KARAKTERISTIK DATA *EPHEMERIS* GERHANA MATAHARI BERBASIS *JET PROPULSION LABORATORY* NASA

TESIS
Diajukan untuk Memenuhi Sebagian Syarat
guna Memperoleh Gelar Magister
dalam Ilmu Falak

oleh:
M. BASTHONI
NIM: 1500028006

PROGRAM STUDI MAGISTER ILMU FALAK
FAKULTAS SYARI'AH DAN HUKUM
UIN WALISONGO SEMARANG
2017

KEMENTERIAN AGAMA REPUBLIK INDONESIA
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SYARI'AH DAN HUKUM
Jalan Prof. Dr. H. Hamka Semarang 50185
Telepon (024) 7601291, Faxsimili (024) 7624691 Website: http://fs.walisongo.ac.id/

PENGESAHAN TESIS

Tesis yang ditulis oleh:

Nama lengkap : **M. Basthoni**
NIM : 1500028006
Judul Penelitian : **Karakteristik Data *Ephemeris* Gerhana Matahari Berbasis *Jet Propulsion Laboratory* NASA.**

telah dilakukan revisi sesuai saran dalam Sidang Ujian Tesis pada tanggal **22 Juni 2017** dan layak dijadikan syarat memperoleh Gelar Magister dalam bidang Ilmu Falak.
Disahkan oleh:

Nama lengkap & Jabatan	tanggal	Tanda tangan
Dr. Achmad Arief Budiman, M.Ag. Ketua Sidang/Penguji	3-7-17	
Dr. H. Ali Imron, M.Ag. Sekretaris Sidang/Penguji	4-7-17	
Dr. Rupi'i Amri, M.Ag. Penguji 1	4-7-17	
Dr. H. Abdul Ghofur, M.Ag. Penguji 2	3/7-2017	

NOTA DINAS

Semarang, 8 Juni 2017

Kepada
Yth. Dekan Fakultas Syari'ah dan Hukum
UIN Walisongo
di Semarang

Assalamu 'alaikum wr. wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi terhadap tesis yang ditulis oleh:

Nama	:	**M. Basthoni**
NIM	:	1500028006
Konsentrasi	:	-
Program Studi	:	Magister Ilmu Falak
Judul	:	**Karakteristik Data *Ephemeris* Gerhana Matahari Berbasis *Jet Propulsion Laboratory* NASA**

Kami memandang bahwa tesis tersebut sudah dapat diajukan kepada Fakultas Syari'ah dan Hukum UIN Walisongo untuk diujikan dalam Sidang Ujian Tesis.

Wassalamu 'alaikum wr. wb.

Pembimbing,

Dr. H. Ahmad Izzuddin, M.Ag.
NIP: 197205121999031003

NOTA DINAS

Semarang, 8 Juni 2017

Kepada
Yth. Dekan Fakultas Syari'ah dan Hukum
UIN Walisongo
di Semarang

Assalamu 'alaikum wr. wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi terhadap tesis yang ditulis oleh:

Nama	:	**M. Basthoni**
NIM	:	1500028006
Konsentrasi	:	-
Program Studi	:	Magister Ilmu Falak
Judul	:	**Karakteristik Data *Ephemeris* Gerhana Matahari Berbasis *Jet Propulsion Laboratory* NASA**

Kami memandang bahwa tesis tersebut sudah dapat diajukan kepada Fakultas Syari'ah dan Hukum UIN Walisongo untuk diujikan dalam Sidang Ujian Tesis.

Wassalamu 'alaikum wr. wb.

Pembimbing,

Dr. H. Ali Imron, M.Ag.
NIP: 197307302003121003

PERNYATAAN KEASLIAN TESIS

Yang bertanda tangan di bawah ini:

Nama lengkap	:	**M. Basthoni**
NIM	:	1500028006
Judul Penelitian	:	**Karakteristik Data *Ephemeris* Gerhana Matahari Berbasis *Jet Propulsion Laboratory* NASA**
Program Studi	:	Magister Ilmu Falak
Konsentrasi	:	-

menyatakan bahwa makalah tesis yang berjudul:

KARAKTERISTIK DATA *EPHEMERIS* GERHANA MATAHARI BERBASIS *JET PROPULSION LABORATORY* NASA

secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 08 Juni 2017

Pembuat Pernyataan,

M. Basthoni
NIM: 1500028006

Abstract

Title : **Characteristics of Ephemeris for Solar Eclipse Based on *Jet Propulsion Laboratory* NASA**
Author : M. Basthoni
NIM : 1500028006

High-precision ephemeris data of the Sun and the Moon are required for calculations in astronomy, such as the timing of the solar eclipse prayer precisely according to the shari'a guidance. Because the time is closely related to the regurality of the position of the Sun and the Moon movement. NASA through the Jet Propulsion Laboratory (JPL), has been releasing several series of Development Ephemeris (DE) to support various astronomical missions. DE200 which was released in 1981 became the base of VSOP82 theory and developed into VSOP87 and also became the basis of ELP2000 theory. JPL also released DE405 (1945) which then used as a database to release the prediction of the solar eclipse in the 2011-2020 timespan, while before 2011 and after 2020, NASA used the theory of VSOP87 and ELP2000 based on the DE200. Why does NASA still use the DE200 (which is implemented in VSOP87 / ELP2000) while the more updated series DE405 has been released? This study is to answer the problem: (1) How is the algorithm of calculation of ephemeris data and solar eclipse based on JPL NASA? (2) How is NASA's JPL-based ephemeris accuracy characteristics as well as its implementation for solar eclipse calculations? These problems are discussed through a comparative study of the accuracy between DE200, DE405, DE406 and DE421 series. The DE series is used as a data source for obtaining ephemeris data that will be implemented in the solar eclipse calculations and the results will be compared with the observed solar eclipse observations made by Fred Espenak, *LangitSelatan* and BMKG.

This study shows that: (1) NASA JPL-based ephemeris data calculation algorithm in principle through 2 (two) major steps is converting ASCII data block files into binary files that can

generate ephemeris data of the Sun and the Moon. While the total solar eclipse algorithm or ring is if $\Delta\lambda$ <= sum_SD and occurs before the greatest eclipse, then it is the first contact time, if $\Delta\lambda$ >= sum_SD and occurs after the peak time of the eclipse, then it is the last contact time . while the greatest eclipse occurs when the value of the smallest elongation angle within the estimated time frame of the solar eclipse. There are similar characteristics between the DE series that are one with the other, ie at 0.01 second or lower intervals all DE series tend to be stable at certain difference values. The spike in difference occurs at 0.1 second ephemeris data interval. While at the beginning of the interval (1 second) there are two existing patterns that all intervals have the same difference and the second pattern between DE200 and the other DE series differs considerably from the difference in accuracy. The DE405 accuracy character is always the same as DE406 on all the solar eclipse intervals and events tested in this study. While the accuracy of the DE200 series at intervals after 1 second (less than 1 second) to predict the total solar eclipse time or the ring in a particular city is always better than the other DE series then followed by DE421 and the last is the DE405 and DE406 series. This study can be used to predict the total or annular solar eclipse in a particular city, so that the implementation of solar eclipse prayer can be timely in accordance with the guidance of sharia

Keywords: ephemeris characteristics, solar eclipse,
JPL NASA

Abstrak

Judul : **Karakteristik Data *Ephemeris* Gerhana Matahari Berbasis *Jet Propulsion Laboratory* NASA**
Penulis : M. Basthoni
NIM : 1500028006

Data *ephemeris* Matahari dan Bulan dengan ketelitian tinggi dibutuhkan untuk perhitungan dalam ilmu falak, misalnya penentuan waktu shalat gerhana Matahari dengan tepat sesuai dengan tuntunan syariat yang waktu pelaksanaannya sangat terkait dengan reguralitas posisi pergerakan dua benda langit tersebut. NASA melalui salah satu divisinya, *Jet Propulsion Laboratory* (JPL), selama ini telah merilis beberapa seri *Development Ephemeris* (DE) untuk mendukung berbagai misi astronomis. Seri DE200 (1981) yang menjadi basis munculnya teori VSOP82 dan dikembangkan menjadi VSOP87 dan juga menjadi basis munculnya teori ELP2000. JPL juga merilis DE405 (1945) yang kemudian dijadikan basis data untuk merilis prediksi gerhana Matahari pada rentang waktu 2011 – 2020. Sementara sebelum tahun 2011 dan setelah 2020, NASA menggunakan teori VSOP87 dan ELP2000 yang berbasis pada DE200. Mengapa NASA tetap menggunakan DE200 (yang diimplementasikan dalam VSOP87/ELP2000) sementara seri yang lebih update yaitu DE405 sudah dirilis? Studi ini dimaksudkan untuk menjawab permasalahan: (1) Bagaimana algoritma perhitungan data ephemeris dan gerhana Matahari berbasis JPL NASA? (2) Bagaimana karakteristik akurasi data ephemeris berbasis JPL NASA dalam implementasinya untuk perhitungan gerhana Matahari? Permasalahan tersebut dibahas melalui studi perbandingan akurasi antara beberapa seri data DE yaitu DE200, DE405, DE406 dan DE421. Sumber data DE tersebut dijadikan sumber data untuk mendapatkan data *ephemeris* yang akan diimplementasikan dalam perhitungan gerhana Matahari dan

hasilnya akan dibandingkan dengan laporan observasi gerhana Matahari yang telah dilakukan oleh Fred Espenak.

Kajian ini menjunjukkan bahwa: (1) Algoritma perhitungan data ephemeris berbasis JPL NASA pada prinsipnya melalui 2 (dua) langkah utama yaitu file data blok ASCII dikonversi menjadi file *binary* kemudian baru bisa diperoleh data *ephemeris* Matahari dan Bulan. Sedangkan algoritma perhitungan gerhana Matahari total atau cincin yaitu jika $\Delta\lambda$ <= sum_SD dan terjadi sebelum waktu puncak gerhana, maka saat itu adalah waktu kontak pertama, jika $\Delta\lambda$ >= sum_SD dan terjadi setelah waktu puncak gerhana, maka saat itu adalah waktu kontak terakhir. Sedangkan puncak gerhana terjadi ketika nilai sudut elongasi paling kecil dalam rentang waktu perkiraan terjadinya gerhana Matahari. (2) Ada kesamaan karakteristik antara seri DE yang satu dengan yang lain, yaitu pada interval 0,01 detik atau lebih kecil semua seri DE cenderung stabil pada nilai selisih tertentu. Lonjakan selisih akurasi terjadi pada interval data *ephemeris* 0,1 detik. Sedangkan di awal interval (1 detik) ada dua pola yang ada yaitu semua interval mempunyai selisih yang sama dan pola yang ke dua antara DE200 dan seri DE yang lain berbeda jauh selisih akurasinya. Selanjutnya karakter akurasi DE405 selalu sama dengan DE406 pada semua interval dan peristiwa gerhana Matahari yang diuji pada penelitian ini. Sedangkan seri DE200 pada interval setelah 1 detik (lebih kecil dari 1 detik) akurasinya untuk memprediksi waktu gerhana Matahari total atau cincin di kota tertentu selalu lebih baik dibanding dengan seri DE yang lain kemudian disusul oleh DE421 dan yang terakhir adalah seri DE405 dan DE406. Temuan tersebut bisa digunakan utntuk memprediksi waktu gerhana Matahari total atau cincin di kota tertentu dan rangkaian ibadah shalat gerhana Matahari dengan tepat waktu sesuai dengan tuntunan syariat.

Kata kunci: karekteristik *ephemeris*, gerhana Matahari,
JPL NASA

PEDOMAN TRANSLITERASI

Keputusan Bersama Menteri Agama dan Menteri P dan K
Nomor: 158/1987 dan Nomor: 0543b/U/1987

1. Konsonan

No.	Arab	Latin
1	ا	tidak dilambangkan
2	ب	b
3	ت	t
4	ث	ṡ
5	ج	j
6	ح	ḥ
7	خ	kh
8	د	d
9	ذ	ż
10	ر	r
11	ز	z
12	س	s
13	ش	sy
14	ص	ṣ
15	ض	ḍ

No.	Arab	Latin
16	ط	ṭ
17	ظ	ẓ
18	ع	‘
19	غ	g
20	ف	f
21	ق	q
21	ك	k
22	ل	l
23	م	m
24	ن	n
25	و	w
26	ه	h
27	ء	’
28	ي	y

2. Vokal Pendek

. َ ...= a	كَتَبَ	kataba
. ... = i	سُئِلَ	su'ila
. ُ ... = u	يَذْهَبُ	yażhabu

3. Vokal Panjang

اَ... = ā	قَالَ	qāla
اِيْ= ī	قِيْلَ	qīla
اُوْ = ū	يَقُوْلُ	yaqūlu

4. Diftong

اَيْ = ai	كَيْفَ	kaifa
اَوْ = au	حَوْلَ	ḥaula

Catatan:
Kata sandang [al-] pada bacaan syamsiyyah atau qamariyyah ditulis [al-] secara konsister supaya selaras dengan teks Arabnya.

KATA PENGANTAR

Segala puji bagi Allah yang maha pengasih dan penyayang, bahwa atas *taufiq* dan hidayah-Nya maka penulis dapat menyelesaikan penyusunan tesis ini. Tesis ini disusun untuk memenuhi salah satu syarat guna memperoleh gelar Magister Ilmu Falak (S2) Fakultas Syari'ah dan Hukum Universitas Islam Negeri (UIN) Walisongo Semarang.

Dalam penyusunan tesis ini penulis banyak mendapatkan bimbingan dan saran-saran dari berbagai pihak sehingga penyusunan tesis ini dapat terselesaikan. Untuk itu penulis menyampaikan terima kasih kepada :

1. Bapak Dr. H. Ahmad Izzuddin, M.Ag. dan Bapak Dr. H. Ali Imron, M.Ag. selaku Dosen Pembimbing yang telah bersedia meluangkan waktu, tenaga dan pikiran untuk memberikan bimbingan dan pengarahan dalam penyusunan tesis ini.
2. Bapak Dr. Muhammad Irfan Hakim, M.Si. yang selalu meluangkan waktu untuk berdiskusi dengan penulis baik secara *on-line* maupun *on-site* dalam proses penyelesaian tesis ini.
3. Yang terhormat Prof. Dr. Ahmad Rofiq, M.A. selaku Direktur Pascasarjana UIN Walisongo Semarang.
4. Yang terhormat Dr. H. Akhmad Arif Junaidi, M.Ag selaku Dekan Fakultas Syari'ah dan Hukum UIN Walisongo Semarang.
5. Pimpinan Perpustakaan Institut dan Pascasarjana yang telah memberikan izin dan layanan kepustakaan yang diperlukan dalam penyusunan tesis ini.
6. Para Dosen Pengajar di lingkungan Magister Ilmu Falak Fakultas Syari'ah dan Hukum UIN Walisongo, yang telah membekali berbagai pengetahuan sehingga penulis mampu menyelesaikan penulisan tesis.
7. Kedua orang tua Bapak H. Muchibuddin dan Ibu Hj. Silaturrohmi serta kedua mertua penulis Bapak H. Ahmad Choiruddin dan Ibu Hj. Mazro'ah yang senantiasa memberikan dukungan dan restunya.

8. Bapak KH. Drs. Ahmad Hadlor Ihsan dan keluarga yang senantiasa memberikan *wejangan* selama penulis di pesantren.

Pada akhirnya penulis menyadari bahwa penulisan tesis ini belum mencapai kesempurnaan dalam arti sebenarnya, namun penulis berharap semoga tesis ini dapat bermanfaat bagi penulis sendiri khususnya dan para pembaca pada umumnya.

Semarang, 08 Juni 2017
Penulis

M. Basthoni

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SINGKATAN

API	: *Application Programming Interface*
ASCII	: *American Standard Code for Information Interchange*
AU	: *Astronomical Unit*
DE	: *Development Ephemeris*
ELP	: *Ephemeride Lunaire Parisienne*
EMB	: *Earth-Moon Barycenter*
FTP	: *File Transfer Protocol*
GMC	: Gerhana Mataharu Cincin
GMT	: Gerhana Mataharu Total
ICRF	: *International Celestial Reference Frame*
JD	: *Julian Day*
JED	: Julian Day *Ephemeris*
JPL	: *Jet Propulsion Laboratory*
NASA	: *National Aeronautics and Space Administration*
VE	: *Vernal Equinox*
VSOP	: *Variations Seculaires Des Orbites Planetaires*

DAFTAR ISTILAH

Algoritma : Urutan logis dalam pemecahan masalah

American Standard Code for Information Interchange : Format teks standar yang digunakan untuk pertukaran data antar komputer dengan *platform* yang berbeda.

Application Programming Interface : Service yang disediakan oleh *programmer* untuk mengkomunikasikan antara beberapa bahasa pemrograman yang berbeda

Astronomical Almanac : Almanac yang dipublikasikan oleh *United States Naval Observatory* (USNO) *and Her Majesty's Nautical Almanac Office* (HMNAO).

Astronomical Unit : Satuan astronomi yang jaraknya sama dengan jarak Bumi ke Matahari.

Bahasa C : Salah bahasa pemrograman yang diciptakan Brian W. Kernighan dan Denis M. Ritchi

Binary : Format file spesifik yang hanya dikenali oleh komputer dengan spesifikasi prosesor tertentu.

Chebyshev Polynomial : Rangkain persamaan suku banyak yang digunakan untuk memodelkan suatu data.

Earth-Moon Barycenter : Titik tengah massa dari sistem Bumi-Bulan.

Ekliptika : Bidang edar Bumi mengelilingi Matahari atau bidang edar gerak semu Matahari mengelilingi Bumi.

Elongasi : Sudut pisah antara dua benda langit dilihat dari Bumi.

Fitting : Teknik penetuan titik-titik data yang belum diketahui nilainya di antara nilai data-data yang sudah diketahui nilainya

International Celestial Reference Frame : sebagai acuan yang mengadopsi lokasi dari 295 ekstragalaksi melalui observasi VLBI (*Very Long Baseline radio Interferometry*)

Konjungsi : Kondisi di mana dua benda langit memiliki nilai koordinat bujur langit yang sama

Kontak pertama : Kondisi ketika piringan Bulan menyentuk piringan Matahari sebelum puncak Gerhana

Kontak terakhir : Kondisi ketika piringan Bulan menyentuk piringan Matahari setelah puncak Gerhana

Librasi : Sebuah osilasi yang sangat lambat dari benda langit yang dilihat dari benda langit lain yang dikelilinginya.

Magnitude : bagian dari diameter Matahari yang disapu atau tertutup oleh piringan Bulan.

Nutasi : Gerak *iregular* dalam orde beberapa detik busur pada sumbu rotasi Bumi.

Okultasi : Peristiwa yang berlangsung ketika suatu obyek ditutupi oleh obyek lain yang lewat di antara obyek yang ditutupi dan pengamat.

On the fly : nilai sebuah *variable* yang dihasilkan ketika proses *runnning* sebuah program

Orbit : Garis edar suatu planet atau benda langit mengelilingi pusat edarnya

Velocity : Besaran yang menyatakan lintasan atau arah perpindahan tiap satuan waktu.

Vernal Equinox : Salah satu titik perpotongan antara bidang equator dan bidang ekliptika.

Visual Basic : Bahasa pemrograman yang diciptakan oleh Microsoft berbasis bahasa basic

Zenit : Titik di angkasa yang berada persis di atas pengamat.

BAB I
PENDAHULUAN

A. Latar Belakang

Ephemeris yang dalam bahasa Arab disebut *zij* atau *taqwim* merupakan tabel yang memuat data astronomis benda-benda langit[1]. Data astronomis yang meliputi jarak, posisi dan pergerakan benda langit tersebut diperoleh dengan pendekatan perhitungan benda-benda langit. Perhitungan tersebut sangat mungkin dilakukan karena benda-benda langit tersebut bergerak dalam sistem yang harmonis, teliti dan rumit namun akurat dan teratur[2]. Sebagaimana firman Allah:

فَالِقُ ٱلْإِصْبَاحِ وَجَعَلَ ٱلَّيْلَ سَكَنًا وَٱلشَّمْسَ وَٱلْقَمَرَ حُسْبَانًا ۚ ذَٰلِكَ تَقْدِيرُ ٱلْعَزِيزِ ٱلْعَلِيمِ

Dia yang menyingsingkan pagi dan menjadikan malam untuk istirahat, dan (menjadikan) Matahari dan Bulan untuk perhitungan. Itulah ketentuan Allah Yang Maha Perkasa lagi Maha Mengetahui (Q.S. al-An'am/6: 96).[3]

---

[1] Susiknan Azhari, *Ensiklopedi Hisab Rukyat*, (Yogyakarta: Pustaka Pelajar, 2012), 61-62. Lihat juga: Arwin Juli Rakhmadi Butar-Butar, *Khazanah Astronomi Islam Abad Pertengahan*, (Purwokerto: UM Purwokerto Press, 2016), 434.

[2] Quraish Shihab, *Tafsir Al-Mishbah*, vol. 12, (Jakarta: Lentera Hati, 2001), 496-498.

[3] Departemen Agama RI, *al-Qur'an dan Terjemahnya,* (Semarang: CV Al Waah, 2004), 188.

Dan juga firman Allah:

ٱلشَّمۡسُ وَٱلۡقَمَرُ بِحُسۡبَانٍ

Matahari dan Bulan (beredar) menurut perhitungan (Ar-Rahman/55: 5).[4]

Ayat di atas menggunakan pilihan kata حسبان untuk menjelaskan perhitungan pergerakan Matahari dan Bulan. Kata حسبان berasal dari kata حسب artinya perhitungan, penambahan *alif* dan *nun* pada kata tersebut menunjukkan arti kesempurnaan dan ketelitian[5]. Oleh karena itu kata حسبان pada ayat di atas memberikan arti bahwa pergerakan Matahari dan Bulan adalah dapat diketahui kadar perhitungannya oleh manusia karena perhitungannya yang teratur. [6] Dari penafsiran tersebut dapat disimpulkan bahwa dengan kekuasaan Allah, manusia dapat memperhitungkan posisi dan pergerakan Matahari dan Bulan karena keduanya bergerak secara teratur, kemudian mengambil manfaat dari hal-hal yang ditimbulkan oleh keteraturan pergerakan tersebut bagi kehidupan mereka.[7]

---

[4] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 773

[5] Quraish Shihab, *Tafsir Al-Mishbah*, vol. 4, 205.

[6] Ismail bin Katsir al-Qurasy, *Tafsir al-Qur'an al-'Aḍīm,* (ttp.: Syirkah an-Nur Asia, tt.), 270.

[7] Ahmad al-Ṣāwy al-Maliky, *Ḥāsyiyah al-'Allāmah al-Ṣāwy 'ala Tafsir al-Jalālain,* (Beirut: Dar al-fikr, tt.), 153.

Dalam keilmuan falak perhitungan tersebut sangat penting, karena beberapa perintah ibadah dalam Islam, waktu pelaksanaannya sangat terkait dengan posisi dan pergerakan Matahari, Bumi dan Bulan tersebut[8], di antaranya adalah perintah pelaksanaan shalat gerhana Matahari. Sejalan dengan hal tersebut Nabi saw. bersabda :

كُسِفَتْ الشَّمْسُ عَلَى عَهْدِ رَسُوْلِ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ يَوْمَ مَاتَ إِبْرَاهِيْمُ فَقَالَ النَّاسُ كُسِفَتْ الشَّمْسُ لِمَوْتِ إِبْرَاهِيْمَ فَقَالَ رَسُوْلُ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ : إِنَّ الشَّمْسَ وَالْقَمَرَ لاَيَنْكَسِفَانِ لِمَوْتِ أَحَدٍ وَلاَلِحَيَاتِهِ فَإِذَا رَأَيْتُمْ فَصَلُّوْا وَادْعُوْا اللهَ

[9]

Ketika Nabi Muhammad SAW masih hidup, gerhana Matahari terjadi pada hari yang bersamaan dengan wafatnya Ibrahim (putra Nabi SAW). Orang-orang pun berkata bahwa gerhana Matahari terjadi karena meninggalnya Ibrahim. Maka kemudian Rasulullah SAW bersabda: Sesungguhnya gerhana Matahari dan Bulan terjadi bukan karena mati atau hidup seseorang, maka ketika kalian melihat gerhana, shalatlah dan berdoalah kepada Allah. (HR. Bukhari).

---

[8] Ahmad Izzuddin, *Ilmu Falak Praktis*, (Semarang: Pustaka Rizki Putra, 2012), 2. Lihat juga: Hendro Setyanto, *Membaca Langit*, (Jakarta: al-Ghuraba, 2008), 45.

[9] Imam Bukhāry, *Ṣaḥīh Bukhāri*, Juz I, (Beirut: Dar Kutub al-Arabiyah; 1992), 316.

Menurut Syamsul Anwar[10], dalam hadis tersebut Nabi saw menjelaskan bahwa gerhana Matahari adalah peristiwa alam yang natural yang menunjukkan kebesaran Allah dan tidak ada kaitannya dengan kematian dan hidup seseorang. Dikatakan natural karena peristiwa gerhana tersebut terjadi karena pergerakan Matahari dan Bulan yang bisa dihitung sebagaimana penafsiran kata حسبان pada ayat tersebut di atas.

Sejalan dengan perkembangan zaman, perhitungan tentang peredaran benda-benda langit juga berkembang sangat cepat. Metode perhitungan yang berkembang pada saat ini tidak lagi manual, tetapi metode yang berbasis teknologi dengan sistem komputer dan algoritmanya.

Di antara algoritma tersebut adalah algoritma VSOP87 untuk menentukan posisi Matahari dan algoritma ELP2000 untuk menentukan posisi Bulan. VSOP (*Variations Seculaires Des Orbites Planetaires*) ini ditulis oleh Pierre Bretagnon pada tahun 1982[11], yang kemudian disempurnakan bersama Gerrad Francou pada tahun 1987, atau sering disebut

---

[10]Syamsul Anwar, *Interkoneksi Studi Hadis dan Astronomi*, (Yogyakarta: Suara Muhammadiyah, 2011), 61-62.

[11] P. Bretagnon, *Theorie du mouvement de l'ensemble des planetes. Solution VSOP82*, Astronomy & Astrophysics, 114, (1982): 278.

dengan VSOP87[12] yang dipublikasikan pada jurnal *Astronomy and Astrophysies*, 202, 309-315 (1988). Dengan penyempurnaan ini, VSOP87 memiliki peningkatan akurasi lebih baik dari 0,01 detik busur[13] dibandingkan VSOP82 yang merupakan hasil *fitting* dari data *Development Ephemeris* seri 200 (DE200)[14] yang dirilis oleh *Jet Propulsion Laboratory* (JPL) NASA pada September 1981.

Sedangkan ELP (*Ephemeride Lunaire Parisienne*) ini ditulis oleh Michella Chapront-Touze dan Jean Chapront pada tahun 1980 yang disebut dengan ELP1900, dan kemudian disempurnakan pada tahun 1988 oleh Jean Chapront dan Michella Chapront-Touze menjadi ELP2000. ELP2000 pada versi awalnya diperoleh dari analisis dan *fitting* dari integrasi numerik beberapa konstanta data JPL DE200.[15]

---

[12] Teori ini dikembangkan oleh Pierre Bretagnon bersama Gerrad Francou pada tahun 1987 dengan berbasis pada teori VSOP82. Lihat: P. Bretagnon, G. Francou, *Planetary Theoris in Rectangular and Sperical Variables. VSOP87 Solutions*, Astronomy & Astrophysics, 202, (1988): 309.

[13] Jean Meeus, *Astronomical Algorithms*, Second English Edition, (Virginia: William-Bell, Inc., Richmond, 1998), 166.

[14] Bretagnon , P., *Theorie du movement de l'ensemble des planets. Solution VSOP82*, Astron. Astophys. 114, 1982, 278. Lihat juga: J.-L. Simon, et.al, *New Analytical Planetary Theories VSOP2013 and TOP2013*, Astronomy & Astrophysics, A&A 557, A49, (2013): 3, doi: 10.1051/0004-6361/201321843.

[15] M. Touze Chapront and J. Chapront, *ELP 2000-85: A Semi-analiytical Lunar Ephemeris Adequate for Historical Times*, Astron. Astrophys. 190, 1988, 346.

JPL sejauh ini terus merilis data *ephemeris*. DE436 adalah rilis data *ephemeris* terbaru yang dirilis pada Maret 2017.[16] Namun untuk keperluan publikasi fenomena gerhana Matahari, NASA mendasarkan pada data *ephemeris* JPL DE405 untuk memprediksi gerhana Matahari yang terjadi pada 2011-2020[17] sementara sebelum tahun 2011[18] dan setelah 2020[19], NASA menggunakan teori VSOP87 dan ELP2000 yang berbasis pada DE200. Prediksi NASA tersebut dalam perkembangan selanjutnya sering dijadikan acuan pembanding keakuratan perhitungan astromonis buku-buku falak modern dan beberapa penelitian. Misal Rinto Anugraha dalam

---

[16] ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/, akses 11 April 2017, 13:15 WIB.

[17] Lihat misalnya : Fred Espenak, *Solar Eclipse Predictions with JPL DE405*, https://eclipse.gsfc.nasa.gov/help/de405-predictions.html, akses 15 April 2017, 10:25 WIB. Informasi penggunaan DE405 juga digunakan untuk memprediksi gerhana Matahari 9 Maret 2016, https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/SE2016Mar09Tgo ogle.html, akses 15 April 2017, 11:00 WIB, dan juga gerhana Matahari 21 Agustus 2017, https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/ SE2017Aug21Tgoogle.html, akses 15 April 2017, 12:10 WIB. Lihat juga prediksi gerhana Matahari 20 Mei 2012 oleh NASA di https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/SE2012May20Ago ogle.html dan prediksi gerhana Matahari 14 Desember 2020 oleh NASA di https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/SE2020Dec14T google.html, akses 6 Juni 2017, 11:28 WIB.

[18] Lihat misalnya prediksi gerhana Matahari 11 Juli 2010 oleh NASA di https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/ SE2010Jul11Tgoogle.html, akses 6 Juni 2017, 11:28 WIB

[19] Lihat misalnya prediksi gerhana Matahari 10 Juni 2021 oleh NASA di https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/ SE2021Jun10Agoogle.html, akses 6 Juni 2017, 11:28 WIB

Mekanika Benda Langit membandingkan hasil perhitungan gerhana Matahari dengan data yang dirilis NASA dan menyimpulkan bahwa hasil perhitungannya selisih kurang dari 1 menit dengan data NASA tersebut.[20]

Sementara itu, mayoritas software falak dalam perhitungan koordinat Matahari dan Bulan menggunakan algoritma Jean Meeus sebagaimana diimplementasikan oleh M. Odeh dalam software *Accurate Times* dan Dr. Ing H. Khafid dalam software *Mawaqit* versi awal. Sedangkan pada versi berikutnya *Mawaqit* menggunakan algoritma VSOP87.

Data Matahari dan Bulan yang diperoleh dari beberapa software tersebut memang sudah cukup signifikan tingkat akurasinya walaupun mempunyai selisih dengan data yang dirilis oleh NASA. Perbedaan data yang diambil, algoritma yang membangun teori dan rumus-rumus yang digunakan menjeadi penyebab perbedaan hasil dari masing-masing algoritma. Sebagaimana diketahui bahwa algoritma Jean Meeus dalam perhitungan posisi Matahari dan Bulan sebenarnya merupakan reduksi dari perhitungan VSOP87 dan ELP-2000/82 yang lebih rumit dan lebih tinggi akurasinya. VSOP87 adalah rujukan perhitungan data Matahari dalam algoritma Jean Meeus, adapun ELP-2000/82 merupakan

[20] Rinto Anugraha, *Mekanika Benda Langit*, (Yogyakarta: Lab. Fisika Material dan Instrumentasi Jurusan Fisika FMIPA UGM, 2012), 147.

rujukan dalam perhitungan data Bulan-nya. Dari ribuan suku koreksi VSOP87 dan ELP-2000/82, Meeus hanya mengambil beberapa ratus suku koreksi saja. Ia hanya mengambil suku-suku koreksi yang dinilai besar dan penting, dan membuang suku-suku koreksi yang kurang penting[21]. Sedangkan VSOP dan ELP tersebut, sebagaimana diuraikan sebelumnya, merupakan hasil *fitting* dari data *Development Ephemeris* seri 200 (DE200).

Berangkat dari latar belakang di atas, penelitian ini mengkaji algoritma dan perbandingan karakteristik akurasi antara beberapa seri *Development Ephemeris* (DE) JPL NASA. Algoritma diartikan sebagai urutan logis dalam pemecahan masalah.[22] Dalam hal ini yang dimaksud adalah urutan logis dalam perhitungan data *ephemeris* berbasis JPL NASA. Sedangkan karakteristik memiliki arti ciri-ciri khusus[23] dan akurasi adalah ketepatan dalam merepresentasikan kontruksi dunia nyata yang dijadikan rujukan.[24] Sehingga yang dimaksud perbandingan karakteristik akurasi dalam kajian ini adalah perbandingan

---

[21] Rinto Anugraha, *Mekanika Benda Langit*, 68.

[22] Departemen Pendidikan Nasional, *Kamus Besar Bahasa Indonesia Pusat Bahasa*, ed. 4, (Jakarta: PT. Gramedia Pustaka Utama, 2008), 75.

[23] Departemen Pendidikan Nasional, *Kamus Besar Bahasa Indonesia*, 623.

[24] Departemen Pendidikan Nasional, *Kamus Besar Bahasa Indonesia*, 56.

ciri-ciri khusus masing-masing seri DE JPL NASA dalam ketepatannya merepresentasikan data empirik (laporan observasi peristiwa gerhana Matahari) yang dijadikan acuan.

Pemilihan kajian ini berdasarkan dengan beberapa alasan. ***Pertama***, data *ephemeris* Matahari dan Bulan dengan ketelitian tinggi dibutuhkan untuk perhitungan dalam ilmu falak, misalnya penentuan waktu shalat gerhana Matahari dengan tepat sesuai dengan tuntunan syariat. Sehingga dengan menemukan seri DE yang paling akurat, kajian ini mampu memberikan sumbangan dalam menentukan waktu shalat gerhana Matahari dengan tepat sesuai dengan standar kaidah ilmu astronomi.

Berkaitan dengan kebutuhan tingkat akurasi tinggi tersebut, Jean Meeus[25] mengatakan bahwa keakuratan yang dibutuhkan dalam perhitungan bergantung pada tujuannya. Misalnya, hasil perhitungan posisi bintang dengan metode akurasi rendah sudah cukup baik untuk pengamat yang ingin mencari benda-benda langit dengan teleskop yang dipasang secara paralaktik (mengukur vertikal ke arah zenit), namun perhitungan tersebut tidak memenuhi keakuratan yang diinginkan jika untuk perhitungan okultasi[26] dan konjungsi

[25] Jean Meeus, *Astronomical Algorithms*, 15.

[26] Okultasi berlangsung ketika suatu obyek ditutupi oleh obyek lain yang lewat di antara obyek yang ditutupi dan pengamat. Hannu Karttunnen, et.al, 1996, *Fundamental Astronomy*, (New York: Springer,1995), 168-169.

karena kecilnya piringan dari obyek benda langit yang diamati.

Alasan ***kedua***, data *ephemeris* JPL NASA menjadi acuan munculnya teori-teori tentang perhitungan posisi dan pergerakan planet, bintang dan obyek langit lain. Di antaranya adalah teori VSOP untuk perhitungan koordinat Matahari dan ELP untuk perhitungan koordinat Bulan yang keduanya merupakan hasil *fitting* dari data *ephemeris* JPL NASA seri DE200. Dengan mengetahui algoritma perhitungan data yang dirilis oleh JPL NASA, diharapkan bisa dikembangkan penelitian lanjutan untuk menemukan teori baru yang menghasilkan komputasi yang lebih akurat.

Ketiga, data peristiwa astronomis yang dirilis oleh NASA, misal gerhana Matahari, sering dijadikan acuan keakuratan hasil perhitungan beberapa penelitian. Sehingga dengan mengetahui algoritma perhitungan dan akurasi data *ephemeris* JPL NASA diharapkan hasil komputasi dalam perhitungan ilmu falak lebih akurat karena berbasis data *ephemeris* yang memiliki akurasi yang lebih baik.

B. Rumusan Masalah

Pada dasarnya penelitian ini menitikberatkan pada sebuah upaya mengungkap karakteristik akurasi data *ephemeris* yang dirilis oleh JPL NASA sehingga akhirnya diketahui dari sisi akurasi mengapa NASA menggunakan seri

data tertentu untuk keperluan rilis informasi kejadian astronomis tertentu. Secara kongkrit permasalahan dalam penelitian ini adalah:

1. Bagaimana algoritma perhitungan data *ephemeris* dan gerhana Matahari berbasis JPL NASA?
2. Bagaimana karakteristik akurasi data *ephemeris* berbasis JPL NASA dalam implementasinya untuk perhitungan gerhana Matahari?

C. Tujuan Penelitian

Adapun yang menjadi tujuan dari penelitian ini adalah:

1. Untuk mengetahui algoritma yang digunakan dalam perhitungan data *ephemeris* dan perhitungan gerhana Matahari berbasis JPL NASA.
2. Untuk mengetahui karakteristik akurasi data *ephemeris* berbasis JPL NASA dalam implementasinya untuk perhitungan waktu gerhana Matahari.

D. Signifikansi Penelitian

Signifikansi dari penelitian ini adalah :

1. Memberikan sumbangan algoritma modern dalam penentuan data *ephemeris* dan perhitungan gerhana Matahari berbasis JPL NASA.

2. Memberikan kejelasan tentang karakteristik akurasi masing-masing seri data *ephemeris* berbasis JPL NASA dalam implementasinya untuk perhitungan waktu gerhana Matahari.

E. Kajian Pustaka

Sejauh penelusuran penulis, belum diketahui penelitian atau pun tulisan yang secara mendetail membahas tentang komputasi data *ephemeris* berbasis JPL NASA. Sekalipun banyak karya-karya tentang algoritma astronomi modern untuk komputasi peristiwa-peristiwa astronomis yang berkaitan dengan ibadah umat Islam, misalnya tentang awal bulan kamariyah, gerhana Matahari dan Bulan atau yang tidak terkait langsung dengan ibadah, misalnya komparasi antara algoritma yang satu dengan yang lain.

Haryono, mahasiswa Pascasarjana Ilmu Falak IAIN Walisongo (2011) dengan judul penelitiannya "*Astronomical Algorithms* Modern Berbasis Teori VSOP87 dan ELP-2000 Dalam Penentuan Awal Bulan Hijriyah". Dalam penelitiannya, ia membuat suatu program dengan menggunakan software Visual Basic yang di dalamnya terdapat perhitungan awal bulan kamariah berdasarkan teori VSOP-87 dan ELP-2000. Adapun keakuratan yang terdapat dalam penelitiannya termasuk kepada *high accury* sebanding

dengan software lainnya seperti *mawaqit*, *accurate time* dan lain-lainnya.

Pada penelitian ini, perbedaan terletak pada kajian algoritma astronomi dalam menentukan data *ephemeris* Matahari dan Bulan yang digunakan menggunakan teori VSOP87 dan ELP-2000 yang selanjutnya digunakan sebagai data penetuan awal bulan Hijriyah. Pada penelitian yang dikaji menggunakan algoritma astronomi dari data JPL NASA yang kemudian diimplementasikan dalam penentuan waktu gerhana Matahari.

Imas Musfiroh dalam tesisnya, *Hisab Awal Bulan Kamariah (Studi Komparatif Sistem Hisab Almanak Nautika dan Astronomical Algorithms Jean Meeus)* membandingkan algoritma perhitungan antara Almanak Nautika dan algoritma Meeus serta kelebihan dan kekurangan masing-masing dalam menentukan awal bulan kamariyah. Hampir sama dengan penelitian yang penulis sajikan sebelumnya, perbedaan pada penelitian ini terdapat pada algoritma yang digunakan.

Sergey M. Kudryavtsev dalam papernya, *Analytical Series Representing DE431 Ephemerides Of Terrestrial Planets*, membandingkan akurasi teori VSOP2013 dan data *ephemeris* JPL NASA seri DE431. Hasilnya, VSOP2013 merupakan teori analitik numerik yang memberikan solusi terbaik untuk memprediksi pergerakan planet dalam rentang waktu lebih dari 100 tahun, namun DE431 menghasilkan

prediksi yang lebih baik pada interval waktu yang lebih panjang. Selisih antara keduanya untuk *ephemeris* Merkurius adalah 0,5 km, 14 km untuk Venus, 12,7 km untuk sistem Bumi-Bulan (EMB) dan 76,5 km untuk Mars.[27]

Perbedaan dengan penelitian ini adalah pada fokus penelitian. Dalam paper tersebut diuraikan perbandingan akurasi antara VSOP2013 dan DE431 sedangkan pada penelitian ini fokus pada pengungkapan algoritma dan perbandingan beberapa seri data *ephemeris* berbasis JPL NASA serta implementasinya dalam penentuan waktu gerhana Matahari.

Oleh karena itu, peneliti merasa belum ada kajian yang membahas secara spesifik terkait dengan perhitungan data *ephemeris* Matahari dan Bulan berbasis data *Development Ephemeris* yang dirilis oleh JPL NASA.

F. Metode Penelitian

Penelitian ini termasuk ke dalam jenis penelitian kualitatif berupa penelitian kepustakaan (*library research*). Penelitian kualitatif merupakan suatu strategi *inquiry* (penyelidikan) yang menekankan pencarian makna,

[27] Sergey M. Kudryavtev, *Analytical Series Representing DE431 Ephemrides of Terrestrial Planets*, MNRAS (March 11, 2016) 456 (4): 4015-4019. doi: 10.1093/mnras/stv2892, http://mnras.oxfordjournals.org/content/456/4/4015.abstract, akses 10 Januari 2016.

pengertian, konsep, karakteristik, gejala, simbol maupun tentang fenomena; fokus dan multimedia; bersifat alami dan holistik; mengutamakan kualitas, menggunakan beberapa cara serta disajikan secara naratif.[28]

Sedangkan penelitian ini memfokuskan pada kajian analisis terhadap data *Development Ephemeris*, file dokumentasi dan *user manual* tentang komputasi berbasis JPL yang dirilis oleh JPL NASA di server FTP (*file transfer protocol*) JPL NASA di ftp://ssd.jpl.nasa.gov/pub/eph/planets/ serta buku *Fundamental Ephemeris Computations: For Use With JPL Data* karya Paul J. Heafner. Dokumen dan literatur tersebut dijadikan sebagai sumber data primer[29] yang dapat memberikan informasi secara lengkap tentang komputasi data *ephemeris* berbasis JPL NASA. Untuk mendukung hasil penelitian menjadi lebih akurat, maka peneliti memerlukan data dukungan (sekunder) berupa literatur yang membahas tentang perhitungan data *ephemeris* Matahari dan Bulan, buku tentang teori *spherical trigonometry*, interpolasi, transformasi koordinat serta buku-buku astronomi lainnya dan buku algoritma pemrograman berbasis bahasa C.

---

[28] A. Muri Yusuf, *Metode Penelitian: Kuantitatif, Kualitatif dan Penelitian Gabungan*, (Jakarta: Prenadamedia Group, 2014), 329.

[29] Sumber data primer adalah sumber data yang langsung memberikan data kepada pengumpul data sedangkan sumber data sekunder merupakan sumber yang tidak langsung memberikan data kepada pengumpul data. Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif dan R & D*, (Bandung: Penerbit Alfabeta, 2006), 253.

Adapun teknik pengumpulan data yang dilakukan dalam penelitian ini adalah metode dokumentasi. Metode dokumentasi digunakan untuk menelaah dokumen-dokumen tertulis, baik dokumen yang bersifat primer maupun sekunder yang kemudian dipilih dan dipilah menurut kesesuaiannya dengan tema penelitian ini.

Setelah mengalami proses pengumpulan data, kemudian peneliti melakukan olah data dan analisis data. Analisis data diarahkan untuk menjawab rumusan masalah yang telah disusun. Analisis yang digunakan adalah *content analisis*[30] (analisis isi) melalui teknik deskriptif dipaparkan algoritma perhitungan *ephemeris* dan gerhana Matahari berbasis JPL NASA.

Deskriptik analitik pada penelitian ini juga dilakukan dengan pendekatan *aritmatik* (ilmu hitung) yaitu melakukan perhitungan pada data dengan menggunakan teori *spherical trigonometry*, transformasi koordinat dan interpolasi dalam perhitungan *ephemeris* berbasis JPL NASA. Selanjutnya teori tersebut diaplikasikan dalam bahasa pemrograman dengan menggunakan Bahasa C menggunakan algoritrma yang dituliskan oleh Paul J. Heafner dalam bukunya tersebut.

---

[30] Menurut Holsti, sebagaimana dikutip oleh Lexy J. Moleong, kajian isi adalah teknik apa pun yang digunakan untuk menarik kesimpulan melalui usaha menemukan karakteristik pesan dan dilakukan secara obyektif dan sistematis. Lexy J. Moleong, *Metodologi Penelitian Kualitatif*, (Bandung: PT Remaja Rosdakarya, 2001), 163.

Penelitian ini juga menggunakan teori analisis komparatif, yakni menggunakan logika perbandingan. Dengan melakukan perbandingan antara beberapa seri data *ephemeris* JPL NASA, yaitu seri DE200, DE405, DE406 dan DE421[31], bisa diketahui perbandingan akurasi masing-masing seri data. Tolak ukur dalam perbandingan akurasi ini adalah adalah laporan observasi peristiwa gerhana Matahari yang telah terjadi karena peristiwa gerhana tersebut lebih mudah diamati oleh siapapun untuk selanjutnya dijadikan acuan

[31] Salah satu pertimbangan mengapa empat seri data tersebut yang dipilih adalah: DE200 merupakan basis dari munculnya teori VSOP dan ELP yang digunakan NASA untuk memprediksi gerhana Matahari yang terjadi sebelum 2011 dan setelah 2020 di samping DE200 tersebut juga digunakan dalam *Astronomical Almanac* (1984-2003). Lihat: *JPL Planetary and Lunar Ephemerides : Export Information*, ftp://ssd.jpl.nasa.gov/pub/eph/planets/README.txt, 12 April 2017, 08:17 WIB.

Sedangkan DE405 digunakan dalam *Astronomical Almanac* (2003) dan data *ephemeris* yang dijadikan dasar NASA dalam merilis informasi gerhana Matahari yang terjadi pada 2011-2020. Lihat: Fred Espenak, *Solar Eclipse Predictions with JPL DE405*, https://eclipse.gsfc.nasa.gov/help/de405-predictions.html, akses 15 April 2017, 10:25 WIB.

DE406 adalah data *ephemeris* yang mirip dengan DE405 yang memiliki cakupan rentang waktu yang lebih panjang yaitu sampai tahun 3000 dan DE421 adalah data ephemeris yang pernah digunakan NASA dalam HORIZONS Sytem *on-line* sebelum digantikan oleh DE431. HORIZONS Sytem *on-line* yaitu sistem informasi data ephemris on-line yang disediakan oleh NASA yang bisa diakses di https://ssd.jpl.nasa.gov/?horizons. Pemilihan seri DE yang diuji terbatas pada seri DE42x karena keterbatasan algoritma dari Heafner (sebagai *tool* dalam penelitian ini) yang belum bisa membaca dengan benar seri DE43x yang memiliki format file dan struktur file yang berbeda dengan seri DE2xx dan seri DE42x.

akurasi data *ephemeris* yang digunakan sebagai basis data perhitungan prediksi gerhana Matahari.

Untuk itu sebelum melakukan uji akurasi masing-masing seri DE, beberapa seri DE tersebut digunakan sebagai basis data perhitungan gerhana Matahari total dan cincin. Dalam perhitungan gerhana tersebut, hanya dilakukan perhitungan kontak-kontak yang berkaitan dengan penentuan waktu shalat gerhana Matahari, yaitu kontak pertama dan terakhir serta tengah gerhana atau puncak gerhana.

Setelah ditentukan waktu kontak pertama, terakhir dan tengah gerhana Matahari total atau cincin berbasis 4 (empat) seri DE tersebut, hasilnya dibandingkan dengan laporan beberapa observasi gerhana Matahari total atau cincin yang telah terjadi. Dalam hal ini hasil obrservasi (data empiris) yang dijadikan acuan adalah observasi yang telah dilakukan oleh Fred Espenak,[32] *LangitSelatan*[33] yang juga dilaporkan

[32] Fred Espenak adalah ilmuan di *Goddard Space Flight Center* NASA dan ahli gerhana di NASA. Dia yang mengelola web site resmi NASA tentang gerhana di http://eclipse.gsfc.nasa.gov dan web site pribadinya tentang fotografi gerhana di www.mreclipse.com dan www.astropixels.com. Pada 2003 *International Astronomical Union* (IAU) memberi penghargaan dengan memberi nama salah satu asteroid dengan nama "Espenak". Lihat: *Biography: Fred Espenak*, http://www.eclipse-chasers.com/article/ SEC2014/speaker/Fred Espenak Bio.pdf, akses 23 Mei 2017, 01:16 WIB, lihat juga: *Fred Espenak – Public Lectures and Speaking Engagements*, http://www.mreclipse.com/ main/lectures.html#Espenak, akses 23 Mei 2017, 01:16 WIB.

[33] *LangitSelatan* sebuah komunitas yang didirikan oleh alumni Astronomi ITB sebagai media komunikasi dan edukasi astronomi di

dan diverifikasi oleh Lembaga Penerbangan dan Antariksa Nasional (LAPAN) dan laporan pengamatan gerhana Matahari oleh Badan Meteorologi, Klimatologi dan Geofisika (BMKG). Laporan pengamatan gerhana Matahari tersebut diambil langsung dari web site masing-masing. Web site secara berurutan adalah http://www.mreclipse.com, http://gerhana.langitselatan.com, http://www.lapan.go.id dan http://media.bmkg.go.id.

Perbandingan dilakukan dengan eksperimen beberapa interval data *ephemeris* mulai dari 1 detik sampai dengan 0,0001 detik[34] dan hasil masing-masing eksperimen dibandingkan dengan hasil observasi Espenak tersebut. Dengan perbandingan tersebut bisa diketahui karakteristik akurasi masing-masing seri DE dalam rentang interval data *ephemeris* yang dimaksud.

Seri DE tertentu dikatakan paling akurat dibandingkan dengan seri yang lain jika seri DE tersebut memiliki selisih paling kecil dengan data empirik laporan observasi tersebut.

Indonesia baik secara *on-line* atau terjun langsung di lapangan. Sejak tahun 2008 *LangitSelatan* bergabung dengan asosiasi *Global Hands on Universe*, sebuah asosiasi internasional dalam hal pengembangan dan pengajaran astronomi bagi guru dan siswa. Lihat: https://langitselatan.com/tentang-ls/, akses 13 Juni 2017, 6:39 WIB.

[34] Rentang interval tersebut dipilih berdasarkan kemampuan hardware laptop yang digunakan untuk memproses data dalam proses komputasi dalam penelitian ini dan dengan pertimbangan bahwa dengan rentang interval tersebut sudah bisa menjawab pertanyaan penelitian ini.

Karena dalam astronomi, *ephemeris* adalah prediksi yang dibuat berdasarkan sebuah model reguralitas. Benar tidaknya suatu *ephemeris* ditentukan dengan fenomena yang terjadi. Oleh karenanya observasi dalam astronomi merupakan satu-satunya cara untuk menguji keakuratan *ephemeris*.[35]

Seri DE yang paling akurat tersebut selanjutnya dijadikan basis data dalam contoh penentuan waktu shalat gerhana Matahari total atau cincin pada waktu dan lokasi tertentu pada peristiwa gerhana Matahari yang akan datang.

G. Sistematika Pembahasan

Uraian pada penelitian ini dibagi menjadi lima bab sebagai berikut:

Bab pertama berisi pendahuluan. Pada bagian pendahuluan ini dikemukakan sketsa permasalahan yang melatarbelakangi penelitian tentang perhitungan data *ephemeris* Matahari dan Bulan berbasis JPL NASA. Kemudian dipaparkan fokus penelitian sebagai titik tolak penelitian ini yang dikemas dalam rumusan masalah, dilanjutkan dengan signifikansi penelitian sebagai arah dari penelitian, kajian pustaka dan metode penelitian sebagai cara mendekati sasaran penelitian.

---

[35] Hendro Setyanto, *Membaca Langit,* (Jakarta: al-Ghuraba, 2008), 41.

Bab kedua menjelaskan tentang tentang tinjauan al-Qur'an dan al-Hadits serta tinjauan sains atas perhitungan waktu sebagai akibat dari pergerakan Matahari, Bumi dan Bulan. Di dalamnya diuraikan tentang tinjauan al-Qur'an dan al-Hadits tentang pergerakan Matahari, Bumi dan Bulan berkaitan dengan perhitungan waktu gerhana Matahari. Selanjutnya teori-teori astronomi, meliputi teori heliosentris, geosentris dan yang terkait dengan keduanya, teori matematika, meliputi teori segitiga bola, transformasi, dan interpolasi, serta tentang gambaran umum algoritma pemrograman berbasis Bahasa C, juga dibahas dalam bab ini.

Bab ketiga membahas tentang JPL NASA terutama tentang algoritma yang digunakan dalam perhitungan data *ephemeris* berbasis JPL NASA dan penerapannya dalam perhitungan gerhana Matahari total atau cincin yang diimplementasikan dalam pemrograman berbasis Bahasa C.

Bab keempat menganalisis dan menguji perbandingan akurasi beberapa seri *Development Ephemeris* JPL NASA dengan hasil observasi gerhana Matahari yang telah terjadi sehingga ditemukan karakteristik akurasi masing-masing seri DE dan seri DE yang paling akurat tersebut digunakan dalam contoh untuk memprediksi waktu shalat gerhana Matahari total atau cincin yang akan terjadi di masa yang akan datang.

Bab kelima penutup, bagian ini berisi kesimpulan dan juga saran-saran guna memberi peluang bagi pembaca untuk mengoreksi dan mengembangkannya.

BAB II
PEREDARAN BENDA LANGIT, DATA *EPHEMERIS* DAN GERHANA MATAHARI

Secara umum ilmu astronomi mempelajari tentang jarak, posisi dan pergerakan benda-benda langit. Jarak, posisi dan pergerakan benda langit tentunya tidak dapat diketahui secara langsung, namun dapat diketahui dengan pengamatan secara berkala dalam rentang waktu tertentu dan hasilnya dijadikan sebagai dasar untuk memodelkannya secara matematis. Sehingga akhirnya ditemukan rumus perhitungan untuk posisi dan gerak benda langit tersebut.

