ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A DESIGN OF A TEST BED FOR CUBESAT ATTITUDE DETERMINATION AND CONTROL SYSTEM

M.Sc. THESIS

Mehmet Şevket ULUDAĞ

Department of Aeronautics and Astronautics Engineering

Aeronautics and Astronautics Engineering Programme

NOVEMBER 2016

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

A DESIGN OF A TEST BED FOR CUBESAT ATTITUDE DETERMINATION AND CONTROL SYSTEM

M.Sc. THESIS

Mehmet Şevket ULUDAĞ
(511131126)

Department of Aeronautics and Astronautics Engineering

Aeronautics and Astronautics Engineering Programme

Thesis Advisor: Prof. Dr. Alim Rüstem ASLAN

NOVEMBER 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KÜP UYDU YÖNELİM BELİRLEME VE KONTROL SİSTEMLERİ İÇİN TEST DÜZENEĞİNİN OLUŞTURULMASI

YÜKSEK LİSANS TEZİ

Mehmet Şevket ULUDAĞ
(511131126)

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Alim Rüstem ASLAN

KASIM 2016

Mehmet Şevket ULUDAĞ, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 511131126 successfully defended the thesis entitled “A DESIGN OF A TEST BED FOR CUBESAT ATTITUDE DETERMINATION AND CONTROL SYSTEM”, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Alim Rüstem ASLAN** Istanbul Technical University	....
Jury Members :	**Assoc. Prof. Dr. Emrah KALEMCİ** SabancıUniversity	....
	Asst. Prof. Dr. Cuma YARIM Istanbul Technical University	....

Date of Submission : 8 November 2016
Date of Defense : 29 November 2016

To my past, present and future family,

FOREWORD

I would like to thank my family for supporting me till this time for all these years. I also would like to thank; to my advisor Prof. Dr. A. Rüstem Aslan for sharing his experience and time, to Prof. Dr. Özcan Kalenderli for providing the Cenco Helmhotlz Cage, to Mustafa Erdem Baş, Sibel Türkoğlu and Demet Çilden for helping me with the topics and contents of my thesis, to Mehmet Deniz Aksulu and İsa Eray Akyol for their help in TeXStudio Coding.

November 2016 Mehmet Şevket ULUDAĞ
(Mechanical Engineer & Electrical Engineer)

TABLE OF CONTENTS

ABBREVIATIONS

A	: Amper
AC	: Alternating Current
ADCS	: Attitude Determination and Control System
cm	: centimeter
CMG	: Control Moment Gyroscope
DC	: Direct Current
DC-DC	: Direct Current to Direct Current
DOA	: Death On Arrival
ECEF	: Earth Centered Earth Fixed
EMC	: Electro Magnetic Compatibility
EPS	: Electrical Power System
FOV	: Field Of View
Hz	: Hertz
kHz	: kilo Hertz
MHz	: Mega Hertz
IGRF	: International Geomagnetic Reference Field
ITU	: Istanbul Technical University
IMU	: Inertial Measurement Unit
km	: kilometer
MCU	: Micro Controller Unit
OBC	: On-Board Computer
PWM	: Pulse Width Modulation
SGP4	: Simplified General Perturbations 4
SSDTL	: Space Systems Design and Test Laboratory
T	: Tesla
TLE	: Two Line Element
TEME	: True Equator, Mean Equinox
U	: Unit
V	: Volt

SYMBOLS

$\mathbf{B}$	: Magnetic Field
$\mathbf{B_{Cartesian}}$	: Magnetic Field in Cartesian Coordinate System
$\mathbf{B_{Spherical}}$	: Magnetic Field in Spherical Coordinate System
$\mathbf{B_{xy}}$	: Magnetic Field on XY Plane in Cartesian Coordinate System
$\mathbf{B_x}$, $\mathbf{B_y}$, $\mathbf{B_z}$,	: Magnetic Field Components in Cartesian Coordinate System
$\mathbf{B_r}$, $\mathbf{B_\theta}$, $\mathbf{B_\phi}$,	: Magnetic Field Components in Spherical Coordinate System
$\mathbf{d}$	: Diameter
$\mathbf{I}$	: Current
$\mathbf{L}$	: Inductance
μ_0	: Vacuum Permeability
$\mathbf{r}$	: Radius
$\vec{r}$	: Radial Vector
$\mathbf{R}$	: Resistance
$\mathbf{V}$	: Voltage

LIST OF TABLES

LIST OF FIGURES

**A DESIGN OF A TEST BED FOR
CUBESAT ATTITUDE DETERMINATION AND CONTROL SYSTEM**

SUMMARY

Owing to the recent developments in miniaturization and integration technologies, CubeSat's can handle more complex practical missions. Such missions require 3-axis control of the satellite along with a miniaturized 3-axis attitude determination and control systems (ADCS). With the QB0 project, the CubeSat's developed in ITU-SSDTL have also started to use such ADCS systems. Whether they are developed or procured, an ADCS system usually requires a suitable test bed to test the behavior and the performance of it. An ADCS system used for LEO missions usually operates within the Earth's magnetic environments. Therefore magnetic field sensors such as magnetometers are employed to measure the mediums' magnetic field or the change in the magnetic field to determine the orientation and motion of a satellite. Magnetic actuators such as magneto torquers are used to align with the magnetic field of the earth or to damp the tumbling motion of a satellite resulting from unbalanced torque distribution on it.

A major goal of the SSDTL is to have such a test bed available in the lab. In addition to related software, the major components are a Helmholtz cage and a suitable air bearing table.

With this in mind, the purpose of the present thesis is to aid the development of such an ADCS test bed designing first a suitable Helmholtz Cage system. The sizing of the test system depends on requirements such as the maximum mass of the satellite to be tested, coordinates of center of mass and disturbance levels to be counteracted. In the present thesis, first a comprehensive nanosatellites literature survey is conducted. The success rates and mission failure reasons are also investigated. Based on this preliminary study it is observed that 3U CubeSat's are generally adopted for most missions. Therefore a Helmholtz Cage that will house a 3U CubeSat is considered. A circular Helmholtz cage that will fit a 3U CubeSat is designed and analyzed. 1 axis and 3 axis Helmholtz cage cases are considered. Magnetic field lines present within the cage are demonstrated. Since the size of the Cage considered was just large enough to house 3U the magnetic field lines were not uniform enough. Therefore similar analysis is carried out for larger cage of double the size of the small one. Both are compared for uniformity of the magnetic fields.

Then a small square cage is designed and analyzed. Again, 1 axis and 3 axis Helmholtz cage cases are considered. Magnetic field lines present within the cage are demonstrated. The large square cage was found to be the best choice for testing a 3U CubeSat. As a result, a square Helmholtz cage which can be used to test a 3U CubeSat at LEO from 250 km to 1500 km.

KÜP UYDU YÖNELİM BELİRLEME VE KONTROL SİSTEMLERİ İÇİN TEST DÜZENEĞİNİN OLUŞTURULMASI

ÖZET

Gelişen teknoloji ve keşfedilen yeni metodlar ile birlikte, daha küçük uyduları tasarlamak ve üretmek mümkün hale gelmiştir. Bu yeni küçük uydulara küp uydu denmektedir. Bir küp uydu 10cmX10cmX10cm boyutlarında ve 1.3 kg kütlesinde olarak tanımlanmıştır. İlk çıkış amaçları öğrencilere bire bir uydu geliştirme tecrübesi edinmeleri için yapılmıştır. 2000'ler yıllardan bu yana küp uydu geleştirilmesi ve de fırlatmaları sürekli artmaktadır. 2012'de fırlatılan uydu sayısı 25 iken, 2017 yılı için planlanan uydu sayısı 311dir. Şu anda geliştirilmekte olan bütün küp uyduların %40'ı üniversiteler tarafından yapılmaktadır. Bu uydular çoğunlukla öğrenciler tarafından geliştirilmektedir. Bunun sonucu olarak uydulardaki risk artmaktadır. 2016 yılına kadar fırlatılan nano uyduların başarısızlık oranı %33'tür. Sadece üniversiteler tarafından geliştirilen uydularda kısmi başarıların da tam başarı olarak kabul edilmesi durumunda tüm küçük uyduların başarı oranı %40 civarında oluyor.

2010 yılına kadar yapılmış olan nanouyduların neredeyse yarısı 3U ve ondan küçük uydular olarak tasarlanmıştır. 2016 yılı verileri incelendiğinde fırlatılan ve planlananlar için bu oran %70'lere ulaşmaktadır ve toplam sayı 1000'den fazladır.

Uydu sayıları bu kadar artarken başarısızlık da artacaktır. Mevcut eğilim göz önüne alındığında uyduların fırlatılmadan önce daha çok test edilmesi gerektiği görülmüştür. Fırlatılan uyduların büyük çoğunluğu yörüngeye ulaşabilmekte fakat ardından kısa bir süre sonra işlevsiz hale gelmektedir. Buradan anlaşıldığı üzere termal-vakum testleri ve titreşim testleri uydunun sadece dayanıklılığını göstermektedir. Fakat uzun vade de uydunun herhangi bir yazılım ya da algoritma sonucunda başarısız hale gelip gelemeyeceği ya da farklı sorunlarda ne gibi sonuçların ortaya çıkacağı öngörülememektedir. Oranın azaltılması adına uyduların fonksiyonel testlerinin arttırılması gerekmektedir. Gerekli olan test ekipmanları uydulardaki alt sistemler temel alınarak belirlenebilir. Bir küp uydu başlıca elektrik düzenleme biriminden, pil biriminden, güneş panellerinden, yapıdan, yönelim belirleme ve kontrol sisteminden, uçuş bilgisayarından, modemden ve bilimsel yükten oluşmaktadır.

Yörüngedeki bir uydunun durumu göz önüne alınırsa, fonksiyonel olduğu zaman boyunca çeşitli etkiler altında kalmakta ve görevler yerine getirmektedir. Sürekli olarak bir manyetik alan etkisinde kalmaktadır, değişen açılarla güneş ışığına maruz kalmaktadır, yıldızlara ve de dünyaya bakarak konum algılamakta, fotoğraf çekmekte, kendisini yönlendirmekte, enerji üretip dağıtımını yapmaktadır. Tüm bunları yeryüzünde yapabilmek için bütünleşik bir test sistemine ihtiyaç duyulmaktadır. Örnek olarak dünyanın manyetik modelini gerçeklemek için Helmholtz kafesi kullanılabilir. Bu kafes ile modellenen manyetik alan içerisinde uydunun manyetometresi denenebilir, toplanan verilere göre yönelim belirleme ve kontrol sisteminin yörünge tahmini ve kontrolü denenebilir. Değişen açılarla güneş ışığına maruz kalabilmesi için

güneş ışığı benzetim düzeneği yapılabilir. Yapılan bu sistem ile güneş panellerinin üretimleri kontrol edilebilir, yönelim belirleme ve kontrol sisteminin alt birimleri ve algoritması kontrol edilebilir. Bir diğer sistem olarak dünya haritası ve de yıldız haritası ekranlarda oluşturularak uydu uzaydaymışçasına fotoğraf çekmesi sağlanabilir, yönelim belirleme ve kontrol sistemin yönelendirmesi, algoritması, tepki tekeri gibi sistemleri denenebilir. Tüm bu sistemlerin birleştirilebilmesi için ayrıca uydunun rahatça haraket edebileceği, düşük sürtünmeli bir düzenek gerekmektedir. Bu hava yatağı ile sağlanabilir. Bu sayede uydu üç eksende serbest olarak hareket edebilir.

Bahsedilen bilgiler ışığında yönelim belirleme ve kontrol sisteminin uzaydaki bütün görev ve durumlarda aktif ya da dolaylı olarak rol oynadığı görülmektedir. Bu yüzden yönelim belirleme ve kontrol sistemi için bir düzeneğin geliştirilmesi diğer sistemler içinde rahat bir başlangıç oluşturacaktır.

Yönelim belirleme ve kontrol sistemi küp uydularda gün geçtikçe daha çok kullanılmaktadır. Bunun sebebi hem sistemlerin fiyatlarının azalması hem de görevlerin zorluklarının ve hassasiyetlerinin artmasından kaynaklanmaktadır. Yönelim belirleme ve kontrol sistemi uydunun yönlendirilmesinde, konumunun belirlenmesinde, uydunun denge halinde tutulmasında, fotoğraf çekerken sabitlenmesinde ve itki sistemleri için yönlendirme oluşturmak üzere kullanılmaktadır.

Bir yönelim belirleme ve kontrol sisteminde birçok duyarga sistemi bulunmaktadır. Güneş ve ufuk senyörü uydunun hangi yöne doğru yöneldiğini, uydunun tutulma zamanlarının anlaşılmasında kullanılabilmektedir. Ebatlarından dolayı uydularda bu araçlardan birer adet olmaktadır. Odak noktası ayarlanmış kamera oldukları için güç ihtiyaçları vardır. Bunun dışında kaba güneş duyargaları da bulunabilir. Bu tarz duyargalar daha çok güneş hücreleri şeklinde olmaktadır. Uydunun her yüzeyinde en az bir tane konulmaktadır. Uydunun tam olarak hangi yüzeyinin güneşe baktığının anlaşılması ve de yörünge tahmin yazılımın hassasiyetinin arttırılmasında kullanmaktadır. Güneş hücresi şeklinde oldukları için kendi enerjilerini kendileri üretebilmektedirler ve bu da sistemi daha güvenli ve bağımsız yapmaktadır. Bir diğer alt sistem de ataletsel ölçüm birimidir. Elektronik, yazılımsal ve mekanik parçaların bir araya gelmesinden oluşmaktadır. İçerisinde manyetometre, ivmeölçer ve gyro bulunabilir. Hepsinin bir arada olmasından dolayı hassasiyetleri o kadar yüksek değildir. Manyetometre daha hassas ve daha düşük manyetik alanları da ölçebilir. Yıldız takip sistemleri de uydunun konumunu ve de yönelimini anlamak için kullanılmaktadır. Bu sayılan sistemler pasif sistemlerdir.

Yönelim belirleme ve kontrol sisteminde aktif yani hareketli elemanlar da bulunmaktadır. Bunlar uydunun sabit tutulması ve de döndürülmesi için kullanılmaktadırlar. Bunlardan en basiti manyetik eğleyicilerdir. Dünya çevresindeki manyetik alanı kullanarak sahip olduğu sargılardan akım geçirmek suretiyle kuvvet oluşturarak uyduyu sabit tutabilir ya da tepki tekerlerinin doyuma girmesini engellemek için sönümlenmelerine yardımcı olmaktadırlar. Ardından sırasıyla tepki tekeri, momentum tekeri ve kontrol moment gyroskobu gelmektedir. Bu sistemler arasında sadece küçük farklılıklar vardır. Tepki tekeri genelde kapalı olup ihtiyaç halinde yüksek hızlara çıkarak uydunun istenilen bir doğrultuya yönlendirilmesini sağlamaktadır. Momentum tekerleri ise sürekli olarak dönmektedir. Bu yüksek hızlı dönme uydunun yörüngede ilerlemesi sırasında uydunun sabit tutulmasını sağlamaktadır.

Uyduların yeryüzünde test edilebilmesi için tavsiye edilen başlangıç sistemi Helmholtz kafesidir. Çünkü diğer bahsedilen test sistemlerine kıyasla daha basit ve ucuzdur.

Bu tez çalışmasında helmholtz kafesi tasarlanmıştır. İlk olarak dairesel kafes sonlu elemanlar yöntemi ile analiz edilmiştir. Planlanan uydu boyutları değerlendirildiğinde 3U ve daha küçük uyduların daha çok olduğu görülmektedir. Bu bilgi ışığında 30cmx30cmx30cm'lük bir hacim içresinde oluşturulacak manyetik alan birçok uydunun ihtiyacını karşılayabilecek nitelikte olmaktadır. Helmholtz kafesleri sahip oldukları sargıların yarıçapı kadar mesafe ile iki sargının yerleştirilmesinden oluşur ve bu aradaki mesafede homojen alan oluşur. Yapılan sonlu eleman analizleri sonucunda 30x30x30cm3 lük hacmin 30cm sargı arası mesafesi yerine 60cm sargılar arası mesafeye sahip bir sistemin içerisine konması manyetik alanın homojen dağılımında içeriye konulacak sistem için çok büyük farklılıklar oluşturmuştur. İkinci olarak da kafes şekillerine göre benzetim ve analizler yapıldı. Kare ve dairesel kafes yapıları incelendiğinde kare yapıların, dairesel yapılara göre daha homojen olduğu görüldü. Bütçe ve zaman yetersizliğinden dolayı sistemin çalışabilirliğinin kanıtlanması amacıyla küçük bir kafes tasarlanmasına karar verildi. Tasarlanan kafes 5cmx5cmx5cm lük bir hacimde homojen bir manyetik alan oluşturacak şekilde benzetimi ve analizi yapıldı..

Mevcut Helmhotz kafesinde manyetik alanın belirlenebilmesi için öncelikle uydunun bir konumunun olması ve o konumdaki manyetik alan bilgilerinin elde edilmiş olması gerekmektedir. Bunu gerçekleştirebilmek için Dünya manyetik modeli IGRF ve yörünge ilerletici SGP4 yazılımları kullanılarak küresel koordinatlara bağlı olarak manyetik alanlar hesaplanmıştır. Ardından bu değerler kartezyen koordinatlara çevrilerek kafeste kullanıma uygun hale getirilmiştir.

Kafesin etkin bir şekilde çalışabilmesi için güç kaynaklarının dikkatli bir şekilde hesaplanması gerekmektedir Çalışacak olan güç kaynaklarının tepki süreleri, kafes içerisinde kullanılacak olan manyetometreden yavaş olmalıdır. Aksi takdirde ölçüm bilgisi tam gelmeden, sistem değeri tekrar değişecek ve de düzenlenecek bu da sürekli olarak sistemin yanlış konumdaymış gibi düzenleme yapmasına sebep olacaktır. Ayrıca sistemin hassasiyeti güç kaynağının anahtarlama elemanın anahtarlama frekansına ve onu kontrol edecek olan mikro denetleyicinin kristal frekansına bağlıdır. İkisi arasındaki oran adım büyüklüğünü vermektedir. Yani adım büyüklüğü ne kadar küçük olursa, manyetik alan değişimi o kadar hassas bir şekilde gerçekleştirilebilir. Planlanan işler olarak ilk önce masa üstü modelinin güç kaynağının üretilmesi gelmektedir. Ardından yazılım ile bağlantısı sağlanarak sistem bir bütün olarak denenecektir. Çalışması durumunda yazılımda bir değişiklik yapmadan sadece kafes ve güç kaynağı ölçeklenerek daha büyük ve asıl sisteme geçiş yapılabilir. Bu tez çalışması sırasında bu tezin çok fazla konuyu bir arada barındırdığı görülmüştür. Aslında bu çalışmanın 3 farklı proje olarak yapılması daha kaliteli ve etkin bir sistemin çıkmasında etkili olacaktır. Manyetik alan ve yörünge ilerletici, kafes tasarımı ve de güç kaynağı tasarımı olmak üzere üçe bölünerek daha detaylı çalışmalar yapılabilir.

1. INTRODUCTION

With the improved technology and new methods its possible to design and manufacture smaller satellites. These smaller satellites are called nanosatellites or picosatellites where in some cases they are called cubesats. A cubesat has been defined as a satellite with the dimensions of 10cmx10cmx10cm and mass of 1.3 kg [1]. First purpose was to give student hands on experience in satellites and to test new technologies in space. Since 2000, number of nanosatellite developments and launches are increasing as seen Figure 1.1. Cubesats are being developed by different companies or universities. All pico and nano satellites universities are developing nearly 40% of which can be seen in Figure 1.2.

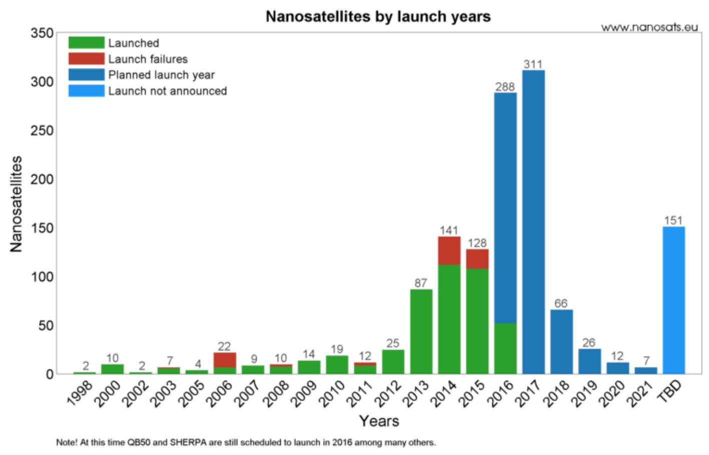


Figure 1.1 : Launched and planned nanosatellite launches [2].

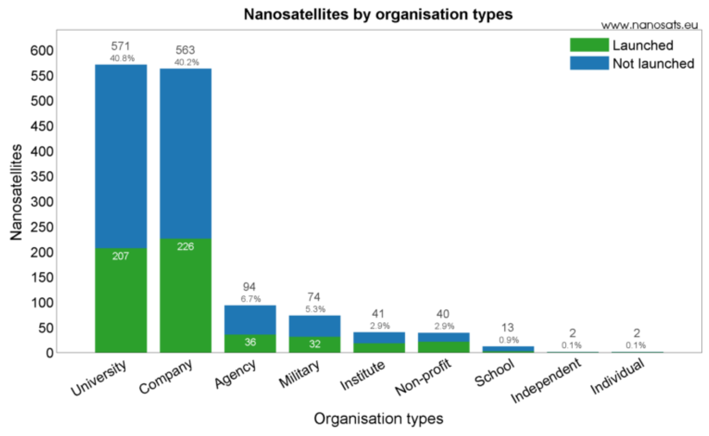


Figure 1.2 : Nanosatellites by organization type [2].

Being a university project and being developed by students; lack of experience and funding can cause immature development of the satellites which will cause failure. Nanosatellite failure rate is around 33% as shown in Figure 1.3. Status of the university nanosatellites can be seen in Figure 1.5, with respect to this it can be seen that even when accepting partial mission success as full mission success, success rate of university satellites are less than 50%.

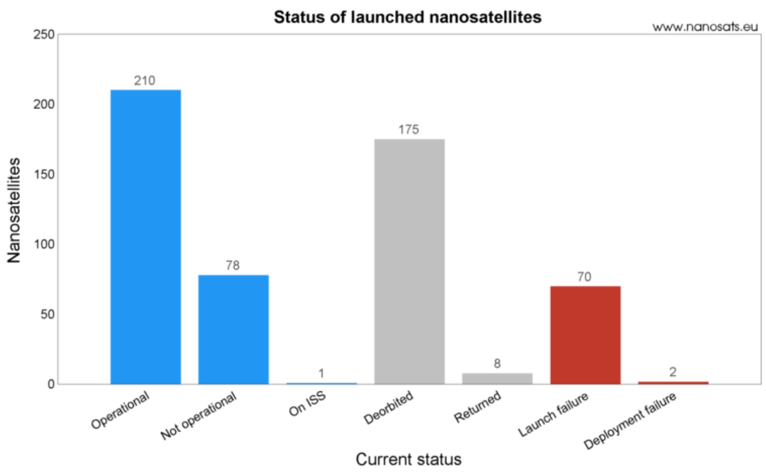


Figure 1.3 : Status of launched nanosatellites till 2016 [2].

With respect to current trend; after the launch of planned pico-nanosatellites it is possible that failure rate of pica-nanosatellites may increase. This can be seen by comparing Figure 1.4 and Figure 1.5. Success of a satellite depends on multiple factors. These factors are funding, timing, experience, testing, software and subsystem level failures. In order to decrease the failure rate and improve the status of satellites that will be launched, their time for functional testing must be increased and quality&number of dedicated equipment should be improved. These topics will be explained in this thesis.

Since a satellite consists of subsystems which are electrical power system, attitude determination and control system, on-board computer, telecommunications system, structure, thermal system and propulsion system. Dedicated test equipment should allow the project team to examine and test every subsystems individually.

1.1 Purpose

Attitude Determination and Control System of a satellite is one of the crucial and complicated part. ADCS's testing and proof of its algorithms will increase the success

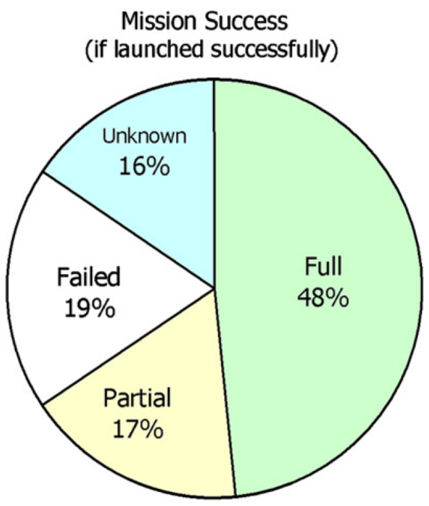


Figure 1.4 : Status of univeristy nanosatellites till 2010 [3].

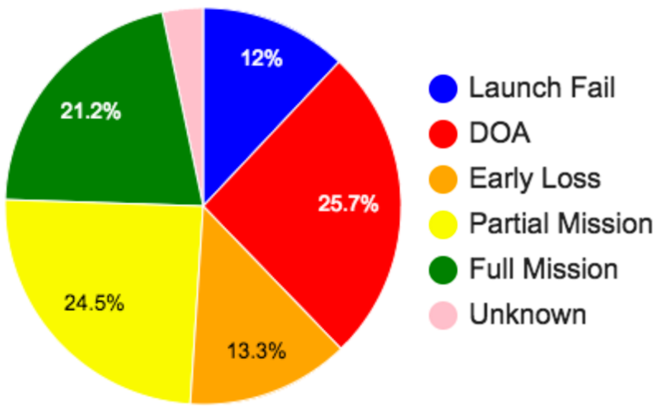


Figure 1.5 : Status of univeristy nanosatellites till 2016 [4].

rate of the whole satellite. Purpose of this thesis is to begin to build an in house nanosatellite testing facility for 3U or smaller satellites. 3U has been chosen with respect to launched and planned satellites (Figure 1.6 & Figure 1.7). This testing is not for environmental tests but for functional tests. In order to achieve this goal a series of systems are explained and suggested with respect to commonly used subsystems in satellites. Recommended systems are explained but thesis is mostly focused on ADCS's testing which can be done by Earth's magnetic model simulation. Due to its relatively cheap and easier design and building phases. Earth's magnetic model will be simulated by using a Helmholtz Cage which will be explained in detail in this thesis.

1.2 Organization

In this thesis; firstly brief introduction about success and failure of the satellites are given and ways to reduce the risk are explained. Then attitude determination and

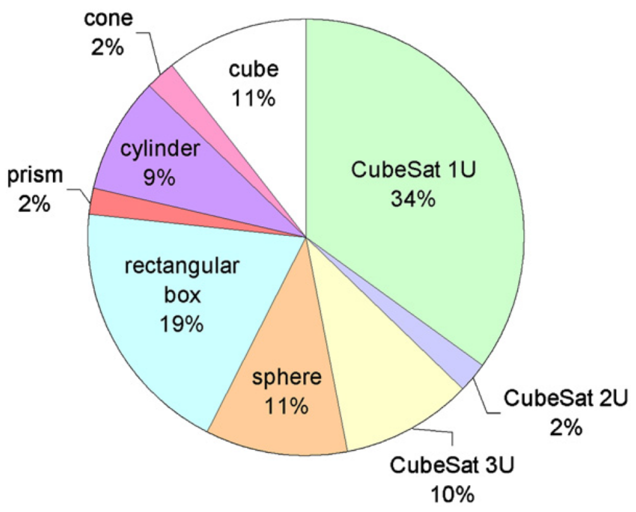


Figure 1.6 : Satellite form factors till 2010 [3].

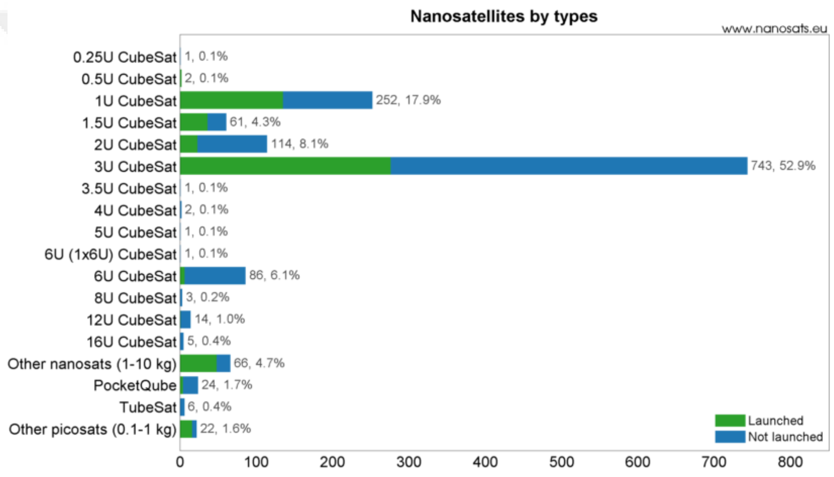


Figure 1.7 : Satellite form factors till 2016 [2].

control system is explained since it is the main subsystem for this thesis's designed test bed and its the subsystem with relations to every orbital action in Chapter 2. After explaining ADCS, a general test setup for a complete satellite is suggested in Chapter 3. With respect to that suggestion in order to design the Helmholtz Cage, magnetic model of the earth is introduced in Chapter 4. Detailed cage geometry and simulations are given in section 5.

2. ATTITUDE DETERMINATION AND CONTROL SYSTEM

Attitude determination and control system is one of the crucial subsystems in satellites. ADCS's task is to understand where the satellite is with respect to any desired location, try to rotate, maneuver or stabilize the satellite in to necessary position. This position is required for taking pictures, making measurements and orienting the satellite for thrust. With respect to its purpose and capabilities an ADCS can be so complicated. There are two types of attitude control one is a passive which can be done by eddy current and hysteresis rods, and the other is active attitude control. Active attitude control can be separated in to two sections as actuators and sensors. An active ADCS (Figure 2.1) basically does determination and control. This is achieved by four simple states which are sensing, algorithm, application of torque and control(Figure 2.2).

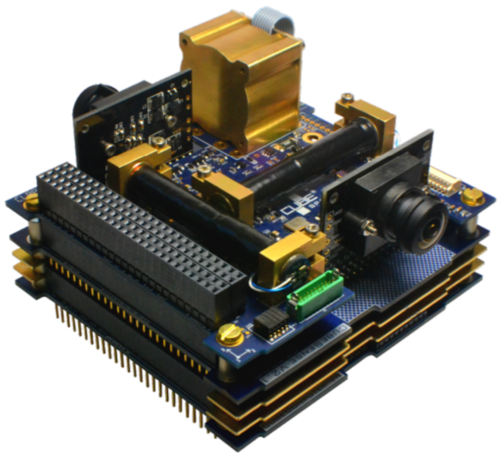


Figure 2.1 : An active ADCS [5].

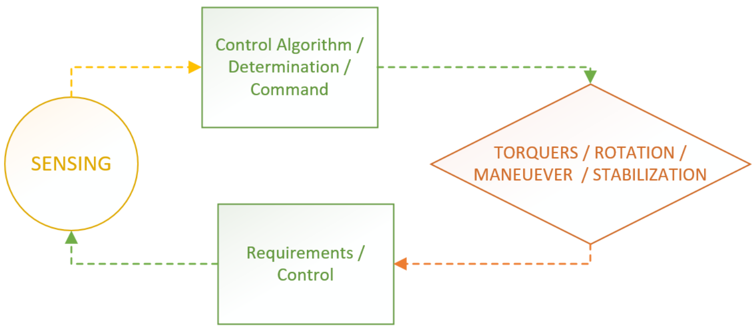

Control Algorithm / Determination / Command
SENSING
TORQUERS / ROTATION / MANEUEVER / STABILIZATION
Requirements / Control

Figure 2.2 : Basic flow chart of an active ADCS.

2.1 Sensors

Sensors in an ADCS are used for attitude determination. Depending on sensors to gather the necessary data, attitude can be determined by using special algorithms. Determination can be done by the system itself by using its preloaded data base. In order to increase the accuracy special commands can be applied on the system from a ground station.

2.1.1 Sun sensor & Earth/Horizon sensor

These sensors are basically cameras where one detects the position of the Sun relative to satellite and other detects the position of the Earth. Their sensitivity is dependent on its field of view(FOV) and needs power to operate. Especially in cubesats there is only one of each sensor on a spacecraft due to limited volume. One of the cameras on Figure 2.1 is a sun sensor and other is a horizon sensor. They are really for important power management because they help the satellite to determine if its in eclipse or not. Main difference between a sun and a horizon sensor is that sun sensor deals with highly dense light from a point source and horizon sensor deals with low light from the limb of the Earth's atmosphere.

There are also coarse sun sensors. Coarse sun sensor is a small solar cell to sense the position of the sun relative to spacecraft. Since they are small solar cells; they do not require any additional power source and can have almost hemispherical field of view depending on their configuration. They are used to increase the determination accuracy. Coarse sun sensors are placed at least one on each plane of a spacecraft while there will be a one sun sensor on one side of the spacecraft.

2.1.2 Inertial measurement unit

Inertial measurement unit(IMU) is set of digital sensors to be used in the satellites(Figure 2.3). IMU is a small integrated unit which consists of accelerometers, gyros and depending on their type they can also have small magnetometers. They need calibration time to time in order to increase their accuracy and to get correct data. In order to get more accurate data individual gyros can be used in satellite. IMUs need calibration to increase the measurement quality. This can be achieved in orbit or on surface before launch.

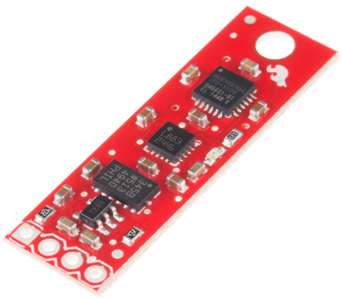


Figure 2.3 : An IMU with accelerometer, gyros and magnetometer [6].

2.1.3 Magnetometer

Magnetometer(Figure 2.4) is for measuring the magnetic field in satellites current position. It loses its capability as the satellite has a higher altitude because of decreasing magnetic field. External magnetometers are more accurate and precise equipments with respect to magnetometers inside IMU. They are used to determine the position of the satellite around the Earth by comparing the measurement with Earths magnetic field data which is embedded in the satellite or by commands from ground station.

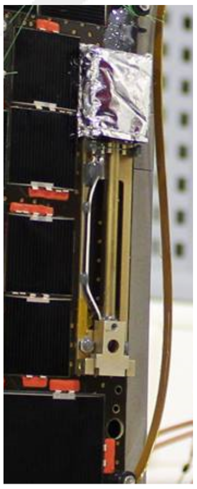


Figure 2.4 : A Magnetometer placed on BeEagleSAT.

2.1.4 Star tracker

Star trackers are the most complex and least used sensor units in cubesats due to their cost and since they are not needed for most pico-nanosatellite missions. This sensor is used in space missions which needs more precise and accurate determination. Star trackers have the map of the sky like a catalog. They took picture of the space and compare the positions of the starts with respect to its preloaded catalog. With comparison of these picture star trackers are able to understand the satellites position during orbit.

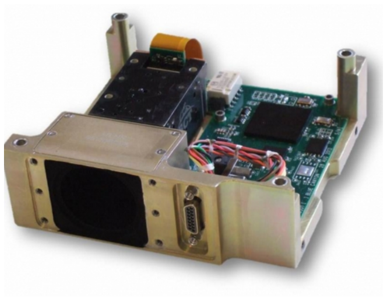


Figure 2.5 : A Star tracker as a subsystem with dimensions of 10cmx10cmx3cm [7].

2.2 Actuators

Actuators are the parts which can be actively controlled. These parts in cubesats are mostly for rotational maneuvers except thrusters which can be use for translational movements too. Different types of actuators are discussed below.

2.2.1 Magnetorquer

Magnetorquers are the cheapest and most basic of an ADCS with respect to other actuators. They interact with the Earth's magnetic field in order to achieve the necessary task. Since they interact with magnetic field; as altitude increases their efficiency (torque capability) decreases due to decreasing magnetic field. They can be used for stabilization, active control and momentum damping. A magnetorquer is either a coil(Figure 2.6) or a rod(Figure 2.1 & Figure 2.6).

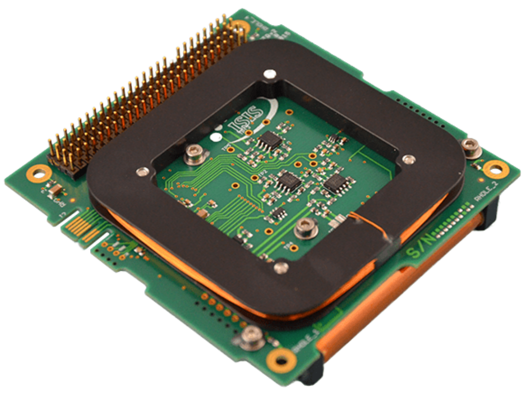


Figure 2.6 : A magnetorquer coil on upper side and a rod on lower side [8].

2.2.2 Reaction wheel

Reaction wheels are commonly used for rotation and stabilization of a cubesat. A reaction wheel starts to work by speeding up or down, in order to create the necessary torque to rotate the satellite to point a necessary direction. A reaction wheel can be seen on the upper right part of the Figure 2.1.

2.2.3 Momentum wheel

Momentum wheel is a wheel spinning at high speeds continuously where they make satellite more stable even in attitude changes during its flight. Since they are continuously spinning their power consumption rates are much higher than reaction wheels.

2.2.4 Control moment gyroscope

Control moment gyroscope(CMG) is a device which is a combination of a reaction wheel and a momentum wheel. It spins at high speeds again for stabilization but also has the capability rotate axis of the spinning wheel in order to supply the needed torque for rotation of the satellite. They are not being used in cubesats right now beacuse of they require high power and large volume inside the satellite.

2.2.5 Thruster

Thrusters are the least used active elements in ADCS due to their need for fuel/gas tank and has a limited time of usage with respect to other actuators. But they have good accuracy and fast reaction time.

3. RECOMMENDED TEST SETUP FOR A CUBESAT

As mentioned in chapter 1(Figure 1.4 & 1.5), failure rates and number of satellites are increasing constantly. There are standards, on going studies, papers and meetings about environmental and functional test. But since satellites consist of different subsystems there should be a distinct test bench for functional check of the every subsystem itself and also as the whole satellite. A satellites basic subsystems are OBC, EPS, ADCS, camera, communications, thermal, propulsion and structure. Since structure is fully mechanical part, its functional testing will not be discussed. In its orbit a cubesat is under the effect of magnetic fields, makes basic rotation or stabilization maneuvers. Gathering raw data or picture for scientific purposes or for estimation of orbit. Nearly all of the actions or work a satellite does in orbit can be done as functional tests on ground before going to space.

3.1 Helmholtz Cage

Magnetic field of Earth is known and measured by previously sent satellites. Their data is applied in to dedicated functions which are being updated every 5 years. By using algorithms which will be explained in chapter 4; 3 pair of Helmholtz coils can be used to generate 3 axis magnetic field around the satellite. From this generated magnetic field magnetometer tests can be performed which will eventually be used in an ADCS to estimate its orientation; or by also using air beds magnetorquers can be tested.

3.2 Air-Bearing Table

Air bearing table is the second stage. In order to give satellite the ability to rotate freely; gravitational force and friction needs to be eliminated. (Since Zero-G planes cannot fly long enough to test a satellite for a day continuously.) An air-bearing table is used to create a cushion of air between interfacing surfaces as a result of this decreasing friction between surfaces. An air bearing table is optimum solution. By using an air-bearing table, satellites' reaction wheels can be tested, satellite can rotate more freely and can take pictures of the desired direction where camera can be tested.

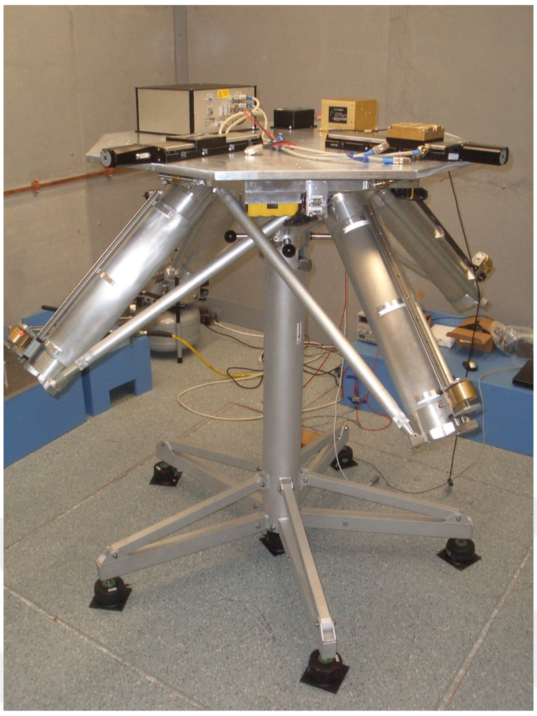


Figure 3.1 : An air bearing table with counter weights [9].

3.3 Star Map

Star trackers are now being used in cubesats too. In order to test them on Earth, there could be screens to display embedded star maps with respect to satellites, earth and stars relative positions. By doing this system in addition to star trackers; ADCS estimation& determination, reaction wheels, camera can be tested.

3.4 Earth Map

With screens to display the map of the Earth with respect to satellites position can help to test ADCS, camera, pointing of the satellite, taking videos, making swaps and controlling coverage areas of the satellites.

3.5 Solar Simulator

Solar simulator is a device with special light and optics, in order to create a sun like beam. These beams will be pointed towards to satellite to create the necessary irradiance similar to in orbit values. By having such an equipment in testing facilities power generation of the solar panels can be tested, power point tracking and working efficiency of the EPS can be controlled. Looking at the sun or gathering data from course sun sensors can also be used to test the ADCS.

