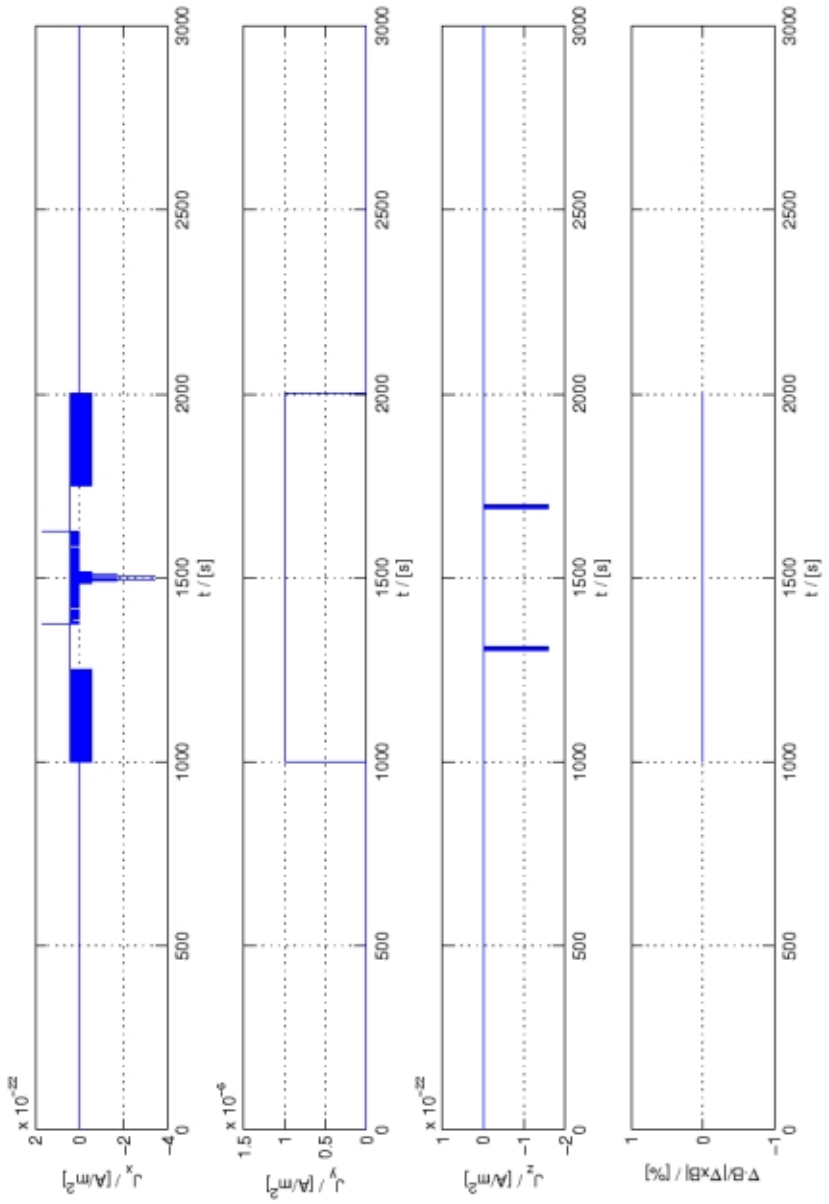

Figure 7.6: Small current sheets: $\vec{j}$ - $d = 1m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

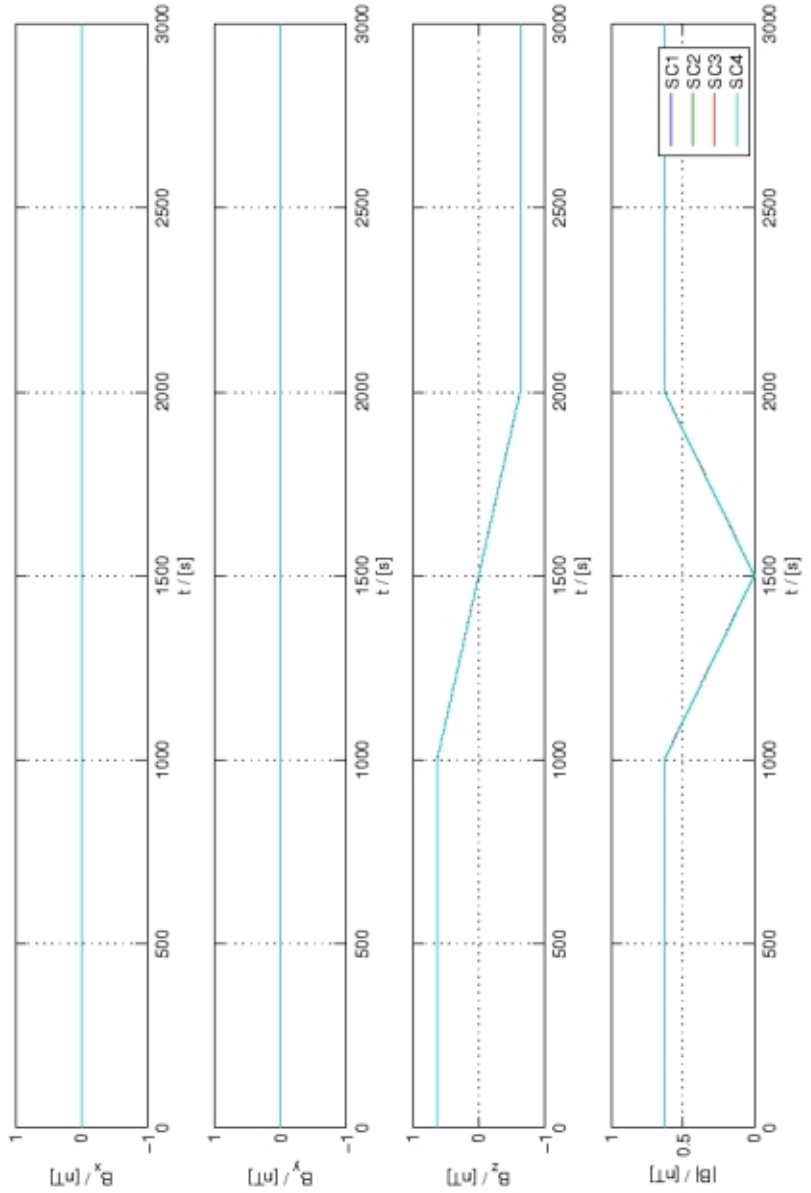

Figure 7.7: Small current sheets: $\vec{B}$ - $d = 1m$, $s = 1000m$, $j = 10^{-6} \frac{A}{m^2}$

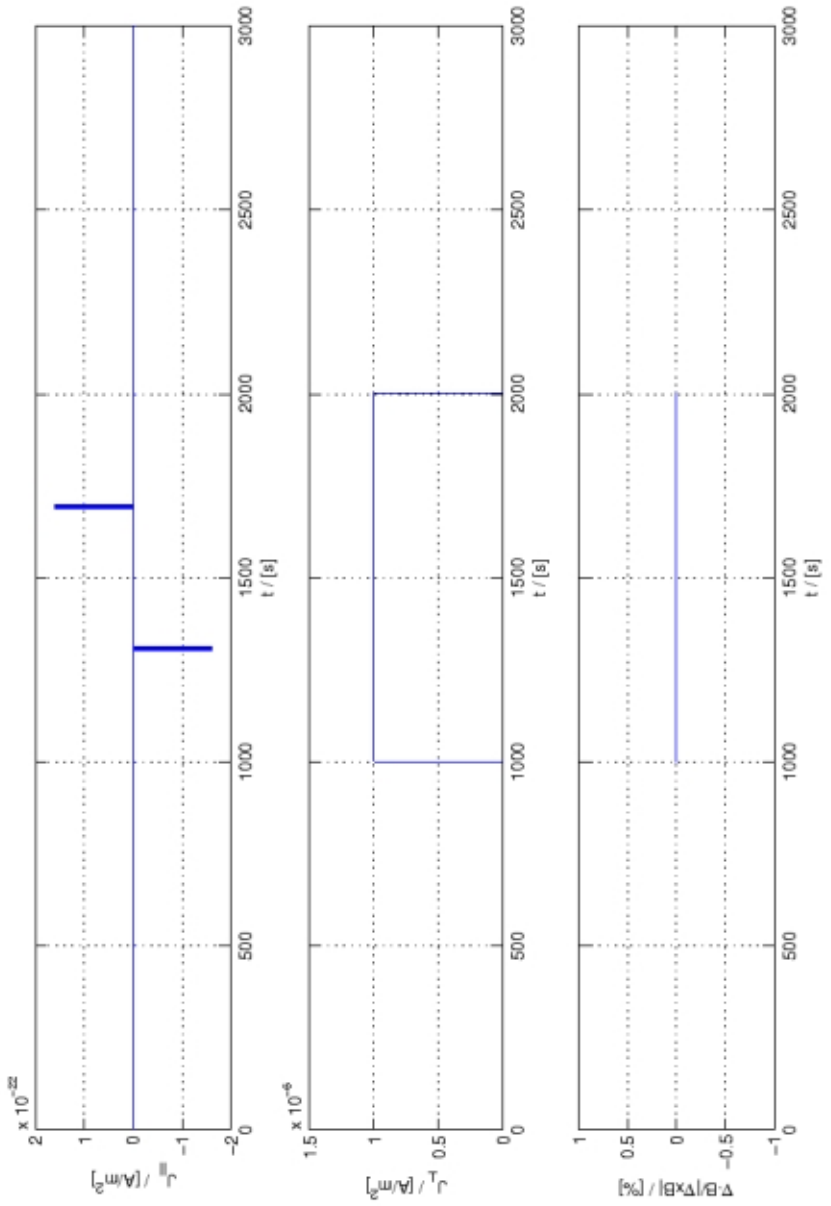

Figure 7.8: Small current sheets: $\vec{j}_{\perp/\parallel}$ - $d = 1m$, $s = 1000m$, $j = 10^{-6} \frac{A}{m^2}$