Dari rumus perhitungan tersebut kemudian bisa diprediksi data-data astronomis yang menunjukkan jarak dan posisi benda-benda langit. Data-data tersebut kemudian dirangkum menjadi sebuah tabel data astronomis yang biasa disebut dengan tabel *ephemeris.* Adapun pengertian *ephemeris* dalam astronomi adalah tabel yang memberikan prediksi posisi benda-benda langit pada waktu yang ditentukan[1]

[1] Ian R idpath, *Dictionary of Astronomy*, (New York: Oxford University Press, 1997), 150. Lihat juga: Leif J. Robinson, *Philip's Astronomy Encyclopedia*, (London: Octopus Publishing Group, 2002), 131. Dalam bahasa Arab disebut *zij* yang bermakna tabel-tabel astronomi. Dalam terminologi klasik, *zij* adalah daftar astronomi hasil observasi dan kalkulasi (*hisab*) para astronom terhadap benda-benda langit yang meliputi gerak, jarak dan posisi harian. Lihat misalnya: Arwin

Berdasarkan uraian tersebut maka dalam bab ini akan diuraikan tentang peredaran benda langit serta beberapa teori yang terkait sehingga akhirnya diperoleh pemahaman bagaimana data *ephemeris* tersebut diperoleh. Selanjutnya juga akan dieksplorasi tentang gerhana Matahari sebagai implementasi bagaimana data *ephemeris* tersebut diterapkan dalam penentuan gerhana Matahari.

A. Peredaran Benda Langit dalam Perspektif Al-Qur'an dan Astronomi

1. Peredaran Benda Langit dalam Perspektif Al-Qur'an

Benda luar angkasa atau benda langit dalam perspektif al-Qur'an meliputi tiga hal: Matahari, Bulan dan bintang. Peredaran benda-benda tersebut ditemukan dalam al-Quran yakni:

فَالِقُ ٱلۡإِصۡبَاحِ وَجَعَلَ ٱلَّيۡلَ سَكَنٗا وَٱلشَّمۡسَ وَٱلۡقَمَرَ حُسۡبَانٗاۚ ذَٰلِكَ تَقۡدِيرُ ٱلۡعَزِيزِ ٱلۡعَلِيمِ

Dia menyingsingkan pagi dan menjadikan malam untuk beristirahat, dan (menjadikan) Matahari dan Bulan untuk perhitungan. Itulah ketentuan Allah Yang Maha Perkasa lagi Maha Mengetahui(QS. al-An'am/6: 96).[2]

Juli Rakhmadi Butar-Butar, *Khazanah Astronomi Islam Abad Pertengahan*, (Purwokerto: UM Purwokerto Press, 2016), 434.

[2] Departemen Agama RI, *al-Qur'an dan Terjemahnya,* (Semarang: CV Al Waah, 2004), 188.

Dari ayat tersebut ditemui kata kunci (*key words*) berkaitan dengan peredaran Matahari dan Bulan. Kata kunci tersebut adalah kata *taqdīr*.

Kata *taqdīr* ditemui dalam al-Quran hanya tiga kali dan semuanya dalam konteks pembicaraan tentang peredaran Matahari dan Bulan, yakni dalam QS. al-An'am/6: 96 di atas, QS, Yāsīn/36: 38:

وَٱلشَّمۡسُ تَجۡرِي لِمُسۡتَقَرّٖ لَّهَاۚ ذَٰلِكَ تَقۡدِيرُ ٱلۡعَزِيزِ ٱلۡعَلِيمِ

Dan Matahari berjalan ditempat peredarannya. Demikianlah ketetapan Yang Maha Perkasa lagi Maha Mengetahui (QS, Yāsīn/36: 38).[3]

dan QS. Fuṣṣilat/41: 12:

فَقَضَىٰهُنَّ سَبۡعَ سَمَٰوَاتٖ فِي يَوۡمَيۡنِ وَأَوۡحَىٰ فِي كُلِّ سَمَآءٍ أَمۡرَهَاۚ وَزَيَّنَّا ٱلسَّمَآءَ ٱلدُّنۡيَا بِمَصَٰبِيحَ وَحِفۡظٗاۚ ذَٰلِكَ تَقۡدِيرُ ٱلۡعَزِيزِ ٱلۡعَلِيمِ

Maka Dia menjadikannya tujuh langit dalam dua masa. Dia mewahyukan pada tiap-tiap langit urusannya. Dan Kami hiasi langit yang dekat dengan bintang-bintang yang cemerlang dan Kami memeliharanya dengan sebaik-baiknya. Demikianlah ketentuan Yang Maha Perkasa lagi Maha Mengetahui (QS. Fuṣṣilat/41: 12).[4]

---

[3] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 629.
[4] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 685.

Kata *taqdīr* dalam ayat-ayat tersebut digunakan untuk makna pengaturan dan ketentuan yang sangat teliti dalam konteks penciptaan alam semesta. Oleh karena itu, kata ini dalam al-Quran hanya digunakan untuk menunjukkan konsistensi hukum-hukum Allah yang berlaku di alam raya. Menurut Shihab kata *takdīr* mengandung arti: 1) menjadikan sesuatu memiliki kadar serta sistem tertentu dan teliti. 2) menetapkan kadar sesuatu, baik yang berkaitan dengan materi, maupun waktu. Penggunaan kata *takdīr* diperkuat oleh kata *al'Azīz* / Maha Perkasa dan *al-'Alīm* / Maha Mengetahui pada akhir ayat bertujuan menjelaskan bahwa pengaturan Allah terhadap benda langit seperti Matahari yang demikian besar, dapat terlaksana karena Dia Maha Perkasa sehingga semua tunduk kepada-Nya, dan Maha Mengetahui sehingga pengaturan-Nya sangat teliti dan mengagumkan.[5] Dengan demikian, peredaran Matahari dan Bulan sudah ditentukan kadar peredarannya, sehingga akan selalu beredar secara konsisten berdasarkan garis edarnya.

Selanjutnya kata *takdīr* dalam QS. al-An'am/6: 96 di atas menguatkan kata *husbānā*. Kata *Husbānā*

[5] Quraish Shihab, *Tafsir al-Misbah*, Vol. 11 (Jakarta: Lentera Hati, 2002), 152-153. Lihat juga: Ahmad Mushthafa al-Marāghy, *Tafsir al-Marāghy*, jil. 8, (Beirut-libanon: Dar al-Fikr, t.t.), 112-113.

secara bahasa terambil dari kata *hisābā* yang berarti kesempurnaan, sehingga ayat tersebut berarti perhitungan yang sempurna dan teliti. Kemudian hasil perhitungan tersebut ditetapkan sebagai takdir. Berkaitan dengan ketelitian dalam penciptaan perhitungan peredaran benda-benda langit juga disinggung dalam firman-Nya yang lain, misalnya dalam QS al-Rahmān/55: 5.[6]

ٱلشَّمۡسُ وَٱلۡقَمَرُ بِحُسۡبَانٍ

Matahari dan Bulan (beredar) menurut perhitungan (QS. ar- Rahmān/55: 5).[7]

Mengacu pada dua ayat ini dapat ditarik kesimpulan bahwa peredaran benda-benda langit sedemikian konsisten dan pasti, sehingga tidak mungkin terjadi tabrakan antar benda langit yang satu dengan benda langit yang lainnya. Menurut Shihab, karena peredaran Matahari dan Bulan sangat konsisten dan pasti, maka dapat dijadikan sebagai alat untuk melakukan

[6] Shihab, *Tafsir al-Misbah*, Vol. 13, 280-281. Dalam hal ini al-Qasimy menambahkan bahwa ketelitian perhitungan tersebut tidak hanya untuk kepentingan ibadah namun juga untuk kepentingan muamalah antar sesama hamba Allah. Muhammad Jamal ad-Din al-Qasimy, *Tafsir al-Qasimy*, jil. 3, (Beirut-libanon: Muassasah at-Tarikh al-Araby, 1994), 383.

[7] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 773.

perhitungan waktu, tahun, bulan, minggu dan hari bahkan menit dan detik.[8]

Dalam peredaran Bulan ditetapkan *manzilah-manzilah*. Ini ditemui dalam QS.Yunus/10: 5:

هُوَ ٱلَّذِى جَعَلَ ٱلشَّمْسَ ضِيَآءً وَٱلْقَمَرَ نُورًا وَقَدَّرَهُۥ مَنَازِلَ لِتَعْلَمُواْ عَدَدَ ٱلسِّنِينَ وَٱلْحِسَابَ ۚ مَا خَلَقَ ٱللَّهُ ذَٰلِكَ إِلَّا بِٱلْحَقِّ ۚ يُفَصِّلُ ٱلْأَيَٰتِ لِقَوْمٍ يَعْلَمُونَ

Dialah yang menjadikan Matahari bersinar dan Bulan bercahaya dan ditetapkan-Nya manzilah-manzilah (tempat-tempat) bagi perjalanan Bulan itu, supaya kamu mengetahui bilangan tahun dan perhitungan (waktu). Allah tidak menciptakan yang demikian itu melainkan dengan hak. Dia menjelaskan tanda-tanda (kebesaran-Nya) kepada orang-orang yang mengetahui (QS. Yunus/10: 5).[9]

dan QS Yāsīn/36: 39:

وَٱلْقَمَرَ قَدَّرْنَٰهُ مَنَازِلَ حَتَّىٰ عَادَ كَٱلْعُرْجُونِ ٱلْقَدِيمِ

Dan telah Kami tetapkan bagi Bulan *manzilah-manzilah*, sehingga (setelah dia sampai ke *manzilah* yang terakhir) kembalilah dia sebagai bentuk tandan yang tua. (QS. Yāsīn/36: 39).[10]

---

[8] Shihab, *Tafsir al-Misbah*, Vol. 5, 334. Lihat juga: Muhammad bin Ahmad al-Anshary al-Qurthuby, *al-Jami' li Ahkam al-Qur'an,* jil. 4, (Beirut-libanon: al-Maktabah al-'Ashriyah, 2009), 31.

[9] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 280.

[10] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 629.

Berdasarkan kedua ayat ini dapat dipahami bahwa Bulan memiliki *manzilah-manzilah* (fase-fase) dalam peredarannya. Dalam ayat tersebut dikatakan *qaddarahu manāzilah*. Al-Ṭabari mengatakan bahwa kata *manāzilah* pada ayat di atas hanya untuk Bulan saja, bukan untuk Matahari . Dia beragumentasi bahwa perhitungan *syahr* dan *sinīn* hanya dapat diketahui dengan *qamar*.[11] Dengan demikian, dapat dipahami bahwa Bulan memiliki *manzilah-manzilah* dalam perjalanannya. Karena Bulan memiliki *manzilah-manzilah*, maka dapat dilihat dari Bumi setiap malam dalam bentuk yang berbeda-beda, sehingga ada Bulan (*hilāl*) dan ada Bulan (*qamar*). Oleh karena itu, akan melahirkan sistem perhitungan atau penanggalan Bulan Kamariah.

Isyarat *manzilah* yang dimiliki oleh Bulan diperkuat oleh hasil penelitian yang menyatakan bahwa perjalanan Bulan dari Bulan mati (*muhaq*) sampai dengan Bulan purnama dan menuju Bulan mati lagi memiliki fase-fase antara lain: 1). Bulan baru / Bulan mati, 2) Kuartir pertama 3) Bulan purnama 4) Kuartir

[11] Al-Ṭabarī , *Jāmi' al-Bayān*, Jil. 6, 532.

ketiga, yakni ketika Bulan beredar ke arah perempat ketiga.[12]

Ayat yang berkaitan dengan peredaran Bulan selain dijelaskan dalam dalam QS. Yāsīn/36: 39 di atas juga diuraikan dalam QS. Yāsīn/36: 37-38.

وَءَايَةٌ لَّهُمُ ٱلَّيۡلُ نَسۡلَخُ مِنۡهُ ٱلنَّهَارَ فَإِذَا هُم مُّظۡلِمُونَ .
وَٱلشَّمۡسُ تَجۡرِي لِمُسۡتَقَرّٖ لَّهَاۚ ذَٰلِكَ تَقۡدِيرُ ٱلۡعَزِيزِ ٱلۡعَلِيمِ

Dan suatu tanda (kebesaran Allah) bagi mereka adalah malam; Kami tanggalkan siang dari (malam) itu, maka seketika itu mereka (berada dalam) kegelapan. Dan Matahari berjalan ditempat peredarannya. Demikianlah ketetapan Yang Maha Perkasa lagi Maha Mengetahui (QS, Yāsīn/36: 37-38).[13]

Menurut Shihab, ayat 37 tersebut mengilustrasikan Bumi dalam keadaan gelap. Kemudian, Matahari memancarkan sinarnya ke Bumi, maka bagian tertentu dari Bumi diliputi oleh sinarnya. Sinar Matahari itu diilustrasikan dengan kulit dan malam diilustrasikan dengan jasmani binatang yang tertutup kulit. Lalu sedikit demi sedikit sinar itu diambil dan dikeluarkan bagaikan binatang yang dikuliti. Setiap saat, berpisah kulit itu dari jasmaninya, setiap itu pula kegelapan muncul, lalu

[12] A. Kadir, *Formula Baru Ilmu Falak*, (Jakarta: Amzah, 2012), 33.

[13] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 629.

setelah selesai pengulitan yakni setelah posisi Matahari meninggalkan Bumi karena peredaran keduanya, maka kegelapan pun menutupi Bumi. Ayat 38 mengandung makna: 1) Matahari bergerak menuju ke tempat perhentiannya setiap hari. 2) Matahari bergerak terus-menerus sampai waktu yang ditetapkan Allah untuk perhentian geraknya, yakni pada saat dunia akan kiamat.[14]

Zuhaili menjelaskan bahwa Bulan dan Matahari tidak mungkin bertemu. Keduanya, menurut Zuhaili memiliki tempat dan kekuasaan. Kekuasaan Bulan pada malam hari dan kekuasaan Matahari pada siang hari. Karena masing-masing memiliki tempat kekuasaan, maka Matahari dan Bulan akan selalu beredar pada garis edarnya dan tidak akan saling bertabrakan.[15] Hal ini didasarkan pada QS. Yāsīn/36: 40.

لَا ٱلشَّمۡسُ يَنۢبَغِي لَهَآ أَن تُدۡرِكَ ٱلۡقَمَرَ وَلَا ٱلَّيۡلُ سَابِقُ ٱلنَّهَارِۚ وَكُلّٞ فِي فَلَكٖ يَسۡبَحُونَ .

Tidaklah mungkin bagi Matahari mendapatkan Bulan dan malam pun tidak dapat mendahului siang. Dan masing-masing beredar pada garis edarnya (QS. Yāsīn/36: 40).[16]

---

[14] Shihab, *Tafsir al-Mishbah*, Vol. 11, 152.

[15] Wahbah Zuhaili, *Tafsīr al-Munīr*, Jilid 23 (Beirut-libanon: Dar al-Fikr al-Ma'ashir, 1998), 17.

[16] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 629.

Sementara, Shihab dengan bertolak dari kata *yasbahūn* pada QS. Yāsīn/36: 40 tersebut, menjelaskan bahwa ruang angkasa diibaratkan samudera luas, di mana benda-benda langit diibaratkan ikan-ikan yang berenang di lautan lepas.[17]

Berdasarkan pendapat di atas dapat dipahami bahwa semua benda-benda di langit bergerak dan beredar, tak terkecuali Matahari. Dalam peredarannya, sangat mustahil terjadi tabrakan, karena telah memiliki keteraturan sistem.

Peredaran Matahari dan Bulan seperti yang dikemukakan di atas dapat terjadi karena semua benda-benda yang ada di langit telah ditundukkan oleh Allah, sebagaimana dinyatakan dalam ayat berikut ini.

ٱللَّهُ ٱلَّذِى رَفَعَ ٱلسَّمَٰوَٰتِ بِغَيْرِ عَمَدٍ تَرَوْنَهَا ۖ ثُمَّ ٱسْتَوَىٰ عَلَى ٱلْعَرْشِ ۖ وَسَخَّرَ ٱلشَّمْسَ وَٱلْقَمَرَ ۖ كُلٌّ يَجْرِى لِأَجَلٍ مُّسَمًّى ۚ يُدَبِّرُ ٱلْأَمْرَ يُفَصِّلُ ٱلْأَيَٰتِ لَعَلَّكُم بِلِقَآءِ رَبِّكُمْ تُوقِنُونَ

[17] Kata *yasbahun* menurut Quraish Shihab mempunyai makna pada mulanya berarti mereka berenang. Menurut Shihab ruang angkasa diibaratkan oleh al-Qur an dengan samudera yang besar. Benda-benda langit diibaratkan dengan ikan-ikan yang berenang dilautan lepas. Allah melukiskan benda-benda itu dengan kata yang digunakan bagi yang berakal mereka berenang. Ini mengisyaratkan ketundukan benda-benda langit itu kepada ketentuan dan takdir yang ditetapkan Allah atasnya. Shihab, *Tafsir al-Mishbah*, Vol. 11, 154.

Allah-lah Yang meninggikan langit tanpa tiang (sebagaimana) yang kamu lihat, kemudian Dia bersemayam di atas ´Arasy, dan menundukkan Matahari dan Bulan. Masing-masing beredar hingga waktu yang ditentukan. Allah mengatur urusan (makhluk-Nya), menjelaskan tanda-tanda (kebesaran-Nya), supaya kamu meyakini pertemuan(mu) dengan Tuhanmu (QS. ar Ra'd/13: 2).[18]

وَسَخَّرَ لَكُمُ ٱلشَّمۡسَ وَٱلۡقَمَرَ دَآئِبَيۡنِۖ وَسَخَّرَ لَكُمُ ٱلَّيۡلَ وَٱلنَّهَارَ

Dan Dia telah menundukkan (pula) bagimu Matahari dan Bulan yang terus menerus beredar (dalam orbitnya); dan telah menundukkan bagimu malam dan siang bagimu. (QS. Ibrahim/14: 33).[19]

Kata سَخَّر pada ayat-ayat tersebut digunakan dalam arti menundukkan sesuatu agar mudah digunakan oleh pihak lain. Sesuatu yang ditundukkan oleh Allah tidak lagi memiliki pilihan dan dengan demikian manusia yang mempelajari dan mengetahui sifat-sifat sesuatu itu akan merasa tenang menghadapinya karena yang ditundukkan tidak akan membangkang. Dari sini, diperoleh "kepastian" hukum-hukum alam.[20] Sayyid

[18] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 336.
[19] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 350-351.
[20] Shihab, *Tafsir al-Mishbah*, Vol. 6, 377-378.

Qutub menambahkan bahwa Allah menundukkan Matahari dan Bulan tersebut dengan menciptakan hukum alam bagi keduanya untuk kemanfaatan manusia dalam kehidupannya.[21]

أَلَمۡ تَرَ أَنَّ ٱللَّهَ يُولِجُ ٱلَّيۡلَ فِي ٱلنَّهَارِ وَيُولِجُ ٱلنَّهَارَ فِي ٱلَّيۡلِ وَسَخَّرَ ٱلشَّمۡسَ وَٱلۡقَمَرَ كُلٌّ يَجۡرِيٓ إِلَىٰٓ أَجَلٖ مُّسَمًّى وَأَنَّ ٱللَّهَ بِمَا تَعۡمَلُونَ خَبِيرٞ

Tidakkah kamu memperhatikan, bahwa sesungguhnya Allah memasukkan malam ke dalam siang dan memasukkan siang ke dalam malam dan Dia tundukkan Matahari dan Bulan masing-masing berjalan sampai kepada waktu yang ditentukan, dan sesungguhnya Allah Maha Mengetahui apa yang kamu kerjakan (QS. Lukman/31: 29).[22]

Sebagaimana penafsiran kata *syakhkhara* pada ayat sebelumnya, menurut Shihab bahwa dalam QS. Lukman/31: 29 tersebut Allah menundukkan Matahari dan Bulan dengan menetapkan untuk keduanya hukum-hukum alam yang mengatur secara teliti perjalannya dan masing-masing berjalan sampai dengan waktu yang ditentukan. Matahari terbit dari sebelah timur ke barat,

---

[21] Sayyid Qutub, *Fiy Dhilal al-Qur'an,* jil.4, (Beirut-Libanon: Dar as-Syuruq, 1992), 2107-2108.

[22] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 584.

Bulan beredar mengelilingi Matahari dengan kecepatan 30 km per detik dan menyelesaikan sekali putaran sekitar 365,25 hari dan ketentuan-ketentuan lain berkaitan dengan peredaran Matahari dan Bulan.[23] Hal ini dikuatkan pula dengan firman Allah berikut ini.

يُولِجُ ٱلَّيۡلَ فِي ٱلنَّهَارِ وَيُولِجُ ٱلنَّهَارَ فِي ٱلَّيۡلِ وَسَخَّرَ ٱلشَّمۡسَ وَٱلۡقَمَرَ كُلٌّ يَجۡرِي لِأَجَلٖ مُّسَمًّىۚ ذَٰلِكُمُ ٱللَّهُ رَبُّكُمۡ لَهُ ٱلۡمُلۡكُۚ وَٱلَّذِينَ تَدۡعُونَ مِن دُونِهِۦ مَا يَمۡلِكُونَ مِن قِطۡمِيرٍ

Dia memasukkan malam ke dalam siang dan memasukkan siang ke dalam malam dan menundukkan Matahari dan Bulan, masing-masing berjalan menurut waktu yang ditentukan. Yang (berbuat) demikian itulah Allah Tuhanmu, kepunyaan-Nya-lah kerajaan. Dan orang-orang yang kamu seru (sembah) selain Allah tiada mempunyai apa-apa walaupun setipis kulit ari (QS. Fāṭir/35: 13).[24]

Senada dengan penafsiran ayat sebelumnya, as-Shabuny menafsiri *sakhkhara* pada QS. Fāṭir/35: 13 tersebut dengan menyatakan bahwa Allah menundukkan Matahari dan Bulan untuk kemaslahatan hamba-Nya. Keduanya berjalan dan beredar sesuai dengan garis edar

[23] Shihab, *Tafsir al-Mishbah*, Vol. 10, 331.
[24] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 618.

masing-masing sesuai dengan ketentuan Allah sampai waktu yang telah ditentukan-Nya yaitu hari kiamat.[25]

خَلَقَ ٱلسَّمَٰوَٰتِ وَٱلۡأَرۡضَ بِٱلۡحَقِّۖ يُكَوِّرُ ٱلَّيۡلَ عَلَى ٱلنَّهَارِ وَيُكَوِّرُ ٱلنَّهَارَ عَلَى ٱلَّيۡلِۖ وَسَخَّرَ ٱلشَّمۡسَ وَٱلۡقَمَرَۖ كُلٌّ يَجۡرِي لِأَجَلٖ مُّسَمًّىۗ أَلَا هُوَ ٱلۡعَزِيزُ ٱلۡغَفَّٰرُ

Dia menciptakan langit dan Bumi dengan (tujuan) yang benar; Dia menutupkan malam atas siang dan menutupkan siang atas malam dan menundukkan Matahari dan Bulan, masing-masing berjalan menurut waktu yang ditentukan. Ingatlah Dialah Yang Maha Perkasa lagi Maha Pengampun (QS. al-Zumar/39: 5).[26]

Mengacu pada beberapa ayat tersebut, dapat ditarik kesimpulan: 1) konsistensi peredaran benda-benda langit terjadi karena masing-masing benda-benda langit telah ditentukan tempat edarnya. 2) Konsistensi peredaran benda-benda langit terjadi karena setiap benda langit telah ditentukan waktu beredarnya. 3) Konsistensi peredaran benda langit dapat terjadi karena setiap benda langit telah ditundukkan oleh Allah.

Konsistensi peredaran benda langit tersebut berdampak dalam bidang ibadah umat Islam. Hal ini

---

[25] Muhammad Ali as-Shabuny, *Shafwat at-Tafasir*, jil. 2, (Beirut-Libanon: Dar ar-Rasyad, 1988), 569.

[26] Departemen Agama RI, *al-Qur'an dan Terjemahnya*, 658.

disebabkan peribadatan umat Islam didasarkan pada regularitas pergerakan benda langit seperti Matahari dan Bulan,[27] terutama penentuan waktu shalat gerhana Matahari.

2. Teori Peredaran Benda Langit

Sejauh penelusuran penulis ada dua teori tentang peredaran benda langit, dalam hal ini adalah Matahari, Bumi dan Bulan, yaitu teori geosentris dan terori heliosentris. Teori geosentris yang digagas oleh Ptolomeus menempatkan Bumi sebagai pusat peredaran planet-planet dan Matahari sedangkan teori heliosentris yang ditemukan oleh Copernicus menyatakan bahwa Matahari sebagai pusat tata surya.

Teori heliosentris Nicolaus Copernicus ini juga mendapat perhatian besar dari para filosof sesudahnya. Setelah melakukan pengamatan dan penelitian yang panjang dan mendalam, mereka membenarkan, mendukung dan menyempurnakan teori heliosentris Nicolas Copernicus tersebut. Mereka di antaranya adalah Isaac Newton (1642-1727 M), Galileo Galilei (1564-1642 M) dan Johannes Kepler (1571-1630 M).[28]

---

[27] Hendro Setyanto, *Membaca Langit,* (Jakarta: al-Ghuraba, 2008), 45.

[28] Azhari, *Ilmu Falak*, 19.

Terkait dengan teori heliosentris, Slamet Hambali[29] melakukan penelitian terkait beberapa ayat yang menjelaskan tentang gerak Matahari, Bulan dan Bumi, yaitu surat Yūnus: 5, surat Yāsīn: 38, dan surat al-Naml: 88. Dalam penelitian tersebut, disimpulkan bahwa Teori Heliocentris Nicolas Copernicus telah membuktikan kebenaran ilmu pengetahuan yang dibawa oleh al-Qur'an. Ayat-ayat al-Qur'an secara tegas menunjukkan bahwa tidak ada pertentangan dengan ilmu pengetahuan modern, khususnya yang berkaitan dengan ilmu astronomi, walaupun al-Qur'an diturunkan jauh sebelumnya, yaitu pada abad ke-7 Masehi.

Dalam penelitian tersebut Slamet Hambali mengutip pendapat Aḥmad Muṣṭafā al-Marāghī dalam kitab tafsirnya Tafsīr al-Marāghī ketika menafsirkan surat Yasin: 38 terkait dengan argumentasi al-Qur'an tentang kebenaran astronomi Islam. Melalui analisis terhadap tiga bukti peristiwa astronomis, yaitu: pergantian siang dan malam, gerak hakiki Matahari, dan gerak semu Matahari, al-Marāghī sampai pada kesimpulan bahwa ada kesesuaian yang sangat presisi antara teori astronomi Islam yang diajarkan oleh al-Qur'an dengan teori

[29] Slamet Hambali, "Astronomi Islam dan Teori Heliocentris Nicolaus Copernicus" dalam *Al-Ahkam*, Volume 23, Nomor 2, Oktober 2013.

astronomi modern. Kesesuaian ayat-ayat al-Qur'an dengan sains modern menurut al-Marāghī juga merupakan bagian dari mu'jizat al-Qur'an yang benar-benar dapat dibuktikan.

3. Sistem Koordinat Benda Langit

Sebagaimana diuraikan di atas bahwa peredaran benda-benda langit telah ditentukan waktu beredarnya oleh Allah, para ilmuwan menterjemahkannya melalui sistem koordinat[30]. Sistem koordinat bisa digunakan untuk mengetahui posisi benda–benda langit. Setiap sistem koordinat memiliki koordinat masing–masing. Posisi benda langit seperti Matahari dapat dinyatakan dalam sistem koordinat tertentu. Selanjutnya nilainya dapat diubah ke dalam sistem koordinat yang lain melalui suatu transformasi koordinat.

Berikut ini dibahas beberapa sistem koordinat yang penting dalam penentuan posisi benda-benda langit, yaitu: Sistem Koordinat Ekliptika Heliosentrik (*Heliocentric Ecliptical Coordinate*), Sistem Koordinat Ekliptika Geosentrik (*Geocentric Ecliptical Coordinate*),

[30] Uraian tentang Sistem Koordinat banyak diuraikan dalam buku-buku astronomi. Dalam tulisan ini banyak dikutip uraian dari Rinto Anugraha, *Mekanika Benda Langit*, (Yogyakarta: Jurusan Fisika Fakultas MIPA Universitas Gajah Mada, 2012), 41-55. Uraian dari buku lain, lihat misalnya: Hannu Karttunen, et.al., *Fundamental Astronomy*, (New York: Springer, 1996), 15-22, Oliver Montenbruck, *Astronomy on the Personal Computer*, (New York: Springer, 1994), 7-33.

Sistem Koordinat Ekuator Geosentrik (*Geocentric Equatorial Coordinate*) dan Sistem Koordinat Horison (*Horizontal Coordinate*).

Keempat sistem koordinat di atas termasuk ke dalam koordinat bola. Pembagian sistem koordinat di atas berasal dari benda langit manakah yang dijadikan pusat koordinat, apakah bidang datar sebagai referensi serta bagaimana cara mengukur posisi benda langit lainnya.

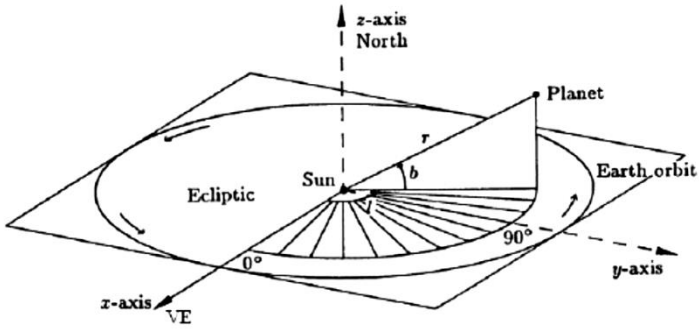
Sistem Koordinat Ekliptika Heliosentrik dan Sistem Koordinat Ekliptika Geosentrik sebenarnya identik. Yang membedakan keduanya hanyalah manakah yang menjadi pusat koordinat. Pada Sistem Koordinat Ekliptika Heliosentrik, yang menjadi pusat koordinat adalah Matahari (*helio* = Matahari). Sedangkan pada Sistem Koordinat Ekliptika Geosentrik, yang menjadi pusat koordinat adalah Bumi (*geo* = Bumi). Karena itu keduanya dapat digabungkan menjadi Sistem Koordinat Ekliptika. Pada Sistem Koordinat Ekliptika, yang menjadi bidang datar sebagai referensi adalah bidang orbit Bumi mengitari Matahari (heliosentrik) yang juga sama dengan bidang orbit Matahari mengitari Bumi (geosentrik).

a. Sistem Koordinat Ekliptika Heliosentrik

Pada koordinat Ekliptika Heliosentrik ini, Matahari (*sun*) menjadi pusat koordinat. Benda

langit lainnya seperti Bumi (*earth*) dan planet bergerak mengitari Matahari. Bidang datar yang identik dengan bidang xy adalah bidang ekliptika yaitu bidang Bumi mengitari Matahari.

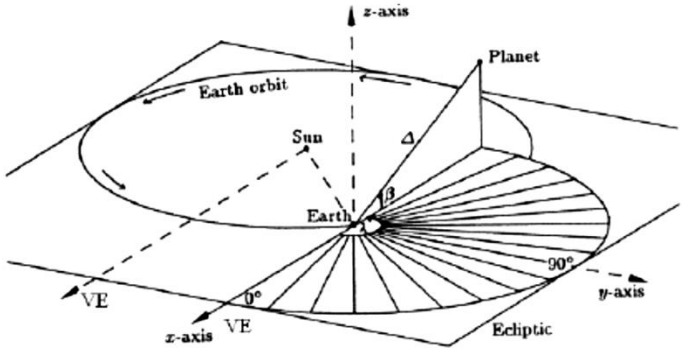
1) Pusat koordinat: Matahari (*Sun*).
2) Bidang datar referensi: Bidang orbit Bumi mengitari Matahari (bidang ekliptika) yaitu bidang xy.
3) Titik referensi: *Vernal Ekuinoks* (VE), didefinisikan sebagai sumbu x.
4) Koordinat:
   - *r* = jarak (radius) benda langit ke Matahari
   - *l* = sudut bujur ekliptika (*ecliptical longitude*), dihitung dari VE berlawanan arah jarum jam
   - *b* = sudut lintang ekliptika (*ecliptical latitude*), yaitu sudut antara garis penghubung benda langit–Matahari dengan bidang ekliptika.



Gambar 2.1. Sistem Koordinat Ekliptika Heliosentrik

b. Sistem Koordinat Ekliptika Geosentrik

Pada sistem koordinat Ekliptika Geosentrik, Bumi menjadi pusat koordinat. Matahari dan planet–planet lainnya nampak bergerak mengitari Bumi. Bidang datar xy adalah bidang ekliptika, sama seperti pada ekliptika heliosentrik.



Gambar 2.2. Sistem Koordinat Ekliptika Geosentrik

1) Pusat Koordinat: Bumi (*Earth*)
2) Bidang datar referensi: Bidang Ekliptika (Bidang orbit Bumi mengitari Matahari, yang sama dengan bidang orbit Matahari mengitari Bumi) yaitu bidang xy.
3) Titik referensi: *Vernal Ekuinoks* (VE) yang didefinisikan sebagai sumbu x.
4) Koordinat:
   - *Delta* (Δ) = jarak benda langit ke Bumi (seringkali diabaikan atau tidak perlu dihitung).
   - *Lambda* (λ) = Bujur Ekliptika (*Ecliptical Longitude*) benda langit menurut Bumi, dihitung dari VE.
   - *Beta* (β) = Lintang Ekliptika (*Ecliptical Latitude*) benda langit menurut Bumi yaitu sudut antara garis penghubung benda langit–Bumi dengan bidang ekliptika.

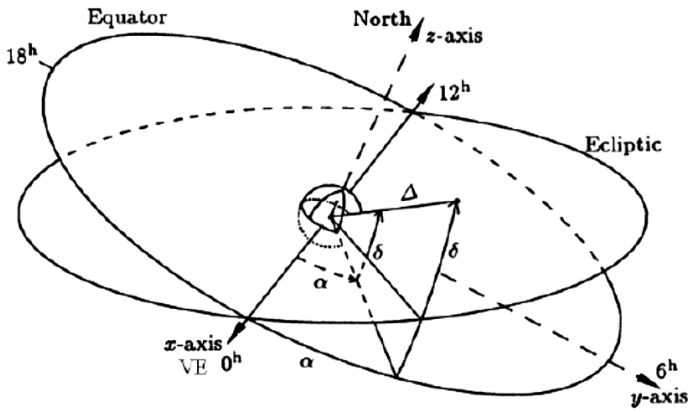
c. Sistem Koordinat Ekuator Geosentrik

Ketika Bumi bergerak mengitari Matahari di bidang Ekliptika, Bumi juga sekaligus berotasi terhadap sumbunya. Penting untuk diketahui, sumbu rotasi Bumi tidak sejajar dengan sumbu

bidang ekliptika. Atau dengan kata lain, bidang ekuator tidak sejajar dengan bidang ekliptika, tetapi membentuk sudut kemiringan (*epsilon*, ε) sebesar kira–kira 23,5 derajat. Sudut kemiringan ini sebenarnya tidak bernilai konstan sepanjang waktu. Nilainya semakin lama semakin mengecil.

1) Pusat koordinat: Bumi
2) Bidang datar referensi: bidang ekuator, yaitu bidang datar yang mengiris Bumi menjadi dua bagian melewati garis khatulistiwa.
3) Koordinat:
   - *Delta* (Δ) = jarak benda langit ke Bumi.
   - *Alpha* (α) = *Right Ascension* = Sudut antara VE dengan proyeksi benda langit pada bidang ekuator, dengan arah berlawanan jarum jam. Biasanya *alpha* bukan dinyatakan dalam satuan derajat, tetapi jam (*hour* disingkat *h*). Satu putaran penuh = 360 derajat = 24 jam = 24 h. Karena itu jika *Alpha* dinyatakan dalam derajat, maka dibagi dengan 12 untuk memperoleh satuan derajat. Titik VE menunjukkan 0 h.
   - *Delta* (δ) = *Declination* (Deklinasi) = Sudut antara garis hubung benda langit–

Bumi dengan bidang ekliptika. Nilainya mulai dari –90 derajat (selatan) hingga 90 derajat (utara). Pada bidang ekuator, deklinasi = 0 derajat.



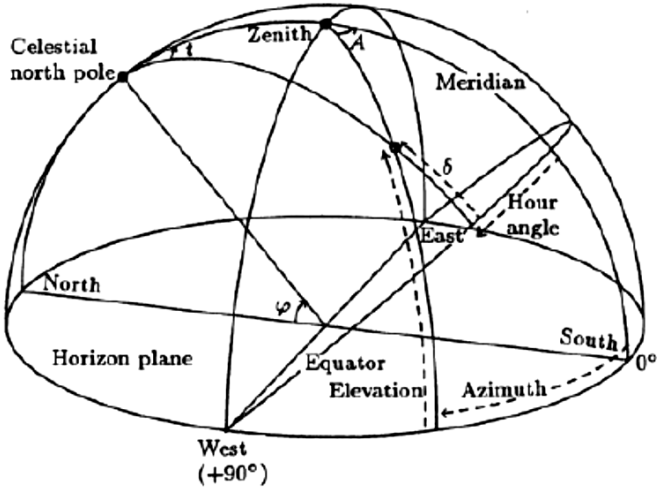
Gambar 2.3. Sistem Koordinat Ekuator Geosentrik

Seringkali, *Alpha* (*right ascension*) dinyatakan dalam bentuk H (*hour angle*). Hubungan antara *Alpha* dengan H adalah H = LST - α. Di mana, LST adalah *Local Sidereal Time*.

d. Sistem Koordinat Horisontal

Pada sistem koordinat ini, pusat koordinat adalah posisi pengamat (bujur dan lintang) yang terletak di permukaan Bumi. Kadang–kadang, ketinggian pengamat dari permukaan Bumi juga ikut diperhitungkan. Bidang datar yang menjadi referensi seperti bidang xy adalah bidang horison

(bidang datar di sekitar pengamat di permukaan Bumi).



Gambar 2.4. Sistem Koordinat Horison

1) Pusat koordinat: Pengamat di permukaan Bumi
2) Bidang datar referensi: Bidang horison (*Horizon plane*)
3) Koordinat:
   - h (*Altitude*) = sudut ketinggian benda langit dari bidang horison. h = 0 derajat berarti benda di bidang horison. h = 90 derajat dan –90 derajat masing–masing menunjukkan posisi di titik *zenith* (tepat di atas kepala) dan nadir (tepat di bawah kaki).

- A (*Azimuth*) = sudut antara arah utara dengan proyeksi benda langit ke bidang horison.

4. Transformasi Sistem Koordinat

Suatu sistem koordinat dengan sistem koordinat lainnya dapat dihubungkan melalui transformasi koordinat. Misalnya, dari algoritma untuk menghitung posisi Bulan menurut sistem koordinat ekliptika geosentrik, kita dapat menentukan jarak Bulan dari pusat Bumi, sudut *lambda* dan *beta*. Selanjutnya, sudut *lambda* dan *beta* ditransformasi untuk mendapat sudut *alpha* dan *delta* dalam sistem koordinat ekuator geosentrik. Dari *alpha* dan *beta*, serta memperhitungkan posisi pengamat (bujur dan lintang) dan waktu saat pengamatan/penghitungan, maka sudut ketinggian (*altitude*) dan *azimuth* Bulan menurut sistem koordinat horison dapat diketahui dengan tepat.

a. Transformasi Koordinat dari Ekliptika Geosentrik (*Lambda, Beta*) ke Ekuator Geosentrik (*Alpha, Delta*).

Dalam koordinat ekliptika geosentrik, sudut lambda adalah bujur ekliptika (*ecliptical longitude*) yang dihitung dari *vernal ekuinoks* (VE), sedangkan sudut *beta* adalah lintang

ekliptika (*ecliptical latitude*) atau sudut ketinggian yang dihitung dari bidang ekliptika.

Vernal ekuinoks ditunjukkan oleh *lambda* = *beta* = 0 derajat. Semua benda yang terletak pada bidang ekliptika memiliki nilai *beta* = 0. Nilai *beta* positif menunjukkan posisi benda di atas (arah utara) bidang ekliptika. *Beta* negatif berarti benda di bawah bidang ekliptika.

Menurut sistem koordinat ekliptika heliosentrik, Bumi mengitari Matahari di bidang ekliptika. Sebaliknya menurut sistem koordinat ekliptika geosentrik, Matahari nampak bergerak mengitari Bumi yang tentu saja juga di bidang ekliptika yang sama. Karena itu, baik Bumi maupun Matahari secara praktis memiliki nilai *beta* = 0 derajat. Dikatakan secara praktis memiliki nilai *beta* = 0, karena sebenarnya nilainya tidak benar–benar tepat sama dengan 0. Akibat pengaruh gravitasi dari planet–planet lain, nilai maksimum *beta* untuk Bumi atau Matahari bisa mencapai (positif/negatif) satu detik busur atau 1/3600 derajat.

Cara menentukan besar bujur dan lintang ekliptika suatu benda langit adalah sebagai berikut. Misal Bumi terletak di pusat koordinat (O) dan

benda langit adalah P. Proyeksikan atau tarik garis tegaklurus dari benda langit tersebut (P) ke bidang ekliptika. Titik proyeksi pada bidang ekliptika sebut saja titik A. Maka, *beta* adalah besar sudut P–O–A. Jika P di atas bidang ekliptika, *beta* positif. Jika P di bawah bidang ekliptika, *beta* negatif. Sedangkan *lambda* adalah besar sudut VE–O–A, dimana VE adalah *vernal ekuinoks*.

Pada bidang ekuator geosentrik, bidang yang menjadi referensi adalah bidang ekuator Bumi, yaitu bidang yang mengiris Bumi menjadi dua bagian sama besar yang melewati garis ekuator atau garis khatulistiwa. Dua koordinat sudut dalam koordinat ekuator geosentrik adalah sudut *alpha* dan *delta*. Sudut *alpha* (*right ascension*) mirip seperti bujur ekliptika (*lambda*), hanya saja di sini bidang referensinya adalah bidang ekuator. Sudut *delta* (*declination*) juga mirip seperti lintang ekliptika (*beta*), hanya saja di sini bidang referensinya juga bidang ekuator.

Karena itu cara menentukan sudut *alpha* dan *delta* mirip seperti pada sudut *lambda* dan *beta*, hanya saja bidang referensinya adalah bidang ekuator. Kembali untuk benda langit yang sama (P), jika P diproyeksikan tegaklurus ke bidang

ekuator, dan titik proyeksi di bidang ekuator adalah titik B, maka *delta* adalah sudut P–O–B, dimana O adalah pusat koordinat (Bumi). Sedangkan sudut *alpha* adalah sudut VE–O–B.

Sangat penting untuk diketahui, bidang ekuator tidak sejajar dengan bidang ekliptika, tetapi membentuk sudut kemiringan (*obliquity*) sebesar *epsilon* (ε) yang besarnya kira–kira sebesar 23,5 derajat. Sudut ini sebenarnya tidak konstan sepanjang masa, tetapi ada kecenderungan untuk terus mengecil. Saat ini, angka sudut kemiringan adalah sebesar 23,43808 derajat = 23:26:17 derajat = 23d 26m 17s (dibaca 23 derajat 26 menit busur 17 detik busur). Angka sudut kemiringan yang lebih akurat jika memperhitungkan faktor koreksi seperti nutasi (osilasi sumbu rotasi Bumi di sekitar nilai rata–ratanya) adalah 23,439607 derajat = 23:26:23 derajat.

Rumus transformasi koordinat dari Ekliptika Geosentrik (*Lambda, Beta*) ke Ekuator Geosentrik (*Alpha, Delta*) adalah sebagai berikut.

λ = lambda, β = beta

α = alpha, δ = delta

ε = epsilon

$$\tan(\alpha) = \frac{\sin(\lambda)\cos(\varepsilon) - \tan(\beta)\sin(\varepsilon)}{\cos(\lambda)}$$

$$\sin(\delta) = \sin(\beta)\cos(\varepsilon) + \sin(\lambda)\cos(\beta)\sin(\varepsilon)$$

b. Transformasi koordinat dari Ekuator Geosentrik (*Alpha, Delta*) ke Ekliptika Geosentrik (*Lambda, Beta*)

Ini adalah kebalikan dari transformasi koordinat sebelumnya. Rumus transformasi koordinat dari Ekuator Geosentrik (*Alpha, Delta*) ke Ekliptika Geosentrik (*Lambda, Beta*) adalah:

$$\tan(\lambda) = \frac{\sin(\alpha)\cos(\varepsilon) + \tan(\delta)\sin(\varepsilon)}{\cos(\alpha)}$$

$$\sin(\beta) = \sin(\delta)\cos(\varepsilon) - \sin(\alpha)\cos(\delta)\sin(\varepsilon)$$

c. Transformasi koordinat dari Ekuator Geosentrik (*Alpha, Delta*) ke Horison (h, A)

Pengamat di Bumi ingin mengetahui posisi benda langit dalam bentuk ketinggian benda tersebut dari horison dan arahnya. Dalam hal ini digunakan sistem koordinat horison dengan koordinat h dan A. Sudut h adalah ketinggian benda langit tersebut dari permukaan horison. Jika benda tersebut di atas horison maka sudut h positif. Sebaliknya jika benda tersebut di bawah horison, maka sudut h negatif.

Untuk perhitungan sudut h yang lebih teliti, faktor pembiasan cahaya oleh atmosfer karena perbedaan indeks bias lapisan atmosfer harus diperhitungkan. Akibat faktor ini, saat benda langit (seperti Matahari) terbit atau terbenam di ufuk, posisi sebenarnya berada sedikit di bawah ufuk atau h negatif.

Sudut antara arah utara dengan proyeksi posisi benda langit ke horison adalah sudut *azimuth*. Sedikit catatan, bahwa dalam berbagai referensi, sudut *azimuth* dihitung dari arah selatan. Untuk membedakannya, sudut *azimuth* dari arah utara adalah A. Sedangkan sudut *azimuth* dari arah selatan disebut As.

Dari sudut *Alpha* di ekuator geosentrik, biasanya dicari dahulu HA (*hour angle*). Hubungan antara HA dan *Alpha* adalah:

HA = LST – Alpha.

dimana LST adalah *Local Sidereal Time.*

Rumus transformasi koordinat dari Ekuator Geosentrik (*Alpha, Delta*) ke Horison (h, A) adalah

$$\sin(h) = \sin(\theta)\sin(\delta) + \cos(\theta)\cos(\delta)\cos(HA)$$

$$\tan(As) = \frac{\sin(HA)}{\cos(H)\sin(\theta) - \tan(\delta)\cos(\theta)}$$

d. Transformasi koordinat dari Horison (h, A) ke Ekuator Geosentrik (*Alpha, Delta*)

Pada transformasi koordinat ini, rumus yang digunakan adalah

$$As = A + 180$$

$$\sin(\delta) = \sin(\theta)\sin(h) - \cos(\theta)\cos(h)\cos(As)$$

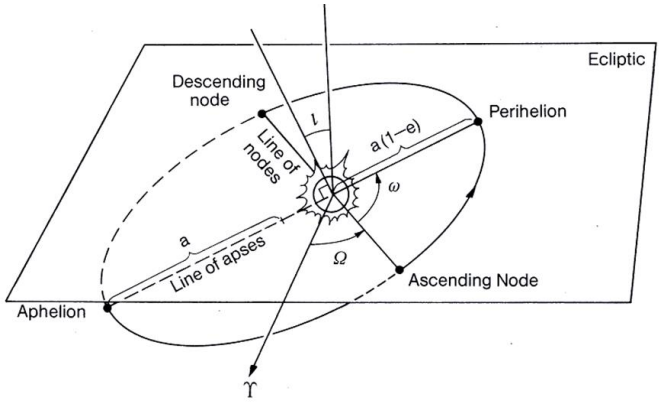
$$\tan(HA) = \frac{\sin(As)}{\cos(As)\sin(\theta) + \tan(h)\cos(\theta)}$$

$$\alpha = LST - HA$$

5. Elemen Orbit Benda Langit

Ukuran, bentuk dan arah dari orbit benda langit dapat diketahui ketika bisa ditentukan elemen orbit dari benda langit tersebut. Sebagaimana pendapat Jean Meeus yang mengatakan bahwa ada beberapa metode untuk menghitung posisi benda langit pada orbit ellips sekitar Matahari pada suatu saat tertentu, salah satunya dengan mengetahui elemen orbit benda langit[31]. Elemen orbit benda langit tersebut diuraikan dalam enam elemen orbit, yaitu :

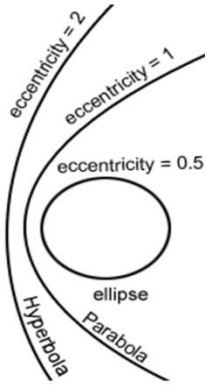
[31] Jean Meeus, *Astronomical Algorithms*, (Virginia: Willmann-Bell, Inc., 1991), 181.



Gambar 2.5. Geometri elemen orbit benda langit

a. *Semi-major axis (a)*, yaitu setengah sumbu utama dari orbit benda langit. Elemen ini menunjukkan karakter ukuran (*size*) dari benda langit tersebut.

b. *Eccentricity (e)*, yaitu ukuran yang menunjukkan sejauh mana penyimpangan bentuk orbit dari bentuk bulat. Nilai dari *eccentricity* berkisar antara 0 dan 1. Jika nilai $e = 0$ maka bentuk orbitnya adalah lingkaran, jika nilainya adalah $0 < e < 1$ maka bentuk orbitnya adalah ellips, jika nilai $e = 1$ maka bentuk orbitnya adalah parabola sedangkan jika nilai $e > 1$ maka bentuk orbitnya adalah hiperbola.



Gambar 2.6. Ilustrasi nilai *eccentricity* dan bentuk orbit

c. *Inclination (i)*, yaitu sudut antara bidang orbit benda langit dan bidang referensi. Misal *inclination* orbit Bulan adalah sudut yang dibentuk oleh bidang orbit Bulan mengelilingi Bumi dan bidang equator.

d. *Longitude of ascending node (Ω)*, yaitu sudut yang dihitung pada bidang refensi mulai dari *vernal equinox* (titik aries) sampai dengan *ascending node* (titik nodal naik) dari orbit.

e. *Argument of perihelion (ω)*, yaitu sudut yang diukur pada bidang orbit mulai dari *ascending node* ke titik *perihelion*.

f. *Time of perihelion (τ)*, yaitu waktu yang dibutuhkan oleh benda langit melalui bidang orbitnya menuju titik *perihelion.*

B. Tinjauan Umum tentang Gerhana Matahari

1. Pengertian Gerhana Matahari

Gerhana dalam bahasa arab dikenal dengan istilah *kusuf* dan *khusuf*. Pada dasarnya istilah *kusuf* dan *khusuf* digunakan untuk menyebut gerhana Matahari maupun gerhana Bulan. Kata *kusuf* lebih dikenal untuk menyatakan gerhana Matahari sedangkan kata *khusuf* untuk gerhana Bulan. Dalam bahasa Inggris dikenal dengan istilah *eclipse*. *Kusuf* artinya menutupi; menggambarkan adanya fenomena alam bahwa (dilihat dari Bumi) Bulan menutupi Matahari, sehingga terjadi gerhana Matahari. Adapun *khusuf* berarti memasuki; menggambarkan adanya fenomena alam bahwa Bulan memasuki bayangan Bumi, sehingga terjadi gerhana Bulan. [32] Dalam astronomi fenomena gerhana diartikan tertutupnya arah pandang pengamat ke benda langit oleh benda langit lainnya yang lebih dekat dengan pengamat.

Gerhana Matahari adalah peristiwa tertutupnya sinar Matahari oleh Bulan sebagian atau seluruhnya sehingga Matahari tidak tampak dari Bumi secara keseluruhan pada saat gerhana Matahari total dan sebagiannya pada saat gerhana sebagian. Terjadi pada saat siang hari pada saat konjungsi/*ijtima'*, yaitu pada

[32] Khazin, *Ilmu Falak*, 185.

saat Matahari, Bulan dan Bumi berada pada bujur astronomi yang sama serta bayangan Bulan akan mengenai Bumi.

Kedudukan bidang orbit Bulan mengelilingi Bumi membentuk sudut 5,1° terhadap bidang orbit Bumi mengelilingi Matahari (bidang ekliptika). Atau sering dikatakan pula bidang orbit Bulan mempunyai inklinasi 5,1° dari bidang ekliptika. Hal inilah yang menyebabkan tidak terjadinya gerhana Bulan setiap konjungsi maupun gerhana Matahari setiap *fullmoon* (Bulan purnama).[33]

2. Gerhana Matahari dalam Perspektif Islam

Peristiwa gerhana merupakan peristiwa alam biasa yang secara astronomis dapat dihitung dan diprediksi kapan akan terjadi. Peristiwa gerhana bukan tanda kelahiran atau kematian seseorang namun gerhana merupakan momen merenungkan kembali tanda Kemahabesaran Allah. Untuk itu umat Islam memberi makna akan kehadiran gerhana melalui ibadah berupa salat gerhana yang dilakukan secara sendirian maupun berjamaah di masjid-masjid atau mushalla serta memperbanyak takbir dan sedekah. Hal ini sejalan

[33] Djamhur Effendi, *Sekelumit Penanggalan Qomariah dan Gerhana Bulan*, http://www.nu.or.id diakses pada tanggal 23 Maret 2017. 13:40 WIB.

dengan yang diajarkan oleh Rasulullah dalam hadis di bawah ini.

أَنَّ الشَّمْسَ انْكَسَفَتْ عَلَى عَهْدِ رَسُوْلِ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ. فَقَامَ قِيَامًا شَدِيْدًا. يَقُوْمُ قَائِمًا ثُمَّ يَرْكَعُ. ثُمَّ يَقُوْمُ ثُمَّ يَرْكَعُ. ثُمَّ يَقُوْمُ ثُمَّ يَرْكَعُ رَكْعَتَيْنِ فِي ثَلَاثِ رَكَعَاتٍ وَأَرْبَعِ سَجْدَاتٍ. فَانْصَرَفَ وَقَدْ تَجَلَّتْ الشَّمْسُ. وَكَانَ إِذَا رَكَعَ قَالَ: "اَللهُ أَكْبَرُ" ثُمَّ يَرْكَعُ. وَإِذَا رَفَعَ رَأْسَهُ قَالَ: "سَمِعَ اللهُ لِمَنْ حَمِدَهُ" فَقَامَ فَحَمِدَ اللهَ وَأَثْنَى عَلَيْهِ. ثُمَّ قَالَ: "إِنَّ الشَّمْسَ وَالْقَمَرَ لَايَنْكَسِفَانِ لِمَوْتِ أَحَدٍ وَلَالِحَيَاتِهِ. وَلَكِنَّهُمَا مِنْ آيَاتِ اللهِ يُخَوِّفُ اللهُ بِهِمَا عِبَادَهُ. فَإِذَا رَأَيْتُمْ كُسُوْفًا فَاذْكُرُوْا اللهَ حَتَّى يَنْجَلِيَا". (رواه مسلم).[34]

Sesungguhnya terjadi gerhana Matahari pada zaman Rasulullah SAW. Kemudian beliau berdiri kemudian ruku' kemudian berdiri lagi dan ruku' lagi kemudian berdiri lagi kemudian ruku' dua kali dalam tiga rakaat dan empat sujud. Ketika beliau selesai shalat, Matahari telah pulih. Ketika ruku' beliau membaca "*Allohu akbar*" kemudian ruku'. Ketika mengangkat kepala (*i'tidal*) beliau mengucap "*samia Allah li man hamidah*". Kemudian beliau berdiri dan memuji Allah kemudian

[34] Muslim bin al-Ḥajjāj, *Ṣaḥīh Muslim*, Jilid I, (Libanon: Dar al-Fikr, 1992), 397-398.

bersabda: ‘Sesungguhnya Matahari dan Bulan tidak mengalami gerhana karena mati atau hidup seseorang, namun keduanya adalah sebagian tanda kebesaran Allah untuk memberikan rasa takut pada hamba-hambaNya. Maka jika kalian melihat gerhana maka dzikirlah padaNya sampai keduanya pulih dari gerhana’.(HR. Muslim).