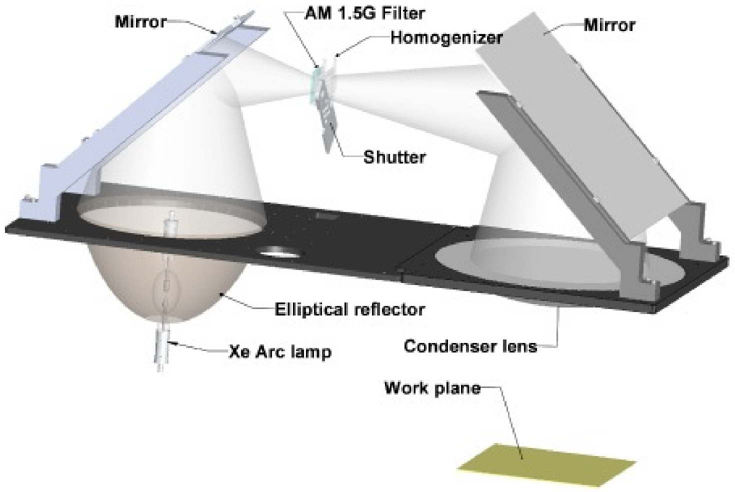


Figure 3.2 : Sun simulator schematic [10].

3.6 Satellite Simulator&Control Desk

Real hardware will be tested in these systems but in order to increase the functional test in detail a satellite simulator is needed. This is going to be used to view the satellite on a screen just like the way in its orbit. Rotations, relative positions and orbital parameters can be seen for the satellite. Last item is a control desk to run all the necessary algorithm for orbit propagation and control of other elements such as solar simulator, maps and Helmholtz Cage. By looking at the big picture these can even be done with a low-budget hardware with respect to satellites itself.

3.7 Recommendation

As mentioned in previous sections there are a lot ways to make a functional test for every subsystem and the satellite's itself. It can be seen that ADCS system is one of the crucial systems because it gets affected from nearly every variable. Not all the recommended test setup is going to be explained in this thesis. But as Helmholtz Cage is less complex with respect to other test setups. Using all of these test equipment together will make the cubesat operator to fully test its satellite and launch it with more confidence and less risk.

4. MAGNETIC MODEL OF EARTH

The magnetic field created by the Helmholtz cage needs to be similar to magnetic field which the satellite will get affected during its orbit. This has two components where one of them is orbit propagator and the other one is the Earth's magnetic model. In order to enable the system to generate desired magnetic field, satellites position information should be known. This will be obtained by using simplified general perturbations 4 (SGP4) for orbit propagation [12]. After acquiring the position data necessary transformations between different coordinate frames need to be done. This change is from tru equator, mean equinox (TEME) to Earth centered Earth fixed (ECEF). ECEF coordinate frame gives the position of the satellite in latitude-longitude-attitude. This is needed because next step is calculating the magnetic field directions and magnitudes. To do this international geomagnetic reference field (IGRF) is going to be used for the Earth magnetic model. All of the necessary MATLAB codes are given in the appendix. From Appendix A.5 to A.7 are Tsyganeko's IGRF, from Appendix A.8 to A.27 are Vallado's SGP4, and from A.28 to A.35 are coordinate frame transformation codes are given.

4.1 Calculation of Position and Magnetic Field

First of all a two line element (TLE) is needed. TURKSAT 3U SAT is chosen as an example. TLE value is used in SGP4 by applying satellites velocity and position information. For SGP4, MATLAB codes of Vallado are used and for IGRF, codes of Tsyganeko is used [11].

TLE of the satellite is;

1 35935U 09051E 10036.22182550 .00001931 00000-0 48744-3 0 1317

2 35935 98.3337 135.0110 0007520 227.8477 132.1769 14.52155217 19587

In order to create the link between codes, a root code has been written which is presented in Appendix A.1. After using the necessary codes for 60 minute time

Table 4.1 : Magnetic field components in spherical coordinates and their components in ECEF.

B_r	B_ϕ	B_θ	B
B_{rx}	$B_{\phi x}$	$B_{\theta x}$	B_{rx}+$B_{\phi x}$+$B_{\theta x}$
B_{ry}	$B_{\phi y}$	$B_{\theta y}$	B_{ry}+$B_{\phi y}$+$B_{\theta y}$
B_{rz}	0	$B_{\theta z}$	B_{rz}+0+$B_{\theta z}$

line; latitude, longitude and attitude of the satellite is known and from there magnetic fields are calculated in spherical coordinates. But since the cage is going to work as 3 perpendicular axes in ECEF those values needed to be transformed to Cartesian coordinates. In Figure 4.1 magnetic field in spherical coordinates can be seen.

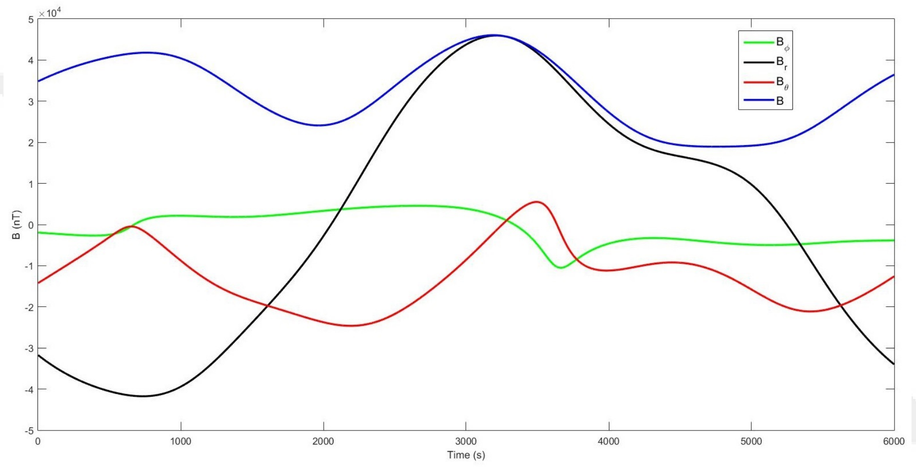


Figure 4.1 : Magnetic Flux of TURKSAT-3USATs orbit in spherical coordinates.

An example of magnetic field directions which are generated by using IGRF; and ECEF coordinate frame can be seen in Figure 4.2. In order to transform these values to ECEF frame every component should be considered individually. After the calculations, magnetic field has 3 components(R radial, ϕ longtitude in radians, θ colatitude in radians). B_r is positive outwards which has an effect on all 3 ECEF axes, B_ϕ is eastwards which has an effect on X and Y axes, B_θ is southwards which has an effect on all 3 ECEF axes X, Y and Z(Figure 4.2).

B_r is positive outwards, with respect to its direction its ECEF components are calculated in equations (4.1) to (4.4) with respect to notations shown in Figure 4.3.

$$B_{rz} = B_r cos\theta \tag{4.1}$$

$$B_{rxy} = B_r sin\theta \tag{4.2}$$

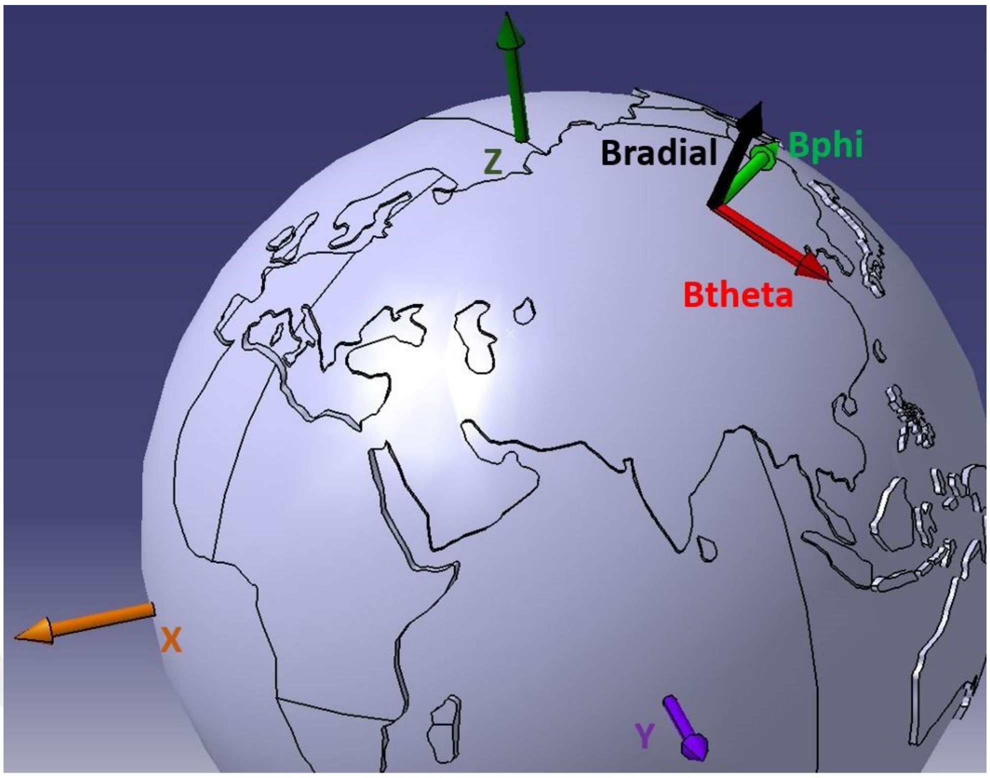


Figure 4.2 : Magnetic field axes and ECEF coordinates

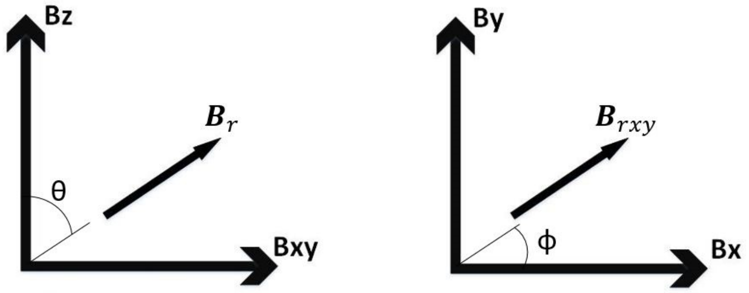


Figure 4.3 : B_r Component of magnetic field calculated by IGRF and its components in ECEF.

$$B_{rx} = B_r sin\theta cos\phi \tag{4.3}$$

$$B_{ry} = B_r sin\theta sin\phi \tag{4.4}$$

B_ϕ is positive outwards, with respect to its direction its ECEF components are calculated in equations (4.5) to (4.7) with respect to notations shown in Figure 4.4.

$$B_{\phi z} = 0 \tag{4.5}$$

$$B_{\phi x} = -B_\phi sin\phi \tag{4.6}$$

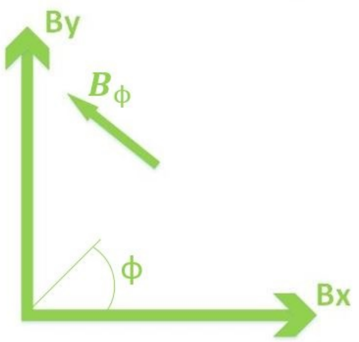


Figure 4.4 : B_ϕ Component of magnetic field calculated by IGRF and its components in ECEF.

$$B_{\phi y} = B_\phi cos\phi \tag{4.7}$$

B_θ is positive outwards, with respect to its direction its ECEF components are calculated in equations (4.8) to (4.11) with respect to notations shown in Figure 4.5.

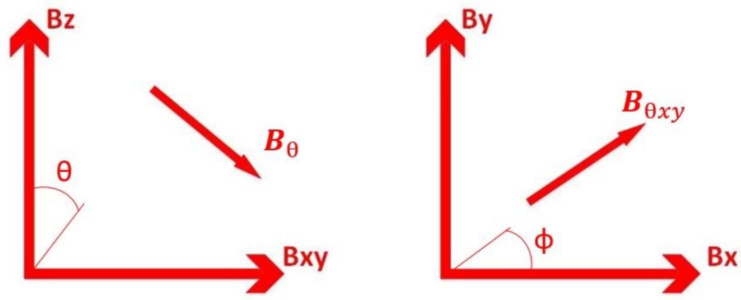


Figure 4.5 : B_θ Component of magnetic field calculated by IGRF and its components in ECEF.

$$B_{\theta z} = -B_\theta sin\theta \tag{4.8}$$

$$B_{\theta xy} = B_\theta cos\theta \tag{4.9}$$

$$B_{\theta x} = B_\theta cos\theta cos\phi \tag{4.10}$$

$$B_{\theta y} = B_\theta cos\theta sin\phi \tag{4.11}$$

Magnetic field components in ECEF are calculated by using equations (4.12) to (4.14). All components are used to find the magnitude of the magnetic field at the same point in order to compare the magnitude both in spherical and ECEF coordinates.

$$B_X = B_{rx} + B_{\theta x} + B_{\phi x} \quad (4.12)$$

$$B_Y = B_{ry} + B_{\theta y} + B_{\phi y} \quad (4.13)$$

$$B_Z = B_{rz} + B_{\theta z} + B_{\phi z} \quad (4.14)$$

$$B_{cartesian} = \sqrt{B_X{}^2 + B_Y{}^2 + B_Z{}^2} \quad (4.15)$$

When necessary transformations are made magnetic field values in caresian coordinates is shown in Figure 4.6. $B_{spherical}$ blue line cannot be seen its because it intersects at every point with $B_{cartesian}$. Those to lines are drawn in order to show that in both coordinate frames overall magnitude is same. This proves that the calculations are correct. Magnetic field components which are calculates in ECEF are going to be used in Helmholtz cage.

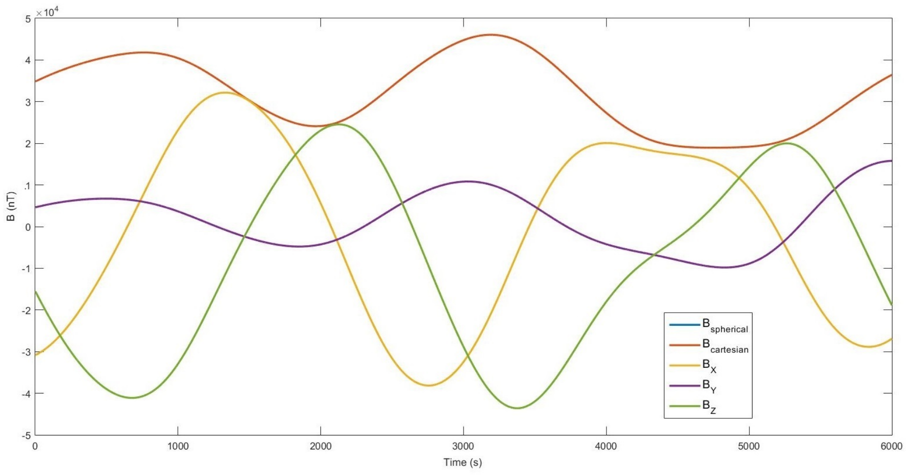


Figure 4.6 : Magnetic flux of TURKSAT-3USATs orbit in cartesian coordinates.

5. HELMHOLTZ CAGE

Helmholtz coils are a pair of circular or square coils on the same axis. With respect to their radius, number of turns and the flowing current value Helmholtz coils can generate a uniform central magnetic field. Helmholtz cage is 3 pair of Helmholtz coils in order to generate a 3 axis magnetic field to simulate the Earth's magnetic field. Helmholtz coils are capable of creating a magnetic fields with respect to number of turns and current applied to those turns. Uniform fields are generated between the coils with respect to their radius. Uniformity depends on the distance between two coils. Details about this will be given in coil design section.

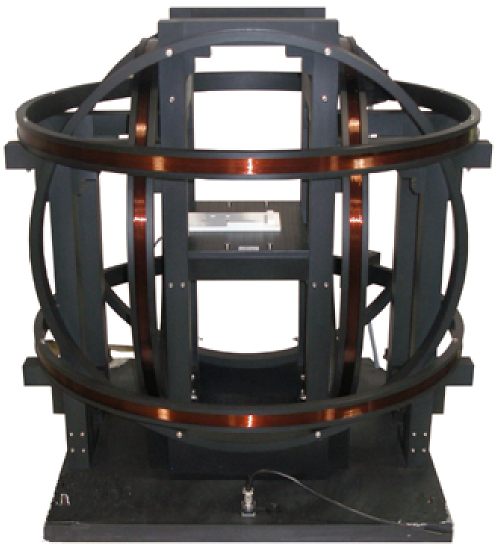


Figure 5.1 : A 3 axis, circular Helmholtz coil system [13].

5.1 Helmholtz Cage Elements

Helmholtz cage is based on 3 elements, software, power supply, coils. These elements are explained in the following section. Before starting a general form factor has been accepted with respect to cubesat trend(Figure 1.7). Design of the cage is made for 3U or smaller satellites. This is important because size of the satellite affect every element of the cage except its software.

5.2 Helmholtz Coil Design

In this section first mathematical model of Helmholtz coils will be explained. With respect to form factor and mathematical model simulations are carried.

5.2.1 Mathematical model of Helmholtz coil

Mathematical model of a Helmholtz coil is explained based on Figure 5.2, where the coordinates and variables for the calculations are shown [14]. The loop which is the

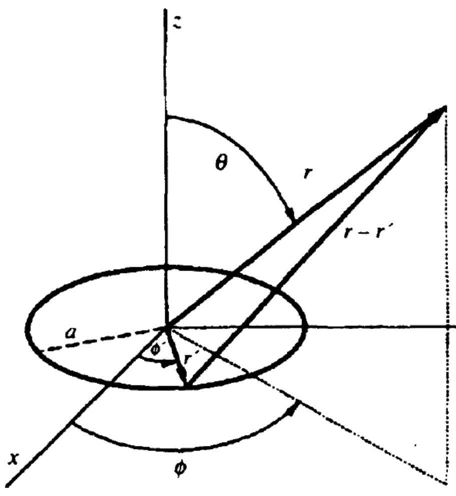


Figure 5.2 : Variables and coordinates for mathematical model [14].

one coil is placed on XY plane with a radius of ’a’. Current in the loop is ’I’. In equation (5.1) vector potential for a general current distribution is given where I is the current applied to loop and $\vec{A}$ is the vector potential.

$$\vec{A}(\vec{r}) = \frac{\mu_0}{4\pi}\int \frac{I\vec{dl}}{|\vec{r}-\vec{r}'|} \tag{5.1}$$

Due to the direction of the current; while $\vec{A}$ is dependent on only component ϕ, its symmetry is independent from ϕ. In order to simplify the calculation of the integral, ϕ has been taken as zero.

$$\vec{dl} = (-a\sin\phi', a\cos\phi', 0)d\phi' \tag{5.2}$$

$$\vec{r} = (r\sin\theta, 0, r\cos\theta) \tag{5.3}$$

$$\vec{r}' = (a\cos\phi', a\sin\phi', 0) \tag{5.4}$$

$$|\vec{r}-\vec{r}'| = \sqrt{r^2+a^2-2ra\sin\theta\cos\phi'} \tag{5.5}$$

By substituting the variables given in equations (5.2), (5.3), (5.4), (5.5) in to equation (5.1), $\sin\phi'$ disappears because its an odd function where $\cos\phi'$ enables the equation to be integrated because its an even function. By integrating equation (5.1), result will be equation (5.6).

$$A_\phi(r,\theta) = \frac{\mu_0 Ia}{2\pi}\int\limits_0^\pi \frac{\cos\phi' d\phi'}{\sqrt{r^2+a^2-2ar\sin\theta\cos\phi'}} \tag{5.6}$$

By transforming the expressions in to cylindrical coordinates using $r^2 = \rho^2 + z^2$ and $\sin\theta = \rho/\sqrt{\rho^2+z^2}$. Equation (5.6) will become;

$$A_\phi(\rho,z) = \frac{\mu_0 IA}{2\pi}\int\limits_0^\pi \frac{\cos\phi' d\phi'}{\sqrt{\rho^2+z^2+a^2-2a\rho\cos\phi'}} \tag{5.7}$$

Integral in equation (5.7) does not have a closed form. This integral can be transformed in to a tabulated function by changing the upper limit the integral to $\pi/2$. In order to make that change, $\phi' = \pi + 2\phi$ is used and placed in. As a result integral becomes;

$$A_\phi(\rho,z) = \frac{\mu_0 IA}{2\pi}\int\limits_0^{\frac{\pi}{2}} \frac{(2\sin^2\phi - 1)d\phi}{\sqrt{(a+\rho)^2+z^2-4a\rho\sin^2\phi}} \tag{5.8}$$

To simplify the equation (5.8) a definition can be made, which is $k^2 = \frac{4a\rho}{(a+\rho)^2+z^2}$. After applying the new definition the integral transformed into;

$$A_\phi(\rho,\phi) = \frac{\mu_0 I}{\pi k}\sqrt{\frac{a}{\rho}}\left[(1-\frac{1}{2}k^2)K(k) - E(k)\right] \tag{5.9}$$

where;

$$K(k) = \int_0^{-\frac{\pi}{2}} \frac{d\phi}{\sqrt{1-k^2\sin^2\phi}} \tag{5.10}$$

$$E(k) = \int_0^{\frac{\pi}{2}} \sqrt{1-k^2\sin^2\phi}\, d\phi \tag{5.11}$$

Both K and E are tabulated equations where also K is the complete elliptic integral of the first kind and E is of the second kind. From $\vec{B} = nab\vec{l}ax\vec{A}$ magnetic fields can be calculated which will result with 2 components;

$$B_\rho(\rho,z) = -\frac{\partial A_\phi}{\partial z} \tag{5.12}$$

$$B_z(\rho,z) = \frac{1}{\rho}\frac{\partial}{\partial\rho}(\rho A_\phi) \tag{5.13}$$

To calculate the derivatives of equations (5.12) and (5.13), derivatives of k and equations (5.10), (5.11) needs to be calculated. Because they are functions of (ρ,z). When they are calculated;

$$\frac{\partial K}{\partial k} = \frac{E}{k(1-k^2)} - \frac{K}{k} \tag{5.14}$$

$$\frac{\partial E}{\partial k} = \frac{E}{k} - \frac{K}{k} \tag{5.15}$$

$$\frac{\partial k}{\partial \rho} = \frac{k}{2\rho} - \frac{k^3}{4\rho} - \frac{k^3}{4a} \tag{5.16}$$

$$\frac{\partial k}{\partial z} = -\frac{zk^3}{4a\rho} \tag{5.17}$$

After applying these last four equations in to B_ρ and B_z, equations become;

$$B_\rho(\rho,z) = \frac{\mu_0 I}{2\pi}\frac{z}{\rho\sqrt{(\rho+a)^2+z^2}}\left[\frac{a^2+\rho^2+z^2}{(a-\rho)^2+z^2}E(k) - K(k)\right] \tag{5.18}$$

$$B_z(\rho,z) = \frac{\mu_0 I}{2\pi}\frac{1}{\sqrt{(\rho+a)^2+z^2}}\left[\frac{a^2-\rho^2-z^2}{(a-\rho)^2+z^2}E(k) - K(k)\right] \tag{5.19}$$

These equations enables us to calculate the magnetic field components at any location in z and ρ axis, for a single loop on XY plane. In order to find the solution for those equations first of all k needs to be defined, then k needs to be applied in the formulas. Since Helmholtz coils consists of two loop on the same axis, where one is on z=-d and the other is on z=+d. Total distance between them is 2d which can be related by the radius of the loops d=a/2. By using these expressions, total magnetic filed can be written as the sum of the each loop's magnetic field. Where;

$$B_\rho^{total}(\rho,z) = B_\rho(\rho,z+d) + B_\rho(\rho,z-d) \tag{5.20}$$

$$B_z^{total}(\rho,z) = B_z(\rho,z+d) + B_z(\rho,z-d) \tag{5.21}$$

These calculations are for a pair of single loops on same axis. For a Helmholtz cage, there should be 3 pair of loops on the axes which are perpendicular to each other. The same equations will be used for 3 axis with the same variables which have different names.

5.2.2 Simulation of suggested Helmholtz cage designs

Magnetic field can be achieved by either a circular or a square Helmholtz cage. In order to put a 3U satellite inside the system, there should be at least 30cm between the coils on the same axis. To fulfill this requirement coils with 30cm radius can be used. But in order to put the whole satellite inside the 30cmx30cmx30cm uniform magnetic field coils should have 60cm radius where the necessary volume for a better uniform magnetic field can be created. Difference between two variations will be done in the simulations. 1 axis cages are analyzed in order to understand and examine the magnetic field generated in between the coil pairs. With respect to 1 axis results, 3 axis versions of these cages are designed and analyzed. In 1 axis version only the magnitude of the magnetic field can be generated while in 3 axis every component of the magnetic field in ECEF can be generated. Calculation method of these values are given in section 4. Square cage design has slightly more uniform magnetic field, which can be seen in Figure(5.3). Where a is the radius of the circular coil and half length of the square coil and z is the distance from the middle point of the pair of coils. Cages are described with their smallest radius because every pair of coil is slightly bigger than the first one in order house every pair inside one another.

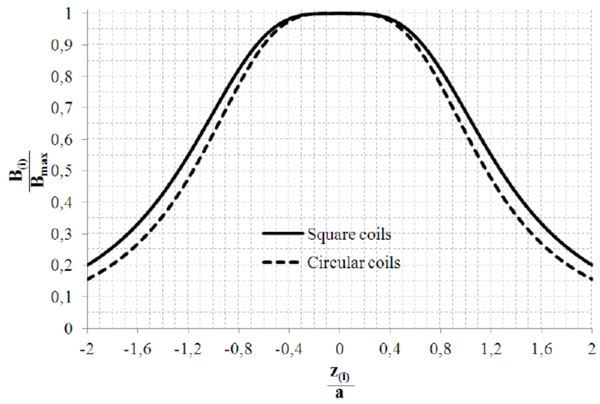


Figure 5.3 : Magnetic flux uniformity difference [15].

5.2.2.1 Circular Helmholtz cage

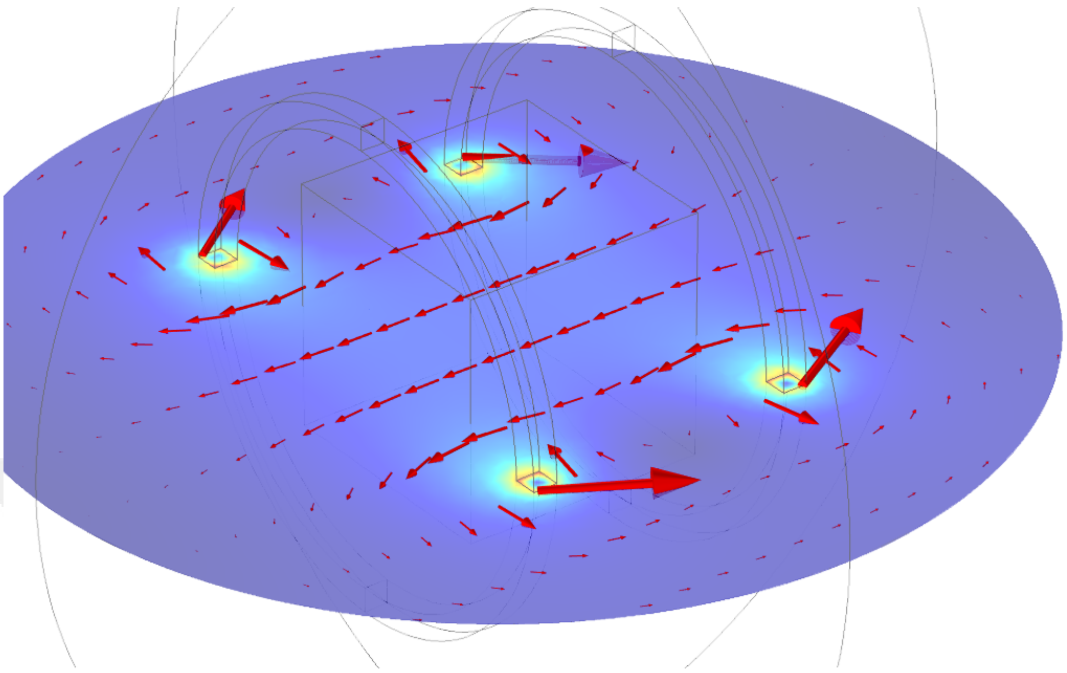
In this section two different sizes of circular Helmholtz cages are simulated. Circular cages with smallest radius 30cm and 60cm are simulated. Simulation results are compared in order to show which one of them has the more uniform magnetic field. Calculations are made with respect the same amount of current and turns. This section is focused on uniformity not the exact value of magnetic field but its difference from one side to another in effective volume. In Figure 5.4(a) it can be seen that in the effective volume area near to its edges magnetic field directions starts to differentiate and uniformity decreases. In Figure 5.4(b) since the volume itself is deeper inside the space between coils, it has more uniform magnetic field direction. Figure 5.5 shows detailed magnetic field directions inside the volume for a one axis Helmholtz coil. Again close to edges and corners uniformity is disturbed(Figure 5.5(a)).

Magnetic field density is shown in Figure 5.6(a) and (b). With the same amount of magnetic field in the center of the measured volume it can be seen that whole volume is more uniform in Figure 5.6(b). Simulated 3 axis system has slightly better results in creating uniform magnetic field. When 3 perpendicular planes compared, the change in magnetic field in Figure 5.8(b) is not so sudden and more uniform than Figure 5.8(a)

5.2.2.2 Square Helmholtz cage

In this section two different sizes of square Helmholtz cages are simulated. Square cages with smallest edge 60cm and 120cm are simulated. These 60 and 120cm values are same with the circular one when their diameter is considered. Their results are compared in order to show that uniform magnetic field is better on the bigger one. Calculations are made with respect the same amount of current and turns. This section is focused on uniformity not the exact value of magnetic field but its difference from one side to another in effective volume. In Figure 5.9(a) it can be seen that in the effective volume area near to its edges magnetic field directions starts to differentiate and uniformity decreases. In Figure 5.9(b) since the volume itself is more deep inside the space between coils, it has more uniform magnetic field direction. Figure 5.10 shows detailed magnetic field directions inside the volume for a one axis Helmholtz coil. Again close to edges and corners uniformity is disturbed(Figure 5.10(a)).

Magnetic field density is shown in Figure 5.6(a) and (b). With the same amount of magnetic field in the center of the measured volume it can be seen that whole volume is more uniform in Figure 5.6(b). Simulated 3 axis system has slightly better results in creating uniform magnetic field. When 3 perpendicular planes compared, the change in magnetic field in Figure 5.13(b) is not so sudden and more uniform than Figure 5.13(a).


(a) 60cm diameter 1 axis

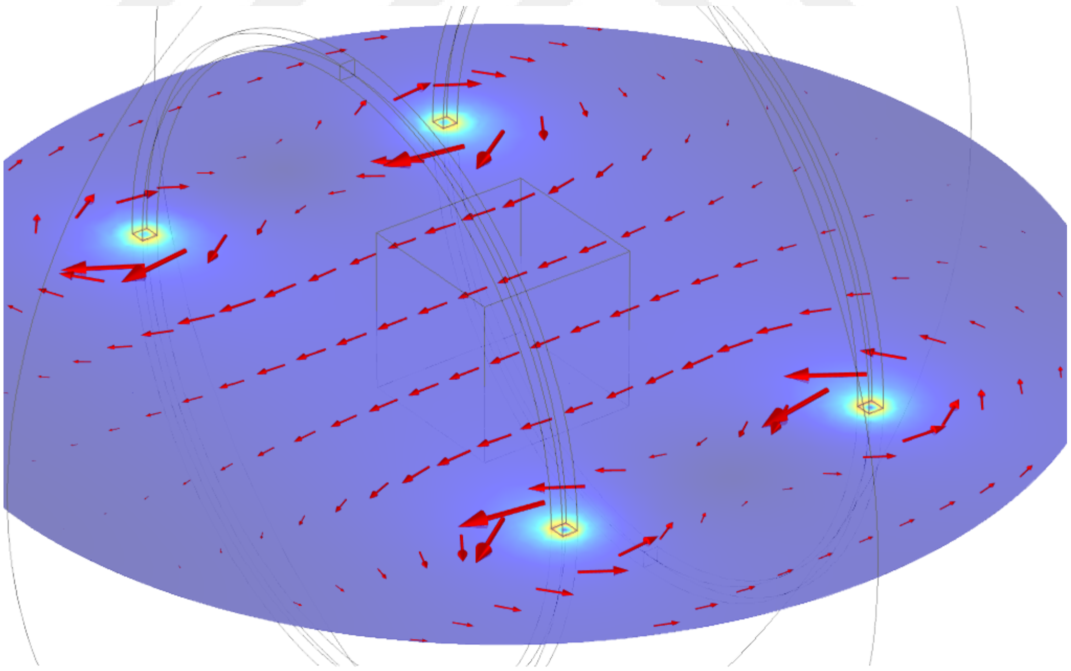
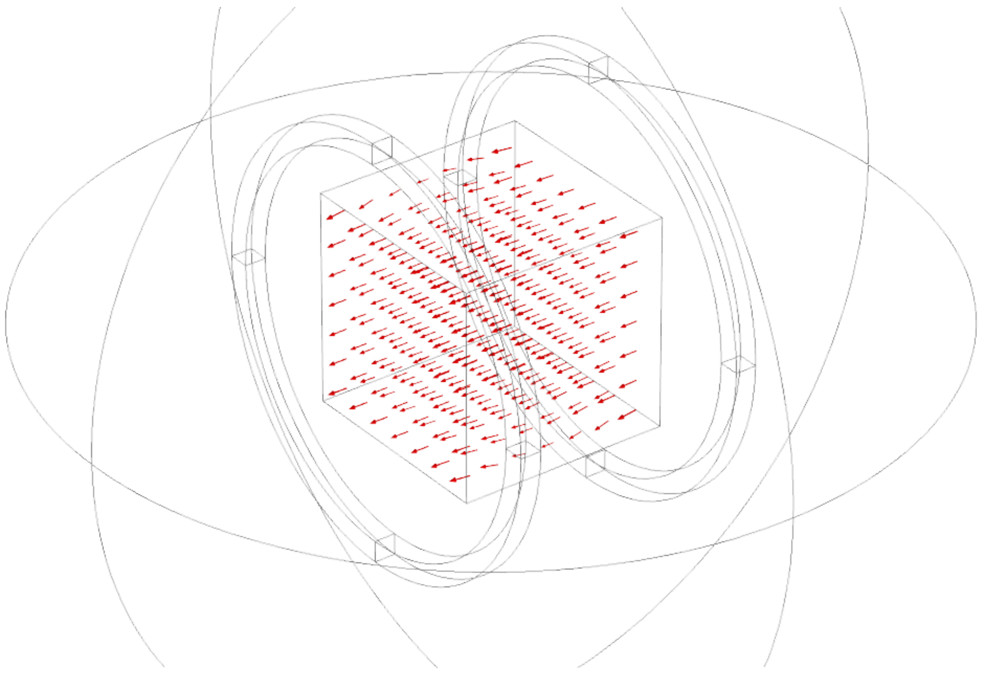
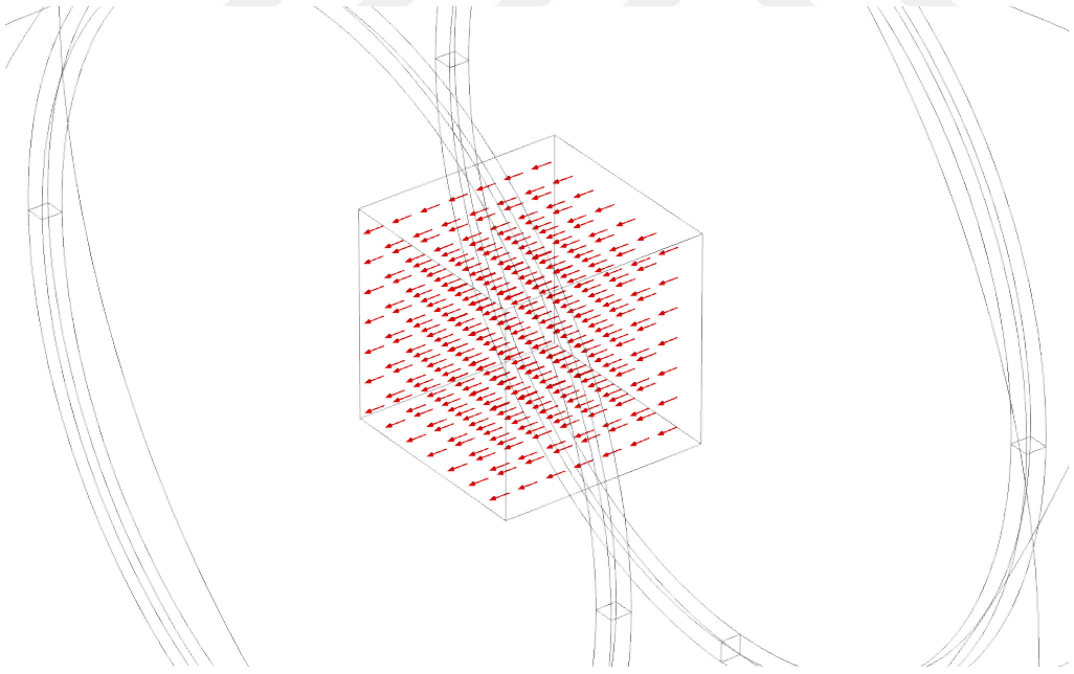

(b) 120cm diameter 1 axis

Figure 5.4 : Magnetic field directions on the plane parallel to centrel axis for 1 axis circular cage.

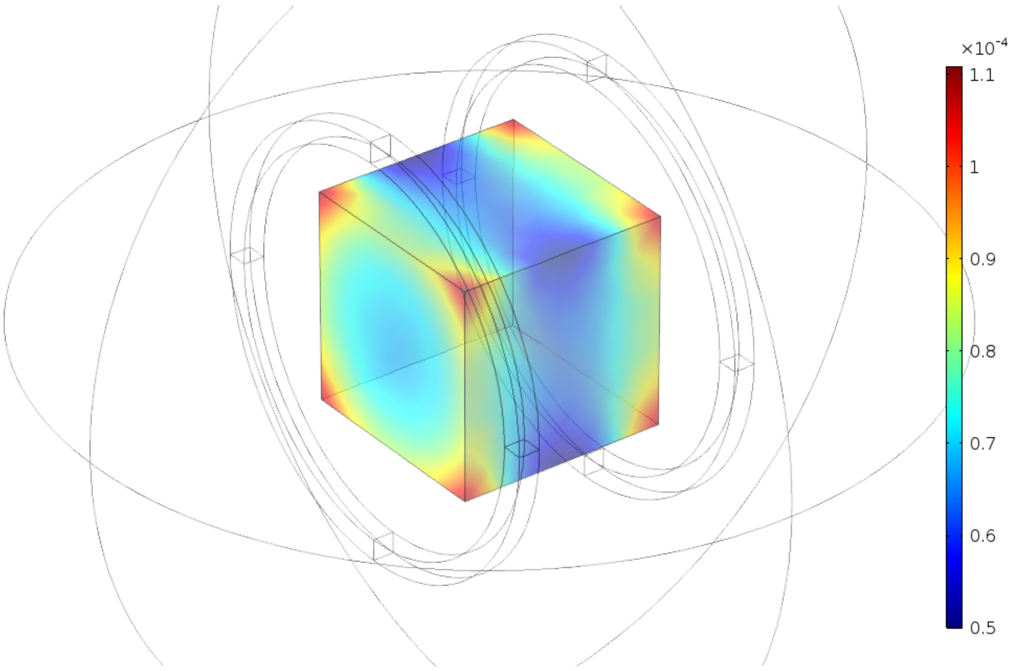


(a) 60cm diameter 1 axis

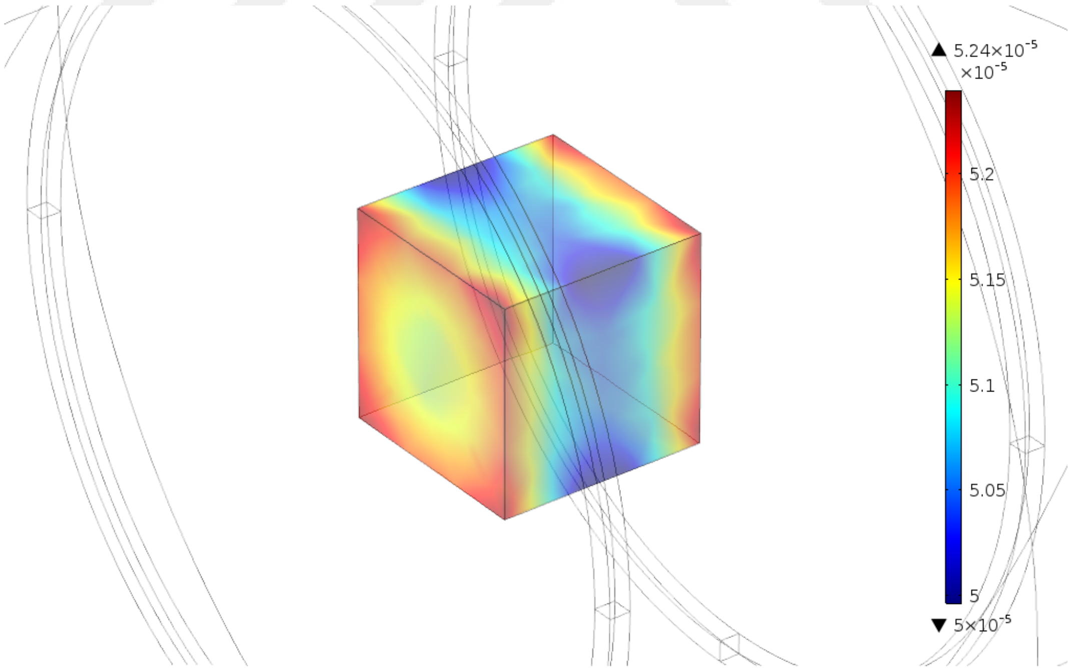


(b) 120cm diameter 1 axis

Figure 5.5 : Magnetic field directions inside of effective volume for 1 axis circular cage.