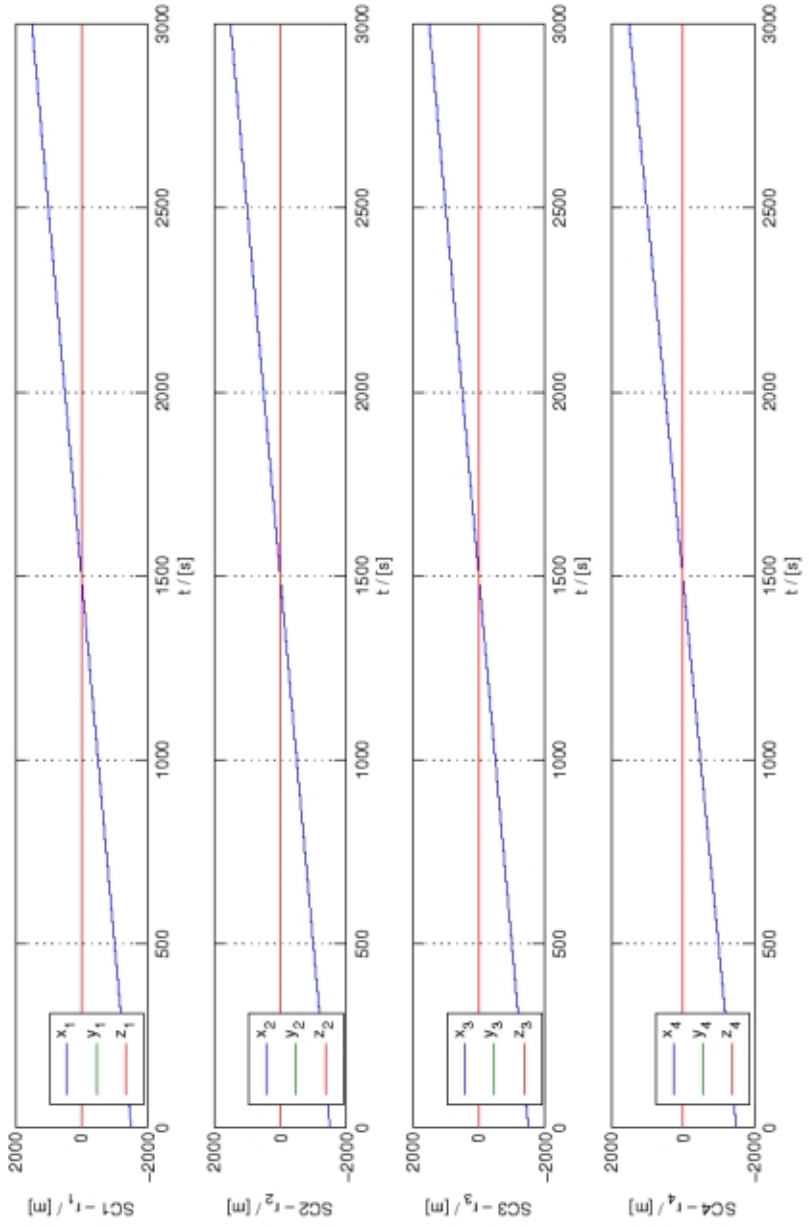

Figure 7.9: Small current sheets: $\vec{r}$ - $d = 1m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

7.3 Cylindrical current sheet with homogenous $\vec{j}$

7.3.1 Derivation of the magnetic field.

The infinite current tube is aligned along the z-axis. Let the radius of the tube be r_0 and let the current density $\vec{j}$ point in the positve z-direction.

First we calculate the magnetic field $\vec{B}$ inside the current sheet ($r < r_0$).

$$\int_C \vec{ds}\ \vec{B} = \mu_0 \int_A \vec{da}\ \vec{j} \tag{7.6}$$

Again we can use the high symmetrie and therefore we observe that $\vec{B}$ changes only in radial direction. One can rewrite eq(7.6) using polar coordinates $da = r\,dr d\phi$ to

$$B(r)\,2\pi r = \mu_0\, j_z \int_0^r dr\, r \int_0^{2\pi} d\phi. \tag{7.7}$$

Integrating and solving for $\vec{B}$ leads to

$$\boxed{B(r) = \tfrac{\mu_0}{2}\, r\, j_z \qquad r < r_0\ .} \tag{7.8}$$

Performing the same steps we get for the outer field ($r > r_0$)

$$\boxed{B(r) = \tfrac{\mu_0}{2} \tfrac{r_0^2}{r}\, j_z \qquad r > r_0} \tag{7.9}$$

From eq(7.8) we observe a linear growing $B(r)$. Outside the current tube $B(r)$ follows a $\propto \frac{1}{r}$ dependency as viewable in eq(7.9).

7.3.2 Plots and results

In this section are the plots for different configurations of the current sheet and the satellites.

Note that the values of the different parameters are far from values in nature!

This is only a phenomenological approach to show the limitations in two easy cases. The errors due to deviations from the tetrahedron shape and currents changing with time are not investigated!

Geometrical coherences of the Current Tube

The infinite current tube is aligned along the z-axis. The radius of the tube is given by r and the current density $\vec{j}$ points in the positve z-direction.
The satellite separation is given by s. The tetrahedron is moving along the x-axis from negative to positive values. The plane with the satellites SC1, SC2 and SC3 on its vertices is parallel to the yz-plane. SC4 penetrates the current tube first, followed

by SC1-SC3. For SC1 and SC3 $y = 0$. Therefore these satellites measures the highest values of $\vec{B}$.
For each plot, 3000 time steps have been used ($1s = 1m$, 1 time step=10 "measurements"). The means that the x-axis values vary from -1500 to $+1500$. Check *sat_pos.m* for more details.

Large current sheets - $s \ll r$

$r = 500m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

For very large currents sheets, the results of the *curlometer* are very accurate fig(7.10). The dimension of the current tube and the amplitude of the current density is well preserved ($d = 500$, $j = 10^{-6}\frac{A}{m^2}$). As expected $\vec{\nabla} \cdot \vec{B}$ is useable as an error estimate. This value is a constant outside the current tube and zero inside.
(The values of j_x and j_y are very small and probably caused by imperfectness of ©MatLabs equation solving method.)