Mitos seputar kematian ini dalam hadis nabi di atas dikaitkan dengan peristiwa meninggalnya anak laki-laki Rasulullah yang bernama Ibrahim pada saat masih kecil. Menurut Syamsul Anwar[35], dalam hadis tersebut Nabi saw menjelaskan bahwa peristiwa gerhana itu adalah peristiwa alam yang natural yang menunjukkan kebesaran Allah dan tidak ada kaitannya dengan kematian dan hidup seseorang.

Terkait dengan peristiwa gerhana, agama Islam mensyari’atkan beberapa hal:

1. Memperbanyak zikir, istighfar, takbir, salat gerhana dan sedekah.
2. Melakukan observasi gerhana menyaksikan salah satu bukti kekuasaan-Nya.
3. Menyeru jama’ah untuk melaksanakan salat gerhana dengan panggilan *ash-salatu jami’ah* dan tidak ada *adzan* maupun *iqamah*.

[35]Syamsul Anwar, *Interkoneksi Studi Hadis dan Astronomi*, (Yogyakarta: Suara Muhammadiyah, 2011), 61-62.

4. Mengerjakan salat gerhana secara berjama'ah.
5. Berkhutbah setelah salat gerhana.[36]

Tentu saja ada aspek perenungan ayat-ayat kauniyah juga, bukan sekadar aspek ibadahnya. Oleh karenanya disarankan pada saat puncak gerhana, jamaah berkesempatan juga untuk melihat langsung proses gerhana tersebut.[37]

Berkaitan dengan waktu pelaksanaan shalat gerhana Matahari, dalam hadis riwayat Muslim tersebut Nabi juga mengisyaratkan waktu pelaksanaannya yaitu sampai Matahari pulih kembali. Sehingga mengetahui kapan mulai terjadi gerhana dan kapan berakhirnya menjadi penting karena berkaitan dengan syarat sah shalat. Menurut Ibn Ali Nawawy[38] awal waktu shalat gerhana Matahari dimulai ketika terjadi perubahan bentuk Matahari karena tertutup piringan Bulan sedangkan waktu shalat gerhana Matahari berakhir ketika Matahari

---

[36] Muhammad Jayusman, "Fenomena Gerhana dalam Wacana Hukum Islam dan Astronomi", dalam *Al-'Adalah*, Vol. X, No. 2 Juli 2011, 244-245.

[37] T Djamaluddin, *Gerhana Matahari Cincin 26 Januari 2009: Salat Gerhana*, http://t-djamaluddin.spaces.live.com diakses pada tanggal 23 Maret 2017.

[38] Muhammad bin Amr bin Ali Nawawy, *Nihayah az-Zain fiy Irsyad al-Mubtadiin*, (Libanon: Dar al-Fikr, t.t.), 110. Lihat juga: as-Syarbiny, *al-Iqna' fiy Hilli,* Juz I, 163.

telah pulih seperti semula atau Matahari sudah terbenam walaupun masih dalam keadaan gerhana.

3. Macam-Macam Gerhana Matahari

Terdapat beberapa macam gerhana Matahari, yaitu:[39]

a. Gerhana Matahari Total/GMT (*Total Solar Eclipse*/*Kusuf Kulli*)

Pada saat GMT seluruh bundaran Matahari di langit tertutup oleh bundaran Bulan, diameter sudut Bulan lebih besar dibanding dengan diameter sudut Matahari. Terjadi di daerah permukaan Bumi yang terkena bayangan umbra Bulan. Durasi selama 7 menit, karena ukuran Bulan lebih kecil dari Bumi. Diawali dan diakhiri dengan gerhana Matahari sebagian.

b. Gerhana Matahari Cincin/GMC (*Annular Solar Eclipse*/*Kusuf Halqi*)

Terjadi saat bundaran Bulan berada di dalam bundaran Matahari, karena diameter sudut Bulan lebih kecil dibanding dengan diameter sudut Matahari; di daerah permukaan Bumi terkena lanjutan umbra. Pada saat gerhana ini, Matahari

[39] Khazin, *Ilmu Falak*, 186-187. Lihat juga: Robin M. Green, *Spherical Astronomy*, (Cambridge: Cambridge University Press, 1985), 441.

terlihat bercahaya dan berbentuk seperti cincin. Terjadi pada saat Bulan berada pada titik terjauhnya dari Bumi (titik *Aphelion*).

Karena bagian bola Matahari yang tampak dari Bumi layaknya piringan itu tidak seluruhnya tertutup oleh bayang-bayang Bulan. Bagian yang terlihat oleh kita yang di Bumi hanya sebagain kecil seperti sabit Matahari yang berbentuk cincin. Inilah cincin dari sebagian cahaya Matahari.

c. Gerhana Matahari Sebagian/Parsial (*Partial Solar Eclipse*/*Kusuf Ba'dhi*)

Terjadi pada saat sebagian bundaran Bulan menutupi sebagian bundaran Matahari di daerah permukaan Bumi berada pada bayangan kabur (*penumbra*) Bulan sebagian sehingga ada bagian Matahari yang tidak terlihat normal (terang). Waktu gerhana Bulan lebih lama dari gerhana Matahari total, karena luasnya bayangan kabur Bulan daripada bayangan inti.

4. Syarat Terjadi Gerhana Matahari

Bidang orbit Bulan dan ekliptika berpotongan pada dua titik simpul. Garis simpul penghubung titik simpul beregresi ke arah barat dengan periode 18,6 tahun. Pada fase Bulan baru, Bulan dan Matahari berada pada bujur ekliptika yang sama. Kerucut umbra Bulan tidak

selalu dapat menyentuh Bumi (yang menyebabkan terjadi gerhana). Gerhana terjadi dekat dengan titik simpul.

Titik simpul beregresi ke barat oleh karena itu Matahari lebih cepat mencapai titik simpul. Periode Matahari dari titik simpul kembali lagi ke titik simpul yang sama dinamakan satu tahun gerhana = 346,62 hari (rata-rata). Periode ini lebih pendek 18,63 hari dengan periode Sideris (365,25 hari). Kemungkinan terjadi gerhana hanya bila Matahari dan Bulan berada pada bujur ekliptika dekat titik simpul $15{,}35° < \Delta\lambda < 18{,}51°$. Pada batas tersebut keadaan maksimum yang dapat dicapai adalah gerhana persinggungan antara Matahari dan Bulan. Satu hari dalam peredaran semunya, Matahari bergerak pada ekliptika (360°/365,25) = ~0,985626283°/hari. Dalam satu Bulan sinodis (360°/346,62) x 29,53 = 30,67°. Syarat minor terjadinya gerhana Matahari $\Delta\lambda$ ~15,35°. Fasa gerhana dicapai 2 x $\Delta\lambda$ ~ 30,7. Matahari bergerak 30,67° terhadap titik node/simpul 29,53 x 360°/346,62.[40]

[40] Tim Penyusun Naskah IDI Hukum, *Islam Untuk Disiplin Ilmu Astronomi; Buku Dasar Pendidikan Agama Islam Pada Perguruan Tinggi Umum Jurusan/Program Studi Astronomi*, (Jakarta: Depag RI, 2000), 89.

C. Interpolasi

Menurut Meeus[41], interpolasi adalah proses atau metode untuk menemukan nilai-nilai untuk waktu tertentu, kuantitas dan lain-lain, berdasarkan data yang disajikan dalam sebuah tabel. Misalkan posisi Matahari yang akan dihitung sangat banyak (> 3) pada saat tertentu pada hari yang sama. Kemudian seseorang dapat menghitung posisi Matahari untuk 0^h, 12^h dan 24^h pada hari itu, dan kemudian menggunakan hasil-hasilnya untuk melakukan interpolasi untuk setiap saat yang diinginkan pada hari tersebut.

Misal tiga nilai tabular y_1, y_2, y_3 dari fungsi y diberikan, sesuai dengan nilai-nilai x_1, x_2, x_3 dari argumen x. Maka terbentuk tabel tingkat perbedaan seperti berikut ini:

x_1	y_1		
		a	
x_2	y_2		c
		b	
x_3	y_3		

Di mana a = y2 - y1 dan b = y3 - y2 disebut perbedaan-perbedaan tingkat pertama. Perbedaan tingkat kedua c adalah sama dengan b - a, yaitu :

$$c = y1 + y3 - 2\,y2$$

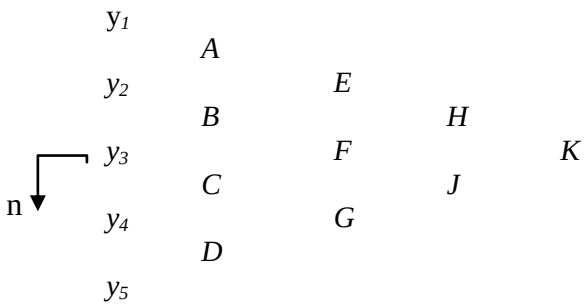
[41] Meeus, *Astronomical Algorithms,* 23-24.

Misalkan *n* adalah faktor interpolasi. Artinya, jika nilai fungsi *y* diperlukan untuk nilai argumen *x*, kita memiliki $n = x - x_2$ dengan satuan seperti pada interval tabel. Nilai *n* adalah positif jika $x > x_2$, yaitu untuk nilai 'setelah' x_2, atau dari x_2 ke bagian bawah tabel. Jika *x* mendahului x_2, maka $n < 0$.

Jika y_2 telah dipilih dengan benar, maka *n* antara -0.5 dan +0.5, Meskipun rumus berikut ini juga akan memberikan hasil yang benar untuk semua nilai *n* antara -1 dan +1.

Sehingga rumus interpolasi untuk tiga nilai tabular adalah : $y = y_2 + \frac{n}{2}(a + b + n\,c)$

Jika misalnya disajikan lima nilai tabular namun tingkat perbedaan ketiga tidak dapat diabaikan, lebih dari tiga nilai tabular harus digunakan. Misal diambil lima nilai tabular berturut-turut, y_1 sampai y_5, kemudian dibentuk, seperti uraian sebelumnya, tabel tingkat perbedaan:



di mana $A = y_2 - y_1$, $H = F - E$, dst. Jika n adalah faktor interpolasi, diukur dari nilai sentral y_3 dalam satuan interval tabular, positif ke arah y_4, rumus interpolasinya adalah

$$y = y_3 + \frac{n}{2}(B + C) + \frac{n^2}{2}F + \frac{n(n^2 - 1)}{12}(H + J) + \frac{n^2(n^2 - 1)}{24}K$$

yang juga dapat ditulis

$$y = y_3 + n\left(\frac{B + C}{2} - \frac{H + J}{12}\right) + n^2\left(\frac{F}{2} - \frac{K}{24}\right) + n^3\left(\frac{H + J}{12}\right) + n^4\left(\frac{K}{24}\right)$$

D. Pemrograman Bahasa C

Bahasa C adalah bahasa pemrograman yang dapat dikatakan berada di antara bahasa tingkat rendah dan tingkat tinggi. Bahasa tingkat rendah artinya bahasa yang berorientasi pada mesin dan tingkat tinggi berorientasi pada manusia. Bahasa tingkat rendah, misalnya bahasa *assembler*. Bahasa tingkat rendah merupakan bahasa yang membutuhkan kecermatan yang teliti bagi pemrogram karena perintahnya harus rinci, ditambah lagi masing-masing pabrik mempunyai sandi perintah sendiri. Bahasa tinggi relatif mudah digunakan, karena ditulis dengan bahasa manusia sehingga mudah dimengerti dan tidak tergantung mesinnya.

Pencipta bahasa C adalah Brian W. Kernighan dan Denis M. Ritchi, sekitar tahun 1972. Penulisan program dalam bahasa C dilakukan dengan membagi dalam blok-blok, sehingga bahasa C disebut dengan bahasa terstruktur. Bahasa C dapat digunakan di berbagai mesin dengan mudah, mulai dari PC sampai dengan mainframe, dengan berbagai sistem operasi misalnya DOS, Windows, UNIX, VMS dan lain-lain.[42]

Bahasa C mempunyai kemampuan untuk mengolah data bilangan, manipulasi *string*, pemrosesan *array*, akses *file*, direktori dan sistem operasi, pengolahan database dan lain-lain. Dengan kemampuan bahasa C tersebut, dimungkinkan untuk mengolah *file* data *ephemeris* JPL NASA yang dirilis dalam bentuk file ASCII (*American Standard Code for Information Interchange*) atau *file* yang biasa dikenal sebagai teks di kalangan umum. Dengan pemrograman bahasa C, *file* tersebut bisa diurai dengan algoritma tertentu untuk diperoleh data *ephemeris* yang diinginkan untuk waktu yang diinginkan.

Berikut contoh strutktur penulisan program bahasa C:

```
#include <stdio.h>
main ()
{
      .........
}
```

[42] Budi Raharjo, *Kumpulan Solusi Pemrograman C*, (Bandung: Informatika, 2016), v.

Program dalam bahasa C selalu berbentuk fungsi seperti ditunjukkan dalam **main ()**. Program yang dijalankan berada di dalam tubuh program yang dimulai dengan tanda kurung buka { dan diakhiri dengan tanda kurung tutup }. Semua yang tertulis di dalam tubuh program ini disebut dengan blok. Tanda () digunakan untuk mengapit argumen suatu fungsi. Argumen adalah suatu nilai yang akan digunakan dalam fungsi tersebut. Dalam fungsi *main* di atas tidak ada argumen, sehingga tak ada data dalam (). Dalam tubuh fungsi antara tanda { dan tanda } ada sejumlah pernyataan yang merupakan perintah yang harus dikerjakan oleh prosesor. Setiap pernyataan diakhiri dengan tanda titik koma (;).

Baris pertama **#include <…>** bukanlah pernyataan, sehingga tidak diakhiri dengan tanda titik koma (;). Baris tersebut meminta kompiler untuk menyertakan file yang namanya ada di antara tanda <…> dalam proses kompilasi. File-file ini (berekstensi .h) berisi deklarasi fungsi ataupun *variable*. File ini disebut *header*. File ini digunakan semacam perpustakaan bagi pernyataan yang ada di tubuh program.

BAB III
ALGORITMA PERHITUNGAN DATA *EPHEMERIS* DAN GERHANA MATAHARI BERBASIS JPL NASA

A. Tinjauan Umum tentang JPL NASA

Jet Propulsion Laboratory (JPL) NASA adalah salah satu divisi di Caltech (*California Institute of Technology*) California Selatan, Boulevard, Pasadena, California, Amerika Serikat yang menjadi pusat eksplorasi robot dari tata surya yang terkemuka AS. Para peneliti di JPL dan Caltech terus melakukan penelitian dan eksplorasi tata surya, termasuk meliputi Bumi dan atmosfernya.[1]

Pertengahan 1930-an, embrio JPL mulai muncul ketika beberapa mahasiswa Caltech dan para penggemar roket amatir belajar dan menguji coba roket. Namun karena terjadi ledakan di kampus, lokasi percobaan kelompok mahasiswa tersebut dipindah ke pegunungan San Gabriel, lokasi JPL sekarang. Pada dekade berikutnya, JPL yang disponsori oleh Angkatan Darat AS mengembangkan teknologi roket dan sistem rudal sebagai respon atas tantangan roket Jerman.[2]

---

[1] *Jet Propulsion Laboratory*, http://www.caltech.edu/content/jet-propulsion-laboratory, akses 11 April 2017, 11:30 WIB.

[2] *History & Archives*, https://www.jpl.nasa.gov/about/history.php, akses 11 April 2017, 11.35 WIB.

Awal eksplorasi ruang angkasa didorong oleh perang dingin antara AS dan Uni Soviet. Pada Oktober 1957, Uni Soviet lebih dahulu meluncurkan satelit Sputnik ke orbit Bumi. Tidak sampai tiga bulan, pada Januari 1958 JPL meluncurkan satelit pertama Amerika yang bernama Explorer 1. Bahkan pesawat ruang angkasa pertama ini membuat penemuan ilmiah penting, yaitu berhasil mendeteksi apa yang kemudian dikenal sebagai sabuk radiasi yang mengelilingi Bumi, yang diberi nama sabuk radiasi Van Allen sesuai dengan ilmuwan yang merancang instrumen utama pada Explorer 1, James Van Allen.

Ketika *National Aeronautics and Space Administration* (NASA) didirikan pada bulan Oktober 1958, JPL dipindahkan dari Angkatan Darat ke lembaga baru tersebut. Transisi dari Angkatan Darat ke NASA juga ditandai perubahan lain. JPL mulai mengalihkan perhatiannya untuk mengembangkan penelitian tentang pesawat ruang angkasa untuk keperluan ilmiah.

Sejak menjadi bagian dari NASA, JPL dimasukkan pada struktur FFRDC (*Federally Funded Research and Development Center*) yang didedikasikan untuk menangani robot untuk eksplorasi ruang angkasa. Namun walaupun telah menjadi bagian struktur NASA, JPL tetap dikelola oleh Caltech dengan sistem kontrak sejak 1958 dan diperbaharui

setiap lima tahun. Dengan demikian, karyawan JPL adalah karyawan Caltech.

Pada era selanjutnya, 1970-an, 1980-an dan 1990-an, NASA memfokuskan JPL pada misi luar angkasa yang lebih kompleks dan besar. Tercatat pada era tersebut telah dihasilkan misi *Voyagers* untuk eksplorasi ke planet-planet terluar, *Vikings* untuk misi ke Mars (kerjasama dengan Langley Research Center NASA), misi *Galileo* ke Jupiter (kerjasama dengan Ames Research Center NASA) dan Cassini-Huygens ke Saturnus (kerjasama dengan Badan Antariksa Eropa dan Italia).[3]

Selain misi-misi tersebut, di dalam JPL terdapat grup penelitian *Solar System Dynamics* (SSD) yang merupakan anggota dari misi desain dan navigasi (*Mission Design and Navigation*) NASA. Salah satu produk dan layanan yang disediakan grup SSD adalah data ephemeris planet, satelit, komet dan asteroid.[4] Data ephemeris tersebut secara umum dibuat untuk mendukung misi pesawat ruang angkasa untuk mengeksplorasi beberapa planet.

Untuk mengakses data ephemeris tersebut, JPL telah menyediakan beberapa jalur akses, salah satunya adalah

---

[3] *History & Archives*, https://www.jpl.nasa.gov/about/history.php, akses 11 April 2017, 11.35 WIB.

[4] *About JPL Solar System Dynamics*, https://ssd.jpl.nasa.gov/?about, 11 April 2017, 12.30 WIB.

Sistem *Horizons* yang bisa diakses di http://ssd.jpl.nasa.gov/?horizons. JPL *Horizons* adalah sistem *online* yang menyediakan akses ke data ephemeris sistem tata surya dengan akurasi yang sangat tinggi. Di antara data ephemeris yang disediakan adalah 732.317 data asteroid, 3.465 komet, 178 satelit planet, 8 planet, Matahari dan lain-lain.[5]

Selain melalui Sistem *Horizons*, data ephemeris juga disediakan dalam bentuk file ASCII (*American Standard Code for Information Interchange*) atau file teks biasa, yang bisa dikonversi menjadi file *binary* sesuai kebutuhan sistem operasi yang digunakan oleh pengguna (*programmer*) untuk selanjutnya digunakan untuk menghasilkan data ephemeris obyek langit tertentu untuk rentang waktu yang diinginkan. File ASCII tersebut bisa diakses di ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/ dan dengan format file yang diuraikan di ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/ascii_format.txt.[6]

JPL NASA telah merilis file ASCII tersebut sejak 1981. Seri data paling awal adalah *Development Ephemeris* seri 102 atau biasa disebut DE102 yang diliris pada September

---

[5] *HORIZONS System*, http://ssd.jpl.nasa.gov/?horizons, 11 April 2017, 12:45 WIB.

[6] *JPL Planetary and Lunar Ephemerides*, https://ssd.jpl.nasa.gov/?planet_eph_export, 11 April 2017, 13:00 WIB.

1981[7] dan yang terakhir adalah seri DE436 yang dirilis pada Maret 2017.[8] Berikut ini data *Development Ephemeris* yang telah dirilis oleh JPL NASA.

Tabel 3.1. Seri *Development Ephemeris* yang telah dirilis JPL NASA[9]

No.	Seri	Keterangan
1.	DE102	a. Dibuat pada September 1981 b. Ada koreksi nutasi tetapi tidak librasi c. Acuan : *dynamical equator* dan *equinox* 1950. d. Epoch : JED 1206160,5 (16 April -1410) sampai JED 2817872,5 (22 Des 3002) e. Jumlah file data : 15 File f. Ukuran setiap file : 34,1 MB
2.	DE200	a. Dibuat pada September 1981 b. Ada koreksi nutasi tetapi tidak librasi c. Acuan : *dynamical equator* dan *equinox* 2000. d. Epoch : JED 2305424,5 (9 Des 1599) sampai JED 2513360,5 (31 Maret 2169).

[7] *JPL Planetary and Lunar Ephemerides*, https://ssd.jpl.nasa.gov/?planet_eph_export, 11 April 2017, 13:00 WIB..

[8] ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/, 11 April 2017, 13:15 WIB.

[9] *JPL PLANETARY AND LUNAR EPHEMERIDES : Export Information*, ftp://ssd.jpl.nasa.gov/pub/eph/planets/README.txt, 12 April 2017, 08:17 WIB. Lihat juga deskripsi masing-masing file data di masing-masing file header. Misal file header DE405 adalah file header.405 yang bisa diakses di ftp://ssd.jpl.nasa.gov/pub/eph/ planets/ascii/DE405/ header.405

		e. Jumlah file data : 29 File f. Ukuran setiap file : 5 MB g. Telah digunakan oleh *Astronomical Almanac* sejak 1984 sampai 2003
3.	DE202	a. Dibuat pada Oktober 1987 b. Ada koreksi nutasi dan librasi c. Acuan : *dynamical equator* dan *equinox* 2000. d. Epoch : JED 2414992,5 (4 Des 1899) sampai JED 2469808,5 (2 Jan 2050) e. Jumlah file data : 6 file f. Ukuran setiap file : 12,1 MB
4.	DE403	a. Dibuat pada Mei 1993 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame*. d. Epoch : JED 2305200,5 (29 April 1599) sampai JED 2524400,5 (22 Juni 2199) e. Jumlah file data : 6 file f. Ukuran setiap file : 29,9 MB
5.	DE405	a. Dibuat pada Mei 1997 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame* d. Epoch : JED 2305424,50 (9 Des 1599) sampai JED 2525008,50 (20 Feb 2201) e. Jumlah file data : 31 file f. Ukuran setiap file : 6,2 MB g. digunakan oleh *Astronomical Almanac* mulai 2003

6.	DE406	a. Dibuat pada Mei 1997 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame.* d. Epoch : JED 0625360,5 (23 Feb -3000) sampai JED 2816912,50 (6 Mei +3000) e. Jumlah file data : 60 file f. Ukuran setiap file : 95 MB
7.	DE410	a. Dibuat pada April 2003 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame.* d. Epoch : JED 2415056,5 (6 Feb 1900) sampai JED 2458832,5 (15 Des 2019). e. Jumlah file data : 3 file f. Ukuran setiap file : 6,2 MB g. Ephemeris digunakan untuk misi eksplorasi Mars
8.	DE413	a. Dibuat oada November 2004 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame.* d. Epoch : JED 2414992,5 (4 Des 1899) sampai JED 2469872,5 (7 Maret 2050) e. Jumlah file data : 6 file f. Ukuran setiap file : 7,5 MB
9.	DE414	a. Dibuat pada Mei 2005 b. Ada koreksi nutasi dan librasi c. Epoch : JED 2414992,5 (4 Des 1899) sampai JED 2469872,5 (7 Maret 2050) d. Jumlah file data : 7 file e. Ukuran setiap file : 29,9 MB

10.	DE418	a. Dibuat pada Agustus 2007 b. Ada koreksi nutasi dan librasi c. Epoch : JED 2414864,5 (29 Juli 1899) sampai JED 2470192,5 (21 Jan 2051) d. Jumlah file data : 1 file e. Ukuran setiap file : 44,9 MB
11.	DE421	a. Dibuat pada Februari 2008 h. Ada koreksi nutasi dan librasi b. Acuan : *International Celestial Reference Frame.* c. Epoch : JED 2414864,5 (29 Juli 1899) sampai JED 2471184,5 (9 Okt 2053) d. Jumlah file data : 2 file e. Ukuran setiap file : 44,9 MB
12.	DE422	a. Dibuat pada September 2009 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame.* d. Epoch : JED 625648,5 (7 Des -3000) sampai JED 2816816,5 (30 Jan 3000) e. Digunakan untuk misi MESSENGER ke Mercury f. Jumlah file data : 60 file g. Ukuran setiap file : 29,9 MB
13.	DE423	a. Dibuat pada February 2010 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame* versi 2.0. d. Epoch : JED 2378480,5 (16 Des 1799) sampai JED 2524624,5 (2 Feb 2200). e. Jumlah file data : 8 file f. Ukuran setiap file : 15 MB g. Digunakan untuk misi MESSENGER ke Mercury

14.	DE424	a. Dibuat pada Nopember 2011 b. Epoch : JED 625296,5 (21 Des -3001) sampai JED 2816816,5 (30 Jan 3000) c. Jumlah file data : 60 file d. Ukuran setiap file : 29,9 MB e. Digunakan untuk mendukung misi navigasi Lab. Ilmiah ke Mars dan bisa juga digunakan untuk keperluan yang umum
15.	DE430	a. Dibuat pada April 2013 b. Ada koreksi librasi dan 1980 nutasi c. Acuan : *International Celestial Reference Frame* versi 2.0. d. Epoch : JED 2287184,5 (21 Des 1549) sampai JED 2688976,5 (25 Jan 2650) e. Jumlah file data : 11 file f. Ukuran setiap file : 29,9 MB
16.	DE431	a. Dibuat pada April 2013 b. Ada koreksi librasi dan 1980 nutasi c. Acuan : *International Celestial Reference Frame* versi 2.0. d. Epoch : JED -3100015,5, (15 Agustus -13200) sampai JED 8000016,5, (15 Maret 17191) e. Digunakan sebagai basis perhitungan pada Sistem *Horizons* f. Jumlah file data : 30 file g. Ukuran setiap file : 298,9 MB
17.	DE432	a. Dibuat pada April 2014 b. Ada koreksi nutasi dan librasi c. Acuan : *International Celestial Reference Frame* versi 2.0. d. Epoch : JED 2287184,5 (21 Des 1549) sampai JED 2688976,5 (25 Jan 2650). e. Jumlah file data : 11 file f. Ukuran setiap file : 27,5 MB

18.	DE433	a. Dibuat pada Juli 2016 b. Epoch : JED 2287184,5 (21 Des 1549) sampai JED 2688976,5 (25 Jan 2650). c. Jumlah file data : 11 file d. Ukuran setiap file : 29,2 MB
19.	DE434	a. Dibuat pada Juli 2016 b. Epoch : JED 2287184,5 (21 Des 1549) sampai JED 2688976,5 (25 Jan 2650) c. Jumlah file data : 11 file d. Ukuran setiap file : 29,2 MB
20.	DE435	a. Dibuat pada Juli 2016 b. Epoch : JED 2287184,5 (21 Des 1549) sampai JED 2688976,5 (25 Jan 2650) c. Jumlah file data : 11 file d. Ukuran setiap file : 29,2 MB
21.	DE436	a. Dibuat pada Maret 2017 b. Epoch : JED 2287184,5 (21 Des 1549) sampai JED 2688976,5 (25 Jan 2650). c. Jumlah file data : 11 file d. Ukuran setiap file : 29,2 MB

B. Format File Data Ephemeris JPL NASA

Sebagaimana diuraikan sebelumnya bahwa salah satu cara menghasilkan data ephemeris obyek langit tertentu dalam rentang waktu yang diinginkan adalah dengan mengolah data file ASCII yang telah disediakan oleh JPL NASA. Untuk itu sebelum diuraikan lebih lanjut tentang algoritma untuk memperoleh data ephemeris perlu diketahui format file data ASCII tersebut.

1. File Header

File *header* berisi semua informasi yang digunakan oleh program lain untuk mengurai (*extract*) nama konstanta, nilai konstanta dan koefisien *Chebyshev Polynomial* yang ada di file *Development Ephemeris* sehingga diperoleh data ephemeris dari obyek atau benda langit tertentu.[10]

Secara umum nama file *header* dari *Development Ephemeris* JPL NASA memiliki format yang sama, yaitu header.XXX. Di mana XXX adalah seri data dari *Development Ephemeris* tersebut.[11] Misal file data DE200 memiliki file *header* dengan format header.200, seri DE405 memiliki file *header* dengan format header.405 dan seri DE421 memiliki file *header* dengan format header.421.

Untuk memperoleh data ephemeris obyek langit tertentu dari *Development Ephemeris* (DE) maka perlu dipahami struktur isi file *header* masing-masing seri DE. Berikut ini diuraikan strukrur dari file *header* tersebut.

---

[10] Paul J. Heafner, *Fundamental Ephemeris Computations for Use with JPL Data*, (Virginian: Willmann-Bell, Inc., 1999), 153.

[11] Heafner, *Fundamental Ephemeris Computations*, 137.

a. KSIZE dan NCOEFF

Baris pertama dari file *header* tersebut berisi sebuah nilai yang disebut KSIZE yang nilainya dua kali jumlah koefisien yang ada pada setiap blok data ephemeris tertentu. Pada baris yang sama berisi nilai yang disebut NCOEFF (jumlah dari koefisien data blok ephemeris) yang nilainya selalu setengah dari nilai KSIZE. Setelah baris yang berisi KSIZE dan NCOEFF adalah baris kosong dan berisi label GROUP 1010 pada baris berikutnya (baris ke-3).[12]

b. GROUP 1010

Setelah label GROUP 1010 pada baris ke-3, terdapat satu baris kosong di bawahnya dan baris berikutnya (baris ke-5) adalah data yang masuk pada grup tersebut. Pada grup ini terdapat 3 data yang terbagi dalam 3 (tiga) baris. Baris pertama berisi seri DE yang digunakan, misal DE200, DE405 atau DE431. Sedangkan baris kedua dan ketiga secara berurutan berisi awal dan akhir *epoch* yang digunakan seri DE tersebut. Jadi GROUP 1010 hanya berisi identitas dan *epoch* yang digunakan

[12] Heafner, *Fundamental Ephemeris Computations*, 153.

masing-masing seri DE dan tidak digunakan dalam proses komputasi data ephemeris.[13]

c. GROUP 1030

Grup data selanjutnya diberi label GROUP 1030. Grup ini dan sebelumnya dipisahkan dengan satu baris kosong. Demikian juga label grup ini dengan data grup dipisahkan dengan satu baris kosong. Data grup ini hanya satu baris yang berisi tiga angka yang secara berurutan adalah angka awal JDE, akhir JDE dan nilai interval dari koefisien *Chebyshev*.

Dalam proses komputasi data ephemeris, dua nomor pertama akan diganti dengan angka *epoch* yang sesuai dengan file blok data tertentu yang akan digunakan oleh *programmer* atau *user*. Jika beberapa file blok data akan digabungkan dalam satu file *binary* maka angka pertama dari dua angka *epoch* tersebut akan diganti dengan awal *epoch* dari file blok data yang awal dan dan angka kedua akan diganti dengan angka akhir *epoch* dari file data blok

[13] Heafner, *Fundamental Ephemeris Computations*, 154.

terakhir. Sedangkan angka ketiga tidak akan diproses dalam proses komputasi.[14]

Misal file header.200 yang memiliki awal *epoch* 2305424,50 dan akhir *epoch* 2513392,50. Jika *programmer* akan menggunakan file ASCII data blok ascp2000.200 yang mempunyai awal *epoch* 2451536,50 dan akhir *epoch* 2451568,50, maka ia akan mengkonversi file data blok tersebut dalam bentuk file *binary* yang mana di baris awal file *binary* tersebut akan berisi informasi seperti pada file header.200 tersebut, namun dengan *epoch* yang diganti dengan awal dan akhir *epoch* file data blok tersebut. Sehingga di bagian *header* file *binary* akan tertulis awal *epoch* 2451536,50 dan akhir *epoch* 2451568,50.

Contoh yang kedua, dengan file *header* yang sama, misal *programmer* akan menggabungkan dua file data blok menjadi satu file *binary.* Jika file data blok yang akan digunakan adalah ascp2000.200 (*epoch* 2451536,50 sampai 2451568,50) dan ascp2020.200 (*epoch* 2458832,50 sampai 2458864,50) maka akan dihasilkan file *binary* yang

[14] Heafner, *Fundamental Ephemeris Computations*, 154.

memiliki *header* dengan *epoch* 2451536,50 sampai 2458864,50.

d. GROUP 1040

Sebagaimana grup sebelumnya, grup ketiga ini dipisahkan satu baris kosong dengan grup sebelumnya dan satu baris kosong dengan data grup. Baris pertama dari GROUP 1040 ini berisi bilangan bulat (*integer*) dan diikuti sejumlah nama konstanta yang sesuai dengan data ephemeris tertentu. Misal pada seri DE200 baris pertama dari grup ini adalah angka 200 dan pada DE405 adalah 156. Hal ini mengindikasikan bahwa pada DE200 terdapat 200 konstanta ephemeris dan pada DE405 terdapat 156 konstanta ephemeris. Nama sejumlah konstanta ephemeris tersebut terletak di bawah angka tersebut dan tersusun dalam 10 kolom serta masing-masing memiliki maksimal 6 huruf. Nama konstanta tersebut memiliki nama yang spesifik antara satu konstanta dengan yang lain. Misal konstanta dari nilai kecepatan cahaya (CLIGHT), nilai *Astronomical Unit* (AU) dalam satuan kilometer dan inisial posisi dan *velocity* beberapa komponen yang digunakan dalam integrasi numerik. Konstanta-konstanta

tersebut akan selalu berubah dari satu data ephemeris ke data ephemeris yang lain.[15]

e. GROUP 1041

Grup selanjutnya adalah GROUP 1041 yang berisi satu bilangan bulat yang mempunyai nilai sama dengan bilangan bulat di baris pertama pada grup sebelumnya (GROUP 1040) dan diikuti sejumlah angka yang disusun dalam tiga kolom. Angka-angka tersebut adalah nilai dari nama-nama konstanta yang disebutkan dalam grup sebelumnya dengan urutan sesuai dengan urutan nama konstanta tersebut. Nilai nol digunakan untuk mengisi kolom jika angka suatu konstanta tersebut bukan kelipatan tiga.

f. GROUP 1050

Grup data selanjutnya adalah GROUP 1050 yang berisi angka yang tersusun dalam 13 kolom dan 3 baris. Setiap kolom berisi data ephemeris dari obyek benda langit tertentu. Secara berurutan dari kolom kiri, 13 kolom tersebut berisi ephemeris dari Merkurius, Venus, EMB (*Earth-Moon barycenter*), Mars, Jupiter, Saturnus, Uranus, Neptunus, Pluto,

[15] Heafner, *Fundamental Ephemeris Computations*, 154.

Bulan, Matahari, koreksi nutasi (bujur dan *obliquity*) dan koreksi librasi Bulan. Namun tidak semua ephemeris memiliki data nutasi dan librasi, sehingga kolom ke-12 dan ke-13 akan bernilai nol. Misal DE200 pada kolom ke-12 tidak bernilai nol sedangkan pada kolom ke-13 bernilai nol. Jadi DE200 memuat data nutasi namun tidak memiliki data librasi.

Sebagaimana disebutkan sebelumnya bahwa setiap blok data ephemeris berisi koefisien KSIZE/2. Angka pada baris pertama setiap kolom dalam GROUP 1050 ini menyatakan posisi dari awal koefisien obyek yang sesuai posisi kolom masing-masing obyek tersebut. Misal nilai 3 pada baris pertama dan kolom pertama mengindikasikan bahwa nilai 3 tersebut adalah koefisien untuk Merkurius yang dimulai dari posisi tiga dalam *array* koefisien KSIZE/2. Pada kenyataanya, angka pada baris pertama kolom pertama selalu berisi angka 3 karena dua nomor awal dari *array* koefisien adalah berisi awal dan akhir JDE untuk setiap koefisien dari file blok data.

Selanjutnya, angka pada baris kedua pada setiap kolom dalam GROUP 1050 secara spesifik

menyatakan berapa jumlah koefisien untuk setiap koordinat kartesius (*x, y, z*) untuk vektor posisi dan *velocity*. Angka ini akan bernilai lebih besar untuk obyek yang bergerak lebih cepat (misal Merkurius dan Bulan). Nutasi hanya memiliki satu komponen posisi dan satu komponen *velocity* dan ada dua nutasi yaitu dalam bujur (*longitude*) dan kemiringan ekliptika (*obliquity*).

Angka pada baris ketiga pada setiap kolom grup ini menyatakan berapa banyak subinterval dari interval dasar yang ada di baris ketiga dari GROUP 1030. Umumnya berisi interval 32 hari. Namun untuk obyek yang mempunyai gerak lebih cepat membutuhkan interval waktu yang lebih pendek yang mencukupi untuk pemodelan gerak obyek tersebut. Misalnya pada ephemeris DE405, Bulan butuh interval 4 hari, maka subintervalnya harus 8 dari rentang waktu 32 hari. Karena itu angka 8 berada di baris ketiga kolom ke sepuluh pada GROUP 1050 dalam file *header* DE405.

g. GROUP 1070

Grup ini adalah grup terakhir dalam struktur file *header* JPL NASA. Dalam GROUP 1070 ini tidak berisi data apapun selain satu baris kosong

setelah label GROUP 1070. Karena grup ini disesuaikan dan terhubung dengan *array* dari koefisien *Chebyshev* yang secara terpisah berada pada file data blok dalam file ASCII yang bisa diunduh dari server *ftp* dan folder yang sama dengan file *header* tersebut untuk setiap seri *Development Ephemeris.*

2. File Data

File seri *Development Ephemeris* (DE) JPL NASA menggunakan format ASCII dengan format nama file ascpXXXX.YYY. Di mana XXXX adalah nilai awal interval data dalam file data blok tersebut dan YYY adalah seri DE. Misal file ascp1980.405 memuat data tahun 1980 sampai 2000 karena data dalam seri DE405 memuat interval 20 tahun.[16]

Sebagaimana diuraikan sebelumnya bahwa seri *Development Ephemeris* JPL NASA terdiri dari file *header* yang berisi tentang beberapa baris yang berupa catatan identitas seri DE, rentang *epoch* dan nilai-nilai konstanta obyek tata surya yang digunakan dalam file data blok (misal ascp2000.405).

---

[16] *Description of JPL Solar System Ephemeris*, http://www.cv.nrao.edu/~rfisher/Ephemerides/ephem_descr.html, 1 Oktober 2016, 12:30 WIB.

Setiap file data blok berisi koefisien *Chebyshev polynomial* koordinat kartesius (*x, y, z*) untuk posisi dan *velocity* sesuai rentang waktu dalam file blok data, yang secara umum adalah interval 32 hari.

Dalam file blok data, semua obyek, terutama obyek yang memiliki gerak lebih cepat seperti Bulan dan planet-planet terdekat, persamaannya dibagi menjadi subinterval untuk mencapai akurasi yang dibutuhkan. Setiap obyek masing-masing memiliki angka tertentu untuk koefisien persamaanya. Angka koefisien tersebut dan angka subinterval dari masing-masing obyek tersebut telah di tentukan dalam file *header* masing-masing seri DE. Persamaan tersebut dalam satuan kilometer untuk koordinat posisi dan kilometer per hari untuk koordinat *velocity*.

Berikut ini ini detail format dan strukrur file data blok, misal file ascp2000.405.

```
   1  1018
 0.24515365000000000D+07  0.24515685000000000D+07 -0.33800878774221092 5D+08
 0.11307876822301477 2D+08  0.39786022058182093 3D+06 -0.25563165098206816 8D+05
 0.92511999917259998 9D+02 -0.92838737850658343 8D+01 -0.26849515736978313 3D+00
 0.12807579214907778 6D-01 -0.81519229521338396 8D-03  0.37051828587190892 8D-04
-0.11695877816014069 8D-05  0.40689377061337142 1D-07 -0.34349849032809493 2D-09
-0.16883127415281528 4D-10 -0.54607022282002218 1D+08 -0.59902498508872911 3D+07
 0.65694833452496083 9D+06  0.75102480087544481 6D+04 -0.36526132760457170 4D+03
 0.12093870147016328 8D+02 -0.66353374465743375 5D+00  0.11911276309545847 4D-01
-0.50919574954729927 5D-03 -0.34507821994851613 8D-05  0.40853776386563041 8D-06
-0.41929777290097942 4D-07  0.20743356062125883 2D-08 -0.88248937697346724 2D-10
-0.25701396072517368 9D+08 -0.43718672400147104 6D+07  0.30965294706358807 2D+06
 0.66629457548318232 5D+04 -0.20470352220849909 5D+03  0.74229522605905167 2D+01
-0.32658776507291847 9D+00  0.50342260990490009 2D-02 -0.18744611821429216 3D-03
```

Gambar 3.1. Contoh sebagian struktur file data blok ascp2000.405.

Angka 1018 pada baris pertama dalam file blok data tersebut adalah jumlah koefisiean dalam file tersebut. Sedangkan angka 1 menunjukkan bahwa data di bawahnya adalah *record* pertama data dalam file tersebut bukan data pada seluruh data ephemeris seri DE405. Demikian juga dua nilai pertama dalam blok data tersebut adalah awal dan akhir *epoch* (dalam Julian Day) yang valid dalam file blok data tersebut.

C. Algoritma Perhitungan Data Ephemeris JPL NASA

Dalam sub bab ini diuraikan algoritma untuk memperoleh data ephemeris yang berasal dari file data ASCII *Development Ephemeris* (DE) seri tertentu yang dirilis oleh JPL NASA yang secara umum melalui 3 (tiga) langkah yaitu : (1) memperoleh file ASCII data ephemeris; (2) mengkonversi

file ASCII tersebut ke format *binary;* dan (3) membaca file data *binary* untuk memperoleh data ephemeris obyek langit. Ketiga langkah tersebut diuraikan lebih detail di bawah ini.

1. Memperoleh File ASCII Data Ephemeris

Ada dua cara untuk memperoleh file ASCII data ephemeris JPL NASA. Pertama dengan cara membeli CD-ROM *JPL Planetary and Lunar Ephemerides* yang disediakan oleh Willmann-Bell, Inc. melalui internet dengan alamat www.willbell.com/software/jpl.htm. CD-ROM tersebut berisi file DE200, DE405 dan DE406.

Kedua, dengan cara mengunduh secara langsung di server *FTP* (*file transfer protocol*) yang telah disediakan oleh JPL NASA di ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii. Di dalam server tersebut terdapat direktori-direktori untuk masing-masing seri DE. Misal file data DE200 dapat diunduh di ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/de200/, file file data DE405 dapat diunduh di ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/de405/ dan seterusnya. Di dalam masing-masing *folder* tersebut terdapat tiga kelompok file yaitu file *header.xxx*, file *tespo.xxx* dan file data ASCII yang dipecah dalam beberapa file yang jumlah dan ukurannya bervariasi antara seri data DE yang satu dengan yang lain.

2. Konversi File ASCII ke Format *Binary*

Langkah pertama untuk memproses file ASCII data ephemeris tersebut adalah mengkonversinya ke format *binary*. Program ASC2EPH yang ditulis oleh Paul J. Heafner bisa digunakan untuk melakukan hal tersebut.[17]

Sebelum melakukan konversi, file ASC2EPH yang ditulis dalam bahasa C di*compile* dahulu sehingga menjadi file *executeable*. Dalam hal ini spesifikasi komputer dan *compiler* yang penulis gunakan adalah Notebook Acer TravelMate P243 dengan Processor Core i3-3110M 2,4GHz, RAM 6 Giga Byte, Hardisk 500 Giga Byte, Sistem Operasi Distro Astro 3 (*Juno*) yang berbasis Linux Debian dan GNU C Compiler (gcc) sebagai *c compiler*.

Fungsi-fungsi yang digunakan oleh file *asc2eph.c* ini disediakan oleh file lain, yaitu *astrolib.h* dan *support.h* dan juga file *library* bawaan *gcc compiler*. Sehingga untuk memudahkan meng*compile* file *asc2eph.c* maka perintah-perintah *compile* dikumpulkan

---

[17] Karena pertimbangan keterbatasan ruang, maka *source code* file tersebut penulis lampirkan di halaman lampiran di akhir tulisan ini. *Source code* tersebut bisa juga dilihat di Heafner, *Fundamental Ephemeris Computations*, 190-203.

jadi satu dalam file *makefile.unx*[18] yang kemudian bisa dengan mudah dipanggil dengan perintah *make* dalam sistem operasi linux. Perintah tersebut adalah :

```
make –f  makefile.unx
```

Dimana opsi `–f` adalah perintah untuk membaca file "`makefile.unx`" yang disertakan dalam perintah tersebut.

Perintah tersebut akan menghasil file *executeable* bernama *asc2eph* yang siap digunakan untuk mengkonversi file ASCII ke file *binary.*

```
./asc2eph -i:ascp2000.405 -o:de405_2000.bin
```

Perintah tersebut akan mengkonversi file ASCII ascp2000.405 sebagai input menjadi file *binary* de405_2000.bin yang siap dibaca dengan program lain untuk menghasilkan data ephemeris berbasis DE405. Langkah yang sama bisa dilakukan untuk seri DE yang lain. Dalam penelitian ini seri DE yang akan digunakan adalah seri DE200, DE405, DE406 dan DE421.

Selanjutnya algoritma secara umum dari program *asc2eph* adalah program akan membaca input data file

[18] Isi file bisa dilihat di lampiran

ASCII yang diinput oleh *user*. Jika berhasil, program ini selanjutnya akan mencari file HEADER.xxx di *folder* yang sama dan sesuai dengan seri data yang diinput oleh *user*. Misal file input data adalah seri DE405 (contoh: ascp2000.405) maka program akan mencari file HEADER.405. Jika file HEADER.405 tidak ditemukan, maka program berhenti. Namun jka file tersebut berhasil ditemukan maka dilanjutkan ke prosedur berikutnya yaitu membaca file HEADER.405 tersebut.

Proses pembacaan file HEADER.405 oleh program ini adalah membaca secara urut mulai KSIZE sampai dengan semua GROUP yang ada (mulai dari GROUP 1010 sampai dengan GROUP 1070) dan mem*parsing* semua konstanta yang ada di dalam masing-masing GROUP. Setelah selesai, program akan membaca file data ASCII (contoh: ascp2000.405) yang berisi konstanta-konstanta yang merupakan lanjutan dari GROUP 1070 yang berada di *file* HEADER.405 tersebut.

Data yang berhasil dibaca oleh program dari file HEADER.405 dan file ascp2000.405 tersebut selanjutnya dimasukkan dalam file *binary* yang namanya sesuai dengan yang diinput oleh *user* (dalam hal ini adalah de405_2000.bin). Jadi, file *binary* tersebut berisi data

dari file HEADER.405 di bagian atas file dan dilanjutkan file data yang diambil dari file ascp2000.405.[19]

3. Membaca File Data *Binary* untuk Memperoleh Data Ephemeris Matahari dan Bulan.

Data ephemeris yang diperoleh dalam proses pembacaan file data *binary* ini tidak semua data ephemeris namun hanya beberapa data yang dibutuhkan dalam proses pengujian akurasi beberapa seri DE dengan kasus perhitungan gerhana Matahari yang akan dibahas dalam sub bab berikutnya. File *binary* yang penulis baca adalah seri DE200, DE405, DE406 dan DE421.

Sedangkan data ephemeris yang dihasilkan dalam penelitian ini adalah data ephemeris Matahari dan Bulan yang meliputi *ecliptic longitude, ecliptic latitude, apparent right ascension, apparent declination, true geocentric distance, semi diameter.* Khusus data Matahari juga dihasilkan data *true obliquity* dan *equation of time* sedangkan khusus data Bulan dihasilkan data *horizontal parallax.*

Pembacaan data ini dilakukan oleh program *sephem* (*Small Ephemeris Program*). Program *sephem* ini adalah sebagai *frontend* dalam proses pembacaan data

[19] Disarikan dari *source code* file *asc2eph.c* terlampir.

ephemeris. Artinya file ini bergantung dengan prosedur-prosedur yang ada di dalam file lain sebagai file *dependencies*. Adapun file-file tersebut adalah file *astrocon.h, support.h, support.c, astrolib.h* dan *astrolib.c.*

Masing-masing file tersebut menangani beberapa prosedur yang ada di dalam file *sephem* yang diuraikan sebagai berikut. File *astrocon.h* berisi tentang konstanta dan perintah-perintah dasar untuk konversi satuan. Misal D2R yang berfungsi mengubah satuan derajat menjadi radian. File *support.h* berisi *prototype* beberapa fungsi dasar penanganan teks yang secara detail intruksinya berada di dalam file *support.c.*

Misal di dalam *support.h* terdapat *prototype* fungsi untuk memotong teks dari kiri sebagai berikut :

```
void left(char str[], int n, char dest[]);
```

yang secara detail isi dari fungsi tersebut terdapat di file *support.c*, yaitu :

```
void left(char str[], int n, char dest[])
{
     if (n > 0) {
          strncpy(dest, str, n);
          dest[n] = '\0';
     } else {
          strcpy(dest, "");
     }
}
```

Inti dari algoritma untuk memperoleh data ephemeris berbasis JPL NASA ada di file *astrolib.h* dan *astrolib.c.* Di mana file *astrolib.h* berisi *prototype* fungsi yang secara detail isinya ditulis di dalam file *astrolib.c* yang kemudian oleh program *sephem* fungsi-fungsi tersebut dipanggil untuk memperoleh data ephemeris.

Secara detail algoritma dari program *sephem* diuraikan sebagai berikut. Pertama ketika dijalankan, program ini akan meminta input file *binary (*misal de405_2000.bin) sebagai basis data untuk menghasilkan data ephemeris. Selanjutnya isi dari file *binary* tersebut dibaca oleh fungsi *ephopn*[20] dan hasilnya disimpan dalam memori sebagai data untuk proses berikutnya, namun jika file *binary* yang diinput tidak ditemukan, program akan berhenti.

```
if (ephopn(InFile) == NULL) {
     LogMsg(stderr, "An error occurred in
ephopn().\n");
     LogClose();
     exit(1);
}
```

Selanjutnya program akan minta input tanggal, bulan, tahun dan waktu dalam UT. Input tersebut

[20] Detail isi perintah fungsi *ephopn* dan fungsi-fungsi lain yang akan dibahas pada uraian berikutnya bisa dilihat di dalam file *astrolib.c* (terlampir)

selanjutnya dikonversi menjadi waktu dalam format JDE oleh fungsi Cal2JED.

```
Cal2JED(month, day, year, utc, 4, utc, utc, 0, &TDB);
```

Setelah diperoleh waktu dalam JED, maka *user* dihadapkan dua pilihan, yaitu menu 'Ephemeris' untuk *generate* data ephemeris Matahari dan Bulan. Menu yang kedua adalah menu 'Solar Eclipse' untuk *generate* prediksi gerhana Matahari secara toposentrik. Menu kedua ini penulis buat sebagai alat untuk menguji tingkat akurasi data ephemeris yang di*generate* dari program *sephem* ini.

Pilihan menu 'Ephemeris' akan membawa *user* ke pilihan berikutnya, yaitu pilihan *Sun* untuk *generate* data ephemeris Matahari dan pilihan *Moon* untuk data ephemeris Bulan.

Interval data ephemeris yang dihasilkan bisa dalam interval jam seperti format yang yang ada di buku Ephemeris Hisab Rukyat Kementerian Agama RI, interval menit dan detik sesuai dengan pilihan yang diinput oleh *user* pada menu yang muncul setelah *user* memilih menu *Sun* atau *Moon*. Hal ini penulis maksudkan supaya data ephemeris yang dirilis dari program *sephem* ini bisa digunakan oleh algoritma

perhitungan yang lain, misal perhitungan gerhana Matahari dan Bulan dengan algoritma Kemenag.

Berikut ini detail algoritma dan *source code*[21] dari data ephemeris yang di*generate* oleh program *sephem*.

```
Reduce(TDB, Targ, Place, StarData, p3);
r = sqrt(p3[0] * p3[0] + p3[1] * p3[1] + p3[2]
     * p3[2]);
RA_hours = atan2(p3[1], p3[0]) * R2H;
if (RA_hours < 0.0) {
     RA_hours += 24.0;
}

DEC_deg = asin(p3[2] / r) * R2D;
```

Source code tersebut memanggil fungsi *reduce* yang berfungsi untuk menurunkan sumber data ephemeris sehingga menjadi *apparent* dan *topocentric* ephemeris. Parameter yang dibutuhkan oleh fungsi *reduce* ini adalah waktu dalam JED, *targ* untuk obyek benda langit yang di*reduce* dan *p3* untuk *array* dari vektor posisi suatu obyek tertentu.

Out put dari fungsi *reduce* tersebut selanjutnya digunakan untuk proses mencari nilai *r* (jarak obyek langit dengan Bumi), nilai *right ascension* dan nilai *declination.* Nilai *r* dirumuskan dengan :

---

[21] *Source code* lengkap bisa dilihat di file *astrolib.c* dan *sephem.c* (terlampir)

$$r = \sqrt{p3[0] * p3[0] + p3[1] * p3[1] + p3[2] * p3[2]}$$

Di mana indeks 0, 1 dan 2 secara berurutan adalah vektor posisi *x*, *y* dan *z* dari obyek yang akan di*reduce.*

Right ascension merupakan nilai *tan* dari perbandingan vektor *y* dan *x* dalam satuan radian yang kemudian diubah menjadi format jam dengan fungsi R2H (*radian to hours*). Sedangkan *declination* adalah nilai *sin* dari perbandingan vektor *z* dan *r* dalam satuan radian yang kemudian diubah dalam format *degree* dengan fungsi R2D (*radian to Degree*).