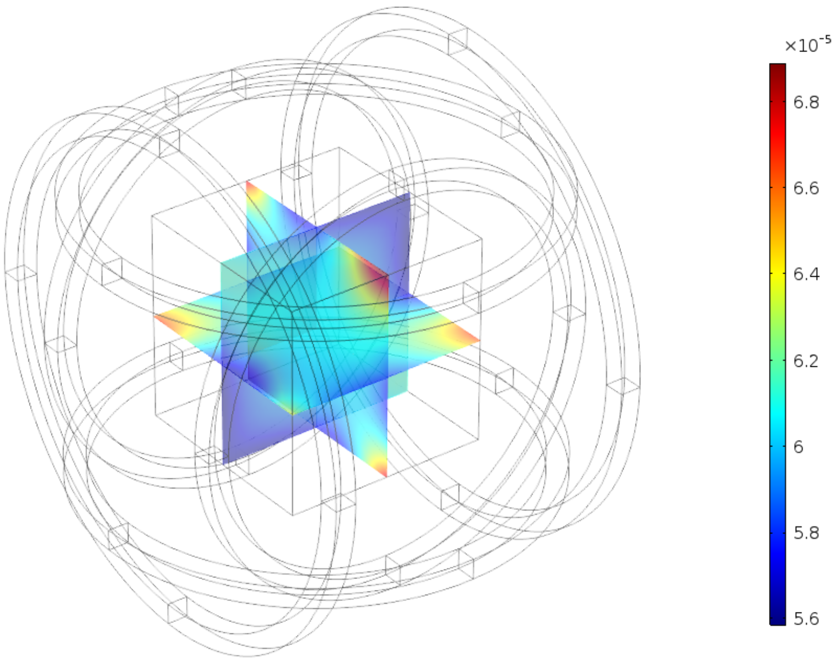


(a) 60cm diameter 1 axis

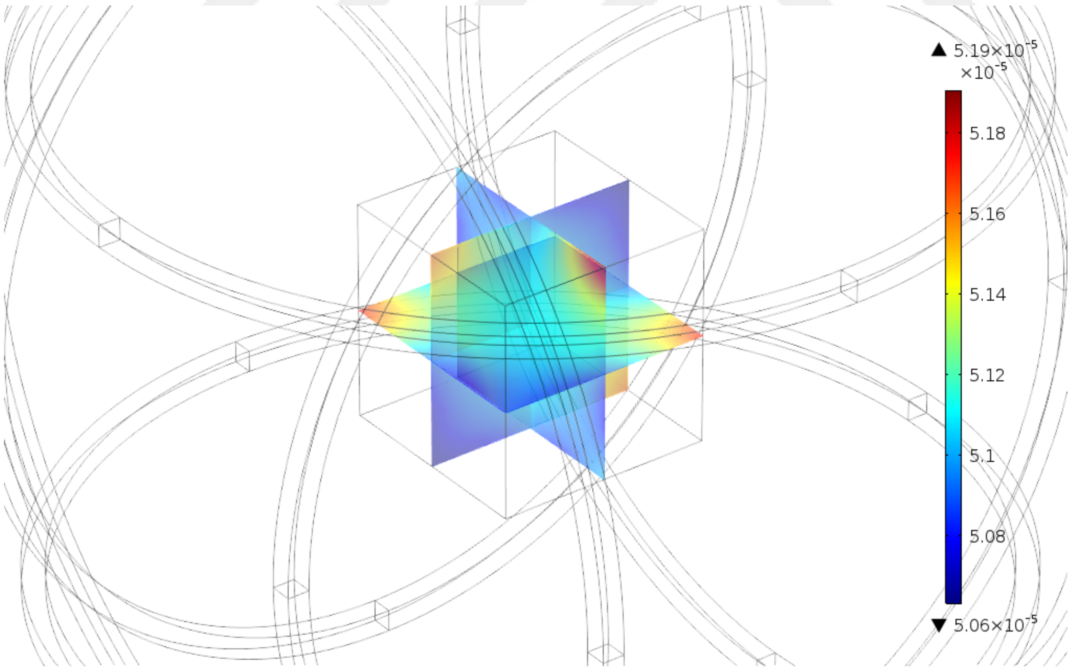


(b) 120cm diameter 1 axis

Figure 5.6 : Magnetic field density inside of effective volume for 1 axis circular cage.

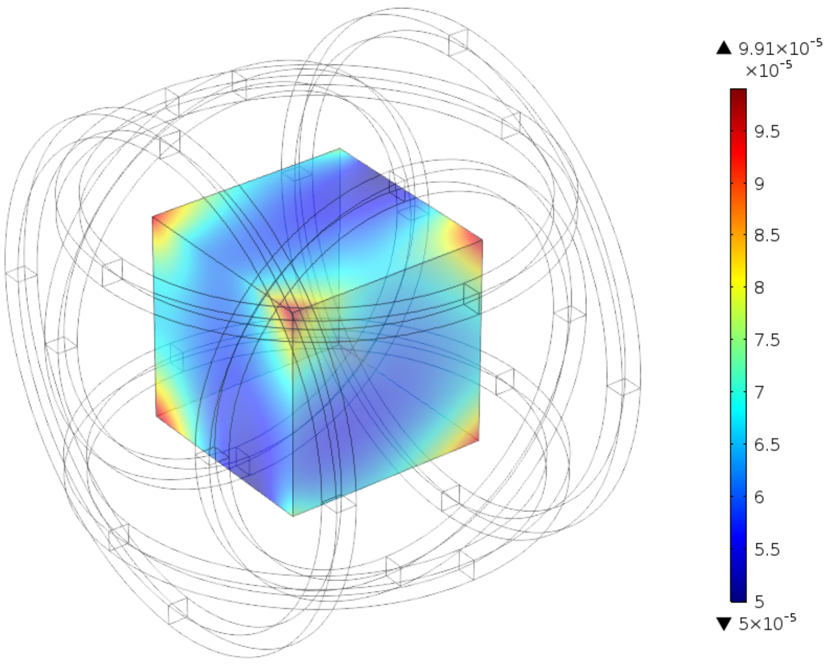


(a) 60cm diameter 3 axis

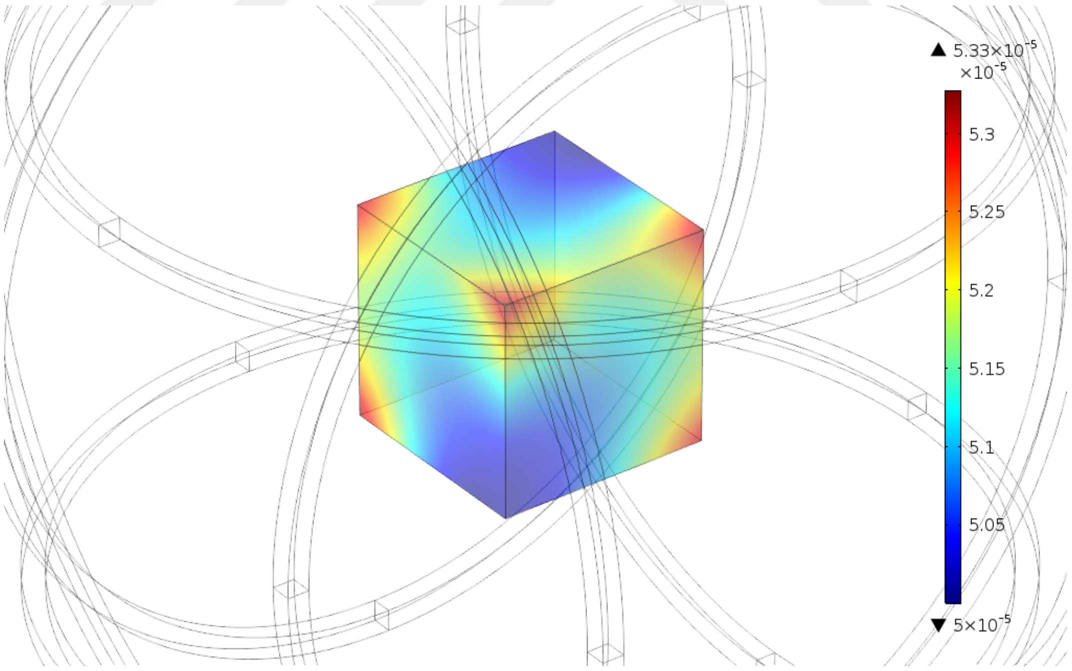


(b) 120cm diameter 3 axis

Figure 5.7 : Magnetic field density on XY, YZ, ZX planes for 3 axis circular cage.

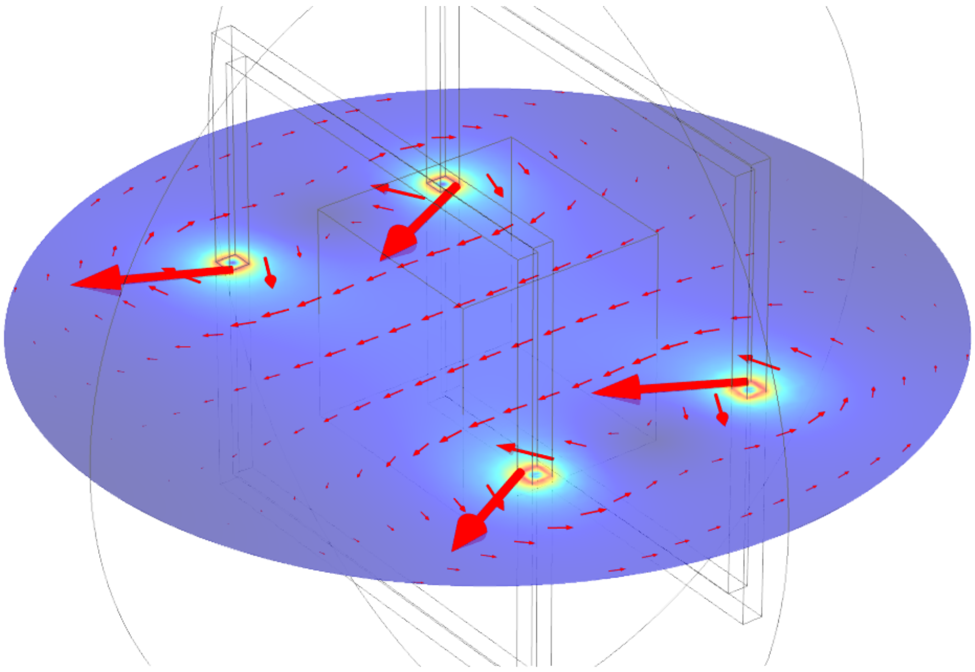


(a) 60cm diameter 3 axis



(b) 120cm diameter 3 axis

Figure 5.8 : Magnetic field density inside of effective volume for 3 axis circular cage.


(a) 60cm edge 1 axis

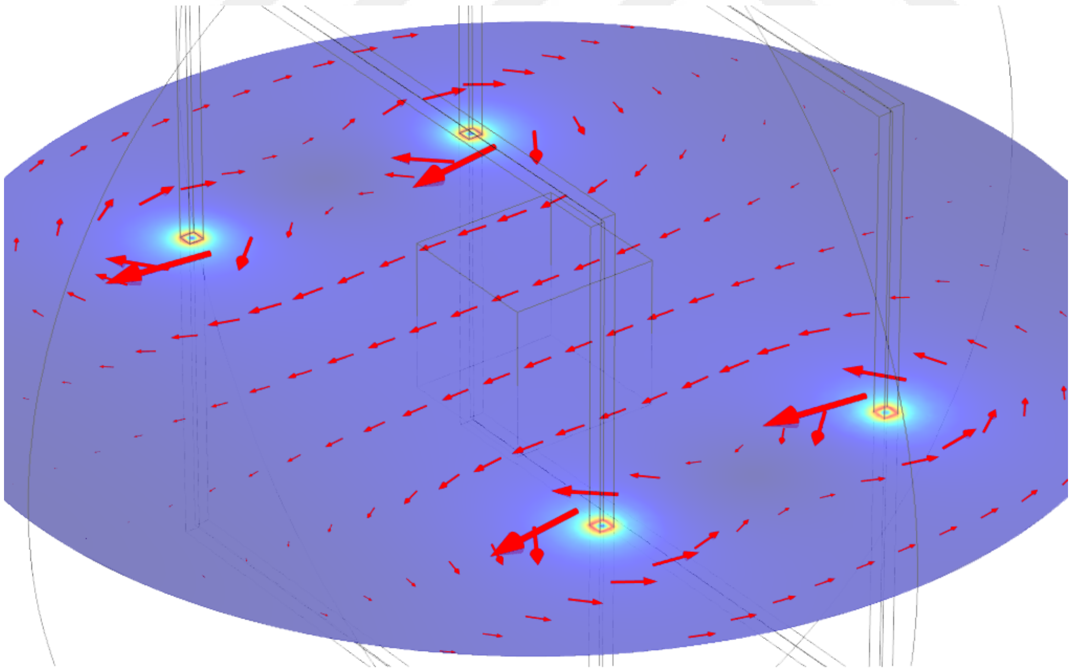
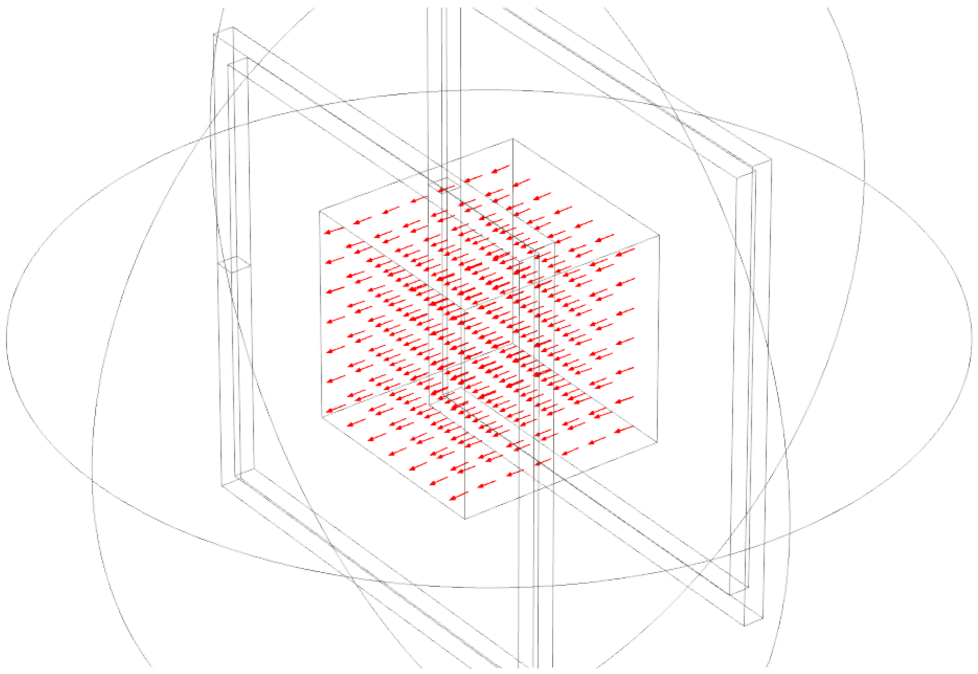

(b) 120cm edge 1 axis

Figure 5.9 : Magnetic field directions on the plane parallel to centrel axis for 1 axis square cage.


(a) 60cm edge 1 axis

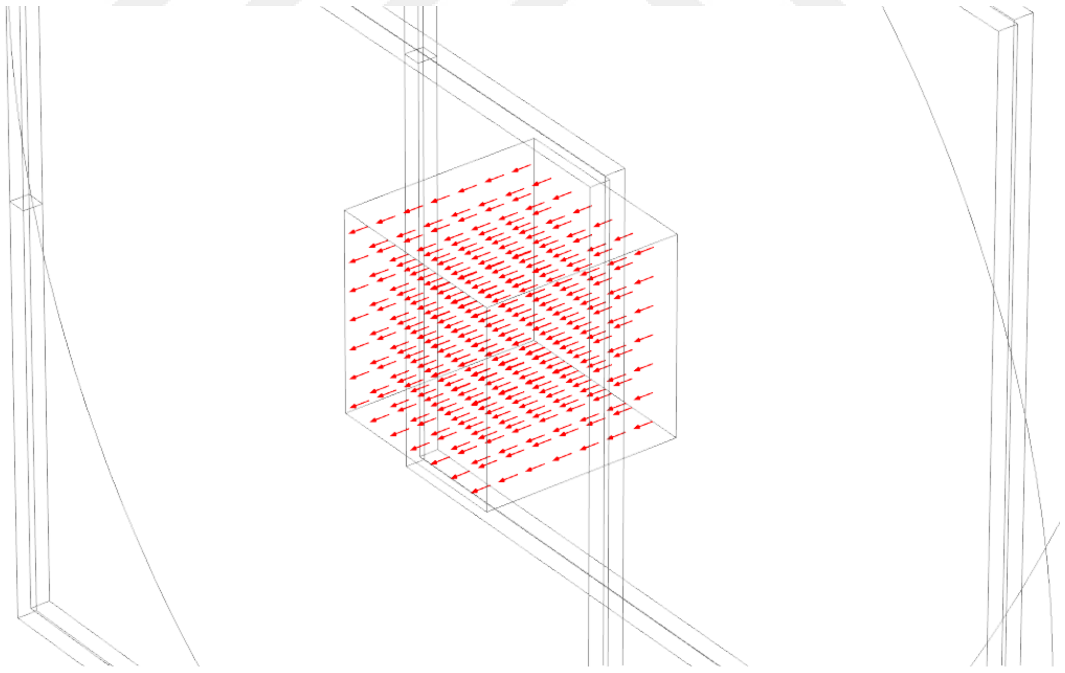
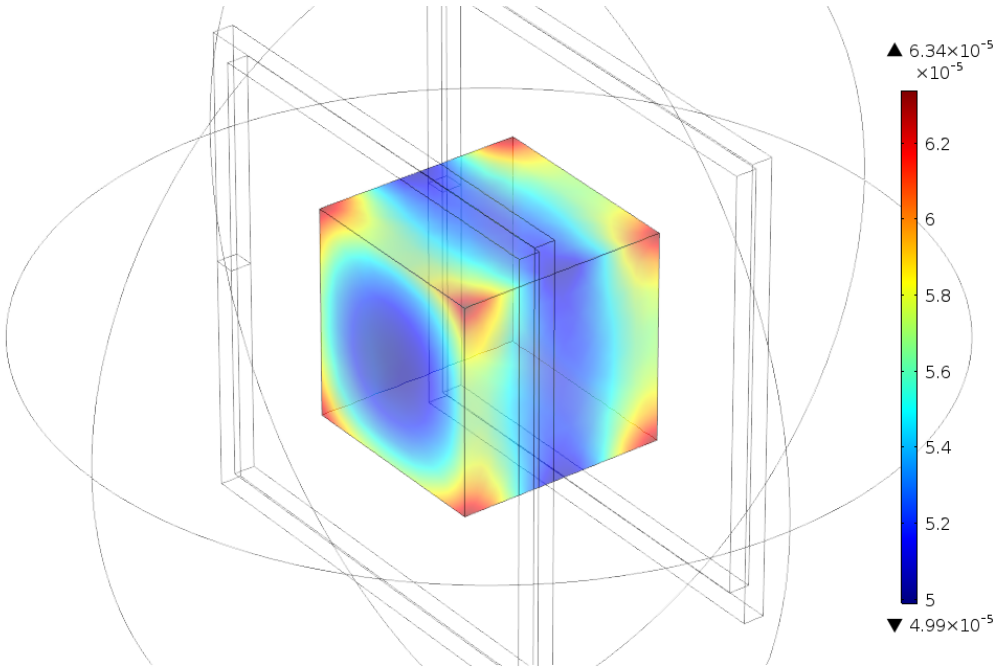
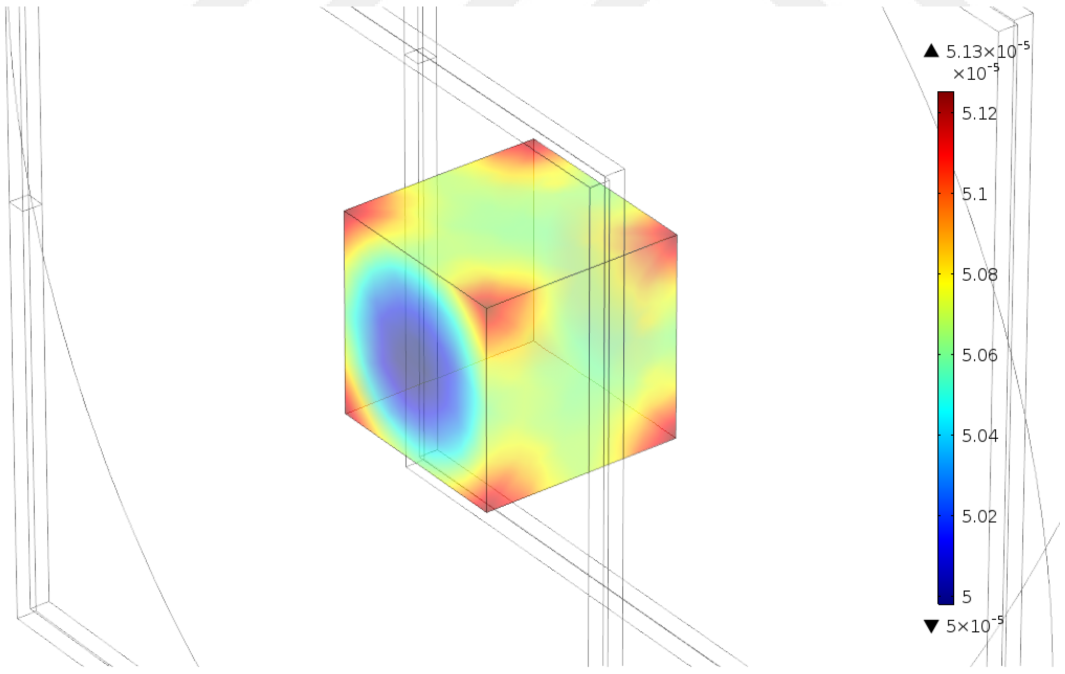

(b) 120cm edge 1 axis

Figure 5.10 : Magnetic field directions inside of effective volume for 1 axis square cage.

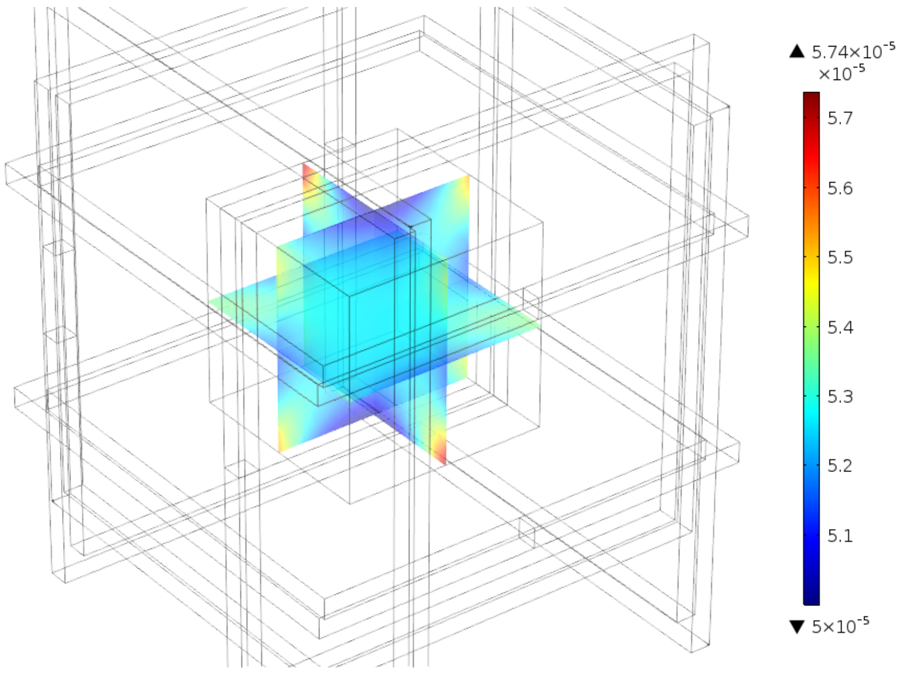


(a) 60cm edge 1 axis

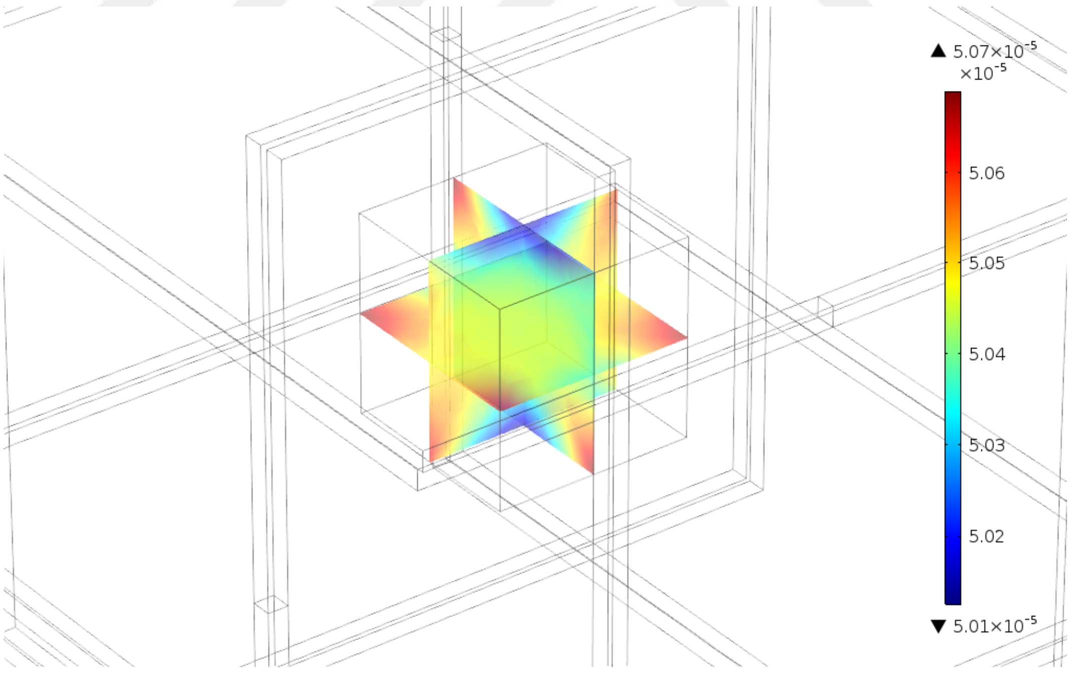


(b) 120cm edge 1 axis

Figure 5.11 : Magnetic field density inside of effective volume for 1 axis square cage.

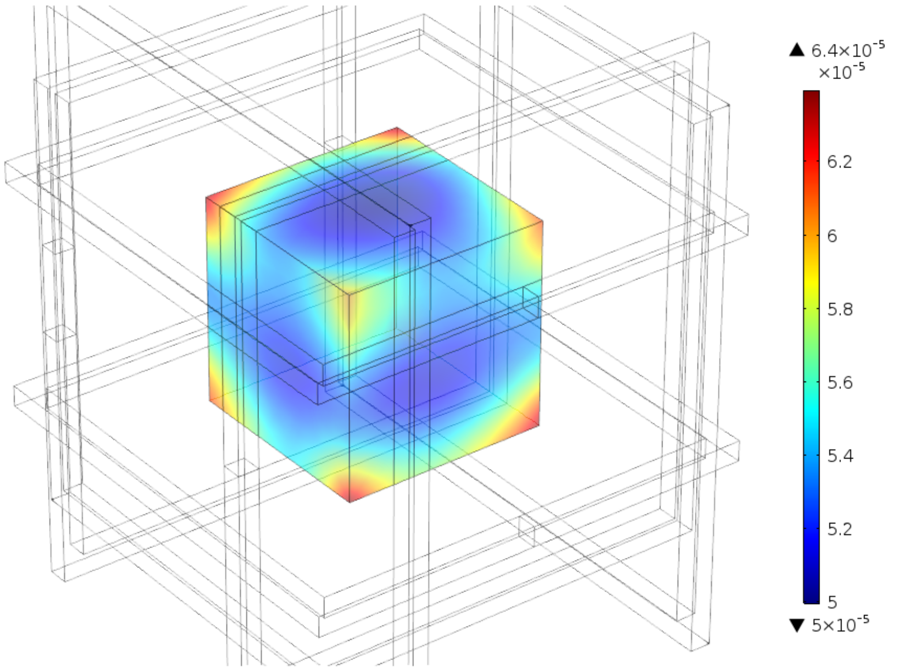


(a) 60cm edge 3 axis

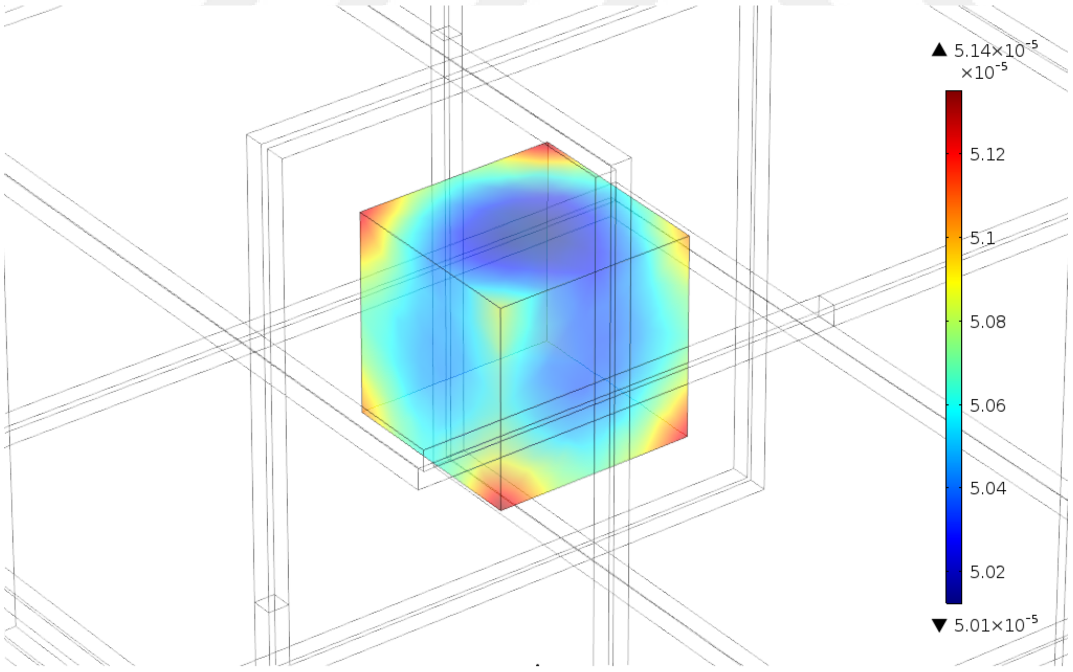


(b) 120cm edge 3 axis

Figure 5.12 : Magnetic field density on XY, YZ, ZX planes for 3 axis square cage.



(a) 60cm edge 3 axis



(b) 120cm edge 3 axis

Figure 5.13 : Magnetic field density inside of effective volume for 3 axis square cage.

5.2.2.3 Comparison of square and circular Helmholtz cages

In the previous sections it has been shown that 60cm (half edge&radius) is better than 30cm (half edge&radius). In order to decide for the proper geometry 60cm versions of both designs are being compared. In Figure 5.14 it can be seen that close to edges and corners of the 30cmx30cmx30cm envelope in (a) there are slight changes in directions while in (b) directions of the magnetic field are more uniform.

Due to better results and simplicity in their design and manufacturing, square cage is chosen to be build in laboratory.

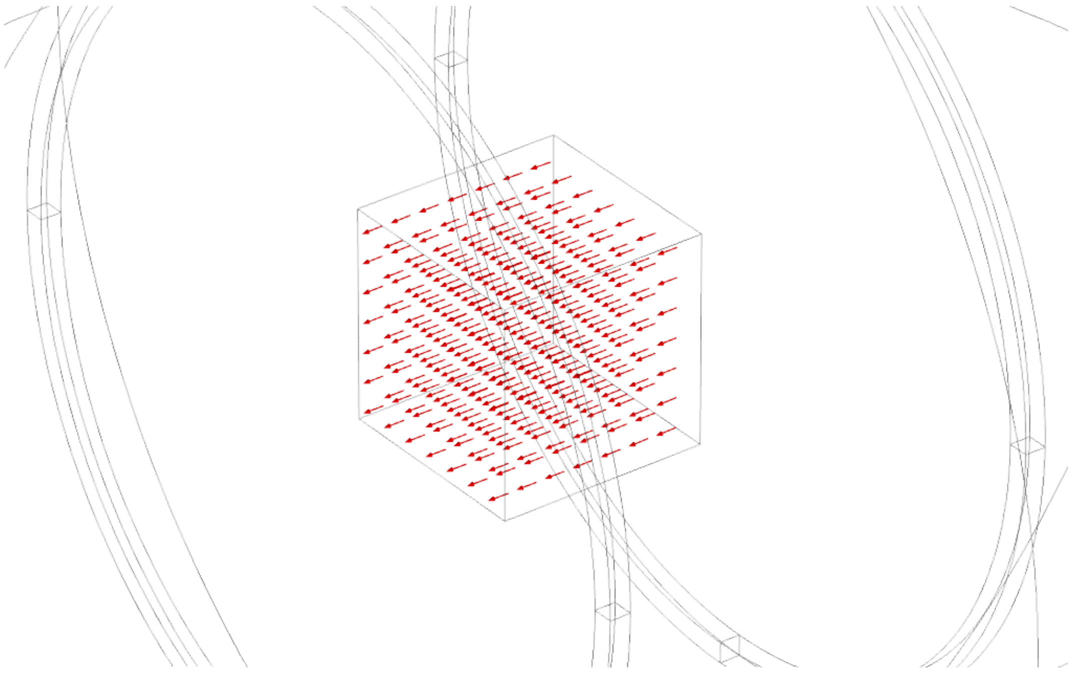
5.3 Proof of Concept for Table Top Model Cage Design

Due to lack of time and funding smaller version of the square cage is designed. After the design phase of the small square cage. First proposed square cage was going to have a 20cm edge(10cm half edge) length. Which will create an envelope of 5cmx5cmx5cm for magnetic field. By placing a magnetometer, internal field can be measured and systems control could be made.Since the given circular cage has the 20cm diameter it is also suitable for proof of concept and to check if the power supplies and the software can work as desired. As mentioned small test setup has 20cm diameter(Figure 5.17). It has capability of creating a 4x10^{-3}T by applying 4A to coils [16]. Each coil has number of turns of 126. Simulations are made by using number of turns and their diameter to check if system is capable of generating 4x10^{-3}T in the center. In Figure 5.18 and Figure 5.19, magnetic field directions are uniform inside the envelope.

In Figure 5.20 on the legend it can be seen that maximum flux is 3.9x10^{-3}T as mentioned in its manuel. Magnetic flux from one coil to another can be seen in Figure 5.21. In the graph its again clear that system is able to give nearly 4x10^{-3}T.

5.4 Power Supply Design

In order to create a fully controllable test setup, power supply units are essential. First a fly-back converter, than 3 buck converters are implemented in to the system to supply the required power directly from grid.(Figure 5.22). But for the table top model only a buck converter which will be powered from an external power supply is enough. Problem with the Helmholtz coils is that they are only consist of coppers wires with no


(a) 120cm diameter 1 axis

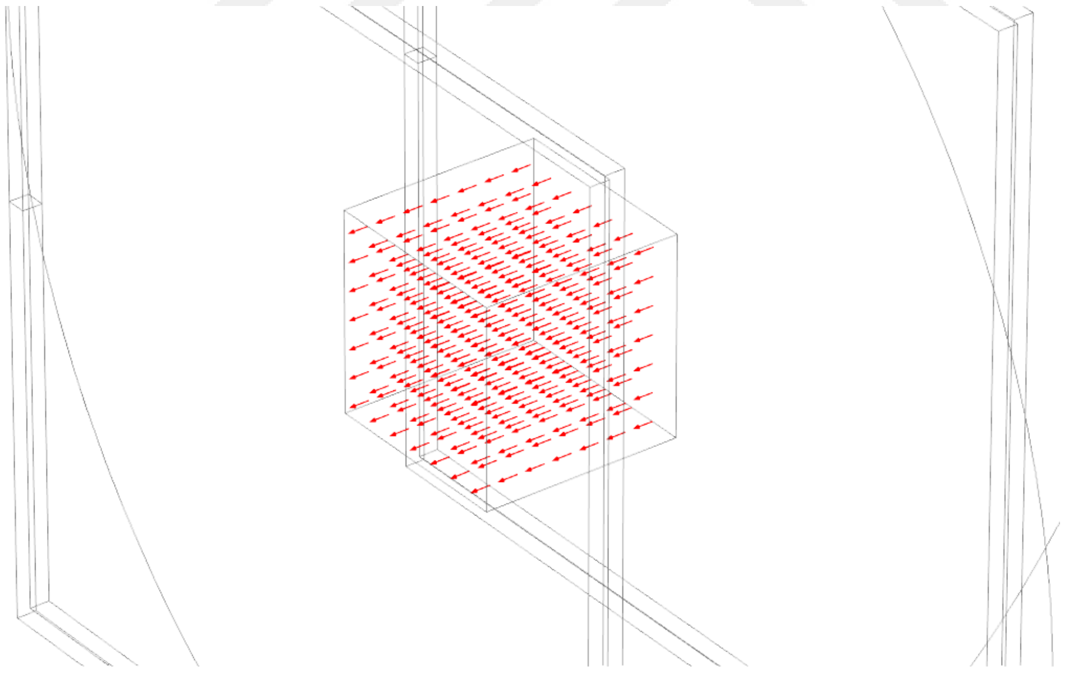
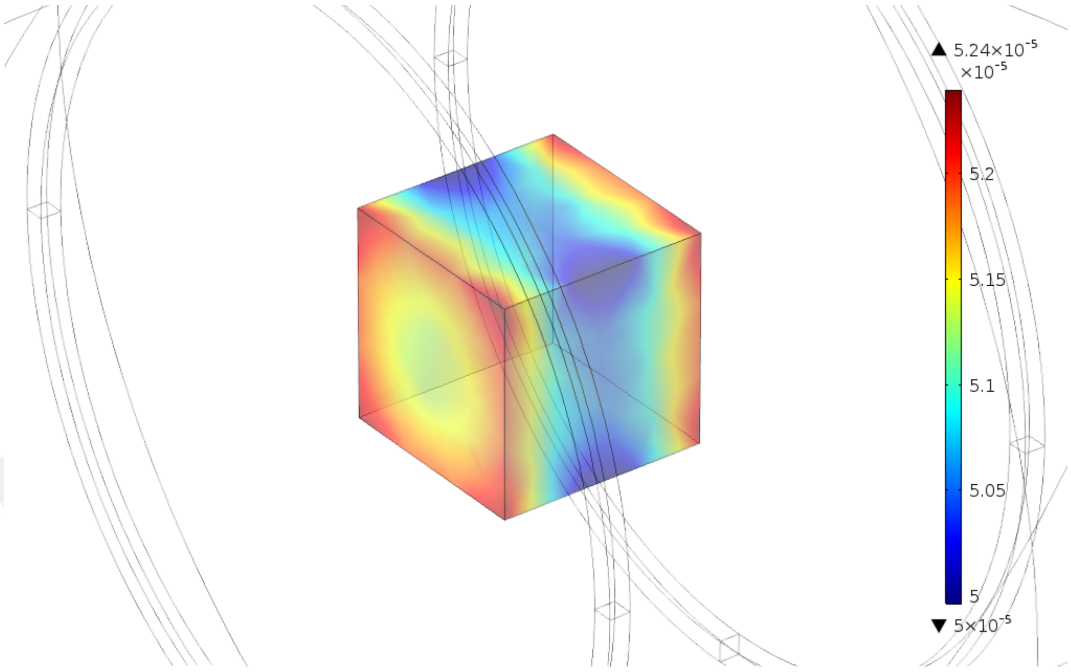
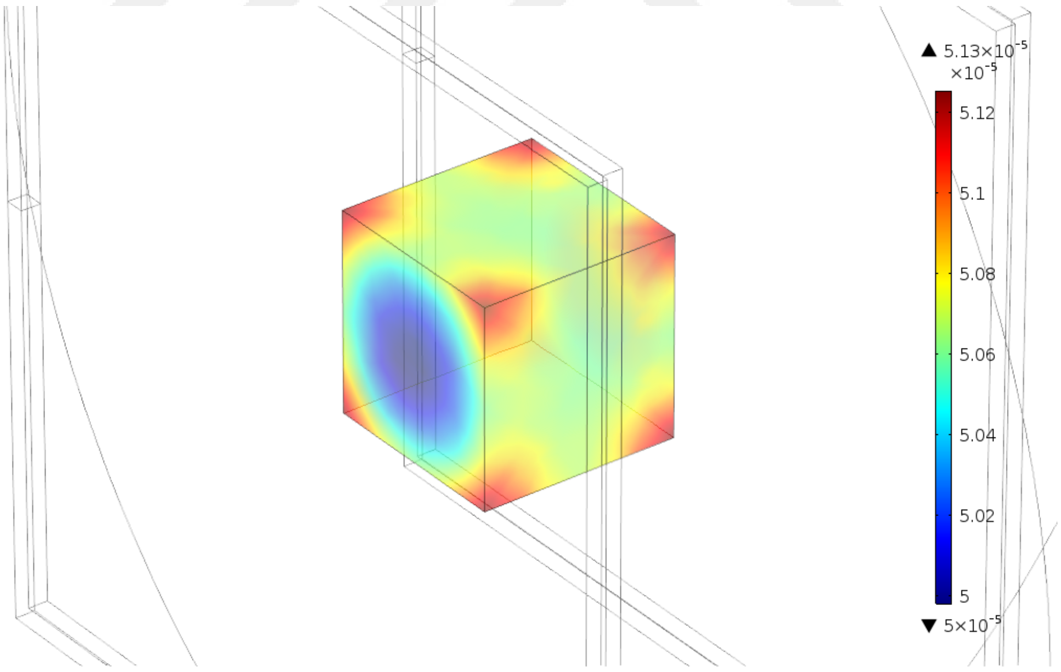

(b) 120cm edge 1 axis

Figure 5.14 : Magnetic field directions inside of effective volume for 1 axis coil geomteries.