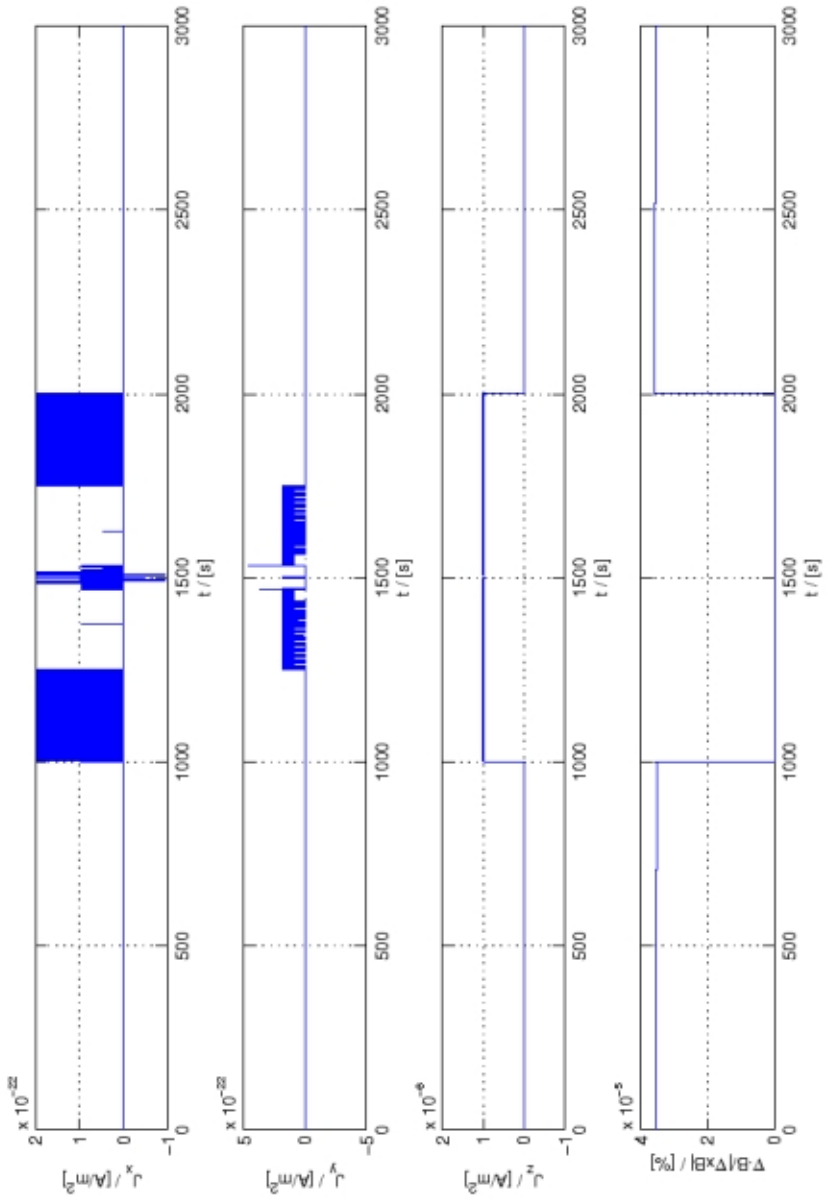

Figure 7.10: Large current sheets: $\vec{j}$ - $r = 500m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

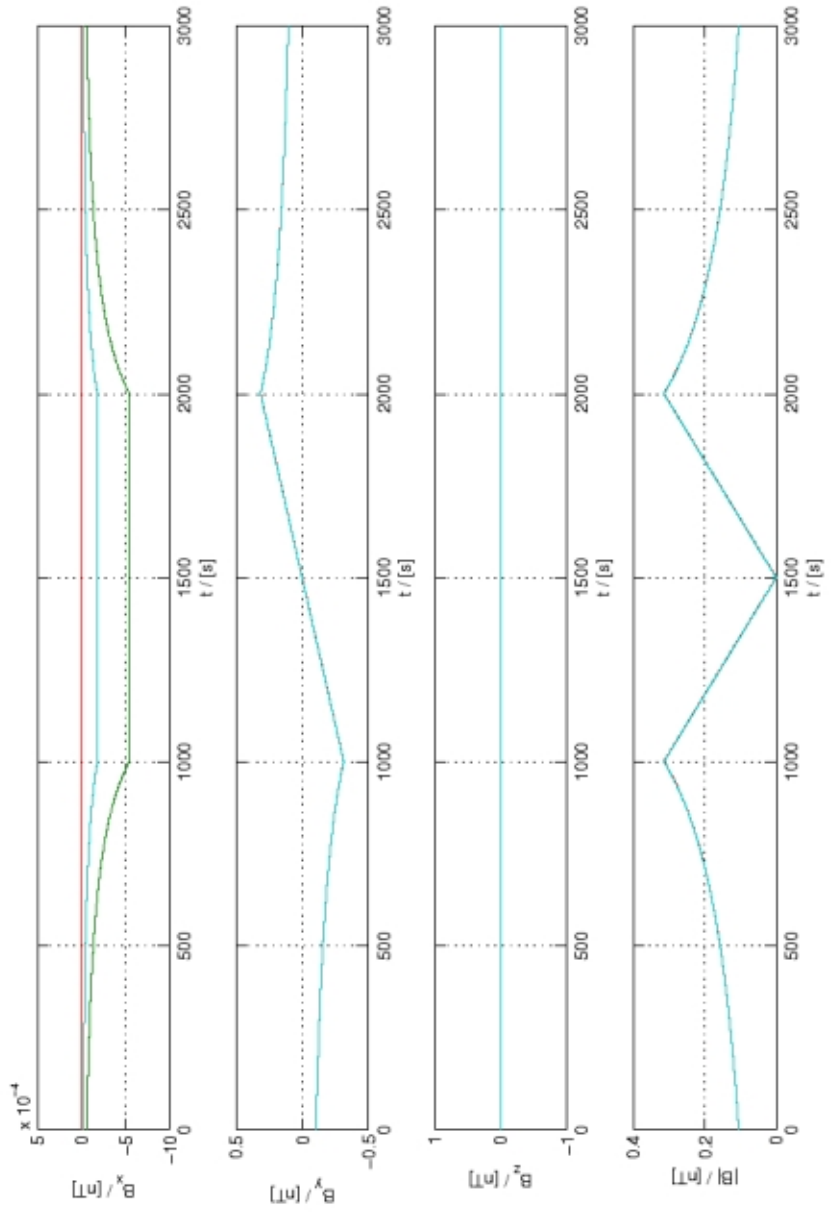

Figure 7.11: Large current sheets: $\vec{B}$ - $r = 500m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

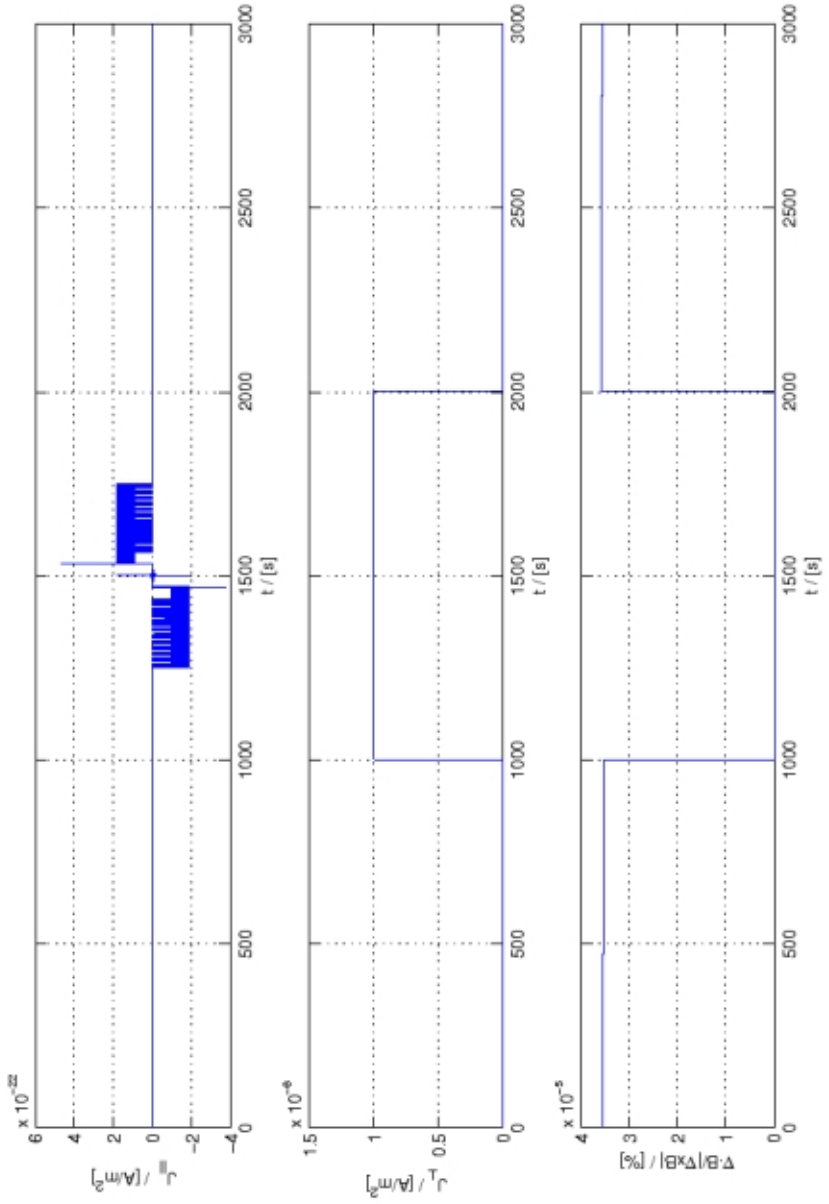

Figure 7.12: Large current sheets: $\vec{j}_{\perp/\parallel}$ - $r = 500m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