Setelah diperoleh nilai *RA*, *Dec* dan *r*, maka nilai-nilai tersebut digunakan sebagai parameter untuk transformasi ke koordinat ekliptika dengan *source code* berikut ini.

```
r1[0] = RA_hours * H2R;
r1[1] = DEC_deg * D2R;
r1[2] = 1.0;
r1[3] = 0.0;
r1[4] = 0.0;
r1[5] = 0.0;

/* Change to rectangular variables */
      Pol2Rec(r1, r2);
/* Get true obliquity */
Obliquity(J2000,TDB,1,&TrueEps,&TrueEpsDot);
/* Transform */
      Eq2Ecl(r2, 0, TrueEps, r2);
/* Change to polar variables */
```

```
        Rec2Pol(r2, r2);

Lambda = r2[0] * R2D;
Beta = r2[1] * R2D;
```

Pertama nilai *RA, Dec* diubah dalam satuan radian kemudian masing-masing dimasukkan dalam *array r1*. Kemudian koordinat polar *r1* diubah menjadi koordinat *rectangular* kemudian ditampung dalam *r2* dengan fungsi *Pol2Rec* (*polar to rectangular*). Sebelum proses transformasi dari koordinat equatorial ke ekliptika, dibutuhkan satu *variable* lagi yaitu *true obliquity* (*TrueEps*). Sehingga dipanggil fungsi *Obliquity*. Setelah nilai *r2* dan *TrueEps* diperoleh selanjutnya dilakukan proses transformasi dengan fungsi *Eq2Ecl* (*equatorial to ecliptic*) dalam format koordinat *rectangular*. Untuk mengubahnya dalam koordinat polar maka dipanggil fungsi *Rec2Pol* (*rectangular to polar*). Akhirnya diperoleh *lambda* (bujur ekliptika) adalah indeks ke-0 dari *r2* dan *beta* (lintang ekliptika) adalah indeks ke-1 dari *r2*.

Proses berikutnya adalah proses perhitungan untuk memperoleh nilai semi diameter Matahari atau Bulan dalam satuan menit. Nilai konstanta *s* dan rumus

semi diameter serta *horizontal parallax* penulis ambil dari buku Jean Meeus, *Astronomical Algorithm.*[22]

$$\text{SemiDiameter} = \frac{\frac{s}{r}}{60}$$

$$\sin \pi = \frac{6378{,}14}{r};\ \pi = Horizontal\ Parallax$$

r = jarak Matahari/Bulan ke Bumi

Rumus tersebut ditulis dalam *source code* sebagai berikut.

```
if (Targ == 10) {
      r = r * AU2KM;
      s = 358473400;
} else
if (Targ == 11) {
      r = r;
      s = 959.63;
}
SemiDiameter = (s/r)/60;
HorizontalMoonParallax=asin(6378.14/r*R2D;
```

Adapun rumus *Equation Of Time* penulis kutip dari algoritma Dr. Eng. Rinto Anugraha[23] sebagai berikut:

---

[22] Untuk rumus semi diameter, lihat: Jean Meeus, *Astronomical Algorithms*, (Richmond: Willmann-Bell, Inc., 1991), 359-361. Rumus *horizontal parallax*, lihat: Meeus, *Astronomical Algorithms*, 307.

[23] Rinto Anugraha, *Mekanika Benda Langit*, (Yogyakarta: Jurusan Fisika Fakultas MIPA UGM, 2012), 78-79.

```
u = (TDB-2451545)/36525;
l0=amodulo((280.46607+36000.7698*u),360)*D2R;
EquationOfTime = (-1*(1789+237*u)*sin(l0)
                 -(7146-62*u)*cos(l0)
                 +(9934-14*u)*sin(2*l0)
                 -(29+5*u)*cos(2*l0)
                 +(74+10*u)*sin(3*l0)
                 +(320-4*u)*cos(3*l0)
                 -212*sin(4*l0))/1000;
```

D. Algoritma Perhitungan Gerhana Matahari Berbasis JPL NASA

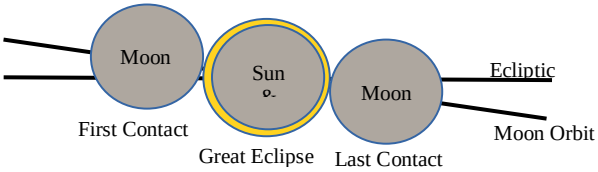
Untuk menguji akurasi data ephemeris seri DE yang paling akurat, maka dalam hal ini akan penulis uji dengan kasus gerhana Matahari total atau cincin. Kontak-kontak penting antara Matahari dan Bulan yang berkaitan dengan awal dan akhir waktu shalat gerhana Matahari menjadi fokus dalam algoritma perhitungan gerhana Matahari ini. Kontak-kontak tersebut adalah kontak pertama dan kontak terakhir serta puncak gerhana[24] sebagaimana diilustrasikan pada Gambar 3.2 dan Gambar 3.3.

[24] Kontak pertama adalah kondisi ketika mulai gerhana sebagian, kontak terakhir atau kontak ke empat adalah kondisi ketika gerhana sebagian berakhir, sedangkan tengah gerhana atau puncak gerhana (*greatest eclipse*) adalah kondisi ketika sumbu kerucut bayangan Bulan mencapai jarak terdekat dengan pusat Bumi. Fred Espenak, *Glossary of Solar Eclipse Terms*, https://eclipse.gsfc.nasa.gov/Sehelp/Seglossary.html, akses 20 Mei 2017, 10:20:14 WIB.

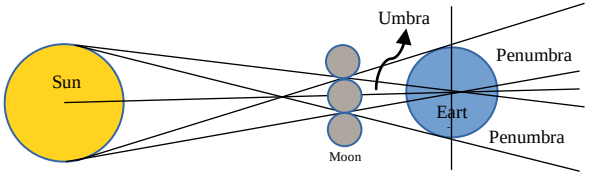
Dari ilustrasi Gambar 3.2 (tengah) dan Gambar 3.4 dapat dimodelkan secara matematis bahwa puncak gerhana Matahari total terjadi ketika bujur ekliptika Matahari dan Bulan serta lintang ekliptika Matahari dan Bulan memiliki nilai yang sama. Dalam arti yang lain puncak gerhana Matahari total atau cincin akan terjadi ketika selisih bujur ekliptika Matahari-Bulan sama dengan nol dan selisih lintang ekliptika Matahari-Bulan juga sama dengan nol. Pada saat itu sudut yang dibentuk antara Matahari dan Bulan dilihat dari Bumi (sudut elongasi[25]) mencapai nilai paling kecil. Sehingga dengan mencari nilai sudut elongasi terkecil mulai awal sampai akhir waktu gerhana Matahari, maka bisa diketahui waktu puncak gerhana Matahari total atau cincin terjadi.

---

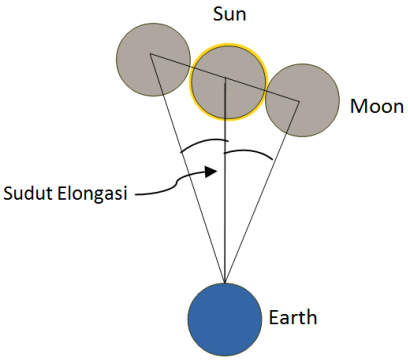
[25] Sudut elongasi berkisar antara 0^o – 180^o. Ketika Bulan dan Matahari konjungsi maka sudut elongasi 0^o, ketika oposisi maka sudut elongasi 180^o. Lihat: Ian Ridpath, *Dictionary of Astronomy*, (New York: Oxford University Press, 1997), 147.



Gambar 3.2. Ilustrasi kontak pertama, tengah gerhana dan kontak terakhir pada gerhana Matahari total atau cincin dilihat dari pengamat di Bumi.



Gambar 3.3. Ilustrasi kontak pertama, tengah gerhana dan kontak terakhir pada gerhana Matahari total atau cincin dan bayangan Bulan yang jatuh di Bumi.



Gambar 3.4. Ilustrasi perubahan sudut elongasi pada kontak pertama, tengah gerhana dan kontak terakhir pada gerhana Matahari total atau cincin.

Sedangkan kontak pertama dan kontak terakhir terjadi ketika nilai mutlak selisih antara Bujur ekliptika Matahari dan Bulan sama dengan 31'31",52 (lihat Gambar 3.2 bagian kanan dan kiri). Nilai tersebut adalah hasil penjumlahan nilai rata-rata semi-diameter Matahari (15' 59",63) dan rata-rata semi-diameter Bulan (15' 32",58).[26] Nilai penjumlahan semi-diameter Matahari dan Bulan tersebut juga bisa ditentukan secara *on the fly*[27] dengan program *sephem* sehingga nilainya *realtime* sesuai dengan nilai semi-diameter Matahari dan Bulan saat proses perhitungan gerhana Matahari.

Selanjutnya dengan melakukan *tracking* nilai selisih bujur ekliptika Matahari dan Bulan yang memenuhi nilai tersebut sebelum terjadi puncak gerhana maka akan didapatkan waktu terjadi kontak pertama. Hal sama dilakukan untuk mencari waktu kontak terakhir, yang membedakan adalah proses *tracking* dilakukan setelah terjadi puncak gerhana.

---

[26] Nilai semi-diameter Bulan adalah 14' 41" sampai 16' 45" dan semi-diameter Matahari adalah 15' 46" dan 16' 18". H.M. Nautical Almanac Office, *Explanatory Supplement to The Astronomical Ephemeris and The American Ephemeris and Nautical Almanac*, (London: Her Majesty's Stattionery Office, 1961), 215.

[27] *On the fly* sering digunakan dalam dunia *programming* untuk menggambarkan nilai sebuah *variable* yang dihasilkan ketika proses *runnning* sebuah program.

Setelah diuraikan algoritma penentuan waktu kontak pertama, tengah gerhana dan kontak terakhir, selanjutnya algoritma tersebut diimplementasikan dalam dalam penulisan program *sephem* dalam menu 'Solar Eclipse'. Di dalam menu 'Solar Eclipse' tersebut dilakukan proses perhitungan gerhana Matahari total atau cincin toposentrik karena targetnya adalah sebagai alat untuk menguji akurasi masing-masing data ephemeris seri DE dengan hasil observasi gerhana Matahari total atau cincin yang telah terjadi pada lokasi dan koordinat tertentu.

Sebagaimana algoritma untuk *generate* data ephemeris yang telah diuraikan sebelumnya, ketika *user* memilih menu 'Solar Eclipse' maka ia akan diarahkan ke input tanggal, bulan, tahun dan jam terjadinya gerhana Matahari yang selanjutnya diubah dalam format JED oleh fungsi Cal2JED.

Selanjutnya program meminta input untuk koordinat lokasi gerhana yang akan dicek yang meliputi bujur, lintang dan ketinggian lokasi tersebut. Nilai bujur, lintang dan ketinggian tempat tersebut selanjutnya dijadikan salah satu parameter dalam fungsi *reduce.* Dari fungsi *reduce* ini selanjutnya bisa dihitung data ephemeris Matahari dan Bulan dengan alur dan algoritma yang sama ketika proses *generate* data ephemeris.

Untuk keperluan mencari waktu tengah gerhana, penulis menambahkan fungsi untuk mengitung sudut elongasi yang dirumuskan berikut ini.

$$cosE = cos(RA_{moon} - RA_{sun}) * cos(DEC_{sun}) * cos(DEC_{moon}) + sin(DEC_{sun}) * sin(DEC_{moon})$$

Di mana:

E	=	sudut elongasi
RA_moon	=	askensio rekta Bulan
RA_sun	=	askensio rekta Matahari
DEC_moon	=	deklinasi Bulan
DEC_sun	=	deklinasi Matahari

yang diterjemahkan dalam bahasa C sebagai berikut.

```
ellongation = acos(cos(RA_hours_m*H2R -
RA_hours_s*H2R))*cos(DEC_deg_s)*cos(DEC_deg_m)
+sin(DEC_deg_s)*sin(DEC_deg_m);
```

Sebagaimana diuraikan sebelumnya bahwa untuk mencari waktu kontak pertama dan terakhir dibutuhkan nilai jumlah semi-diameter Matahari dan Bulan serta nilai mutlak (absolut) selisih bujur ekliptika Matahari dan Bulan yang dirumus berikut ini.

```
/*  jumlah Semidiameter */
sum_SD = SemiDiameter_s + SemiDiameter_m;
/* selisih Ecliptic Longitude */
delta_Longitude = fabs(Lambda_s*R2D
                    - Lambda_m*R2D);
```

Langkah berikutnya melakukan proses *tracking* waktu terjadi kontak pertama, tengah gerhana dan kontak terakhir pada rentang waktu mulai saat akan gerhana sampai gerhana berakhir dengan fungsi berikut ini.

```
if(counter==0){
       Elo = ellongation;
       Elo_t = TDB;
}
if(Elo>ellongation){
       Elo = ellongation;
       Elo_t = TDB;
}else{
       Elo = Elo;
       Elo_t = Elo_t;
}
if(delta_Longitude <= SD && flag==0) {
       C1_time = TDB;
}
if(delta_Longitude >= SD && flag==1
   && flag2==0){
       flag2=1;
       C4_time = TDB;
}
```

Secara berurutan, *source code* tersebut akan mencari nilai elongasi terkecil dan waktu terjadinya untuk menentukan kapan terjadi puncak gerhana. Dalam proses *tracking* berikutnya dicari waktu kontak pertama yaitu ketika nilai mutlak selisih bujur ekliptika (*delta_Longitude*) lebih kecil atau sama dengan jumlah semi-diameter Matahari dan Bulan. Demikian juga waktu kontak terakhir ditemukan ketika nilai mutlak selisih bujur ekliptika (*delta_Longitude*) lebih besar atau sama dengan jumlah semi-diameter Matahari dan Bulan. Dalam *source code* tersebut penulis tidak menggunakan tanda

tepat sama dengan (=) karena dalam proses *running* program sulit dicari nilai yang benar-benar sama, sehingga penulis memilih menggunakan tanda <= (lebih kecil atau sama dengan) dan >= (lebih besar atau sama dengan).

Setelah proses *tracking* selesai, nilai-nilai *variable* yang ditemukan ditampilkan di layar dan file *log* dengan langkah-langkah berikut ini.

```
LogMsg(stdout,    "\nTopocentric    Solar    Eclipse
Prediction\n");
LogMsg(stdout, "\nDate (dd/mm/yyyy) : %d/%d/%d\n",
day, month, year);

LogMsg(stdout, "\nLocation : %s", location);
LogMsg(stdout, "\nLatitude : %lf %s", obsr_lat_GM,
lat);
LogMsg(stdout, "\nLongitude : %lf %s", obsr_lon_GM,
lon);
LogMsg(stdout, "\nelevation : %0.0lf m", obsr_ele);

JED2Cal(C1_time, &tahun, &bulan, &hari, &jam);
waktu_c1 = jam;
jam_c1 = (int)waktu_c1;
menit_c1 = (int)((waktu_c1 - jam_c1)*60);
detik_c1 = (((waktu_c1 - jam_c1)*60)-menit_c1)*60;
LogMsg(stdout, "First Contact       : %d:%d:%0.2lf
UT\n\n", jam_c1, menit_c1, detik_c1);

JED2Cal(Elo_t, &tahun, &bulan, &hari, &jam);
jam_elo = (int)jam;
menit_elo = (int)((jam - jam_elo)*60);
detik_elo = (((jam - jam_elo)*60)-menit_elo)*60;
LogMsg(stdout, "Great Eclipse       : %d:%d:%0.2lf
UT\n\n", jam_elo, menit_elo, detik_elo);

JED2Cal(C4_time, &tahun, &bulan, &hari, &jam);
waktu_c4 = jam;
jam_c4 = (int)waktu_c4;
menit_c4 = (int)((waktu_c4 - jam_c4)*60);
detik_c4 = (((waktu_c4 - jam_c4)*60)-menit_c4)*60;
```

```
LogMsg(stdout, "Last Contact        : %d:%d:%0.2lf
UT", jam_c4, menit_c4, detik_c4);

dur_time = waktu_c4 - waktu_c1;
jam_dur = (int)dur_time;
menit_dur = (int)((dur_time - jam_dur)*60);
detik_dur    =    (((dur_time    -    jam_dur)*60)-
menit_dur)*60;

LogMsg(stdout, "Duration           : %d:%d:%0.2lf",
jam_dur, menit_dur, detik_dur);
```

Sebelum menampilkan waktu kontak pertama, puncak gerhana dan kontak terakhir, penulis lebih dahulu menampilkan tanggal, nama lokasi, lintang, bujur dan tinggi tempat lokasi terjadinya gerhana yang telah diinput sebelumnya oleh *user*. Setelah waktu yang dicari ditemukan, waktu tersebut diubah dalam jam, menit dan detik sehingga lebih *user friendly* ketika ditampilkan di layar dan disimpan dalam file *log*. Hal sama juga diterapkan untuk nilai durasi gerhana yang merupakan selisih antara waktu kontak terakhir dikurangi waktu kontak pertama.

Dari uraian tentang algoritma perhitungan gerhana Matahari total atau cincin tersebut dapat diketahui bahwa algoritma tersebut hanya memperhitungkan perkiraan terjadinya kontak pertama, kontak terakhir dan tengah gerhana atau puncak gerhana Matahari total atau cincin. Hal ini karena fokus dari hasil algortima tersebut digunakan sebagai alat bantu untuk menguji akurasi beberapa seri DE pada kontak-kontak tersebut yang selanjutnya digunakan sebagai acuan

untuk menentukan awal dan akhir waktu shalat gerhana Matahari total atau cincin. Sehingga ada beberapa fitur lain yang belum ter*cover* oleh algoritma tersebut yang bisa dikembangkan pada penelitian berikutnya. Misal prediksi kemungkinan terjadi gerhana atau tidak pada tanggal yang diinput dan algoritma untuk penentuan kontak-kontak dalam gerhana sebagian.

BAB IV
KARAKTERISTIK AKURASI DATA *EPHEMERIS* BERBASIS JPL NASA DALAM PERHITUNGAN GERHANA MATAHARI

Ada dua hal pokok yang menjadi fokus kajian dalam bab ini yaitu membandingkan akurasi beberapa seri *Develompent Ephemeris* (DE200, DE405, DE406 dan DE421). Untuk mengujinya, beberapa seri DE tersebut digunakan sebagai basis data dalam perhitungan gerhana Matahari dalam program *sephem* sehingga dihasilkan perhitungan gerhana Matahari berbasis beberapa seri DE. Hasil perhitungan tersebut selanjutnya dibandingkan dengan laporan hasil observasi yang telah dilakukan oleh Fred Espenak, *LangitSelatan* dan BMKG untuk diketahui seri DE mana yang nilainya paling dekat dengan hasil observasi tersebut. Dalam proses perbandingan tersebut juga akan diketahui karekteristik akurasi masing-masing seri DE.

Setelah ditemukan seri DE mana yang paling akurat maka selanjutnya dengan menggunakan program *sephem*, seri DE tersebut digunakan dalam contoh untuk memprediksi awal dan akhir pelaksanaan waktu shalat gerhana Matahari untuk wilayah tertentu. Dalam hal ini diambil contoh gerhana Matahari cincin yang akan terjadi pada 29 Desember 2019 dan melewati Sumatera.

A. Perbandingan Akurasi Beberapa Seri *Development Ephemeris* JPL NASA

Untuk mendapatkan hasil komputasi data ephemeris Matahari dan Bulan yang akurat, perlu dilakukan penelitian secara kontinu atau setiap terjadi perubahan kejadian alam. Karena secara astronomis, fenomena alam termasuk pergerakan Matahari dan Bulan (yang menjadi sumber data ephemeris) akan mengalami perubahan sekalipun sangat halus. Sehingga diharapkan dengan penelitian yang kontinu tersebut, data dalam proses perhitungan sesuai dengan data empiris di lapangan dan pengamatan di lapangan sesuai dengan prediksi dalam perhitungan.

Perhitungan (hisab) bukanlah sebuah metode yang muncul secara tiba-tiba. Sebab perhitungan diawali dari pengamatan yang panjang. Benar tidaknya sebuah perhitungan tentunya harus diuji secara langsung dengan melalui pengamatan terhadap fenomena alam yang dihitung. Sebagus dan sebaik apa pun sebuah metode perhitungan, jika tidak sesuai dengan fenomena alam yang dihitung tentu tidak dapat dikatakan benar.[1]

---

[1] Hendro Setyanto, *Membaca Langit*, (Jakarta: al-Ghuraba, 2008), 26. Hal ini senada dengan pendapat Ahmad Izzuddin ketika memfomulasikan penyatuan madzhab hisab dan rukyah dalam penentuan awal bulan Kamariyah. Menurutnya, formulasi yang lebih tepat adalah madzhab yang mendasarkan hisab-rukyahnya pada data penelitian

Untuk itu, akurasi seri DE yang mana yang paling baik, bisa diketahui dengan meneliti sedekat mana seri DE tersebut dengan hasil observasi fenomena astronomis di lapangan, misal gerhana Matahari. Dalam hal ini, hasil observasi gerhana Matahari yang dilakukan oleh Fred Espenak, *LangitSelatan* dan BMKG tersebut dijadikan pembanding untuk melihat seri DE mana yang paling akurat.

Adapun langkah-langkahnya adalah pertama penulis akan menjalankan program *sephem* (*Simple Ephemeris Program*) untuk memperoleh hasil prediksi gerhana Matahari pada tanggal yang sesuai dengan hasil observasi tersebut. Hasil prediksi tersebut didasarkan pada data beberapa seri DE yaitu DE200, DE405, DE406 dan DE421 dengan interval data mulai dari 1 detik sampai dengan 0,0001 detik. Kemudian hasilnya dibandingkan dengan hasil observasi tersebut. Sehingga akhirnya dengan simulasi komputasi tersebut diketahui seri DE mana yang paling mendekati hasil observasi.

1. Perhitungan Gerhana Matahari dengan *Simple Ephemeris Program* (*sephem*)

Data yang perlu disiapkan dalam proses komputasi menggunakan program *sephem* adalah

---

kontemporer sehingga dihasilkan kriteria yang akurat. Lihat: Ahmad Izzuddin, *Fiqih Hisab Rukyah*, (Jakarta: Penerbit Erlangga, 2007), 176.

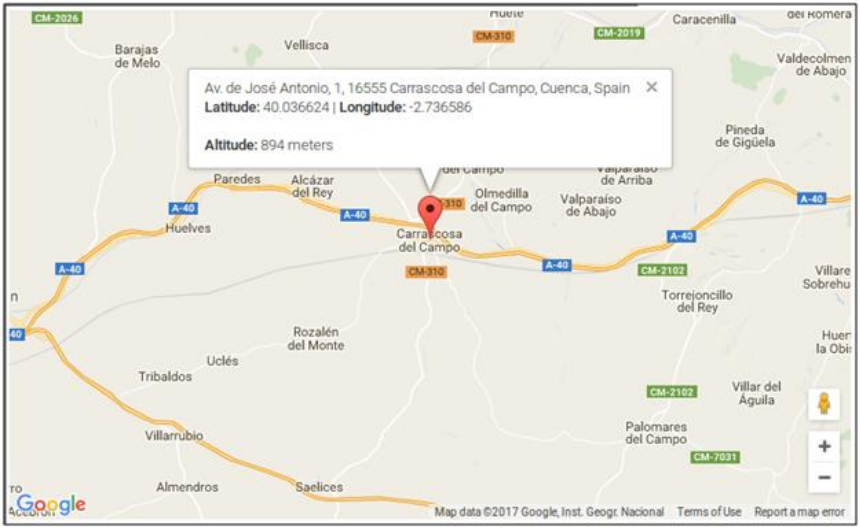
tanggal, waktu, koordinat dan ketinggian tempat terjadinya gerhana Matahari. Namun sebelumnya juga perlu disiapkan pula file data *binary* seri DE yang akan digunakan sebagai basis data perhitungan gerhana Matahari tersebut. Berikut ini file data *binary* yang digunakan dalam penelitian ini (Tabel 4.1).

Tabel 4.1. File *binary* seri DE yang digunakan sebagai basis data perhitungan gerhana Matahari

No	Seri DE	Nama File	Rentang Tahun (dd/mm/yyyy)
1	DE200	de200_2000.bin	24/12/1999 – 16/1/2020
2	DE405	de405_2000.bin	24/12/1999 – 16/1/2020
3	DE406	de406_2000.bin	24/12/1999 – 12/1/2100
4	DE421	de421_1900.bin	04/12/1989 – 02/1/2050

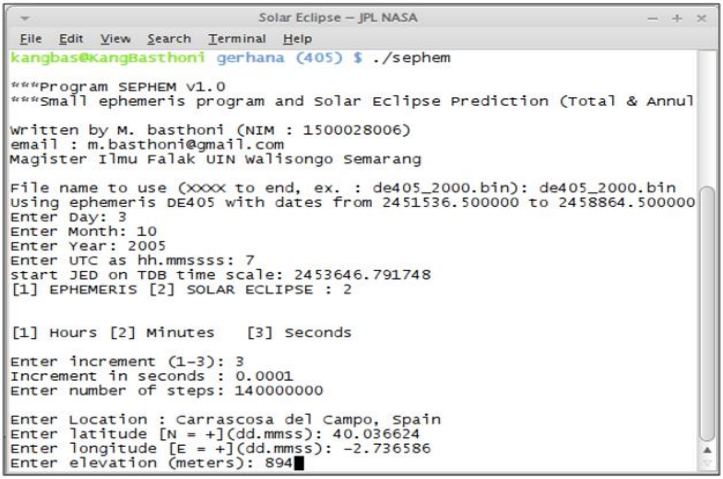
Berikut ini contoh langkah-langkah perhitungan gerhana Matahari dengan data sebagai berikut:

1. Tanggal : 3 Oktober 2005
2. Perkiraan awal gerhana : 7 UT
3. Lokasi : Carrascosa del Campo, Spanyol
4. Lintang tempat : 40,036624 LU
5. Bujur tempat : -2,736586 BB
6. Tinggi tempat : 894 meter



Gambar 4.1. Koordinat Carrascosa del Campo, Spanyol[2]

[2] *Report* observasi oleh Espenak hanya menyebutkan nama lokasi, kota dan negara tempat observasi. Untuk itu koordinat (lintang dan bujur) dan tinggi tempat observasi tersebut penulis lacak dari web site www.gps-coordinates.net yang menggunakan API (*Application Programming Interface*) dari Google Maps.



Gambar 4.2. Tampilan proses *running* program *sephem*

```
Topocentric Solar Eclipse Prediction

Date (dd/mm/yyyy) : 3/10/2005

-------------------------------------------------
Location          : Carrascosa del Campo, Spain
Latitude          : 40.036624 N
Longitude         : -2.736586 W
elevation         : 894 m
-------------------------------------------------
First Contact     : 7:38:53.609983312 UT

Great Eclipse     : 8:59:17.260423885 UT

Last Contact      : 10:30:37.266346799 UT
-------------------------------------------------
Duration          : 2:51:43.656363487
-------------------------------------------------
```

Gambar 4.3. Out put program *sephem*

Data-data tersebut kemudian dijadikan input dalam proses perhitungan gerhana Matahari sebagaimana Gambar 4.2. tersebut. Dalam hal ini dicoba dilakukan

eksperimen dengan beberapa variasi interval data[3] yaitu interval 1 detik, 0,1 detik, 0,01 detik, 0,001 detik dan 0,0001 detik. Konsekuensinya jumlah data ephemeris[4] yang di*generate* untuk perhitungan awal, tengah dan akhir gerhana juga bervariasi mulai dari 14.000 sampai dengan 140.000.000 data. Variasi jumlah data tersebut penulis hitung dengan estimasi awal data dimulai 30 menit sebelum perkiraan terjadinya awal gerhana (kontak pertama) dan 30 menit setelah akhir gerhana (kontak terakhir) dan perkiraan durasi gerhana misalnya adalah 3 jam. Sehingga untuk interval 1 detik diperoleh :

30 menit x 60	= 1.800 detik.
1.800 detik x 2	= 3.600 detik.
3 jam x 3600	= 10.800 detik.
10.800 + 3.600	= 14.400 detik.

Jadi data yang dibutuhkan untuk interval 1 detik adalah 14.400 data.

Untuk interval data yang lebih kecil, maka nilai 14.400 tersebut dibagi dengan interval data yang dikehendaki. Misal data yang dibutukan untuk interval 0,0001 detik adalah 14.400 dibagi 0,0001 detik sehingga dibutuhkan data sebesar 144.000.000 data. Jadi dengan

[3] Lihat input *Increment in seconds* pada Gambar 4.2.
[4] Lihat input *Enter number of steps* pada Gambar 4.2.

estimasi tersebut penulis membulatkannya untuk interval 1 detik sampai 0,0001 detik menjadi 14.000 sampai dengan 140.000.000 data.

Dalam eksperimen ini penulis membatasi sampai interval 0,0001 detik karena berkaitan dengan kemampuan perangkat keras laptop yang penulis gunakan. Sebagai gambaran untuk interval 1 detik dibutuhkan data 14.000 data yang dalam proses perhitungan awal, tengah dan akhir gerhana Matahari dibutuhkan waktu 2 detik. Artinya per 1 detik bisa diproses 7.000 data. Sehingga untuk satu seri DE dibutuhkan waktu proses sebagaimana Tabel 4.2. di bawah ini.

Tabel 4.2. Waktu yang dibutuhkan
dalam proses perhitungan untuk satu seri DE.

Interval	Jumlah data	Waktu proses
1 detik	14.000	2 detik
0,1 detik	140.000	20 detik
0,01 detik	1.400.000	200 detik = 3,33 menit
0,001 detik	14.000.000	2.000 detik = 33,33 menit
0,0001 detik	140.000.000	20.000 detik = 5,56 jam
Jumlah	155.554.000	22.222 detik = 6,17 jam

Jadi untuk memproses 4 (empat) seri DE yang akan diuji pada penelitian ini dibutuhkan waktu 24,68 jam.

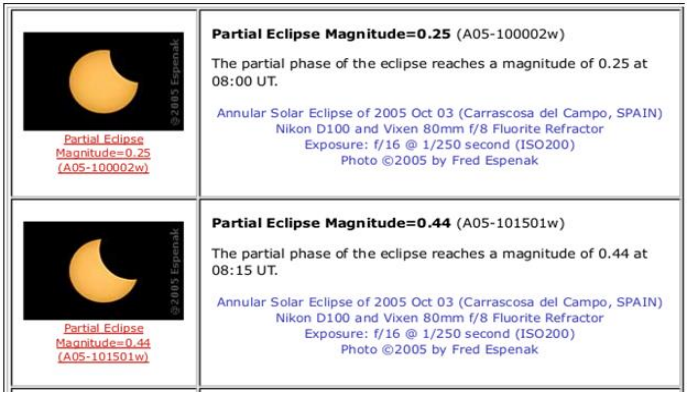
Pertimbangan lain mengapa hanya dibatasi sampai interval 0,0001 detik karena dengan interval tersebut sudah bisa diketahui *trend* selisih masing-masing seri DE dengan hasil observasi gerhana Matahari yang dijadikan acuan pembanding.

2. Perbandingan dengan Hasil Observasi

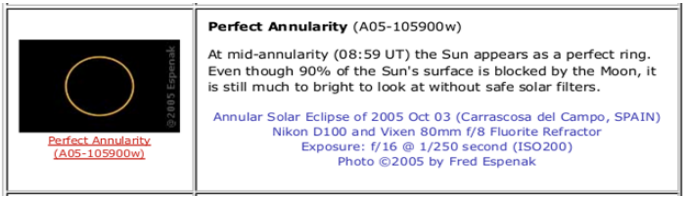
a. Data Observasi Fred Espenak

Sebelum membandingkan hasil komputasi gerhana Matahari dari program *sephem* dengan hasil observasi, berikut ini penulis uraikan data hasil observasi Fred Espenak untuk gerhana Matahari cincin yang telah terjadi pada 3 Oktober 2005 dengan lokasi observasi di Carrascosa del Campo Spanyol.[5] Beberapa kontak-kontak penting peristiwa gerhana pada tanggal tersebut bisa dilihat pada gambar-gambar di bawah ini.

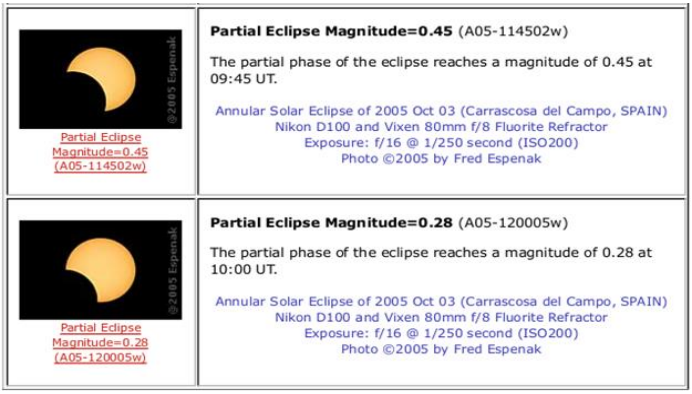
[5] Fred Espenak, *2005 Annular Solar Eclipse Photo Gallery A (Overview),* http://mreclipse.com/Sephoto/ASE2005/ASE2005galleryA.html, akses 18 Mei 2007, 7:14 WIB.



Gambar 4.4. Foto beberapa saat setelah kontak pertama.



Gambar 4.5. Foto ketika puncak gerhana Matahari cincin.



Gambar 4.6. Foto beberapa saat sebelum kontak terakhir.

Gambar 4.4 sampai Gambar 4.6 memberikan informasi kontak-kontak penting dalam observasi yang dilakukan oleh Espenak. Namun hanya waktu tengah gerhana (Gambar 4.5) yang secara eksplisit disebut oleh Espenak, yaitu terjadi pada pukul 08:59 UT. Sementara waktu untuk kontak pertama atau awal gerhana (Gambar 4.4) dan kontak terakhir atau akhir gerhana (Gambar 4.6) disebut secara implisit dengan indikator *magnitude* gerhana.

Magnitude secara spesifik digunakan oleh Fred Espenak untuk menggambarkan bagian dari diameter Matahari yang disapu atau tertutup oleh piringan Bulan. *Magnitude* gerhana bisa dinyatakan dalam persen atau desimal. Jadi *magnitude* 0,50 nilainya sama dengan 50%.[6]

Berdasarkan Gambar 4.4. bisa diperkirakan waktu terjadi kontak pertama. Gambar yang atas pada Gambar 4.4 tertulis *magnitude* 0,25 atau 25% dan terjadinya pada pukul 8:00 UT. Sedangkan pada gambar yang bawah tertulis *magnitude* 0,44 atau 44% dan terjadinya pada pukul 8:15 UT. Dari data tersebut

[6] Fred Espenak, *Glossary of Solar Eclipse Terms*, https://eclipse.gsfc.nasa.gov/ Sehelp/Seglossary.html, akses 27 Mei 2017, 11:24:50 WIB.

dapat diketahui bahwa selisih *magnitude* dan waktu antara kedua gambar tersebut secara berurutan adalah 19% dan 15 menit. Artinya laju pergerakan Bulan menutup Matahari pada waktu tersebut adalah 1,2667% per menit. Sehingga estimasi waktu terjadinya kontak pertama adalah 31,667 menit sebelum pukul 8.00 UT atau pukul 7:28:20 UT.

Demikian juga berdasarkan Gambar 4.6. bisa diperkirakan waktu terjadi kontak terakhir. Gambar yang atas pada Gambar 4.6 tertulis *magnitude* 0,45 atau 45% dan terjadinya pada pukul 9:45 UT. Sedangkan pada gambar yang bawah tertulis *magnitude* 0,28 atau 28% dan terjadinya pada pukul 10:00 UT. Sehingga selisih *magnitude* dan waktu antara dua gambar tersebut secara berurutan adalah 17% dan 15 menit. Artinya laju pergerakan Bulan menutup Matahari pada waktu tersebut adalah 1,133% per menit. Sehingga estimasi waktu terjadinya kontak terakhir adalah 31,733 menit setelah pukul 10.00 UT atau pukul 10:31:44 UT.

Jadi estimasi terjadi kontak pertama dan terakhir berdasarkan Gambar 4.4 dan Gambar 4.6 tersebut secara berurutan adalah pukul 7:28:20 UT dan 10:31:44 UT. Meskipun demikian estimasi itu

sendiri bukan bagian dari observasi. Oleh sebab itu yang jadi acuan utama pada uji akurasi ini adalah nilai tengah gerhana (8:59:00 UT) yang secara eksplisit dilaporkan oleh Espenak pada Gambar 4.5.

Berikut ini dibandingkan hasil komputasi gerhana Matahari dari program *sephem* untuk tanggal dan lokasi yang sama dengan hasil observasi Fred Espenak yang telah diuraikan sebelumnya. Yang jadi indikator utama dalam perbandingan akurasi ini adalah seberapa dekat hasil dari program *sephem* tersebut dengan hasil observasi Espenak terutama pada puncak gerhana, karena waktu pada puncak gerhana ini secara eksplisit disebutkan oleh Espenak dalam laporan observasinya. Semakin dekat atau semakin kecil selisih antara observasi Espenak dan seri DE tertentu yang diuji, maka seri DE tersebut memiliki akurasi yang lebih baik karena lebih dekat dengan bukti empirik, dalam hal ini adalah laporan observasi Espenak tersebut.

Di bawah ini adalah tabel perbandingan antara observasi Espenak dengan seri DE200, DE405, DE406 dan DE421 dalam interval data ephemeris 1 detik sampai dengan 0,0001 detik.

Tabel 4.3. Perbandingan Observasi Espenak dengan hasil program *sephem* dengan interval 1 detik.

Kontak	Espenak	Seri DE		Selisih
Tengah Gerhana	8 : 59 : 0	200	8 : 59 : 17,05	0 : 0 : 17,0503
		405	8 : 59 : 17,05	0 : 0 : 17,0503
		406	8 : 59 : 17,05	0 : 0 : 17,0503
		421	8 : 59 : 17,05	0 : 0 : 17,0503

Data pada tabel 4.3 menunjukkan bahwa pada interval data ephemeris 1 detik menghasilkan perhitungan gerhana Matahari yang sama antara seri DE200, DE405, DE406 dan DE421 pada tengah (puncak gerhana). Dan semua seri DE yang diuji pada penelitian ini juga memiliki selisih yang sama dengan laporan observasi Espenak, yaitu selisih 17,0503 detik untuk tengah gerhana sebagai acuan utama perbandingan pada penelitian ini.

Tabel 4.4. Perbandingan Observasi Espenak dengan hasil program *sephem* dengan interval 0,1 detik.

Kontak	Espenak	Seri DE		Selisih
Tengah Gerhana	8 : 59 : 0	200	8 : 59 : 17,06	0 : 0 : 17,064
		405	8 : 59 : 17,26	0 : 0 : 17,2644
		406	8 : 59 : 17,26	0 : 0 : 17,2644
		421	8 : 59 : 17,26	0 : 0 : 17,2644

Sementara ketika interval data ephemeris diperkecil menjadi 0,1 detik (Tabel 4.4), terlihat bahwa semua seri DE memiliki kesamaan, yakni selisihnya membesar dibanding ketika digunakan data ephemeris dengan interval 1 detik (Tabel 4.3) dan DE200 memiliki selisih yang berbeda dengan seri DE yang lain dan DE200 memiliki selisih terkecil dengan hasil observasi Espenak pada tengah gerhana (17,064 detik).

Tabel 4.5. Perbandingan Observasi Espenak dengan hasil program *sephem* dengan interval 0,01 detik.

Kontak	Espenak	Seri DE		Selisih
Tengah Gerhana	8 : 59 : 0	200	8 : 59 : 17,02	0 : 0 : 17,0187
		405	8 : 59 : 17,27	0 : 0 : 17,2692
		406	8 : 59 : 17,27	0 : 0 : 17,2692
		421	8 : 59 : 17,25	0 : 0 : 17,2491

Perubahahan yang ditunjukkan ketika interval data ephemeris diperkecil menjadi 0,01 detik (Tabel 4.5) adalah selisih tengah gerhana semakin kecil kecuali untuk seri DE405 dan DE406 selisihnya semakin membesar pada waktu tengah gerhana. Namun demikian, DE200 masih memiliki selisih terkecil untuk waktu tengah gerhana dibanding seri

DE yang lain, bahkan lebih kecil dibanding dengan ketika digunakan data ephemeris dengan interval 1 detik (Tabel 4.3).

Tabel 4.6. Perbandingan Observasi Espenak dengan hasil program *sephem* dengan interval 0,001 detik.

Kontak	Espenak	Seri DE		Selisih
Tengah Gerhana	8 : 59 : 0	200	8 : 59 : 17,02	0 : 0 : 17,0162
		405	8 : 59 : 17,26	0 : 0 : 17,2606
		406	8 : 59 : 17,26	0 : 0 : 17,2606
		421	8 : 59 : 17,25	0 : 0 : 17,2455

Data Tabel 4.6 menunjukkan perubahan akurasi yang hampir mirip dengan data pada Tabel 4.5. Selisih semua seri DE dengan observasi Espenak mengecil untuk tengah gerhana.. Seilish pada tengah gerhana DE200 tetap yang terkecil dibanding dengan seri DE yang lain. Bahkan DE200 selisihnya lebih kecil dibanding dengan ketika digunakan interval 1 detik, sementara seri DE yang lain tetap lebih besar dibanding data pada tengah gerhana pada Tabel 4.3 tersebut (17,0503 detik).

Tabel 4.7. Perbandingan Observasi Espenak dengan hasil program *sephem* dengan interval 0,0001 detik.

Kontak	Espenak	Seri DE		Selisih
Tengah Gerhana	8 : 59 : 0	200	8 : 59 : 17,02	0 : 0 : 17,016
		405	8 : 59 : 17,26	0 : 0 : 17,2604
		406	8 : 59 : 17,26	0 : 0 : 17,2604
		421	8 : 59 : 17,24	0 : 0 : 17,2449

Sebagaimana Tabel 4.6, data Tabel 4.7 juga menunjukkan karakteristik yang sama untuk semua seri DE. Selisih dengan observasi Espenak semakin mengecil untuk tengah gerhana. Pada eksperimen data ephemeris dengan interval 0,0001 detik ini DE200 masih konsisten memiliki selisih terkecil dibanding dengan seri DE yang lain.

b. Data Observasi *LangitSelatan*

Perbandingan selanjutnya adalah dengan data hasil observasi *langitselatan* yang juga dilaporkan oleh LAPAN untuk gerhana Matahari total yang telah terjadi pada 9 Maret 2016 dengan lokasi di SMKN 1 Maba Halmahera dengan koordinat 0,691482 LU 128,219721 BT dan ketinggian tempat 114 meter.[7]

[7] Koordinat (lintang dan bujur) dan tinggi tempat observasi tersebut dilacak dari web site www.gps-coordinates.net yang

Dalam pengamatan gerhana Matahari 9 Maret 2016 di Maba terdapat kegiatan yang dilakukan oleh dua tim, yaitu tim LAPAN dan NASA yang dilakukan di Pendopo kota – Alun-alun Jiko Mobon, serta tim *LangitSelatan* yang melakukan pengamatan dari SMK 1 Maba.[8]

Salah satu pertimbangan memilih Halmahera sebagai lokasi pengamatan adalah ketinggian Matahari pada jam gerhana dan pola cuaca pada bulan tersebut. Lokasinya yang menghadap Samudera Pasifik menjadi salah satu alasan pemilihan tempat ini, karena potensi pemantauan menjadi lebih maksimal karena perkiraan totalitas gerhana paling lama serta kondisi cuaca yang umumnya lebih cerah dibandingkan tempat lainnya.[9]

---

menggunakan API (*Application Programming Interface*) dari Google Maps.

[8] Menurut data BPS, Maba adalah salah satu kecataman di wilayah Kabupaten Halmahera Timur Maluku Utara. Lihat: http://mfdonline.bps.go.id/index.php?link=hasil_pencarian, akses 12 juni 2017, 11:15 WIB. Sedangkan SMK 1 Maba menurut data Kementerian Pendidikan dan Kebudayaan berlokasi di Jl. Poros Soagimalaha Maba Desa Soagimalaha Kecamatan Kota Maba Kabupaten Halmahera Timur Provinsi Maluku Utara. Lihat: http://referensi.data.kemdikbud.go.id/tabs.php?npsn=60200659, 12 Juni 2017, 11:20 WIB.

[9] Lembaga Penerbangan dan Antariksa Nasional, *The Elipse Gerhana Matahari Total, Catatam Peristiwa 9 Maret 2016,* 110,

Hal ini pula yang menyebabkan para peneliti *National Aeronautics and Space Administration* (NASA) memilih Maba sebagai konsentrasi aktivitasnya, bekerja sama dengan LAPAN. Mereka ingin mengamati fase-fase terjadinya gerhana, mulai dari pergerakan piringan bulan yang sedikit demi sedikit menutup piringan matahari, sampai membentuk cincin, hingga memasuki fase total. Bagi NASA sendiri, peristiwa ini merupakan kesempatan untuk menguji coba teknologi teleskop barunya yang akan digunakan untuk memantau Gerhana Matahari Besar di Amerika Serikat yang akan terjadi pada tahun 2017

Sayangnya, pemantauan langsung GMT tidak maksimal karena langit tertutup awan. Pada pukul 08.30 WIT (satu jam menjelang GMT) langit tertutup awan tebal, sehingga menghalangi pengamatan. Menjelang pukul 09.30 beberapa kali matahari tampak jelas di sela-sela awan, namun hanya berlangsung beberapa saat dan segera tertutup awan kembali. GMT terjadi pada pada pukul 09.37 hingga 09.40 WIT. Dikarenakan cuaca yang kurang mendukung, GMT

---

https://www.lapan.go.id/files_arsip/The-Eclipse-Gerhana-Matahari-Total-Catatan-Peristiwa-9-Maret-2016.pdf, akses 2 Juni 2017, 21:50 WIB.

terekam secara kurang jelas di balik awan. Peneliti LAPAN yang memantau gerhana di Maba hanya mendapatkan data totalitas selama 60 detik.

Sementara itu, di tempat lain, tim *LangitSelatan* melakukan pengamatan gerhana di halaman SMK Negeri 1 Maba. Proses gerhana dimulai sejak pukul 08.37 hingga 11.23 Waktu Indonesia Timur (WIT), sementara GMT terjadi selama 3 menit dari 09.52 hingga 09.55 WIT. Namun, meski sebelumnya diprediksi cerah, cuaca di Maba berubah seiring dengan turunnya hujan. Ini menyebabkan pemantauan langsung GMT di Maba menjadi tidak maksimal namun kontak-kontak penting bisa tercatat oleh tim *LangitSelatan*.[10]

Dalam web site resmi *LangitSelatan* dilaporkan secara detail hasil pengamatan tersebut. Kontak-kontak yang dilaporkan adalah sebagai berikut:[11]

---

[10] Lembaga Penerbangan dan Antariksa Nasional, *The Elipse Gerhana Matahari Total, Catatam Peristiwa 9 Maret 2016,* 110, https://www.lapan.go.id/files_arsip/The-Eclipse-Gerhana-Matahari-Total-Catatan-Peristiwa-9-Maret-2016.pdf, akses 2 Juni 2017, 21:50 WIB.

[11] Avivah Yamami, *GMT 2016 dari Maba dan Buli*, http://gerhana.langitselatan.com/gmt-2016-dari-maba, akses 2 Juni 2017, 21:50 WIB.

1. Kontak 1 : 08:37:00,1 WIT
2. Kontak 2 : 09:52:58,5 WIT
3. Puncak gerhana : 09:54:38,1 WIT
4. Kontak 3 : 09:56:18,3 WIT
5. Kontak 4 : 11:23:05,4 WIT

Dari data-data kontak tersebut hanya 3 kontak yang akan dijadikan acuan pembanding dalam kajian ini yaitu :

1. Kontak 1 : 08:37:00.1 WIT – 9
   = 23:37:00,1 UT
2. Puncak gerhana : 09:54:38.1 WIT – 9
   = 00:54:38,1 UT
3. Kontak 4 : 11:23:05.4 WIT – 9
   = 02:23:05,4 UT

Tabel 4.8. Perbandingan Observasi *langitselatan* dengan hasil program *sephem* dengan interval 1 detik.

No	Kontak	Langitselatan	Seri DE		Selisih
1	Awal	23 : 37 : 0,1	200	23 : 38 : 32,15	0 : 1 : 32,05
			405	23 : 38 : 33,15	0 : 1 : 33,05
			406	23 : 38 : 33,15	0 : 1 : 33,05
			421	23 : 38 : 33,15	0 : 1 : 33,05
2	Tengah	0 : 54 : 38,1	200	0 : 56 : 0,58	0 : 1 : 22,48
			405	0 : 56 : 1,58	0 : 1 : 23,48
			406	0 : 56 : 1,58	0 : 1 : 23,48
			421	0 : 56 : 1,58	0 : 1 : 23,48
3	Akhir	2 : 23 : 5,4	200	2 : 25 : 33,63	0 : 2 : 28,23
			405	2 : 25 : 34,63	0 : 2 : 29,23
			406	2 : 25 : 34,63	0 : 2 : 29,23
			421	2 : 25 : 34,63	0 : 2 : 29,23

Tabel 4.8 menunjukkan bahwa semua seri DE memiliki selisih yang sama dengan laporan observasi pada semua kontak yang diuji.

Tabel 4.9. Perbandingan Observasi *langitselatan* dengan hasil program *sephem* dengan interval 0,1 detik.

No	Kontak	Langitselatan	Seri DE		Selisih
1	Awal	23 : 37 : 0,1	200	23 : 38 : 32,16	0 : 1 : 32,06
			405	23 : 38 : 32,56	0 : 1 : 32,46
			406	23 : 38 : 32,56	0 : 1 : 32,46
			421	23 : 38 : 32,56	0 : 1 : 32,46
2	Tengah	0 : 54 : 38,1	200	0 : 56 : 0,8	0 : 1 : 22,7
			405	0 : 56 : 1,3	0 : 1 : 23,2
			406	0 : 56 : 1,3	0 : 1 : 23,2
			421	0 : 56 : 1,3	0 : 1 : 23,2
3	Akhir	2 : 23 : 5,4	200	2 : 25 : 33,23	0 : 2 : 27,83
			405	2 : 25 : 33,73	0 : 2 : 28,33
			406	2 : 25 : 33,73	0 : 2 : 28,33
			421	2 : 25 : 33,73	0 : 2 : 28,33

Sementara ketika interval data ephemeris diperkecil menjadi 0,1 detik (Tabel 4.9) seri DE200 masih memiliki selisih yang baik dibanding seri yang lain. Sedangkan seri DE405, DE406 dan DE421 memiliki selisih yang sama pada semua kontak yang diuji.

Tabel 4.10. Perbandingan Observasi *langitselatan* dengan hasil program *sephem* dengan interval 0,01 detik.

No	Kontak	Langitselatan	Seri DE		Selisih
1	Awal	23 : 37 : 0,1	200	23 : 38 : 32,12	0 : 1 : 32,02
			405	23 : 38 : 32,51	0 : 1 : 32,41
			406	23 : 38 : 32,51	0 : 1 : 32,41
			421	23 : 38 : 32,49	0 : 1 : 32,39
2	Tengah	0 : 54 : 38,1	200	0 : 56 : 0,81	0 : 1 : 22,71
			405	0 : 56 : 1,27	0 : 1 : 23,17
			406	0 : 56 : 1,27	0 : 1 : 23,17
			421	0 : 56 : 1,25	0 : 1 : 23,15
3	Akhir	2 : 23 : 5,4	200	2 : 25 : 33,22	0 : 2 : 27,82
			405	2 : 25 : 33,73	0 : 2 : 28,33
			406	2 : 25 : 33,73	0 : 2 : 28,33
			421	2 : 25 : 33,7	0 : 2 : 28,3

Eksperimen berikutnya dengan interval 0,01 detik (Tabel 4.10) menunjukkan seri DE200 masih konsisten memiliki selisih yang lebih baik dibanding seri yang lain pada semua kontak yang diuji. Selisih terkecil kedua ditunjukkan oleh seri DE421 sementara DE405 dan DE406 menempati urutan terakhir dan selisih antara keduanya selalu sama.

Tabel 4.11. Perbandingan Observasi *langitselatan* dengan hasil program *sephem* dengan interval 0,001 detik.

No	Kontak	Langitselatan	Seri DE		Selisih
1	Awal	23 : 37 : 0,1	200	23 : 38 : 32,12	0 : 1 : 32,02
			405	23 : 38 : 32,51	0 : 1 : 32,41
			406	23 : 38 : 32,51	0 : 1 : 32,41
			421	23 : 38 : 32,49	0 : 1 : 32,39
2	Tengah	0 : 54 : 38,1	200	0 : 56 : 0,81	0 : 1 : 22,71
			405	0 : 56 : 1,28	0 : 1 : 23,18
			406	0 : 56 : 1,28	0 : 1 : 23,18
			421	0 : 56 : 1,26	0 : 1 : 23,16
3	Akhir	2 : 23 : 5,4	200	2 : 25 : 33,22	0 : 2 : 27,82
			405	2 : 25 : 33,72	0 : 2 : 28,32
			406	2 : 25 : 33,72	0 : 2 : 28,32
			421	2 : 25 : 33,7	0 : 2 : 28,3

Tabel 4.11 menunjukkan seri DE200 masih memiliki selisih yang lebih baik dibanding seri yang lain pada semua kontak yang diuji. Demikian juga DE421 masih menempati posisi yang kedua dan yang terakhir ditempati oleh DE405 dan DE406 secara bersama-sama.

Tabel 4.12. Perbandingan Observasi *langitselatan* dengan hasil program *sephem* dengan interval 0,0001 detik.

No	Kontak	Langitselatan	Seri DE		Selisih
1	Awal	23 : 37 : 0,1	200	23 : 38 : 32,12	0 : 1 : 32,02
			405	23 : 38 : 32,51	0 : 1 : 32,41
			406	23 : 38 : 32,51	0 : 1 : 32,41
			421	23 : 38 : 32,49	0 : 1 : 32,39
2	Tengah	0 : 54 : 38,1	200	0 : 56 : 0,81	0 : 1 : 22,71
			405	0 : 56 : 1,28	0 : 1 : 23,18
			406	0 : 56 : 1,28	0 : 1 : 23,18
			421	0 : 56 : 1,26	0 : 1 : 23,16
3	Akhir	2 : 23 : 5,4	200	2 : 25 : 33,21	0 : 2 : 27,81
			405	2 : 25 : 33,72	0 : 2 : 28,32
			406	2 : 25 : 33,72	0 : 2 : 28,32
			421	2 : 25 : 33,7	0 : 2 : 28,3

Dalam eksperimen ketika interval diperkecil menjadi 0,0001 detik (Tabel 4.12) seri DE200 masih memiliki selisih yang baik dibanding seri yang lain DE405 pada semua kontak yang diuji. Demikian juga untuk selisih dari DE421, DE405 dan DE406 memiliki pola yang sama dengan Tabel 4.11.

c. Data Observasi BMKG

Kegiatan pengamatan gerhana Matahari 9 Maret 2016 di Palu yang dikoordinasikan oleh Badan Meteorologi, Klimatologi, dan Geofisika (BMKG) ini dilakukan di Anjungan Pantai Talise[12]. Dimulai dari pukul 08.40 WITA, gerhana Matahari total di Palu berlangsung selama dua menit 15 detik. Pada rentang waktu itu, cahaya di langit Palu meredup sebelum secara perlahan kembali bersinar terang seiring dengan selesainya fase-fase gerhana Matahari.[13] Berikut ini kontak-kontak penting pada peristiwa gerhana Matahari di Palu.