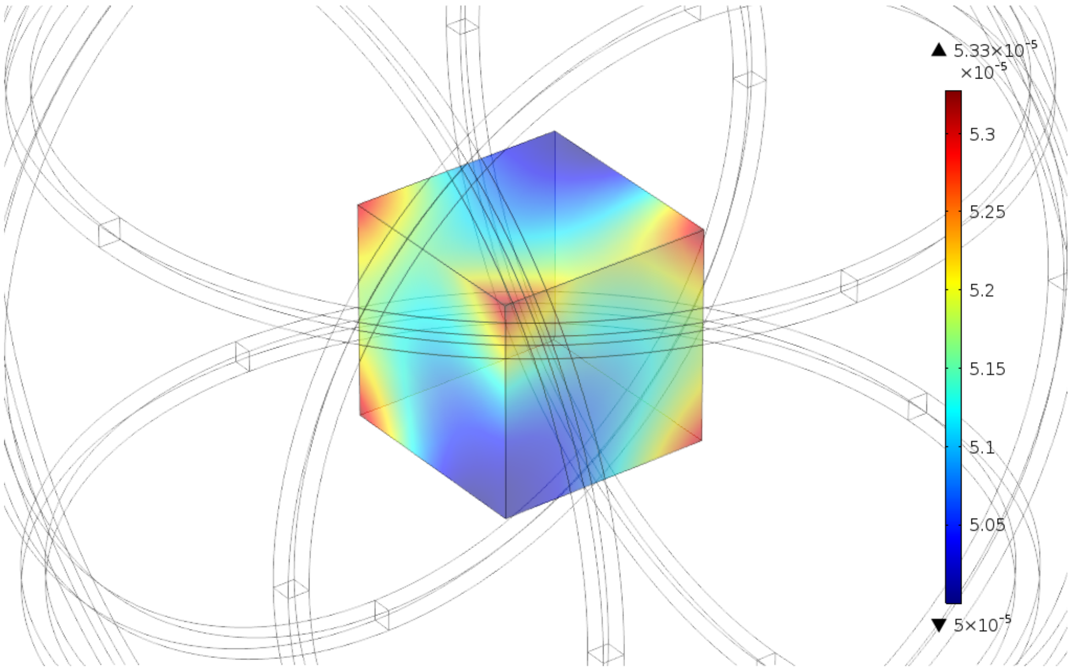


(a) 120cm diameter 1 axis

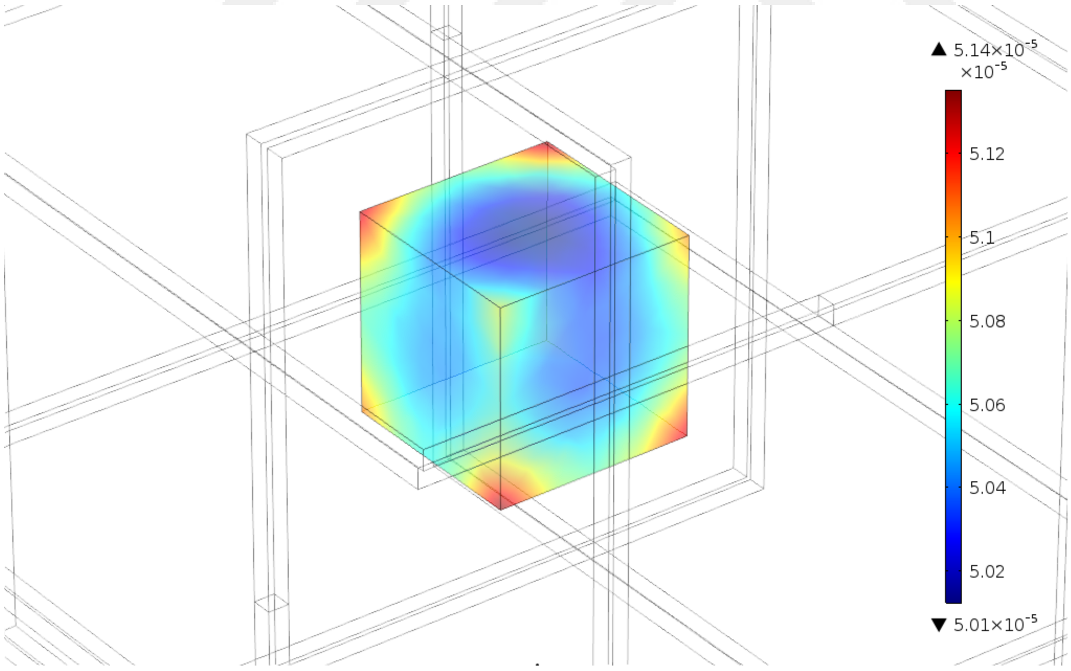


(b) 120cm edge 1 axis

Figure 5.15 : Magnetic field density inside of effective volume for 1 axis coil geomteries.



(a) 120cm diameter 3 axis



(b) 120cm edge 3 axis

Figure 5.16 : Magnetic field density inside of effective volume for 3 axis coil Geomteries.

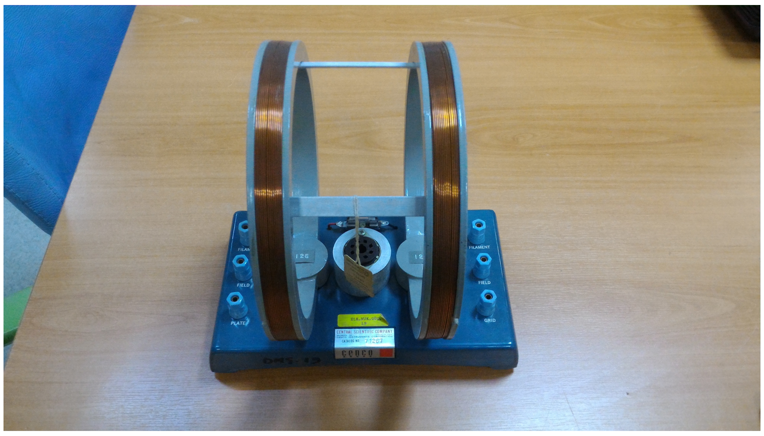
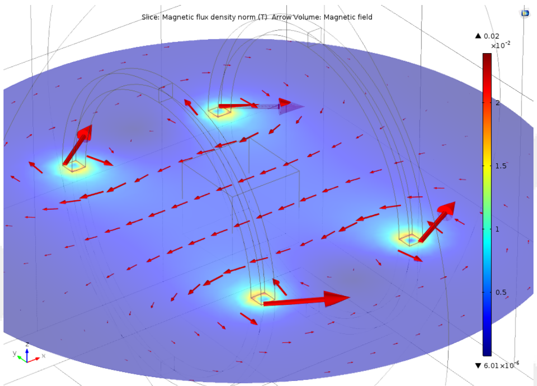


Figure 5.17 : 20cm diameter circular Helmholtz coil.


Slice: Magnetic flux density norm (T) Arrow Volume: Magnetic field
▲ 0.02
×10⁻²
2
1.5
1
0.5
▼ 6.01×10⁻⁶
Figure 5.18 : Planar view of magnetic field direction of Cenco coil.

additional electrically resistive material in the power flows path. In other words they will be acting like a short circuit. But because of needed length for copper wire in coils are long enough that their material property causes enough resistivity. Both ways buck converters can be designed but this makes the system safer. In a 3 axis cage there are 6 coils, but only 3 buck converters will be used to control every axis (pair of coils). One of the reasons is to increase the resistance. But the main reason is to decrease the complexity and overcome some timing problems. Even if the same elements and products are going to be used, supplying a pair of coil from different converters may cause delays and it will be harder to control. But when using only 1 converter for 1 pair of coils, since the current needed to be same on both of the coils of a pair, its certain that both currents are the same. Voltage drops are negligible because magnetic field is generated by the current. This is because of the cur In Figure 5.22 values of the elements are not the final values. They are to show the whole systems itself. In order to calculate the necessary values for the elements copper wire diameter, length and MCU should be chosen first. But because of lack of funding and time and also with the new cage calculations are going to be made in a simpler way with respect to small coil.

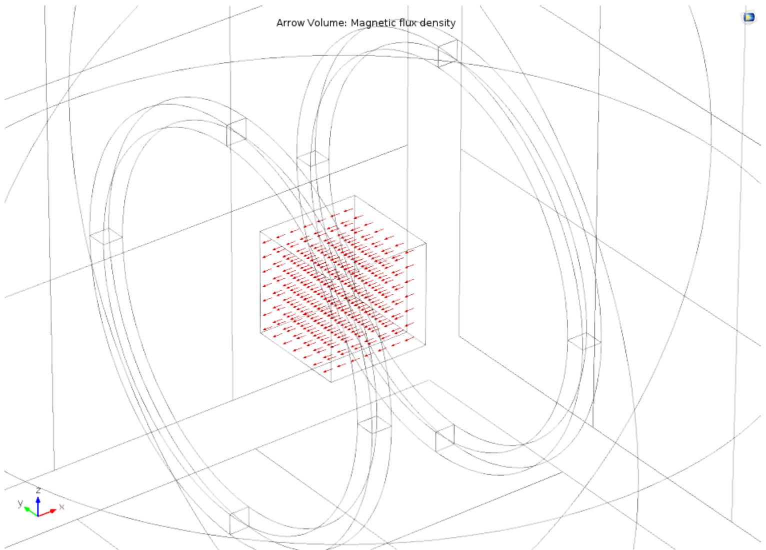


Figure 5.19 : Magnetic field directions in 5cmx5cmx5cm envelope.

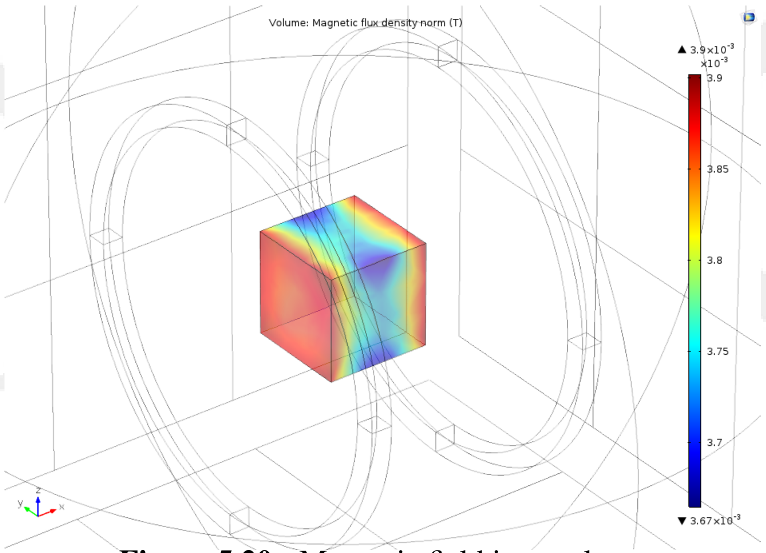


Figure 5.20 : Magnetic field in envelope.

5.4.1 Sizing of power supply

First step is to calculate the resistance of the coil pair and needed feasible voltage to be applied on coils. Necessary maximum magnetic field can be generated by 100 turncurrent value. Other known and predetermined values are; radius of coil, conductivity of conductor, resistivity of conductor, density of conductor, conductor diameter, selling weight, selling length and type of coil. These calculations are made by using Appendix A.2. Calculation of the converter is made by using Appendix A.3. Before using the codes in appendix, number of turns and required minimum and maximum currents found by using COMSOL. Cenco coil does not require such a big system since it has only one axis. A buck converter is enough to supply the system. In

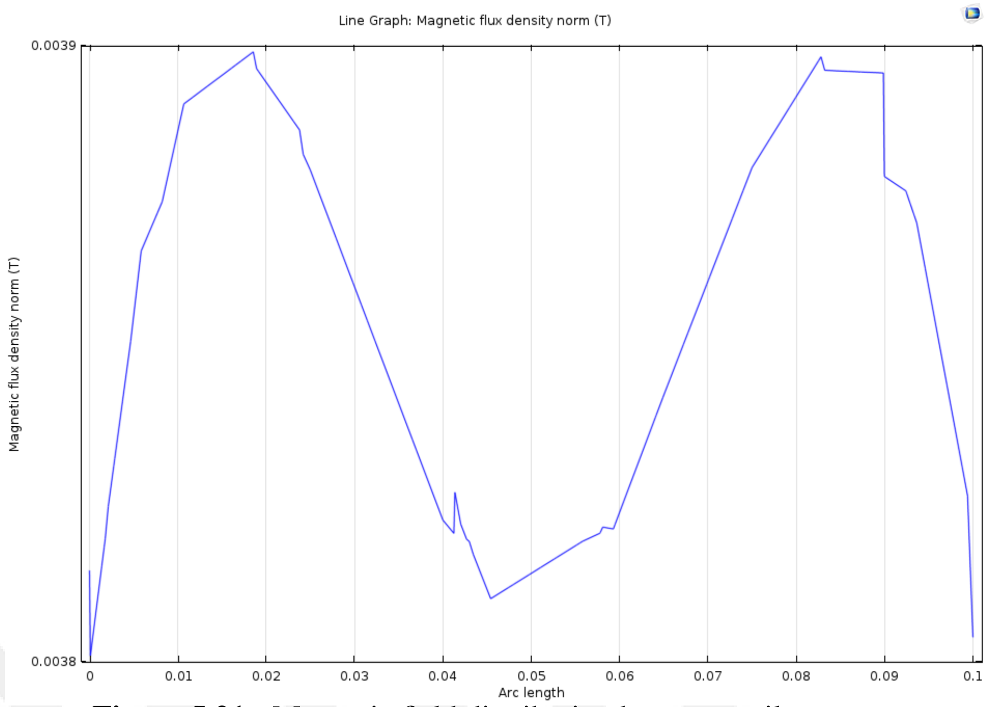


Figure 5.21 : Magnetic field distribution between coils centers.

order to calculate the converters resistance of the coils should be calculated and again an MCU should be choosen in order to calculate manageable PWM steps. This is important because by using the PWM, output voltage of the converters are controlled. In order to generate the magnetic field there should be a predetermined maximum and minimum values. With respect to maximum and minimum values of the magnetic field different PWM will be applied to controller. However while changing between these values, step size of the PWM is defined from the switching frequency of the converter and the MCU's working frequency. Step size is calculated by $\frac{SwitchingFrequency}{MCUFrequency}$ and cannot be higher then this value. Since coils are controlled by the current and current flows with respect to output voltage of the converters. Magnetic field can be change with respect to same step size for every desired value. This means that if more accurate and precise magnetic values are needed than the step size should be smaller. This causes another issue for power supply sizing. Since MCU frequency comes by the way its bought, only switching frequency can be changed. When changing the value of the switching frequency elements of the converter such as inductor, capacitor and switching element will get bigger. It turns into an optimization problem. But since there is no lack of space for the test setup, switching frequency can be as low as possible.

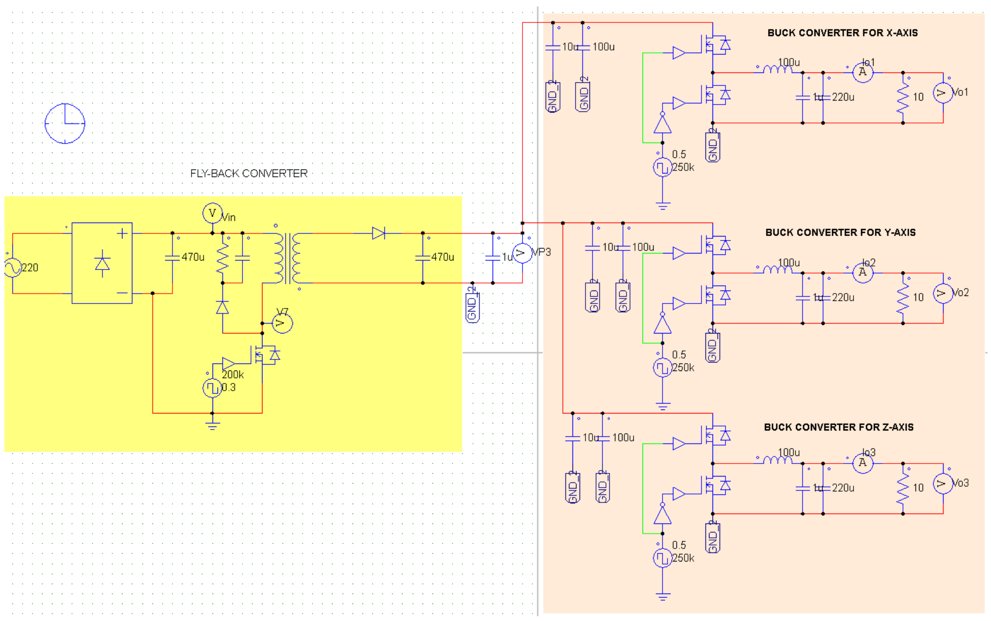


Figure 5.22 : Electrical schematic of test setups power supplies for 3 axis cage.

5.4.2 Buck converter calculations of cable top cage setup

By applying the values given in Cenco's manual [16] in to Appendix A.2 and Appendix A.3 for 4A maximum current to coils, necessary maximum output voltage is 6.85 V. Planned MCU is STM32F746NG which has a working frequency of 216 MHz. Switching frequency is chosen as 250 kHz. Which will allow system to change the voltage thus the current thus magnetic field can be change with the precision and step size of 0.11%. If 100 kHz was chosen step size would be nearly 0.4%. Converter calculations are therefore made by using 4A maximum output current, 6.85V maximum output voltage, 1.71Ω resistivity of coils and input voltage as 10V.

5.4.3 Buck converter simulations of table top cage setup

Simulations of the converter are made by using PSIM. In Figure 5.23 schematic of the converter is shown. Values of the elements are written with respect to calculations made in previous section. In Figure 5.24 the time when the system is stable is around 0.013 seconds. In Figure 5.25 voltage ripple of the output potential can be seen where after stability the maximum voltage is 6.8385691V and minimum voltage is 6.8162799V. This will effect the magnetic field and can cause additional disturbances. Also current ripple can be seen in Figure 5.26 where after stability the maximum current is 3.9850253A and minimum current is 3.9847161A. Where voltage ripple is 1.5% and current ripple is 0.004%. This low ripple(noise) values and fast stability

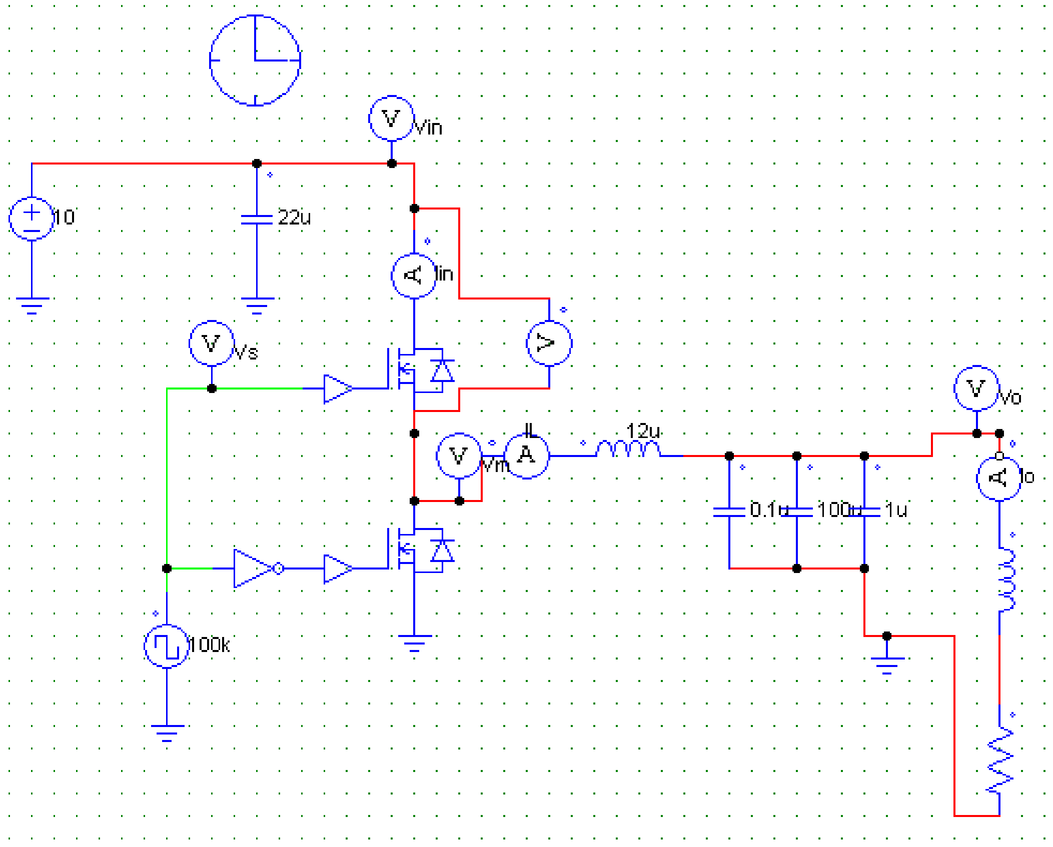
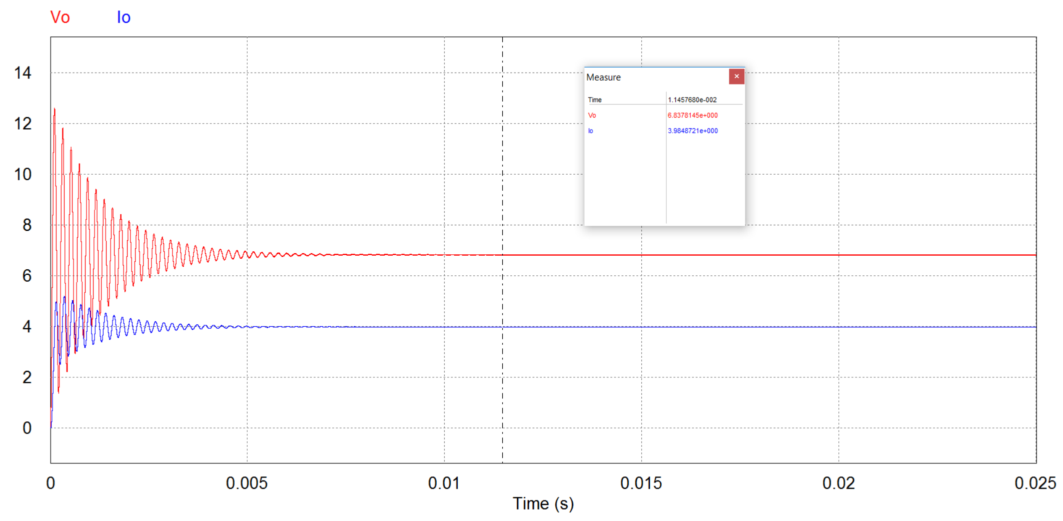


Figure 5.23 : Buck converter schematic for Cenco Helmholtz coil.

is important. An example magnetic field change in Figure 4.6, on every axis there are sharp changes. For negative magnetic field output voltage will be reversed by switches. But in the bigger system there will be an additional power supply in addition to reversing switch in order to have a fast response system.

5.5 Interfaces of Cage

Equations for calculation of the magnetic field will be dealt on a computer due to their complex functions. An MCU will control the reversing switches, buck converters which supplies power to the cage; and magnetometer for feedback of magnetic field direction and magnitude. It is important that reaction time of the power supply should be greater than the sampling time of the magnetometer. If their timings are reversed in a way that power supplies react faster than the sampling time, system cannot reach the desired magnetic values and thus putting itself in a higher noise state and nearly infinite loop.



Time	1.1457680e-002
Vo	6.8378145e+000
Io	3.9848721e+000

Figure 5.24 : Graph of output voltage and current.

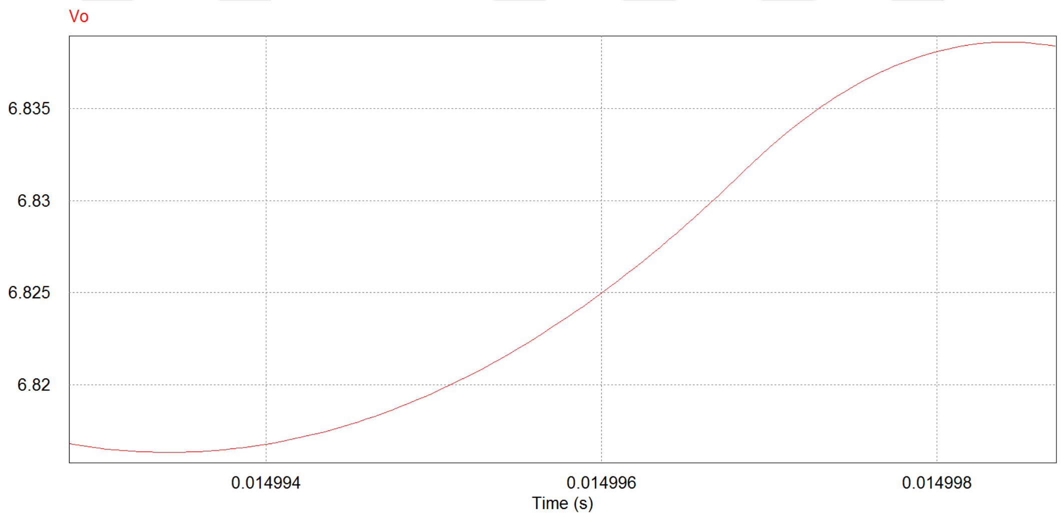


Figure 5.25 : Output voltage ripple.

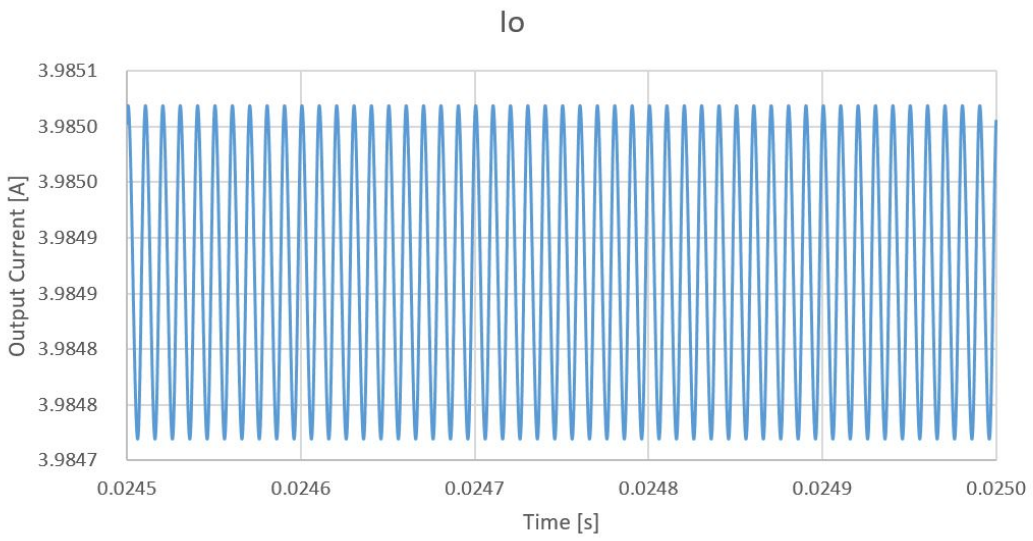


Figure 5.26 : Output current ripple.

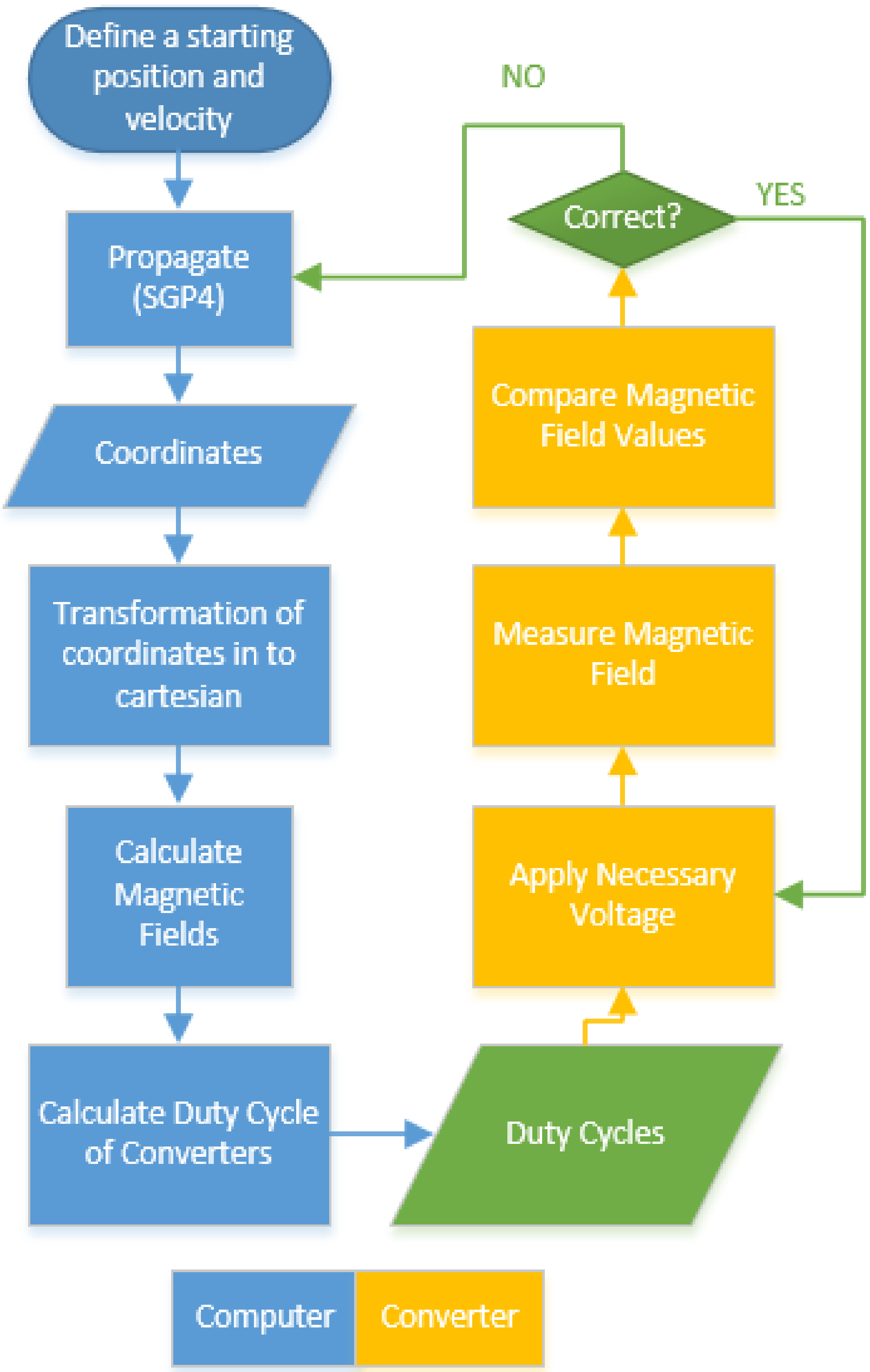

Define a starting position and velocity
Propagate (SGP4)
Coordinates
Transformation of coordinates in to cartesian
Calculate Magnetic Fields
Calculate Duty Cycle of Converters
Duty Cycles
Apply Necessary Voltage
Measure Magnetic Field
Compare Magnetic Field Values
Correct?
NO
YES
Computer
Converter

Figure 5.27 : Basic flow chart of the Helmholtz cage test setup.

6. CONCLUSION AND FUTURE WORK

Current and planned launches and satellite trends explained. Increasing failure rate is a big problem. Testing gets more important. In this thesis first step is taken to build an in house full functional test setup. Different size and geometry Helmholtz cages are designed and compared. It is seen that square cage has more uniform magnetic field distribution inside the desired envelope. Desired envelope should be in the middle and its should have a half of the radius's length for its one edge or 0.25 of length of a square edge.

This is a multidisciplinary work. Suggestion is that to separate this topic in to 3 different projects in order to have a more redundant and accurate system.

First; a power supply for table top model of the cage will be made to test basic system functions. Secondly a small magnetometer and special testing mechanism will be build in order to measure the magnetic field grid by grid inside the envelope. Results will be compared and calibrations are going to be made. Future work is to build a Helmholtz cage which will be able to generate a 30cmx30cmx30cm uniform envelope for testing satellites. New power supply design will be made to enable smaller step size with faster response time.

REFERENCES

[1] **Url-1** http://www.ulalaunch.com/ula-releases-application-for-university. aspx, Date:10.08.2016

[2] **Url-2** http://www.nanosats.eu/, Date:20.09.2016

[3] **Bouwmeester, J. and Guo, J.** (2010). Survey of worldwide pico-and nanosatellite missions, distributions and subsystem technology, *ActaAstronautica67(2010)854–862*

[4] **Swartwout, M. and Jayne, C.** (2016). University-Class Spacecraft by the Numbers: Success, Failure, Debris. (But Mostly Success.), *30th Annual AIAA/USU Conference on Small Satellites,* Logan, UT, USA.

[5] **Url-3** http://www.cubespace.co.za/, Date:20.09.2016

[6] **Url-4** https://www.sparkfun.com/products/10724,, Date:18.09.2016

[7] **Url-5** http://bluecanyontech.com/, Date: 18.09.2016

[8] **Url-6** https://www.cubesatshop.com/, Date: 18.09.2016

[9] **Url-7** http://hs-bremen.de/internet/de/forschung/projekte/detail/index_15601.html, Date: 24.09.2016

[10] **Url-8** http://abet-technologies.com/solar-simulators/solar-simulator-elements/, Date: 03.09.2016

[11] **Url-9** http://geo.phys.spbu.ru/ tsyganenko/modeling.html, Date: 12.07.2016

[12] **Vallado, D.** (2013). Fundamentals of Astrodynamics and Applications

[13] **Url-10** http://www.directvacuum.com/, Date: 23.09.2016

[14] **DeTroye, D. and Chase, R.** (1994). The Calculation and Measurement of Helmholtz Coil Fields

[15] **Restrepo-Alvarez, A.F.**, **Franco-Mejia, E. and Pinedo-Jaramillo, C.R.** (2012). Study and analysis of magnetic field homogeneity of square and circular Helmholtz coil pairs: A Taylor series approximation, *IEEE - Andean Region International Conference*, Universidad Politecnica Salesiana Cuenca, Azuay, Ecuador.

[16] **Central Scientific Company**. Operating Instructions - Specific Charge of Electron Apparatus and Electron Tube, *CENCO NOS. 71267 AND 71266*, 1700 Irving Park Road, Chicago, Ill, USA

[17] **Url-11** http://www.st.com/en/microcontrollers/stm32f746ng.html, Date: 23.09.2016

APPENDICES

APPENDIX A.1

```
clear all
clc
file1=pwd;
addpath(genpath(file1))
global GEOPACK2
%% Constants
% Iteration Number
N=6000;
%  The Sample Time
delt=1; % sec
dt=delt;
% External Torque
Nt=3.6E-10;
%  Constants
RE=6378;              % RE: Earth's Radius at Equator, km
mu=398601;         % mu: The Earth's gravitational constant ...
    (km^3/s^2)
%%
%~~~~~~~~~~~~~~~~~~~~~~~~%
%    ORBIT PROPAGATION    %
%~~~~~~~~~~~~~~~~~~~~~~~~%
startyear=2005;
startdayofyr=1;
mon1=1; day1=1; hr1=0; minute1=0; sec1=0;
stopyear=2005;
stopdayofyr=1;
mon2=1; day2=1; hr2=1; minute2=40; sec2=0; % for N=6000
% mon2=1; day2=1; hr2=1; minute2=40; sec2=0; % for N=6000*10
deltamin=delt/60; % minutes
starttime=[startyear;startdayofyr;mon1;day1;hr1;minute1;sec1];
stoptime=[stopyear;stopdayofyr;mon2;day2;hr2;minute2;sec2];
[RLL]=Orbit_Prop (N, starttime, stoptime, deltamin); % in ECEF: ...
%latitude(deg), longitude(deg), altitude (m)
h=RLL(:,3); % altitude in meters
lat=RLL(:,1);
lon=RLL(:,2);
wo=sqrt(mu/(RLL(:,3)/1000+RE).^3);
%% Orbital Parameters
file_ID=fopen('Sat.TLE');
[raan, ap, inc]=TLE_Reader(mu, file_ID); %% raan(deg), ap (deg), ...
    inc (deg)
%% Sun Direction Model
ISEC=sec1; MIN=minute1; IHOUR=hr1; IDAY=day1; IYEAR=startyear;

for i=1:N
THETA(i)=(90-lat(i))*pi/180;
PHI(i)=lon(i)*pi/180;
if ISEC<61
ISEC=ISEC+i-1;
else
ISEC=ISEC-60;
```

```
MIN=MIN+1;
end
if MIN>60 || MIN==60
MIN=mod(MIN,60);
IHOUR=IHOUR+1;
end
if IHOUR>24 || IHOUR==24
IHOUR=mod(IHOUR,24);
IDAY=IDAY+1;
end
[GST(i),SLONG,SRASN,SDEC] = GEOPACK_SUN(IYEAR,IDAY,IHOUR,MIN,ISEC);
GEOPACK_RECALCnew (IYEAR,IDAY,IHOUR,MIN,ISEC);
RR=(h(i)/1000+RE)./RE; % km
[BR(i),BTHETA(i),BPHI(i)] = ...
    GEOPACK_IGRF_GEO(RR,THETA(i),PHI(i)); % nT
[BX(i),BY(i),BZ(i)]= Bspcar(BR(i), ...
    BTHETA(i),BPHI(i),PHI(i),THETA(i),1);
% [A(i),B(i),C(i)] = GEOPACK_SPHCAR (BR(i),BTHETA(i),BPHI(i),1);
% [Aa(i),Ba(i),Ca(i)] = GEOPACK_SPHCAR (A(i),B(i),C(i),-1);
end

Br1=sqrt(BR.^2+BTHETA.^2+BPHI.^2);

figure
plot(BPHI,'y'); %azimuth
hold on; plot(BR,'k'); %radius
hold on; plot(BTHETA,'r'); %polar angle
hold on; plot(Br1,'b'); %polar angle
legend('B_\phi','B_r','B_\theta');
xlabel('Time (s)');
ylabel('B (nT)');

Br3=sqrt(BX.^2+BY.^2+BZ.^2);
% Br4=sqrt(Aa.^2+Ba.^2+Ca.^2);

figure
subplot(2,2,1)
plot(Br1,'b');
hold on; plot(Br3,'k');
% hold on; plot(Bz,'r');
% legend('B_x','B_y','B_z');
xlabel('Time (s)');
ylabel('Br (nT)');
subplot(2,2,2)
plot(BX,'b');
legend('B_x');
xlabel('Time (s)');
ylabel('B (nT)');
subplot(2,2,3)
plot(BY,'k');
legend('B_y');
xlabel('Time (s)');
ylabel('B (nT)');
subplot(2,2,4)
plot(BZ,'r');
legend('B_z');
xlabel('Time (s)');
ylabel('B (nT)');
```

APPENDIX A.2

```
TC = input('\nEnter TurnCurrent Value\n');
Envelope = input('\nEnter the edge value of the envolpe cube in...
 [cm]\n');
control=1;
Type_Coil=0;
while control==1
Type_Coil = ('\nEnter 1 for circular cage, 2 for square cage\n');
if Type_Coil==1
Radius = Envelope*2;
Circumference = pi()*Radius*2;
control=2;
elseif Type_Coil==2
Edge = Envelope*4;
Circumference = Edge*4;
control=2;
else
Type_Coil = ('\nEnter 1 for circular cage, 2 for square cage\n');
control=1;
end
end
Imin = ('\nEnter the desired minimum current value to be applied...
to coils\n');
Imax = ('\nEnter the desired maximum current value to be applied...
 to coils\n');
Conductivity = ('\nEnter the conductivity value of the wire used...
 in coils in [S/m]\n'); %5.96e7
Resistivity = ('\nEnter the Resistivity value of the wire used in...
 coils in [ohm/m]\n'); %1.7e-8
Density_of_Wire = ('\nEnter the Density value of the wire used in...
 coils in [g/cm^3]\n'); %8.96
Dw = ('\nEnter the Wire Diameter value of the wire used in coils ...
    in...
 [mm]\n'); %1
Area_Wire = ((pi()*Dw^2)/4)*0.01;
Sell_Weigth = ('\nEnter the Selling Weight of the wire used in ...
    coils...
 in [kg]\n'); %1
Sell_Length = Sell_Weight/(Area_Wire*Density_of_Wire);

CopperLength_1coil = (Circumference*Turn)/100;
R_1coil = (Resistivity*CopperLength_1coil)/(Area_Wire*0.0001);
R_2coil = 2*R_1coil;
Vneeded_max = Imax*R_2coil
Vneeded_min = Imin*R_2coil
```

APPENDIX A.3

```
Vin = input('\nEnter the input voltage of coil converters [V]\n ...
Must be greater than desired output voltage\n');
Vout_min = input('\nEnter the minimum output voltage for required...
 from coils [V]\nWhich is calculated in TelHesap.m\n');
Vout_max = input('\nEnter the maximum output voltage for required...
 from coils [V]\nWhich is calculated in TelHesap.m\n');
Vout=Vout_max;
fsw = input('\nEnter the switching frequecny[Hz]\n');
D = Vout/Vin;
DeltaIL = input('\nEnter the desired current ripple\n');
Iout = input('\nEnter the maximum current\n');
Lmin = (Vout*(1-D))/(DeltaIL*fsw)
Ipk = Iout + DeltaIL;
DVout = input('\nEnter the desired output voltage ripple in %...
 form\n');
Cmin_out = (Vout*(1-D))/(DVout*Vout*0.01*8*Lmin*(2*fsw)^2)
DVin = input('\nEnter the input voltage ripple in % form\n');
Cmin_in = ((Iout)/(2*fsw*DVin*Vin*0.01))*((D/2)-D^2)
```

APPENDIX A.4

```
function [BX,BY,BZ]= Bspcar(BR, BTHETA,BPHI,PHI,THETA,N)

for i=1:N
% BR
Bxr(i)=BR(i)*sin(THETA)*cos(PHI);
Byr(i)=BR(i)*sin(THETA)*sin(PHI);
Bzr(i)=BR(i)*cos(THETA);

%BPHI
Bxp(i)=-BPHI(i)*sin(PHI);
Byp(i)=BPHI(i)*cos(PHI);
Bzp(i)=0;

%BTHETA
Bxt(i)=BTHETA(i)*cos(THETA)*cos(PHI);
Byt(i)=BTHETA(i)*cos(THETA)*sin(PHI);
Bzt(i)=-BTHETA(i)*sin(THETA);

%B Cartesian
BX(i)=Bxr(i)+Bxp(i)+Bxt(i);
BY(i)=Byr(i)+Byp(i)+Byt(i);
BZ(i)=Bzr(i)+Bzp(i)+Bzt(i);

end

end
```