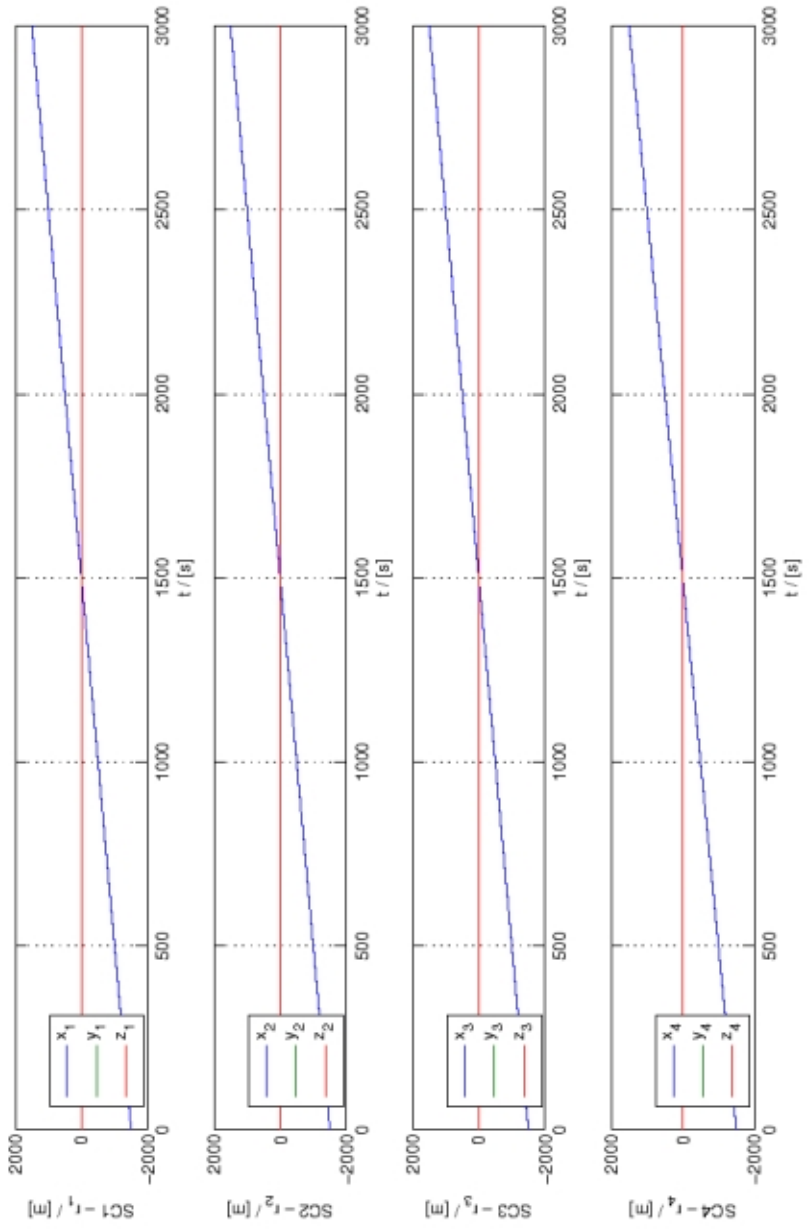

Figure 7.13: Large current sheets: $\vec{r}$ - $r = 500m$, $s = 1m$, $j = 10^{-6}\frac{A}{m^2}$

Medium current sheets - $s \approx r$

$r = 500m$, $s = 500m$, $j = 10^{-6} \frac{A}{m^2}$

Increasing the satellite separation leads to daubing of the edges fig(7.14). This can be understood by having a look on $|\vec{B}|$ **in fig(7.15).**
A very remarkable thing is the appearance of a backward current at around $750s$**.**
The separation of the satellites is still smaller than the diameter of the tube. Therefore one gets proper results if all satellites are in the tube.
At this point $\vec{\nabla} \cdot \vec{B}$ **fails to give an error estimate.**

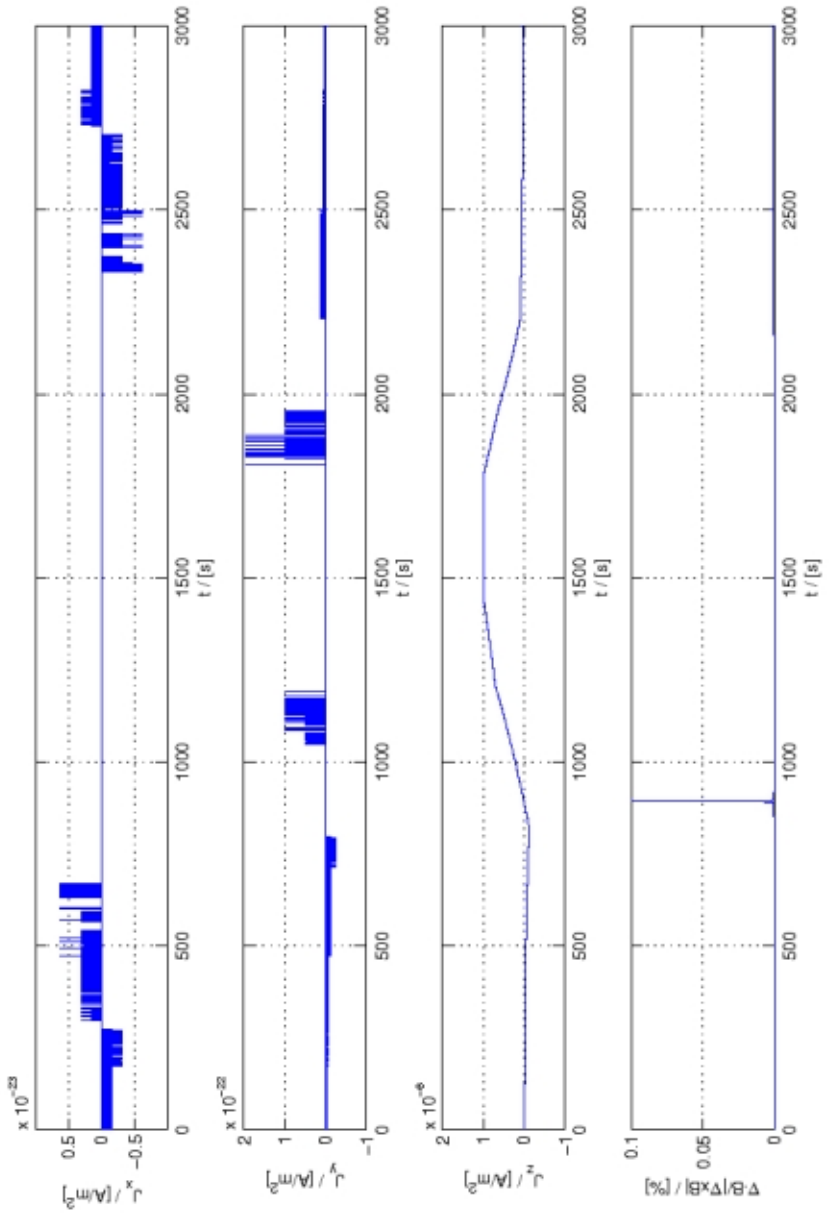

Figure 7.14: Medium current sheets: $\vec{j}$ - $r = 500m$, $s = 500m$, $j = 10^{-6}\frac{A}{m^2}$

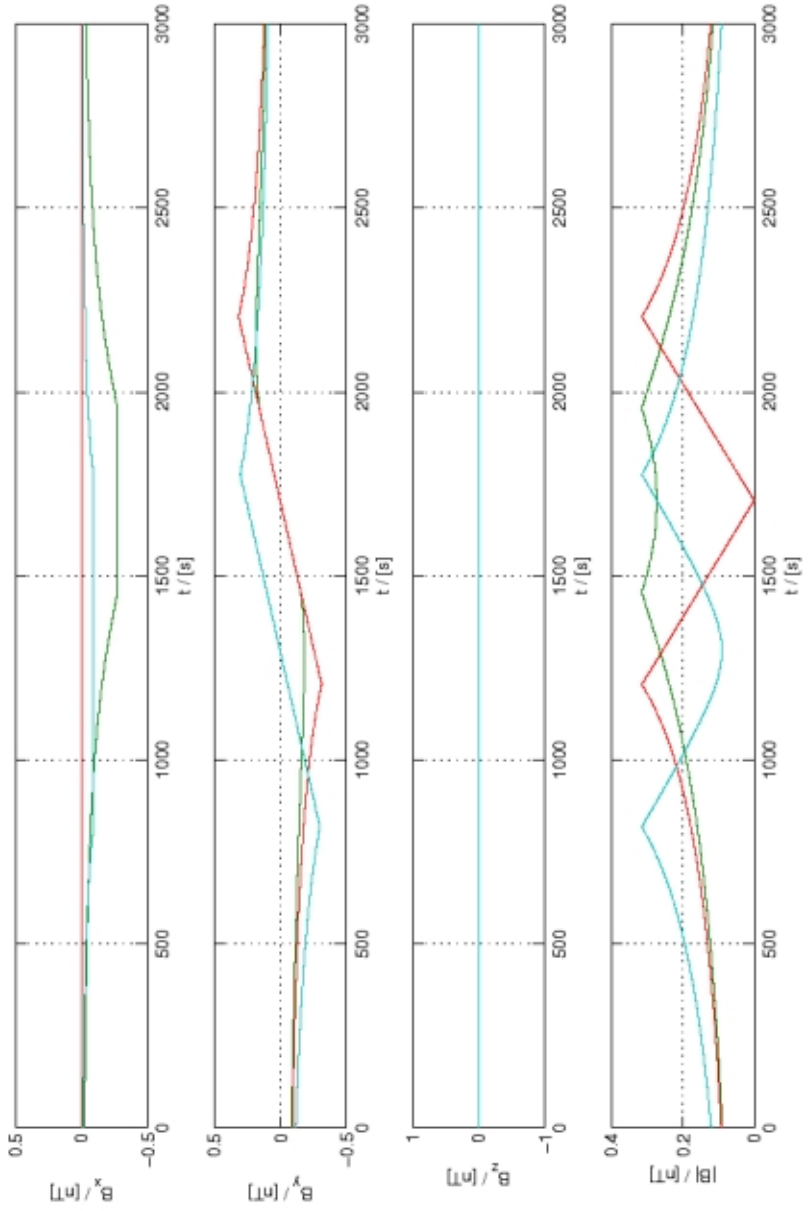

Figure 7.15: Medium current sheets: $\vec{B}$ - $r = 500m$, $s = 500m$, $j = 10^{-6}\frac{A}{m^2}$