[12] Berdasarkan informasi yang dilacak dari www.gps-coordinates.com, Pantai Talise, Palu Timur, Palu, Kota Palu, Sulawesi Tengah yang menjadi tempat pengamatan GMT oleh BMKG memiliki koodinat: Lintang -0,885296 LS, Bujur 119,863314 BT dan tinggi tempat 2 meter.

[13] Pengamatan yang dilakukan oleh BMKG ini juga dilaporkan oleh pihak LAPAN. Lihat: Lembaga Penerbangan dan Antariksa Nasional, *The Elipse Gerhana Matahari Total, Catatam Peristiwa 9 Maret 2016*, 110, https://www.lapan.go.id/files_arsip/The-Eclipse-Gerhana-Matahari-Total-Catatan-Peristiwa-9-Maret-2016.pdf, akses 2 Juni 2017, 21:50 WIB.

Kontak Pertama	07:27:50.1 WITA
Kontak Kedua	08:37:48.0 WITA
Puncak Gerhana	08:38:47.0 WITA
Kontak Ketiga	08:39:46.0 WITA
Kontak Keempat	10:0:30.3 WITA
Durasi Gerhana	2 jam: 32 menit : 40.2 detik
Durasi Totalitas	1 menit : 58.0 detik
Magnitudo Gerhana	1.005

Gambar 4.7. Kontak-kontak Penting Gerhana Matahari Total di Kota Palu. (sumber: bmkg.go.id)[14]

Dari data-data kontak tersebut hanya 3 kontak yang akan dijadikan acuan pembanding dalam kajian ini yaitu :

[14] BMKG Multimedia, *Pengamatan Gerhana Matahari Total di Palu*, http://media.bmkg.go.id/gmt.bmkg?s=gmt_palu, akses 13 Juni 2017, 9.00 WIB.

1. Kontak 1 : 07 : 27 : 50,1 WITA – 8
   = 23 : 27 : 50,1 UT
2. Puncak gerhana : 08 : 38 : 47,0 WITA – 8
   = 0 : 38 : 47 UT
3. Kontak 4 : 10 : 0 : 30,3 WITA – 8
   = 2 : 0 : 30,3 UT

Tabel 4.13. Perbandingan Observasi BMKG dengan hasil program *sephem* dengan interval 1 detik.

No	Kontak	BMKG	Seri DE		Selisih
1	Awal	23 : 27 : 50,1	200	23 : 28 : 37,82	0 : 0 : 47,72
			405	23 : 28 : 37,82	0 : 0 : 47,72
			406	23 : 28 : 37,82	0 : 0 : 47,72
			421	23 : 28 : 37,82	0 : 0 : 47,72
2	Tengah	0 : 38 : 47	200	0 : 40 : 23,48	0 : 1 : 36,48
			405	0 : 40 : 23,48	0 : 1 : 36,48
			406	0 : 40 : 23,48	0 : 1 : 36,48
			421	0 : 40 : 23,48	0 : 1 : 36,48
3	Akhir	2 : 0 : 30,3	200	2 : 2 : 32,53	0 : 2 : 2,23
			405	2 : 2 : 32,53	0 : 2 : 2,23
			406	2 : 2 : 32,53	0 : 2 : 2,23
			421	2 : 2 : 32,53	0 : 2 : 2,23

Tabel 4.13 menunjukkan bahwa semua seri DE memiliki selisih yang sama dengan laporan observasi pada semua kontak yang diuji.

Tabel 4.14. Perbandingan Observasi BMKG dengan hasil program *sephem* dengan interval 0,1 detik.

No	Kontak	BMKG	Seri DE		Selisih
1	Awal	23 : 27 : 50,1	200	23 : 28 : 36,95	0 : 0 : 46,85
			405	23 : 28 : 37,35	0 : 0 : 47,25
			406	23 : 28 : 37,35	0 : 0 : 47,25
			421	23 : 28 : 37,35	0 : 0 : 47,25
2	Tengah	0 : 38 : 47	200	0 : 40 : 23,33	0 : 1 : 36,33
			405	0 : 40 : 23,74	0 : 1 : 36,74
			406	0 : 40 : 23,74	0 : 1 : 36,74
			421	0 : 40 : 23,74	0 : 1 : 36,74
3	Akhir	2 : 0 : 30,3	200	2 : 2 : 31,79	0 : 2 : 1,49
			405	2 : 2 : 32,29	0 : 2 : 1,99
			406	2 : 2 : 32,29	0 : 2 : 1,99
			421	2 : 2 : 32,19	0 : 2 : 1,89

Ketika interval data ephemeris diperkecil menjadi 0,1 detik (Tabel 4.14) seri DE200 masih memiliki selisih yang baik dibanding seri yang lain. Sedangkan seri DE405, DE406 dan DE421 memiliki selisih yang sama pada semua kontak yang diuji.

Tabel 4.15. Perbandingan Observasi BMKG dengan hasil program *sephem* dengan interval 0,01 detik.

No	Kontak	BMKG	Seri DE		Selisih
1	Awal	23 : 27 : 50,1	200	23 : 28 : 36,93	0 : 0 : 46,83
			405	23 : 28 : 37,28	0 : 0 : 47,18
			406	23 : 28 : 37,28	0 : 0 : 47,18
			421	23 : 28 : 37,27	0 : 0 : 47,17
2	Tengah	0 : 38 : 47	200	0 : 40 : 23,3	0 : 1 : 36,3
			405	0 : 40 : 23,72	0 : 1 : 36,72
			406	0 : 40 : 23,72	0 : 1 : 36,72
			421	0 : 40 : 23,71	0 : 1 : 36,71
3	Akhir	2 : 0 : 30,3	200	2 : 2 : 31,72	0 : 2 : 1,42
			405	2 : 2 : 32,2	0 : 2 : 1,9
			406	2 : 2 : 32,2	0 : 2 : 1,9
			421	2 : 2 : 32,18	0 : 2 : 1,88

Uji coba selanjutnya ketika interval data *ephemeris* diubah menjadi 0,01 detik (Tabel 4.15) diperoleh hasil bahwa seri DE200 masih konsisten memiliki selisih yang lebih baik dibanding seri yang lain pada semua kontak yang diuji. Selisih terkecil kedua ditunjukkan oleh seri DE421 sementara DE405 dan DE406 menempati urutan terakhir dan selisih antara keduanya selalu sama.

Tabel 4.16. Perbandingan Observasi BMKG dengan hasil program *sephem* dengan interval 0,001 detik.

No	Kontak	BMKG	Seri DE		Selisih
1	Awal	23 : 27 : 50,1	200	23 : 28 : 36,92	0 : 0 : 46,82
			405	23 : 28 : 37,28	0 : 0 : 47,18
			406	23 : 28 : 37,28	0 : 0 : 47,18
			421	23 : 28 : 37,27	0 : 0 : 47,17
2	Tengah	0 : 38 : 47	200	0 : 40 : 23,31	0 : 1 : 36,31
			405	0 : 40 : 23,73	0 : 1 : 36,73
			406	0 : 40 : 23,73	0 : 1 : 36,73
			421	0 : 40 : 23,71	0 : 1 : 36,71
3	Akhir	2 : 0 : 30,3	200	2 : 2 : 31,72	0 : 2 : 1,42
			405	2 : 2 : 32,19	0 : 2 : 1,89
			406	2 : 2 : 32,19	0 : 2 : 1,89
			421	2 : 2 : 32,17	0 : 2 : 1,87

Tabel 4.16 menunjukkan seri DE200 masih memiliki selisih yang lebih baik dibanding seri yang lain pada semua kontak yang diuji. Demikian juga DE421 masih menempati posisi yang kedua dan yang terakhir tetap ditempati oleh DE405 dan DE406. Perubahan selisih antara interval sebelumnya (Tabel 4.15) sangat kecil.

Tabel 4.17. Perbandingan Observasi BMKG dengan hasil program *sephem* dengan interval 0,0001 detik.

No	Kontak	BMKG	Seri DE		Selisih
1	Awal	23 : 27 : 50,1	200	23 : 28 : 36,92	0 : 0 : 46,82
			405	23 : 28 : 37,28	0 : 0 : 47,18
			406	23 : 28 : 37,28	0 : 0 : 47,18
			421	23 : 28 : 37,27	0 : 0 : 47,17
2	Tengah	0 : 38 : 47	200	0 : 40 : 23,31	0 : 1 : 36,31
			405	0 : 40 : 23,73	0 : 1 : 36,73
			406	0 : 40 : 23,73	0 : 1 : 36,73
			421	0 : 40 : 23,71	0 : 1 : 36,71
3	Akhir	2 : 0 : 30,3	200	2 : 2 : 31,72	0 : 2 : 1,42
			405	2 : 2 : 32,19	0 : 2 : 1,89
			406	2 : 2 : 32,19	0 : 2 : 1,89
			421	2 : 2 : 32,17	0 : 2 : 1,87

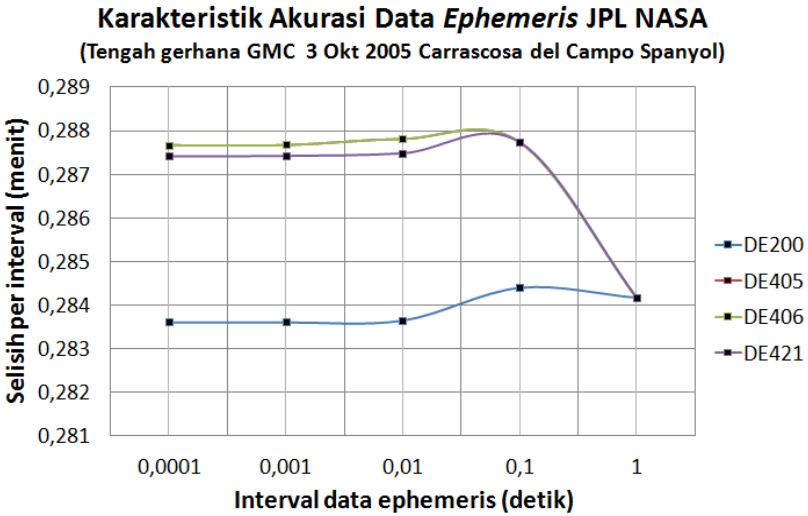
Dalam eksperimen ketika interval diperkecil menjadi 0,0001 detik (Tabel 4.17) seri DE200 masih memiliki selisih yang baik dibanding seri yang lain DE405 pada semua kontak yang diuji. Demikian juga untuk selisih dari DE421, DE405 dan DE406 memiliki pola yang sama dengan Tabel 4.16 dan nilai selisih semua seri DE sama dengan Tabel 4.16 tersebut.

3. Karakteristik Akurasi Data *Ephemeris* Berbasis JPL NASA dalam Perhitungan Gerhana Matahari

Pada pembahasan sebelumnya telah diuraikan bahwa seri DE dinilai akurat jika seri tersebut memiliki selisih terkecil dengan hasil observasi dibandingkan dengan seri DE yang lain. Pada uraian berikut ini akan diuraikan karakteristik akurasi masing-masing seri DE dalam mendekati hasil observasi. Dengan kata lain akan disajikan pola keakuratan masing-masing masing-masing seri DE dalam rentang interval data *ephemeris* yang diujicobakan dalam penelitian ini.

Tabel 4.18. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan tengah gerhana sebagai acuan utama pada GMC 3 Oktober 2005 di Spanyol

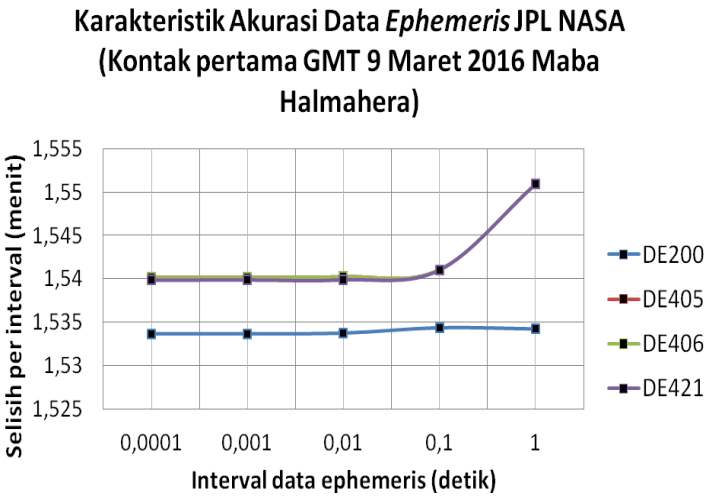
No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Tengah Gerhana	DE200	0 : 0 : 17,0503	0 : 0 : 17,064	0 : 0 : 17,0187	0 : 0 : 17,0162	0 : 0 : 17,016
2		DE405	0 : 0 : 17,0503	0 : 0 : 17,2644	0 : 0 : 17,2692	0 : 0 : 17,2606	0 : 0 : 17,2604
3		DE406	0 : 0 : 17,0503	0 : 0 : 17,2644	0 : 0 : 17,2692	0 : 0 : 17,2606	0 : 0 : 17,2604
4		DE421	0 : 0 : 17,0503	0 : 0 : 17,2644	0 : 0 : 17,2491	0 : 0 : 17,2455	0 : 0 : 17,2449



Gambar 4.8. Bentuk grafik untuk Tabel 4.18.

Tabel 4.19. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan kontak pertama sebagai acuan utama pada GMT 9 Maret 2016 di Halmahera

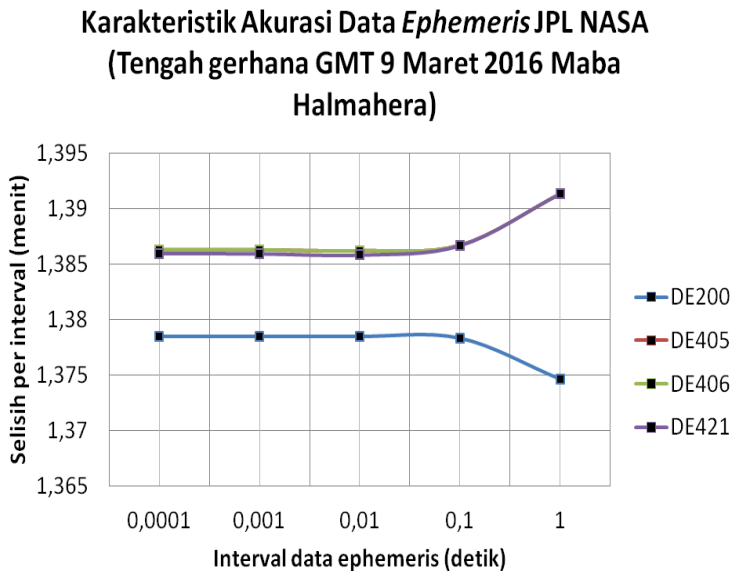
No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Kontak Pertama	DE200	0 : 1 : 32,05	0 : 1 : 32,06	0 : 1 : 32,02	0 : 1 : 32,02	0 : 1 : 32,02
2		DE405	0 : 1 : 33,05	0 : 1 : 32,46	0 : 1 : 32,41	0 : 1 : 32,41	0 : 1 : 32,41
3		DE406	0 : 1 : 33,05	0 : 1 : 32,46	0 : 1 : 32,41	0 : 1 : 32,41	0 : 1 : 32,41
4		DE421	0 : 1 : 33,05	0 : 1 : 32,46	0 : 1 : 32,39	0 : 1 : 32,39	0 : 1 : 32,39



Gambar 4.9. Bentuk grafik untuk Tabel 4.19.

Tabel 4.20. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan tengah gerhana sebagai acuan utama pada GMT 9 Maret 2016 di Halmahera

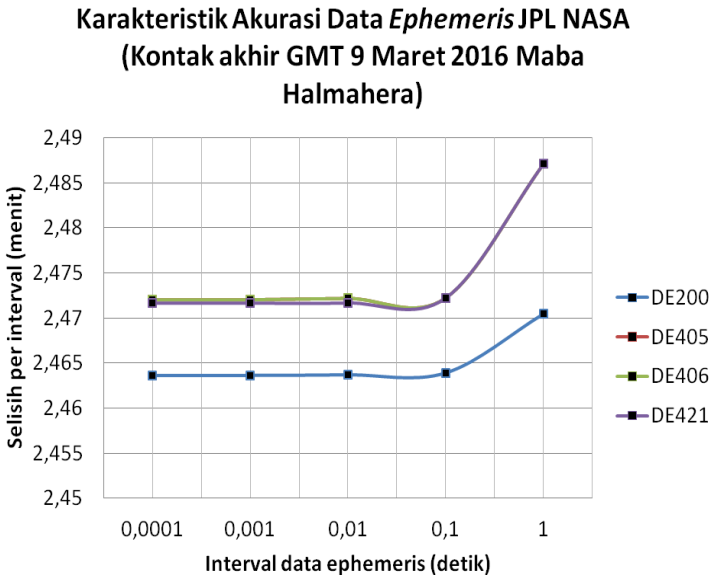
No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Tengah Gerhana	DE200	0 : 1 : 22,48	0 : 1 : 22,7	0 : 1 : 22,71	0 : 1 : 22,71	0 : 1 : 22,71
2		DE405	0 : 1 : 23,48	0 : 1 : 23,2	0 : 1 : 23,17	0 : 1 : 23,18	0 : 1 : 23,18
3		DE406	0 : 1 : 23,48	0 : 1 : 23,2	0 : 1 : 23,17	0 : 1 : 23,18	0 : 1 : 23,18
4		DE421	0 : 1 : 23,48	0 : 1 : 23,2	0 : 1 : 23,15	0 : 1 : 23,16	0 : 1 : 23,16



Gambar 4.10. Bentuk grafik untuk Tabel 4.20.

Tabel 4.21. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan kontak akhir sebagai acuan utama pada GMT 9 Maret 2016 di Halmahera

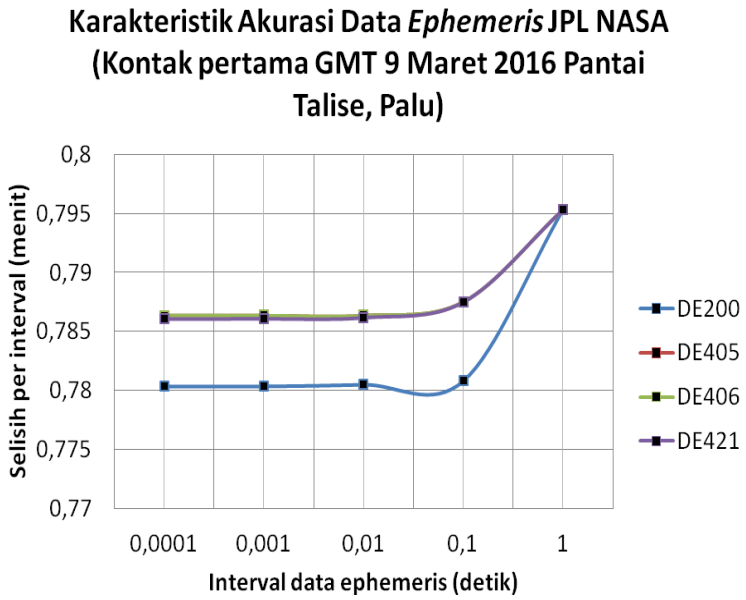
No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Kontak Terakhir	DE200	0 : 2 : 28,23	0 : 2 : 27,83	0 : 2 : 27,82	0 : 2 : 27,82	0 : 2 : 27,81
2		DE405	0 : 2 : 29,23	0 : 2 : 28,33	0 : 2 : 28,33	0 : 2 : 28,32	0 : 2 : 28,32
3		DE406	0 : 2 : 29,23	0 : 2 : 28,33	0 : 2 : 28,33	0 : 2 : 28,32	0 : 2 : 28,32
4		DE421	0 : 2 : 29,23	0 : 2 : 28,33	0 : 2 : 28,3	0 : 2 : 28,3	0 : 2 : 28,3



Gambar 4.11. Bentuk grafik untuk Tabel 4.21.

Tabel 4.22. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan kontak pertama sebagai acuan utama pada GMT 9 Maret 2016 di Palu

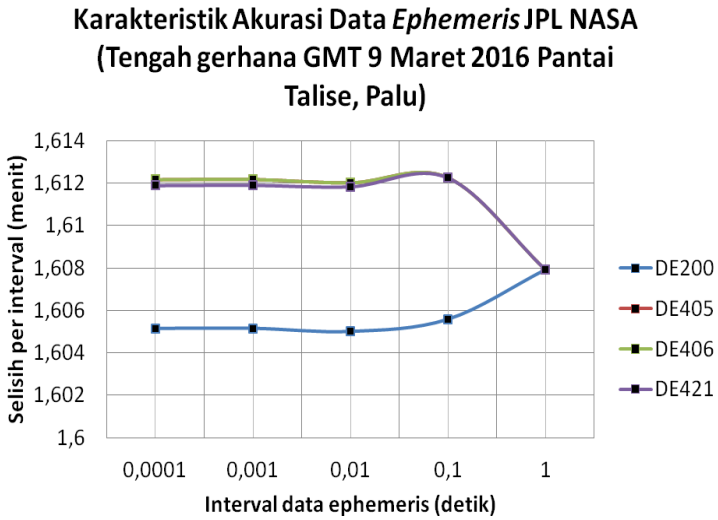
No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Kontak Pertama	DE200	0 : 0 : 47,72	0 : 0 : 46,85	0 : 0 : 46,83	0 : 0 : 46,82	0 : 0 : 46,82
2		DE405	0 : 0 : 47,72	0 : 0 : 47,25	0 : 0 : 47,18	0 : 0 : 47,18	0 : 0 : 47,18
3		DE406	0 : 0 : 47,72	0 : 0 : 47,25	0 : 0 : 47,18	0 : 0 : 47,18	0 : 0 : 47,18
4		DE421	0 : 0 : 47,72	0 : 0 : 47,25	0 : 0 : 47,17	0 : 0 : 47,17	0 : 0 : 47,17



Gambar 4.12. Bentuk grafik untuk Tabel 4.22.

Tabel 4.23. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan tengah gerhana sebagai acuan utama pada GMT 9 Maret 2016 di Palu

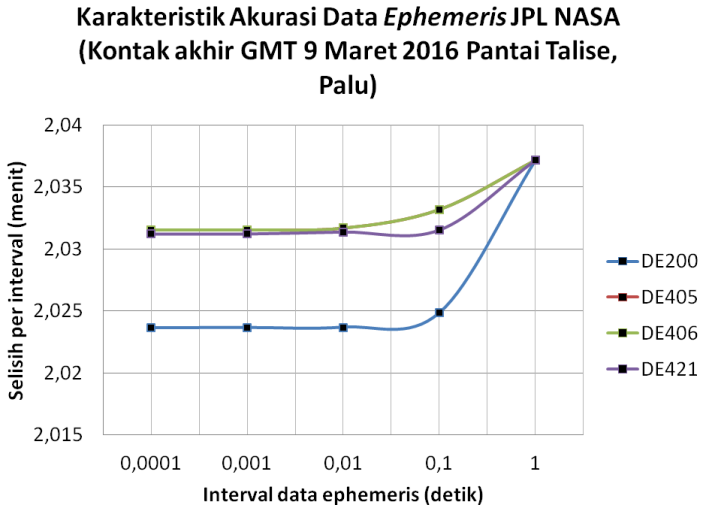
No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Tengah Gerhana	DE200	0 : 1 : 36,48	0 : 1 : 36,33	0 : 1 : 36,3	0 : 1 : 36,31	0 : 1 : 36,31
2		DE405	0 : 1 : 36,48	0 : 1 : 36,74	0 : 1 : 36,72	0 : 1 : 36,73	0 : 1 : 36,73
3		DE406	0 : 1 : 36,48	0 : 1 : 36,74	0 : 1 : 36,72	0 : 1 : 36,73	0 : 1 : 36,73
4		DE421	0 : 1 : 36,48	0 : 1 : 36,74	0 : 1 : 36,71	0 : 1 : 36,71	0 : 1 : 36,71



Gambar 4.13. Bentuk grafik untuk Tabel 4.23.

Tabel 4.24. Karakteristik akurasi masing-masing seri DE dalam beberapa variasi interval data *ephemeris* dengan acuan kontak akhir sebagai acuan utama pada GMT 9 Maret 2016 di Palu

No	Kontak	seri DE	Interval Data Ephemeris (detik)				
			1	0,1	0,01	0,001	0,0001
1	Kontak Terakhir	DE200	0 : 2 : 2,23	0 : 2 : 1,49	0 : 2 : 1,42	0 : 2 : 1,42	0 : 2 : 1,42
2		DE405	0 : 2 : 2,23	0 : 2 : 1,99	0 : 2 : 1,9	0 : 2 : 1,89	0 : 2 : 1,89
3		DE406	0 : 2 : 2,23	0 : 2 : 1,99	0 : 2 : 1,9	0 : 2 : 1,89	0 : 2 : 1,89
4		DE421	0 : 2 : 2,23	0 : 2 : 1,89	0 : 2 : 1,88	0 : 2 : 1,87	0 : 2 : 1,87



Gambar 4.14. Bentuk grafik untuk Tabel 4.24.

Berdasarkan Tabel 4.18 sampai Tabel 4.24 dan Gambar 4.8 sampai Gambar 4.14 diperoleh karakteristik akurasi masing-masing seri DE. Ada kesamaan karakteristik antara seri DE yang satu dengan yang lain, yaitu pada interval 0,01 detik atau lebih kecil semua seri DE cenderung stabil pada nilai selisih tertentu. Lonjakan selisih akurasi terjadi ketika masuk pada pada uji coba pada interval data *ephemeris* 0,1 detik. Sedangkan di awal interval (1 detik) ditemukan dua pola yaitu semua interval mempunyai selisih yang sama dan pola yang ke dua antara DE200 dan seri DE yang lain berbeda jauh selisih akurasinya.

Data-data dan gambar tersebut pula juga menginformasikan bahwa akurasi DE405 selalu sama dengan DE406 pada semua interval dalam peristiwa gerhana Matahari yang diuji pada penelitian ini. Hal ini mengkonfirmasi pernyataan Standish[15] dalam tulisannya bahwa DE406 adalah versi *Long Ephemeris* dari DE405. Artinya keduanya memiliki karakteristik akurasi yang sama namun rentang data yang dicakup oleh DE406 lebih panjang yaitu mulai tahun -3.000 sampai tahun 3.000.

---

[15] E M Standish, *JPL Planetary and Lunar Ephemerides, DE405/LE405*, Jet Propulsion Laboratory, Interoffice Memorandum, IOM 312.F-98-048, August 26, 1998, 5.

Sedangkan seri DE200 pada interval setelah 1 detik akurasinya selalu lebih baik dibanding dengan seri DE yang lain. Dari data-data tersebut disimpulkan bahwa seri DE200 memiliki akurasi yang lebih baik kemudian disusul oleh DE421 dan yang terakhir adalah seri DE405 dan DE406. Hal ini dikarenakan seri DE200 menggunakan acuan sistem referensi yang berbeda dengan seri DE405, DE406 maupun DE421. DE200 menggunakan acuan sistem referensi *equinox* dinamis dengan *epoch* J2000[16] sedangkan seri DE yang lain menggunakan acuan sistem referensi *International Celestial Reference Frame* (ICRF).[17]

Setelah ditemukan bahwa DE200 adalah seri *Development Ephemeris* yang paling akurat, selanjutnya

---

[16] DE200 awalnya berasal dari DE118 yang mengacu pada sistem referensi *equinox* dinamis *Besselian* 1950 (B1950). Selanjutnya DE118 tersebut dirotasi dengan acuan sistem referensi *equinox* dinamis *Julian* 2000 (J2000). E.M. Standish, *Orientation of the JPL Ephemerides, DE200/LE200, to the Dynamical Equinox of J2000*, Astronomy & Astrophysics, 114, (1982), 297. *Equinox* adalah titik perpotongan antara bidang ekliptika dan bidang equator. Titik perpotongan ini mengalami pergeseran dengan berjalannya waktu walau pun pergeseran tersebut sangat halus. Susiknan Azhari, *Ensiklopedi Hisab Rukyat*, (Yogyakarta: Pustaka Pelajar, 2012), 37.

[17] Acuan ICRF didefinisikan sebagai acuan yang mengadopsi lokasi dari 295 ekstragalaksi melalui observasi VLBI (*Very Long Baseline radio Interferometry*). A.M. Gontier, et.al, "Maintenance of ICRF Using the Most Stable Sources" dalam *The International Celestial Reference System and Frame, ICRS Center Report for 2001-2004*, IERS Technical Note, No. 34, (Jerman: IERS ICRS Center, 2006), 7-8.

pada pembahasan berikutnya seri DE ini digunakan sebagai basis data dalam contoh untuk menentukan kontak-kontak penting gerhana Matahari yang selanjutnya bisa digunakan untuk penentuan awal dan akhir waktu shalat gerhana Matahari di lokasi tertentu pada waktu tertentu.

B. Waktu Shalat Gerhana Matahari Total atau Cincin berdasar Data Ephemeris JPL NASA

Al-Quran menegaskan bahwa Matahari dan Bulan adalah dua di antara tanda alam yang menunjukkan kebesaran Allah sebagaimana firman Allah :

وَمِنْ ءَايَـٰتِهِ ٱلَّيْلُ وَٱلنَّهَارُ وَٱلشَّمْسُ وَٱلْقَمَرُ ۚ لَا تَسْجُدُوا۟ لِلشَّمْسِ وَلَا لِلْقَمَرِ وَٱسْجُدُوا۟ لِلَّهِ ٱلَّذِى خَلَقَهُنَّ إِن كُنتُمْ إِيَّاهُ تَعْبُدُونَ

Dan sebagian dari tanda-tanda kebesaran-Nya ialah malam, siang, Matahari dan Bulan. Janganlah bersujud kepada Matahari dan jangan (pula) kepada Bulan, tetapi bersujudlah kepada Allah yang menciptakannya, jika kamu hanya menyembah kepada-Nya.(Q.S. Fuṣṣilat/41: 37)[18]

[18] Departemen Agama RI, *al-Qur'an dan Terjemahnya,* (Semarang: CV Al Waah, 2004), 698.

Menurut Muhammad al-Syarbiny[19] ayat tersebut berkaitan dengan pensyariatan shalat gerhana. Menurutnya larangan menyembah Matahari dan Bulan dalam ayat tersebut serta perintah menyembah Allah yang telah menciptakan keduanya adalah ketika terjadi gerhana. Hal ini dia dasarkan pada hadis Nabi Muhammad SAW.:

أَنَّ الشَّمْسَ انْكَسَفَتْ عَلَى عَهْدِ رَسُوْلِ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ. فَقَامَ قِيَامًا شَدِيْدًا. يَقُوْمُ قَائِمًا ثُمَّ يَرْكَعُ. ثُمَّ يَقُوْمُ ثُمَّ يَرْكَعُ. ثُمَّ يَقُوْمُ ثُمَّ يَرْكَعُ رَكْعَتَيْنِ فِي ثَلَاثِ رَكَعَاتٍ وَأَرْبَع سَجْدَاتٍ. فَانْصَرَفَ وَقَدْ تَجَلَّتْ الشَّمْسُ. وَكَانَ إِذَا رَكَعَ قَالَ: "اللهُ أَكْبَرُ" ثُمَّ يَرْكَعُ. وَإِذَا رَفَعَ رَأْسَهُ قَالَ: "سَمِعَ اللهُ لِمَنْ حَمِدَهُ" فَقَامَ فَحَمِدَ اللهَ وَأَثْنَى عَلَيْهِ. ثُمَّ قَالَ: "إِنَّ الشَّمْسَ وَالْقَمَرَ لَايَنْكَسِفَانِ لِمَوْتِ أَحَدٍ وَلَالِحَيَاتِهِ. وَلَكِنَّهُمَا مِنْ آيَاتِ اللهِ يُخَوِّفُ اللهُ بِهِمَا عِبَادَهُ. فَإِذَا رَأَيْتُمْ كُسُوْفًا فَاذْكُرُوا اللهَ حَتَّى يَنْجَلِيَا". (رواه مسلم).[20]

Sesungguhnya terjadi gerhana Matahari pada zaman Rasulullah SAW. Kemudian beliau berdiri kemudian ruku' kemudian berdiri lagi dan ruku' lagi kemudian berdiri lagi

[19] Muhammad al-Syarbiny, *al-Iqna' fiy Ḥilli Alfaẓ Abiy Syujā'*, Juz I, 163.

[20] Muslim bin al-Ḥajjaj, *Ṣaḥih Muslim*, Jilid I, (Libanon: Dar al-Fikr, 1992), 397-398.

kemudian ruku' dua kali dalam tiga rakaat dan empat sujud. Ketika beliau selesai shalat, Matahari telah pulih. Ketika ruku' beliau membaca "*Allahu akbar*" kemudian ruku'. Ketika mengangkat kepala (*i'tidal*) beliau mengucap "*samia Allah li man ḥamidah*". Kemudian beliau berdiri dan memuji Allah kemudian bersabda: 'Sesungguhnya Matahari dan Bulan tidak mengalami gerhana karena mati atau hidup seseorang, namun keduanya adalah sebagian tanda kebesaran Allah untuk memberikan rasa takut pada hamba-hambaNya. Maka jika kalian melihat gerhana maka dzikirlah padaNya sampai keduanya pulih dari gerhana'.(HR. Muslim).

Hadis tersebut juga menjadi salah satu dasar bahwa hukum shalat gerhana adalah *sunnah muakkadah*.[21] Walaupun dalam hadis tersebut ada kata perintah, namun tidak sampai menunjukkan kewajiban karena ada indikasi lain yang menunjukkan sebaliknya.[22] Indikasi tersebut adalah Nabi

[21] Dalam hal ini Ibn Rusyd menyatakan bahwa para ulama sepakat bahwa hukum shalat gerhana Matahari adalah *sunnah*. Perbedaan para ulama hanya di dalam teknis pelaksanaan shalat tersebut. Lihat Muhammad bin Ahmad bin Muhammad bin Ahmad bin Rusyd, *Bidāyah al-Mujtahid wa Nihāyah al-Muqtaṣid*, Juz I, (Surabaya: Dar Ihya al-Kutub al-'Arabiyah, t.t.), 152. Muhammad Syaṭā al-Dimyaṭy juga berpendapat bahwa berdasar hadis riwayat Muslim tersebut hukum shalat gerhana Matahari adalah *sunnah muakkadah*. Lihat Muhammad Syaṭā al-Dimyaṭy, *Hasyiyah 'Iānah at-Ṭālibīn*, Juz I (Libanon: Dar al-Fikr, t.t.), 262.

[22] Hal ini sejalan dengan kaidah fiqih yang berbunyi :

الأصل في الأمر للوجوب إلامادل الدليل على خلافه

Lihat: Abdul Hamid Hakim, *Mabādi' Awwaliyah*, (Jakarta: Sa'adiyah Putra), 8. Dalam hal ini Muhammad bin Ahmad al-Maḥally menambahkan bahwa makna perintah, khususnya yang menggunakan

selalu melanggengkan melaksanakan shalat gerhana ketika terjadi gerhana namun di kesempatan lain, Nabi juga bersabda bahwa shalat selain lima waktu adalah shalat *taṭawwu'* (sunnah). Sebagaimana sabda Nabi SAW berikut ini.

جَآءَ رَجُلٌ إِلَى رَسُوْلِ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ مِنْ أَهْلِ نَجْد ثَائِرَ الرَّأْسِ يُسْمَعُ دَوِيُّ صَوْتِهِ وَلَايُفْقَهُ مَايَقُوْلُ حَتَّى دَنَا فَإِذَا هُوَ يَسْأَلُ عَنِ الْإِسْلَامِ فَقَالَ رَسُوْلُ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ خَمْسُ صَلَوَاتٍ فِي الْيَوْمِ وَاللَّيْلَةِ فَقَالَ هَلْ عَلَيَّ غَيْرُهَا؟ قَالَ لَا إِلَّا أَنْ تَطَوَّعَ ... الحديث (رواه البخاري)[23]

Telah datang seorang lelaki penduduk *Najd* dengan menengadahkan kepalanya, suaranya terdengar lantang namun tidak bisa dipahami apa yang diucapkanya sampai dia mendekat ke Rasulullah SAW. Tiba-tiba dia bertanya tentang Islam. Nabi pun bersabda : (kerjakan) shalat lima kali sehari semalam. Kemudian dia bertanya: apakah masih ada kewajiban (shalat) yang lain? Nabi bersabda: 'Tidak, kecuali kamu melaksanakan *taṭawwu'* (shalat sunnah)'.... (HR. al-Bukhari).

---

bentuk إِفْعَلْ mempunyai banyak kemungkinan makna di antanya adalah wajib dan sunnah. Menurut mayoritas ulama makna perintah adalah wajib. Namun ketika ada *'alaqah* (kondisi lain yang mempengaruhinya) maka makna perintah bisa bergeser di antara wajib dan sunnah. Lihat: Muhammad bin Ahmad al-Maḥally, *Ḥāsyiyah al-'Allāmah al-Bannāny,* Juz I, (Indonesia: Dar Ihya' al-Kutub al-'Arabiyyah), 372.

[23] Muhammad bin Ismail al-Bukhāry, *Matn al-Bukhāry*, Juz I, (Semarang: Toha Putera,t.t.), 17.

Berkaitan dengan waktu pelaksanaan shalat gerhana Matahari, dalam hadis riwayat Muslim tersebut Nabi juga mengisyaratkan waktu pelaksanaannya yaitu sampai Matahari pulih kembali. Sehingga mengetahui kapan mulai terjadi gerhana dan kapan berakhirnya menjadi penting karena berkaitan dengan syarat sah shalat.

Para ulama sepakat bahwa salah satu syarat sah shalat adalah mengetahui masuk waktu shalat tersebut. Al-Malibary[24] menegaskan bahwa barang siapa melaksanakan shalat tanpa mengetahui waktunya maka shalatnya tidak sah walau pun ia melaksanakannya tepat pada waktunya. Karena yang jadi pedoman sah atau tidaknya suatu ibadah adalah adanya keyakinan pada diri yang melaksanakan ibadah (*ẓann al-mukallaf*) dan ibadah tersebut benar-benar dilaksanakan tepat pada waktu ibadah yang dimaksud (*ma fiy nafs al-amr*). Pendapat senada juga ditulis oleh al-Baijury.[25] Ia berpendapat bahwa orang yang akan melaksanakan shalat gerhana harus yakin bahwa gerhana sedang terjadi. Apabila ia ragu-ragu maka ia tidak diperbolehkan melakukan shalat gerhana

---

[24] Al-Dimyaṭy, *Ḥāsyiyah 'Iānah at-Ṭālibīn,* Juz I, 115. Lihat juga: Abi Yahya Zakariya, *Fath al-Wahhāb,* Juz I, (Semarang: Usaha Keluarga, t.t.), 48.

[25] Ibrahim al-Baijury, *Ḥāsyiyah as-Syaikh Ibrāhim al-Baijury,* Juz I, (Libanon: Dar al-Fikr, t.t.), 238.

tersebut karena pada prinsipnya (*aṣal*) gerhana tidak terjadi. Sebagaimana kaidah fiqih yang berbunyi:

الأصل العدم [26]

"Asal dari sesuatu hal adalah ketiadaan".

Berkaitan dengan waktu shalat gerhana Matahari, Ibn Ali Nawawy[27] menambahkan bahwa awal waktu shalat gerhana Matahari dimulai ketika terjadi perubahan bentuk Matahari karena tertutup piringan Bulan sedangkan waktu shalat gerhana Matahari berakhir ketika Matahari telah pulih seperti semula atau Matahari sudah terbenam walaupun masih dalam keadaan gerhana.

Secara astronomis, awal gerhana sebagai awal waktu shalat gerhana dan akhir gerhana sebagai akhir waktu shalat gerhana disebut kontak pertama dan kontak terakhir.[28] Waktu kontak pertama dan terakhir bisa diprediksi dengan program

---

[26] Abdul Hamid Hakim, *as-Sulam,* (Jakarta: as-Sa'diyah Putera, t.t.), 55.

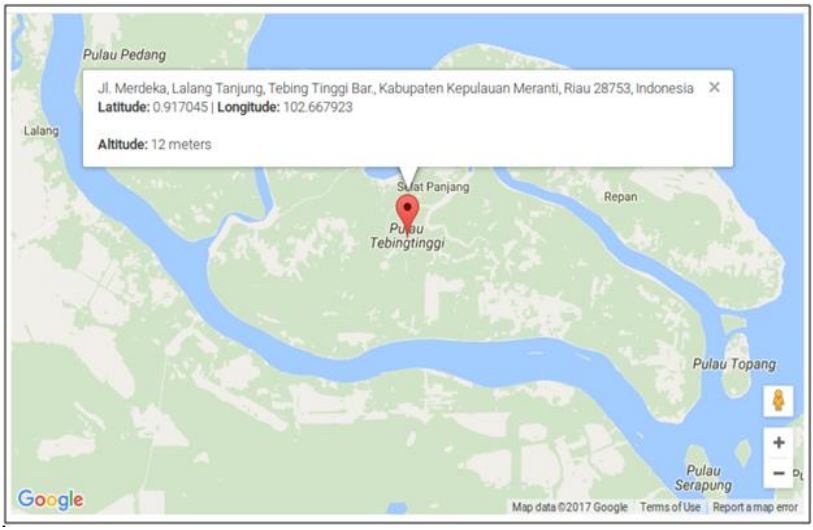
[27] Muhammad bin Amr bin Ali Nawawy, *Nihayah az-Zain fiy Irsyad al-Mubtadiin,* (Libanon: Dar al-Fikr, t.t.), 110. Lihat juga: as-Syarbiny, *al-Iqna' fiy Hilli,* Juz I, 163.

[28] Kontak pertama didefinisikan sebagai kondisi ketika fase gerhana parsial dimulai atau dalam artian ketika piringan Bulan mulai menutupi piringan Matahari. Sebaliknya kontak terakhir (atau kontak ke-4 dalam gerhana Matahari total atau cincin) adalah kondisi ketika fase gerhana parsial berakhir atau dalam artian ketika piringan Bulan meninggalkan piringan Matahari. Espenak, *Glossary of Solar Eclipse*, akses 27 Mei 2017, 11:24:50 WIB. Lihat juga: Muhyiddin Khazin, *Ilmu Falak dalam Teori dan Praktik,* (Yogyakarta: Buana Pustaka, 2004), 188.

sephem yang memanfaatkan data ephemeris JPL NASA seri DE200 yang terbukti memiliki ketelitian yang baik dan sangat dekat laporan hasil observasi sebagaimana diuraikan sebelumnya.

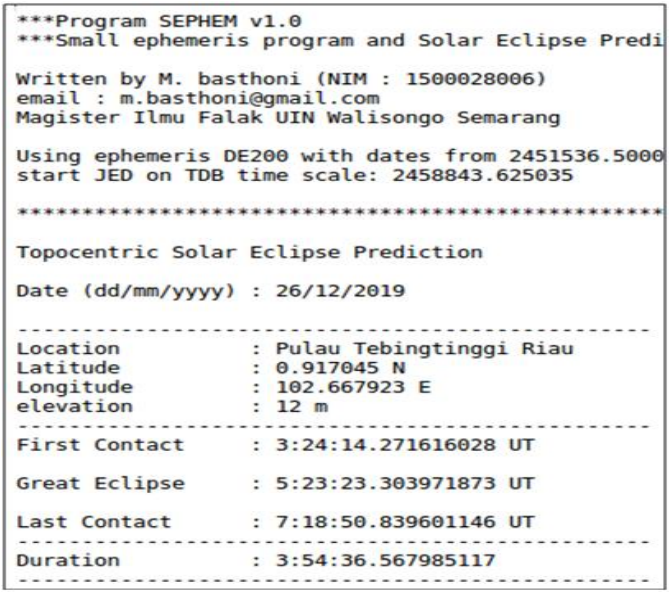
Untuk itu dengan menggunakan program *sephem* yang memanfaatkan data ephemeris JPL NASA seri DE200 tersebut, penulis akan menyajikan perhitungan awal dan akhir waktu shalat gerhana Matahari cincin pada 29 Desember 2019 di Pulau Tebingtinggi Riau[29] dengan koordinat sebagaimana Gambar 4.15 di bawah ini.

[29] Menurut prediksi dari kalkulasi Espenak, tanggal 29 Desember 2019 akan terjadi gerhana Matahari Cincin yang salah satunya melewati Sumatra. Lihat: Fred Espenak, *Solar Eclipses 2011-2020*, https://eclipse.gsfc.nasa.gov/SEdecade/SEdecade2011.html, akses 30 Mei 2017, 10:20:15 WIB.



Gambar 4. 15. Koordinat Pulau Tebingtinggi Riau
(sumber: www.gps-coordinates.com)

Dengan mengikuti alur yang telah diuraikan sebelumnya tentang cara menjalankan program *sephem* dan data-data yang diperlukan dimasukkan dengan lengkap maka akan didapatkan prediksi terjadinya gerhana Matahari cincin di lokasi yang telah ditentukan sebelumnya (Gambar 4.15).



Gambar 4.16. Prediksi gerhana Matahari di Pulau Tebingtinggi pada 26 Desember 2019.

Data pada Gambar 4.16 tersebut menunjukkan bahwa gerhana Matahari cincin akan terjadi mulai pukul 3:24:14 UT dan berakhir pada pukul 7:18:50 UT. Karena zona waktu untuk wilayah Pulau Tebingtinggi adalah UT + 7 jam maka penduduk Pulau Tebingtinggi bisa memulai melaksanakan shalat gerhana Matahari pukul 10:24:14 WIB sampai dengan pukul 14:18:50 WIB. Waktu yang dimiliki oleh penduduk Pulau Tebingtinggi untuk melaksanakan shalat gerhana Matahari juga cukup panjang yaitu 3 jam 54 menit 36 detik.

BAB V
PENUTUP

A. Kesimpulan

Berdasarkan pembahasan yang telah diuraikan dalam bab-bab sebelumnya, dapat diambil kesimpulan sebagai berikut.

1. Algoritma perhitungan data *ephemeris* berbasis JPL NASA pada prinsipnya melalui 2 (dua) langkah utama yaitu mengubah atau mengkonversi file data blok yang masih dalam format file ASCII menjadi file *binary* menggunakan program *asc2eph.* Selanjutnya membaca file *binary* tersebut untuk memperoleh data *ephemeris* benda langit dalam rentang waktu yang dikehendaki. Sedangkan algoritma perhitungan gerhana Matahari total atau cincin menggunakan 3 (tiga) parameter utama, yaitu penjumlahan semi-diameter Matahari – Bulan (*sum_SD*), selisih bujur Matahari - Bulan ($\Delta\lambda$) serta sudut elongasi antara Matahari dan Bulan (E). Jika $\Delta\lambda$ <= *sum_SD* dan terjadi sebelum waktu puncak gerhana, maka saat itu adalah waktu kontak pertama, jika $\Delta\lambda$ >= *sum_SD* dan terjadi setelah waktu puncak gerhana, maka saat itu adalah waktu kontak terakhir. Sedangkan puncak gerhana

terjadi ketika nilai E paling kecil dalam rentang waktu perkiraan terjadinya gerhana Matahari.

2. Ada kesamaan karakteristik antara seri DE yang satu dengan yang lain, yaitu pada interval 0,01 detik atau lebih kecil semua seri DE cenderung stabil pada nilai selisih tertentu. Lonjakan selisih akurasi terjadi pada interval data *ephemeris* 0,1 detik. Sedangkan di awal interval (1 detik) ditemukan dua pola yaitu semua interval mempunyai selisih yang sama dan pola yang ke dua antara DE200 dan seri DE yang lain berbeda jauh selisih akurasinya. Selanjutnya karakter akurasi DE405 sama dengan DE406 pada semua interval dalam peristiwa gerhana Matahari yang diuji pada penelitian ini. Sedangkan seri DE200 pada interval setelah 1 detik (lebih kecil dari 1 detik) akurasinya untuk memprediksi waktu gerhana Matahari total atau cincin di kota tertentu selalu lebih baik dibanding dengan seri DE yang lain kemudian disusul oleh DE421 dan yang terakhir adalah seri DE405 dan DE406.

B. Saran

Perhitungan data *ephemeris* berbasis *Development Ephemeris* (DE) JPL NASA dalam penelitian ini masih dalam tahap awal. Penelitian yang menggunakan algoritma yang dikembangkan oleh Paul J. Heafner sebagai *tool* untuk

mengekplorasi data JPL NASA ini masih terbatas pada seri DE maksimal seri DE42x, sehingga algoritma Heafner ini penting untuk diekplorasi lebih lanjut sehingga bisa meng*cover* semua seri DE yang ada.

Walau pun algoritma Heafner ini tidak menyediakan fitur untuk komputasi gerhana Matahari, namun dengan data *ephemeris* yang dihasilkan bisa dieksplorasi lebih lanjut sehingga bisa dikembangkan untuk perhitungan gerhana Matahari total dan cincin sebagaimana penulis sajikan dalam penelitian ini. Sehingga berangkat dari data *ephemeris* dari penelitian ini masih bisa dikembangkan lebih lanjut untuk perhitungan jenis gerhana yang lain bahkan juga dimungkinkan untuk perhitungan ilmu falak yang lain misal penentuan awal bulan kamariyah berbasis data *ephemeris* JPL NASA.

C. Penutup

Rasa syukur penulis haturkan kepada Allah SWT. yang telah memberikan petunjuk serta kekuatan lahir dan batin sehingga penulis dapat menyelesaikan tesis ini. Meskipun telah berupaya dengan optimal, penulis menyadari bahwa tesis ini masih memiliki kelemahan dan kekurangan dari berbagai segi dan jauh dari sempurna. Sehingga saran dan kritik konstruktif penulis harapkan untuk kebaikan dan kesempurnaan tesis ini.

Akhirnya penulis berharap dan berdo'a semoga tesis ini dapat memberikan manfaat bagi penulis khususnya dan para pembaca umumnya. *Āmīn yā rabbal 'ālamīn.*

DAFTAR KEPUSTAKAAN

Sumber Jurnal Ilmiah

Bretagnon, P., *Theorie du movement de l'ensemble des planets. Solution VSOP82*, Astron. Astophys. 114, 1982.

Chapront, M. Touze and J. Chapront, *ELP 2000-85: A Semi-analiytical Lunar Ephemeris Adequate for Historical Times*, Astron. Astrophys. 190, 1988.

Gontier, A.M., et.al, "Maintenance of ICRF Using the Most Stable Sources" dalam *The International Celestial Reference System and Frame, ICRS Center Report for 2001-2004*, IERS Technical Note, No. 34, (Jerman: IERS ICRS Center, 2006).

Hambali, Slamet, "Astronomi Islam dan Teori Heliocentris Nicolaus Copernicus" dalam *Al-Ahkam*, Volume 23, Nomor 2, Oktober 2013.

Jayusman, Muhammad, "Fenomena Gerhana dalam Wacana Hukum Islam dan Astronomi", dalam *Al-'Adalah*, Vol. X, No. 2 Juli 2011.

Simon, J.-L., et.al, *New Analytical Planetary Theories VSOP2013 and TOP2013*, Astronomy & Astrophysics, A&A 557, A49, (2013): 3, doi: 10.1051/0004-6361/201321843.

Standish, E.M., *Orientation of the JPL Ephemerides, DE200/LE200, to the Dynamical Equinox of J2000*, Astronomy & Astrophysics, 114, (1982).

Sumber Buku

Al-Baijury, Ibrahim, *Hasyiyah as-Syaikh Ibrahim al-Baijury*, Juz I, (Libanon: Dar al-Fikr, t.t.).

Al-Maḥally, Muhammad bin Ahmad, *Hasyiyah al-'Allamah al-Bannany,* Juz I, (Indonesia: Dar Ihya' al-Kutub al-'Arabiyyah).

Anugraha, Rinto, *Mekanika Benda Langit*, (Yogyakarta: Lab. Fisika Material dan Instrumentasi Jurusan Fisika FMIPA UGM, 2012).

Anwar, Syamsul, *Interkoneksi Studi Hadis dan Astronomi*, (Yogyakarta: Suara Muhammadiyah, 2011).

As-Syarbiny, Muhammad, *al-Iqna' fiy Hilli Alfadh Abiy Syuja',* Juz I.

At-Ṭabarī, Muhammad bin Jarīr , *Jāmi' al-Bayān fī Ta'wīl al-Qurān*, Jil. 7, (Beirut:Dar al-Kutub al-'ilmiah, 1999).

Azhari, Susiknan, *Ensiklopedi Hisab Rukyat,* (Yogyakarta: Pustaka Pelajar, 2012).

----------, *Ilmu Falak dalam Teori dan Praktek*, (Yogyakarta: Lazuardi, 2001).

Bukhary, Muhammad bin Ismail, *Matn al-Bukhary*, Juz I, (Semarang: Toha Putera,t.t.).

Butar-Butar, Arwin Juli Rakhmadi, *Khazanah Astronomi Islam Abad Pertengahan,* (Purwokerto: UM Purwokerto Press, 2016).

Departemen Agama RI, *al-Qur'an dan Terjemahnya,* (Semarang: CV Al Waah, 2004).

Dimyathy, Muhammad Syatha, *Hasyiyah 'Ianah at-Thalibin,* Juz I (Libanon: Dar al-Fikr, t.t.), 262.

Green, Robin M., *Spherical Astronomy,* (Cambridge: Cambridge University Press, 1985).

H.M. Nautical Almanac Office, *Explanatory Supplement to The Astonomical Ephemeris and The American Ephemeris and Nautical Almanac*, (London: Her Majesty's Stattionery Office, 1961).

Hakim, Abdul Hamid, *as-Sulam,* (Jakarta: as-Sa'diyah Putera, t.t.).

----------, *Mabadi' Awwaliyah,* (Jakarta: Sa'adiyah Putra).

Heafner, Paul J., *Fundamental Ephemeris Computations for Use with JPL Data*, (Virginian: Willmann-Bell, Inc., 1999), 153.

Ibn Ali Nawawy, Muhammad bin Amr, *Nihayah az-Zain fiy Irsyad al-Mubtadiin,* (Libanon: Dar al-Fikr, t.t.).

Ibn Rusyd, Muhammad bin Ahmad bin Muhammad bin Ahmad, *Bidayah al-Mujtahid wa Nihayah al-Muqtashid*, Juz I, (Surabaya: Dar Ihya al-Kutub al-'Arabiyah, t.t.).

Ibn Katsir, Ismail al-Qurasy, *Tafsir al-Qur'an al-'Adhim,* (ttp.: Syirkah an-Nur Asia, tt.).

Idpath, Ian R , *Dictionary of Astronomy*, (New York: Oxford University Press, 1997).

Izzuddin, Ahmad, *Fiqih Hisab Rukyah*, (Jakarta: Penerbit Erlangga, 2007).