APPENDIX A.5

```
function [BR,BTHETA,BPHI] = GEOPACK_IGRF_GEO(R,THETA,PHI)
% function [BR,BTHETA,BPHI] = GEOPACK_IGRF_GEO(R,THETA,PHI)
% c
%       SUBROUTINE IGRF_GEO (R,THETA,PHI,BR,BTHETA,BPHI)
% C  CALCULATES COMPONENTS OF THE MAIN (INTERNAL) GEOMAGNETIC ...
   FIELD IN THE SPHERICAL GEOGRAPHIC
% C  (GEOCENTRIC) COORDINATE SYSTEM, USING IAGA INTERNATIONAL ...
   GEOMAGNETIC REFERENCE MODEL
% C  COEFFICIENTS  (e.g., ...
   http://www.ngdc.noaa.gov/IAGA/wg8/igrf2000.html)
```

```
% C
% C  BEFORE THE FIRST CALL OF THIS SUBROUTINE, OR IF THE DATE ...
    (IYEAR AND IDAY) WAS CHANGED,
% C  THE MODEL COEFFICIENTS SHOULD BE UPDATED BY CALLING THE ...
    SUBROUTINE RECALC
% C
% C-----INPUT PARAMETERS:
% C
% C   R, THETA, PHI - SPHERICAL GEOGRAPHIC (GEOCENTRIC) COORDINATES:
% C   RADIAL DISTANCE R IN UNITS RE=6371.2 KM, COLATITUDE THETA ...
    AND LONGITUDE PHI IN RADIANS
% C
% C-----OUTPUT PARAMETERS:
% C
% C     BR, BTHETA, BPHI - SPHERICAL COMPONENTS OF THE MAIN ...
    GEOMAGNETIC FIELD IN NANOTESLA
% C      (POSITIVE BR OUTWARD (Up), BTHETA SOUTHWARD, BPHI EASTWARD)
% C
% C     LAST MODIFICATION:  MARCH 30, 2003.
% C     THIS VERSION OF THE  CODE ACCEPT DATES FROM 1965 THROUGH ...
    2005.
% c
% C     AUTHOR: N. A. TSYGANENKO
% C
% C

%       COMMON /GEOPACK2/ G(66),H(66),REC(66)
global GEOPACK2

%       DIMENSION A(11),B(11)

C=cos(THETA);
S=sin(THETA);
CF=cos(PHI);
SF=sin(PHI);
% C
PP=1./R;
P=PP;
% C
% C  IN THIS NEW VERSION, THE OPTIMAL VALUE OF THE PARAMETER NM ...
    (MAXIMAL ORDER OF THE SPHERICAL
% C    HARMONIC EXPANSION) IS NOT USER-PRESCRIBED, BUT ...
    CALCULATED INSIDE THE SUBROUTINE, BASED
% C      ON THE VALUE OF THE RADIAL DISTANCE R:
% C
IRP3=R+3;
NM=4+30/IRP3;
if (NM > 10), NM=10; end;

K=NM+1;
for N=1:K,
%       DO 150 N=1,K
P=P*PP;
A(N)=P;
B(N)=P*N; % 150
end

P=1.;
D=0.;
```

```
BBR=0.;
BBT=0.;
BBF=0.;

for M=1:K,
%       DO 200 M=1,K
%IF(M == 1) GOTO 160
if (M~=1),
MM=M-1;
W=X;
X=W*CF+Y*SF;
Y=Y*CF-W*SF;
%          GOTO 170
else
X=0.; % 160
Y=1.;
end
Q=P;%170
Z=D;
BI=0.;
P2=0.;
D2=0.;
for N=M:K,
%          DO 190 N=M,K
AN=A(N);
MN=N*(N-1)/2+M;
E=GEOPACK2.G(MN);
HH=GEOPACK2.H(MN);
W=E*Y+HH*X;
BBR=BBR+B(N)*W*Q;
BBT=BBT-AN*W*Z;
%             IF(M == 1) GOTO 180
if (M~=1),
QQ=Q;
if (S < 1.E-5), QQ=Z; end
BI=BI+AN*(E*X-HH*Y)*QQ;
end
XK=GEOPACK2.REC(MN); % 180
DP=C*Z-S*Q-XK*D2;
PM=C*Q-XK*P2;
D2=Z;
P2=Q;
Z=DP;
Q=PM; % 190
end
D=S*D+C*P;
P=S*P;
%          IF(M == 1) GOTO 200
if (M~=1),
BI=BI*MM;
BBF=BBF+BI;
end
end %200   CONTINUE
% C
BR=BBR;
BTHETA=BBT;
%       IF(S < 1.E-5) GOTO 210
if (S>=1.E-5),
BPHI=BBF/S;
```

```
return;
end
if (C < 0.), BBF=-BBF; end % 210
BPHI=BBF;

%        RETURN
%        END
% end of function IGRF_GEO
% C
% ...
   c==========================================================================
% c
```

APPENDIX A.6

```
function GEOPACK_RECALCnew (IYEAR,IDAY,IHOUR,MIN,ISEC)
%        SUBROUTINE RECALC_08 ...
    (IYEAR,IDAY,IHOUR,MIN,ISEC,VGSEX,VGSEY,VGSEZ)
% C
% C  1. PREPARES ELEMENTS OF ROTATION MATRICES FOR ...
   TRANSFORMATIONS OF VECTORS BETWEEN
% C      SEVERAL COORDINATE SYSTEMS, MOST FREQUENTLY USED IN ...
   SPACE PHYSICS.
% C
% C  2. PREPARES COEFFICIENTS USED IN THE CALCULATION OF THE ...
   MAIN GEOMAGNETIC FIELD
% C       (IGRF MODEL)
% C
% C  THIS SUBROUTINE SHOULD BE INVOKED BEFORE USING THE ...
   FOLLOWING SUBROUTINES:
% C  IGRF_GEO_08, IGRF_GSW_08, DIP_08, GEOMAG_08, GEOGSW_08, ...
   MAGSW_08, SMGSW_08, GSWGSE_08,
% c  GEIGEO_08, TRACE_08, STEP_08, RHAND_08.
% C
% C  THERE IS NO NEED TO REPEATEDLY INVOKE RECALC_08, IF ...
   MULTIPLE CALCULATIONS ARE MADE
% C    FOR THE SAME DATE/TIME AND SOLAR WIND FLOW DIRECTION.
% C
% C-----INPUT PARAMETERS:
% C
% C      IYEAR   -  YEAR NUMBER (FOUR DIGITS)
% C      IDAY  -  DAY OF YEAR (DAY 1 = JAN 1)
% C      IHOUR -  HOUR OF DAY (00 TO 23)
% C      MIN   -  MINUTE OF HOUR (00 TO 59)
% C      ISEC  -  SECONDS OF MINUTE (00 TO 59)
% C      VGSEX,VGSEY,VGSEZ - GSE (GEOCENTRIC SOLAR-ECLIPTIC) ...
   COMPONENTS OF THE OBSERVED
% C                                SOLAR WIND FLOW VELOCITY (IN KM/S)
% C
% C  IMPORTANT: IF ONLY QUESTIONABLE INFORMATION (OR NO ...
   INFORMATION AT ALL) IS AVAILABLE
% C               ON THE SOLAR WIND SPEED, OR, IF THE STANDARD GSM ...
   AND/OR SM COORDINATES ARE
% C               INTENDED TO BE USED, THEN SET VGSEX=-400.0 AND ...
   VGSEY=VGSEZ=0. IN THIS CASE,
% C               THE GSW COORDINATE SYSTEM BECOMES IDENTICAL TO ...
   THE STANDARD GSM.
```

```
% C
% C              IF ONLY SCALAR SPEED V OF THE SOLAR WIND IS ...
   KNOWN, THEN SETTING
% C              VGSEX=-V, VGSEY=29.78, VGSEZ=0.0 WILL TAKE INTO ...
   ACCOUNT THE ~4 degs
% C              ABERRATION OF THE MAGNETOSPHERE DUE TO EARTH'S ...
   ORBITAL MOTION
% C
% C              IF ALL THREE GSE COMPONENTS OF THE SOLAR WIND ...
   VELOCITY ARE AVAILABLE,
% C              PLEASE NOTE THAT IN SOME SOLAR WIND DATABASES ...
   THE ABERRATION EFFECT
% C              HAS ALREADY BEEN TAKEN INTO ACCOUNT BY ...
   SUBTRACTING 29.78 KM/S FROM VYGSE;
% C              IN THAT CASE, THE UNABERRATED (OBSERVED) VYGSE ...
   VALUES SHOULD BE RESTORED
% C              BY ADDING BACK THE 29.78 KM/S CORRECTION. ...
   WHETHER OR NOT TO DO THAT, MUST
% C              BE EITHER VERIFIED WITH THE DATA ORIGINATOR OR ...
   DETERMINED BY AVERAGING
% C              VGSEY OVER A SUFFICIENTLY LONG TIME INTERVAL.
% C
% C-----OUTPUT PARAMETERS:  NONE (ALL OUTPUT QUANTITIES ARE PLACED
% C                         INTO THE COMMON BLOCKS /GEOPACK1/ ...
   AND /GEOPACK2/)
% C
% C    OTHER SUBROUTINES CALLED BY THIS ONE: SUN_08
% C
% C    AUTHOR:  N.A. TSYGANENKO
% C    DATE:    DEC.1, 1991
% C
% C    REVISION OF JANUARY 30, 2015:
% C
% C     The table of IGRF coefficients was extended to include ...
   those for the epoch 2015 (igrf-12)
% c          (for details, see ...
   http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html)
% C
% C    REVISION OF NOVEMBER 30, 2010:
% C
% C     The table of IGRF coefficients was extended to include ...
   those for the epoch 2010
% c          (for details, see ...
   http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html)
% C
% C    REVISION OF NOVEMBER 15, 2007: ADDED THE POSSIBILITY TO ...
   TAKE INTO ACCOUNT THE OBSERVED
% C     DEFLECTION OF THE SOLAR WIND FLOW FROM STRICTLY RADIAL ...
   DIRECTION. TO THAT END, THREE
% C     GSE COMPONENTS OF THE SOLAR WIND VELOCITY WERE ADDED TO ...
   THE INPUT PARAMETERS.
% C
% c    CORRECTION OF MAY 9, 2006:  INTERPOLATION OF THE ...
   COEFFICIENTS (BETWEEN
% C     LABELS 50 AND 105) IS NOW MADE THROUGH THE LAST ELEMENT ...
   OF THE ARRAYS
% C     G(105)  AND H(105) (PREVIOUSLY MADE ONLY THROUGH N=66, ...
   WHICH IN SOME
% C     CASES CAUSED RUNTIME ERRORS)
% c
% C    REVISION OF MAY 3, 2005:
% C     The table of IGRF coefficients was extended to include ...
    those for the epoch 2005
% c       the maximal order of spherical harmonics was also ...
    increased up to 13
% c          (for details, see ...
    http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html)
% c
% C    REVISION OF APRIL 3, 2003:
% c    The code now includes preparation of the model ...
    coefficients for the subroutines
% c    IGRF_08 and GEOMAG_08. This eliminates the need for the ...
    SAVE statements, used
% c    in the old versions, making the codes easier and more ...
    compiler-independent.
% C
%        SAVE ISW
% C
%        COMMON /GEOPACK1/ ...
    ST0,CT0,SL0,CL0,CTCL,STCL,CTSL,STSL,SFI,CFI,
%       * ...
    SPS,CPS,DS3,CGST,SGST,PSI,A11,A21,A31,A12,A22,A32,A13,A23,A33,
%       * E11,E21,E31,E12,E22,E32,E13,E23,E33
global GEOPACK1;
% C
% C  THE COMMON BLOCK /GEOPACK1/ CONTAINS ELEMENTS OF THE ...
    ROTATION MATRICES AND OTHER
% C   PARAMETERS RELATED TO THE COORDINATE TRANSFORMATIONS ...
    PERFORMED BY THIS PACKAGE
% C
%        COMMON /GEOPACK2/ G(105),H(105),REC(105)
global GEOPACK2;
% C
% C  THE COMMON BLOCK /GEOPACK2/ CONTAINS COEFFICIENTS OF THE ...
    IGRF FIELD MODEL, CALCULATED
% C    FOR A GIVEN YEAR AND DAY FROM THEIR STANDARD EPOCH ...
    VALUES. THE ARRAY REC CONTAINS
% C    COEFFICIENTS USED IN THE RECURSION RELATIONS FOR LEGENDRE ...
    ASSOCIATE POLYNOMIALS.
% C
%        DIMENSION ...
    G65(105),H65(105),G70(105),H70(105),G75(105),H75(105),
%       + ...
    G80(105),H80(105),G85(105),H85(105),G90(105),H90(105),G95(105),
%       + ...
    H95(105),G00(105),H00(105),G05(105),H05(105),G10(105),H10(105),
%       + G15(105),H15(105),DG15(45),DH15(45)
% C
%        DATA ISW /0/
% c
G65=[0.,-30334.,-2119.,-1662.,2997.,1594.,1297.,-2038.,1292.,...
856.,957.,804.,479.,-390.,252.,-219.,358.,254.,-31.,-157.,-62.,...
45.,61.,8.,-228.,4.,1.,-111.,75.,-57.,4.,13.,-26.,-6.,13.,1.,13.,...
5.,-4.,-14.,0.,8.,-1.,11.,4.,8.,10.,2.,-13.,10.,-1.,-1.,5.,1.,-2.,...
-2.,-3.,2.,-5.,-2.,4.,4.,0.,2.,2.,0.,39*0.];

H65=[0.,0.,5776.,0.,-2016.,114.,0.,-404.,240.,-165.,0.,148.,...
-269.,13.,-269.,0.,19.,128.,-126.,-97.,81.,0.,-11.,100.,68.,-32.,...
```

```
-8.,-7.,0.,-61.,-27.,-2.,6.,26.,-23.,-12.,0.,7.,-12.,9.,-16.,4.,...
24.,-3.,-17.,0.,-22.,15.,7.,-4.,-5.,10.,10.,-4.,1.,0.,2.,1.,2.,...
6.,-4.,0.,-2.,3.,0.,-6.,39*0.];
% c
G70=[0.,-30220.,-2068.,-1781.,3000.,1611.,1287.,-2091.,1278.,...
838.,952.,800.,461.,-395.,234.,-216.,359.,262.,-42.,-160.,-56.,...
43.,64.,15.,-212.,2.,3.,-112.,72.,-57.,1.,14.,-22.,-2.,13.,-2.,...
14.,6.,-2.,-13.,-3.,5.,0.,11.,3.,8.,10.,2.,-12.,10.,-1.,0.,3.,...
1.,-1.,-3.,-3.,2.,-5.,-1.,6.,4.,1.,0.,3.,-1.,39*0.];

H70=[0.,0.,5737.,0.,-2047.,25.,0.,-366.,251.,-196.,0.,167.,...
-266.,26.,-279.,0.,26.,139.,-139.,-91.,83.,0.,-12.,100.,72.,-37.,...
-6.,1.,0.,-70.,-27.,-4.,8.,23.,-23.,-11.,0.,7.,-15.,6.,-17.,6.,...
21.,-6.,-16.,0.,-21.,16.,6.,-4.,-5.,10.,11.,-2.,1.,0.,1.,1.,3.,...
4.,-4.,0.,-1.,3.,1.,-4.,39*0.];
% c
G75=[0.,-30100.,-2013.,-1902.,3010.,1632.,1276.,-2144.,1260.,...
830.,946.,791.,438.,-405.,216.,-218.,356.,264.,-59.,-159.,-49.,...
45.,66.,28.,-198.,1.,6.,-111.,71.,-56.,1.,16.,-14.,0.,12.,-5.,...
14.,6.,-1.,-12.,-8.,4.,0.,10.,1.,7.,10.,2.,-12.,10.,-1.,-1.,4.,...
1.,-2.,-3.,-3.,2.,-5.,-2.,5.,4.,1.,0.,3.,-1.,39*0.];

H75=[0.,0.,5675.,0.,-2067.,-68.,0.,-333.,262.,-223.,0.,191.,...
-265.,39.,-288.,0.,31.,148.,-152.,-83.,88.,0.,-13.,99.,75.,-41.,...
-4.,11.,0.,-77.,-26.,-5.,10.,22.,-23.,-12.,0.,6.,-16.,4.,-19.,6.,...
18.,-10.,-17.,0.,-21.,16.,7.,-4.,-5.,10.,11.,-3.,1.,0.,1.,1.,3.,...
4.,-4.,-1.,-1.,3.,1.,-5.,39*0.];
% c
G80=[0.,-29992.,-1956.,-1997.,3027.,1663.,1281.,-2180.,1251.,...
833.,938.,782.,398.,-419.,199.,-218.,357.,261.,-74.,-162.,-48.,...
48.,66.,42.,-192.,4.,14.,-108.,72.,-59.,2.,21.,-12.,1.,11.,-2.,...
18.,6.,0.,-11.,-7.,4.,3.,6.,-1.,5.,10.,1.,-12.,9.,-3.,-1.,7.,2.,...
-5.,-4.,-4.,2.,-5.,-2.,5.,3.,1.,2.,3.,0.,39*0.];

H80=[0.,0.,5604.,0.,-2129.,-200.,0.,-336.,271.,-252.,0.,212.,...
-257.,53.,-297.,0.,46.,150.,-151.,-78.,92.,0.,-15.,93.,71.,-43.,...
-2.,17.,0.,-82.,-27.,-5.,16.,18.,-23.,-10.,0.,7.,-18.,4.,-22.,9.,...
16.,-13.,-15.,0.,-21.,16.,9.,-5.,-6.,9.,10.,-6.,2.,0.,1.,0.,3.,...
6.,-4.,0.,-1.,4.,0.,-6.,39*0.];
% c
G85=[0.,-29873.,-1905.,-2072.,3044.,1687.,1296.,-2208.,1247.,...
829.,936.,780.,361.,-424.,170.,-214.,355.,253.,-93.,-164.,-46.,...
53.,65.,51.,-185.,4.,16.,-102.,74.,-62.,3.,24.,-6.,4.,10.,0.,21.,...
6.,0.,-11.,-9.,4.,4.,4.,-4.,5.,10.,1.,-12.,9.,-3.,-1.,7.,1.,-5.,...
-4.,-4.,3.,-5.,-2.,5.,3.,1.,2.,3.,0.,39*0.];

H85=[0.,0.,5500.,0.,-2197.,-306.,0.,-310.,284.,-297.,0.,232.,...
249.,69.,-297.,0.,47.,150.,-154.,-75.,95.,0.,-16.,88.,69.,-48.,...
-1.,21.,0.,-83.,-27.,-2.,20.,17.,-23.,-7.,0.,8.,-19.,5.,-23.,11.,...
14.,-15.,-11.,0.,-21.,15.,9.,-6.,-6.,9.,9.,-7.,2.,0.,1.,0.,3.,...
6.,-4.,0.,-1.,4.,0.,-6.,39*0.];
% c
G90=[0., -29775.,  -1848.,  -2131.,   3059.,   1686.,   1314.,...
-2239.,   1248.,    802.,    939.,    780.,    325.,   -423.,...
141.,   -214.,    353.,    245.,   -109.,   -165.,    -36.,...
61.,      65.,     59.,   -178.,      3.,     18.,    -96.,...
77.,     -64.,      2.,     26.,     -1.,      5.,      9.,...
0.,      23.,      5.,     -1.,    -10.,    -12.,      3.,...
4.,       2.,     -6.,      4.,      9.,      1.,    -12.,...
```

```
9.,      -4.,      -2.,       7.,       1.,      -6.,      -3.,...
-4.,       2.,      -5.,      -2.,       4.,       3.,       1.,...
3.,       3.,       0.,   39*0.];
% C
H90=[0.,       0.,   5406.,       0.,  -2279.,   -373.,       0.,...
-284.,    293.,   -352.,       0.,    247.,   -240.,      84.,...
-299.,      0.,     46.,    154.,   -153.,    -69.,      97.,...
0.,    -16.,      82.,      69.,     -52.,       1.,      24.,...
0.,    -80.,     -26.,       0.,      21.,      17.,     -23.,...
-4.,       0.,      10.,     -19.,       6.,     -22.,      12.,...
12.,     -16.,     -10.,       0.,     -20.,      15.,      11.,...
-7.,      -7.,       9.,       8.,      -7.,       2.,       0.,...
2.,       1.,       3.,       6.,      -4.,       0.,      -2.,...
3.,      -1.,      -6.,   39*0.];
% C
G95=[0., -29692.,  -1784.,  -2200.,   3070.,   1681.,   1335.,...
-2267.,   1249.,    759.,    940.,    780.,    290.,   -418.,...
122.,   -214.,    352.,    235.,   -118.,   -166.,    -17.,...
68.,      67.,      68.,   -170.,      -1.,      19.,     -93.,...
77.,     -72.,       1.,      28.,       5.,       4.,       8.,...
-2.,      25.,       6.,      -6.,      -9.,     -14.,       9.,...
6.,      -5.,      -7.,       4.,       9.,       3.,     -10.,...
8.,      -8.,      -1.,      10.,      -2.,      -8.,      -3.,...
-6.,       2.,      -4.,      -1.,       4.,       2.,       2.,...
5.,       1.,       0.,    39*0.];
% C
H95=[0.,       0.,   5306.,       0.,  -2366.,   -413.,       0.,...
-262.,    302.,   -427.,       0.,    262.,   -236.,      97.,...
-306.,      0.,     46.,    165.,   -143.,    -55.,     107.,...
0.,    -17.,      72.,      67.,     -58.,       1.,      36.,...
0.,    -69.,     -25.,       4.,      24.,      17.,     -24.,...
-6.,       0.,      11.,     -21.,       8.,     -23.,      15.,...
11.,     -16.,      -4.,       0.,     -20.,      15.,      12.,...
-6.,      -8.,       8.,       5.,      -8.,       3.,       0.,...
1.,       0.,       4.,       5.,      -5.,      -1.,      -2.,...
1.,      -2.,      -7.,    39*0.];
% C
G00=[0.,-29619.4, -1728.2, -2267.7,  3068.4,  1670.9,  1339.6,...
-2288.,  1252.1,   714.5,   932.3,   786.8,    250.,   -403.,...
111.3,  -218.8,   351.4,   222.3,  -130.4,  -168.6,   -12.9,...
72.3,    68.2,    74.2,  -160.9,    -5.9,    16.9,   -90.4,...
79.0,   -74.0,      0.,    33.3,     9.1,     6.9,     7.3,...
-1.2,    24.4,     6.6,    -9.2,    -7.9,   -16.6,     9.1,...
7.0,    -7.9,     -7.,      5.,     9.4,      3.,   - 8.4,...
6.3,    -8.9,    -1.5,     9.3,    -4.3,    -8.2,    -2.6,...
-6.,     1.7,    -3.1,    -0.5,     3.7,      1.,      2.,...
4.2,     0.3,    -1.1,     2.7,    -1.7,    -1.9,     1.5,...
-0.1,     0.1,    -0.7,     0.7,     1.7,     0.1,     1.2,...
4.0,    -2.2,    -0.3,     0.2,     0.9,    -0.2,     0.9,...
-0.5,     0.3,    -0.3,    -0.4,    -0.1,    -0.2,    -0.4,...
-0.2,    -0.9,     0.3,     0.1,    -0.4,     1.3,    -0.4,...
0.7,    -0.4,     0.3,    -0.1,     0.4,      0.,     0.1];
% C
H00=[0.,       0.,  5186.1,      0., -2481.6,  -458.0,      0.,...
-227.6,   293.4,  -491.1,      0.,   272.6,  -231.9,   119.8,...
-303.8,      0.,    43.8,   171.9,  -133.1,   -39.3,   106.3,...
0.,   -17.4,    63.7,    65.1,   -61.2,     0.7,    43.8,...
0.,   -64.6,   -24.2,     6.2,     24.,    14.8,   -25.4,...
-5.8,     0.0,    11.9,   -21.5,     8.5,   -21.5,    15.5,...
```

```
8.9,   -14.9,    -2.1,     0.0,   -19.7,    13.4,    12.5,...
-6.2,    -8.4,     8.4,     3.8,    -8.2,     4.8,     0.0,...
1.7,     0.0,     4.0,     4.9,    -5.9,    -1.2,    -2.9,...
0.2,    -2.2,    -7.4,     0.0,     0.1,     1.3,    -0.9,...
-2.6,     0.9,    -0.7,    -2.8,    -0.9,    -1.2,    -1.9,...
-0.9,     0.0,    -0.4,     0.3,     2.5,    -2.6,     0.7,...
0.3,     0.0,     0.0,     0.3,    -0.9,    -0.4,     0.8,...
0.0,    -0.9,     0.2,     1.8,    -0.4,    -1.0,    -0.1,...
0.7,     0.3,     0.6,     0.3,    -0.2,    -0.5,    -0.9];
% C
G05=[0.,-29554.6, -1669.0, -2337.2,  3047.7,  1657.8,  1336.3,...
-2305.8,  1246.4,   672.5,   920.6,   798.0,   210.7,  -379.9,...
100.0,  -227.0,   354.4,   208.9,  -136.5,  -168.1,   -13.6,...
73.6,    69.6,    76.7,  -151.3,   -14.6,    14.6,   -86.4,...
79.9,   -74.5,    -1.7,    38.7,    12.3,     9.4,     5.4,...
1.9,    24.8,     7.6,   -11.7,    -6.9,   -18.1,    10.2,...
9.4,   -11.3,    -4.9,     5.6,     9.8,     3.6,    -6.9,...
5.0,   -10.8,    -1.3,     8.8,    -6.7,    -9.2,    -2.2,...
-6.1,     1.4,    -2.4,    -0.2,     3.1,     0.3,     2.1,...
3.8,    -0.2,    -2.1,     2.9,    -1.6,    -1.9,     1.4,...
-0.3,     0.3,    -0.8,     0.5,     1.8,     0.2,     1.0,...
4.0,    -2.2,    -0.3,     0.2,     0.9,    -0.4,     1.0,...
-0.3,     0.5,    -0.4,    -0.4,     0.1,    -0.5,    -0.1,...
-0.2,    -0.9,     0.3,     0.3,    -0.4,     1.2,    -0.4,...
0.8,    -0.3,     0.4,    -0.1,     0.4,    -0.1,    -0.2];
% C
H05=[0.,      0.0,  5078.0,      0.0, -2594.5,  -515.4,      0.0,...
-198.9,   269.7,  -524.7,      0.0,   282.1,  -225.2,   145.2,...
-305.4,      0.0,    42.7,   180.3,  -123.5,   -19.6,   103.9,...
0.0,   -20.3,    54.8,    63.6,   -63.5,     0.2,    50.9,...
0.0,   -61.1,   -22.6,     6.8,    25.4,    10.9,   -26.3,...
-4.6,     0.0,    11.2,   -20.9,     9.8,   -19.7,    16.2,...
7.6,   -12.8,    -0.1,     0.0,   -20.1,    12.7,    12.7,...
-6.7,    -8.2,     8.1,     2.9,    -7.7,     6.0,     0.0,...
2.2,     0.1,     4.5,     4.8,    -6.7,    -1.0,    -3.5,...
-0.9,    -2.3,    -7.9,     0.0,     0.3,     1.4,    -0.8,...
-2.3,     0.9,    -0.6,    -2.7,    -1.1,    -1.6,    -1.9,...
-1.4,     0.0,    -0.6,     0.2,     2.4,    -2.6,     0.6,...
0.4,     0.0,     0.0,     0.3,    -0.9,    -0.3,     0.9,...
0.0,    -0.8,     0.3,     1.7,    -0.5,    -1.1,     0.0,...
0.6,     0.2,     0.5,     0.4,    -0.2,    -0.6,    -0.9];
% C
G10=[0.00,-29496.57,-1586.42,-2396.06,3026.34,1668.17,1339.85,...
-2326.54,  1232.10,  633.73,  912.66, 808.97, 166.58,-356.83,...
89.40,  -230.87,  357.29,  200.26,-141.05,-163.17,  -8.03,...
72.78,    68.69,   75.92, -141.40, -22.83,  13.10, -78.09,...
80.44,   -75.00,   -4.55,   45.24,  14.00,  10.46,   1.64,...
4.92,    24.41,    8.21,  -14.50,  -5.59, -19.34,  11.61,...
10.85,   -14.05,   -3.54,    5.50,   9.45,   3.45,  -5.27,...
3.13,   -12.38,   -0.76,    8.43,  -8.42, -10.08,  -1.94,...
-6.24,     0.89,   -1.07,   -0.16,   2.45,  -0.33,   2.13,...
3.09,    -1.03,   -2.80,    3.05,  -1.48,  -2.03,   1.65,...
-0.51,     0.54,   -0.79,    0.37,   1.79,   0.12,   0.75,...
3.75,    -2.12,   -0.21,    0.30,   1.04,  -0.63,   0.95,...
-0.11,     0.52,   -0.39,   -0.37,   0.21,  -0.77,   0.04,...
-0.09,    -0.89,    0.31,    0.42,  -0.45,   1.08,  -0.31,...
0.78,    -0.18,    0.38,    0.02,   0.42,  -0.26,  -0.26];
% C
H10=[0.00,  0.00, 4944.26,    0.00,-2708.54, -575.73,    0.00,...
```

```
-160.40,251.75, -537.03,    0.00,  286.48, -211.03,  164.46,...
-309.72,  0.00,   44.58,  189.01, -118.06,   -0.01,  101.04,...
0.00,-20.90,   44.18,   61.54,  -66.26,    3.02,   55.40,...
0.00,-57.80,  -21.20,    6.54,   24.96,    7.03,  -27.61,...
-3.28,  0.00,   10.84,  -20.03,   11.83,  -17.41,   16.71,...
6.96,-10.74,    1.64,    0.00,  -20.54,   11.51,   12.75,...
-7.14, -7.42,    7.97,    2.14,   -6.08,    7.01,    0.00,...
2.73, -0.10,    4.71,    4.44,   -7.22,   -0.96,   -3.95,...
-1.99, -1.97,   -8.31,    0.00,    0.13,    1.67,   -0.66,...
-1.76,  0.85,   -0.39,   -2.51,   -1.27,   -2.11,   -1.94,...
-1.86,  0.00,   -0.87,    0.27,    2.13,   -2.49,    0.49,...
0.59,  0.00,    0.13,    0.27,   -0.86,   -0.23,    0.87,...
0.00, -0.87,    0.30,    1.66,   -0.59,   -1.14,   -0.07,...
0.54,  0.10,    0.49,    0.44,   -0.25,   -0.53,   -0.79];
% C
G15=[0.,-29442.0, -1501.0, -2445.1,  3012.9,  1676.7,  1350.7,...
-2352.3,  1225.6,   582.0,   907.6,   813.7,   120.4,  -334.9,...
70.4,  -232.6,   360.1,   192.4,  -140.9,  -157.5,     4.1,...
70.0,    67.7,    72.7,  -129.9,   -28.9,    13.2,   -70.9,...
81.6,   -76.1,    -6.8,    51.8,    15.0,     9.4,    -2.8,...
6.8,    24.2,     8.8,   -16.9,    -3.2,   -20.6,    13.4,...
11.7,   -15.9,    -2.0,     5.4,     8.8,     3.1,    -3.3,...
0.7,   -13.3,    -0.1,     8.7,    -9.1,   -10.5,    -1.9,...
-6.3,     0.1,     0.5,    -0.5,     1.8,    -0.7,     2.1,...
2.4,    -1.8,    -3.6,     3.1,    -1.5,    -2.3,     2.0,...
-0.8,     0.6,    -0.7,     0.2,     1.7,    -0.2,     0.4,...
3.5,    -1.9,    -0.2,     0.4,     1.2,    -0.8,     0.9,...
0.1,     0.5,    -0.3,    -0.4,     0.2,    -0.9,     0.0,...
0.0,    -0.9,     0.4,     0.5,    -0.5,     1.0,    -0.2,...
0.8,    -0.1,     0.3,     0.1,     0.5,    -0.4,    -0.3];
% c
H15=[0.0,    0.0,  4797.1,     0.0, -2845.6,  -641.9,     0.0,...
-115.3,  244.9,  -538.4,     0.0,   283.3,  -188.7,   180.9,...
-329.5,    0.0,    47.3,   197.0,  -119.3,    16.0,   100.2,...
0.0,  -20.8,    33.2,    58.9,   -66.7,     7.3,    62.6,...
0.0,  -54.1,   -19.5,     5.7,    24.4,     3.4,   -27.4,...
-2.2,    0.0,    10.1,   -18.3,    13.3,   -14.6,    16.2,...
5.7,   -9.1,     2.1,     0.0,   -21.6,    10.8,    11.8,...
-6.8,   -6.9,     7.8,     1.0,    -4.0,     8.4,     0.0,...
3.2,   -0.4,     4.6,     4.4,    -7.9,    -0.6,    -4.2,...
-2.8,   -1.2,    -8.7,     0.0,    -0.1,     2.0,    -0.7,...
-1.1,    0.8,    -0.2,    -2.2,    -1.4,    -2.5,    -2.0,...
-2.4,    0.0,    -1.1,     0.4,     1.9,    -2.2,     0.3,...
0.7,   -0.1,     0.3,     0.2,    -0.9,    -0.1,     0.7,...
0.0,   -0.9,     0.4,     1.6,    -0.5,    -1.2,    -0.1,...
0.4,   -0.1,     0.4,     0.5,    -0.3,    -0.4,    -0.8];
% c
DG15=[0.0,  10.3,    18.1,    -8.7,    -3.3,     2.1,     3.4,...
-5.5,  -0.7,   -10.1,    -0.7,     0.2,    -9.1,     4.1,...
-4.3,  -0.2,     0.5,    -1.3,    -0.1,     1.4,     3.9,...
-0.3,  -0.1,    -0.7,     2.1,    -1.2,     0.3,     1.6,...
0.3,  -0.2,    -0.5,     1.3,     0.1,    -0.6,    -0.8,...
0.2,   0.2,     0.0,    -0.6,     0.5,    -0.2,     0.4,...
0.1,  -0.4,     0.3];
% c
DH15=[0.0,   0.0,   -26.6,     0.0,   -27.4,   -14.1,     0.0,...
8.2,  -0.4,     1.8,     0.0,    -1.3,     5.3,     2.9,...
-5.2,   0.0,     0.6,     1.7,    -1.2,     3.4,     0.0,...
0.0,   0.0,    -2.1,    -0.7,     0.2,     0.9,     1.0,...
```

```
0.0,   0.8,     0.4,    -0.2,    -0.3,    -0.6,     0.1,...
-0.2,   0.0,    -0.3,     0.3,     0.1,     0.5,    -0.2,...
-0.3,   0.3,     0.0];
% C
%        IF (VGSEY.EQ.0..AND.VGSEZ.EQ.0..AND.ISW.NE.1) THEN
%        PRINT *, ''
%        PRINT *,
%       *' RECALC_08: RADIAL SOLAR WIND --> GSW SYSTEM IDENTICAL ...
   HERE'
%        PRINT *,
%       *' TO STANDARD GSM (I.E., XGSW AXIS COINCIDES WITH ...
   EARTH-SUN LINE)'
%        PRINT *, ''
%        ISW=1
%        ENDIF
%
%        IF ((VGSEY.NE.0..OR.VGSEZ.NE.0.).AND.ISW.NE.2) THEN ...
             %  CORRECTED NOV.27, 2009
%        PRINT *, ''
%        PRINT *,
%       *' WARNING: NON-RADIAL SOLAR WIND FLOW SPECIFIED IN ...
   RECALC_08;'
%        PRINT *,
%       *' HENCE XGSW AXIS IS ASSUMED ORIENTED ANTIPARALLEL TO ...
   V_SW VECTOR'
%        PRINT *, ''
%        ISW=2
%        ENDIF
% C
IY=IYEAR;
% C
% C  WE ARE RESTRICTED BY THE INTERVAL 1965-2020, FOR WHICH ...
   EITHER THE IGRF/DGRF COEFFICIENTS OR SECULAR VELOCITIES
% c    ARE KNOWN; IF IYEAR IS OUTSIDE THIS INTERVAL, THEN THE ...
   SUBROUTINE USES THE
% C      NEAREST LIMITING VALUE AND PRINTS A WARNING:
% C
if (IY < 1965),
IY=1965;
% 10   FORMAT(//1X,
%       *'**** RECALC WARNS: YEAR IS OUT OF INTERVAL 1965-2020: ...
   IYEAR=',I4,
%       * /,6X,'CALCULATIONS WILL BE DONE FOR IYEAR=',I4,/)
%        WRITE (*,10) IYEAR,IY
disp(sprintf(['**** RECALC WARNS: YEAR IS OUT OF INTERVAL ...
   1965-2020: IYEAR=%d', ...
'CALCULATIONS WILL BE DONE FOR IYEAR=%d'],IYEAR,IY));
end

if(IY > 2020) ,
IY=2020;
%        WRITE (*,10) IYEAR,IY
disp(sprintf(['**** RECALC WARNS: YEAR IS OUT OF INTERVAL ...
   1965-2020: IYEAR=%d', ...
'CALCULATIONS WILL BE DONE FOR IYEAR=%d'],IYEAR,IY));
end

% C
% C  CALCULATE THE ARRAY REC, CONTAINING COEFFICIENTS FOR THE ...
    RECURSION RELATIONS,
% C  USED IN THE IGRF SUBROUTINE FOR CALCULATING THE ASSOCIATE ...
    LEGENDRE POLYNOMIALS
% C  AND THEIR DERIVATIVES:
% c
for N=1:14,
%       DO 20 N=1,14
N2=2*N-1;
N2=N2*(N2-2);
for M=1:N,
%          DO 20 M=1,N
MN=N*(N-1)/2+M;
GEOPACK2.REC(MN)=((N-M)*(N+M-2))/(N2); % 20
end
end

% C
%        IF (IY.LT.1970) GOTO 50          !INTERPOLATE BETWEEN ...
    1965 - 1970
%        IF (IY.LT.1975) GOTO 60          !INTERPOLATE BETWEEN ...
    1970 - 1975
%        IF (IY.LT.1980) GOTO 70          !INTERPOLATE BETWEEN ...
    1975 - 1980
%        IF (IY.LT.1985) GOTO 80          !INTERPOLATE BETWEEN ...
    1980 - 1985
%        IF (IY.LT.1990) GOTO 90          !INTERPOLATE BETWEEN ...
    1985 - 1990
%        IF (IY.LT.1995) GOTO 100         !INTERPOLATE BETWEEN ...
    1990 - 1995
%        IF (IY.LT.2000) GOTO 110         !INTERPOLATE BETWEEN ...
    1995 - 2000
%        IF (IY.LT.2005) GOTO 120         !INTERPOLATE BETWEEN ...
    2000 - 2005
%        IF (IY.LT.2010) GOTO 130         !INTERPOLATE BETWEEN ...
    2005 - 2010
%        IF (IY.LT.2015) GOTO 140         !INTERPOLATE BETWEEN ...
    2010 - 2015
% C
% C        EXTRAPOLATE BEYOND 2015:
% C
if (IY >= 2015),
DT=(IY)+(IDAY-1)/365.25-2015;
GEOPACK2.G = G15;
GEOPACK2.H = H15;
GEOPACK2.G(1:45) = GEOPACK2.G(1:45)+DG15*DT;
GEOPACK2.H(1:45) = GEOPACK2.H(1:45)+DH15*DT;
%        DO 40 N=1,105
%           G(N)=G15(N)
%           H(N)=H15(N)
%           IF (N.GT.45) GOTO 40
%           G(N)=G(N)+DG15(N)*DT
%           H(N)=H(N)+DH15(N)*DT
% 40     CONTINUE
%        GOTO 300
% C
% C        INTERPOLATE BETWEEEN 1965 - 1970:
% C
elseif (IY < 1970),
F2=((IY)+(IDAY-1)/365.25-1965)/5.;
F1=1.-F2;
GEOPACK2.G = G65*F1+G70*F2;
GEOPACK2.H = H65*F1+H70*F2;
% 50    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1965)/5.
%       F1=1.-F2
%       DO 55 N=1,105
%          G(N)=G65(N)*F1+G70(N)*F2
% 55       H(N)=H65(N)*F1+H70(N)*F2
%       GOTO 300
% C
% C        INTERPOLATE BETWEEN 1970 - 1975:
% C
elseif (IY < 1975),
F2=((IY)+(IDAY-1)/365.25-1970)/5.;
F1=1.-F2;
GEOPACK2.G = G70*F1 + G75*F2;
GEOPACK2.H = H70*F1 + H75*F2;
% 60    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1970)/5.
%       F1=1.-F2
%       DO 65 N=1,105
%          G(N)=G70(N)*F1+G75(N)*F2
% 65       H(N)=H70(N)*F1+H75(N)*F2
%       GOTO 300
% C
% C        INTERPOLATE BETWEEN 1975 - 1980:
% C
elseif (IY < 1980),
F2=((IY)+(IDAY-1)/365.25-1975)/5.;
F1=1.-F2;
GEOPACK2.G = G75*F1+G80*F2;
GEOPACK2.H = H75*F1+H80*F2;
% 70    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1975)/5.
%       F1=1.-F2
%       DO 75 N=1,105
%          G(N)=G75(N)*F1+G80(N)*F2
% 75       H(N)=H75(N)*F1+H80(N)*F2
%       GOTO 300
% C
% C        INTERPOLATE BETWEEN 1980 - 1985:
% C
elseif (IY < 1985),
F2=((IY)+(IDAY-1)/365.25-1980)/5.;
F1=1.-F2;
GEOPACK2.G = G80*F1 + G85*F2;
GEOPACK2.H = H80*F1 + H85*F2;
% 80    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1980)/5.
%       F1=1.-F2
%       DO 85 N=1,105
%          G(N)=G80(N)*F1+G85(N)*F2
% 85       H(N)=H80(N)*F1+H85(N)*F2
%       GOTO 300
% C
% C        INTERPOLATE BETWEEN 1985 - 1990:
% C
elseif (IY < 1990),
F2=((IY)+(IDAY-1)/365.25-1985)/5.;
F1=1.-F2;
GEOPACK2.G = G85*F1+G90*F2;
```

```
GEOPACK2.H = H85*F1+H90*F2;
% 90     F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1985)/5.
%        F1=1.-F2
%        DO 95 N=1,105
%           G(N)=G85(N)*F1+G90(N)*F2
% 95        H(N)=H85(N)*F1+H90(N)*F2
%        GOTO 300
% C
% C        INTERPOLATE BETWEEN 1990 - 1995:
% C
elseif (IY < 1995),
F2=((IY)+(IDAY-1)/365.25-1990)/5.;
F1=1.-F2;
GEOPACK2.G = G90*F1 + G95*F2;
GEOPACK2.H = H90*F1 + H95*F2;
% 100    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1990)/5.
%        F1=1.-F2
%        DO 105 N=1,105
%           G(N)=G90(N)*F1+G95(N)*F2
% 105       H(N)=H90(N)*F1+H95(N)*F2
%        GOTO 300
% C
% C        INTERPOLATE BETWEEN 1995 - 2000:
% C
elseif (IY < 2000),
F2=((IY)+(IDAY-1)/365.25-1995)/5.;
F1=1.-F2;
GEOPACK2.G = G95*F1+G00*F2;
GEOPACK2.H = H95*F1+H00*F2;
% 110    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-1995)/5.
%        F1=1.-F2
%        DO 115 N=1,105    !  THE 2000 COEFFICIENTS (G00) GO ...
    THROUGH THE ORDER 13, NOT 10
%           G(N)=G95(N)*F1+G00(N)*F2
% 115       H(N)=H95(N)*F1+H00(N)*F2
%        GOTO 300
% C
% C        INTERPOLATE BETWEEN 2000 - 2005:
% C
elseif (IY < 2005),
F2=((IY)+(IDAY-1)/365.25-2000)/5.;
F1=1.-F2;
GEOPACK2.G = G00*F1+G05*F2;
GEOPACK2.H = H00*F1+H05*F2;
% 120    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-2000)/5.
%        F1=1.-F2
%        DO 125 N=1,105
%           G(N)=G00(N)*F1+G05(N)*F2
% 125       H(N)=H00(N)*F1+H05(N)*F2
%        GOTO 300
% C
% C        INTERPOLATE BETWEEN 2005 - 2010:
% C
elseif (IY < 2010),
F2=((IY)+(IDAY-1)/365.25-2005)/5.;
F1=1.-F2;
GEOPACK2.G = G05*F1+G10*F2;
GEOPACK2.H = H05*F1+H10*F2;
% 130    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-2005)/5.
%        F1=1.-F2
%        DO 135 N=1,105
%           G(N)=G05(N)*F1+G10(N)*F2
% 135       H(N)=H05(N)*F1+H10(N)*F2
%        GOTO 300
% C
% C        INTERPOLATE BETWEEN 2010 - 2015:
% C
elseif (IY < 2015),
F2=((IY)+(IDAY-1)/365.25-2010)/5.;
F1=1.-F2;
GEOPACK2.G = G10*F1+G15*F2;
GEOPACK2.H = H10*F1+H15*F2;
end
% 140    F2=(FLOAT(IY)+FLOAT(IDAY-1)/365.25-2010)/5.
%        F1=1.-F2
%        DO 145 N=1,105
%           G(N)=G10(N)*F1+G15(N)*F2
% 145       H(N)=H10(N)*F1+H15(N)*F2
%        GOTO 300
% C
% C   COEFFICIENTS FOR A GIVEN YEAR HAVE BEEN CALCULATED; NOW ...
    MULTIPLY
% C   THEM BY SCHMIDT NORMALIZATION FACTORS:
% C
S=1.; % 300
for N=2:14
%       DO 150 N=2,14
MN=N*(N-1)/2+1;
S=S*(2*N-3)/(N-1);
GEOPACK2.G(MN)=GEOPACK2.G(MN)*S;
GEOPACK2.H(MN)=GEOPACK2.H(MN)*S;
P=S;
for M=2:N,
%          DO 150 M=2,N
AA=1.;
if (M == 2), AA=2.; end
P=P*sqrt(AA*(N-M+1)/(N+M-2));
MNN=MN+M-1;
GEOPACK2.G(MNN)=GEOPACK2.G(MNN)*P;
GEOPACK2.H(MNN)=GEOPACK2.H(MNN)*P; % 120
end
end
G10=-GEOPACK2.G(2);
G11= GEOPACK2.G(3);
H11= GEOPACK2.H(3);
% C
% C  NOW CALCULATE GEO COMPONENTS OF THE UNIT VECTOR EzMAG, ...
    PARALLEL TO GEODIPOLE AXIS:
% C   SIN(TETA0)*COS(LAMBDA0), SIN(TETA0)*SIN(LAMBDA0), AND ...
    COS(TETA0)
% C         ST0 * CL0                ST0 * SL0                CT0
% C
SQ=G11^2+H11^2;
SQQ=sqrt(SQ);
SQR=sqrt(G10^2+SQ);
GEOPACK1.SL0=-H11/SQQ;
GEOPACK1.CL0=-G11/SQQ;
GEOPACK1.ST0=SQQ/SQR;
```

```
GEOPACK1.CT0=G10/SQR;
GEOPACK1.STCL=GEOPACK1.ST0*GEOPACK1.CL0;
GEOPACK1.STSL=GEOPACK1.ST0*GEOPACK1.SL0;
GEOPACK1.CTSL=GEOPACK1.CT0*GEOPACK1.SL0;
GEOPACK1.CTCL=GEOPACK1.CT0*GEOPACK1.CL0;
% C
% C  NOW CALCULATE GEI COMPONENTS (S1,S2,S3) OF THE UNIT VECTOR ...
    S = EX_GSE
% C    POINTING FROM THE EARTH'S CENTER TO SUN
% C
%      CALL SUN (IY,IDAY,IHOUR,MIN,ISEC,GST,SLONG,SRASN,SDEC)
[GST,SLONG,SRASN,SDEC] = GEOPACK_SUN (IY,IDAY,IHOUR,MIN,ISEC);
% C
% C  S1,S2, AND S3 ARE THE COMPONENTS OF THE UNIT VECTOR ...
    EXGSM=EXGSE IN THE
% C   SYSTEM GEI POINTING FROM THE EARTH'S CENTER TO THE SUN:
% C
S1=cos(SRASN)*cos(SDEC);
S2=sin(SRASN)*cos(SDEC);
S3=sin(SDEC);
GEOPACK1.CGST=cos(GST);
GEOPACK1.SGST=sin(GST);
% C
% C  DIP1, DIP2, AND DIP3 ARE THE COMPONENTS OF THE UNIT VECTOR ...
    EZSM=EZMAG
% C   IN THE SYSTEM GEI:
% C
DIP1=GEOPACK1.STCL*GEOPACK1.CGST-GEOPACK1.STSL*GEOPACK1.SGST;
DIP2=GEOPACK1.STCL*GEOPACK1.SGST+GEOPACK1.STSL*GEOPACK1.CGST;
DIP3=GEOPACK1.CT0;
% C
% C  NOW CALCULATE THE COMPONENTS OF THE UNIT VECTOR EYGSM IN ...
    THE SYSTEM GEI
% C   BY TAKING THE VECTOR PRODUCT D x S AND NORMALIZING IT TO ...
    UNIT LENGTH:
% C
Y1=DIP2*S3-DIP3*S2;
Y2=DIP3*S1-DIP1*S3;
Y3=DIP1*S2-DIP2*S1;
Y=sqrt(Y1*Y1+Y2*Y2+Y3*Y3);
Y1=Y1/Y;
Y2=Y2/Y;
Y3=Y3/Y;
% C
% C   THEN IN THE GEI SYSTEM THE UNIT VECTOR Z = EZGSM = EXGSM x ...
    EYGSM = S x Y
% C    HAS THE COMPONENTS:
% C
Z1=S2*Y3-S3*Y2;
Z2=S3*Y1-S1*Y3;
Z3=S1*Y2-S2*Y1;
% C
% C    THE VECTOR EZGSE (HERE DZ) IN GEI HAS THE COMPONENTS ...
    (0,-sin(DELTA),
% C      cos(DELTA)) = (0.,-0.397823,0.917462); HERE DELTA = ...
    23.44214 DEG FOR
% C   THE EPOCH 1978 (SEE THE BOOK BY GUREVICH OR OTHER ...
    ASTRONOMICAL HANDBOOKS).
% C    HERE THE MOST ACCURATE TIME-DEPENDENT FORMULA IS USED:
```

```
% C
DJ=(365*(IY-1900)+floor((IY-1901)/4) +IDAY) ...
-0.5+(IHOUR*3600+MIN*60+ISEC)/86400.;
T=DJ/36525.;
OBLIQ=(23.45229-0.0130125*T)/57.2957795;
DZ1=0.;
DZ2=-sin(OBLIQ);
DZ3=cos(OBLIQ);
% C
% C  THEN THE UNIT VECTOR EYGSE IN GEI SYSTEM IS THE VECTOR ...
   PRODUCT DZ x S :
% C
DY1=DZ2*S3-DZ3*S2;
DY2=DZ3*S1-DZ1*S3;
DY3=DZ1*S2-DZ2*S1;
% C
% C   THE ELEMENTS OF THE MATRIX GSE TO GSM ARE THE SCALAR PRODUCTS:
% C   CHI=EM22=(EYGSM,EYGSE), SHI=EM23=(EYGSM,EZGSE), ...
   EM32=(EZGSM,EYGSE)=-EM23,
% C     AND EM33=(EZGSM,EZGSE)=EM22
% C
GEOPACK1.CHI=Y1*DY1+Y2*DY2+Y3*DY3;
GEOPACK1.SHI=Y1*DZ1+Y2*DZ2+Y3*DZ3;
GEOPACK1.HI=asin(GEOPACK1.SHI);
% C
% C    TILT ANGLE: PSI=ARCSIN(DIP,EXGSM)
% C
GEOPACK1.SPS=DIP1*S1+DIP2*S2+DIP3*S3;
GEOPACK1.CPS=sqrt(1.-GEOPACK1.SPS^2);
GEOPACK1.PSI=asin(GEOPACK1.SPS);
% C
% C    THE ELEMENTS OF THE MATRIX MAG TO SM ARE THE SCALAR PRODUCTS:
% C CFI=GM22=(EYSM,EYMAG), SFI=GM23=(EYSM,EXMAG); THEY CAN BE ...
   DERIVED AS FOLLOWS:
% C
% C IN GEO THE VECTORS EXMAG AND EYMAG HAVE THE COMPONENTS ...
   (CT0*CL0,CT0*SL0,-ST0)
% C  AND (-SL0,CL0,0), RESPECTIVELY.    HENCE, IN GEI THE ...
   COMPONENTS ARE:
% C  EXMAG:    CT0*CL0*cos(GST)-CT0*SL0*sin(GST)
% C            CT0*CL0*sin(GST)+CT0*SL0*cos(GST)
% C            -ST0
% C  EYMAG:    -SL0*cos(GST)-CL0*sin(GST)
% C            -SL0*sin(GST)+CL0*cos(GST)
% C             0
% C  THE COMPONENTS OF EYSM IN GEI WERE FOUND ABOVE AS Y1, Y2, ...
   AND Y3;
% C  NOW WE ONLY HAVE TO COMBINE THE QUANTITIES INTO SCALAR ...
   PRODUCTS:
% C
EXMAGX=GEOPACK1.CT0*(GEOPACK1.CL0*GEOPACK1.CGST-GEOPACK1.SL0*GEOPACK1.SGST);
EXMAGY=GEOPACK1.CT0*(GEOPACK1.CL0*GEOPACK1.SGST+GEOPACK1.SL0*GEOPACK1.CGST);
EXMAGZ=-GEOPACK1.ST0;
EYMAGX=-(GEOPACK1.SL0*GEOPACK1.CGST+GEOPACK1.CL0*GEOPACK1.SGST);
EYMAGY=-(GEOPACK1.SL0*GEOPACK1.SGST-GEOPACK1.CL0*GEOPACK1.CGST);
GEOPACK1.CFI=Y1*EYMAGX+Y2*EYMAGY;
GEOPACK1.SFI=Y1*EXMAGX+Y2*EXMAGY+Y3*EXMAGZ;
% C
GEOPACK1.XMUT=(atan2(GEOPACK1.SFI,GEOPACK1.CFI)+3.1415926536)*3.8197186342;
```

```
% C
% C  THE ELEMENTS OF THE MATRIX GEO TO GSM ARE THE SCALAR PRODUCTS:
% C
% C   A11=(EXGEO,EXGSM), A12=(EYGEO,EXGSM), A13=(EZGEO,EXGSM),
% C   A21=(EXGEO,EYGSM), A22=(EYGEO,EYGSM), A23=(EZGEO,EYGSM),
% C   A31=(EXGEO,EZGSM), A32=(EYGEO,EZGSM), A33=(EZGEO,EZGSM),
% C
% C   ALL THE UNIT VECTORS IN BRACKETS ARE ALREADY DEFINED IN GEI:
% C
% C  EXGEO=(CGST,SGST,0), EYGEO=(-SGST,CGST,0), EZGEO=(0,0,1)
% C  EXGSM=(S1,S2,S3),  EYGSM=(Y1,Y2,Y3),   EZGSM=(Z1,Z2,Z3)
% C                                                              ...
   AND  THEREFORE:
% C
GEOPACK1.A11=S1*GEOPACK1.CGST+S2*GEOPACK1.SGST;
GEOPACK1.A12=-S1*GEOPACK1.SGST+S2*GEOPACK1.CGST;
GEOPACK1.A13=S3;
GEOPACK1.A21=Y1*GEOPACK1.CGST+Y2*GEOPACK1.SGST;
GEOPACK1.A22=-Y1*GEOPACK1.SGST+Y2*GEOPACK1.CGST;
GEOPACK1.A23=Y3;
GEOPACK1.A31=Z1*GEOPACK1.CGST+Z2*GEOPACK1.SGST;
GEOPACK1.A32=-Z1*GEOPACK1.SGST+Z2*GEOPACK1.CGST;
GEOPACK1.A33=Z3;
% C
%  10   FORMAT(//1X,
%       *'**** RECALC WARNS: YEAR IS OUT OF INTERVAL 1965-2005: ...
   IYEAR=',I4,
%       * /,6X,'CALCULATIONS WILL BE DONE FOR IYEAR=',I4,/)
%        RETURN
%        END
% end of function RECALC
% c
% ...
   c======================================================================
% C
```