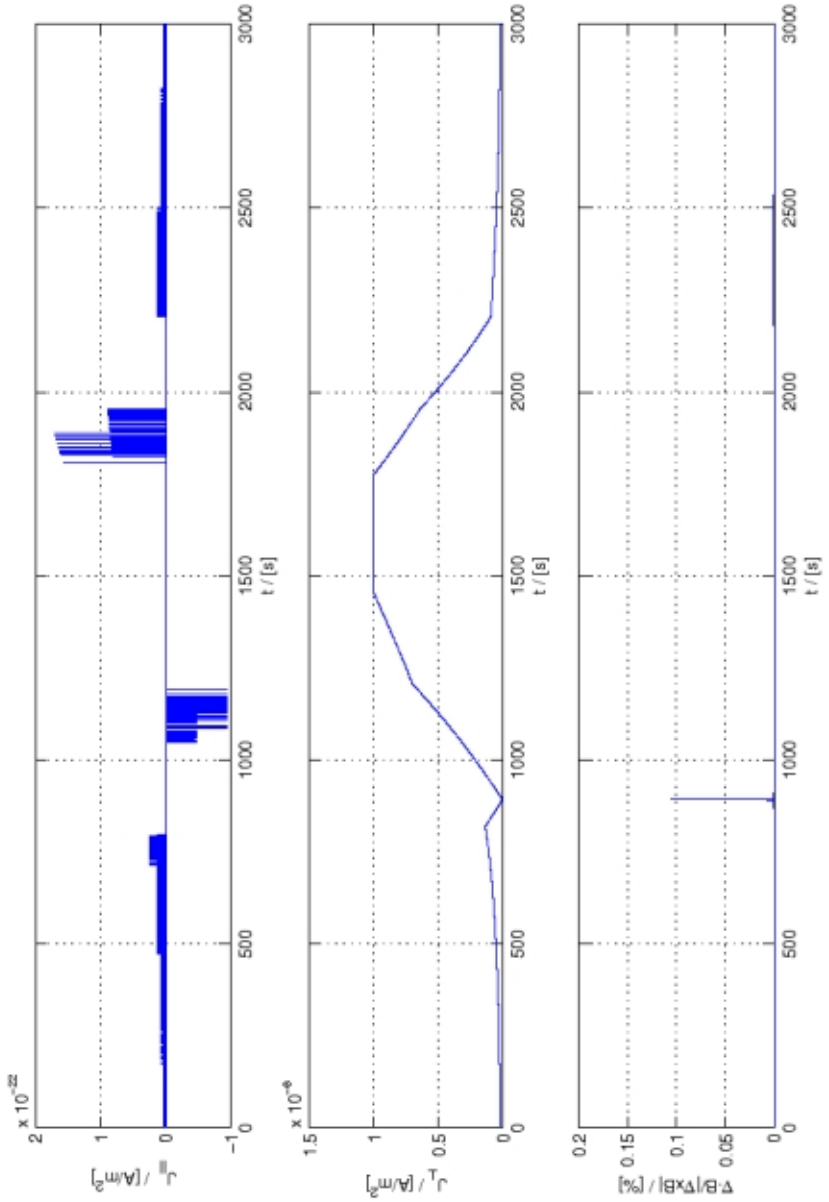

Figure 7.16: Medium current sheets: $\vec{j}_{\perp/\parallel}$ - $r = 500m$, $s = 500m$, $j = 10^{-6}\frac{A}{m^2}$

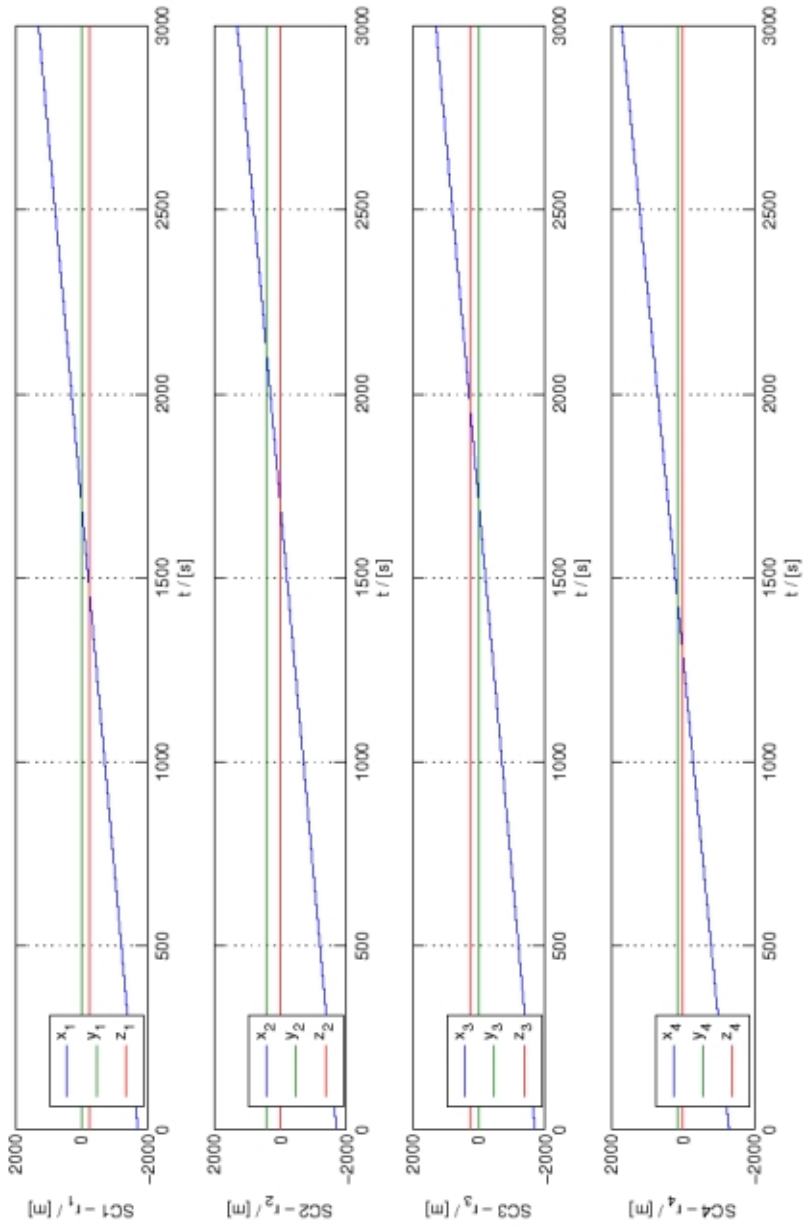

Figure 7.17: Medium current sheets: $\vec{r}$ - $r = 500m$, $s = 500m$, $j = 10^{-6}\frac{A}{m^2}$

Small current sheets - $s > r$

$r = 500m$, $s = 1000m$, $j = 10^{-6} \frac{A}{m^2}$

The form of the current tube fig(7.18) is more widenend compared to medium scales. The separation is now that big, that its impossible to fit all the satellites inside the tube at the same time. Therefore one observes that the maximum amplitude is 40% **smaller than expected. The backward current becomes increased.**