----------, *Ilmu Falak Praktis*, (Semarang: Pustaka Rizki Putra, 2012).

Jauhari, Tantowi, *Jawahir fi Tafsir al-Qur'an al-Karim*, juz 14, (Mesir: Musthofa al-Baaby al-Khaaly wa Awladuhu, t.t.)

Kadir, A., *Formula Baru Ilmu Falak,* (Jakarta: Amzah, 2012).

Karttunnen, Hannu, et.al, 1996, *Fundamental Astronomy*, (New York: Springer,1995).

Khazin, Muhyidin, *Ilmu Falak dalam Teori dan Praktik*, (Yogyakarta:Buana pustaka, 2004).

al-Marāghy, Ahmad Mushthafa, *Tafsir al-Marāghy*, jil. 8, (Beirut-libanon: Dar al-Fikr, t.t.).

Meeus, Jean, *Astronomical Algorithms*, Second English Edition, (Virginia: William-Bell, Inc., Richmond, 1998)

Moleong, Lexy J., *Metodologi Penelitian Kualitatif,* (Bandung: PT Remaja Rosdakarya, 2001).

Montenbruck, Oliver, *Astronomy on the Personal Computer,* (New York: Springer, 1994).

Muhammad bin Amr bin Ali Nawawy, *Nihayah az-Zain fiy Irsyad al-Mubtadiin,* (Libanon: Dar al-Fikr, t.t.).

Muslim bin al-Hajjaj, *Shahih Muslim,* Jilid I, (Libanon: Dar al-Fikr, 1992).

Oriti, Ronald. A., et.al., *Introduction to Astronomy,* (California: Glencoe Publishing co. Inc., 1977)

Pannekoek, A., *A History of Astronomy*, (New York: Dover Publication, Inc., 1989).

al-Qasimy, Muhammad Jamal al-Din, *Tafsir al-Qasimy*, jil. 3, (Beirut-libanon: Muassasah at-Tarikh al-Araby, 1994).

al-Qurṭuby, Muhammad bin Ahmad al-Anshary, *al-Jami' li Ahkam al-Qur'an,* jil. 4, (Beirut-libanon: al-Maktabah al-'Ashriyah, 2009).

Qutub, Sayyid, *Fiy Dhilal al-Qur'an,* jil.4, (Beirut-Libanon: Dar as-Syuruq, 1992).

Raharjo, Budi, *Kumpulan Solusi Pemrograman C*, (Bandung: Informatika, 2016).

Robinson, Leif J., *Philip's Astronomy Encyclopedia,* (London: Octopus Publishing Group, 2002).

al-Ṣabuny, Muhammad Ali, *Shafwat at-Tafasir*, jil. 2, (Beirut-Libanon: Dar ar-Rasyad, 1988).

al- Ṣawy, Ahmad al-Maliky, *Hasyiyah al-'Allamah as-Shawy 'ala Tafsir al-Jalalain,* (Beirut: Dar al-fikr, tt.).

Setyanto, Hendro, *Membaca Langit*, (Jakarta: al-Ghuraba, 2008).

Shihab, Quraish *Tafsir al-Misbah*, Vol. 5 (Jakarta: Lentera Hati, 2002).

----------, *Tafsir al-Misbah*, Vol. 11 (Jakarta: Lentera Hati, 2002).

----------, *Tafsir al-Misbah*, Vol. 13 (Jakarta: Lentera Hati, 2002).

Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif dan R & D*, (Bandung: Penerbit Alfabeta, 2006).

Tim Penyusun Naskah IDI Hukum, *Islam Untuk Disiplin Ilmu Astronomi; Buku Dasar Pendidikan Agama Islam Pada Perguruan Tinggi Umum Jurusan/Program Studi Astronomi*, (Jakarta: Depag RI, 2000).

Yusuf, A. Muri, *Metode Penelitian: Kuantitatif, Kualitatif dan Penelitian Gabungan*, (Jakarta: Prenadamedia Group, 2014).

Zakariya, Abi Yahya, *Fath al-Wahhab*, Juz I, (Semarang: Usaha Keluarga, t.t.).

Zuhaili, Wahbah, *Tafsīr al-Munīr*, Jilid 23 (Beirut-libanon: Dar al-Fikr al-Ma'ashir, 1998).

Sumber Lain

About JPL Solar System Dynamics, https://ssd.jpl.nasa.gov/?about, 11 April 2017, 12.30 WIB.

Biography: Fred Espenak, http://www.eclipse-chasers.com/article/SEC2014/speaker/Fred Espenak Bio.pdf, akses 23 Mei 2017, 01:16 WIB.

BMKG Multimedia, *Pengamatan Gerhana Matahari Total di Palu*, http://media.bmkg.go.id/gmt.bmkg?s=gmt_palu, akses 13 Juni 2017, 9.00 WIB.

Description of JPL Solar System Ephemeris, http://www.cv.nrao.edu/~rfisher/Ephemerides/ephem_descr.html, 1 Oktober 2016, 12:30 WIB.

Djamaluddin, T, *Gerhana Matahari Cincin 26 Januari 2009: Salat Gerhana*, http://t-djamaluddin.spaces.live.com diakses pada tanggal 23 Maret 2017.

Effendi, Djamhur, *Sekelumit Penanggalan Qomariah dan Gerhana Bulan*, http://www.nu.or.id diakses pada tanggal 23 Maret 2017. 13:40 WIB.

Espenak, Fred, *2005 Annular Solar Eclipse Photo Gallery A (Overview)*, http://mreclipse.com/Sephoto/ASE2005/ASE2005galleryA . html, akses 18 Mei 2007, 7:14 WIB.

Espenak, Fred, *Glossary of Solar Eclipse Terms*, https://eclipse.gsfc.nasa.gov/Sehelp/Seglossary. html, akses 20 Mei 2017, 10:20:14 WIB.

Espenak, Fred, *Glossary of Solar Eclipse Terms*, https://eclipse.gsfc.nasa.gov/ Sehelp/Seglossary.html, akses 27 Mei 2017, 11:24:50 WIB.

Espenak, Fred, *Solar Eclipse Predictions with JPL DE405*, https://eclipse.gsfc.nasa.gov/help/de405-predictions.html, akses 15 April 2017, 10:25 WIB.

Espenak, Fred, *Solar Eclipses 2011-2020*, https://eclipse.gsfc.nasa.gov/SEdecade/SEdecade2011.html , akses 30 Mei 2017, 10:20:15 WIB.

Fred Espenak – Public Lectures and Speaking Engagements, http://www.mreclipse.com/ main/lectures.html#Espenak, akses 23 Mei 2017, 01:16 WIB.

ftp://ssd.jpl.nasa.gov/pub/eph/ planets/ascii/DE405/header.405

ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/, 11 April 2017, 13:15 WIB.

ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/, akses 11 April 2017, 13:15 WIB.

History & Archives, https://www.jpl.nasa.gov/about/ history.php, akses 11 April 2017, 11.35 WIB.

HORIZONS System, http://ssd.jpl.nasa.gov/?horizons, 11 April 2017, 12:45 WIB.

http://mfdonline.bps.go.id/index.php?link=hasil_pencarian, akses 12 juni 2017, 11:15 WIB.

http://referensi.data.kemdikbud.go.id/tabs.php?npsn=60200659, 12 Juni 2017, 11:20 WIB.

https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/SE2016Mar09Tgoogle.html, akses 15 April 2017, 11:00 WIB

https://eclipse.gsfc.nasa.gov/SEgoogle/SEgoogle2001/SE2017Aug21Tgoogle.html, akses 15 April 2017, 12:10 WIB.

Jet Propulsion Laboratory, http://www.caltech.edu/content/jet-propulsion-laboratory, akses 11 April 2017, 11:30 WIB.

JPL PLANETARY AND LUNAR EPHEMERIDES : Export Information, ftp://ssd.jpl.nasa.gov/pub/eph/planets/ README.txt, 12 April 2017, 08:17 WIB.

JPL Planetary and Lunar Ephemerides, https://ssd.jpl.nasa.gov/?planet_eph_export, 11 April 2017, 13:00 WIB.

www.gps-coordinates.net

Lampiran 1.
Source code Small Ephemeris Program (file*: sephem.c*)

```
/*************************************************************
Project            : sephem
Filename           : sephem.c
Algorithms Author  : Joe Heafner.
Purpose            : 1. A small command line ephemeris program.
                     2. Total or Annular Solar Eclipse Prediction
                        (by M. Basthoni)

Modified by M. Basthoni - 17/4/2017 - 00:23:04
         Added : Topocentric Solar Eclipse Prediction (Total or Annular)
*************************************************************/

/* Header Files ***************************************/
#include  <stdio.h>
#include  <stdlib.h>
#include  <string.h>
#include  <time.h>
#include  <errno.h>
#include  "astrolib.h"
#include  "support.h"

/* Function Prototypes ********************************/
void ParseCmdLine(int argc, char *argv[], char *InFile);
void PrintBanner(void);

/* Globals ********************************************/

char szVersion[] = "SEPHEM v1.0";
char szLogFile[] = "sephem.log";
extern short int NUMDE;
extern double SS[];
extern double obsr_lon, obsr_lat, obsr_ele;

/*************************************************************
Name:    main
Purpose: Main routine for sephem.
Inputs:  argc - Number of command-line arguments.
         argv - Pointer to array of command-line arguments.
Outputs: None.
Returns: 0 if execution successful.
Status:  Finished.
Errors:  None known.
*************************************************************/
int main (int argc, char *argv[]) {

  char InFile[MAX_NAME_SIZE+1] = "";
  char location[MAX_NAME_SIZE+1] = "";
  char RA[100], DEC[100], body[100],nol[100];
  char line[1024];
  char PL[20];
  int RedGeo = 0, SolSysSum = 0, Place = 0;
  int i, Targ, Cent, NumSteps, counter;
  double TDB = 0.0;
  double xx, yy, zz;
  double xxdot, yydot, zzdot;
  double r, RA_hours, DEC_deg, DateInc;
  double StarData[6], p3[6];
```

```
  double element[6], state[6], mu, mass;

  double dpsi, deps, dpsidot, depsdot, EquationOfTime, TrueEps, TrueEpsDot,
TEps, u, l0;
  char temp[30], right_buffer[30];
  char t[30], dp[30];

  double SemiDiameter, s;

  double HorizontalMoonParallax;

  double  Alpha, Delta;
  double  Lambda, Beta, r1[6], r2[6];

  int  tahun, bulan, hari;
  double jam, jam2, menit, detik;

  int  year, month, day;
  double utc;

  /*  note : 10/5/2017 - 22:29:45 - increment of data */
  int increment;
  double const_increment;
  char label_increment[40];

  /* Open the log file */
  if (LogOpen(szLogFile) == FALSE) {
    fprintf(stderr,
      "Could not open log file '%s': %s\n\n", szLogFile, strerror(errno));
    exit(1);    /* Exit with an error code */
  }

  PrintBanner();
  ParseCmdLine(argc, argv, InFile);

  if (strlen(InFile) == 0) {
    /* Prompt user for file name */
    do {
      fprintf(stdout, "File name to use (XXXX to end, ex. : de405_2000.bin):
");
      fflush(stdout);
      fgets(InFile, MAX_NAME_SIZE+1, stdin);
      Trim(InFile);
    } while (strlen(InFile) == 0);
  }

  /* We have a filename now */
  if (strcmp(InFile, "XXXX") == 0) {
    LogMsg(stdout, "\nOK\n");
    LogClose();
    remove(szLogFile);
    return (0);
  }

  /* Test to see if filename exists */
  if (!fexist(InFile)) {
    LogMsg(stdout, "Requested input file does not exist.\n");
    LogMsg(stdout, "Specify another file or move to another directory.\n");
    LogMsg(stdout, "\nOK\n");
    LogClose();
    exit(1);
  }
```

```
  /* Open the input file and read in the header info */
  if (ephopn(InFile) == NULL) {
    LogMsg(stderr, "An error occurred in ephopn().\n");
    LogClose();
    exit(1);
  }

  LogMsg(stdout,
    "Using ephemeris DE%d with dates from %lf to %lf\n",
    NUMDE, SS[0], SS[1]);

  do {
    fprintf(stdout, "Enter Day: ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%d", &day) != 1);

  do {
    fprintf(stdout, "Enter Month: ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%d", &month) != 1);

  do {
    fprintf(stdout, "Enter Year: ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%d", &year) != 1);

  do {
    fprintf(stdout, "Enter UTC as hh.mmssss: ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &utc) != 1);

  if(utc==0){
    utc=0/1;
  }


  Cal2JED(month, day, year, utc, 4, utc, utc, 0, &TDB);
  counter = 0;
  if(counter==0){
      TDB = TDB - 0.0003723;
  }

  LogMsg(stdout,"start JED on TDB time scale: %lf\n", TDB);


  do {
    do {
      /* 12/5/2017 - 14:04:00 - add Solar Eclipse Prediction  */
      fprintf(stdout, "[1] EPHEMERIS [2] SOLAR ECLIPSE : ");
fflush(stdout);
      fgets(line, sizeof(line), stdin);
    } while (sscanf(line, "%d", &RedGeo) != 1);
  } while ((RedGeo != 1) && (RedGeo != 2));


  if (RedGeo == 1) {
    do {
      do {
        fprintf(stdout, "[1] APPARENT [2] TOPOCENTRIC : ");
```

```
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%d", &Place) != 1);
} while ((Place < 1) || (Place > 2));


if (Place == 2) {
  do {
    obsr_lat = 0.0;
    fprintf(stdout, "Enter latitude [N = +](dd.mmss): ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &obsr_lat) != 1);
  obsr_lat = deg(obsr_lat) * D2R;

  do {
    obsr_lon = 0.0;
    fprintf(stdout, "Enter longitude [E = +](dd.mmss): ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &obsr_lon) != 1);
  obsr_lon = deg(obsr_lon) * D2R;

  do {
    obsr_ele = 0.0;
    fprintf(stdout, "Enter elevation (meters): ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &obsr_ele) != 1);
}

switch (Place) {
  case 1:
    strcpy(PL, "APPARENT");
    break;
  case 2:
    strcpy(PL, "TOPOCENTRIC");
    break;
}


  Targ = 12;
  do {
    do {
        fprintf(stdout, "\n\n[1] Sun [2] Moon\n\n");
        fprintf(stdout, "Enter body number (1-2): ");
        fflush(stdout);
        fgets(line, sizeof(line), stdin);
    } while (sscanf(line, "%d", &Targ) != 1);
  } while (Targ < 1 || Targ > 2);

  if (Targ==1){
     Targ=11;
  } else
  if (Targ==2){
     Targ=10;
  }

  do {
    do {
        fprintf(stdout, "\n\n[1] Hours [2] Minutes   [3] Seconds\n\n");
        fprintf(stdout, "Enter increment (1-3): ");
        fflush(stdout);
```

```
      fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%d", &increment) != 1);
} while (increment < 1 || increment > 3);

  if(increment==1){
    const_increment = 0.04166670003905890000;
    strcpy(label_increment, "hours");
  } else
  if(increment==2){
    const_increment = 0.00069440016523003600;
    strcpy(label_increment, "minutes");
  } else{
    const_increment = 0.00001160008832812310;
    strcpy(label_increment, "seconds");
  }


do {
  fprintf(stdout, "Increment in %s : ", label_increment);
  fflush(stdout);
  fgets(line, sizeof(line), stdin);
} while (sscanf(line, "%lf", &DateInc) != 1);

DateInc = DateInc*const_increment;


do {
  do {
    fprintf(stdout, "Enter number of steps: ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%d", &NumSteps) != 1);
} while (NumSteps <= 0);

  if (Targ == 1) {
      strcpy(body, "Mercury");
  } else
  if (Targ == 2) {
      strcpy(body, "Venus");
  } else
  if (Targ == 3) {
      strcpy(body, "Earth");
  } else
  if (Targ == 4) {
      strcpy(body, "Mars");
  } else
  if (Targ == 5) {
      strcpy(body, "Jupiter");
  } else
  if (Targ == 6) {
      strcpy(body, "Saturn");
  } else
  if (Targ == 7) {
      strcpy(body, "Uranus");
  } else
  if (Targ == 8) {
      strcpy(body, "Neptune");
  } else
  if (Targ == 9) {
      strcpy(body, "Pluto");
  } else
  if (Targ == 10) {
      strcpy(body, "Moon");
```

```
        } else
        if (Targ == 11) {
            strcpy(body, "Sun");
        } else{
            strcpy(body, "");
        }

      LogMsg(stdout, "\n%s EPHEMERIS OF BODY %d (%s)\n", PL, Targ, body);

      if(Targ==10){
          LogMsg(stdout, ================================\n");
          LogMsg(stdout, "             |             |Ecliptic |Ecliptic
|Apparent  |Apparent   |True         |Semi        |Horizontal\n");
          LogMsg(stdout, " Date & Time | JDE         |Longitude|Latitude
|Right     |Declination|Geocentric   |Diameter    |Parallax  \n");
          LogMsg(stdout, "             |             |(DEG)    |(DEG)
|Ascension |(DEG)      |Distance     |(ArcMinute) |(DEG)     \n");
          LogMsg(stdout, "             |             |         |
|(DEG)     |           |(KM)         |            |          \n");
          LogMsg(stdout, "================================\n");
      }else
      if(Targ==11){
          LogMsg(stdout, "================================\n");
          LogMsg(stdout, "             |             |Ecliptic  |Ecliptic
|Apparent  |Apparent   |True      |Semi       |True     |Equation\n");
          LogMsg(stdout, " Date & Time | JDE         |Longitude |Latitude
|Right     |Declination|Geocentric|Diameter   |Obliquity|Of\n");
          LogMsg(stdout, "             |             |          |(DEG)     |(ArcSec)
|Ascension |(DEG)      |Distance  |(ArcMinute)|(DEG)    |Time\n");
          LogMsg(stdout, "             |             |          |          |
|(DEG)     |           |(AU)      |           |          |\n");
          LogMsg(stdout, "================================\n");
      }

      do {
        Reduce(TDB, Targ, Place, StarData, p3);
        r = sqrt(p3[0] * p3[0] + p3[1] * p3[1] + p3[2] * p3[2]);
        RA_hours = atan2(p3[1], p3[0]) * R2H;

        if (RA_hours < 0.0) {
          RA_hours += 24.0;
        }

        FmtDms(RA_hours*H2R*R2D, 2, 0, RA);
        DEC_deg = asin(p3[2] / r) * R2D;

        FmtDms(DEC_deg, 2, 0, DEC);

        if (Targ == 10) {
            r = r * AU2KM;
            s = 358473400;
        } else
        if (Targ == 11) {
            r = r;
            s = 959.63;
        } else{
            r = r;
            s = 0.0;
        }

        SemiDiameter = (s/r)/60;

        GetDpsiDeps(TDB, &dpsi, &deps, &dpsidot, &depsdot);
```

```
        /* Get true obliquity */
        Obliquity(J2000, TDB, 1, &TrueEps, &TrueEpsDot);

        Alpha = RA_hours * H2R;
        Delta = DEC_deg * D2R;
        r1[0] = Alpha;
        r1[1] = Delta;
        r1[2] = 1.0;
        r1[3] = 0.0;
        r1[4] = 0.0;
        r1[5] = 0.0;

        /* Change to rectangular variables */
        Pol2Rec(r1, r2);
        /* Transform */
        Eq2Ecl(r2, 0, TrueEps, r2);
        /* Change to polar variables */
        Rec2Pol(r2, r2);

        Lambda = r2[0] * R2D;
        Beta = r2[1] * R2D;

        if(Targ==11){
            Beta = Beta*3600;
        }else{
            Beta = Beta;
        }

        FmtDms(Lambda, 2, 0, Lam);
        FmtDms(Beta, 2, 0, Bet);

        JED2Cal(TDB, &tahun, &bulan, &hari, &jam);

        /*  note : 3/5/2017 - 22:27:00 - EquationOfTime */

        u=(TDB-2451545)/36525;
        l0=amodulo((280.46607+36000.7698*u),360)*D2R;
        EquationOfTime=(-1*(1789+237*u)*sin(l0)-(7146-62*u)*cos(l0)+(9934-
14*u)*sin(2*l0)-(29+5*u)*cos(2*l0)+(74+10*u)*sin(3*l0)+(320-4*u)*cos(3*l0)-
212*sin(4*l0))/1000;

        HorizontalMoonParallax = asin(6378.14 / r)*R2D;


              LogMsg(stdout, " %d/%d/%d %2.0lf %lf  %5.5lf ", hari, bulan,
tahun, jam, TDB, Lambda);

                    LogMsg(stdout, " %5.5lf ", Beta);
              LogMsg(stdout, " %5.5lf ", RA_hours*H2R*R2D);
              LogMsg(stdout, " %5.5lf ", DEC_deg);
              LogMsg(stdout, "   %5.5lf ", r);
              LogMsg(stdout, "   %5.5lf ", SemiDiameter);

              if(Targ==10){
                LogMsg(stdout, " %5.5lf \n", HorizontalMoonParallax);
              } else if (Targ==11){
                LogMsg(stdout, " %5.5lf ", TEps);

                LogMsg(stdout, " %5.5lf\n", EquationOfTime);
              }

        counter++;
```

```
        TDB += DateInc;
      } while (counter < NumSteps);

          if(Targ==10){
                  LogMsg(stdout, "============================\n");
                  } else {
                  LogMsg(stdout, "============================\n");
                  }
}

   else {/*       SOLAR ECLIPSE */

        int Targ_m, Targ_s, Elo_c;
        char RA_m[100], DEC_m[100], RA_s[100], DEC_s[100];
        double xx_m, yy_m, zz_m, xx_s, yy_s, zz_s;
        double xxdot_m, yydot_m, zzdot_m, xxdot_s, yydot_s, zzdot_s;
        double r_m, r_m_radii, RA_hours_m, DEC_deg_m, r_s, r_s_radii,
RA_hours_s, DEC_deg_s;
        double p3_m[6], p3_s[6];
        double ellongation, Elo, Elo_t, TDB_awal;

        double Alpha_m, Alpha_s, Delta_m, Delta_s, Lambda_m, Lambda_s,
Beta_m, Beta_s, SemiDiameter_m, SemiDiameter_s;
        double r1_m[6], r1_s[6], r2_m[6], r2_s[6];
        char Lam_deg_m[40], Lam_deg_s[40], Bet_deg_m[40], Bet_deg_s[40],
SD_m[40], SD_s[40];

        double sum_SD;
        char sum_SD_deg[40];

        double delta_Longitude, delta_Latitude;
        double C1_time, C4_time, dur_time;
        int flag, flag2;
        int counter_c1, counter_c4;
        int jam_elo, menit_elo, jam_c1, menit_c1, jam_c4, menit_c4, jam_dur,
menit_dur;
        double detik_elo, detik_c1, detik_c4, detik_dur;

        double obsr_lat_GM, obsr_lon_GM;
        double waktu_c4, waktu_c1;
        char lat[2], lon[2];

    TDB_awal = TDB;
    counter = 0;
    flag = 0;
    flag2 = 0;


      do {
        do {
            fprintf(stdout, "\n\n[1] Hours [2] Minutes   [3] Seconds\n\n");
            fprintf(stdout, "Enter increment (1-3): ");
            fflush(stdout);
            fgets(line, sizeof(line), stdin);
        } while (sscanf(line, "%d", &increment) != 1);
      } while (increment < 1 || increment > 3);

        if(increment==1){
          const_increment = 0.04166670003905890000;
          strcpy(label_increment, "hours");
        } else
        if(increment==2){
          const_increment = 0.00069440016523003600;
```

```
      strcpy(label_increment, "minutes");
    } else{
      const_increment = 0.00001160008832812310;
      strcpy(label_increment, "seconds");
    }

  do {
    fprintf(stdout, "Increment in %s : ", label_increment);
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &DateInc) != 1);

  DateInc = DateInc*const_increment;

  do {
    do {
      fprintf(stdout, "Enter number of steps: ");
      fflush(stdout);
      fgets(line, sizeof(line), stdin);
    } while (sscanf(line, "%d", &NumSteps) != 1);
  } while (NumSteps <= 0);


Place = 2;

do {
  fprintf(stdout, "\nEnter Location : ");
  fflush(stdout);
  fgets(location, MAX_NAME_SIZE+1, stdin);
  Trim(location);
} while (strlen(location) == 0);

  do {
    obsr_lat = 0.0;
    fprintf(stdout, "Enter latitude [N = +](dd.mmss): ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &obsr_lat_GM) != 1);
  obsr_lat = deg(obsr_lat_GM) * D2R;

  do {
    obsr_lon = 0.0;
    fprintf(stdout, "Enter longitude [E = +](dd.mmss): ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &obsr_lon_GM) != 1);
  obsr_lon = deg(obsr_lon_GM) * D2R;

  do {
    obsr_ele = 0.0;
    fprintf(stdout, "Enter elevation (meters): ");
    fflush(stdout);
    fgets(line, sizeof(line), stdin);
  } while (sscanf(line, "%lf", &obsr_ele) != 1);


  Targ_m = 10; /* untuk Bulan */
  Targ_s = 11; /* untuk Matahari */
  do {
    Reduce(TDB, Targ_m, Place, StarData, p3_m);
    Reduce(TDB, Targ_s, Place, StarData, p3_s);

    /* distance r_m = Bumi-Bulan, r_s = Bumi-Matahari */
```

```
        r_m = sqrt(p3_m[0] * p3_m[0] + p3_m[1] * p3_m[1] + p3_m[2] *
p3_m[2]);
        r_s = sqrt(p3_s[0] * p3_s[0] + p3_s[1] * p3_s[1] + p3_s[2] *
p3_s[2]);
        /* RA Matahari dan Bulan */
        RA_hours_m = atan2(p3_m[1], p3_m[0])*R2H;
        RA_hours_s = atan2(p3_s[1], p3_s[0])*R2H;

        if (RA_hours_m < 0.0) {
          RA_hours_m += 24.0;
        }

        if (RA_hours_s < 0.0) {
          RA_hours_s += 24.0;
        }

        FmtDms(RA_hours_m*H2R*R2D, 2, 0, RA_m);
        FmtDms(RA_hours_s*H2R*R2D, 2, 0, RA_s);

        /* Deklinasi Bulan dan Matahari */
        DEC_deg_m = asin(p3_m[2] / r_m);
        DEC_deg_s = asin(p3_s[2] / r_s);

        FmtDms(DEC_deg_m*R2D, 2, 0, DEC_m);
        FmtDms(DEC_deg_s*R2D, 2, 0, DEC_s);

        u=(TDB-2451545)/36525;
        l0=amodulo((280.46607+36000.7698*u),360)*D2R;
        EquationOfTime=(-1*(1789+237*u)*sin(l0)-(7146-62*u)*cos(l0)+(9934-
14*u)*sin(2*l0)-(29+5*u)*cos(2*l0)+(74+10*u)*sin(3*l0)+(320-4*u)*cos(3*l0)-
212*sin(4*l0))/1000/60;

        /* Sudut elongasi antara Matahari dan Bulan dilihat dari Bumi */
        ellongation = acos(cos(RA_hours_m*H2R -
RA_hours_s*H2R))*cos(DEC_deg_s)*cos(DEC_deg_m)+sin(DEC_deg_s)*sin(DEC_deg_m)
;

        Obliquity(J2000, TDB, 1, &TrueEps, &TrueEpsDot);

        Alpha_m = RA_hours_m*H2R;
        Alpha_s = RA_hours_s*H2R;
        Delta_m = DEC_deg_m;
        Delta_s = DEC_deg_s;

        r1_m[0] = Alpha_m;
        r1_m[1] = Delta_m;
        r1_m[2] = 1.0;
        r1_m[3] = 0.0;
        r1_m[4] = 0.0;
        r1_m[5] = 0.0;

        r1_s[0] = Alpha_s;
        r1_s[1] = Delta_s;
        r1_s[2] = 1.0;
        r1_s[3] = 0.0;
        r1_s[4] = 0.0;
        r1_s[5] = 0.0;

        /* Change Polar to rectangular variables */
        Pol2Rec(r1_m, r2_m);
        Pol2Rec(r1_s, r2_s);

        /* Transform */
```

```
        Eq2Ecl(r2_m, 0, TrueEps, r2_m);
        Eq2Ecl(r2_s, 0, TrueEps, r2_s);

        /* Change rectangular to polar variables */
        Rec2Pol(r2_m, r2_m);
        Rec2Pol(r2_s, r2_s);

        Lambda_m = r2_m[0]; /* radian */
        Beta_m = r2_m[1];

        Lambda_s = r2_s[0];
        Beta_s = r2_s[1];


        FmtDms(Lambda_m*R2D, 2, 0, Lam_deg_m);
        FmtDms(Beta_m*R2D, 2, 0, Bet_deg_m);

        FmtDms(Lambda_s*R2D, 2, 0, Lam_deg_s);
        FmtDms(Beta_s*R2D, 2, 0, Bet_deg_s);

        SemiDiameter_m = (358473400/(r_m*AU2KM))/60/60;
        SemiDiameter_s = (959.63/r_s)/60/60;

        FmtDms(SemiDiameter_m, 2, 0, SD_m);
        FmtDms(SemiDiameter_s, 2, 0, SD_s);

        sum_SD = SemiDiameter_s + SemiDiameter_m;
        FmtDms(sum_SD, 2, 0, sum_SD_deg);

        delta_Longitude = fabs(Lambda_s*R2D - Lambda_m*R2D);

        delta_Latitude = fabs(Beta_s*R2D - Beta_m*R2D);

        JED2Cal(TDB, &tahun, &bulan, &hari, &jam);

        LogMsg(stdout, " %d %d/%d/%d %lf JDE %lf  ", counter+1, hari, bulan,
tahun, jam, TDB);
        LogMsg(stdout, " Lamb_m %s Lamb_s %s", Lam_deg_m, Lam_deg_s);
        LogMsg(stdout, " Beta_m %s Beta_s %s", Bet_deg_m, Bet_deg_s);
        LogMsg(stdout, " RA_m %s RA_s %s", RA_m, RA_s);
        LogMsg(stdout, " DEC_m %s DEC_s %s", DEC_m, DEC_s);
        LogMsg(stdout, " R_m %lf R_s %lf", r_m*AU2KM, r_s);
        LogMsg(stdout, " SD_m %s SD_s %s", SD_m, SD_s);
        LogMsg(stdout, " sum_SD %lf ", sum_SD);
        LogMsg(stdout, " Delta_Long %lf Delta_Lat %lf ", delta_Longitude,
delta_Latitude);
        LogMsg(stdout, " EoT %lf", EquationOfTime);
        LogMsg(stdout, " Elo %lf\n", ellongation); */

    printf("Please wait .........");
    printf("%d of %d Completed\r", counter, NumSteps);

              if(counter==0){
                Elo = ellongation;
                Elo_t = TDB;
                Elo_c = counter;
              }

              if(Elo>ellongation){
                Elo = ellongation;
                Elo_t = TDB;
                Elo_c = counter;
              }else{
```

```
              Elo = Elo;
              Elo_t = Elo_t;
              Elo_c = Elo_c;
            }

            if(delta_Longitude <= sum_SD && flag==0) {
              flag=1;
              counter_c1=counter;
              C1_time = TDB;
            }

            if(delta_Longitude >= sum_SD && flag==1 && flag2==0){
              counter_c4=counter;
              flag2=1;
              C4_time = TDB;
            }

         counter++;
        TDB += DateInc;
      } while (counter < NumSteps);

      if(obsr_lat_GM<0){
        strcpy(lat,"S");
      } else {
        strcpy(lat,"N");
      }
      if(obsr_lon_GM<0){
        strcpy(lon,"W");
      } else {
        strcpy(lon,"E");
      }

      LogMsg(stdout, "\n*********************************\n");
      LogMsg(stdout, "\nTopocentric Solar Eclipse Prediction\n");
      LogMsg(stdout, "\nDate (dd/mm/yyyy) : %d/%d/%d\n", day, month, year);
      LogMsg(stdout, "\n--------------------------------------");
      LogMsg(stdout, "\nLocation          : %s", location);
      LogMsg(stdout, "\nLatitude          : %lf %s", obsr_lat_GM, lat);
      LogMsg(stdout, "\nLongitude         : %lf %s", obsr_lon_GM, lon);
      LogMsg(stdout, "\nelevation         : %0.0lf m", obsr_ele);
      LogMsg(stdout, "\n--------------------------------------\n");

      JED2Cal(C1_time, &tahun, &bulan, &hari, &jam);
      waktu_c1 = jam;
      jam_c1 = (int)waktu_c1;
      menit_c1 = (int)((waktu_c1 - jam_c1)*60);
      detik_c1 = (((waktu_c1 - jam_c1)*60)-menit_c1)*60;
      LogMsg(stdout, "First Contact     : %d:%d:%0.9lf UT
%d\n\n", jam_c1, menit_c1, detik_c1, counter_c1+1);


      JED2Cal(Elo_t, &tahun, &bulan, &hari, &jam);
      jam_elo = (int)jam;
      menit_elo = (int)((jam - jam_elo)*60);
      detik_elo = (((jam - jam_elo)*60)-menit_elo)*60;
      LogMsg(stdout, "Great Eclipse     : %d:%d:%0.9lf UT
%d\n\n", jam_elo, menit_elo, detik_elo, Elo_c+1);

      JED2Cal(C4_time, &tahun, &bulan, &hari, &jam);
      waktu_c4 = jam;
      jam_c4 = (int)waktu_c4;
      menit_c4 = (int)((waktu_c4 - jam_c4)*60);
      detik_c4 = (((waktu_c4 - jam_c4)*60)-menit_c4)*60;
```

```
      LogMsg(stdout, "Last Contact      : %d:%d:%0.9lf UT
%d", jam_c4, menit_c4, detik_c4, counter_c4+1);

      dur_time = waktu_c4 - waktu_c1;
      if(dur_time<0){
        dur_time += 24.00;
      }
      jam_dur = (int)dur_time;
      menit_dur = (int)((dur_time - jam_dur)*60);
      detik_dur = (((dur_time - jam_dur)*60)-menit_dur)*60;
      LogMsg(stdout, "\n-------------------------------------\n");
      LogMsg(stdout, "Duration          : %d:%d:%0.9lf", jam_dur, menit_dur,
detik_dur);
      LogMsg(stdout, "\n-------------------------------------\n");

  /*LogMsg(stdout, "\nOK\n"); */
  LogClose(); /* Close the log file */
  return(0);
}
}

/*************************************************************
Name:    ParseCmdLine
Purpose: Routine to parse the command line.
Inputs:  argc - Number of command-line arguments.
         argv - Pointer to array of command-line arguments.
Outputs: InFile - Filename given by user for input file.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void ParseCmdLine(int argc, char *argv[], char *InFile) {

  /*
      Command line parser ported from Basic and optimized for C
      by Varian Swieter.
  */

  int i;

  for (i = 1; i < argc; i++) {
    /*
        Find next occurance of "/" or "-" by looking at
        the first character of each argument.
    */
    if (argv[i][0] != '-' && argv[i][0] != '/') {
      /* Found an argument that did not begin with "/" or "-" */
      LogMsg(stderr,
        "Command line arguments must begin with '-' or '/'.\n");
      PrintUsage();
      LogClose();
      remove(szLogFile);
      exit(1);
    }

    switch (argv[i][1])  {
      case '\0':
        LogMsg(stderr, "Space not allowed between - and option.\n");
        LogMsg(stderr, "Type SEPHEM -h for help.\n");
        LogClose();
        remove(szLogFile);
        exit(1);
      case 'I':
```

```
      case 'i':
        if (strlen(argv[i]) == 2) {
          /* We were given nothing after the "-I" so return empty string */
          strcpy(InFile, "");
        } else {
          if (argv[i][2] != ':') {
            /* User missed out colon so return empty string */
            strcpy(InFile, "");
          } else {
            strcpy(InFile, &(argv[i][3]));
          }
        }
        break;
      case '?': /* Using this for help will cause problems under UNIX */
                /* because it is used by the shell for substitution.  */
      case 'h':
        PrintUsage();
        LogMsg(stdout, "OK\n");
        LogClose();
        remove(szLogFile);
        exit(0);
      case 'v': /* Undocumented command line option */
                /* to print version information.    */
        LogMsg(stdout, "OK\n");
        LogClose();
        remove(szLogFile);
        exit(0);
      default:
        LogMsg(stderr, "Option not recognized.\n");
        LogMsg(stderr, "Type SEPHEM -h for help.\n");
        LogClose();
        remove(szLogFile);
        exit(1);
    }
  }
}

/*****************************************************************
Name:    PrintBanner
Purpose: Prints a banner to stdout and log file.
Inputs:  None.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void PrintBanner() {

  LogMsg(stdout, "\n");
  LogMsg(stdout, "***Program %s  ", szVersion);
  LogMsg(stdout, "                                \n");
  LogMsg(stdout, "***Small ephemeris program and Solar Eclipse Prediction
(Total & Annular)   \n");
  LogMsg(stdout, "                                \n");
  LogMsg(stdout, "Written by M. basthoni (NIM : 1500028006)
Ephemeris Algorithm by Joe Heafner\n");
  LogMsg(stdout, "email : m.basthoni@gmail.com
email : heafnerj@interpath.com\n");
  LogMsg(stdout, "Magister Ilmu Falak UIN Walisongo Semarang\n\n");
}

/* End Of File - sephem.c ***************************************/
```

Lampiran 2.
Source code file*: support.h*

```
#ifndef _SUPPORT_
#define _SUPPORT_

/*************************************************************
Project:  fecsoftc
Filename: support.h
Author:   Joe Heafner
Purpose:  General support routines header file.
Thanks to Charles Gamble for extensive modifications.
*************************************************************/

#include <sys/types.h>
#include <netinet/in.h>
#include <stdarg.h>
#include <sys/param.h>
/* #include <386/endian.h> */

#define MAX_NAME_SIZE       255
#define MAX_EXTENSION_SIZE  5
#define BIG_ENDIAN_TEST     (htonl(1) == 1)

#ifndef TRUE
 #define TRUE   (1)
 #define FALSE  (0)
#endif

/* Function prototypes */
void make_little_endian(char *ptr, int len);
void convert_little_endian(char *ptr, int len);
void reverse_bytes(char *ptr, int len);
void ucase(char str[]);
void left(char str[], int n, char dest[]);
void right(char str[], int n, char dest[]);
void Trim(char str[]);
void RightTrim(char str[]);
void LeftTrim(char str[]);
int  fexist(char *filename);
int  LogOpen(char *filename);
void LogClose(void);
void LogMsg(FILE *fptr, const char *format, ...);

#endif /* _SUPPORT_ */
```

Lampiran 3.
Source code file*: support.c*

```
/*****************************************************************
Project:        fecsoftc
Filename:       support.c
Author:         Joe Heafner
Purpose:        General support routines.
Thanks to Charles Gamble for extensive modifications.
*****************************************************************/

/* Header Files ***************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/param.h>
#include "support.h"

/* Globals ********************************************************/
static FILE *fpLogFile = NULL;

/*****************************************************************
Name:    make_little_endian
Purpose: Reverses the byte ordering of the given block of memory IF this
         machine is BIG ENDIAN. Used to ensure that binary data written
         out to file is little-endian in nature.
Inputs:  ptr - Pointer to block of memory to reverse.
         len - Length of block of memory to reverse.
Outputs: ptr - Byte-reversed block of memory.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void make_little_endian(char *ptr, int len) {

  if (BIG_ENDIAN_TEST) {
    reverse_bytes(ptr, len);
  }
}

/*****************************************************************
Name:    convert_little_endian
Purpose: Reverses the byte ordering of the given block of memory IF this
         machine is BIG ENDIAN. Used to convert the little-endian binary
         data read in from a file into the native endian format for this
         machine.
Inputs:  ptr - Pointer to block of memory to reverse.
         len - Length of block of memory to reverse.
Outputs: ptr - Byte-reversed block of memory.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void convert_little_endian(char *ptr, int len) {

  if (BIG_ENDIAN_TEST) {
    reverse_bytes(ptr, len);
```

```
  }
}

/*****************************************************************
Name:    reverse_bytes
Purpose: Reverses the byte ordering of the given block of memory.
Inputs:  ptr - Pointer to block of memory to reverse.
         len - Length of block of memory to reverse.
Outputs: ptr - Byte-reversed block of memory.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void reverse_bytes(char *ptr, int len) {

  int i;
  char *tmp;

  if ((tmp = malloc(len)) == NULL) {
    fprintf(stderr, "Memory allocation error (reverse_bytes:%d)\n", len);
    exit(1);
  }

  for (i = 0; i < len; i++) {
      tmp[i] = ptr[(len-1)-i];
  }

  memcpy(ptr, tmp, len);
  free(tmp);
}

/*****************************************************************
Name:    ucase
Purpose: Converts string to uppercase. This emulates the BASIC function
         ucase(a$).
Inputs:  str - String to convert.
Outputs: str - Converted string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void ucase(char str[]) {

  int i;
  int len = strlen(str);

  for (i = 0; i < len; i++)
    str[i] = toupper(str[i]);
}

/*****************************************************************
Name:    left
Purpose: Copies the first n characters of string str into dest.
Inputs:  str  - String to copy.
         n    - Number of characters to copy.
Outputs: dest - Copied string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void left(char str[], int n, char dest[]) {

  /* Safety test - make sure n is >= 0 */
```

```
  if (n > 0) {
    strncpy(dest, str, n);
    dest[n] = '\0'; /* Terminate just in case strlen(str) >= n */
  } else {
    strcpy(dest, "");
  }
}

/*****************************************************************
Name:    right
Purpose: Copies the last n characters of string str into dest.
Inputs:  str  - String to copy.
         n    - Number of characters to copy.
Outputs: dest - Copied string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void right(char str[], int n, char dest[]) {

  int len;

  if (n > 0) {
    len = strlen(str);
    if (n >= len) {
      /* Just copy whole string */
      strcpy(dest, str);
    } else {
      /* Copy section of string */
      strcpy(dest, &str[len-n]);
    }
  } else {
    strcpy(dest, "");
  }
}

/*****************************************************************
Name:    Trim
Purpose: Removes whitespace from either end of string.
Inputs:  str - String to trim.
Outputs: str - Trimmed string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void Trim(char str[]) {

  RightTrim(str); /* Remove trailing whitespace */
  LeftTrim(str);  /* Remove preceeding whitespace */
}

/*****************************************************************
Name:    RightTrim
Purpose: Removes trailing whitespace from string.
Inputs:  str - String to trim.
Outputs: str - Trimmed string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void RightTrim(char str[]) {

  int i;
```

```
  i = strlen(str) - 1;
  while (i >= 0 && isspace(str[i])) {
    str[i] = '\0';
    i--;
  }
}

/*****************************************************************
Name:    LeftTrim
Purpose: Removes preceeding whitespace from string.
Inputs:  str - String to trim.
Outputs: str - Trimmed string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void LeftTrim(char str[]) {

  unsigned int i;

  i = 0;
  while (i < strlen(str) && isspace(str[i]))
    i++;

  if (i)
   strcpy(str, &str[i]);
}

/*****************************************************************
Name:    fexist
Purpose: Tests to see if a file exists.
Inputs:  filename - Filename to test.
Outputs: None.
Returns: 0 if the file does not exist.
         1 if the file exists.
Status:  Finished.
Errors:  None known.
*****************************************************************/
int fexist(char *filename) {

  FILE *fp;

  if ((fp = fopen(filename,"r")) == NULL) {
    /* File could not be opened */
    return(0);
  } else {
    /* File was opened successfully */
    fclose(fp);
    return(1);
  }
}

/*****************************************************************
Name:    LogOpen
Purpose: Open a log file with the specified name.
Inputs:  filename - Name of log file to open.
Outputs: None.
Returns: TRUE if successful, FALSE otherwise.
Status:  Finished.
Errors:  None known.
*****************************************************************/
int LogOpen(char *filename) {
```

```
  /* Check to see if file is already open */
  if (fpLogFile)
    return (TRUE);

  /* Open the log file */
  if ((fpLogFile = fopen(filename, "wt")) == NULL)
    return (FALSE);
  else
    return (TRUE);
}

/*****************************************************************
Name:    LogClose
Purpose: Closes the log file.
Inputs:  None.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void LogClose() {

  /* Check to see if file is open */
  if (fpLogFile) {
    fclose(fpLogFile);
    fpLogFile = NULL;
  }
}

/*****************************************************************
Name:    LogMsg
Purpose: Prints a message to a given file pointer and to the log file.
         Log file must have been opened previously with LogOpen().
Inputs:  fptr     - File pointer to print with.
         format   - Format string (like printf).
         va_alist - Variable length argument list.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void LogMsg(FILE* fptr, const char *format, ...) {

  char buffer[1024];
  va_list argptr;

  va_start(argptr, format);
  vsprintf(buffer, format, argptr);
  va_end(argptr);

  /* Write to the log file if it is open */
  if (fpLogFile)
    fprintf(fpLogFile, "%s", buffer);

  /* Write to the given file pointer if it is valid */
  if (fptr)
    fprintf(fptr, "%s", buffer);
}

/* End Of File - support.c ***************************************/
```

Lampiran 4.
Source code file*: astrocon.h*

```
#ifndef _CONSTS_
#define _CONSTS_

/* Gaussian gravitational constant */
#define GAUSSK (0.01720209895)
/* values related to PI */
#ifndef PI
#define PI (139755218526789.0 / 44485467702853.0)
#endif

#define TWOPI  (2.0 * PI)
#define PIDIV2 (0.5 * PI)

/* degrees to radians and radians to degrees */
#define D2R (PI / 180.0)
#define R2D (180.0 / PI)
/* hours to radians and radians to hours */
#define H2R (PI / 12.0)
#define R2H (12.0 / PI)

/* arcseconds to radians and radians to arcseconds */
#define A2R (PI / 648000.0)
#define R2A (648000.0 / PI)
/* AU to km and km to AU */
#define AU2KM (149597870.66)
#define KM2AU (1.0 / 149597870.66)

/* speed of light in km/s and AU/day */
#define CKMS (299792.458)
#define CAUD (CKMS * 86400.0 * KM2AU)

/* Earth's angular velocity in rad/s */
#define EarAngVelRD (6.30038748685432)
/* Earth's equatorial radius in km and AU */
#define EarthRadKM (6378.137)
#define EarthRadAU (EarthRadKM * KM2AU)
/* Earth's flattening factor */
#define EFlat (1.0 / 298.257)

/* JED of standard epoch */
#define J2000 (2451545.0)

/* days in a Julian century */
#define JulCty (36525.0)

/* Earth's gravitational constant */
#define MUC (2.0 * GAUSSK * GAUSSK / (CAUD * CAUD))

/* global state vector variable */
extern double StateVector[15][15][2][6];

extern double obsr_lon;
extern double obsr_lat;
extern double obsr_ele;

#endif /* _CONSTS_ */
```

Lampiran 5.
Source code file*: astrolib.h*

```
#ifndef _ASTROLIB_
#define _ASTROLIB_

/*************************************************************
Project:  fecsoftc
Filename: astrolib.h
Author:   Joe Heafner
Purpose:  Library routines header file.
Thanks to Charles Gamble for extensive modifications.
*************************************************************/

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include "support.h"
#include "astrocon.h"

#ifndef TRUE
 #define TRUE  (1)
 #define FALSE (0)
#endif

#ifndef SEEK_SET
 #define SEEK_SET  (0)
 #define SEEK_CUR  (1)
 #define SEEK_END  (2)
#endif

/* double precision matrix data type */
typedef double** DMatrix;

/* function prototypes */
void Aberrate(double *p1, double *EBdot, double *p2);
double amodulo(double a, double b);
void Cal2JED(int m, int d, int y, double utc, int s, double tai_utc,
  double ut1_utc, int w, double *jed);
void constants(char *FileName);
void Conway(double uele[], double mu, double jed, double *r_cos_nu,
  double *r_sin_nu);
DMatrix createDMatrix(int n);
double deg(double x);
double dms(double x);
double DRound(double x, int n);
FILE *ephopn(char *FileName);
void Epoch2JED(char *epoch, double *jed);
void Eq2Ecl(double *a, int s, double eps, double *b);
void Eq2Hor(double *a, int s, double *b);
void errprt(int group, char *message);
double fix(double x);
void FmtDms(double x, int n, int m, char *s);
void free_matrix(double **m, int nr, int nc);
void freeDMatrix(DMatrix m, int n);
void FunArgIAU(double jed, double *funarg);
void GeocenObs(double jed, double *obsr_geo);
void GetGST(double jed, int s, double *gst);
void GetInvQMatrix(DMatrix QMatrix, DMatrix InvQMatrix);
int  GetNumde(void);
```

```
void GetPrecessParams(double jed1, double jed2, double *zeta,
  double *z, double *theta, double *zetadot,
  double *zdot, double *thetadot);
void GetQMatrix(double phi, double phidot, int axis, int s,
  DMatrix QMatrix);
void GetRPNmat(double jed1, double jed2, int rpn, int d,
  DMatrix m3, DMatrix m6);
void GetStateVector(double JD, int TARG, int CENT, int recpol,
  double StateVector[15][15][2][6]);
void HelEphemeris(double *uelement, double mu, double jed, double *posvel);
void interp(int buff, double *t, int ncf, int ncm, int na, int fl,
  double pv[3][2]);
void Interp1(double *x, double *y, int L, int m, double *arg, double *v);
double Interpol(double *x, double *y, int i, double arg);
void JED2Cal(double jed, int *yr, int *mo, int *dy, double *ti);
void JED2Epoch(double jed, char *s, char *epoch);
double **matrix(int nr, int nc);
void MatXMat(DMatrix a, DMatrix b, DMatrix c, int n);
void MatXVec(DMatrix a, double *b, double *c, int l, int m);
void MRotate(double *vin, int axis, double phi, double *vout);
void GetDpsiDeps(double jed, double *dpsi, double *deps,
  double *dpsidot, double *depsdot);
void Obliquity(double jed1, double jed2, int m, double *obl,
  double *obldot);
void pleph(double jd, int targ, int cent, double *rrd, int *inside);
void Pol2Rec(double *a, double *b);
void PrecessElements(char *eqnx1, double *element, char *eqnx2);
void QRotate(double *vin, int axis, double phi, double phidot,
  int s, double *vout);
void RayBend(double *earth_hel, double *body_geo, double *body_hel,
  double *p1);
void Rec2Pol(double *a, double *b);
void Reduce (double jed, int body, int place, double StarData[],
  double p3[]);
void Refract(double ra1, double dec1, double lha, double temp,
  double press, double *ra2, double *dec2);
void RotMat(int axis, double phi, DMatrix r, DMatrix dr);
void RST(double jed, double *ra, double *dec, double z0, double deltat,
  char *ris, char *trn, char *set);
void split(double tt, double *ipart, double *fpart);
void SplitStateVector(double *pv, double *p, double *v);
void state(double *jed, int LList[], double pv[6][13], double *nut);
void Stat2Elem(double *posvel, double mu, double jed, double *uelement);
double StumpffN(double x, int Norder);
void Transpose(DMatrix a, DMatrix b, int n);
void Uvector(double *a, double *unita);
void Vcross(double *a, double *b, double *acrossb);
void Vdot(int n, double *a, double *b, double *adotb);
double Vecmag(double *a);

#endif /* _ASTROLIB_ */
```