APPENDIX A.7

```
function [GST,SLONG,SRASN,SDEC] = ...
   GEOPACK_SUN(IYEAR,IDAY,IHOUR,MIN,ISEC)
% function [GST,SLONG,SRASN,SDEC] = ...
   GEOPACK_SUN(IYEAR,IDAY,IHOUR,MIN,ISEC)
%       SUBROUTINE SUN ...
    (IYEAR,IDAY,IHOUR,MIN,ISEC,GST,SLONG,SRASN,SDEC)
% C
% C  CALCULATES FOUR QUANTITIES NECESSARY FOR COORDINATE ...
   TRANSFORMATIONS
% C  WHICH DEPEND ON SUN POSITION (AND, HENCE, ON UNIVERSAL TIME ...
   AND SEASON)
% C
% C-------  INPUT PARAMETERS:
% C  IYR,IDAY,IHOUR,MIN,ISEC -  YEAR, DAY, AND UNIVERSAL TIME IN ...
   HOURS, MINUTES,
% C    AND SECONDS  (IDAY=1 CORRESPONDS TO JANUARY 1).
% C
% C-------  OUTPUT PARAMETERS:
```

```
% C  GST - GREENWICH MEAN SIDEREAL TIME, SLONG - LONGITUDE ALONG ...
   ECLIPTIC
% C  SRASN - RIGHT ASCENSION,  SDEC - DECLINATION  OF THE SUN ...
   (RADIANS)
% C  ORIGINAL VERSION OF THIS SUBROUTINE HAS BEEN COMPILED FROM:
% C  RUSSELL, C.T., COSMIC ELECTRODYNAMICS, 1971, V.2, PP.184-196.
% C
% C  LAST MODIFICATION:  MARCH 31, 2003 (ONLY SOME NOTATION CHANGES)
% C
% C     ORIGINAL VERSION WRITTEN BY:    Gilbert D. Mead
% C

%      DOUBLE PRECISION DJ,FDAY
RAD = 57.295779513;
% C
if (IYEAR < 1901) || (IYEAR > 2099), return; end
FDAY=(IHOUR*3600+MIN*60+ISEC)/86400.D0;
DJ=365*(IYEAR-1900)+floor((IYEAR-1901)/4)+IDAY-0.5D0+FDAY;
T=DJ/36525.;
VL=rem(279.696678+0.9856473354*DJ,360.D0);
GST=rem(279.690983+.9856473354*DJ+360.*FDAY+180.,360.D0)/RAD;
G=rem(358.475845+0.985600267*DJ,360.D0)/RAD;
SLONG=(VL+(1.91946-0.004789*T)*sin(G)+0.020094*sin(2.*G))/RAD;
if(SLONG > 6.2831853), SLONG=SLONG-6.2831853; end
if (SLONG < 0.), SLONG=SLONG+6.2831853; end
OBLIQ=(23.45229-0.0130125*T)/RAD;
SOB=sin(OBLIQ);
SLP=SLONG-9.924E-5;
% C
% C   THE LAST CONSTANT IS A CORRECTION FOR THE ANGULAR ...
   ABERRATION  DUE TO
% C   THE ORBITAL MOTION OF THE EARTH
% C
SIND=SOB*sin(SLP);
COSD=sqrt(1.-SIND^2);
SC=SIND/COSD;
SDEC=atan(SC);
SRASN=3.141592654-atan2(cos(OBLIQ)/SOB*SC,-cos(SLP)/COSD);
%       RETURN
%       END
% end of function SUN
% C
% ...
   C=================================================================================
% c
```

APPENDIX A.8

```
% ...
   ------------------------------------------------------------------------------
%
%                            function angl
%
%  this function calculates the angle between two vectors.  the ...
   output is
%    set to 999999.1 to indicate an undefined value.  be sure to ...
   check for
```

```
%    this at the output phase.
%
%  author        : david vallado                  719-573-2600   ...
   27 may 2002
%
%  revisions
%    vallado     - fix tolerances ...
                                 5 sep 2002
%
%  inputs          description                    range / units
%    vec1        - vector number 1
%    vec2        - vector number 2
%
%  outputs       :
%    theta       - angle between the two vectors  -pi to pi
%
%  locals        :
%    temp        - temporary real variable
%
%  coupling      :
%
% [theta] = angl ( vec1,vec2 );
% ...
   ------------------------------------------------------------------------------
   }

function [theta] = angl ( vec1,vec2 )

small     = 0.00000001;
undefined = 999999.1;

magv1 = magnitude(vec1);
magv2 = magnitude(vec2);

if magv1*magv2 > small^2
temp= dot(vec1,vec2) / (magv1*magv2);
if abs( temp ) > 1.0
temp= sign(temp) * 1.0;
end
theta= acos( temp );
else
theta= undefined;
end
```

APPENDIX A.9

```
% ...
   ------------------------------------------------------------------------------
%
%                             function constastro
%
%  this function sets constants for various astrodynamic operations.
%
%  author        : david vallado                  719-573-2600 ...
      2 apr 2007
%
%  revisions
```

```
%
%  inputs        : description                    range / units
%    none
%
%  outputs       :
%    re, flat, omegaearth, mu;
%    eccearth, eccearthsqrd;
%    renm, reft, tusec, tumin, tuday, omegaearthradptu, ...
   omegaearthradpmin;
%    velkmps, velftps, velradpmin;
%    degpsec, radpday;
%    speedoflight, au, earth2moon, moonradius, sunradius;
%
%  locals        :
%                -
%
%  coupling      :
%    none.
%
% constastro;
% ...
   ------------------------------------------------------------------------------
constmath;

% -----------------------  physical constants  ----------------
% WGS-84/EGM-96 constants used here
re         = 6378.137;         % km
flat       = 1.0/298.257223563;
omegaearth = 7.292115e-5;     % rad/s
mu         = 398600.4418;      % km3/s2
mum        = 3.986004418e14;   % m3/s2

% derived constants from the base values
eccearth = sqrt(2.0*flat - flat^2);
eccearthsqrd = eccearth^2;

renm = re / nm2m;
reft = re * 1000.0 / ft2m;

tusec = sqrt(re^3/mu);
tumin = tusec / 60.0;
tuday = tusec / 86400.0;

omegaearthradptu  = omegaearth * tusec;
omegaearthradpmin = omegaearth * 60.0;

velkmps = sqrt(mu / re);
velftps = velkmps * 1000.0/ft2m;
velradpmin = velkmps * 60.0/re;
%for afspc
%velkmps1 = velradpmin*6378.135/60.0   7.90537051051763
%mu1 = velkmps*velkmps*6378.135        3.986003602567418e+005
degpsec = (180.0 / pi) / tusec;
radpday = 2.0 * pi * 1.002737909350795;

speedoflight = 2.99792458e8; % m/s
au = 149597870.0;      % km
earth2moon = 384400.0; % km
moonradius =   1738.0; % km
```

```
sunradius  = 696000.0; % km

masssun   = 1.9891e30;
massearth = 5.9742e24;
massmoon  = 7.3483e22;
```

APPENDIX A.10

```
% ...
   -------------------------------------------------------------------------------
%
%                            function constmath
%
%  this function sets constants for mathematical operations.
%
%  author        : david vallado                  719-573-2600 ...
      2 apr 2007
%
%  revisions
%
%  inputs        : description                    range / units
%    none
%
%  outputs       :
%    rad, twopi, halfpi;
%    ft2m, mile2m, nm2m, mile2ft, mileph2kmph, nmph2kmph;
%
%  locals        :
%                -
%
%  coupling      :
%    none.
%
% constmath;
% ...
   -------------------------------------------------------------------------------

small = 1.0e-10;

infinite  = 999999.9;
undefined = 999999.1;

% -------------------------  mathematical  --------------------
rad    = 180.0 / pi;
twopi  = 2.0 * pi;
halfpi = pi * 0.5;

% -------------------------  conversions  ---------------------
ft2m    =    0.3048;
mile2m  = 1609.344;
nm2m    = 1852;
mile2ft = 5280;
mileph2kmph = 0.44704;
nmph2kmph   = 0.5144444;
```

APPENDIX A.11

```
% ...
  ------------------------------------------------------------------------------
%
%                            function days2mdh
%
%  this function converts the day of the year, days, to the ...
   equivalent month
%    day, hour, minute and second.
%
%  author        : david vallado                  719-573-2600   ...
   22 jun 2002
%
%  revisions
%                -
%
%  inputs          description                    range / units
%    year        - year                           1900 .. 2100
%    days        - julian day of the year         0.0  .. 366.0
%
%  outputs       :
%    mon         - month                          1 .. 12
%    day         - day                            1 .. 28,29,30,31
%    hr          - hour                           0 .. 23
%    minute      - minute                         0 .. 59
%    sec         - second                         0.0 .. 59.999
%
%  locals        :
%    dayofyr     - day of year
%    temp        - temporary extended values
%    inttemp     - temporary integer value
%    i           - index
%    lmonth(12)  - integer array containing the number of days ...
   per month
%
%  coupling      :
%    none.
%
% [mon,day,hr,minute,sec] = days2mdh ( year,days);
% ...
  -----------------------------------------------------------------------------

function [mon,day,hr,minute,sec] = days2mdh ( year,days);

% --------------- set up array of days in month  --------------
for i= 1 : 12
lmonth(i) = 31;
if i == 2
lmonth(i)= 28;
end;
if i == 4 | i == 6 | i == 9 | i == 11
lmonth(i)= 30;
end;
end

dayofyr= floor(days );

% ----------------- find month and day of month ---------------
if rem(year-1900,4) == 0
lmonth(2)= 29;
```

```
end

i= 1;
inttemp= 0;
while ( dayofyr > inttemp + lmonth(i) ) & ( i < 12 )
inttemp= inttemp + lmonth(i);
i= i+1;
end

mon= i;
day= dayofyr - inttemp;

% ----------------- find hours minutes and seconds ------------
temp= (days - dayofyr )*24.0;
hr  = fix( temp );
temp= (temp-hr) * 60.0;
minute = fix( temp );
sec = (temp-minute) * 60.0;
```

APPENDIX A.12

```
% ...
   -----------------------------------------------------------------------------
%
%                              function getgravc
%
%  this function gets constants for the propagator. note that mu ...
   is identified to
%    facilitiate comparisons with newer models.
%
%  author        : david vallado                  719-573-2600   ...
   21 jul 2006
%
%  inputs        :
%    whichconst  - which set of constants to use  721, 72, 84
%
%  outputs       :
%    tumin       - minutes in one time unit
%    mu          - earth gravitational parameter
%    radiusearthkm - radius of the earth in km
%    xke         - reciprocal of tumin
%    j2, j3, j4  - un-normalized zonal harmonic values
%    j3oj2       - j3 divided by j2
%
%  locals        :
%
%  coupling      :
%
%  references    :
%    norad spacetrack report #3
%    vallado, crawford, hujsak, kelso  2006
% [tumin, mu, radiusearthkm, xke, j2, j3, j4, j3oj2] = ...
   getgravc(whichconst);
%  ...
   --------------------------------------------------------------------------- .
   */
```

```
function [tumin, mu, radiusearthkm, xke, j2, j3, j4, j3oj2] = ...
   getgravc(whichconst);

global tumin mu radiusearthkm xke j2 j3 j4 j3oj2
switch whichconst
case 721
% -- wgs-72 low precision str#3 constants --
mu      = 398600.79964;        %// in km3 / s2
radiusearthkm = 6378.135;      %// km
xke     = 0.0743669161;
tumin   = 1.0 / xke;
j2      =   0.001082616;
j3      =  -0.00000253881;
j4      =  -0.00000165597;
j3oj2   =  j3 / j2;
case 72
% ------------ wgs-72 constants ------------
mu      = 398600.8;            %// in km3 / s2
radiusearthkm = 6378.135;      %// km
xke     = 60.0 / sqrt(radiusearthkm*radiusearthkm*radiusearthkm/mu);
tumin   = 1.0 / xke;
j2      =   0.001082616;
j3      =  -0.00000253881;
j4      =  -0.00000165597;
j3oj2   =  j3 / j2;
case 84
% ------------ wgs-84 constants ------------
mu      = 398600.5;            %// in km3 / s2
radiusearthkm = 6378.137;      %// km
xke     = 60.0 / sqrt(radiusearthkm*radiusearthkm*radiusearthkm/mu);
tumin   = 1.0 / xke;
j2      =   0.00108262998905;
j3      =  -0.00000253215306;
j4      =  -0.00000161098761;
j3oj2   =  j3 / j2;
otherwise
fprintf('unknown gravity option (%d)\n',whichconst);
end;  % case
```

APPENDIX A.13

```
% ...
   -----------------------------------------------------------------------------
%
%                            function gstime
%
%  this function finds the greenwich sidereal time (iau-82).
%
%  author        : david vallado                  719-573-2600 ...
     7 jun 2002
%
%  revisions
%                -
%
%  inputs          description                    range / units
%    jdut1       - julian date of ut1             days from 4713 bc
%
```

```
%  outputs       :
%    gst         - greenwich sidereal time        0 to 2pi rad
%
%  locals        :
%    temp        - temporary variable for reals   rad
%    tut1        - julian centuries from the
%                  jan 1, 2000 12 h epoch (ut1)
%
%  coupling      :
%
%  references    :
%    vallado       2007, 193, Eq 3-43
%
% gst = gstime(jdut1);
% ...
   -----------------------------------------------------------------------------

function gst = gstime(jdut1);

twopi      = 2.0*pi;
deg2rad    = pi/180.0;

% ------------------------  implementation   ------------------
tut1= ( jdut1 - 2451545.0 ) / 36525.0;

temp = - 6.2e-6 * tut1 * tut1 * tut1 + 0.093104 * tut1 * tut1  ...
+ (876600.0 * 3600.0 + 8640184.812866) * tut1 + 67310.54841;

% 360/86400 = 1/240, to deg, to rad
temp = rem( temp*deg2rad/240.0,twopi );

% ------------------------ check quadrants --------------------
if ( temp < 0.0 )
temp = temp + twopi;
end

gst = temp;
```

APPENDIX A.14

```
% ...
   -----------------------------------------------------------------------------
%
%                            procedure initl
%
%   this procedure initializes the spg4 propagator. all the ...
   initialization is
%     consolidated here instead of having multiple loops inside ...
   other routines.
%
% Author:
%   Jeff Beck
%   beckja@alumni.lehigh.edu
%   1.0 (aug 7, 2006) - update for paper dav
%   1.1 nov 16, 2007 - update for better compliance
% original comments from Vallado C++ version:
```

```
%   author        : david vallado                  719-573-2600 ...
     28 jun 2005
%
%   inputs        :
%     ecco        - eccentricity                           0.0 - 1.0
%     epoch       - epoch time in days from jan 0, 1950. 0 hr
%     inclo       - inclination of satellite
%     no          - mean motion of satellite
%     satn        - satellite number
%
%   outputs       :
%     ainv        - 1.0 / a
%     ao          - semi major axis
%     con41       -
%     con42       - 1.0 - 5.0 cos(i)
%     cosio       - cosine of inclination
%     cosio2      - cosio squared
%     einv        - 1.0 / e
%     eccsq       - eccentricity squared
%     method      - flag for deep space                    'd', 'n'
%     omeosq      - 1.0 - ecco * ecco
%     posq        - semi-parameter squared
%     rp          - radius of perigee
%     rteosq      - square root of (1.0 - ecco*ecco)
%     sinio       - sine of inclination
%     gsto        - gst at time of observation               rad
%     no          - mean motion of satellite
%
%   locals        :
%     ak          -
%     d1          -
%     del         -
%     adel        -
%     po          -
%
%   coupling      :
%     gstime      - find greenwich sidereal time from the julian ...
    date
%
%   references    :
%     hoots, roehrich, norad spacetrack report #3 1980
%     hoots, norad spacetrack report #6 1986
%     hoots, schumacher and glover 2004
%     vallado, crawford, hujsak, kelso  2006
%  ...
    ----------------------------------------------------------------------------*/

function [  ainv,   ao,     con41,  con42,  cosio,  cosio2, einv,...
eccsq,   method, omeosq, posq,    rp,     rteosq, sinio,...
gsto,    no]...
= initl(  ecco,    epoch,  inclo,  no,     satn)

% /* -------------------- wgs-72 earth constants ...
    ----------------- */
%     // sgp4fix identify constants and allow alternate values
global tumin mu radiusearthkm xke j2 j3 j4 j3oj2
x2o3   = 2.0 / 3.0;
global opsmode
```

```
% /* ------------- calculate auxillary epoch quantities ...
   ---------- */
eccsq  = ecco * ecco;
omeosq = 1.0 - eccsq;
rteosq = sqrt(omeosq);
cosio  = cos(inclo);
cosio2 = cosio * cosio;

% /* ------------------ un-kozai the mean motion ...
   ----------------- */
ak     = (xke / no)^x2o3;
d1     = 0.75 * j2 * (3.0 * cosio2 - 1.0) / (rteosq * omeosq);
del    = d1 / (ak * ak);
adel   = ak * (1.0 - del * del - del *...
(1.0 / 3.0 + 134.0 * del * del / 81.0));
del    = d1/(adel * adel);
no     = no / (1.0 + del);

ao     = (xke / no)^x2o3;
sinio = sin(inclo);
po     = ao * omeosq;
con42 = 1.0 - 5.0 * cosio2;
con41 = -con42-cosio2-cosio2;
ainv   = 1.0 / ao;
einv   = 1.0 / ecco;
posq   = po * po;
rp     = ao * (1.0 - ecco);
method = 'n';

% sgp4fix modern approach to finding sidereal time
if  (opsmode ~= 'a')
gsto = gstime(epoch + 2433281.5);
else
% sgp4fix use old way of finding gst
% count integer number of days from 0 jan 1970
ts70   = epoch - 7305.0;
ids70 = floor(ts70 + 1.0e-8);
tfrac = ts70 - ids70;
% find greenwich location at epoch
c1     = 1.72027916940703639e-2;
thgr70= 1.7321343856509374;
fk5r   = 5.07551419432269442e-15;
twopi = 6.283185307179586;
c1p2p = c1 + twopi;
gsto   = rem( thgr70 + c1*ids70 + c1p2p*tfrac + ts70*ts70*fk5r, ...
   twopi);
end

if ( gsto < 0.0 )
gsto = gsto + twopi;
end;


global idebug dbgfile
if isempty(idebug)
idebug = 0;
elseif idebug
debug5;
end
```

```
return;
```

APPENDIX A.15

```
% ...
   ------------------------------------------------------------------------------
%
%                              function invjday
%
%  this function finds the year, month, day, hour, minute and second
%    given the julian date. tu can be ut1, tdt, tdb, etc.
%
%  author        : david vallado                  719-573-2600   ...
   27 may 2002
%
%  revisions
%                -
%
%  inputs          description                    range / units
%    jd          - julian date                    days from 4713 bc
%
%  outputs       :
%    year        - year                           1900 .. 2100
%    mon         - month                          1 .. 12
%    day         - day                            1 .. 28,29,30,31
%    hr          - hour                           0 .. 23
%    min         - minute                         0 .. 59
%    sec         - second                         0.0 .. 59.999
%
%  locals        :
%    days        - day of year plus fractional
%                  portion of a day               days
%    tu          - julian centuries from 0 h
%                  jan 0, 1900
%    temp        - temporary real values
%    leapyrs     - number of leap years from 1900
%
%  coupling      :
%    days2mdhms  - finds month, day, hour, minute and second ...
   given days and year
%
%  references    :
%    vallado       2007, 208, alg 22, ex 3-13
%
% [year,mon,day,hr,min,sec] = invjday ( jd );
% ...
   -----------------------------------------------------------------------------

function [year,mon,day,hr,min,sec] = invjday ( jd )

% ----------------- find year and days of the year ---------------
temp    = jd-2415019.5;
tu      = temp / 365.25;
year    = 1900 + floor( tu );
leapyrs= floor( ( year-1901 )*0.25 );
```

```
%     days   = temp - ((year-1900)*365.0 + leapyrs ) + ...
   0.00000000001; % nudge by 8.64x10-7 sec to get even outputs
days    = temp - ((year-1900)*365.0 + leapyrs );

% ------------ check for case of beginning of a year -------------
if days < 1.0
year    = year - 1;
leapyrs= floor( ( year-1901 )*0.25 );
days    = temp - ((year-1900)*365.0 + leapyrs );
end

% ------------------- find remaining data  -----------------------
[mon,day,hr,min,sec] = days2mdh( year,days );
%     sec= sec - 0.00000086400;
```

APPENDIX A.16

```
%----------------------------- Begin Function ...
   -----------------------------
%Purpose:
%--------
%Convert a specified Julian Date Vector to Greenwhich Apparent ...
   Sidereal Time.
%
%Inputs:
%-------
%JD              [N x M x L]                         Julian Date ...
   Vector
%
%
%Outputs:
%--------
%GAST            [N x M x L]                         Greenwich ...
   Apparent Sidereal
%                                                    Time in ...
   degrees from
%                                                    0-360
%
%References:
%-----------
%Approximate Sidereal Time,
%http://www.usno.navy.mil/USNO/astronomical-applications/...
%astronomical-information-center/approx-sider-time
%
%Universal Sidereal Times, The Astronomical Almanac For The Year ...
   2004
%
%Programed by:
%-------------
%Darin Koblick 07-17-2010
%Darin Koblick 05-28-2012 Updated Mean obliquity of the ecliptic ...
   terms
%--------------------------------------------------------------------------
function GAST = JD2GAST(JD)
%THETAm is the mean siderial time in degrees
THETAm = JD2GMST(JD);
```

```
%Compute the number of centuries since J2000
T = (JD - 2451545.0)./36525;

%Mean obliquity of the ecliptic (EPSILONm)
% see http://www.cdeagle.com/ccnum/pdf/demogast.pdf equation 3
% also see Vallado, Fundamentals of Astrodynamics and ...
    Applications, second edition.
%pg. 214 EQ 3-53
EPSILONm = 23.439291-0.0130111.*T - 1.64E-07.*(T.^2) + ...
    5.04E-07.*(T.^3);

%Nutations in obliquity and longitude (degrees)
% see http://www.cdeagle.com/ccnum/pdf/demogast.pdf equation 4
L = 280.4665 + 36000.7698.*T;
dL = 218.3165 + 481267.8813.*T;
OMEGA = 125.04452 - 1934.136261.*T;

%Calculate nutations using the following two equations:
% see http://www.cdeagle.com/ccnum/pdf/demogast.pdf equation 5
dPSI = -17.20.*sind(OMEGA) - 1.32.*sind(2.*L) - .23.*sind(2.*dL) ...
+ .21.*sind(2.*OMEGA);
dEPSILON = 9.20.*cosd(OMEGA) + .57.*cosd(2.*L) + ...
    .10.*cosd(2.*dL) - ...
.09.*cosd(2.*OMEGA);

%Convert the units from arc-seconds to degrees
dPSI = dPSI.*(1/3600);
dEPSILON = dEPSILON.*(1/3600);

%(GAST) Greenwhich apparent sidereal time expression in degrees
% see http://www.cdeagle.com/ccnum/pdf/demogast.pdf equation 1
GAST = mod(THETAm + dPSI.*cosd(EPSILONm+dEPSILON),360);
```

APPENDIX A.17

```
%------------------------------ Begin Function ...
    -----------------------------
%Purpose:
%--------
%Convert a specified Julian Date Vector to Greenwhich Mean ...
    Sidereal Time.
%
%Inputs:
%-------
%JD              [N x M x L]                         Julian Date ...
    Vector
%
%
%Outputs:
%--------
%GMST            [N x M x L]                         Greenwich ...
    Mean Sidereal
%                                                    Time in ...
    degrees from
%                                                    0-360
%
%References:
```

```
%-----------
%Approximate Sidereal Time,
%http://www.usno.navy.mil/USNO/astronomical-applications/...
%astronomical-information-center/approx-sider-time
%
%Universal Sidereal Times, The Astronomical Almanac For The Year ...
   2004
%
%Programed by:
%-------------
%Darin Koblick 07-11-2010
%----------------------------------------------------------------------------
function GMST = JD2GMST(JD)
%Find the Julian Date of the previous midnight, JD0
JD0 = NaN(size(JD));
JDmin = floor(JD)-.5;
JDmax = floor(JD)+.5;
JD0(JD > JDmin) = JDmin(JD > JDmin);
JD0(JD > JDmax) = JDmax(JD > JDmax);
H = (JD-JD0).*24;        %Time in hours past previous midnight
D = JD - 2451545.0;      %Compute the number of days since J2000
D0 = JD0 - 2451545.0;    %Compute the number of days since J2000
T = D./36525;            %Compute the number of centuries since J2000
%Calculate GMST in hours (0h to 24h) ... then convert to degrees
GMST = mod(6.697374558 + 0.06570982441908.*D0  + ...
   1.00273790935.*H + ...
0.000026.*(T.^2),24).*15;
%----------------------------End ...
   Function----------------------------------
```

APPENDIX A.18

```
% ...
   ------------------------------------------------------------------------------
%
%                            function jday.m
%
%  this function finds the julian date given the year, month, ...
   day, and time.
%
%  author        : david vallado                  719-573-2600   ...
   27 may 2002
%
%  revisions
%                -
%
%  inputs          description                    range / units
%    year        - year                           1900 .. 2100
%    mon         - month                          1 .. 12
%    day         - day                            1 .. 28,29,30,31
%    hr          - universal time hour            0 .. 23
%    min         - universal time min             0 .. 59
%    sec         - universal time sec             0.0 .. 59.999
%    whichtype   - julian .or. gregorian calender   'j' .or. 'g'
%
%  outputs       :
%    jd          - julian date                    days from 4713 bc
```

```
%
%  locals        :
%    none.
%
%  coupling      :
%    none.
%
%  references    :
%    vallado       2007, 189, alg 14, ex 3-14
%
% jd = jday(yr, mon, day, hr, min, sec)
% ...
   ------------------------------------------------------------------------------

function jd = jday(yr, mon, day, hr, min, sec)

% ------------------------  implementation   ------------------
jd = 367.0 * yr  ...
- floor( (7 * (yr + floor( (mon + 9) / 12.0) ) ) * 0.25 )   ...
+ floor( 275 * mon / 9.0 ) ...
+ day + 1721013.5  ...
+ ( (sec/60.0 + min ) / 60.0 + hr ) / 24.0;
%  - 0.5 * sign(100.0 * yr + mon - 190002.5) + 0.5;
```

APPENDIX A.19

```
% ...
   -------------------------------------------------------------------------------
%
%                              function mag
%
%  this function finds the magnitude of a vector.  the tolerance ...
   is set to
%    0.000001, thus the 1.0e-12 for the squared test of underflows.
%
%  author        : david vallado                  719-573-2600   ...
   30 may 2002
%
%  revisions
%    vallado     - fix tolerance to match coe, eq, etc ...
              3 sep 2002
%
%  inputs          description                    range / units
%    vec         - vector
%
%  outputs       :
%    mag         - magnitude
%
%  locals        :
%    none.
%
%  coupling      :
%    none.
%
% mag = ( vec );
% ...
    --------------------------------------------------------------------------------
    }

function mag = magnitude ( vec )

temp= vec(1)*vec(1) + vec(2)*vec(2) + vec(3)*vec(3);

if abs( temp ) >= 1.0e-16
mag= sqrt( temp );
else
mag= 0.0;
end
```