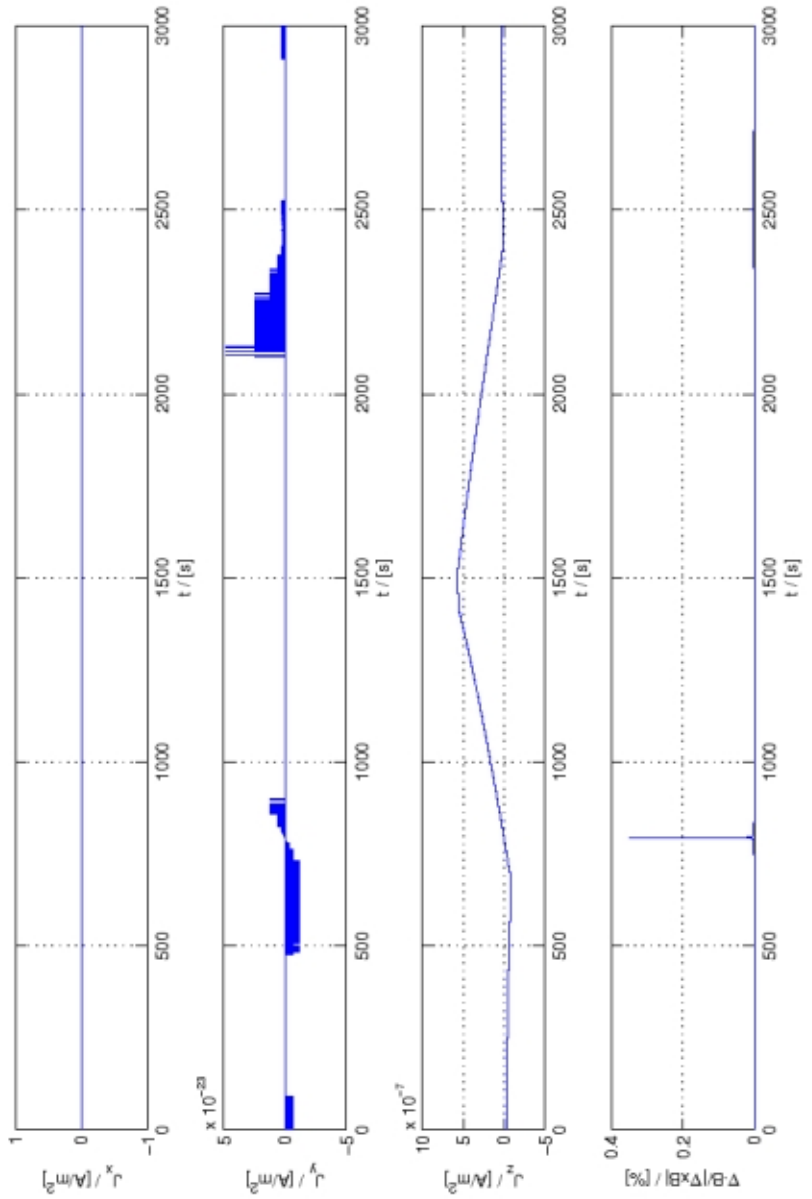

Figure 7.18: Small current sheets: $\vec{j}$ - $r = 500m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

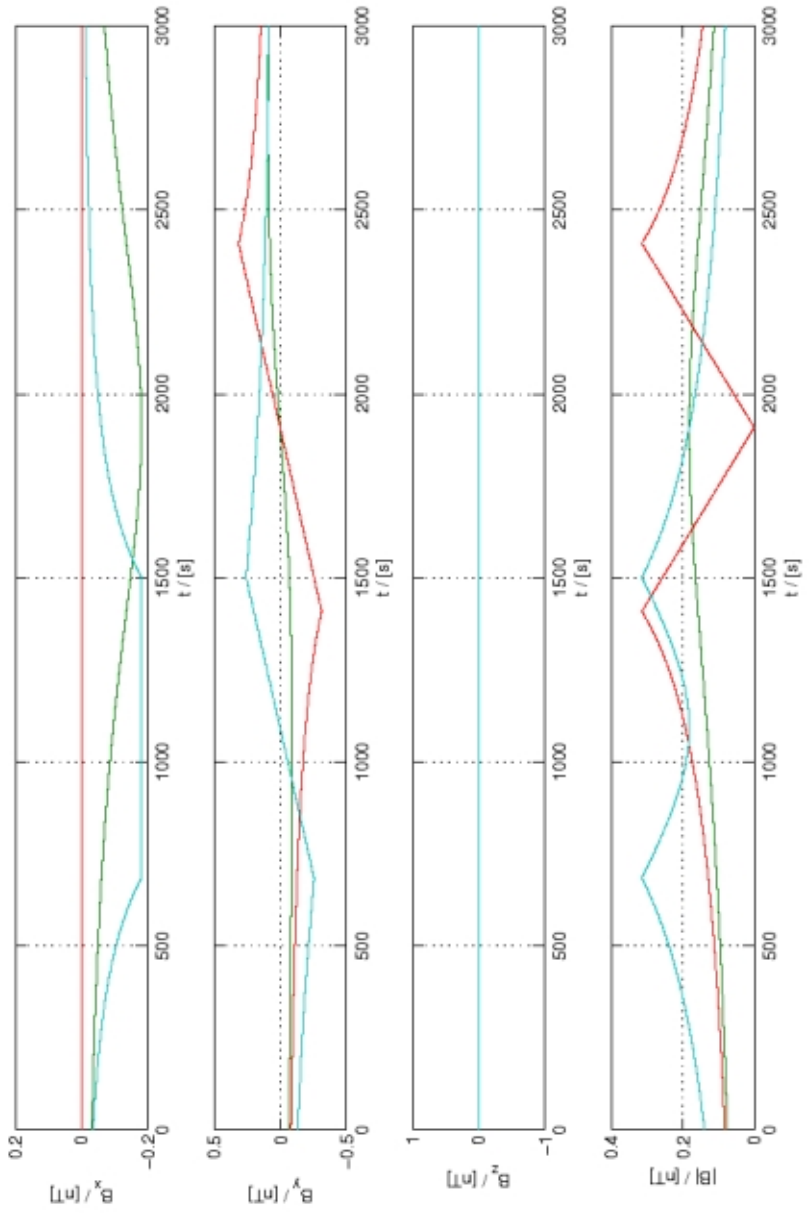

Figure 7.19: Small current sheets: $\vec{B}$ - $r = 500m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

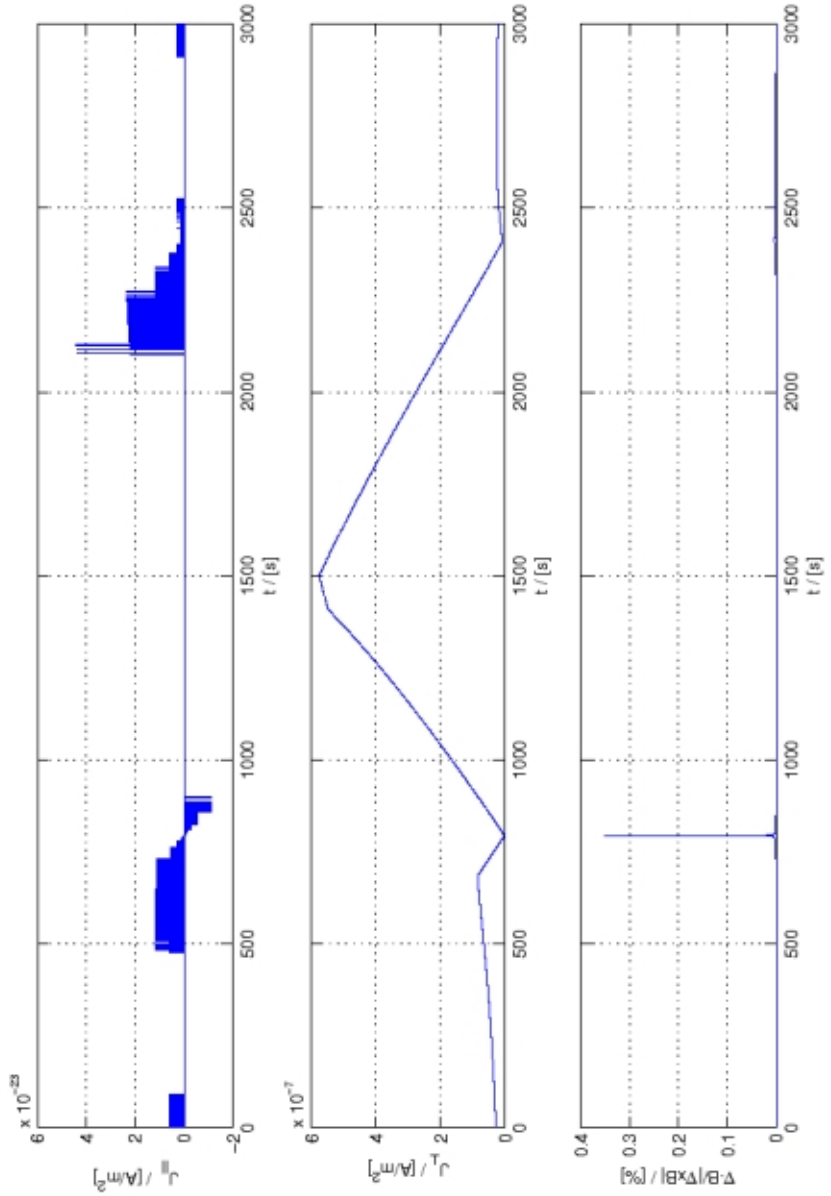

Figure 7.20: Small current sheets: $\vec{j}_{\perp/\parallel}$ - $r = 500m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