Lampiran 6.
Source code file*: astrolib.c*

```
/*************************************************************
Project:  fecsoftc
Filename: astrolib.c
Author:   Joe Heafner
Purpose:  Astronomical ephemeris library.
Thanks to Charles Gamble for extensive modifications.
*************************************************************/

/* Header Files ************************************************/
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include "astrolib.h"
#include "astrocon.h"
#include "support.h"

#ifndef MAX_FILENAME
  #define MAX_FILENAME  255
#endif

/* Globals *****************************************************/
/* Ephemeris header information */
char ttl[3][65], cnam[400][7];
int  ncon;
double SS[3], cval[400], au, emrat;
short int NUMDE, ipt[3][12], lpt[3];

/* Other data file specs not in header */
long LengthOfFile, ncoeff, BlockLength;
int LengthOfHeader, NumBlocks, bary, bsav;
double db[1100], pvsun[6];

/* ======== modifikasi oleh : M. Basthoni ======================
Senin, 20 Maret 2017, Jam 23:00 WIB
         Keterangan :
         variable : obsr_lon, obsr_lat, obsr_ele,
                    StateVector[15][15][2][6] belum dideklarasikan
============================================================== */
double obsr_lon, obsr_lat, obsr_ele, StateVector[15][15][2][6];

static FILE *fpBinaryFile = NULL;

/* Temporary data used when reading in binary file */
short tmpShort;
double tmpDouble;
int tmpInt;
long tmpLong;

/*************************************************************
Name:    Aberrate
Purpose: Function to correct an input vector for aberration.
Inputs:  p1    - Zero-offset geocentric state vector of object.
         EBdot - Zero-offset barycentric velocity of Earth.
Outputs: p2 - Zero-offset geocentric state vector of object corrected
              for aberration.
Returns: Nothing.
```

```
Status:  Finished.
Errors:  None known.
************************************************************/
void Aberrate(double p1[], double EBdot[], double p2[]) {

  double v[3], up[3], p[3], pdot[3], magP, magV, p1dotv, beta;
  int i;

  /* Extract the pos. portion of the state vector */
  SplitStateVector(p1, p, pdot);
  magP = Vecmag(p);

  /* Need to make Ppos() a unit vector */
  Uvector(p, up);

  for (i=0; i<3; i++) {
    v[i] = EBdot[i] / CAUD;
  }

  Vdot(3, up, v, &p1dotv);
  magV = Vecmag(v);

  beta = 1.0 / sqrt(1.0 - magV * magV);

  for (i=0; i<3; i++) {
    p2[i] = ((up[i] / beta) +
      (1.0 + p1dotv / (1.0 + (1.0 / beta))) * v[i]) / (1.0 + p1dotv);
    /* Make p2[] a non-unit vector */
    p2[i] = magP * p2[i];
    p2[i+3] = p1[i+3];
  }
  return;
}

/*************************************************************
Name:    amodulo
Purpose: Function to reduce a to the range 0 <= a < b.
Inputs:  a - See above.
         b - See above.
Outputs: None.
Returns: See above.
Status:  Finished.
Errors:  None known.
*************************************************************/
double amodulo(double a, double b) {

  double x;

  x = a - b * floor(a/b);
  return (x);
}

/*************************************************************
Name:    Cal2JED
Purpose: Function to compute the Julian date on the specified time scale.
         The Julian day algorithm of Meeus is used, and the algorithm for
         converting to TDB is due to Hirayama et al. and can be found in
         REFERENCE FRAMES IN ASTRONOMY AND GEOPHYSICS ed. by Kovalevsky
         et al., 1989, pp. 439-442.
Inputs:  m       - Month.
         d       - Day.
         y       - Year.
         utc     - UTC as hh.mmssss.
```

```
         s       - 1 for UT1, 2 for UT2, 3 for TDT,
                   4 for TDB, 5 for UTC.
         tai_utc - TAI-UTC in seconds.
         utl_utc - UT1-UTC in seconds.
         w       - 1 for MJD, 0 otherwise.
Outputs: jed - Appropriate JED.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void Cal2JED(int m, int d, int y, double utc, int s,
  double tai_utc, double utl_utc, int w, double *jed) {
  double time, datnum, a, b, term1, corr, T;

  /* Convert to decimal hours */
  time = deg(utc);

  if ((m == 1) || (m == 2)) {
    y = y - 1;
    m = m + 12;
  }

  /*
      Test to see if date is in Gregorian calendar.
      Converts date to a number of the form YYYY.MMDD.
  */
  datnum = (double) y + 0.01 * (double) m + 0.0001 * (double) d;

  if (datnum >= 1582.1015) {
    /* Gregorian calendar */
    a = fix(0.01 * (double) y);
    b = 2.0 - a + fix(0.25 * (double) a);
  } else {
    /* Julian calendar */
    a = 0.0;
    b = 0.0;
  }

  if (y < 0) {
    /* Handles negative years */
    term1 = fix(365.25 * (double) y - 0.75); /* change here */
  } else {
    term1 = fix(365.25 * (double) y);
  }

  *jed = term1 + fix(30.6001 * ((double) m + 1.0)) +
    (double) d + time / 24.0 + 1720994.5 + (double) b;

  switch (s) {
    case (1):
      corr = utl_utc / 86400.0;
      break;
    case (2):
      corr = utl_utc / 86400.0;
      *jed = *jed + corr;
      /* Compute date in Besselian years */
      T = 2000.0 + (*jed - 2451544.5333981) / 365.242198781;
      corr = 0.022 * sin(TWOPI * T);
      corr += -0.012 * cos(TWOPI * T);
      corr += -0.006 * sin(2.0 * TWOPI * T);
      corr += 0.007 * cos(2.0 * TWOPI * T);
      corr = corr / 86400.0;
      break;
```

```
    case (3):
      corr = (tai_utc + 32.184) / 86400.0;
      break;
    case (4):
      /* First convert to TDT */
      corr = (tai_utc + 32.184) / 86400.0;
      *jed = *jed + corr;
      T = (*jed - J2000) / JulCty;
      /* Now compute the new correction in microseconds */
      corr = 1656.675     * sin((35999.3729 * T + 357.5387) * D2R);
      corr +=  22.418     * sin((32964.467  * T + 246.199)  * D2R);
      corr +=  13.84      * sin((71998.746  * T + 355.057)  * D2R);
      corr +=   4.77      * sin(( 3034.906  * T +  25.463)  * D2R);
      corr +=   4.67      * sin((34777.259  * T + 230.394)  * D2R);
      corr +=  10.216 * T * sin((35999.373  * T + 243.451)  * D2R);
      corr +=   0.171 * T * sin((71998.746  * T + 240.98 )  * D2R);
      corr +=   0.027 * T * sin(( 1222.114  * T + 194.661)  * D2R);
      corr +=   0.027 * T * sin(( 3034.906  * T + 336.061)  * D2R);
      corr +=   0.026 * T * sin((  -20.186  * T +   9.382)  * D2R);
      corr +=   0.007 * T * sin((29929.562  * T + 264.911)  * D2R);
      corr +=   0.006 * T * sin((  150.678  * T +  59.775)  * D2R);
      corr +=   0.005 * T * sin(( 9037.513  * T + 256.025)  * D2R);
      corr +=   0.043 * T * sin((35999.373  * T + 151.121)  * D2R);
      /* Convert from microseconds to seconds */
      corr = corr * 0.000001;
      /* Convert to days */
      corr = corr / 86400.0;
      break;
    case (5):
      corr = 0.0;
      break;
    default:
      corr = 0.0;
  }

  *jed += corr;

  /* Return modified JED if requested */
  if (w == 1) *jed -= 2400000.5;
}

/*****************************************************************
Name:    constants
Purpose: Function to display the constants from the ephemeris file.
         Also prints other useful information about the file.
Inputs:  FileName - Filename of file to be read in.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void constants(char *FileName) {

  int i;
  char Y[1024] = "";

  /* Open the file and read the header data */
  if (ephopn(FileName) == NULL) {
    LogMsg(stderr, "An error occurred in ephopn().\n");
    LogClose();
    exit(1);
  }
```

```
for (i=0; i<3; i++)
  LogMsg(stdout, "%s\n", ttl[i]);

LogMsg(stdout, "First valid JED in file  %9.1lf\n", SS[0]);
LogMsg(stdout, "Final valid JED in file  %9.1lf\n", SS[1]);
LogMsg(stdout, "Block interval  %2.0lf days\n",     SS[2]);
LogMsg(stdout,
  "Length of file                % 8.0ld bytes\n", LengthOfFile);
LogMsg(stdout,
  "Length of header              % 8.0d bytes\n" , LengthOfHeader);
LogMsg(stdout,
  "Length of each block          % 8.0ld bytes\n", BlockLength);
LogMsg(stdout,
  "Coeffs per block              % 8.0ld\n"      , ncoeff);
LogMsg(stdout,
  "Blocks in file                % 8.0d\n"       , NumBlocks);

LogMsg(stdout, "Bodies present in data file:  ");
if (ipt[0][0] != 0) LogMsg(stdout, "MER ");
if (ipt[0][1] != 0) LogMsg(stdout, "VEN ");
if (ipt[0][2] != 0) LogMsg(stdout, "EMB ");
if (ipt[0][3] != 0) LogMsg(stdout, "MAR ");
if (ipt[0][4] != 0) LogMsg(stdout, "JUP ");
if (ipt[0][5] != 0) LogMsg(stdout, "SAT ");
if (ipt[0][6] != 0) LogMsg(stdout, "URA ");
if (ipt[0][7] != 0) LogMsg(stdout, "NEP ");
if (ipt[0][8] != 0) LogMsg(stdout, "PLU ");
if (ipt[0][9] != 0) LogMsg(stdout, "MOO ");
if (ipt[0][10] != 0) LogMsg(stdout, "SUN\n");

if (ipt[0][11] == 0 || ipt[1][11] == 0 || ipt[2][11] == 0)
  LogMsg(stdout, "This ephemeris does not contain nutations.\n");
else
  LogMsg(stdout, "This ephemeris contains nutations.\n");

if (lpt[0] == 0 || lpt[1] == 0 || lpt[2] == 0)
  LogMsg(stdout, "This ephemeris does not contain librations.\n");
else
  LogMsg(stdout, "This ephemeris contains librations.\n");

LogMsg(stdout, "This ephemeris has %d ephemeris constants.\n", ncon);

do {
  fprintf(stdout, "Display constant names and values (Y/N)? ");
  fflush(stdout);
  fgets(Y, sizeof(Y), stdin);
  Trim(Y);
  ucase(Y);
} while ((strcmp(Y, "Y") != 0) && (strcmp(Y, "N") != 0));

if (strcmp(Y, "Y") == 0) {
  LogMsg(stdout, "\n");
  for (i=0; i<ncon; i++) {
    LogMsg(stdout, "%6s  %+16.15E      ", cnam[i], cval[i]);
    if ((i+1) % 2 == 0) {
      LogMsg(stdout, "\n");
    }
  }

  if ((i % 2) != 0) {
    LogMsg(stdout, "\n");
  }
}
```

```
}

/*****************************************************************
Name:    Conway
Purpose: Function to solve Kepler's equation using the method of
         Conway-Laguerre for universal variables.
Inputs:  uele - Array of universal orbital elements.
         mu   - Gravitational constant for the body.
         jed  - Time for which computations are to be performed.
Outputs: r_cos_nu - r*cos(true anomaly).
         r_sin_nu - r*sin(true anomaly).
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void Conway(double uele[], double mu, double jed, double *r_cos_nu,
  double *r_sin_nu) {

  double tspp, alpha, s, x, c1, c2, c3, f, fp, fpp, term, ds;
  int niter;

  /* Compute time since perihelion passage in days */
  tspp = jed - uele[5];

  /* Compute alpha */
  alpha = mu * (1.0 - uele[1]) / uele[0];

  /* Initial guess for s */
  s = 0.0;

  /* Initialize iteration counter */
  niter = 0;

  do {
    /* Compute Stumpff functions */
    x = alpha * s * s;
    c1 = StumpffN(x, 1);
    c2 = StumpffN(x, 2);
    c3 = StumpffN(x, 3);

    f = uele[0] * s + mu * uele[1] * s * s * s * c3 - tspp;
    fp = uele[0] + mu * uele[1] * s * s * c2;
    fpp = mu * uele[1] * s * c1;

    /* Evaluate Laguerre's method */
    term = 16.0 * fp * fp - 20.0 * f * fpp;
    ds = -5.0 * f / (fp + fp/fabs(fp) * sqrt(fabs(term)));

    s = s + ds;

    niter = niter + 1;

    /* Check for convergence or more than ten iterations */
    if (niter > 10)
    {
      LogMsg(stderr, "Conway: more than ten iterations.\n");
      exit(1);
    }

    if (fabs(f) < 1e-12) break;

  } while(1);
```

```
  /* Now compute r_sin_nu and r_cos_nu */
  *r_sin_nu = sqrt(mu * uele[0] * (1.0 + uele[1])) * s * c1;
  *r_cos_nu = uele[0] - mu * s * s * c2;
}

/*************************************************************
Name:    createDMatrix
Purpose: Creates an n x n double matrix.
Inputs:  n - Dimension of matrix to create.
Outputs: None.
Returns: DMatrix.
Status:  Finished.
Errors:  None known.
*************************************************************/
DMatrix createDMatrix(int n) {

  DMatrix m;
  int i;

  m = calloc(n, sizeof(double*));
  if (m == NULL) {
    return NULL;
  }

  for (i=0; i<n; i++) {
    m[i] = calloc(n, sizeof(double));
    if (m[i] == NULL) {
      freeDMatrix(m, i); /* Avoids garbage */
      return NULL;
    }
  }

  return m;
}

/*************************************************************
Name:    deg
Purpose: Converts dd.mmssss to dd.dddddd.
         Converts hh.mmssss to hh.hhhhhh.
Inputs:  x - Value to convert.
Outputs: None.
Returns: Converted value.
Status:  Finished.
Errors:  None known.
*************************************************************/
double deg(double x) {

  double dd, d, fixdd, ddfixdd;

  if (x == 0.0) return (0.0);

  dd = fabs(x);
  fixdd = floor(dd);
  ddfixdd = dd - fixdd + 5.0e-10;  /* fudge factor */
  d = fixdd + floor(100.0 * ddfixdd) / 60.0;
  d = d + (10000.0 * ddfixdd - 100.0 * floor(100.0 * ddfixdd)) / 3600.0;

  return ((x / dd) * d);
}

/*************************************************************
Name:    dms
Purpose: Converts dd.dddddd to dd.mmssss.
```

```
         Converts hh.hhhhhh to hh.mmssss.
Inputs:  x - Value to convert.
Outputs: None.
Returns: Converted value.
Status:  Finished.
Errors:  None known.
*****************************************************************/
double dms(double x) {

  double dd, fdd, dmfd, d;

  if (x == 0.0) return (0.0);

  dd = fabs(x);
  fdd = floor(dd);
  dmfd = dd - fdd + 5.0e-10;  /* fudge factor */
  d = fdd + floor(60.0 * dmfd) / 100.0 + 0.006 *
    (60.0 * dmfd - floor(60.0 * dmfd));

  return ((x / dd) * d);
}

/*************************************************************
Name:    DRound
Purpose: Function to round a number to a given number of decimal places.
Inputs:  x - Value to round.
         n - Number of decimal places.
Outputs: None.
Returns: Rounded number.
Status:  Finished.
Errors:  None known.
*****************************************************************/
double DRound(double x, int n) {

  double a, y;
  int sgn;

  if (x > 0) {
    sgn = 1;
  }
  else if (x < 0) {
    sgn = -1;
  } else {
    sgn = 0;
  }

  a = pow(10.0, (double) n);
  y = floor((a * x) + ((double) sgn) * 0.5) / a;

  return y;
}

/*************************************************************
Name:    ephopn
Purpose: Function to open a binary ephemeris data file for access,
         read the header information, and store it in memory for
         later use.
Inputs:  FileName - Filename of file to be read in.
Outputs: None.
Returns: NULL on error.
         Open file pointer otherwise.
Status:  Finished.
Errors:  None known.
```

```
*****************************************************************/
FILE *ephopn(char *FileName) {

  /* Make sure all variables read from the data file are global */
  int i, j;
  static int efirst;
  long curpos;

  if (!efirst) {
    /* Default file name is JPLEPH */
    if (strlen(FileName) == 0) {
      strcpy(FileName, "JPLEPH");
    }

    /*
        The existence of the data file MUST be confirmed
        in the calling program. It is NOT checked here!
    */
    if ((fpBinaryFile = fopen(FileName, "rb")) == NULL) {
      LogMsg(stderr,
        "EPHOPN: Can't open binary data file: %s\n", strerror(errno));
      return (NULL);
    }

    /* BEGINNING OF HEADER */
    /* 195 bytes */
    for (i = 0; i < 3; i++) {
      /* char TTL[3][65] */
      if (fread(ttl[i], sizeof(char), 65, fpBinaryFile) != 65) {
        LogMsg(stderr,
          "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
        return (NULL);
      }
      ttl[i][64] = '\0';
    }

    /* 2 bytes */
    if (fread(&tmpShort, sizeof(short), 1, fpBinaryFile) != 1) {
      LogMsg(stderr,
        "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
      return (NULL);
    }

    convert_little_endian((char *) &tmpShort, sizeof(short));
    ncon = (int) tmpShort;

    /* ncon*6 bytes */
    for (j = 0; j < ncon; ++j) {
      /* char CNAM[NCON][7] */
      if (fread(&cnam[j], sizeof(char), 6, fpBinaryFile) != 6) {
        LogMsg(stderr,
          "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
        return (NULL);
      }
      cnam[j][6] = '\0';
    }

    /* 24 bytes, 8 each */
    for (j = 0; j < 3; j++) {
      /* double SS[3] */
      if (fread(&tmpDouble, sizeof(double), 1, fpBinaryFile) != 1) {
        LogMsg(stderr,
          "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
```

```
        return (NULL);
      }
      convert_little_endian((char *) &tmpDouble, sizeof(double));
      SS[j] = (double) tmpDouble;
    }

    /* 16 bytes, 8 each */
    if (fread(&tmpDouble,    sizeof(double), 1, fpBinaryFile) != 1) {
      LogMsg(stderr,
        "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
      return (NULL);
    }
    convert_little_endian((char *) &tmpDouble, sizeof(double));
    au = (double) tmpDouble;

    if (fread(&tmpDouble, sizeof(double), 1, fpBinaryFile) != 1) {
      LogMsg(stderr,
        "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
      return (NULL);
    }
    convert_little_endian((char *) &tmpDouble, sizeof(double));
    emrat = (double) tmpDouble;

    /* 72 bytes */
    for (i = 0; i < 3; i++) {
      /* short IPT[3][12] */
      for (j = 0; j < 12; j++) {
        if (fread(&tmpShort, sizeof(short), 1, fpBinaryFile) != 1) {
          LogMsg(stderr,
            "EPHOPN: Error reading binary data file: %s\n",
strerror(errno));
          return (NULL);
        }
        convert_little_endian((char *) &tmpShort, sizeof(short));
        ipt[i][j] = (short) tmpShort;
      }
    }

    /* 2 bytes */
    if (fread(&tmpShort, sizeof(short), 1, fpBinaryFile) != 1) {
      LogMsg(stderr,
        "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
      return (NULL);
    }
    convert_little_endian((char *) &tmpShort, sizeof(short));
    NUMDE = (short) tmpShort;

    /* 6 bytes, 2 each */
    for (i = 0; i < 3; i++) {
      if (fread(&tmpShort, sizeof(short), 1, fpBinaryFile) != 1) {
        LogMsg(stderr,
          "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
        return (NULL);
      }
      convert_little_endian((char *) &tmpShort, sizeof(short));
      lpt[i] = (short) tmpShort;
    }

    /* ncon*8 bytes */
    for (j = 0; j < ncon; j++) {
      if (fread(&tmpDouble, sizeof(double), 1, fpBinaryFile) != 1) {
        LogMsg(stderr,
          "EPHOPN: Error reading binary data file: %s\n", strerror(errno));
```

```
        return (NULL);
      }
      convert_little_endian((char *) &tmpDouble, sizeof(double));
      cval[j] = (double) tmpDouble;
    }
    /* END OF HEADER */

    /* This block used to be a separate function */
    /* but it didn't work correctly.             */
    if ((curpos = ftell(fpBinaryFile)) == -1L) {
      LogMsg(stderr, "EPHOPN: ftell() returned -1L.\n");
      return (NULL);
    }

    if (fseek(fpBinaryFile, 0L, SEEK_END) != 0) {
      LogMsg(stderr, "EPHOPN: fseek() failed.\n");
      return (NULL);
    }

    LengthOfFile = ftell(fpBinaryFile);

    if (fseek(fpBinaryFile, curpos, SEEK_SET) != 0) {
      LogMsg(stderr, "EPHOPN: fseek() failed.\n");
      return (NULL);
    }
    /* end block */

    LengthOfHeader = 317L + 14L * (long) ncon;

    /* Length of block of coeffs = 8*NCOEFF bytes */
    /* calculate number of coeffs per block */
    ncoeff = 0L;

    for (j = 0; j < 12; j++) {
      if (j == 11)
        ncoeff = ncoeff + (long) ipt[1][j] * (long) ipt[2][j] * 2L;
      else
        ncoeff = ncoeff + (long) ipt[1][j] * (long) ipt[2][j] * 3L;
    }

    ncoeff = ncoeff + (long) lpt[1] * (long) lpt[2] * 3L + 2L;
    BlockLength = ncoeff * 8L;
    NumBlocks = (LengthOfFile - (long) LengthOfHeader) / BlockLength;
    efirst = TRUE;
  }

  return (fpBinaryFile);
}

/*****************************************************************
Name:    Epoch2JED
Purpose: Function to convert an epoch to its corresponding JED.
Inputs:  epoch - J or B epoch (e.g. J2000).
Outputs: jed - Appropriate JED.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void Epoch2JED(char *epoch, double *jed) {

  double date;
  char buffer[80];
```

```
  strcpy(buffer, epoch);

  ucase(buffer);

  sscanf(buffer, "%*c %lf", &date);

  if (epoch[0] == 'J') {
    /* Julian epoch */
    *jed = (date - 2000.0) * 365.25 + 2451545.0;
  } else {
    /* Besselian epoch */
    *jed = (date - 1900.0) * 365.242198781731 + 2415020.31352;
  }

  return;
}

/*****************************************************************
Name:    Eq2Ecl
Purpose: Subprogram to convert an equatorial state vector to an
         ecliptic state vector or the reverse transformation.
Inputs:  a   - Zero-offset input vector.
         s   - 0 for eq  -> ecl, 1 for ecl -> eq.
         eps - Mean or apparent obliquity.
Outputs: b - Zero-offset output vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void Eq2Ecl(double a[], int s, double eps, double b[]) {

  DMatrix r, dr;

  r  = createDMatrix(3);
  dr = createDMatrix(3);

  if (s == 0) {
    /* Equatorial to ecliptic */
    RotMat(1,  eps, r, dr);
  } else {
    /* Ecliptic to equatorial */
    RotMat(1, -eps, r, dr);
  }

  MatXVec(r, a, b, 3, 3);

  freeDMatrix(r, 3);
  freeDMatrix(dr,3);

  return;
}

/*****************************************************************
Name:    Eq2Hor
Purpose: Function to convert an equatorial state vector to a
         horizon state vector or the reverse transformation.
Inputs:  a - Zero-offset input vector.
         s - 0 for eq -> hor, 1 for hor -> eq.
Outputs: b - Zero-offset output vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
```

```
void Eq2Hor(double a[], int s, double b[]) {

  DMatrix r2, r3, r, dr2, dr3, dr;

  r2 = createDMatrix(3);
  r3 = createDMatrix(3);
  r  = createDMatrix(3);
  dr2 = createDMatrix(3);
  dr3 = createDMatrix(3);
  dr = createDMatrix(3);

  if (s == 0) {
    /* Equatorial to horizon */
    RotMat(2, PIDIV2 - obsr_lat, r2, dr2);
    RotMat(3, -PI, r3, dr3);
    MatXMat(r3, r2, r, 3);
  } else {
    /* Horizon to equatorial */
    RotMat(3, PI, r3, dr3);
    RotMat(2, obsr_lat - PIDIV2, r2, dr2);
    MatXMat(r2, r3, r, 3);
  }

  MatXVec(r, a, b, 3, 3);

  freeDMatrix(r2, 3);
  freeDMatrix(r3, 3);
  freeDMatrix(r, 3);
  freeDMatrix(dr2, 3);
  freeDMatrix(dr3, 3);
  freeDMatrix(dr, 3);

  return;
}

/*****************************************************************
Name:    errprt
Purpose: Function to print error messages.
Inputs:  group   - Error code number.
         message - Error message.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void errprt(int group, char *message) {

  LogMsg(stderr, "\nERROR #%d %s\n", group, message);
  exit(1);
}

/*****************************************************************
Name:    fix
Purpose: Function to return the integer part of a number. Basically,
         this function emulates the PowerBasic fix() function.
Inputs:  x - Decimal value.
Outputs: None.
Returns: Integral part of a number.
Status:  Finished.
Errors:  None known.
*****************************************************************/
double fix(double x) {
```

```
  if (x < 0) {
    return ceil(x);
  } else {
    return floor(x);
  }
}

/*************************************************************
Name:    FmtDms
Purpose: Function to format angular and time quantities. Basically,
         this function is a pretty-print version of dms().
         Modified with permission from Rex Shudde's code by Joe Heafner.
Inputs:  x - Decimal value.
         n - Number of seconds decimal digits.
         m - 0 convert decimal degrees to:
               ddd mm ss      if n = 0
               ddd mm ss.f    if n = 1
               ddd mm ss.ff   if n = 2
               ddd mm ss.fff  if n = 3
               ddd mm ss.ffff if n = 4
         m - 1 converts decimal hours to:
               hh mm ss       if n = 0
               hh mm ss.f     if n = 1
               hh mm ss.ff    if n = 2
               hh mm ss.fff   if n = 3
               hh mm ss.ffff  if n = 4
Outputs: s - Destination for output string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void FmtDms(double x, int n, int m, char *s) {

  double absx, deg, min, sec;
  int nf;
  unsigned int fig;
  static int defd;
  static char dh[3];
  static char mm[3];
  static char ss[3];
  char buffer[16] = "              ";
  char right_buffer[16], left_buffer[16];

  strcpy(s, "              ");

  if (!defd) {
    defd++;
    dh[0] = '\xf8'; /* f8 is the hex code for degree symbol */
    dh[1] = 'h';
    dh[2] = '\0';
    mm[0] = '\x27'; /* 27 is the hex code for arcmin symbol */
    mm[1] = 'm';
    mm[2] = '\0';
    ss[0] = '\x22'; /* 22 is the hex code for arcsec symbol */
    ss[1] = 's';
    ss[2] = '\0';
  }

  absx = fabs(x);

  /* determine how many digits before the decimal */
  if (absx < 100) {
    fig = 2;
```

```
} else {
  fig = 3;
}

deg = floor(absx);
absx = 60.0 * (absx - deg);
min = floor(absx);
sec = 60.0 * (absx - min);
sec = DRound(sec, n);

if (sec >= 60.0) {
  sec = 0.0;
  min = min + 1.0;
}

if (min >= 60.0) {
  min = 0.0;
  deg = deg + 1.0;
}

if (((deg == 24.0) && m == 1) || (deg == 360.0)) {
  deg = 0.0;
}

strcpy(s, " ");
if (x < 0) {
  strcpy(s, "-");
}
else if (m == 0) {
  strcpy(s, "+");
}

/* begin building up the return string in buffer */
sprintf(buffer, "%.0f", (1000.0+deg));
right(buffer, fig, right_buffer);
strcat(s, right_buffer);
left(dh, m+1, left_buffer);
right(left_buffer, 1, right_buffer);
strcat(s, right_buffer);

sprintf(buffer, "%.0f", (100.0+min));
right(buffer, 2, right_buffer);
strcat(s, right_buffer);
left(mm, m+1, left_buffer);
right(left_buffer, 1, right_buffer);
strcat(s, right_buffer);

switch (n) {
  case 0:
    sprintf(buffer, "%.0f", (100.0+sec+0.0000001));
    break;
  case 1:
    sprintf(buffer, "%.1f", (100.0+sec+0.0000001));
    break;
  case 2:
    sprintf(buffer, "%.2f", (100.0+sec+0.0000001));
    break;
  case 3:
    sprintf(buffer, "%.3f", (100.0+sec+0.0000001));
    break;
  case 4:
    sprintf(buffer, "%.4f", (100.0+sec+0.0000001));
}
```

```
  if (n == 0) {
    nf = 2;
  } else {
    nf = n + 3;
  }

  right(buffer, strlen(buffer)-1, right_buffer);
  left(right_buffer, nf, left_buffer);
  strcat(s, left_buffer);

  left(ss, m+1, left_buffer);
  right(left_buffer, 1, right_buffer);
  strcat(s, right_buffer);
}

/*****************************************************************
Name:    free_matrix
Purpose: Free a double matrix allocated by matrix().
Inputs:  m    - Pointer to matrix.
         nrow - Number of rows.
         ncol - Number of columns.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void free_matrix(double **m, int nrow, int ncol) {

  int i;

  for (i = 0;i < nrow;i++) {
    free(m[i]);
  }

  free(m);
}

/*****************************************************************
Name:    freeDMatrix
Purpose: Free all storage associated with matrix m.
Inputs:  m - Matrix to free.
         n - Dimension of matrix.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void freeDMatrix(DMatrix m, int n)
{
  int i;

  for (i=0; i<n; i++)
  {
    free(m[i]); /* storage for doubles in row i */
  }
  free(m); /* storage for pointers to rows */

  return;
}

/*****************************************************************
Name:    FunArgIAU
```

```
Purpose: Function to compute the fundamental arguments using the IAU
         expressions.
Inputs:  jed - JED on TDB scale.
Outputs: funarg[0]   - Mean anomaly of Moon.
         funarg[1]   - Mean anomaly of Sun.
         funarg[2]   - Argument of lat. of Moon.
         funarg[3]   - Mean elongation of Moon.
         funarg[4]   - Mean longitude of Moon's ascending node.
         funarg[5]   - Mean longitude of Moon.
         funarg[i+6] - Derivative of ith fundamental argument.
         NOTE: All derivatives are in units of rad/day.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void FunArgIAU(double jed, double *funarg) {

  static double jedz;
  double T, T2, T3, L, Ldot, lp, lpdot, F, Fdot, D, Ddot;
  double N, Ndot, LL, LLdot;

  if (jed == jedz) return;
  jedz = jed;

  T = (jed - J2000) / JulCty;
  T2 = T * T;
  T3 = T2 * T;

  /* Compute fundamental arguments */
  L = 485866.733 + (1325. * 1296000. + 715922.633) * T
    + 31.31 * T2 + 0.064 * T3;
  Ldot = (1325. * 1296000. + 715922.633) + 2. * 31.31 * T
    + 3. * 0.064 * T2;
    L = amodulo(L * A2R, TWOPI);
    Ldot = amodulo(Ldot * A2R / 36525., TWOPI);
  lp = 1287099.804 + (99. * 1296000. + 1292581.224) * T
    - 0.577 * T2 - 0.012 * T3;
  lpdot = (99. * 1296000. + 1292581.224) - 2. * 0.577 * T
    - 3. * 0.012 * T2;
    lp = amodulo(lp * A2R, TWOPI);
    lpdot = amodulo(lpdot * A2R / 36525., TWOPI);
  F = 335778.877 + (1342. * 1296000. + 295263.137) * T
    - 13.257 * T2 + 0.011 * T3;
  Fdot = (1342. * 1296000. + 295263.137) - 2. * 13.257 * T
    + 3. * 0.011 * T2;
    F = amodulo(F * A2R, TWOPI);
    Fdot = amodulo(Fdot * A2R / 36525., TWOPI);
  D = 1072261.307 + (1236. * 1296000. + 1105601.328) * T
    - 6.891 * T2 + 0.019 * T3;
  Ddot = (1236. * 1296000. + 1105601.328) - 2. * 6.891 * T
    + 3. * 0.019 * T2;
    D = amodulo(D * A2R, TWOPI);
    Ddot = amodulo(Ddot * A2R / 36525., TWOPI);
  N = 450160.28 - (5. * 1296000. + 482890.539) * T
    + 7.455 * T2 + 0.008 * T3;
  Ndot = (5. * 1296000. + 482890.539) + 2. * 7.455 * T
    + 3. * 0.008 * T2;
    N = amodulo(N * A2R, TWOPI);
    Ndot = amodulo(Ndot * A2R / 36525., TWOPI);
  LL = 785939.157 + (1336. * 1296000. + 1108372.598) * T
    - 5.802 * T2 + 0.019 * T3;
  LLdot = (1336. * 1296000. + 1108372.598) - 2. * 5.802 * T
    + 3. * 0.019 * T2;
```

```
    LL = amodulo(LL * A2R, TWOPI);
    LLdot = amodulo(LLdot * A2R / 36525., TWOPI);

  funarg[0]  = L;
  funarg[6]  = Ldot;
  funarg[1]  = lp;
  funarg[7]  = lpdot;
  funarg[2]  = F;
  funarg[8]  = Fdot;
  funarg[3]  = D;
  funarg[9]  = Ddot;
  funarg[4]  = N;
  funarg[10] = Ndot;
  funarg[5]  = LL;
  funarg[11] = LLdot;
}

/*************************************************************
Name:    GeocenObs
Purpose: Subprogram to compute the geocentric equatorial
         state vectors of an observer.  The vectors are
         referred to the J2000.0 frame.
Inputs:  JED - Julian Date on TDB scale.
Outputs: obsr_geo - Observer's geocentric state vector
                    in the celestial reference frame.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void GeocenObs(double jed, double obsr_geo[]) {

  double gast,h,CosLat,SinLat,Cos2Lat,Sin2Lat,CosLon,SinLon,C,S;
  double PolMotX,PolMotY,g[6],dpsi,deps,dpsidot,depsdot;
  double zeta,z,theta,zetadot,zdot,thetadot;
  double MeanEps,TrueEps,MeanEpsDot,TrueEpsDot;

  /* Elevation is expected to be in meters. */
  h = (obsr_ele / 1000.0) * KM2AU;
  CosLat  = cos(obsr_lat);
  Cos2Lat = CosLat * CosLat;
  SinLat  = sin(obsr_lat);
  Sin2Lat = SinLat * SinLat;
  CosLon  = cos(obsr_lon);
  SinLon  = sin(obsr_lon);
  C = 1.0 / sqrt(Cos2Lat + (1.0 - EFlat) * (1.0 - EFlat) * Sin2Lat);
  S = (1.0 - EFlat) * (1.0 - EFlat) * C;

  /* Compute the observer's state vector in the Earth-fixed frame. */
  g[0] = (EarthRadAU * C + h) * CosLat * CosLon;
  g[1] = (EarthRadAU * C + h) * CosLat * SinLon;
  g[2] = (EarthRadAU * S + h) * SinLat;
  g[3] = 0.0;
  g[4] = 0.0;
  g[5] = 0.0;

  /* Compute the required angular quantities and their derivatives. */
  GetPrecessParams(J2000, jed, &zeta, &z, &theta,
    &zetadot, &zdot, &thetadot);
  GetDpsiDeps(jed, &dpsi, &deps, &dpsidot, &depsdot);
  GetGST(jed, 1, &gast);
  Obliquity(J2000, jed, 0, &MeanEps, &MeanEpsDot);
  Obliquity(J2000, jed, 1, &TrueEps, &TrueEpsDot);
```

```
  /* Somehow get the quantities PolMotX, PolMotY */
  PolMotX = 0.0;
  PolMotY = 0.0;

  /* Wobble */
  QRotate(g, 2, PolMotX, 0, 1, g);
  QRotate(g, 1, PolMotY, 0, 1, g);

  /* Spin */
  QRotate(g, 3, -gast, -EarAngVelRD, 1, g);

  /* Nutation */
  QRotate(g, 1,  TrueEps,  TrueEpsDot, 1, g);
  QRotate(g, 3,  dpsi,  dpsidot, 1, g);
  QRotate(g, 1, -MeanEps, -MeanEpsDot, 1, g);

  /* Precession */
  QRotate(g, 3,  zeta,  zetadot, 1, g);
  QRotate(g, 2, -theta, -thetadot, 1, g);
  QRotate(g, 3,  z, zdot, 1, obsr_geo);
}

/*****************************************************************
Name:    GetGST
Purpose: Function to compute the Greenwich mean or Greenwich apparent
         sidereal time, depending on which is required.  Strictly speaking,
         the computed time is the DYNAMICAL SIDEREAL TIME, but for all
         practical purposes is the same as the observed sidereal time.
Inputs:  jed - JED on TDB scale.
         s   - 1 for apparent sidereal time, otherwise mean sidereal time.
Outputs: gst - GMST or GAST in radians.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void GetGST(double jed, int s, double *gst) {

  static double jedz, sz;
  double T, dpsi, deps, dpsidot, depsdot;
  double meaneps, trueeps, meanepsdot, trueepsdot, eqeqnx, eqeqnxdot;

  if ((jed == jedz) && (s == sz)) return;
  jedz = jed;
  sz = s;

  T = (jed - J2000) / JulCty;

  /* compute GMST in seconds */
  *gst = 67310.54841 + T * ((876600.0 * 3600.0 + 8640184.812866)
    + T * (0.093104 + T * (-0.0000062)));

  /* convert to radians */
  *gst = amodulo(*gst / 3600.0, 24.0) * H2R;

  if (s == 1) {
    /* get nutation quantities */
    GetDpsiDeps(jed, &dpsi, &deps, &dpsidot, &depsdot);

    /* get mean obliquity */
    Obliquity(J2000, jed, 0, &meaneps, &meanepsdot);

    /* get true obliquity */
    Obliquity (J2000, jed, 1, &trueeps, &trueepsdot);
```

```
    /* get equation of the equinoxes */
    eqeqnx    = dpsi * cos(trueeps);
    eqeqnxdot = dpsidot * cos(trueeps) - dpsi * trueepsdot * sin(trueeps);
    *gst     += eqeqnx;
  }

  *gst = amodulo(*gst, TWOPI);
}

/*********************************************************************
Name:    GetInvQMatrix
Purpose: Function to compute the elements of the inverse of a
         given Q-matrix.
Inputs:  QMatrix - Zero-offset 6X6 Q-matrix.
Outputs: InvQMatrix - Zero-offset inverse of QMatrix.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*********************************************************************/
void GetInvQMatrix(DMatrix QMatrix, DMatrix InvQMatrix) {

  int i, j;

  for (i= 0; i<3; i++) {
    for (j=0; j<3; j++) {
      InvQMatrix[i][j] = QMatrix[j][i];
      InvQMatrix[i][j+3] = 0.0;
      InvQMatrix[i+3][j] = QMatrix[j+3][i];
      InvQMatrix[i+3][j+3] = QMatrix[j+3][i+3];
    }
  }
}

/*********************************************************************
Name:    GetPrecessParams
Purpose: Function that computes the general precession parameters and their
         derivatives for precessing equatorial rectangular coordinates and
         velocities from jed1 to jed2. Lieske's formulae are used.
Inputs:  jed1 - Initial Julian Date.
         jed2 - Final Julian Date.
Outputs: zeta     - equatorial precession parameter.
         z        - equatorial precession parameter.
         theta    - equatorial precession parameter.
         zetadot  - derivative in rad/day.
         zdot     - derivative in rad/day.
         thetadot - derivative in rad/day.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*********************************************************************/
void GetPrecessParams(double jed1, double jed2, double *zeta, double *z,
  double *theta, double *zetadot, double *zdot, double *thetadot) {

  double T1, T2, c1, c2, c3, c4, c5, c6, p1, p2, x, xdot;

  T1 = (jed1 - J2000) / JulCty;
  T2 = (jed2 - jed1) / JulCty;

  /* compute zeta, z, theta, zetadot, zdot, thetadot */
  c1 = 2306.2181;
  c2 =    1.39656;
  c3 =   -0.000139;
```

```
  c4 =    0.30188;
  c5 =   -0.000344;
  c6 =    0.017998;
  p1 = c1 + c2 * T1 + c3 * T1 * T1;
  p2 = c4 + c5 * T1;
  x = p1 * T2 + p2 * T2 * T2 + c6 * T2 * T2 * T2;
  xdot = p1 + 2.0 * p2 * T2 + 3.0 * c6 * T2 * T2;
  *zeta = x * A2R;
  *zetadot = xdot * A2R / JulCty;

  c1 = 2306.2181;
  c2 =    1.39656;
  c3 =   -0.000139;
  c4 =    1.09468;
  c5 =    0.000066;
  c6 =    0.018203;
  p1 = c1 + c2 * T1 + c3 * T1 * T1;
  p2 = c4 + c5 * T1;
  x = p1 * T2 + p2 * T2 * T2 + c6 * T2 * T2 * T2;
  xdot = p1 + 2.0 * p2 * T2 + 3.0 * c6 * T2 * T2;
  *z = x * A2R;
  *zdot = xdot * A2R / JulCty;

  c1 = 2004.3109;
  c2 =   -0.85330;
  c3 =   -0.000217;
  c4 =   -0.42665;
  c5 =   -0.000217;
  c6 =   -0.041833;
  p1 = c1 + c2 * T1 + c3 * T1 * T1;
  p2 = c4 + c5 * T1;
  x = p1 * T2 + p2 * T2 * T2 + c6 * T2 * T2 * T2;
  xdot = p1 + 2.0 * p2 * T2 + 3.0 * c6 * T2 * T2;
  *theta = x * A2R;
  *thetadot = xdot * A2R / JulCty;
}

/*****************************************************************
Name:    GetQMatrix
Purpose: Function to compute the elements of the Q-matrix for a
         given angle and its time derivative.
Inputs:  phi    - Angle in radians
         phidot - Time der. of phi.
         axis   - 1 for x-axis,
                  2 for y-axis,
                  3 for z-axis.
         s      - 0 for inertial to inertial,
                  1 for inertial to rotating.
Outputs: QMatrix - Zero-offset 6X6 Q-matrix.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void GetQMatrix(double phi, double phidot, int axis,
  int s, DMatrix QMatrix) {

  DMatrix r, dr;
  int i, j;

  r  = createDMatrix(3);
  dr = createDMatrix(3);

  /* form the 3X3 r[] and dr[] matrices */
```

```
  RotMat(axis, phi, r, dr);

  /* form the 6X6 Q-matrix */
  for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
      QMatrix[i][j] = r[i][j];
      QMatrix[i][j+3] = 0.0;
      QMatrix[i+3][j] = phidot * dr[i][j] * s;
      QMatrix[i+3][j+3] = r[i][j];
    }
  }

  freeDMatrix(r, 3);
  freeDMatrix(dr, 3);
}

/*************************************************************
Name:    GetRPNmat
Purpose: Function to compute the precession matrix, the nutation
         matrix, or the combined R matrix for equatorial coordinates.
Inputs:  jed1 - Initial JED on TDB scale.
         jed2 - Final JED on TDB scale.
         rpn  - 1 for precession matrix,
                2 for nutation matrix,
                3 for R matrix.
         d    - 1 for zero-offset 3X3 matrix,
                2 for zero-offset 6X6 Qmatrix.
Outputs: m3 - Requested zero-offset 3X3 matrix.
         m6 - Requested zero-offset 6X6 matrix.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void GetRPNmat(double jed1, double jed2, int rpn, int d,
  DMatrix m3, DMatrix m6) {

  double dpsi, deps, dpsidot, depsdot, trueeps, trueepsdot,
    meaneps, meanepsdot, zeta, zetadot, z, zdot, theta, thetadot;
  DMatrix p1mat, p2mat, p3mat, pmat, n1mat, n2mat, n3mat,
    nmat, rmat, p, n;

  p1mat = createDMatrix(6);
  p2mat = createDMatrix(6);
  p3mat = createDMatrix(6);
  pmat  = createDMatrix(6);
  n1mat = createDMatrix(6);
  n2mat = createDMatrix(6);
  n3mat = createDMatrix(6);
  nmat  = createDMatrix(6);
  rmat  = createDMatrix(6);
  p     = createDMatrix(6);
  n     = createDMatrix(6);

  switch (rpn) {
    case 1:
      /* compute precession matrix */
      GetPrecessParams(jed1,jed2,&zeta,&z,&theta,&zetadot,&zdot,&thetadot);
      GetQMatrix(-zeta, -zetadot, 3, 0, p1mat);
      GetQMatrix(theta, thetadot, 2, 0, p2mat);
      GetQMatrix(-z, -zdot, 3, 0, p3mat);
      MatXMat(p2mat, p1mat, pmat, 6);
      MatXMat(p3mat, pmat, m3, 6);
      freeDMatrix(p3mat, 6);
```

```
      freeDMatrix(p2mat, 6);
      freeDMatrix(p1mat, 6);
      freeDMatrix(pmat, 6);
      break;
    case 2:
      /* compute nutation matrix */
      GetDpsiDeps(jed2, &dpsi, &deps, &dpsidot, &depsdot);
      Obliquity(jed1, jed2, 0, &meaneps, &meanepsdot);
      Obliquity(jed1, jed2, 1, &trueeps, &trueepsdot);
      GetQMatrix(meaneps, meanepsdot, 1, 0, n1mat);
      GetQMatrix(-dpsi, -dpsidot, 3, 0, n2mat);
      GetQMatrix(-trueeps, -trueepsdot, 1, 0, n3mat);
      MatXMat(n2mat, n1mat, nmat, 6);
      MatXMat(n3mat, nmat, m3, 6);
      freeDMatrix(n3mat, 6);
      freeDMatrix(n2mat, 6);
      freeDMatrix(n1mat, 6);
      freeDMatrix(nmat, 6);
      break;
    case 3:
      /* compute R matrix */
      GetPrecessParams(jed1,jed2,&zeta,&z,&theta,&zetadot,&zdot,&thetadot);
      GetQMatrix( -zeta, -zetadot, 3, 0, p1mat);
      GetQMatrix(theta, thetadot, 2, 0, p2mat);
      GetQMatrix(-z, -zdot, 3, 0, p3mat);
      MatXMat(p2mat, p1mat, pmat, 6);
      MatXMat(p3mat, pmat, p, 6);
      GetDpsiDeps(jed2, &dpsi, &deps, &dpsidot, &depsdot);
      Obliquity(jed1, jed2, 0, &meaneps, &meanepsdot);
      Obliquity(jed1, jed2, 1, &trueeps, &trueepsdot);
      GetQMatrix(meaneps, meanepsdot, 1, 0, n1mat);
      GetQMatrix(-dpsi, -dpsidot, 3, 0, n2mat);
      GetQMatrix(-trueeps, -trueepsdot, 1, 0, n3mat);
      MatXMat(n2mat, n1mat, nmat, 6);
      MatXMat(n3mat, nmat, n, 6);
      MatXMat(n, p, m3, 6);
      freeDMatrix(p3mat, 6);
      freeDMatrix(p2mat, 6);
      freeDMatrix(p1mat, 6);
      freeDMatrix(pmat, 6);
      freeDMatrix(p, 6);
      freeDMatrix(n3mat, 6);
      freeDMatrix(n2mat, 6);
      freeDMatrix(n1mat, 6);
      freeDMatrix(nmat, 6);
      freeDMatrix(n, 6);
      break;
  }
}

/*****************************************************************
Name:    GetStateVector
Purpose: Function to retrieve the state vector of a given body
         w.r.t. a given origin at a given time. This routine is
         a wrapper for PLEPH.
Inputs:  jd     - Julian date on TDB time scale.
         targ   - Body identification number.
                  1=MER,  2=VEN,  3=EAR,  4=MAR
                  5=JUP,  6=SAT,  7=URA,  8=NEP
                  9=PLU,  10=MOO, 11=SUN, 12=SSB
                  13=EMB, 14=NUT, 15=LIB
         cent   - Origin identification number.
                  (Numbering as above).
```

```
         recpol - 1 for cartesian, 2 for orthogonal polar.
Outputs: StateVector - A 15x15x2x6 array containing the requested
                       state vector.
         NOTE: StateVector has the following form:
         StateVector[targ-1][cent-1][recpol-1][component]
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void GetStateVector(double jd, int targ, int cent, int recpol,
  double StateVector[15][15][2][6]) {

  static double b[6] = {0.};
  static double rrd[6] = {0.};
  static double dpsi, deps, dpsidot, depsdot;
  int inside, i;

  /* Get the state vector from JPL ephemeris */
  pleph(jd, targ, cent, rrd, &inside);

  if (!inside) {
    LogMsg(stderr,
      "GetStateVector: requested date not covered by ephemeris file.\n");
    for (i=0; i<6; i++) {
      StateVector[targ-1][cent-1][recpol-1][i] = 999.;
    }
    exit (1);
  }

  /* Check for nutations */
  if (targ == 14) {
    dpsi    = rrd[0];
    deps    = rrd[1];
    dpsidot = rrd[2];
    depsdot = rrd[3];
    recpol  = 1;
  }

  if (recpol == 1) {
    for (i=0; i<6; i++) {
      StateVector[targ-1][cent-1][recpol-1][i] = rrd[i];
    }
  } else {
    /* Convert rectangular vector to polar vector */
    Rec2Pol(rrd, b);
    for (i=0; i<6; i++) {
      StateVector[targ-1][cent-1][recpol-1][i] = b[i];
    }
  }

  return;
}

/*************************************************************
Name:    HelEphemeris
Purpose: Function to compute an orbiting body's heliocentric
         ecliptic state vector for a single instant of time
         given the universal orbital elements and the time.
         Reference: Mansfield. 1986. AIAA Paper 86-2269-CP.
Inputs:  uelement[] - Array of universal orbital elements:
         uelement[0] = q
         uelement[1] = e
         uelement[2] = i
```

```
         uelement[3] = node
         uelement[4] = arg. peri.
         uelement[5] = T
         mu          - Gravitational constant for the body.
         jed         - Time.
Outputs: posvel[] - State vector:
         posvel[0..2] = position vector.
         posvel[3..5] = velocity vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void HelEphemeris(double *uelement, double mu, double jed, double *posvel) {

  int i;
  double LongPeri, RetLongPeri, cosi, sini, coslp, sinlp, cosrlp, sinrlp;
  double cosw, sinw, rcosnu, rsinnu, r, cosnu, sinnu, param, p[3], q[3];

  /* Compute longitude of perihelion */
  LongPeri = uelement[4] + uelement[3];
  /* Compute retrograde longitude of perihelion */
  RetLongPeri = uelement[4] - uelement[3];

  /* Compute the P vector */
  cosi   = cos(uelement[2]);
  sini   = sin(uelement[2]);
  coslp  = cos(LongPeri);
  sinlp  = sin(LongPeri);
  cosrlp = cos(RetLongPeri);
  sinrlp = sin(RetLongPeri);
  cosw   = cos(uelement[4]);
  sinw   = sin(uelement[4]);
  p[0] = 0.5 * (1.0 + cosi) * coslp + 0.5 * (1.0 - cosi) * cosrlp;
  p[1] = 0.5 * (1.0 + cosi) * sinlp - 0.5 * (1.0 - cosi) * sinrlp;
  p[2] = sinw * sini;

  /* Compute the Q vector */
  q[0] = -0.5 * (1.0 + cosi) * sinlp - 0.5 * (1.0 - cosi) * sinrlp;
  q[1] =  0.5 * (1.0 + cosi) * coslp - 0.5 * (1.0 - cosi) * cosrlp;
  q[2] = cosw * sini;

  /* Solve Kepler's equation */
  Conway(uelement, mu, jed, &rcosnu, &rsinnu);

  /* Compute magnitude of radius vector */
  r = sqrt(rcosnu * rcosnu + rsinnu * rsinnu);
  cosnu = rcosnu / r;
  sinnu = rsinnu / r;

  /* Compute heliocentric ecliptic position vector */
  for (i = 0; i < 3; i++) {
    posvel[i] = p[i] * rcosnu + q[i] * rsinnu;
  }

  /* Compute heliocentric ecliptic velocity vector */
  param = uelement[0] * (1.0 + uelement[1]);

  for (i = 3; i < 6; i++) {
    posvel[i]   = -p[i-3] * sqrt(mu / param) * sinnu;
    posvel[i] = posvel[i] + q[i-3] * sqrt(mu / param) *
      (uelement[1] + cosnu);
  }
}
```

```
/*************************************************************
Name:    interp
Purpose: This subroutine differentiates and interpolates a set of
         Chebyshev coefficients to give position and velocity.
Inputs:  buff - 1st location of array of Chebyshev coefficients.
         t    - t[0] is fractional time in interval covered by
                coefficients at which interpolation is wanted
                (0 <= t[0] <= 1). t[1] is length of whole
                interval in input time units.
         ncf  - Number of coefficients per component.
         ncm  - Number of components per set of coefficients.
         na   - Number of sets of coefficients in full array
                (i.e., # of sub-intervals in full interval).
         fl   - Integer flag:  =1 for positions only.
                               =2 for pos and vel.
Outputs: dumpv - Interpolated quantities requested. Dimension
         expected is pv[ncm][fl].
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void interp(int buff, double *t, int ncf, int ncm, int na,
  int fl, double dumpv[3][2]) {

  int i, j, l, n, m;
  static int bcoef, np, nv;
  static double pc[18], vc[18], cbody[1200];
  static double cbuf[15][3][8]={0.};
  static double twot, dna, dt1, temp, ll, tc, vfac;

  np = 2;
  nv = 3;
  twot = 0.0;
  pc[0] = 1.0;
  pc[1] = 0.0;
  vc[1] = 1.0;

  /*
      Entry point. Get correct sub-interval number for this set
      of coefficients and then get normalized chebyshev time
      within that subinterval.
  */

  dna = (double) na;
  dt1 = floor(t[0]);
  temp = dna*t[0];
  ll = floor((temp - dt1) + 1.0);

  /* 'tc' is the normalized chebyshev time (-1 <= tc <= 1) */
  tc = 2.0*(fmod(temp, 1.0) + dt1) - 1.0;

  /*
      Check to see whether chebyshev time has changed,
      and compute new polynomial values if it has.
      (The element pc[1] is the value of 'tc' and hence
      contains the value of 'tc' on the previous call.)
  */

  if (tc != pc[1]) {
    np = 2;
    nv = 3;
    pc[1] = tc;
```

```
  twot = tc + tc;
}

/*
    Be sure that at least 'ncf' polynomials have been evaluated
    and are stored in the array pc[].
*/

if (np < ncf) {
  for (i=np; i<ncf; i++) {
    pc[i] = twot * pc[i-1] - pc[i-2];
  }
  np = ncf;
}

/* interpolate to get position for each component */

/* number of coefficients for body */
bcoef= ncf*na*ncm;

/* stored body's coefficients in an array */
n = buff;
for (m=0; m<bcoef; m++) {
  cbody[m] = db[n];
  n++;
}

/* fill the cbuf[][][] array */
n = 0;
/* loop for each sub-interval */
for (l=0; l<na; l++) {
  /* loop for each component */
  for (i=0; i<ncm; i++) {
    /* loop for each set of coeffs */
    for (j=0; j<ncf; j++) {
      cbuf[j][i][l] = cbody[n];
      n++;
    }
  }
}

for (i=0; i<ncm; i++) {
  dumpv[i][0] = 0.0;
  for (j=ncf-1; j>=0; j--) {
    dumpv[i][0] = dumpv[i][0] + pc[j] * cbuf[j][i][((int)ll)-1];
  }
}
if (fl <= 1) return;

/*
    If velocity interpolation is wanted, be sure enough
    derivative polynomials have been generated and stored.
*/

vfac = (dna+dna)/t[1];
vc[2] = twot+twot;
if (nv < ncf) {
  for (i=nv; i<ncf; i++) {
    vc[i]=twot * vc[i-1] + pc[i-1] + pc[i-1] - vc[i-2];
  }
  nv = ncf;
}
```

```
  /* interpolate to get velocity for each component */

  for (i=0; i<ncm; i++) {