APPENDIX A.20

```
% ...
    --------------------------------------------------------------------------------
%
%                            function newtonm
%
%  this function performs the newton rhapson iteration to find the
%    eccentric anomaly given the mean anomaly.  the true anomaly ...
    is also
%    calculated.
%
%  author        : david vallado                  719-573-2600 ...
       9 jun 2002
%
%  revisions
%                -
%
%  inputs          description                    range / units
%    ecc         - eccentricity                   0.0  to
%    m           - mean anomaly                   -2pi to 2pi rad
%
%  outputs       :
%    e0          - eccentric anomaly              0.0  to 2pi rad
%    nu          - true anomaly                   0.0  to 2pi rad
%
%  locals        :
%    e1          - eccentric anomaly, next value  rad
%    sinv        - sine of nu
%    cosv        - cosine of nu
%    ktr         - index
%    r1r         - cubic roots - 1 to 3
%    r1i         - imaginary component
%    r2r         -
%    r2i         -
%    r3r         -
%    r3i         -
%    s           - variables for parabolic solution
%    w           - variables for parabolic solution
%
%  coupling      :
%    cubic       - solves a cubic polynomial
%
```

```
%  references    :
%    vallado       2001, 72-75, alg 2, ex 2-1
%
% [e0,nu] = newtonm ( ecc,m );
% ...
   ---------------------------------------------------------------------------

function [e0,nu] = newtonm ( ecc,m );

% -------------------------  implementation   -----------------
numiter =    50;
small   =     0.00000001;
halfpi  = pi * 0.5;

% -------------------------- hyperbolic  ----------------------
if ( (ecc-1.0 ) > small )
% --------------------  initial guess -----------------------
if ( ecc < 1.6  )
if ( ((m<0.0 ) & (m>-pi)) | (m>pi) )
e0= m - ecc;
else
e0= m + ecc;
end
else
if ( (ecc < 3.6 ) & (abs(m) > pi) )
e0= m - sign(m)*ecc;
else
e0= m/(ecc-1.0 );
end
end
ktr= 1;
e1 = e0 + ( (m-ecc*sinh(e0)+e0) / (ecc*cosh(e0) - 1.0 ) );
while ((abs(e1-e0)>small ) & ( ktr<=numiter ))
e0= e1;
e1= e0 + ( ( m - ecc*sinh(e0) + e0 ) / ( ecc*cosh(e0) - 1.0  ) );
ktr = ktr + 1;
end
% ----------------  find true anomaly  --------------------
sinv= -( sqrt( ecc*ecc-1.0  ) * sinh(e1) ) / ( 1.0  - ...
   ecc*cosh(e1) );
cosv= ( cosh(e1) - ecc ) / ( 1.0  - ecc*cosh(e1) );
nu  = atan2( sinv,cosv );
else
% ---------------------- parabolic -------------------------
if ( abs( ecc-1.0  ) < small )
%                 c = [ 1.0/3.0; 0.0; 1.0; -m];
%                 [r1r] = roots (c);
%                 e0= r1r;
s = 0.5  * (halfpi - atan( 1.5 *m ) );
w = atan( tan( s )^(1.0 /3.0 ) );
e0= 2.0 *cot(2.0 *w);
ktr= 1;
nu = 2.0  * atan(e0);
else
% -------------------- elliptical ----------------------
if ( ecc > small )
% -----------  initial guess -------------
if ( ((m < 0.0 ) & (m > -pi)) | (m > pi) )
e0= m - ecc;
```

```
else
e0= m + ecc;
end
ktr= 1;
e1 = e0 + ( m - e0 + ecc*sin(e0) ) / ( 1.0  - ecc*cos(e0) );
while (( abs(e1-e0) > small ) & ( ktr <= numiter ))
ktr = ktr + 1;
e0= e1;
e1= e0 + ( m - e0 + ecc*sin(e0) ) / ( 1.0  - ecc*cos(e0) );
end
% -------------  find true anomaly  ---------------
sinv= ( sqrt( 1.0 -ecc*ecc ) * sin(e1) ) / ( 1.0 -ecc*cos(e1) );
cosv= ( cos(e1)-ecc ) / ( 1.0  - ecc*cos(e1) );
nu  = atan2( sinv,cosv );
else
% -------------------- circular -------------------
ktr= 0;
nu= m;
e0= m;
end
end
end
```

APPENDIX A.21

```
% ...
    ----------------------------------------------------------------------------
%
%                           function newtonnu
%
%  this function solves keplers equation when the true anomaly ...
    is known.
%    the mean and eccentric, parabolic, or hyperbolic anomaly is ...
    also found.
%    the parabolic limit at 168 is arbitrary. the hyperbolic ...
    anomaly is also
%    limited. the hyperbolic sine is used because it's not ...
    double valued.
%
%  author        : david vallado                  719-573-2600   ...
    27 may 2002
%
%  revisions
%    vallado     - fix small                                     ...
    24 sep 2002
%
%  inputs          description                    range / units
%    ecc         - eccentricity                   0.0  to
%    nu          - true anomaly                   -2pi to 2pi rad
%
%  outputs       :
%    e0          - eccentric anomaly              0.0  to 2pi ...
    rad       153.02 deg
%    m           - mean anomaly                   0.0  to 2pi ...
    rad       151.7425 deg
%
%  locals        :
```

```
%    e1           - eccentric anomaly, next value  rad
%    sine         - sine of e
%    cose         - cosine of e
%    ktr          - index
%
%  coupling      :
%    arcsinh      - arc hyperbolic sine
%    sinh         - hyperbolic sine
%
%  references    :
%    vallado       2007, 85, alg 5
%
% [e0,m] = newtonnu ( ecc,nu );
% ...
   -------------------------------------------------------------------------------

function [e0,m] = newtonnu ( ecc,nu );

% ---------------------  implementation   ---------------------
e0= 999999.9;
m = 999999.9;
small = 0.00000001;

% --------------------------- circular ------------------------
if ( abs( ecc ) < small  )
m = nu;
e0= nu;
else
% ---------------------- elliptical -----------------------
if ( ecc < 1.0-small  )
sine= ( sqrt( 1.0 -ecc*ecc ) * sin(nu) ) / ( 1.0 +ecc*cos(nu) );
cose= ( ecc + cos(nu) ) / ( 1.0  + ecc*cos(nu) );
e0  = atan2( sine,cose );
m   = e0 - ecc*sin(e0);
else
% -------------------- hyperbolic  --------------------
if ( ecc > 1.0 + small  )
if (ecc > 1.0 ) & (abs(nu)+0.00001 < pi-acos(1.0 /ecc))
sine= ( sqrt( ecc*ecc-1.0  ) * sin(nu) ) / ( 1.0  + ecc*cos(nu) );
e0  = asinh( sine );
m   = ecc*sinh(e0) - e0;
end
else
% ----------------- parabolic ---------------------
if ( abs(nu) < 168.0*pi/180.0  )
e0= tan( nu*0.5  );
m = e0 + (e0*e0*e0)/3.0;
end
end
end
end

if ( ecc < 1.0  )
m = rem( m,2.0 *pi );
if ( m < 0.0  )
m= m + 2.0 *pi;
end
e0 = rem( e0,2.0 *pi );
end
```

APPENDIX A.22

```
%__________________________________________________|
% SGP4 - Orbit Propagation Function
%
%    mon          - month                          1 .. 12
%    day          - day                            1 .. 28,29,30,31
%    hr           - hour                           0 .. 23
%    minute       - minute                         0 .. 59
%    sec          - second                         0.0 .. 59.999
%
% Inputs:
%                     N
%                     starttime
%                     stoptime
%                     deltamin: time step (minutes)
%__________________________________________________|
%__________________________________________________|
% David Vallado
% https://celestrak.com/software/vallado-sw.asp
% Fundamentals of Astrodynamics and Applications
% Fourth Edition
% Last updated May, 2015
% Also:
% Charles Rino
% http://www.mathworks.com/matlabcentral
%      /fileexchange/28888-satellite-orbit-computation
%      /content/SGP4/sgp4.m
%__________________________________________________|
%__________________________________________________|
% Demet Cilden, Istanbul Technical University, Turkey
% dmtcldn@gmail.com
% July, 2015
%__________________________________________________|

% starttime=[startyear;startdayofyr;mon1;day1;hr1;minute1;sec1];
% stoptime=[stopyear;stopdayofyr;mon2;day2;hr2;minute2;sec2];

function [r_RLL]=Orbit_Prop (N, starttime, stoptime, deltamin)

% sgp4init
global tumin mu radiusearthkm xke j2 j3 j4 j3oj2
global opsmode
opsmode='i';
typerun = 'm';
typeinput = 'd';
whichconst = 84;
rad = 180.0 / pi;

% input 2-line element set file
infilename = 'Sat.TLE';
infile = fopen(infilename, 'r');
if (infile == -1)
fprintf(1,'Failed to open file: %s\n', infilename);
return;
end
```

```
outfile = fopen('tmat.out', 'wt');
global idebug dbgfile
%----------------- propagation -------------------
while (~feof(infile))
longstr1 = fgets(infile, 130);
while ( (longstr1(1) == '#') && (feof(infile) == 0) )
longstr1 = fgets(infile, 130);
end
if (feof(infile) == 0)

longstr2 = fgets(infile, 130);
if idebug
catno = strtrim(longstr1(3:7));
dbgfile = fopen(strcat('sgp4test.dbg.',catno), 'wt');
fprintf(dbgfile,'this is the debug output\n\n' );
end
[satrec, startmfe, stopmfe] = twoline2rv( whichconst, ...
longstr1, longstr2, typerun, typeinput,starttime, ...
    stoptime,deltamin);

fprintf(outfile, '%d xx\n', satrec.satnum);
[satrec, ro ,vo] = sgp4 (satrec,  0.0);

fprintf(outfile, ' %16.8f %16.8f %16.8f %16.8f %12.9f %12.9f ...
    %12.9f\n',...
satrec.t,ro(1),ro(2),ro(3),vo(1),vo(2),vo(3));
tsince = startmfe;
if ( abs(tsince) > 1.0e-8 )
tsince = tsince - deltamin;
end
ffk=0;
while ((tsince < stopmfe) && (satrec.error == 0))
tsince = tsince + deltamin;
ffk=ffk+1;
if(tsince > stopmfe)
tsince = stopmfe;
end
[satrec, ro, vo] = sgp4 (satrec,  tsince);
orbit_r1(ffk,:)=ro;
orbit_v1(ffk,:)=vo;
if (satrec.error > 0)
fprintf(1,'# *** error: t:= %f *** code = %3i\n', tsince, ...
    satrec.error);
end
if (satrec.error == 0)
jd(ffk) = satrec.jdsatepoch + tsince/1440.0;
[year,mon,day,hr,minute,sec] = invjday ( jd );
[p,a,ecc,incl,node,argp,nu,m,arglat,truelon,lonper ] = rv2coe ...
    (ro,vo,mu);
end %// if satrec.error == 0
end %// while propagating the orbit
if (idebug && (dbgfile ~= -1))
fclose(dbgfile);
end
end %// if not eof
end %// while through the input file

position_ECI(:,:)=orbit_r1(1:N,:);   % in ECI Coordinates
velocity_ECI(:,:)=orbit_v1(1:N,:);
```

```
[position_ECEF velocity_ECEF] = ...
    ECItoECEF(jd(1:N),position_ECI',velocity_ECI'); % in ECEF ...
    Coordinates
r_RLL=ecef_lla(1000*position_ECEF'); % latitude(deg), ...
    longitude(deg), altitude (m)
%           r_RLL=ecef2lla(1000*position_ECEF'); % ...
    latitude(deg), longitude(deg), altitude (m)

fclose(infile);
fclose(outfile);
end
```

APPENDIX A.23

```
%


% ...
    ---------------------------------------------------------------------------------
%
%                          function rv2coe
%
%  this function finds the classical orbital elements given the ...
    geocentric
%    equatorial position and velocity vectors.
%
%  author        : david vallado                  719-573-2600   ...
    21 jun 2002
%
%  revisions
%    vallado     - fix special cases ...
                              5 sep 2002
%    vallado     - delete extra check in inclination code        ...
    16 oct 2002
%    vallado     - add constant file use                         ...
    29 jun 2003
%    vallado     - add mu ...
                                            2 apr 2007
%
%  inputs          description                    range / units
%    r           - ijk position vector            km
%    v           - ijk velocity vector            km / s
%    mu          - gravitational parameter        km3 / s2
%
%  outputs       :
%    p           - semilatus rectum               km
%    a           - semimajor axis                 km
%    ecc         - eccentricity
%    incl        - inclination                    0.0  to pi rad
%    omega       - longitude of ascending node    0.0  to 2pi rad
%    argp        - argument of perigee            0.0  to 2pi rad
%    nu          - true anomaly                   0.0  to 2pi rad
%    m           - mean anomaly                   0.0  to 2pi rad
%    arglat      - argument of latitude      (ci) 0.0  to 2pi rad
%    truelon     - true longitude            (ce) 0.0  to 2pi rad
%    lonper      - longitude of periapsis    (ee) 0.0  to 2pi rad
%
```

```
%  locals        :
%    hbar         - angular momentum h vector      km2 / s
%    ebar         - eccentricity     e vector
%    nbar         - line of nodes    n vector
%    c1           - v**2 - u/r
%    rdotv        - r dot v
%    hk           - hk unit vector
%    sme          - specfic mechanical energy      km2 / s2
%    i            - index
%    e            - eccentric, parabolic,
%                   hyperbolic anomaly             rad
%    temp         - temporary variable
%    typeorbit    - type of orbit                  ee, ei, ce, ci
%
%  coupling      :
%    mag          - magnitude of a vector
%    angl         - find the angl between two vectors
%    newtonnu     - find the mean anomaly
%
%  references    :
%    vallado        2007, 121, alg 9, ex 2-5
%
% [p,a,ecc,incl,omega,argp,nu,m,arglat,truelon,lonper ] = rv2coe ...
   (r,v);
% ...
   ------------------------------------------------------------------------------

function [p,a,ecc,incl,omega,argp,nu,m,arglat,truelon,lonper ] = ...
    rv2coe (r,v, mu)

constmath;
constastro;  % don't overwrite mu

% -------------------------  implementation   -----------------
magr= magnitude( r );
magv= magnitude( v );
% ------------------  find h n and e vectors   ----------------
[hbar] = cross( r,v );
magh= magnitude( hbar );
if ( magh > small )
nbar(1)= -hbar(2);
nbar(2)=  hbar(1);
nbar(3)=   0.0;
magn = magnitude( nbar );
c1 = magv*magv - mu /magr;
rdotv= dot( r,v );
for i= 1 : 3
ebar(i)= (c1*r(i) - rdotv*v(i))/mu;
end
ecc = magnitude( ebar );

% ------------  find a e and semi-latus rectum   ----------
sme= ( magv*magv*0.5  ) - ( mu /magr );
if ( abs( sme ) > small )
a= -mu  / (2.0 *sme);
else
a= infinite;
end
p = magh*magh/mu;
```

```
% -----------------  find inclination   -------------------
hk= hbar(3)/magh;
incl= acos( hk );

% --------  determine type of orbit for later use  --------
% ------ elliptical, parabolic, hyperbolic inclined -------
typeorbit= 'ei';
if ( ecc < small )
% ----------------  circular equatorial ---------------
if  (incl<small) | (abs(incl-pi)<small)
typeorbit= 'ce';
else
% --------------  circular inclined ---------------
typeorbit= 'ci';
end
else
% - elliptical, parabolic, hyperbolic equatorial --
if  (incl<small) | (abs(incl-pi)<small)
typeorbit= 'ee';
end
end

% ----------  find longitude of ascending node ------------
if ( magn > small )
temp= nbar(1) / magn;
if ( abs(temp) > 1.0  )
temp= sign(temp);
end
omega= acos( temp );
if ( nbar(2) < 0.0  )
omega= twopi - omega;
end
else
omega= undefined;
end

% ---------------- find argument of perigee ---------------
if ( typeorbit == 'ei' )
argp = angl( nbar,ebar);
if ( ebar(3) < 0.0  )
argp= twopi - argp;
end
else
argp= undefined;
end

% ------------  find true anomaly at epoch    -------------
if ( typeorbit(1:1) == 'e' )
nu =  angl( ebar,r);
if ( rdotv < 0.0  )
nu= twopi - nu;
end
else
nu= undefined;
end

% ----  find argument of latitude - circular inclined -----
if ( typeorbit == 'ci' )
```

```
arglat = angl( nbar,r );
if ( r(3) < 0.0  )
arglat= twopi - arglat;
end
m = arglat;
else
arglat= undefined;
end

% -- find longitude of perigee - elliptical equatorial ----
if  ( ecc>small ) & (typeorbit=='ee')
temp= ebar(1)/ecc;
if ( abs(temp) > 1.0  )
temp= sign(temp);
end
lonper= acos( temp );
if ( ebar(2) < 0.0  )
lonper= twopi - lonper;
end
if ( incl > halfpi )
lonper= twopi - lonper;
end
else
lonper= undefined;
end

% -------- find true longitude - circular equatorial ------
if  ( magr>small ) & ( typeorbit=='ce' )
temp= r(1)/magr;
if ( abs(temp) > 1.0  )
temp= sign(temp);
end
truelon= acos( temp );
if ( r(2) < 0.0  )
truelon= twopi - truelon;
end
if ( incl > halfpi )
truelon= twopi - truelon;
end
m = truelon;
else
truelon= undefined;
end

% ------------ find mean anomaly for all orbits -----------
if ( typeorbit(1:1) == 'e' )
[e,m] = newtonnu(ecc,nu );
end

else
p    = undefined;
a    = undefined;
ecc  = undefined;
incl = undefined;
omega= undefined;
argp = undefined;
nu   = undefined;
m    = undefined;
arglat = undefined;
```

```
truelon= undefined;
lonper = undefined;
end
```

APPENDIX A.24

```
% ...
   -------------------------------------------------------------------------------
%
%                               procedure sgp4
%
%  this procedure is the sgp4 prediction model from space ...
   command. this is an
%    updated and combined version of sgp4 and sdp4, which were ...
   originally
%    published separately in spacetrack report #3. this version ...
   follows the
%    methodology from the aiaa paper (2006) describing the ...
   history and
%    development of the code.
%
% Author:
%   Jeff Beck
%   beckja@alumni.lehigh.edu
%    current :
%               7 may 08  david vallado
%                            update small eccentricity check
%    changes :
%              16 nov 07  david vallado
%                            misc fixes for better compliance
%   1.0 (aug 7, 2006) - update for paper dav
% original comments from Vallado C++ version:
%   author        : david vallado                 719-573-2600 ...
     28 jun 2005
%
%   inputs        :
%     satrec    - initialised structure from sgp4init() call.
%     tsince    - time eince epoch (minutes)
%
%   outputs       :
%     r           - position vector                     km
%     v           - velocity                            km/sec
%     return code - non-zero on error.
%                    1 - mean elements, ecc >= 1.0 or ecc < ...
   -0.001 or a < 0.95 er
%                    2 - mean motion less than 0.0
%                    3 - pert elements, ecc < 0.0  or  ecc > 1.0
%                    4 - semi-latus rectum < 0.0
%                    5 - epoch elements are sub-orbital
%                    6 - satellite has decayed
%
%   locals        :
%     am          -
%     axnl, aynl        -
%     betal       -
%     COSIM    , SINIM    , COSOMM   , SINOMM   , Cnod     , Snod     ...
   , Cos2u    ,
%     Sin2u    , Coseo1   , Sineo1   , Cosi     , Sini     , Cosip    ...
   , Sinip    ,
%     Cosisq   , Cossu    , Sinsu    , Cosu     , Sinu
%     Delm         -
%     Delomg       -
%     Dndt         -
%     Eccm         -
%     EMSQ         -
%     Ecose        -
%     El2          -
%     Eo1          -
%     Eccp         -
%     Esine        -
%     Argpm        -
%     Argpp        -
%     Omgadf       -
%     Pl           -
%     R            -
%     RTEMSQ       -
%     Rdotl        -
%     Rl           -
%     Rvdot        -
%     Rvdotl       -
%     Su           -
%     T2   , T3    , T4      , Tc
%     Tem5, Temp , Temp1 , Temp2   , Tempa   , Tempe   , Templ
%     U    , Ux    , Uy      , Uz      , Vx      , Vy      , Vz
%     inclm        - inclination
%     mm           - mean anomaly
%     nm           - mean motion
%     nodem       - longi of ascending node
%     xinc         -
%     xincp        -
%     xl           -
%     xlm          -
%     mp           -
%     xmdf         -
%     xmx          -
%     xmy          -
%     nodedf      -
%     xnode        -
%     nodep       -
%     np           -
%
%   coupling       :
%     getgravconst
%     dpper
%     dspace
%
%   references     :
%     hoots, roehrich, norad spacetrack report #3 1980
%     hoots, norad spacetrack report #6 1986
%     hoots, schumacher and glover 2004
%     vallado, crawford, hujsak, kelso  2006
%  ...
   ----------------------------------------------------------------------------*/

function [satrec, r, v] = sgp4(satrec,tsince);
```

```
% /* ------------------ set mathematical constants ...
   --------------- */
twopi = 2.0 * pi;
x2o3  = 2.0 / 3.0;
% sgp4fix divisor for divide by zero check on inclination
% the old check used 1.0 + cos(pi-1.0e-9), but then compared it to
% 1.5 e-12, so the threshold was changed to 1.5e-12 for consistancy
temp4    =   1.5e-12;

%  // sgp4fix identify constants and allow alternate values
global tumin mu radiusearthkm xke j2 j3 j4 j3oj2
vkmpersec     = radiusearthkm * xke/60.0;

% /* --------------------- clear sgp4 error flag ...
   ----------------- */
satrec.t     = tsince;
satrec.error = 0;
mrt = 0.0;

% /* ------- update for secular gravity and atmospheric drag ...
   ----- */
xmdf    = satrec.mo + satrec.mdot * satrec.t;
argpdf  = satrec.argpo + satrec.argpdot * satrec.t;
nodedf  = satrec.nodeo + satrec.nodedot * satrec.t;
argpm   = argpdf;
mm      = xmdf;
t2      = satrec.t * satrec.t;
nodem   = nodedf + satrec.nodecf * t2;
tempa   = 1.0 - satrec.cc1 * satrec.t;
tempe   = satrec.bstar * satrec.cc4 * satrec.t;
templ   = satrec.t2cof * t2;

if (satrec.isimp ~= 1)
delomg = satrec.omgcof * satrec.t;
delm   = satrec.xmcof *...
((1.0 + satrec.eta * cos(xmdf))^3 -...
satrec.delmo);
temp   = delomg + delm;
mm     = xmdf + temp;
argpm  = argpdf - temp;
t3     = t2 * satrec.t;
t4     = t3 * satrec.t;
tempa  = tempa - satrec.d2 * t2 - satrec.d3 * t3 -...
satrec.d4 * t4;
tempe  = tempe + satrec.bstar * satrec.cc5 * (sin(mm) -...
satrec.sinmao);
templ  = templ + satrec.t3cof * t3 + t4 * (satrec.t4cof +...
satrec.t * satrec.t5cof);
end

nm    = satrec.no;
em    = satrec.ecco;
inclm = satrec.inclo;
if (satrec.method == 'd')
tc = satrec.t;
[satrec.atime,em,argpm,inclm,satrec.xli,mm,...
satrec.xni,nodem,dndt,nm] = dspace(...
satrec.d2201,satrec.d2211,satrec.d3210,...
satrec.d3222,satrec.d4410,satrec.d4422,...
satrec.d5220,satrec.d5232,satrec.d5421,...
satrec.d5433,satrec.dedt,satrec.del1,...
satrec.del2,satrec.del3,satrec.didt,...
satrec.dmdt,satrec.dnodt,satrec.domdt,...
satrec.irez,satrec.argpo,satrec.argpdot,satrec.t,...
tc,satrec.gsto,satrec.xfact,satrec.xlamo,satrec.no,...
satrec.atime,em,argpm,inclm,satrec.xli,mm,...
satrec.xni,nodem,nm);
end % // if method = d

if (nm <= 0.0)
%        fprintf(1,'# error nm %f\n', nm);
satrec.error = 2;
end
am = (xke / nm)^x2o3 * tempa * tempa;
nm = xke / am^1.5;
em = em - tempe;

% // fix tolerance for error recognition
if ((em >= 1.0) || (em < -0.001) || (am < 0.95))
%        fprintf(1,'# error em %f\n', em);
satrec.error = 1;
end
%   sgp4fix change test condition for eccentricity
if (em < 1.0e-6)
em  = 1.0e-6;
end
mm     = mm + satrec.no * templ;
xlm    = mm + argpm + nodem;
emsq   = em * em;
temp   = 1.0 - emsq;
nodem  = rem(nodem, twopi);
argpm  = rem(argpm, twopi);
xlm    = rem(xlm, twopi);
mm     = rem(xlm - argpm - nodem, twopi);

% /* ----------------- compute extra mean quantities ...
   ------------- */
sinim = sin(inclm);
cosim = cos(inclm);

% /* -------------------- add lunar-solar periodics ...
   -------------- */
ep     = em;
xincp  = inclm;
argpp  = argpm;
nodep  = nodem;
mp     = mm;
sinip  = sinim;
cosip  = cosim;
if (satrec.method == 'd')
[ep,xincp,nodep,argpp,mp] = dpper(...
satrec.e3,satrec.ee2,satrec.peo,...
satrec.pgho,satrec.pho,satrec.pinco,...
satrec.plo,satrec.se2,satrec.se3,...
satrec.sgh2,satrec.sgh3,satrec.sgh4,...
satrec.sh2,satrec.sh3,satrec.si2,...
satrec.si3,satrec.sl2,satrec.sl3,...
satrec.sl4,satrec.t,satrec.xgh2,...
satrec.xgh3,satrec.xgh4,satrec.xh2,...
satrec.xh3,satrec.xi2,satrec.xi3,...
satrec.xl2,satrec.xl3,satrec.xl4,...
satrec.zmol,satrec.zmos,satrec.inclo,...
satrec.init,ep,xincp,nodep,argpp,mp);
if (xincp < 0.0)
xincp  = -xincp;
nodep = nodep + pi;
argpp  = argpp - pi;
end
if ((ep < 0.0 ) || ( ep > 1.0))
%            fprintf(1,'# error ep %f\n', ep);
satrec.error = 3;
end
end % // if method = d

% /* --------------------- long period periodics ...
   ------------------- */
if (satrec.method == 'd')
sinip =  sin(xincp);
cosip =  cos(xincp);
satrec.aycof = -0.5*j3oj2*sinip;
% // sgp4fix for divide by zero with xinco = 180 deg
if (abs(cosip+1.0) > 1.5e-12)
satrec.xlcof = -0.25 * j3oj2 * sinip * (3.0 + 5.0 * cosip) /...
(1.0+cosip);
else
satrec.xlcof = -0.25 * j3oj2 * sinip * (3.0 + 5.0 * cosip) /...
temp4;
end;
end
axnl = ep * cos(argpp);
temp = 1.0 / (am * (1.0 - ep * ep));
aynl = ep* sin(argpp) + temp * satrec.aycof;
xl   = mp + argpp + nodep + temp * satrec.xlcof * axnl;

% /* --------------------- solve kepler's equation ...
   --------------- */
u    = rem(xl - nodep, twopi);
eo1  = u;
tem5 = 9999.9;
ktr = 1;
% //   sgp4fix for kepler iteration
% //   the following iteration needs better limits on corrections
while (( abs(tem5) >= 1.0e-12) && (ktr <= 10) )
sineo1 = sin(eo1);
coseo1 = cos(eo1);
tem5   = 1.0 - coseo1 * axnl - sineo1 * aynl;
tem5   = (u - aynl * coseo1 + axnl * sineo1 - eo1) / tem5;
if(abs(tem5) >= 0.95)
if tem5 > 0.0
tem5 = 0.95;
else
tem5 = -0.95;
end
end
eo1    = eo1 + tem5;
ktr = ktr + 1;
end
```

```
% /* ------------- short period preliminary quantities ...
   ----------- */
ecose = axnl*coseo1 + aynl*sineo1;
esine = axnl*sineo1 - aynl*coseo1;
el2   = axnl*axnl + aynl*aynl;
pl    = am*(1.0-el2);
if (pl < 0.0)
%       fprintf(1,'# error pl %f\n', pl);
satrec.error = 4;
r = [0;0;0];
v = [0;0;0];
else
rl     = am * (1.0 - ecose);
rdotl  = sqrt(am) * esine/rl;
rvdotl = sqrt(pl) / rl;
betal  = sqrt(1.0 - el2);
temp   = esine / (1.0 + betal);
sinu   = am / rl * (sineo1 - aynl - axnl * temp);
cosu   = am / rl * (coseo1 - axnl + aynl * temp);
su     = atan2(sinu, cosu);
sin2u  = (cosu + cosu) * sinu;
cos2u  = 1.0 - 2.0 * sinu * sinu;
temp   = 1.0 / pl;
temp1  = 0.5 * j2 * temp;
temp2  = temp1 * temp;

% /* -------------- update for short period periodics ...
   ------------ */
if (satrec.method == 'd')
cosisq                 = cosip * cosip;
satrec.con41  = 3.0*cosisq - 1.0;
satrec.x1mth2 = 1.0 - cosisq;
satrec.x7thm1 = 7.0*cosisq - 1.0;
end
mrt   = rl * (1.0 - 1.5 * temp2 * betal * satrec.con41) +...
0.5 * temp1 * satrec.x1mth2 * cos2u;
su    = su - 0.25 * temp2 * satrec.x7thm1 * sin2u;
xnode = nodep + 1.5 * temp2 * cosip * sin2u;
xinc  = xincp + 1.5 * temp2 * cosip * sinip * cos2u;
mvt   = rdotl - nm * temp1 * satrec.x1mth2 * sin2u / xke;
rvdot = rvdotl + nm * temp1 * (satrec.x1mth2 * cos2u +...
1.5 * satrec.con41) / xke;

% /* --------------------- orientation vectors ...
   ------------------- */
sinsu =  sin(su);
cossu =  cos(su);
snod  =  sin(xnode);
cnod  =  cos(xnode);
sini  =  sin(xinc);
cosi  =  cos(xinc);
xmx   = -snod * cosi;
xmy   =  cnod * cosi;
ux    =  xmx * sinsu + cnod * cossu;
uy    =  xmy * sinsu + snod * cossu;
uz    =  sini * sinsu;
vx    =  xmx * cossu - cnod * sinsu;
vy    =  xmy * cossu - snod * sinsu;
```

```
vz    =  sini * cossu;

% /* --------- position and velocity (in km and km/sec) ...
   ---------- */
r(1) = (mrt * ux)* radiusearthkm;
r(2) = (mrt * uy)* radiusearthkm;
r(3) = (mrt * uz)* radiusearthkm;
v(1) = (mvt * ux + rvdot * vx) * vkmpersec;
v(2) = (mvt * uy + rvdot * vy) * vkmpersec;
v(3) = (mvt * uz + rvdot * vz) * vkmpersec;
end % // if pl > 0

% // sgp4fix for decaying satellites
if (mrt < 1.0)
%         printf("# decay condition %11.6f \n",mrt);
satrec.error = 6;
end

global idebug dbgfile
if idebug
debug7;
end

return;
```

APPENDIX A.25

```
% ...
   ------------------------------------------------------------------------------
%
%                              procedure sgp4init
%
%   this procedure initializes variables for sgp4.
%
% Author:
%   Jeff Beck
%   beckja@alumni.lehigh.edu
%   1.0 (aug 7, 2006) - update for paper dav
% original comments from Vallado C++ version:
%   author        : david vallado                  719-573-2600 ...
     28 jun 2005
%
%   inputs        :
%     satn        - satellite number
%     bstar       - sgp4 type drag coefficient              kg/m2er
%     ecco        - eccentricity
%     epoch       - epoch time in days from jan 0, 1950. 0 hr
%     argpo       - argument of perigee (output if ds)
%     inclo       - inclination
%     mo          - mean anomaly (output if ds)
%     no          - mean motion
%     nodeo      - right ascension of ascending node
%
%   outputs       :
%     satrec      - common values for subsequent calls
%     return code - non-zero on error.
%                     1 - mean elements, ecc >= 1.0 or ecc < ...
   -0.001 or a < 0.95 er
%                     2 - mean motion less than 0.0
%                     3 - pert elements, ecc < 0.0  or  ecc > 1.0
%                     4 - semi-latus rectum < 0.0
%                     5 - epoch elements are sub-orbital
%                     6 - satellite has decayed
%
%   locals        :
%     CNODM  , SNODM  , COSIM  , SINIM  , COSOMM , SINOMM
%     Cc1sq  , Cc2    , Cc3
%     Coef   , Coef1
%     cosio4      -
%     day         -
%     dndt        -
%     em          - eccentricity
%     emsq        - eccentricity squared
%     eeta        -
%     etasq       -
%     gam         -
%     argpm       - argument of perigee
%     ndem        -
%     inclm       - inclination
%     mm          - mean anomaly
%     nm          - mean motion
%     perige      - perigee
%     pinvsq      -
%     psisq       -
%     qzms24      -
%     rtemsq      -
%     s1, s2, s3, s4, s5, s6, s7        -
%     sfour       -
%     ss1, ss2, ss3, ss4, ss5, ss6, ss7         -
%     sz1, sz2, sz3
%     sz11, sz12, sz13, sz21, sz22, sz23, sz31, sz32, sz33 ...
        -
%     tc          -
%     temp        -
%     temp1, temp2, temp3       -
%     tsi         -
%     xpidot      -
%     xhdot1      -
%     z1, z2, z3          -
%     z11, z12, z13, z21, z22, z23, z31, z32, z33         -
%
%   coupling      :
%     getgravconst
%     initl       -
%     dscom       -
%     dpper       -
%     dsinit      -
%     sgp4        -
%
%   references    :
%     hoots, roehrich, norad spacetrack report #3 1980
%     hoots, norad spacetrack report #6 1986
%     hoots, schumacher and glover 2004
%     vallado, crawford, hujsak, kelso  2006
%  ...
   ----------------------------------------------------------------------------*

function [satrec] = sgp4init(whichconst, satrec, xbstar, xecco, ...
   epoch, ...
xargpo, xinclo, xmo, xno, xnodeo);

% /* ------------------------ initialization ...
   --------------------- */
% /* ----------- set all near earth variables to zero ...
   ------------ */
satrec.isimp   = 0;   satrec.method = 'n'; satrec.aycof    = 0.0;
satrec.con41   = 0.0; satrec.cc1    = 0.0; satrec.cc4      = 0.0;
satrec.cc5     = 0.0; satrec.d2     = 0.0; satrec.d3       = 0.0;
satrec.d4      = 0.0; satrec.delmo  = 0.0; satrec.eta      = 0.0;
satrec.argpdot = 0.0; satrec.omgcof = 0.0; satrec.sinmao   = 0.0;
satrec.t       = 0.0; satrec.t2cof  = 0.0; satrec.t3cof    = 0.0;
satrec.t4cof   = 0.0; satrec.t5cof  = 0.0; satrec.x1mth2   = 0.0;
satrec.x7thm1  = 0.0; satrec.mdot   = 0.0; satrec.nodedot = 0.0;
satrec.xlcof   = 0.0; satrec.xmcof  = 0.0; satrec.nodecf  = 0.0;

% /* ----------- set all deep space variables to zero ...
   ------------ */
satrec.irez  = 0;   satrec.d2201 = 0.0; satrec.d2211 = 0.0;
satrec.d3210 = 0.0; satrec.d3222 = 0.0; satrec.d4410 = 0.0;
satrec.d4422 = 0.0; satrec.d5220 = 0.0; satrec.d5232 = 0.0;
satrec.d5421 = 0.0; satrec.d5433 = 0.0; satrec.dedt  = 0.0;
satrec.del1  = 0.0; satrec.del2  = 0.0; satrec.del3  = 0.0;
satrec.didt  = 0.0; satrec.dmdt  = 0.0; satrec.dnodt = 0.0;
satrec.domdt = 0.0; satrec.e3    = 0.0; satrec.ee2   = 0.0;
satrec.peo   = 0.0; satrec.pgho  = 0.0; satrec.pho   = 0.0;
satrec.pinco = 0.0; satrec.plo   = 0.0; satrec.se2   = 0.0;
satrec.se3   = 0.0; satrec.sgh2  = 0.0; satrec.sgh3  = 0.0;
satrec.sgh4  = 0.0; satrec.sh2   = 0.0; satrec.sh3   = 0.0;
satrec.si2   = 0.0; satrec.si3   = 0.0; satrec.sl2   = 0.0;
satrec.sl3   = 0.0; satrec.sl4   = 0.0; satrec.gsto  = 0.0;
satrec.xfact = 0.0; satrec.xgh2  = 0.0; satrec.xgh3  = 0.0;
satrec.xgh4  = 0.0; satrec.xh2   = 0.0; satrec.xh3   = 0.0;
satrec.xi2   = 0.0; satrec.xi3   = 0.0; satrec.xl2   = 0.0;
satrec.xl3   = 0.0; satrec.xl4   = 0.0; satrec.xlamo = 0.0;
satrec.zmol  = 0.0; satrec.zmos  = 0.0; satrec.atime = 0.0;
satrec.xli   = 0.0; satrec.xni   = 0.0;

% sgp4fix - note the following variables are also passed ...
   directly via satrec.
% it is possible to streamline the sgp4init call by deleting the "x"
% variables, but the user would need to set the satrec.* values ...
   first. we
% include the additional assignment in case twoline2rv is not used.
satrec.bstar      = xbstar;
satrec.ecco       = xecco;
satrec.argpo      = xargpo;
satrec.inclo      = xinclo;
satrec.mo         = xmo;
satrec.no         = xno;
satrec.nodeo      = xnodeo;

%     /* -------------------- wgs-72 earth constants ...
   ----------------- */
```

```
%     // sgp4fix identify constants and allow alternate values
global tumin mu radiusearthkm xke j2 j3 j4 j3oj2
[tumin, mu, radiusearthkm, xke, j2, j3, j4, j3oj2] = getgravc( ...
    whichconst );

ss     = 78.0 / radiusearthkm + 1.0;
qzms2t = ((120.0 - 78.0) / radiusearthkm)^4;
x2o3   =  2.0 / 3.0;
% sgp4fix divisor for divide by zero check on inclination
% the old check used 1.0 + cos(pi-1.0e-9), but then compared it to
% 1.5 e-12, so the threshold was changed to 1.5e-12 for consistancy
temp4    =   1.5e-12;

satrec.init = 'y';
satrec.t    = 0.0;

[ainv,  ao,     satrec.con41,   con42,  cosio,  cosio2, einv,   ...
    eccsq,...
satrec.method,  omeosq, posq,   rp,     rteosq, sinio,...
satrec.gsto,    satrec.no]...
= initl(...
satrec.ecco,    epoch,  satrec.inclo,   satrec.no,...
satrec.satnum);

satrec.error = 0;

% sgp4fix remove this check as it is unnecessary
% the mrt check in sgp4 handles decaying satellite cases even if ...
    the starting
%if (rp < 1.0)
%   printf("# *** satn%d epoch elts sub-orbital ***\n", satn);
%    satrec.error = 5;
%end

if ((omeosq >= 0.0 ) | ( satrec.no >= 0.0))
satrec.isimp = 0;
if (rp < (220.0 / radiusearthkm + 1.0))
satrec.isimp = 1;
end
sfour  = ss;
qzms24 = qzms2t;
perige = (rp - 1.0) * radiusearthkm;

% /* - for perigees below 156 km, s and qoms2t are altered - */
if (perige < 156.0)
sfour = perige - 78.0;
if (perige < 98.0)
sfour = 20.0;
end
qzms24 = ((120.0 - sfour) / radiusearthkm)^4.0;
sfour  = sfour / radiusearthkm + 1.0;
end
pinvsq = 1.0 / posq;

tsi  = 1.0 / (ao - sfour);
satrec.eta   = ao * satrec.ecco * tsi;
etasq = satrec.eta * satrec.eta;
eeta  = satrec.ecco * satrec.eta;
psisq = abs(1.0 - etasq);
```

```
coef  = qzms24 * tsi^4.0;
coef1 = coef / psisq^3.5;
cc2   = coef1 * satrec.no * (ao * (1.0 + 1.5 * etasq + eeta *...
(4.0 + etasq)) + 0.375 * j2 * tsi / psisq * satrec.con41 *...
(8.0 + 3.0 * etasq * (8.0 + etasq)));
satrec.cc1   = satrec.bstar * cc2;
cc3   = 0.0;
if (satrec.ecco > 1.0e-4)
cc3 = -2.0 * coef * tsi * j3oj2 * satrec.no * sinio / satrec.ecco;
end
satrec.x1mth2 = 1.0 - cosio2;
satrec.cc4    = 2.0* satrec.no * coef1 * ao * omeosq *...
(satrec.eta * (2.0 + 0.5 * etasq) + satrec.ecco *...
(0.5 + 2.0 * etasq) - j2 * tsi / (ao * psisq) *...
(-3.0 * satrec.con41 * (1.0 - 2.0 * eeta + etasq *...
(1.5 - 0.5 * eeta)) + 0.75 * satrec.x1mth2 *...
(2.0 * etasq - eeta * (1.0 + etasq)) * cos(2.0 * satrec.argpo)));
satrec.cc5 = 2.0 * coef1 * ao * omeosq * (1.0 + 2.75 *...
(etasq + eeta) + eeta * etasq);
cosio4 = cosio2 * cosio2;
temp1  = 1.5 * j2 * pinvsq * satrec.no;
temp2  = 0.5 * temp1 * j2 * pinvsq;
temp3  = -0.46875 * j4 * pinvsq * pinvsq * satrec.no;
satrec.mdot     = satrec.no + 0.5 * temp1 * rteosq * ...
    satrec.con41 +...
0.0625 * temp2 * rteosq * (13.0 - 78.0 * cosio2 + 137.0 * cosio4);
satrec.argpdot  = -0.5 * temp1 * con42 + 0.0625 * temp2 *...
(7.0 - 114.0 * cosio2 + 395.0 * cosio4) +...
temp3 * (3.0 - 36.0 * cosio2 + 49.0 * cosio4);
xhdot1          = -temp1 * cosio;
satrec.nodedot = xhdot1 + (0.5 * temp2 * (4.0 - 19.0 * cosio2) +...
2.0 * temp3 * (3.0 - 7.0 * cosio2)) * cosio;
xpidot          =  satrec.argpdot+ satrec.nodedot;
satrec.omgcof   = satrec.bstar * cc3 * cos(satrec.argpo);
satrec.xmcof    = 0.0;
if (satrec.ecco > 1.0e-4)
satrec.xmcof = -x2o3 * coef * satrec.bstar / eeta;
end
satrec.nodecf = 3.5 * omeosq * xhdot1 * satrec.cc1;
satrec.t2cof   = 1.5 * satrec.cc1;

% // sgp4fix for divide by zero with xinco = 180 deg
if (abs(cosio+1.0) > 1.5e-12)
satrec.xlcof   = -0.25 * j3oj2 * sinio *...
(3.0 + 5.0 * cosio) / (1.0 + cosio);
else
satrec.xlcof   = -0.25 * j3oj2 * sinio *...
(3.0 + 5.0 * cosio) / temp4;
end
satrec.aycof   = -0.5 * j3oj2 * sinio;
satrec.delmo   = (1.0 + satrec.eta * cos(satrec.mo))^3;
satrec.sinmao  = sin(satrec.mo);
satrec.x7thm1  = 7.0 * cosio2 - 1.0;

% /* --------------- deep space initialization ------------- */
if ((2*pi / satrec.no) >= 225.0)
satrec.method = 'd';
satrec.isimp  = 1;
tc    =  0.0;
```

```
inclm = satrec.inclo;

[sinim,cosim,sinomm,cosomm,snodm,cnodm,day,satrec.e3,satrec.ee2,...
em,emsq,gam,satrec.peo,satrec.pgho,satrec.pho,satrec.pinco,...
satrec.plo,rtemsq,satrec.se2,satrec.se3,satrec.sgh2,...
satrec.sgh3,satrec.sgh4,satrec.sh2,satrec.sh3,satrec.si2,...
satrec.si3,satrec.sl2,satrec.sl3,satrec.sl4,s1,s2,s3,s4,s5,...
s6,s7,ss1,ss2,ss3,ss4,ss5,ss6,ss7,sz1,sz2,sz3,sz11,sz12,...
sz13,sz21,sz22,sz23,sz31,sz32,sz33,satrec.xgh2,satrec.xgh3,...
satrec.xgh4,satrec.xh2,satrec.xh3,satrec.xi2,satrec.xi3,...
satrec.xl2,satrec.xl3,satrec.xl4,nm,z1,z2,z3,z11,z12,z13,...
z21,z22,z23,z31,z32,z33,satrec.zmol,satrec.zmos] = ...
dscom(epoch,satrec.ecco,satrec.argpo,tc,satrec.inclo,...
satrec.nodeo,satrec.no);

[satrec.ecco,satrec.inclo,satrec.nodeo,satrec.argpo,satrec.mo]...
= dpper(satrec.e3,satrec.ee2,satrec.peo,satrec.pgho,...
satrec.pho,satrec.pinco,satrec.plo,satrec.se2,satrec.se3,...
satrec.sgh2,satrec.sgh3,satrec.sgh4,satrec.sh2,satrec.sh3,...
satrec.si2,satrec.si3,satrec.sl2,satrec.sl3,satrec.sl4,...
satrec.t,satrec.xgh2,satrec.xgh3,satrec.xgh4,satrec.xh2,...
satrec.xh3,satrec.xi2,satrec.xi3,satrec.xl2,satrec.xl3,...
satrec.xl4,satrec.zmol,satrec.zmos,inclm,satrec.init,...
satrec.ecco,satrec.inclo,satrec.nodeo,satrec.argpo,satrec.mo);

argpm  = 0.0;
nodem  = 0.0;
mm     = 0.0;

[em,argpm,inclm,mm,nm,nodem,satrec.irez,satrec.atime,...
satrec.d2201,satrec.d2211,satrec.d3210,satrec.d3222,...
satrec.d4410,satrec.d4422,satrec.d5220,satrec.d5232,...
satrec.d5421,satrec.d5433,satrec.dedt,satrec.didt,...
satrec.dmdt,dndt,satrec.dnodt,satrec.domdt,satrec.del1,...
satrec.del2,satrec.del3,...
... %ses,sghl,sghs,sgs,shl,shs,sis,sls,theta,...
satrec.xfact,satrec.xlamo,satrec.xli,satrec.xni] ...
= dsinit(...
cosim,emsq,satrec.argpo,s1,s2,s3,s4,s5,sinim,ss1,ss2,ss3,...
ss4,ss5,sz1,sz3,sz11,sz13,sz21,sz23,sz31,sz33,satrec.t,tc,...
satrec.gsto,satrec.mo,satrec.mdot,satrec.no,satrec.nodeo,...
satrec.nodedot,xpidot,z1,z3,z11,z13,z21,z23,z31,z33,em,...
argpm,inclm,mm,nm,nodem,satrec.ecco,eccsq);
end

% /* ----------- set variables if not deep space ----------- */
if (satrec.isimp ~= 1)
cc1sq          = satrec.cc1 * satrec.cc1;
satrec.d2    = 4.0 * ao * tsi * cc1sq;
temp           = satrec.d2 * tsi * satrec.cc1 / 3.0;
satrec.d3    = (17.0 * ao + sfour) * temp;
satrec.d4    = 0.5 * temp * ao * tsi *...
(221.0 * ao + 31.0 * sfour) * satrec.cc1;
satrec.t3cof = satrec.d2 + 2.0 * cc1sq;
satrec.t4cof = 0.25 * (3.0 * satrec.d3 + satrec.cc1 *...
(12.0 * satrec.d2 + 10.0 * cc1sq));
satrec.t5cof = 0.2 * (3.0 * satrec.d4 +...
12.0 * satrec.cc1 * satrec.d3 +...
6.0 * satrec.d2 * satrec.d2 +...
15.0 * cc1sq * (2.0 * satrec.d2 + cc1sq));
end
end % // if omeosq = 0 ...

% /* finally propogate to zero epoch to initialise all others. */
% sgp4fix take out check to let satellites process until they ...
   are actually below earth surface
%if(satrec.error == 0)
[satrec, r, v] = sgp4(satrec, 0.0);
%end

satrec.init = 'n';

global idebug dbgfile
if idebug
debug6;
end

return;
```