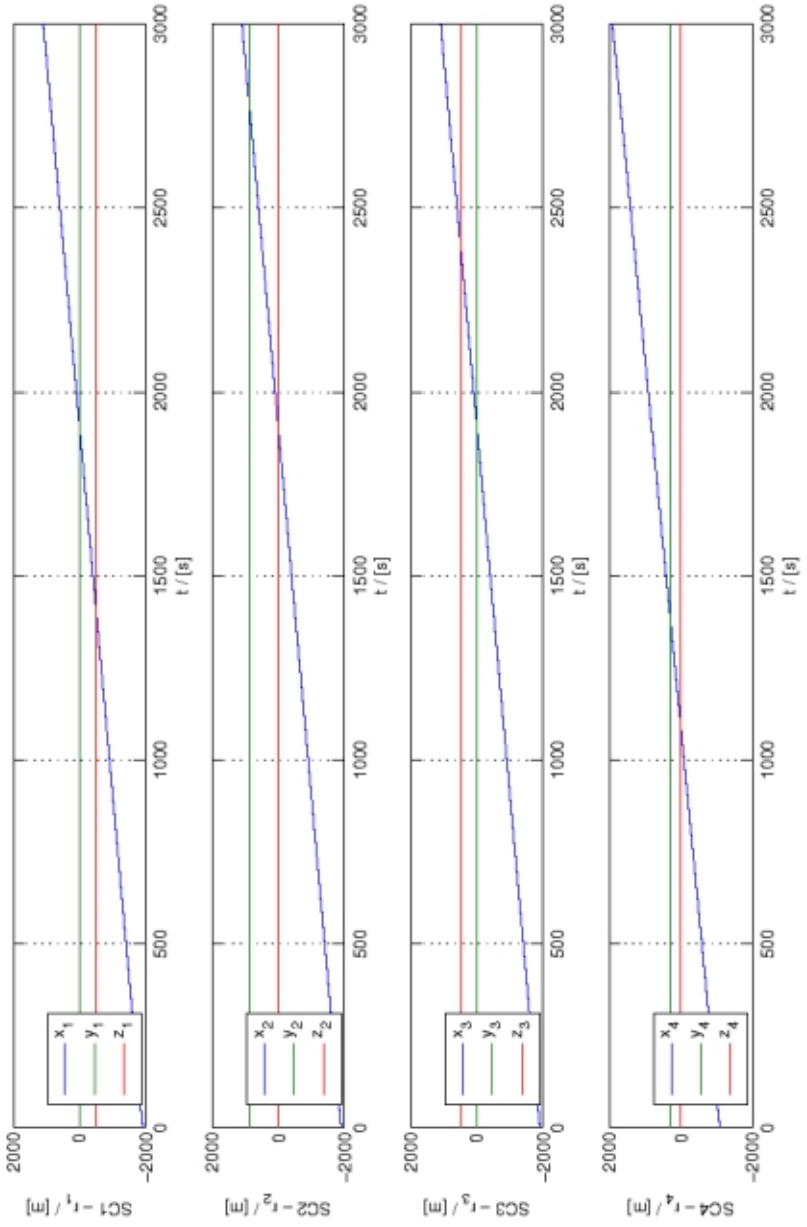

Figure 7.21: Small current sheets: $\vec{r}$ - $r = 500m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

Tiny current sheets - $s \gg r$

$r = 5m$, $s = 1000m$, $j = 10^{-6} \frac{A}{m^2}$

For very large satellite scales compared to the current sheet diameter the method fails at all. In the current density appears a point of discontinuity after around $1900s$**. The backward current is equal to the forward current. Due to the high satellite separation the maximum of the current density seams to be shifted. Note that SC2 and SC4 are to far away, compared to SC3 and SC1, to show a big current. The first current peak is therefore to small to be displayed.**

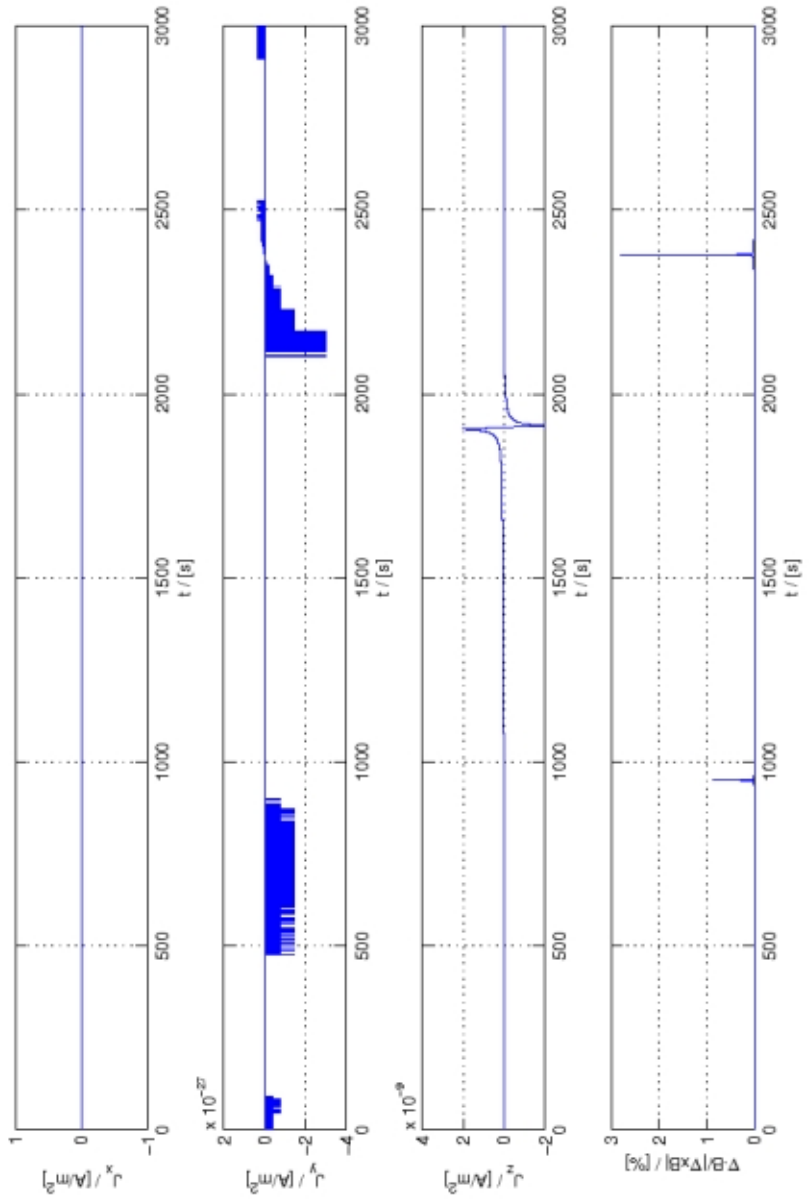

Figure 7.22: Tiny current sheets: $\vec{j}$ - $r = 5m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

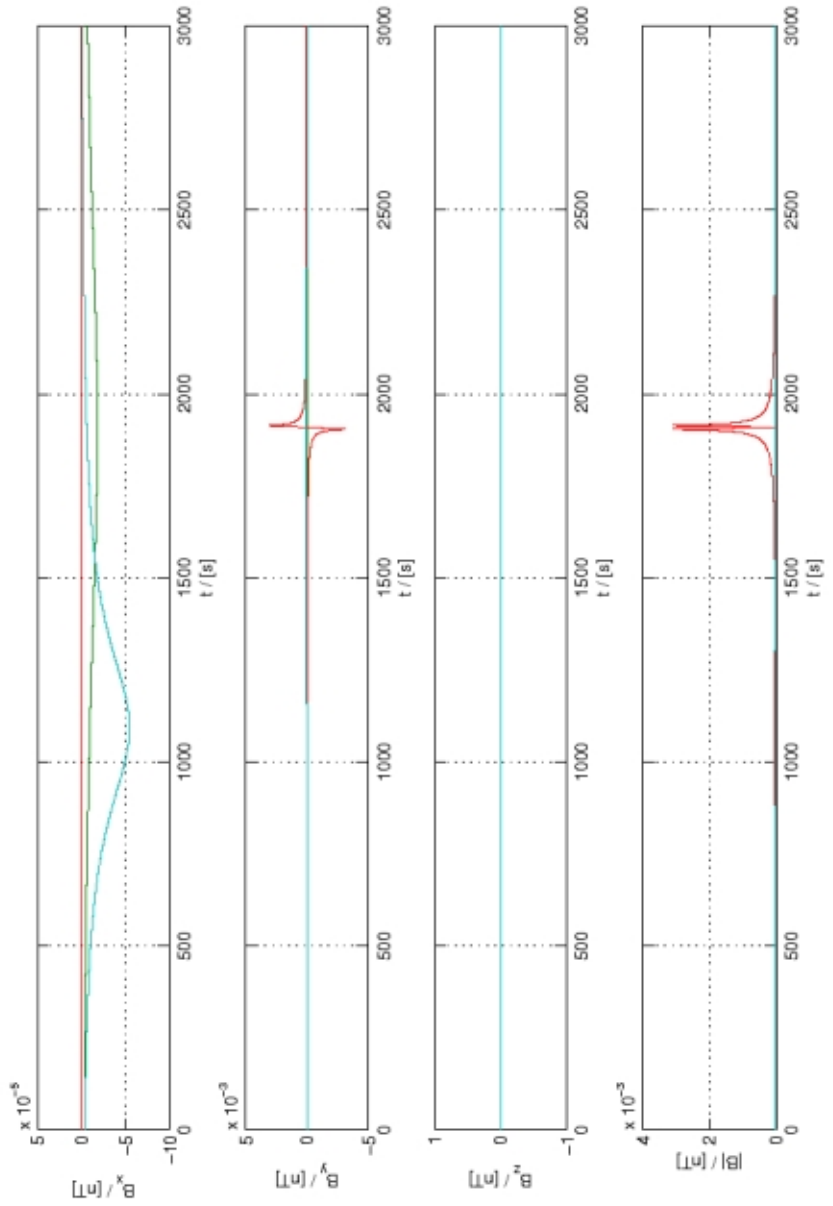

Figure 7.23: Tiny current sheets: $\vec{B}$ - $r = 5m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

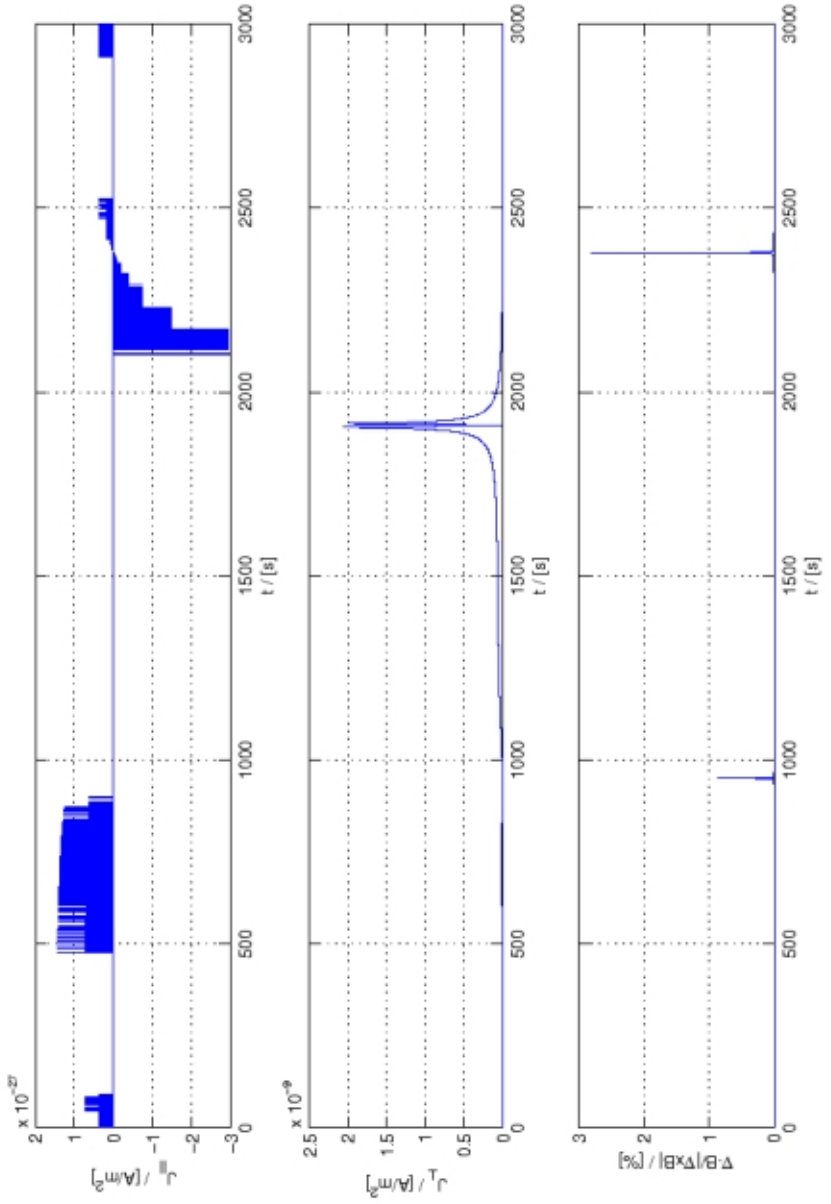

Figure 7.24: Tiny current sheets: $\vec{j}_{\perp/\parallel}$ - $r = 5m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

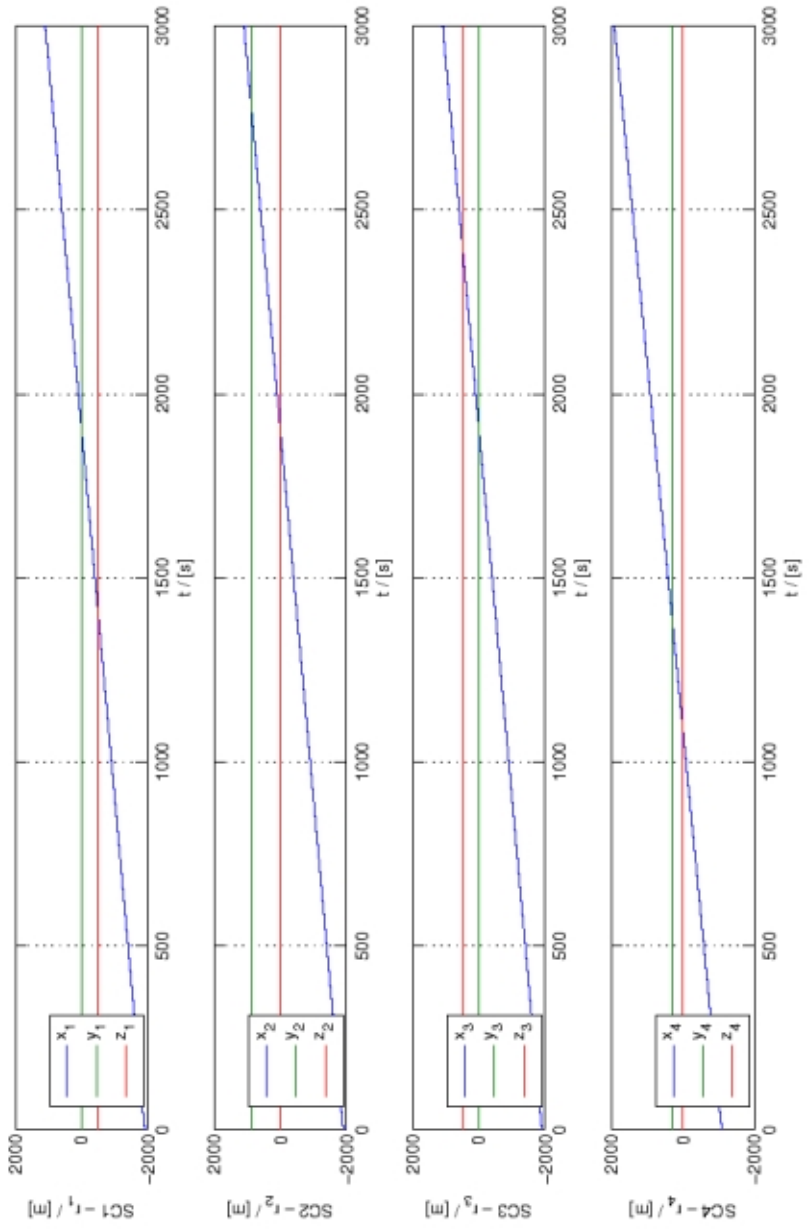

Figure 7.25: Tiny current sheets: $\vec{r}$ - $r = 5m$, $s = 1000m$, $j = 10^{-6}\frac{A}{m^2}$

7.4 Summary

Four main effects in case of a *current tube* with homogenous current density $\vec{J}_{av}$ can be observed:

1) Seemingly widening of the tube diameter by increasing the satellite separation.

2) Decrease of the current density amplitude with increasing satellite separation.

3) First appearance of backward currents if $s \approx r$.

4) $\vec{\nabla} \cdot \vec{B}$ not useable as an error estimate for thin current tubes.

In case of a flat current sheet, only the points 1) and 2) can be observed.
Due to the special character of these sheets the next step could be, to make this investigations with more artless current sheets.
It is possible that case 3) not occur in nature because of the shape of the *magnetopause*. Because of the large scales I estimate that the sheet is locally flat and therefore point 1) and 2) are the main sources of error.

Chapter 8

References

8.1 Bibliography

G. Paschmann, P. W. Daly (Eds.), *Analysis methods for multi-spacecraft data*, ISSI Scientific Report (electronic ed.), 2000
W. Baumjohann, R. A. Treumann, *Basic space plasma physics*, Imperial College Press, 1996

8.2 Links - www

ESA Science - Cluster II, *www.esa.int*
Imperial College Cluster Group, *www.sp.ph.ic.ac.uk/cluster/*
Cluster II in Oulu, *spaceweb.oulu.fi/projects/cluster/*
Mathworks - ©MatLab, *www.mathworks.com*
Mathworld - ©Mathematica, *mathworld.wolfram.com*