APPENDIX A.26

```
%   "Two Line Elements (TLE)" data upload:


function [raan_d, ap_d, i_d]=TLE_Reader(mu, file_ID)

Line1=fscanf(file_ID,'%d%6d%*c%5d%*3c%2d%f%f%5d%*c%*d%5d%*c%*d%d%5d',[1,10]);
Line2=fscanf(file_ID,'%d%6d%f%f%f%f%f%11f',[1,8]);

fclose(file_ID);

eyear=Line1(4);
eday=Line1(5);
% seconds
time=(eyear*365.25+eday)*24*60*60;
% inclination (deg)
i_d=Line2(3);
% radian
i=i_d/180*pi;
% Right Ascension of the Ascending Node (deg)
raan_d=Line2(4);
% radian
raan=raan_d/180*pi;
% eccentricity
e=Line2(5)/1e+07;
% Argument of Perigee (deg)
ap_d=Line2(6);
% radian
ap=ap_d/180*pi;
% Mean Anomaly (deg)
M_d=Line2(7);
% radian
M=M_d/180*pi;
% Mean Motion (Revs/Day)
rev=Line2(8);
% rad/sec
```

```
n=rev*2*pi/(24*60*60);
% Eccentric Anomaly (rad)
E=eM_E(e,M);
% The semi-major axis (km)
a=(mu/n^2)^(1/3);
% Periapsis (km)
rp=a/(1+e);
% Apoapsis (km)
ra=a/(1-e);
% True Anomaly (rad)
ta=2*atan(((1+e)/(1-e))^0.5*tan(E/2));
% Orbital Period (sec)
T=2*pi/n;
```

APPENDIX A.27

```
%  ...
   ------------------------------------------------------------------------------
%
%                              procedure twoline2rv
%
%  this function converts the two line element set character ...
   string data to
%    variables and initializes the sgp4 variables. several ...
   intermediate varaibles
%    and quantities are determined. note that the result is a ...
   structure so multiple
%    satellites can be processed simultaneously without having ...
   to reinitialize. the
%    verification mode is an important option that permits quick ...
   checks of any
%    changes to the underlying technical theory. this option ...
   works using a
%    modified tle file in which the start, stop, and delta time ...
   values are
%    included at the end of the second line of data. this only ...
   works with the
%    verification mode. the catalog mode simply propagates from ...
   -1440 to 1440 min
%    from epoch and is useful when performing entire catalog runs.
%
% Author:
%   Jeff Beck
%   beckja@alumni.lehigh.edu
%   1.0  aug  6, 2006 - update for paper dav
%   2.0  mar  8, 2007 - misc fixes and manual operation updates
%   2.01 may  9, 2007 - fix for correction to year of 57
%   2.02 oct  8, 2007 - fix for manual jdstart jdstop matlab formats
% original comments from Vallado C++ version:
%   author        : david vallado                  719-573-2600 ...
      1 mar 2001
%
%   inputs        :
%   longstr1      - TLE character string
%   longstr2      - TLE character string
%   typerun       - character for mode of SGP4 execution
%                    'c' = catalog mode (propagates at 20 min ...
   timesteps from
%                            one day before epoch to one day after)
%                    'v' = verification mode (propagates ...
   according to start,
%                            stop, and timestep specified in ...
   longstr2)
%                    'm' = manual mode (prompts user for start, ...
   stop, and
%                            timestep for propagation)
%   typeinput      - type of manual input           mfe 'm', ...
   epoch 'e', dayofyr 'd'
%
%   outputs        :
%     satrec       - structure containing all the sgp4 satellite ...
   information
%
%   coupling       :
%     getgravconst
%     days2mdhms   - conversion of days to month, day, hour, ...
   minute, second
%     jday         - convert day month year hour minute second ...
   into julian date
%     sgp4init     - initialize the sgp4 variables
%
%   references     :
%     norad spacetrack report #3
%     vallado, crawford, hujsak, kelso  2006
%
% [satrec, startmfe, stopmfe, deltamin] = twoline2rv(whichconst, ...
   longstr1, ...
%          longstr2, typerun,typeinput)
%  ...
   ----------------------------------------------------------------------------*

function [satrec, startmfe, stopmfe] = twoline2rv(whichconst, ...
   longstr1, ...
longstr2, typerun,typeinput,starttime, stoptime, deltamin)

global tumin radiusearthkm xke j2 j3 j4 j3oj2

deg2rad  =   pi / 180.0;         %  0.01745329251994330;  % ...
   [deg/rad]
xpdotp   =  1440.0 / (2.0*pi);   % 229.1831180523293;  % ...
   [rev/day]/[rad/min]

revnum = 0;
elnum  = 0;
year   = 0;
satrec.error = 0;

%     // set the implied decimal points since doing a formated read
%     // fixes for bad input data values (missing, ...)
for (j = 11:16)
if (longstr1(j) == ' ')
longstr1(j) = '_';
end
end
```

```
if (longstr1(45) ~= ' ')
longstr1(44) = longstr1(45);
end
longstr1(45) = '.';

if (longstr1(8) == ' ')
longstr1(8) = 'U';
end

if (longstr1(10) == ' ')
longstr1(10) = '.';
end

for (j = 46:50)
if (longstr1(j) == ' ')
longstr1(j) = '0';
end
end
if (longstr1(52) == ' ')
longstr1(52) = '0';
end
if (longstr1(54) ~= ' ')
longstr1(53) = longstr1(54);
end
longstr1(54) = '.';

longstr2(26) = '.';

for (j = 27:33)
if (longstr2(j) == ' ')
longstr2(j) = '0';
end
end

if (longstr1(63) == ' ')
longstr1(63) = '0';
end

if ((length(longstr1) < 68) || (longstr1(68) == ' '))
longstr1(68) = '0';
end

% parse first line
carnumb = str2num(longstr1(1));
satrec.satnum = str2num(longstr1(3:7));
classification = longstr1(8);
intldesg = longstr1(10:17);
satrec.epochyr = str2num(longstr1(19:20));
satrec.epochdays = str2num(longstr1(21:32));
satrec.ndot = str2num(longstr1(34:43));
satrec.nddot = str2num(longstr1(44:50));
nexp = str2num(longstr1(51:52));
satrec.bstar = str2num(longstr1(53:59));
ibexp = str2num(longstr1(60:61));
numb = str2num(longstr1(63));
elnum = str2num(longstr1(65:68));

% parse second line
if (typerun == 'v')
```

```
cardnumb = str2num(longstr2(1));
satrec.satnum = str2num(longstr2(3:7));
satrec.inclo = str2num(longstr2(8:16));
satrec.nodeo = str2num(longstr2(17:25));
satrec.ecco = str2num(longstr2(26:33));
satrec.argpo = str2num(longstr2(34:42));
satrec.mo = str2num(longstr2(43:51));
satrec.no = str2num(longstr2(52:63));
revnum = str2num(longstr2(64:68));
startmfe = str2num(longstr2(70:81));
stopmfe  = str2num(longstr2(83:96));
deltamin = str2num(longstr2(97:105));
else
cardnumb = str2num(longstr2(1));
satrec.satnum = str2num(longstr2(3:7));
satrec.inclo = str2num(longstr2(8:16));
satrec.nodeo = str2num(longstr2(17:25));
satrec.ecco = str2num(longstr2(26:33));
satrec.argpo = str2num(longstr2(34:42));
satrec.mo = str2num(longstr2(43:51));
satrec.no = str2num(longstr2(52:63));
revnum = str2num(longstr2(64:68));
end

%     // ---- find no, ndot, nddot ----
satrec.no    = satrec.no / xpdotp; %//* rad/min
satrec.nddot= satrec.nddot * 10.0^nexp;
satrec.bstar= satrec.bstar * 10.0^ibexp;

%     // ---- convert to sgp4 units ----
satrec.a     = (satrec.no*tumin)^(-2/3);                   % [er]
satrec.ndot = satrec.ndot  / (xpdotp*1440.0);              % [rad/min^2]
satrec.nddot= satrec.nddot / (xpdotp*1440.0*1440);         % [rad/min^3]

%     // ---- find standard orbital elements ----
satrec.inclo = satrec.inclo  * deg2rad;
satrec.nodeo = satrec.nodeo * deg2rad;
satrec.argpo = satrec.argpo  * deg2rad;
satrec.mo     = satrec.mo     *deg2rad;

satrec.alta = satrec.a*(1.0 + satrec.ecco) - 1.0;
satrec.altp = satrec.a*(1.0 - satrec.ecco) - 1.0;

%     // ...
   -----------------------------------------------------------------
%     // find sgp4epoch time of element set
%     // remember that sgp4 uses units of days from 0 jan 1950 ...
   (sgp4epoch)
%     // and minutes from the epoch (time)
%     // ...
   ---------------------------------------------------------------

%     // ------------- temp fix for years from 1957-2056 ...
   ----------------
%     // ------ correct fix will occur when year is 4-digit in ...
   2le ------
if (satrec.epochyr < 57)
year= satrec.epochyr + 2000;
else
```

```
year= satrec.epochyr + 1900;
end;

[mon,day,hr,minute,sec] = days2mdh ( year,satrec.epochdays );
satrec.jdsatepoch = jday( year,mon,day,hr,minute,sec );

%     // input start stop times manually
if ((typerun ~= 'v') && (typerun ~= 'c'))
% ------------- enter start/stop ymd hms values --------------------
%             if (typeinput == 'e')
%                 startyear = input('input start year');
%                 startmon  = input('input start mon');
%                 startday  = input('input start day');
%                 starthr   = input('input start hr');
%                 startmin  = input('input start min');
%                 startsec  = input('input start sec');
%                 jdstart = jday( ...
   startyear,startmon,startday,starthr,startmin,startsec );
%
%                 stopyear = input('input stop year');
%                 stopmon  = input('input stop mon');
%                 stopday  = input('input stop day');
%                 stophr   = input('input stop hr');
%                 stopmin  = input('input stop min');
%                 stopsec  = input('input stop sec');
%                 jdstop = jday( ...
   stopyear,stopmon,stopday,stophr,stopmin,stopsec );
%
%                 startmfe = (jdstart - satrec.jdsatepoch) * 1440.0;
%                 stopmfe  = (jdstop - satrec.jdsatepoch) * 1440.0;
%                 deltamin = input('input time step in minutes ');
%             end;
% -------- enter start/stop year and days of year values -----------
if (typeinput == 'd')
startyear    = starttime(1,1);% input('input start year');
%                 startdayofyr = starttime(2,1);%input('input ...
   start dayofyr');
stopyear     = stoptime(1,1);%input('input stop year');
%                 stopdayofyr  = stoptime(2,1);%input('input stop ...
   dayofyr');

mon1=starttime(3,1); day1=starttime(4,1);hr1=starttime(5,1);
minute1=starttime(6,1);sec1=starttime(7,1);%days2mdh ( ...
   startyear,startdayofyr);
jdstart = jday( startyear,mon1,day1,hr1,minute1,sec1);
mon2=stoptime(3,1); day2=stoptime(4,1);hr2=stoptime(5,1);
minute2=stoptime(6,1);sec2=stoptime(7,1);%days2mdh ( ...
   stopyear,stopdayofyr);
jdstop = jday( stopyear,mon2,day2,hr2,minute2,sec2);

startmfe = (jdstart - satrec.jdsatepoch) * 1440.0;
stopmfe  = (jdstop - satrec.jdsatepoch) * 1440.0;
%                 deltamin = input('input time step in minutes ');
end;
% ------------------ enter start/stop mfe values -------------------
%             if (typeinput == 'm')
%                 startmfe = input('input start mfe: ');
%                 stopmfe  = input('input stop mfe: ');
%                 deltamin = input('input time step in minutes: ');
%             end;
end;
%     // perform complete catalog evaluation
%      if (typerun == 'c')
%          startmfe =  -1440.0;
%          stopmfe  =  1440.0;
%          deltamin = 20.0;
%      end;

%     // ------------- initialize the orbit at sgp4epoch ...
   ---------------
sgp4epoch = satrec.jdsatepoch - 2433281.5; % days since 0 Jan 1950
[satrec] = sgp4init(whichconst, satrec, satrec.bstar, ...
   satrec.ecco, sgp4epoch, ...
satrec.argpo, satrec.inclo, satrec.mo, satrec.no, satrec.nodeo);
```

APPENDIX A.28

```
function [latitude, longitude, altitude] = ecef2geod(x, y, z, tol)

% ECEF2GEOD Convert ECEF coordinates to geodetic coordinates.
%
% Usage: [LATITUDE, LONGITUDE, ALTITUDE] = ECEF2GEOD(X, Y, Z, TOL)
%     or [LATITUDE, LONGITUDE, ALTITUDE] = ECEF2GEOD(XYZ, TOL)
%     or LLA = ECEF2GEOD(X, Y, Z, TOL)
%     or LLA = ECEF2GEOD(XYZ, TOL)
%
% Converts Earth-centered, Earth fixed (ECEF) coordinates X, Y, ...
   and Z to
% geodetic coordinates LATITUDE, LONGITUDE, and ALTITUDE. For a ...
   matrix
% input, the first dimension with length 3 is assumed to have ...
   the three
% separate X, Y, and Z inputs across it. The World Geodetic ...
   System 1984
% (WGS84) ellipsoid model of the Earth is assumed.
%
% Inputs:
%   -X: x coordinates of the point in meters.
%   -Y: y coordinates of the point in meters.
%   -Z: z coordinates of the point in meters.
%   -TOL: Maximum error tolerance in the latitude in radians ...
   (optional,
%   default is 1e-12).
%   -XYZ: Matrix with at least one dimension with length 3, the ...
   first of
%   which corresponding to the dimension across which the three ...
   inputs
%   above go.
%
% Ouputs:
%   -LATITUDE: Geodetic latitude in degrees.
%   -LONGITUDE: Geodetic longitude in degrees.
%   -ALTITUDE: Height above the Earth in meters.
%   -LLA: When just one output is requested, the three outputs ...
   above are
```

```
%   returned as a row vector for scalar inputs, an M-by-3 matrix ...
    for column
%   vector inputs, a 3-by-M matrix for row vector inputs, or the ...
    three
%   outputs concatenated either along the next largest dimension ...
    when the
%   inputs are separate arguments or the same dimension that the ...
    inputs
%   went across when a single matrix is input.
%
% See also: ECEF2LLA, GEOD2ECEF.

% Input checking.
if nargin <= 2
error(nargchk(1, 2, nargin));
if nargin == 1
tol = [];
else
tol = y;
end
sizex = size(x); first3 = find(sizex == 3, 1, 'first');
x = reshape(permute(x, [first3, 1:(first3 - 1), ...
(first3 + 1):ndims(x)]), 3, []);
sizex(first3) = 1;
y = reshape(x(2, :), sizex);
z = reshape(x(3, :), sizex);
x = reshape(x(1, :), sizex);
else
error(nargchk(3, 4, nargin));
end
if nargin <= 3 || isempty(tol)
tol = 1e-12;
end

% WGS84 parameters.
a = 6378137; f = 1/298.257223563; b = a*(1 - f); e2 = 1 - (b/a)^2;

% Longitude is easy:
longitude = atan2(y, x)*180/pi;

% Compute latitude recursively.
rd = hypot(x, y);
[latitude, Nphi] = recur(asin(z ./ hypot(x, hypot(y, z))), z, a, ...
    e2, ...
rd, tol, 1);
sinlat = sin(latitude); coslat = cos(latitude); latitude = ...
    latitude*180/pi;

% Get altitude from latitude.
altitude = rd.*coslat + (z + e2*Nphi.*sinlat).*sinlat - Nphi;

% Shape output according to number of arguments.
if nargout <= 1
if nargin <= 2
latitude = cat(first3, latitude, longitude, altitude);
else
dims = ndims(latitude);
if dims == 2
if size(latitude, 2) == 1
```

```
latitude = [latitude, longitude, altitude];
else
latitude = [latitude; longitude; latitude];
end
else
latitude = cat(dims + 1, latitude, longitude, altitude);
end
end
end

function [latitude, Nphi] = recur(lat_in, z, a, e2, rd, tol, iter)
thisNphi = a ./ sqrt(1 - e2*sin(lat_in).^2);
nextlat = atan((z + thisNphi*e2.*sin(lat_in))./rd);
if all(abs(lat_in - nextlat) < tol) || iter > 100
latitude = nextlat; Nphi = thisNphi;
else
[latitude, Nphi] = recur(nextlat, z, a, e2, rd, tol, iter + 1);
end
```

APPENDIX A.29

```
  function [x_ECI]=ECEF_ECI(x_ECEF,GST)
  x_ECI=zeros(3,1);
  x_ECI(1)=x_ECEF(1)*cos(GST)-x_ECEF(2)*sin(GST);
  x_ECI(2)=x_ECEF(2)*cos(GST)+x_ECEF(1)*sin(GST);
  x_ECI(3)=x_ECEF(3);
  end
```

APPENDIX A.30

```
% ECEF2LLA - convert earth-centered earth-fixed (ECEF)
%            cartesian coordinates to latitude, longitude,
%            and altitude
% [lat,lon,alt] = ecef2lla(x,y,z)
% lat = geodetic latitude (deg)
% lon = longitude (deg)
% alt = height - WGS84 ellipsoid (m)
% P:
% x = ECEF X-coordinate (m)
% y = ECEF Y-coordinate (m)
% z = ECEF Z-coordinate (m)
%
% MATLAB Defalt function

function [lla] = ecef_lla(P)
NN=size(P);
n=NN(1);
for i=1:n
x=P(i,1);
y=P(i,2);
z=P(i,3);
% WGS84 ellipsoid constants:
a = 6378137;
e = 8.1819190842622e-2;
```

```
% calculations:
b   = sqrt(a^2*(1-e^2));
ep  = sqrt((a^2-b^2)/b^2);
p   = sqrt(x.^2+y.^2);
th  = atan2(a*z,b*p);
lon = atan2(y,x);
lat = atan2((z+ep^2.*b.*sin(th).^3),(p-e^2.*a.*cos(th).^3));
N   = a./sqrt(1-e^2.*sin(lat).^2);
alt = p./cos(lat)-N;

% return lon in range [0,2*pi)
lon = mod(lon,2*pi);

k=abs(x)<1 & abs(y)<1;
alt(k) = abs(z(k))-b;
lla(i,1)=lat*180/pi;
lla(i,2)=lon*180/pi;
lla(i,3)=alt;
end
return
```

APPENDIX A.31

```
function r_RLL=ECEF_RLL(x_ECEF)
r_RLL(:,1)=(x_ECEF(:,1).^2+x_ECEF(:,2).^2+x_ECEF(:,3).^2).^0.5;
r_RLL(:,2)=asin(x_ECEF(:,3)./r_RLL(:,1))./pi.*180;
r_RLL(:,3)=x_ECEF(:,2)./abs(x_ECEF(:,2)).*acos(x_ECEF(:,1)./(x_ECEF(:,1).^2+x_ECEF(:,2)
```

APPENDIX A.32

```
  function x_ECEF=ECI_ECEF(x_ECI,wD,t,tEq,tGEQ)
  for i=1:length(x_ECI)
  A=[cos(wD*(t(i)+tEq+tGEQ)),-sin(wD*(t(i)+tEq+tGEQ)),0;
  sin(wD*(t(i)+tEq+tGEQ)),cos(wD*(t(i)+tEq+tGEQ)),0;
  0,0,1];
  x_ECEF(i,:)=x_ECI(i,:)*A';
  end
```

APPENDIX A.33

```
% ---------------------- Begin Code Sequence ...
   -----------------------------%
% Purpose: ...
                                                                  ...
   %
% Convert ECI (CIS, Epoch J2000.0) Coordinates to WGS 84 (CTS, ...
   ECEF)      %
% Coordinates. This function has been vectorized for speed. ...
                  %
```

```
% ...
                                                                                          ...
   %
% Inputs: ...
                                                                                  ...
   %
%------- ...
                                                                                   ...
   %
%JD                      [1 x N]                        Julian ...
   Date Vector
%
%r_ECI                   [3 x N]                        Position ...
   Vector
%                                                       in ECI ...
   coordinate
%                                                       frame of ...
   reference
%
%v_ECI                   [3 x N]                        Velocity ...
   Vector in
%                                                       ECI ...
   coordinate
%                                                       frame of ...
   reference
%
%a_ECI                   [3 x N]                        ...
   Acceleration Vector
%                                                       in ECI ...
   coordinate
%                                                       frame of ...
   reference
%
%
% Outputs:
%--------- ...
                                                                                 ...
   %
%r_ECEF                  [3 x N]                        Position ...
   Vector in
%                                                       ECEF ...
   coordinate
%                                                       frame of ...
   reference
%
%v_ECEF                  [3 x N]                        Velocity ...
   vector in
%                                                       ECEF ...
   coordinate
%                                                       frame of ...
   reference
%
%a_ECEF                  [3 x N]                        ...
   Acceleration Vector
%                                                       in ECEF ...
   coordinate
%                                                       frame of ...
   reference
%
```

```
% References:
%-------------
%Orbital Mechanics with Numerit, ...
   http://www.cdeagle.com/omnum/pdf/csystems.pdf
%
%
% Function Dependencies:
%------------------
% JD2GMST
%------------------------------------------------------------------ ...
         %
% Programed by Darin Koblick  07-05-2010 ...
                                    %
% Modified on 03/01/2012 to add acceleration vector support ...
                 %
%------------------------------------------------------------------ ...
         %
function [r_ECEF v_ECEF] = ECItoECEF(JD,r_ECI,v_ECI)
%Enforce JD to be [N x 1]
JD = JD(:);

%Calculate the Greenwich Apparent Sideral Time (THETA)
%See http://www.cdeagle.com/omnum/pdf/csystems.pdf equation 27
THETA = JD2GAST(JD);

%Average inertial rotation rate of the earth radians per second
omega_e = 7.29211585275553e-005;

%Assemble the transformation matricies to go from ECI to ECEF
%See http://www.cdeagle.com/omnum/pdf/csystems.pdf equation 26
r_ECEF = squeeze(MultiDimMatrixMultiply(T3D(THETA),r_ECI));
v_ECEF = squeeze(MultiDimMatrixMultiply(T3D(THETA),v_ECI) + ...
MultiDimMatrixMultiply(Tdot3D(THETA,omega_e),r_ECI));
% a_ECEF = squeeze(MultiDimMatrixMultiply(T3D(THETA),a_ECI) + ...
%     2.*MultiDimMatrixMultiply(Tdot3D(THETA,omega_e),v_ECI) + ...
%     MultiDimMatrixMultiply(Tddot3D(THETA,omega_e),r_ECI));
%----------------------------- End ...
   Code--------------------------------------%

function C = MultiDimMatrixMultiply(A,B)
C = sum(bsxfun(@times,A,repmat(permute(B',[3 2 1]),[size(A,2) 1 ...
   1])),2);

function T = T3D(THETA)
T = zeros([3 3 length(THETA)]);
T(1,1,:) = cosd(THETA);
T(1,2,:) = sind(THETA);
T(2,1,:) = -T(1,2,:);
T(2,2,:) = T(1,1,:);
T(3,3,:) = ones(size(THETA));

function Tdot = Tdot3D(THETA,omega_e)
Tdot = zeros([3 3 length(THETA)]);
Tdot(1,1,:) = -omega_e.*sind(THETA);
Tdot(1,2,:) = omega_e.*cosd(THETA);
Tdot(2,1,:) = -Tdot(1,2,:);
Tdot(2,2,:) = Tdot(1,1,:);

function Tddot = Tddot3D(THETA,omega_e)
```

```
Tddot = zeros([3 3 length(THETA)]);
Tddot(1,1,:) = -(omega_e.^2).*cosd(THETA);
Tddot(1,2,:) = -(omega_e.^2).*sind(THETA);
Tddot(2,1,:) = -Tddot(1,2,:);
Tddot(2,2,:) =  Tddot(1,1,:);
```

APPENDIX A.34

```
function [x, y, z] = geod2ecef(latitude, longitude, altitude)

% GEOD2ECEF Convert geodetic coordinates to ECEF coordinates.
%
% Usage: [X, Y, Z] = GEOD2ECEF(LATITUDE, LONGITUDE, ALTITUDE)
%     or [X, Y, Z] = GEOD2ECEF(LLA)
%     or XYZ = GEOD2ECEF(LATITUDE, LONGITUDE, ALTITUDE)
%     or XYZ = GEOD2ECEF(LLA)
%
% Converts geodetic coordinates LATITUDE, LONGITUDE, and ...
    ALTITUDE to
% Earth-centered, Earth fixed (ECEF) coordinates X, Y, and Z. ...
    The inputs
% can either be three separate arguments or 1 matrix. For a ...
    matrix input,
% the first dimension with length 3 is assumed to have the three ...
    separate
% LATITUDE, LONGITUDE, and ALTITUDE inputs across it. The World ...
    Geodetic
% System 1984 (WGS84) ellipsoid model of the Earth is assumed.
%
% Inputs:
%   -LATITUDE: Geodetic latitude in degrees.
%   -LONGITUDE: Geodetic longitude in degrees.
%   -ALTITUDE: Height above the Earth in meters.
%   -LLA: Matrix with at least one dimension with length 3, the ...
    first of
%   which corresponding to the dimension across which the three ...
    inputs
%   above go.
%
% Ouputs:
%   -X: x coordinates of the point in meters.
%   -Y: y coordinates of the point in meters.
%   -Z: z coordinates of the point in meters.
%   -XYZ: When just one output is requested, the three outputs ...
    above are
%   returned as a row vector for scalar inputs, an M-by-3 matrix ...
    for column
%   vector inputs, a 3-by-M matrix for row vector inputs, or the ...
    three
%   outputs concatenated either along the next largest dimension ...
    when the
%   inputs are separate arguments or the same dimension that the ...
    inputs
%   went across when a single matrix is input.
%
% See also: LLA2ECEF, ECEF2GEOD.
% Input checking/conversion.
error(nargchk(1, 3, nargin));
if nargin == 1
sizelatitude = size(latitude);
first3 = find(sizelatitude == 3, 1, 'first');
latitude = reshape(permute(latitude, [first3, 1:(first3 - 1), ...
(first3 + 1):ndims(latitude)]), 3, []);
sizelatitude(first3) = 1;
longitude = reshape(latitude(2, :), sizelatitude);
altitude = reshape(latitude(3, :), sizelatitude);
latitude = reshape(latitude(1, :), sizelatitude);
end
latitude = latitude*pi/180; longitude = longitude*pi/180;

% WGS84 parameters.
a = 6378137; f = 1/298.257223563; b = a*(1 - f); e2 = 1 - (b/a)^2;

% Conversion from:
% en.wikipedia.org/wiki/Geodetic_system#Conversion_calculations
Nphi = a ./ sqrt(1 - e2*sin(latitude).^2);
x = (Nphi + altitude).*cos(latitude).*cos(longitude);
y = (Nphi + altitude).*cos(latitude).*sin(longitude);
z = (Nphi.*(1 - e2) + altitude).*sin(latitude);

% Shape output according to number of arguments.
if nargout <= 1
if nargin == 1
x = cat(first3, x, y, z);
else
dims = ndims(x);
if dims == 2
if size(x, 2) == 1
x = [x, y, z];
else
x = [x; y; x];
end
else
x = cat(dims + 1, x, y, z);
end
end
end
```

APPENDIX A.35

```
function [A,B,C] = GEOPACK_SPHCAR (X,Y,Z,J)
% function [A,B,C] = GEOPACK_SPHCAR (X,Y,Z,J)
%      SUBROUTINE SPHCAR (R,THETA,PHI,X,Y,Z,J)
% C
% C   CONVERTS SPHERICAL COORDS INTO CARTESIAN ONES AND VICA VERSA
% C    (THETA AND PHI IN RADIANS).
% C
% C                  J>0            J<0
% C-----INPUT:   J,R,THETA,PHI     J,X,Y,Z
% C----OUTPUT:      X,Y,Z        R,THETA,PHI
% C
% C  NOTE: AT THE POLES (X=0 AND Y=0) WE ASSUME PHI=0 (WHEN ...
   CONVERTING
% C         FROM CARTESIAN TO SPHERICAL COORDS, I.E., FOR J<0)
% C
% C   LAST MOFIFICATION:  APRIL 1, 2003 (ONLY SOME NOTATION ...
    CHANGES AND MORE
% C                              COMMENTS ADDED)
% C
% C   AUTHOR:  N. A. TSYGANENKO
% C
if (J<0),
[A,B,C] = GEOPACK_CAR2SPH(X,Y,Z);
else
[A,B,C] = GEOPACK_SPH2CAR(X,Y,Z);
end

function [R,THETA,PHI] = GEOPACK_CAR2SPH(X,Y,Z)
SQ=X^2+Y^2;
R=sqrt(SQ+Z^2);
if (SQ == 0.),
PHI=0.;
if (Z >= 0.)
THETA=0.;
return
end
THETA=3.141592654; % 1
return
end
SQ=sqrt(SQ); % 2
PHI=atan2(Y,X);
THETA=atan2(SQ,Z);
if (PHI < 0.), PHI=PHI+6.28318531; end

function [X,Y,Z] = GEOPACK_SPH2CAR(R,THETA,PHI)
SQ=R*sin(THETA); %  3
X=SQ*cos(PHI);
Y=SQ*sin(PHI);
Z=R*cos(THETA);

%       RETURN
%       END
% end of function SPHCAR
% C
% ...
    C============================================================================
% c
```

CURRICULUM VITAE

Name Surname: Mehmet Şevket Uludağ

Place and Date of Birth: Diyarbakır, 27.01.1990

E-Mail: uludagm@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2015, Istanbul Technical University, Faculty of Mechanical Engineering, Department of Mechanical Engineering
- **B.Sc.:** 2013, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Department of Electrical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- Research Assistant - Turkish German University since March 2016
- Research Assistant - Istanbul Technical University 2016-2015
- Researcher - ITU Space Systems Design and Test Laboratory since 2010

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- A.R. Aslan, A.Y. Ozyildirim, M. Suer, O. Sarı, M.E. Bas, E. Yakut, M.S. Uludag, B. Karabulut, M.D. Aksulu, E. Erdogan, S. Turkoglu, M. Karataş, C. Cenik, *ASELSAT: High Resolution High Speed Cubesat*, 7th Nanosatellite Symposium, Varna, Bulgaria, 18-23 October 2016
- A.R. Aslan, B. Karabulut, M.S. Uludag, M.E. Bas, E. Yakut, M.D. Aksulu, S. Turkoglu, M. Karataş, C. Cenik, M. Suer, İ. Arslan, K. Arslankoz, *BEAGLESAT and HAVELSAT: Two 2U CubeSat's for QB50*, 7th Nanosatellite Symposium, Varna, Bulgaria, 18-23 October 2016
- A. Rustem Aslan, Mengu Cho, M. Sevket Uludag, Bogac Karabulut, M. Erdem Bas, M. Deniz Aksulu, Mustafa Karatas, Erdinc Yakut, Murat Suer, Baris Dinc; *The Integration and Testing of UBAKUSAT*, 3rd IAA Conference On University Satellite Missions And Cubesat Workshop, Rome, Italy, 30 November – 05 December 2015
- S. Turkoglu, M.S. Uludag, A.R. Aslan; *TVAC Tests and Itegration of QB50 ADCS to BeEagleSAT*, 3rd IAA Conference On University Satellite Missions And Cubesat Workshop, Rome, Italy, 30 November – 05 December 2015

- Sibel Turkoglu, Mehmet Sevket Uludag, Ozcan Kalenderli, Alim Rustem Aslan; *Küp Uydu için Dünya Etrafindaki Manyetik Alan Benzetimi Yapacak Helmholtz Bobini Tasarımı ve Analizi*, Ulusal Havacılık ve Teknolojisi Uygulamaları Kongresi, Izmir, Turkey, 23-24 October 2015 (Turkish) (Desing and Analysis of a Helmholtz Coil for simulating Earth Magnetic Field for Cubesat)

- Emrah Kalemci, Alim Rustem Aslan, Mustafa Erdem Bas, Mehmet Deniz Aksulu, Isa Eray Akyol, Mehmet Sevket Uludag; *Laboratory Performance of X-RAY Detector On 2U CubeSat BeEagleSAT*, 66th International Astronautical Congress, Jerusalem, Israel, IAC-15-B4.2.8, 12-16 October 2015

- M.E. Bas, M.D. Aksulu, I.E. Akyol, M.S. Uludag, A.R Aslan;*An On-Board Computer & a Beacon Modem into a Single Subsystem for CubeSats*, 7th European CubeSat Symposium, Liège, Belgium, 9 – 11 September 2015

- E. Yakut, M. Suer, M.S. Uludag, E. Bas, S. Turkoglu, A.R. Aslan, A. Hacıoğlu, M. Çelebi, M.E. Aydemir, S. Basturk, A. Telli, S. Gokcebag; *QB50 - BeEagleSat Inner - Outer Design Details and ADCS Testing-Integration*, 7th European CubeSat Symposium, Liège, Belgium, 9 – 11 September 2015

- A. Rustem Aslan, Mansur Celebi, Ahmet Sofyalı Sibel Türkoğlu, M. Sevket Uludag, I. Eray Akyol, M. Deniz Aksulu, Erdinc Yakut, Murat Süer, Serhan Gökçebağ, M. Erdem Bas, *The Integration and Testing of BeEagleSat*, 30th International Symposium on Space Technology and Science, Kobe-Hyogo Japan, (30th ISTS)(6th NSAT) , 4-10 July 2015

- M S Uludag, M E Bas, M O Gulbahce, D A Kocabaş, A R Aslan *Thermal Vacuum Chamber Test of a DC-DC Converter*, 7th International Conference Series on Recent Advances in Space Technologies, Istanbul, Turkey (7th RAST), 16-19 June 2015

- M.E.Bas, M.S.Uludag, I.E.Akyol, M.D.Aksulu, M.Karatas, A.R.Aslan *Implementation of an On-Board Computer & a Modem into a Ssingle Subsystem for CubeSat*, 6th European CubeSat Symposium, Estavayer-le-Lac, Switzerland, 14-16 October 2014

- Alim Rustem Aslan, Emrah Kalemci, Mustafa Erdem Bas, Isa Eray Akyol, Mehmet Sevket Uludag, Mehmet Deniz Aksulu, Ertan Umit *Development And In Orbıt Testıng Of An X Ray Detector Wıthın A 2u Cubesat* , 65th International Astronautical Congress, Toronto, Canada, 29 September– 3 October 2014; IAC-14-B4.2.9

- Aslan,A.R., Umit,E., Şimsek,M., Uludag,M.S., Aksulu,M.D., Sofyalı,A., Suer,M., Ilarslan,M., Bas,M.E., Akyol,I.E., Kalemci,E.,*QB50 Projesi Kapsamında Beeaglesat Küp Uydusunun Geliştirilmesi*, V. Ulusal Havacılık Konferansı, Erciyes Üniversitesi, Kayseri, Turkey, 2014, UHUK-2014-156 (Turkish) (Development of Beeaglesat Cubesat for QB50 Project)

- Aslan,R., Yagci,B., Umit,E., Bas,M.E., Uludag,M.S., Ozen,O.E., Suer,M., Sofyali,A., Yarim,C., *LESSONS LEARNED DEVELOPING A 3U COMMUNICATION CUBESAT*,64th International Astronautical Congress, Beijing, China,2013, IAC-13,D1,5.6x19780

- Aslan,R., Yagci,H.B., Umit,M.E., Sofyalı,A., Bas,M.E., Uludag,M.S., Ozen,O.E., Aksulu,M.D., Yakut,E., Oran,C., Suer,M., Akyol,I.E., Ecevit,A.B., Ersoz,M.S., Öz,I., Gulgonul,S., Dinc,B., Dengiz,T.,*Development of a LEO Communication CubeSat*, 6th International Conference Series on Recent Advances in Space Technologies-RAST, 06/2013

- Umit, M.E., Baş, M.E.,Akyol, I.E., Uludag, M.S., Ecevit, A.B., Aslan,A.R., *Indigenous Hardware-in-the-Loop Simulator Development for University Satellites*, IAA-B9-0510P, 9th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 12/2013 [POSTER]

- Umit, M.E., Baş,M.E., Akyol,I.E., Uludag,M.S., Ecevit,A.B., Aslan,A.R., *Solar Emulator and Simulator Design for CUBESATS*, IAC-12-C3, 4, 9, x15321, 63rd International Astronautical Congress, Napoli, Italy, 2012.

- Umit, M.E., Baş,M.E.,Akyol,I.E.,Uludag, M.S.,Ecevit,A.B., Aslan,A.R., *TURKSAT 3USAT Küp Uydusu için Elektrik Güç Sistemi Tasarlanması*, IV.Ulusal Havacılık Konferansı, Hava Harp Okulu, Istanbul, Turkiye, 2012, UHUK*2012-076 (Turkish) (Design of an Electrical Power System for TURKSAT 3USAT Cubesat)