    dumpv[i][1] = 0.0;
    for (j=ncf-1; j>=1; j--) {
      dumpv[i][1] = dumpv[i][1] + vc[j] * cbuf[j][i][((int)ll)-1];
    }
    dumpv[i][1] = dumpv[i][1] * vfac;
  }
}

/*************************************************************
Name:    Interp1
Purpose: Subprogram to perform interpolation on a list
         of tabular values to find maxima/minima or
         to find the zero of a function within the limits
         of the table.  The algorithms used are those of Meeus.
Inputs:  x[] - Array of function arguments.
         y[] - Array of function values at the
               corresponding arguments.
         L   - Equals 3 for 3-pt. interpolation.
               5 for 5-pt. interpolation.
         m   - Equals 0 for zero of function.
               1 for a extremum of the function.
Outputs: arg - Argument corresponding to the
               extremum or zero of the tabular function.
         v   - The value of the function corresponding to arg.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Interp1(double *x, double *y, int L, int m, double *arg, double *v) {

  double a, b, c, d, e, f, g, h, j, k, nm, n0, n1;

  if (L == 3) {
    /* Use a 3 pt. table */
    a = y[1] - y[0];
    b = y[2] - y[1];
    c = b - a;
    if (m == 1) {
      /* Find extremum of the tabular function */
      *v = y[1] - (a + b) * (a + b) / (8.0 * c);
      nm = -(a + b) / (2.0 * c);
      *arg = x[1] + nm * (x[1] - x[0]);
    } else {
      n1 = 0.0; /* Find zero of the tabular function */
      do {
        n0 = -2.0 * y[1] / (a + b + c * n1);
        if (fabs(n0 - n1) < 0.000000000000001)
          break;
        n1 = n0;
      } while (1 == 1);
      *v = y[1] + 0.5 * n0 * (x[1] - x[0]) * (a + b + n0 * c);
      *arg = x[1] + n0 * (x[1] - x[0]);
    }
  } else {
    /* Use a 5 pt. table */
    a = y[1] - y[0];
    b = y[2] - y[1];
    c = y[3] - y[2];
    d = y[4] - y[3];
    e = b - a;
```

```
    f = c - b;
    g = d - c;
    h = f - e;
    j = g - f;
    k = j - h;
    if (m == 0) {
      /* Find extremum of tabular function */
      n1 = 0.0;
      do {
        nm = (6.0 * b + 6.0 * c - h - j + 3.0 * n1 * n1 *
          (h + j) + 2.0 * n1 * n1 * n1 * k) / (k - 12.0 * f);
        if (fabs(nm - n1) < 0.000000000000001)
          break;
        n1 = nm;
      } while (1 == 1);
      nm = nm * (x[3] - x[2]);
      *arg = x[2] + nm;
      *v = y[2] + 0.5 * nm * (b + c) + 0.5 * nm * nm * f + nm
        * (nm * nm - 1.0) * (h + j) / 12.0 + nm * nm * (nm * nm - 1.0)
        * k / 24.0;
    } else {
      /* Find zero of the tabular function */
      n1 = 0.0;
      do {
        n0 = (-24.0 * y[2] + n1 * n1 * (k - 12.0 * f) - 2.0 *
          n1 * n1 * n1 * (h + j) - n1 * n1 * n1 * n1 * k) / (2.0
          * (6.0 * b + 6.0 * c - h - j));
        if (fabs(n0 - n1) < 0.000000000000001)
          break;
        n1 = n0;
      } while (1 == 1);
      n0 = n0 * (x[3] - x[2]);
      *arg = x[2] + n0;
      *v = y[2] + 0.5 * n0 * (b + c) + 0.5 * n0 * n0 * f + n0 * (n0 * n0 -
1.0) * (h + j) / 12.0 + n0 * n0 * (n0 * n0 - 1.0) * k / 24.0;
    }
  }
}

/*****************************************************************
Name:    Interpol
Purpose: Function for 3 or 5 point interpolation
         using Meeus' algorithm.
Inputs:  x[] - Aarray of function arguments.
         y[] - Array of function values at
               the three arguments.
         i   - Switch indicating whether a
               3 or 5 point interpolation
               is performed (i=3 for 3 pt.,
               i=5 for 5 pt.).
         arg - Argument for which a function value
               is needed.
Outputs: None.
Returns: Interpolated value.
Status:  Finished.
Errors:  None known.
*****************************************************************/
double Interpol(double *x, double *y, int i, double arg) {

  double d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, n;

  if (i == 3) {
    d1 = y[1] - y[0];
```

```
    d2 = y[2] - y[1];
    d3 = d2 - d1;
    n = (arg - x[1]) / (x[1] - x[0]);
    return (y[1] + (n / 2.0) * (d1 + d2 + n * d3));
  } else {
    d1 = y[1] - y[0];
    d2 = y[2] - y[1];
    d3 = y[3] - y[2];
    d4 = y[4] - y[3];
    d5 = d2 - d1;
    d6 = d3 - d2;
    d7 = d4 - d3;
    d8 = d6 - d5;
    d9 = d7 - d6;
    d10 = d9 - d8;
    n = (arg - x[2]) / (x[2] - x[1]);
    return (y[2] + n * ((d2 + d3) / 2.0 - (d8 + d9) / 12.0) +
      n * n * (d6 / 2.0 - d10 / 24.0) + n * n * n *
      ((d8 + d9) / 12.0) + n * n * n * n * (d10 / 24.0));
  }
}

/*************************************************************
Name:    JED2Cal
Purpose: Function to convert a JED to an ordinary calendar date.
         The algorithm is that of Meeus.
Inputs:  jed - Julian Ephemeris Date.
Outputs: yr - Year.
         mo - Month.
         dy - Day.
         ti - Time of day in decimal hours.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void JED2Cal(double jed, int *yr, int *mo, int *dy, double *ti) {

  static double jedz;
  double z, f, a, b, c, d, e, alpha, daywtime;

  if (jed == jedz) return;
  jedz = jed;

  z = floor(jed + 0.5);
  f = (jed + 0.5) - z + 5.0e-10; /* fudge factor */
  if (z < 2299161.0) {
    a = z;
  } else {
    alpha = floor((z - 1867216.25) / 36524.25);
    a = z + 1.0 + alpha - floor(0.25 * alpha);
  }

  b = a + 1524.0;
  c = floor((b - 122.1) / 365.25);
  d = floor(365.25 * c);
  e = floor((b - d) / 30.6001);
  daywtime = b - d - floor(30.6001 * e) + f;
  *dy = (int) floor(daywtime);
  *ti = 24.0 * (daywtime - (double) *dy);

  if (e  < 13.5) *mo = (int) floor(e - 1.0);
  if (e  > 13.5) *mo = (int) floor(e - 13.0);
  if (*mo > 2.5) *yr = (int) floor(c - 4716.0);
```

```
  if (*mo < 2.5) *yr = (int) floor(c - 4715.0);
}

/*************************************************************
Name:    JED2Epoch
Purpose: Function to convert a given JED to its corresponding Julian
         or Besselian epoch. For example, 2451545.0 becomes J2000.
Inputs:  jed - Input jed.
         s   - J or B, whichever is desired.
Outputs: epoch - J or B epoch as a string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void JED2Epoch(double jed, char *s, char *epoch) {

  char d[80];
  double date;

  if (strcmp(s,"J") == 0) {
    date = 2000.0 + (jed - 2451545.0) / 365.25;
    sprintf(d, "%f", date);
  } else {
    date = 1900.0 + (jed - (double) 2415020.31352) /
      (double) 365.242198781731;
    sprintf(d, "%f", date);
  }

  strcpy(epoch, s);
  strcat(epoch, d);
}

/*************************************************************
Name:    LightTime
Purpose: Subprogram to compute the vectors body_geo() and body_hel()
         from a barycentric ephemeris of the object, Earth, and the Sun.
Inputs:  JED       - Desired time of reduction.
         Body      - Body number, 99 for stellar reduction.
         earth_ssb - Barycentric state vector of Earth at JED.
         earth_hel - Heliocentric state vector of Earth.
         StarData  - Stellar data.
Outputs: body_geo - Geometric geocentric state vector of object.
         body_hel - Heliocentric state vector of object.
         Ltime    - Light time in days to the object.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void LightTime(double jed, int body, double earth_ssb[],
  double earth_hel[], double StarData[], double body_geo[],
  double body_hel[], double *Ltime) {

  int inside, i;
  double E[3],Edot[3],RQB[6],RSB[6],RQ[6],RP[6],Q[3],Qdot[3],P[3],Pdot[3];
  double unit_body_hel[6], unit_body_geo[6];
  double magE,magQ,magP,trialtau,tau,xtime,RelTerm,
         TrueHelDist=0,TrueGeoDist=0;
  double AppHelDist,AppGeoDist,RA,DE,sinRA,cosRA,sinDE,cosDE;
  double plx,r,muRA,muDE,rdot,drda[3],drdd[3],DT;

  SplitStateVector(earth_hel, E, Edot);
  magE = Vecmag(E);
```

```
if (body != 99) {
  trialtau = 0.0;
  do {
    xtime = jed - trialtau;

    /* body's barycentric state vector */
    pleph(xtime, body, 12, RQB, &inside);

    /* Sun's barycentric state vector */
    pleph(xtime, 11, 12, RSB, &inside);

    /* body's heliocentric state vector */
    for (i=0; i<6; i++) {
      RQ[i] = RQB[i] - RSB[i];
      /* body's geocentric state vector */
      RP[i] = RQB[i] - earth_ssb[i];
    }

    /* Calculate improved value of the light time */
    SplitStateVector(RQ, Q, Qdot);
    SplitStateVector(RP, P, Pdot);
    magQ = Vecmag(Q);
    magP = Vecmag(P);

    if (trialtau == 0) {
      TrueHelDist = magQ;
      TrueGeoDist = magP;
    }

    if (magQ == 0) {
      RelTerm = 0.0;
    } else {
      RelTerm = MUC * log((magE + magP + magQ) / fabs(magE - magP +
                          magQ));
    }
    tau = (magP + RelTerm) / CAUD;
    if (fabs(trialtau - tau) < 1e-9) {
      break;
    }
    trialtau = tau;
  } while(1);
} else {
  RA = StarData[0];
  DE = StarData[1];
  sinRA = sin(RA);
  cosRA = cos(RA);
  sinDE = sin(DE);
  cosDE = cos(DE);
  if (StarData[2] == 0) {
    plx = 1e-7;
  } else {
    plx = StarData[2];
  }

  /* Star's distance in AU */
  r = R2A / plx;

  /* Star's barycentric position vector */
  RQB[0] = r*cosRA*cosDE;
  RQB[1] = r*sinRA*cosDE;
  RQB[2] = r      *sinDE;

  /* Convert proper motions and radial velocity */
```

```
  /* to units of AU/day */
  muRA = StarData[3] * 15.0 * cosDE / (JulCty * plx);
  muDE = StarData[4] / (JulCty * plx);
  rdot = StarData[5] * 86400.0 * KM2AU;

  /* Partial of unit vector wrt RA */
  drda[0] = - sinRA * cosDE;
  drda[1] =   cosRA * cosDE;
  drda[2] =   0.0;

  /* Partial of unit vector wrt DE */
  drdd[0] = - cosRA * sinDE;
  drdd[1] = - sinRA * sinDE;
  drdd[2] =           cosDE;

  /* Star's barycentric velocity vector */
  for (i=3; i<6; i++) {
    RQB[i] = muRA*drda[i-3] + muDE*drdd[i-3] + rdot*RQB[i-3]/r;
  }

  /* Correction for space motion */
  DT = jed - J2000;
  for (i=0; i<3; i++) {
    RQB[i] = RQB[i] + RQB[i+3]*DT;
  }

  /* Sun's barycentric state vector */
  pleph(jed, 11, 12, RSB, &inside);

  /* Star's heliocentric state vector */
  for (i=0; i<6; i++) {
    RQ[i] = RQB[i] - RSB[i];
  }

  /* Star's geo/topocentric state vector */
  /* This correction is annual parallax. */
  for (i=0; i<6; i++) {
    RP[i] = RQB[i] - earth_ssb[i];
  }

  SplitStateVector(RQ, Q, Qdot);
  SplitStateVector(RP, P, Pdot);
  magQ = Vecmag(Q);
  magP = Vecmag(P);

  TrueGeoDist = magP;
  TrueHelDist = magQ;

  tau = r / CAUD;
}

*Ltime = tau;
AppHelDist = magQ;
AppGeoDist = magP;

for (i=0; i<6; i++) {
  body_hel[i] = RQ[i];
  body_geo[i] = RP[i];
}

Uvector(body_hel, unit_body_hel);
Uvector(body_geo, unit_body_geo);
```

```
  /* I don't understand why this next loop is needed, but it is. */
  for (i=3; i<6; i++) {
   unit_body_hel[i] = RQ[i];
   unit_body_geo[i] = RP[i];
  }

  for (i=0; i<6; i++) {
    body_geo[i] = unit_body_geo[i] * TrueGeoDist;
    body_hel[i] = unit_body_hel[i] * TrueHelDist;
  }
}

/*************************************************************
Name:    matrix
Purpose: Function that allocates a double matrix with dimensions
         m[nrow][ncol].
Inputs:  nrow - Number of rows.
         ncol - Number of columns.
Outputs: None.
Returns: Pointer to 2-D matrix of doubles.
Status:  Finished.
Errors:  None known.
*************************************************************/
double **matrix(int nrow, int ncol) {

  long i;
  double **m;

  /* Allocate pointer to rows */
  m = (double **) malloc(nrow * sizeof(double *));
  if (!m) {
    LogMsg(stderr, "matrix(): allocation failure 1\n");
    exit(1);
  }

  /* Allocate rows and set pointers to them */
  for (i = 0;i < nrow;i++) {
    m[i] = (double *) malloc(ncol * sizeof(double));
    if (!m[i]) {
      LogMsg(stderr, "matrix(): allocation failure 2\n");
      exit(1);
    }
  }
  /* return pointer to array of pointers to rows */
  return m;
}

/*************************************************************
Name:    MatXMat
Purpose: Square matrix multiplication subroutine.
         NOTE: If a[][] and c[][] have the same name in the
         calling program, the original a[][] matrix is overwritten.
Inputs:  a[][] - Zero-offset matrix a.
         b[][] - Zero-offset matrix b.
         n     - Dimension of matrix.
Outputs: c[][] - Zero-offset matrix a x b.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void MatXMat(DMatrix a, DMatrix b, DMatrix c, int n) {

  int i, j, k;
```

```
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      c[i][j] = 0.0;
      for (k=0; k<n; k++) {
        c[i][j] += a[i][k] * b[k][j];
      }
    }
  }
}

/*************************************************************
Name:    MatXVec
Purpose: Matrix/vector multiplication subroutine.
Inputs:  a[][] - Zero-offset matrix a (l rows by m columns).
         b[]   - Zero-offset vector b (m rows by 1 column).
         l     - Dimension of vector (i.e. lx1).
         m     - Dimension of matrix (i.e. mxl).
Outputs: c[]   - Zero-offset vector a x b (l rows by 1 column).
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void MatXVec (DMatrix a, double b[], double c[], int l, int m) {

  int i, j;
  double s, temp[6];

  for (i=0; i<l; i++) {
    s = 0.0;
    for (j=0; j<m; j++) {
      s += a[i][j] * b[j];
    }
    temp[i] = s;
  }

  for (i=0; i<l; i++) {
    c[i] = temp[i];
  }
}

/*************************************************************
Name:    MRotate
Purpose: Function to perform a matrix rotation of an input vector
         through a given angle about a desired axis. A right-handed
         orthogonal coordinate system is assumed, and a 3X3
         rotation matrix is used, not a Q-matrix.
         NOTE: The vin[] and vout[] can have the same name in the
         calling program.
Inputs:  vin[] - Zero-offset input vector.
         axis  - 1 for rot. about x-axis,
                 2 for rot. about y-axis,
                 3 for rot. about z-axis.
         phi   - Rotation angle in radians.
Outputs: vout[] - Zero-offset transformed vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void MRotate(double vin[], int axis, double phi, double vout[]) {

  double cosphi, sinphi, T;
```

```
  cosphi = cos(phi);
  sinphi = sin(phi);

  switch (axis) {
    case 1:  /* rotate about x-axis */
      T = cosphi * vin[1] + sinphi * vin[2];
      vout[2] = -sinphi * vin[1] + cosphi * vin[2];
      vout[1] = T;
      vout[0] = vin[0];
      break;
    case 2:  /* rotate about y-axis */
      T = cosphi * vin[0] - sinphi * vin[2];
      vout[2] = sinphi * vin[0] + cosphi * vin[2];
      vout[0] = T;
      vout[1] = vin[1];
      break;
    case 3:  /* rotate about z-axis */
      T = cosphi * vin[0] + sinphi * vin[1];
      vout[1] = -sinphi * vin[0] + cosphi * vin[1];
      vout[0] = T;
      vout[2] = vin[2];
      break;
    default:
      LogMsg(stderr,"MRotate: axis not valid.\n");
      exit(1);
  }
}

/*************************************************************
Name:    GetDpsiDeps
Purpose: Function to compute dpsi, deps, dpsidot, and depsdot.
Inputs:  jed - JED on TDB scale.
Outputs: dpsi    - Nutation in longitude.
         deps    - Nutation in obliquity.
         dpsidot - Derivative of dpsi in radians/day.
         depsdot - Derivative of deps in radians/day.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void GetDpsiDeps(double jed, double *dpsi, double *deps,
  double *dpsidot, double *depsdot) {

  double L, Ldot, LP, LPdot, F, Fdot, D, Ddot, N, Ndot, LL, LLdot;
  double T, c1, c2, c3, c4, c5, lamp, oamp, ls, os, arg, argdot;
  double funarg[12];
  int j;
  static double dpsiz, depsz, dpsidotz, depsdotz, jedz;

  /* 1980 IAU nutation coefficients */
  static double nc[106][9] = {
    {-171996., -174.2, 92025., 8.9,  0., 0., 0., 0., 1.},
    {   2062.,    0.2,  -895., 0.5,  0., 0., 0., 0., 2.},
    {     46.,     0.,   -24.,  0., -2., 0., 2., 0., 1.},
    {     11.,     0.,     0.,  0.,  2., 0.,-2., 0., 0.},
    {     -3.,     0.,     1.,  0., -2., 0., 2., 0., 2.},
    {     -3.,     0.,     0.,  0.,  1.,-1., 0.,-1., 0.},
    {     -2.,     0.,     1.,  0.,  0.,-2., 2.,-2., 1.},
    {      1.,     0.,     0.,  0.,  2., 0.,-2., 0., 1.},
    { -13187.,   -1.6,  5736., -3.1, 0., 0., 2.,-2., 2.},
    {   1426.,   -3.4,    54., -0.1, 0., 1., 0., 0., 0.},
    {   -517.,    1.2,   224., -0.6, 0., 1., 2.,-2., 2.},
    {    217.,   -0.5,   -95.,  0.3, 0.,-1., 2.,-2., 2.},
```

```
{    129.,    0.1,   -70.,  0.,  0., 0., 2.,-2., 1.},
{     48.,     0.,     1.,  0.,  2., 0., 0.,-2., 0.},
{    -22.,     0.,     0.,  0.,  0., 0., 2.,-2., 0.},
{     17.,   -0.1,     0.,  0.,  0., 2., 0., 0., 0.},
{    -15.,     0.,     9.,  0.,  0., 1., 0., 0., 1.},
{    -16.,    0.1,     7.,  0.,  0., 2., 2.,-2., 2.},
{    -12.,     0.,     6.,  0.,  0.,-1., 0., 0., 1.},
{     -6.,     0.,     3.,  0., -2., 0., 0., 2., 1.},
{     -5.,     0.,     3.,  0.,  0.,-1., 2.,-2., 1.},
{      4.,     0.,    -2.,  0.,  2., 0., 0.,-2., 1.},
{      4.,     0.,    -2.,  0.,  0., 1., 2.,-2., 1.},
{     -4.,     0.,     0.,  0.,  1., 0., 0.,-1., 0.},
{      1.,     0.,     0.,  0.,  2., 1., 0.,-2., 0.},
{      1.,     0.,     0.,  0.,  0., 0.,-2., 2., 1.},
{     -1.,     0.,     0.,  0.,  0., 1.,-2., 2., 0.},
{      1.,     0.,     0.,  0.,  0., 1., 0., 0., 2.},
{      1.,     0.,     0.,  0., -1., 0., 0., 1., 1.},
{     -1.,     0.,     0.,  0.,  0., 1., 2.,-2., 0.},
{  -2274.,   -0.2,   977., -0.5, 0., 0., 2., 0., 2.},
{    712.,    0.1,    -7.,   0., 1., 0., 0., 0., 0.},
{   -386.,   -0.4,   200.,   0., 0., 0., 2., 0., 1.},
{   -301.,     0.,   129., -0.1, 1., 0., 2., 0., 2.},
{   -158.,     0.,    -1.,   0., 1., 0., 0.,-2., 0.},
{    123.,     0.,   -53.,   0.,-1., 0., 2., 0., 2.},
{     63.,     0.,    -2.,   0., 0., 0., 0., 2., 0.},
{     63.,    0.1,   -33.,   0., 1., 0., 0., 0., 1.},
{    -58.,   -0.1,    32.,   0.,-1., 0., 0., 0., 1.},
{    -59.,     0.,    26.,   0.,-1., 0., 2., 2., 2.},
{    -51.,     0.,    27.,   0., 1., 0., 2., 0., 1.},
{    -38.,     0.,    16.,   0., 0., 0., 2., 2., 2.},
{     29.,     0.,    -1.,   0., 2., 0., 0., 0., 0.},
{     29.,     0.,   -12.,   0., 1., 0., 2.,-2., 2.},
{    -31.,     0.,    13.,   0., 2., 0., 2., 0., 2.},
{     26.,     0.,    -1.,   0., 0., 0., 2., 0., 0.},
{     21.,     0.,   -10.,   0.,-1., 0., 2., 0., 1.},
{     16.,     0.,    -8.,   0.,-1., 0., 0., 2., 1.},
{    -13.,     0.,     7.,   0., 1., 0., 0.,-2., 1.},
{    -10.,     0.,     5.,   0.,-1., 0., 2., 2., 1.},
{     -7.,     0.,     0.,   0., 1., 1., 0.,-2., 0.},
{      7.,     0.,    -3.,   0., 0., 1., 2., 0., 2.},
{     -7.,     0.,     3.,   0., 0.,-1., 2., 0., 2.},
{     -8.,     0.,     3.,   0., 1., 0., 2., 2., 2.},
{      6.,     0.,     0.,   0., 1., 0., 0., 2., 0.},
{      6.,     0.,    -3.,   0., 2., 0., 2.,-2., 2.},
{     -6.,     0.,     3.,   0., 0., 0., 0., 2., 1.},
{     -7.,     0.,     3.,   0., 0., 0., 2., 2., 1.},
{      6.,     0.,    -3.,   0., 1., 0., 2.,-2., 1.},
{     -5.,     0.,     3.,   0., 0., 0., 0.,-2., 1.},
{      5.,     0.,     0.,   0., 1.,-1., 0., 0., 0.},
{     -5.,     0.,     3.,   0., 2., 0., 2., 0., 1.},
{     -4.,     0.,     0.,   0., 0., 1., 0.,-2., 0.},
{      4.,     0.,     0.,   0., 1., 0.,-2., 0., 0.},
{     -4.,     0.,     0.,   0., 0., 0., 0., 1., 0.},
{     -3.,     0.,     0.,   0., 1., 1., 0., 0., 0.},
{      3.,     0.,     0.,   0., 1., 0., 2., 0., 0.},
{     -3.,     0.,     1.,   0., 1.,-1., 2., 0., 2.},
{     -3.,     0.,     1.,   0.,-1.,-1., 2., 2., 2.},
{     -2.,     0.,     1.,   0.,-2., 0., 0., 0., 1.},
{     -3.,     0.,     1.,   0., 3., 0., 2., 0., 2.},
{     -3.,     0.,     1.,   0., 0.,-1., 2., 2., 2.},
{      2.,     0.,    -1.,   0., 1., 1., 2., 0., 2.},
{     -2.,     0.,     1.,   0.,-1., 0., 2.,-2., 1.},
{      2.,     0.,    -1.,   0., 2., 0., 0., 0., 1.},
```

```
  {     -2.,     0.,     1.,   0., 1., 0., 0., 0., 2.},
  {      2.,     0.,     0.,   0., 3., 0., 0., 0., 0.},
  {      2.,     0.,    -1.,   0., 0., 0., 2., 1., 2.},
  {      1.,     0.,    -1.,   0.,-1., 0., 0., 0., 2.},
  {     -1.,     0.,     0.,   0., 1., 0., 0.,-4., 0.},
  {      1.,     0.,    -1.,   0.,-2., 0., 2., 2., 2.},
  {     -2.,     0.,     1.,   0.,-1., 0., 2., 4., 2.},
  {     -1.,     0.,     0.,   0., 2., 0., 0.,-4., 0.},
  {      1.,     0.,    -1.,   0., 1., 1., 2.,-2., 2.},
  {     -1.,     0.,     1.,   0., 1., 0., 2., 2., 1.},
  {     -1.,     0.,     1.,   0.,-2., 0., 2., 4., 2.},
  {      1.,     0.,     0.,   0.,-1., 0., 4., 0., 2.},
  {      1.,     0.,     0.,   0., 1.,-1., 0.,-2., 0.},
  {      1.,     0.,    -1.,   0., 2., 0., 2.,-2., 1.},
  {     -1.,     0.,     0.,   0., 2., 0., 2., 2., 2.},
  {     -1.,     0.,     0.,   0., 1., 0., 0., 2., 1.},
  {      1.,     0.,     0.,   0., 0., 0., 4.,-2., 2.},
  {      1.,     0.,     0.,   0., 3., 0., 2.,-2., 2.},
  {     -1.,     0.,     0.,   0., 1., 0., 2.,-2., 0.},
  {      1.,     0.,     0.,   0., 0., 1., 2., 0., 1.},
  {      1.,     0.,     0.,   0.,-1.,-1., 0., 2., 1.},
  {     -1.,     0.,     0.,   0., 0., 0.,-2., 0., 1.},
  {     -1.,     0.,     0.,   0., 0., 0., 2.,-1., 2.},
  {     -1.,     0.,     0.,   0., 0., 1., 0., 2., 0.},
  {     -1.,     0.,     0.,   0., 1., 0.,-2.,-2., 0.},
  {     -1.,     0.,     0.,   0., 0.,-1., 2., 0., 1.},
  {     -1.,     0.,     0.,   0., 1., 1., 0.,-2., 1.},
  {     -1.,     0.,     0.,   0., 1., 0.,-2., 2., 0.},
  {      1.,     0.,     0.,   0., 2., 0., 0., 2., 0.},
  {     -1.,     0.,     0.,   0., 0., 0., 2., 4., 2.},
  {      1.,     0.,     0.,   0., 0., 1., 0., 1., 0.}
};

if (jedz != jed) {
  jedz = jed;

  T = (jed - J2000) / JulCty;

  FunArgIAU(jed, funarg);
  L = funarg[0];
  Ldot = funarg[6];
  LP = funarg[1];
  LPdot = funarg[7];
  F  = funarg[2];
  Fdot  = funarg[8];
  D  = funarg[3];
  Ddot  = funarg[9];
  N  = funarg[4];
  Ndot  = funarg[10];
  LL = funarg[5];
  LLdot = funarg[11];

  /* evaluate the series */
  dpsiz = 0.;  /* initialize to zero */
  depsz = 0.;
  dpsidotz = 0.;
  depsdotz = 0.;
  for (j=0; j<106; j++) {
    lamp = nc[j][0];
    oamp = nc[j][2];
    ls   = nc[j][1];
    os   = nc[j][3];
    c1   = nc[j][4];
```

```
      c2   = nc[j][5];
      c3   = nc[j][6];
      c4   = nc[j][7];
      c5   = nc[j][8];
      arg  = c1 * L + c2 * LP + c3 * F + c4 * D + c5 * N;
      arg  = amodulo(arg, TWOPI);
      dpsiz += (lamp + ls * T) * sin(arg);
      depsz += (oamp + os * T) * cos(arg);
      argdot = c1 * Ldot + c2 * LPdot + c3 * Fdot + c4 * Ddot + c5 * Ndot;
      argdot = amodulo(argdot, TWOPI);
      dpsidotz += (lamp + ls * T) * argdot * cos(arg) +
        ls * sin(arg) / JulCty;
      depsdotz -= (oamp + os * T) * argdot * sin(arg) +
        os * cos(arg) / JulCty;
    }

    /* normalize and convert units */
    dpsiz *= 0.0001 * A2R;
    depsz *= 0.0001 * A2R;
    dpsidotz *= 0.0001 * A2R;
    depsdotz *= 0.0001 * A2R;
  }

  *dpsi = dpsiz;
  *deps = depsz;
  *dpsidot = dpsidotz;
  *depsdot = depsdotz;
}

/*************************************************************
Name:    Obliquity
Purpose: Function to compute the mean obliquity of the ecliptic
         and its derivative using Lieske's formula.
Inputs:  jed1 - Initial JED on the TDB scale.
         jed2 - Final JED on the TDB scale.
         m    - 0 for mean obliquity,
                1 for true obliquity.
Outputs: obl    - Obliquity of the ecliptic in radians.
         obldot - Derivative in radians/day.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Obliquity(double jed1, double jed2, int m, double *obl,
  double *obldot) {

  double t1, t2, e0, e1, e2, e3, e4, e5, e6, epsbar;
  double dpsi, deps, dpsidot, depsdot;

  t1 = (jed1 - J2000) / JulCty;
  t2 = (jed2 - jed1) / JulCty;

  e0 = 84381.448;
  e1 =   -46.815;
  e2 =    -0.00059;
  e3 =     0.001813;
  epsbar = e0 + e1 * t1 + e2 * t1 * t1 + e3 * t1 * t1 * t1;
  e1 = -46.815;
  e2 =  -0.00117;
  e3 =   0.005439;
  e4 =  -0.00059;
  e5 =   0.005439;
  e6 =   0.001813;
```

```
  *obl = epsbar + (e1 + t1 * (e2 + t1 * e3)) * t2
    + (e4 + e5 * t1) * t2 * t2
    + e6 * t2 * t2 * t2;
  *obldot = e1 + t1 * (e2 + t1 * e3)
    + 2.0 * (e4 + e5 * t1) * t2
    + 3.0 * e6 * t2 * t2;

  if (m == 1) {
    /* need true obliquity */
    GetDpsiDeps(jed2, &dpsi, &deps, &dpsidot, &depsdot);
    /* Unit conversion is needed because obl is  */
    /* in arc seconds, nutations are in radians. */
    *obl = *obl + deps * R2A;
    *obldot = *obldot + depsdot * R2A;
  }

  /* convert to radians and radians/day */
  *obl = *obl * A2R;
  *obldot = *obldot * A2R / JulCty;
}

/*************************************************************
Name:    pleph
Purpose: Function that reads the JPL planetary ephemeris and gives the
         position and velocity of the point 'targ' with respect to 'cent'.
Inputs:  jd   - JED at which interpolation is wanted.
         targ - Number of target point.
         cent - Number of center point.
         The numbering convention for 'targ' and 'cent' is:
         1 = Mercury           8 = Neptune
         2 = Venus             9 = Pluto
         3 = Earth            10 = Moon
         4 = Mars             11 = Sun
         5 = Jupiter          12 = SSB
         6 = Saturn           13 = EMB
         7 = Uranus           14 = Nutations  (if present)
         15 = Librations (if present)
         If nutations are wanted, set targ = 14. For librations,
         set targ = 15. 'cent' will be ignored on either call.
Outputs: rrd[]  - 6 element array containing the state vector of 'targ'
                  relative to 'cent'. The units are AU and AU/DAY. For
                  librations the units are RAD and RAD/DAY. For
                  nutations the first 4 elements of RRD[] are set to
                  nutations and rates, in RAD and RAD/DAY.
         inside - TRUE if 'jd' is within the ephemeris time span.
                  If not, 'inside' is set to FALSE.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void pleph(double jd, int targ, int cent, double *rrd, int *inside) {

  int i;
  static double fac;
  static int nemb, ipv, ncmp, lme;
  static int pfirst;
  static double jdtot;
  static double pv[6][13]={0.};
  static double embf[2], ve[2], jed[2];
  static int LList[12], LLst[13];
  static int L[2], tc[2];

  /*
```

```
    pv[] is a zero-offset 6x13 matrix. The column number (0-12) specifies
    a body number. The row number (0-5) specifies a component of the
    body's state vector x,y,z,xdot,ydot,zdot in that order.
*/

/* necessary for zero-offset arrays */
targ = targ - 1;
cent = cent - 1;
/*
    From here on, 'targ' and 'cent' are one less than their
    calling values.
*/

/* 1st time in, be sure ephemeris is initialized */
if (!pfirst) {
  pfirst = TRUE;
  ipv = 2; /* we want pos and vel */
  ephopn("");
  ve[0] = 1.0/(1.0+emrat);
  ve[1] = emrat*ve[0];

  jed[0] = 0.0;
  jed[1] = 0.0;

  embf[0] = -1.0;
  embf[1] =  1.0;
  for (i=0; i<12; i++) {
    LList[i] = 0;
  }
  L[0] = 0;
  L[1] = 0;
  tc[0] = 0;
  tc[1] = 0;
  for (i=0; i<13; i++) {
    LLst[i] = i;
    if (i ==  2) LLst[i] = 9;
    if (i == 11) LLst[i] = 10;
    if (i == 12) LLst[i] = 2;
  }
  fac = 0.0;
  nemb = 1;
}

/* Initialize jed[] for state() and set up component count */
jed[0] = jd;
jed[1] = 0.0;

jdtot = jed[0] + jed[1];

if ((jdtot >= SS[0]) && (jdtot <= SS[1])) {
  *inside = TRUE;
} else {
  *inside = FALSE;
  return;
}

ncmp = 3*ipv; /* total number of components */

/* check for nutation call */
if (targ == 13) {
  if (ipt[1][11] > 0) {
    LList[10] = ipv;
    state(jed, LList, pv, rrd);
```

```
    LList[10] = 0;
    return;
  } else {
    LogMsg(stderr, "pleph: no nutations on the ephemeris file.");
    exit(1);
  }
}

/* check for librations */
if (targ == 14) {
  if (lpt[1] > 0) {
    LList[11] = ipv;
    state(jed, LList, pv, rrd);
    LList[11] = 0;
    for (i=0; i<ncmp; i++) {
      rrd[i] = pv[i][10];
    }
    return;
  } else {
    LogMsg(stderr, "pleph: no librations on the ephemeris file.");
    exit(1);
  }
}

/* check for targ = cent */
if (targ == cent) {
  for (i=0; i<ncmp; i++) {
    rrd[i] = 0.0;
  }
  return;
}

/* force barycentric output by state() */
bsav = bary;
bary = TRUE;

/* set up proper entries in LList[] array for state() call */
tc[0] = targ;
tc[1] = cent;
lme = 0;

for (i=0; i<2; i++) {
  L[i] = LLst[tc[i]];
  if (L[i] < 10) LList[L[i]] = ipv;
  if (tc[i] == 2) {
    lme = 2;
    fac = -ve[0];
  }
  else if (tc[i] == 9) {
    lme = 9;
    fac = ve[1];
  }
  else if (tc[i] == 12) {
    nemb = i;
  }
}

if ((LList[9] == ipv) && (L[0] != L[1])) LList[2] = ipv - LList[2];

/* make call to state() */
state(jed, LList, pv, rrd);

/* case: Earth-to-Moon */
```

```
  if ((targ == 9) && (cent == 2)) {
    for (i=0; i<ncmp; i++) {
      rrd[i] = pv[i][9];
    }

    /* case: Moon-to-Earth */
  }
  else if ((targ == 2) && (cent == 9)) {
    for (i=0; i<ncmp; i++) {
      rrd[i] = -pv[i][9];
    }

    /* case: EMB-to-Moon or -Earth */
  }
  else if ((targ == 12 || cent == 12) && LList[9] == ipv) {
    for (i=0; i<ncmp; i++) {
      rrd[i] = pv[i][9]*fac*embf[nemb];
    }

    /* otherwise, get Earth or Moon vector and then get output vector */
  } else {
    for (i=0; i<ncmp; i++) {
      pv[i][10] = pvsun[i];
      pv[i][12] = pv[i][2];
      if (lme > 0) pv[i][lme] = pv[i][2]+fac*pv[i][9];
      rrd[i] = pv[i][targ] - pv[i][cent];
    }
  }

  /* clear state() body array and restore barycenter flag */
  LList[2] = 0;
  LList[L[0]] = 0;
  LList[L[1]] = 0;
  bary = bsav;
}

/*************************************************************
Name:    Pol2Rec
Purpose: Function to convert a polar state vector into a cartesian
         state vector.
         NOTE:  THIS ROUTINE EXPECTS THE POLAR VELOCITY VECTOR TO BE
         THE TOTAL VELOCITY CORRECTED FOR THE EFFECT OF LATITUDE.
Inputs:  a[] - Zero-offset polar state vector.
Outputs: b[] - Zero-offset cartesian state vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Pol2Rec(double a[], double b[]) {

  double lambda, beta, R, v_lambda, v_beta, v_r;
  double lambda_dot, beta_dot, r_dot, CosL, SinL, CosB, SinB;

  lambda   = a[0];
  beta     = a[1];
  R        = a[2];
  v_lambda = a[3];
  v_beta   = a[4];
  v_r      = a[5];

  /* separate the angluar derivatives from the total velocity components */
  CosL = cos(lambda);
  SinL = sin(lambda);
```

```
  CosB = cos(beta);
  SinB = sin(beta);

  lambda_dot = v_lambda / (R * CosB);
  beta_dot = v_beta / R;
  r_dot = v_r;

  /* position vector components */
  b[0] = R * CosL * CosB;
  b[1] = R * SinL * CosB;
  b[2] = R * SinB;

  /* velocity vector components */
  b[3] = r_dot * CosL * CosB - R * lambda_dot * SinL * CosB -
    R * beta_dot * CosL * SinB;
  b[4] = r_dot * SinL * CosB + R * lambda_dot * CosL * CosB -
    R * beta_dot * SinL * SinB;
  b[5] = r_dot * SinB + R * beta_dot * CosB;
}

/*************************************************************
Name:    PrecessElements
Purpose: Subprogram to transform angular orbital elements
         from equinox eqnx1 to equinox eqnx2.
Inputs:  eqnx1   - String containing initial
                   eqninox.  e.g. B1950 or 2415020.5.
         element - Array containing elements
                   referred to eqnx1 as follows:
                   element[0] = inclination.
                   element[1] = long. of asc. node.
                   element[2] = arg. of peri.
         eqnx2   - String containing final
                   eqninox.  e.g. J2000 or 2451545.
Outputs: element[] - Array containing elements referred to eqnx2.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void PrecessElements(char *eqnx1, double *element, char *eqnx2) {

  double  jd1, jd2, tt, t, tt2, t2, t3, tmp;
  double  SmallPI, LargePI, pa, cosi, sini, sinisinn, sinicosn;
  double  newi, newnode, sindwsini, cosdwsini, dw, newargp;

  ucase(eqnx1);
  ucase(eqnx2);
  if ((eqnx1[0] == 'B') || (eqnx1[0] == 'J')) {
    Epoch2JED(eqnx1, &jd1);
  } else {
    jd1 = atof(eqnx1);
  }

  if ((eqnx2[0] == 'B') || (eqnx2[0] == 'J')) {
    Epoch2JED(eqnx2, &jd2);
  } else {
    jd2 = atof(eqnx2);
  }

  tt = (jd1 - 2451545.0) / 365250.0;
  t = (jd2 - jd1) / 365250.0;
  tt2 = tt * tt;
  t2 = t * t;
  t3 = t2 * t;
```

```
  /* Compute precessional quantities */
  SmallPI = (470.029 - 6.603 * tt + 0.598 * tt2) * t +
    (-3.302 + 0.598 * tt) * t2 + 0.06 * t3;
  LargePI = 174.8763838888889 + (32894.789 * tt) / 3600.0 +
    (60.622 * tt2) / 3600.0 + ((-8698.089 - 50.491 * tt) * t) /
    3600.0 + (3.536 * t2) / 3600.0;
  pa = (50290.966 + 222.226 * tt - 0.042 * tt2) * t +
    (111.113 - 0.042 * tt) * t2 - 0.006 * t3;
  SmallPI = SmallPI * A2R;
  LargePI = LargePI * D2R;
  pa = pa * A2R;

  /* Compute new inclination and node */
  cosi = cos(element[0]) * cos(SmallPI) + sin(element[0]) *
    sin(SmallPI) * cos(LargePI - element[1]);

  sinisinn = sin(element[0]) * sin(LargePI - element[1]);
  sinicosn = -sin(SmallPI) * cos(element[0]) + cos(SmallPI) *
    sin(element[0]) * cos(LargePI - element[1]);

  sini = sqrt(sinisinn * sinisinn + sinicosn * sinicosn);
  newi = atan2(sini, cosi);
  if (newi < 0.0)
    newi += TWOPI;

  tmp = atan2(sinisinn, sinicosn);
  if (tmp < 0.0)
    tmp += TWOPI;
  newnode = pa + LargePI - tmp;
  newnode = amodulo(newnode, TWOPI);

  /* Compute new argument of perihelion */
  sindwsini = sin(SmallPI) * sin(LargePI - element[1]);
  cosdwsini = sin(element[0]) * cos(SmallPI) - cos(element[0]) *
    sin(SmallPI) * cos(LargePI - element[1]);
  dw = atan2(sindwsini, cosdwsini);
  if (dw < 0.0)
    dw += TWOPI;

  newargp = element[2] + dw;
  newargp = amodulo(newargp, TWOPI);

  /* Put new elements in element[] array */
  element[0] = newi;
  element[1] = newnode;
  element[2] = newargp;
}

/*************************************************************
Name:    QRotate
Purpose: Function to perform a matrix rotation of a state vector through
         a given angle about a desired axis.  A right-handed orthogonal
         coordinate system is assumed, and a 6X6 Q-matrix is used.
         NOTE: The vin[] and vout[] can have the same name in
         the calling program.
Inputs:  vin    - Zero-offset input state vector.
         axis   - 1 for x-axis,
                  2 for y-axis,
                  3 for z-axis.
         phi    - Rotation angle in radians.
         phidot - Derivative of phi.
         s      - 0 for inertial to inertial,
```

```
                1 for inertial to rotating.
Outputs: vout[] - Zero-offset transformed state vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void QRotate(double vin[], int axis, double phi, double phidot,
  int s, double vout[]) {

  int i;
  double temp[6];
  DMatrix QMatrix;

  QMatrix = createDMatrix(6);

  GetQMatrix(phi, phidot, axis, s, QMatrix);

  for (i=0; i<6; i++) {
    temp[i] = vin[i];
  }

  MatXVec(QMatrix, temp, vout, 6, 6);

  freeDMatrix(QMatrix, 6);
}

/*************************************************************
Name:    RayBend
Purpose: Function to correct an input vector for relativistic light
         deflection due to the Sun's gravity field.
Inputs:  earth_hel[] - Zero-offset heliocentric state vector of Earth.
         body_geo[]  - Zero-offset geocentric state vector of object.
         body_hel[]  - Zero-offset heliocentric state vector of object.
Outputs: p1[] - Zero-offset geocentric state vector of object
                corrected for light deflection.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void RayBend(double earth_hel[], double body_geo[], double body_hel[],
  double p1[6]) {

  double ue[3], up[3], uq[3], E[3], Edot[3], P[3], Pdot[3], Q[3], Qdot[3];
  double magE, magP, pdotq, edotp, qdote;
  int i;

  /* extract the pos. portions of the state vectors */
  SplitStateVector(earth_hel, E, Edot);
  SplitStateVector( body_geo, P, Pdot);
  SplitStateVector( body_hel, Q, Qdot);

  /* form unit vectors */
  Uvector(E, ue);
  Uvector(P, up);
  Uvector(Q, uq);

  /* form dot products and other quantities */
  magE = Vecmag(E);
  magP = Vecmag(P);
  Vdot(3, up, uq, &pdotq);
  Vdot(3, ue, up, &edotp);
  Vdot(3, uq, ue, &qdote);
```

```
  for (i=0; i<3; i++) {
    p1[i] = up[i] + (MUC / magE) * (pdotq * ue[i] - edotp * uq[i])
      / (1.0 + qdote);
    /* make p1[] a non-unit vector */
    p1[i] = magP * p1[i];
    p1[i+3] = body_geo[i+3];
  }
}

/*************************************************************
Name:    Rec2Pol
Purpose: Subprogram to convert a cartesian state vector into a
         polar state vector. NOTE: THE POLAR VELOCITY VECTOR IS
         THE TOTAL VELOCITY, CORRECTED FOR THE EFFECT OF LATITUDE.
Inputs:  a[] - Zero-offset cartesian state vector.
Outputs: b[] - Zero-offset polar state vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Rec2Pol(double a[], double b[]) {

  double x, y, z, x_dot, y_dot, z_dot,
    rho, r, lambda, beta, lambda_dot,
    beta_dot, r_dot;

  x = a[0];
  y = a[1];
  z = a[2];
  x_dot = a[3];
  y_dot = a[4];
  z_dot = a[5];

  rho = sqrt(x * x + y * y);
    r = sqrt(rho * rho + z * z);

  lambda = atan2(y, x);
  if (lambda < 0.0)
    lambda += TWOPI;

  beta   = atan2(z, rho);
  if (beta < 0.0)
    beta += TWOPI;

  if (z < 0) {
    beta = beta - TWOPI;
  }

  if (rho == 0) {
    lambda_dot = 0.0;
    beta_dot = 0.0;
  } else {
    lambda_dot = (x*y_dot-y*x_dot) / (rho*rho);
    beta_dot = (z_dot*rho*rho-z*(x*x_dot+y*y_dot))/(r*r*rho);
  }

  r_dot = (x * x_dot + y * y_dot + z * z_dot) / r;

  /* position vector components */
  b[0] = lambda;
  if (b[0] >= TWOPI)
    b[0] = b[0] - TWOPI;
```

```
  b[1] = beta;
  b[2] = r;
  /* total velocity vector components */
  b[3] = r * lambda_dot * cos(beta);
  b[4] = r * beta_dot;
  b[5] = r_dot;
}

/*************************************************************
Name:    Reduce
Purpose: Function for reducing source planetary or solar ephemerides
         to apparent, topocentric, virtual, local, or astrometric
         place. The processes are rigorous and include all corrections.
         This function is intended for use with barycentric source
         ephemerides such as DE200.
Inputs:  jed        - Desired time of reduction.
         body       - Body number (99 for stellar reduction).
         place      - 1 for apparent place, 2 for topocentric place,
                      3 for virtual place, 4 for local place,
                      5 for astrometric place.
         StarData[] - Array containing stellar data.
Outputs: p3[] - Array containing the requested state vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Reduce(double jed, int body, int place, double StarData[],
  double p3[]) {

  int i;
  double Ltime;
  double earth_ssb[6], earth_hel[6], body_geo[6];
  double body_hel[6], obsr_geo[6];
  double eb[6], ebdot[6], p1[6], p2[6];
  double zeta,z,theta,zetadot,zdot,thetadot;
  double dpsi,deps,dpsidot,depsdot;
  double TrueEps,MeanEps,TrueEpsDot,MeanEpsDot;

  if ((body == 3) || (body == 12) || (body == 13) ||
    (body == 14) || (body == 15)) {
    LogMsg(stderr,
      "Reduce: can't perform reduction for specified body.\n");
    exit(1);
  }

  /* Earth's barycentric state vector */
  GetStateVector(jed, 3, 12, 1, StateVector);
  for (i=0; i<6; i++) {
    earth_ssb[i] = StateVector[3-1][12-1][1-1][i];
  }

  /* Earth's heliocentric state vector */
  GetStateVector(jed, 3, 11, 1, StateVector);
  for (i=0; i<6; i++) {
    earth_hel[i] = StateVector[3-1][11-1][1-1][i];
  }

  if ((place == 2) || (place == 4)) {
    /* Compute geocentric state vector of observer */
    GeocenObs(jed, obsr_geo);
    /* Translate origin from geocenter to topocenter */
    for (i=0; i<6; i++) {
      earth_ssb[i] = earth_ssb[i] + obsr_geo[i];
```

```
    }
  }

  /*
      Compute geo/topocentric state vector of object corrected
      for light time.
  */
  LightTime(jed, body, earth_ssb, earth_hel, StarData, body_geo,
    body_hel, &Ltime);

  if (place == 5) {
    /* Compute the astrometric place */
    for (i=0; i<6; i++) {
      p3[i] = body_geo[i];
    }
  }

  if (place < 5) {
    /* Perform correction for relativistic light deflection */
    RayBend(earth_hel, body_geo, body_hel, p1);

    /* Perform correction for aberration */
    SplitStateVector(earth_ssb, eb, ebdot);
    Aberrate(p1, ebdot, p2);

    if (place < 3) {
      /* Correction for precession and nutation from J2000.0 */
      GetPrecessParams(J2000, jed, &zeta, &z, &theta, &zetadot,
        &zdot, &thetadot);
      GetDpsiDeps(jed, &dpsi, &deps, &dpsidot, &depsdot);
      Obliquity(J2000, jed, 0, &MeanEps, &MeanEpsDot);
      Obliquity(J2000, jed, 1, &TrueEps, &TrueEpsDot);

      /* First correct for precession */
      QRotate(p2, 3, -zeta, -zetadot, 1, p3);
      QRotate(p3, 2, theta, thetadot, 1, p3);
      QRotate(p3, 3, -z, -zdot, 1, p3);

      /* Now correct for nutation */
      QRotate(p3, 1, MeanEps, MeanEpsDot, 1, p3);
      QRotate(p3, 3, -dpsi, -dpsidot, 1, p3);
      QRotate(p3, 1, -TrueEps, -TrueEpsDot, 1, p3);
    } else {
      for (i=0; i<6; i++) {
        p3[i] = p2[i];
      }
    }
  }

  return;
}

/*************************************************************************
Name:    Refract
Purpose: Subprogram to correct right ascension and declination
         for atmospheric refraction.
         Reference:  Taff. Computational Spherical Astronomy, pp. 78-80.
Inputs:  ra1   - Uncorrected RA and DEC in radians.
         dec1  - Uncorrected RA and DEC in radians.
         lha   - Local apparent hour angle in radians.
         temp  - Temperature in deg F.
         press - Air pressure in inches of Hg.
Outputs: ra2  - Corrected RA and DEC in radians.
```

```
         dec2 - Corrected RA and DEC in radians.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
**********************************************************************/
void Refract(double ra1, double dec1, double lha, double temp,
  double press, double *ra2, double *dec2) {

  double cosz, z, r1, r2, r, tanzr, rr, diff, rlocal, denom;

  cosz = sin(obsr_lat) * sin(dec1) + cos(obsr_lat) * cos(dec1) * cos(lha);
  z = acos(cosz);

  r1 = 58.294 * A2R;
  r2 = -0.0668 * A2R;

  r = 0.0;
  do {
    tanzr = tan(z - r);
    rr = r1 * tanzr + r2 * tanzr * tanzr * tanzr;
    printf("%f\n", rr * R2A);
    diff = fabs(rr - r);
    r = rr;
  } while (diff > 1e-15);

  rlocal = (17.0 * press) * r / (460.0 + temp);

  denom = 1.0 / (cos(dec1) * sin(z));
  *dec2 = dec1 + (rlocal * (sin(obsr_lat) - sin(dec1) * cos(z))) * denom;
  denom = 1.0 / (cos(*dec2) * sin(z));
  *ra2 = ra1 + (rlocal * cos(obsr_lat) * sin(lha)) * denom;
}

/*************************************************************
Name:    RotMat
Purpose: Function to compute the 3X3 rotation matrix and its partial
         derivative for a rotation about the Ith coordinate axis
         through an angle phi.
Inputs:  axis - 1 for x-axis,
                2 for y-axis,
                3 for z-axis.
         phi  - Rotation angle in radians.
Outputs: r[]  - Zero-offset 3X3 rotation matrix.
         dr[] - Zero-offset 3X3 partial derivative of the matrix r[].
Returns: Nothing.
Status:  Finished.
Errors:  None known.
**************************************************************/
void RotMat(int axis, double phi, DMatrix r, DMatrix dr) {

  double cosphi, sinphi;

  cosphi = cos(phi);
  sinphi = sin(phi);

  switch (axis) {
    case 1:  /* rotate about x-axis */
      r[0][0] = 1.0;
      r[0][1] = 0.0;
      r[0][2] = 0.0;
      r[1][0] = 0.0;
      r[1][1] =  cosphi;
      r[1][2] =  sinphi;
```

```
      r[2][0] = 0.0;
      r[2][1] = -sinphi;
      r[2][2] =  cosphi;
      dr[0][0] = 0.0;
      dr[0][1] = 0.0;
      dr[0][2] = 0.0;
      dr[1][0] = 0.0;
      dr[1][1] = -sinphi;
      dr[1][2] =  cosphi;
      dr[2][0] = 0.0;
      dr[2][1] = -cosphi;
      dr[2][2] = -sinphi;
      break;
    case 2:  /* rotate about y-axis */
      r[0][0] =  cosphi;
      r[0][1] = 0.0;
      r[0][2] = -sinphi;
      r[1][0] = 0.0;
      r[1][1] = 1.0;
      r[1][2] = 0.0;
      r[2][0] =  sinphi;
      r[2][1] = 0.0;
      r[2][2] =  cosphi;
      dr[0][0] = -sinphi;
      dr[0][1] = 0.0;
      dr[0][2] = -cosphi;
      dr[1][0] = 0.0;
      dr[1][1] = 0.0;
      dr[1][2] = 0.0;
      dr[2][0] =  cosphi;
      dr[2][1] = 0.0;
      dr[2][2] = -sinphi;
      break;
    case 3:  /* rotate about z-axis */
      r[0][0] =  cosphi;
      r[0][1] =  sinphi;
      r[0][2] = 0.0;
      r[1][0] = -sinphi;
      r[1][1] =  cosphi;
      r[1][2] = 0.0;
      r[2][0] = 0.0;
      r[2][1] = 0.0;
      r[2][2] = 1.0;
      dr[0][0] = -sinphi;
      dr[0][1] =  cosphi;
      dr[0][2] = 0.0;
      dr[1][0] = -cosphi;
      dr[1][1] = -sinphi;
      dr[1][2] = 0.0;
      dr[2][0] = 0.0;
      dr[2][1] = 0.0;
      dr[2][2] = 0.0;
      break;
    default:
      LogMsg(stderr,"RotMat: axis not valid.\n");
      exit(1);
  }
}

/*****************************************************************
Name:    RST_Interpolate
Purpose: Interpolation routine used by RST.
Inputs:  c      - flag
```

```
         z0     - The "standard" zenith distance for the object
                  at rise or set.  This quantity has different
                  values for different objects according to the
                  following table:
                  z0 = 90d 50'          Sun.
                  z0 = 90d 34' + s - pi Moon.
                  z0 = 90d 34'          stars and planets.
                  z0 = 108d             astronomical twilight.
                  z0 = 102d         nautical twilight.
                  z0 =  96d             civil twilight.
                  z0 should be given by the program that calls RST.
                  z0 is expressed in radians.
         oldm   - intermediate value
         gast0  - GAST at 0h
         deltat - TDT-UT1 in seconds of time.
         ra[]   - Array containing the object's apparent right
                  ascension at times jed-1, jed, and jed+1.
         dec[]  - Array containing the object's apparent
                  declination at times jed-1, jed, and jed+1.
         rx, dx - RA and DEC of object on day x
Outputs: newm   - intermediate value
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
static void RST_Interpolate(int c, double z0, double oldm, double gast0,
  double deltat, double *ra, double *dec, double r1, double r2, double r3,
  double d1, double d2, double d3, double *newm) {

  double alpha, dm, h, gast, delta, alt, n;

  *newm = oldm;
  do {
    gast = gast0 + 6.300388093 * (*newm);
    gast = amodulo(gast, TWOPI);
    n = *newm + deltat / 86400.0;
    alpha = ra[1] + 0.5 * n * (r1 + r2 + n * r3);
    alpha = amodulo(alpha, TWOPI);
    delta = dec[1] + 0.5 * n * (d1 + d2 + n * d3);
    h = gast + obsr_lon - alpha;
    alt = asin(sin(delta) * sin(obsr_lat) + cos(delta) * cos(obsr_lat) *
cos(h));
    if (c == 0) {
      /* h must satisfy -PI <= h <= PI */
      h = amodulo(h, TWOPI);
      if (h > PI) {
        h = h - TWOPI;
      }
      dm = -h / TWOPI;
    } else {
      dm = (alt - PIDIV2 + z0) / (TWOPI * cos(delta) * cos(obsr_lat) *
sin(h));
    }
    *newm = (*newm) + dm;
  } while (fabs(dm) >= 1e-15);
}

/*****************************************************************
Name:    RST
Purpose: Subprogram to compute the times of rise, set, and transit for
         any object given the observer's location and arrays containing
         the APPARENT right ascension and declination of the object for
         three dates centered on the input JED.  The algorithm is
```

```
         completely rigorous,  and takes into account atmospheric
         refraction and the object's sidereal motion in the intervals
         between rising, setting, and transiting.  The algorithm is
         explained in Chapter 42 of Meeus' ASTRONOMICAL FORMULAE FOR
         CALCULATORS. With the appropriate values for z0, this routine
         can also be used to compute the times of civil, nautical, or
         astronomical twilight. Note that the times are on the UT1 scale!
         Reference:  Meeus.  ASTRONOMICAL FORMULAE FOR CALCULATORS,
         4TH ED., Chapter 42.
Inputs:  jed    - Julian day number at 0h UT1
         ra[]   - Array containing the object's apparent right
                  ascension at times jed-1, jed, and jed+1.
         dec[]  - Array containing the object's apparent
                  declination at times jed-1, jed, and jed+1.
         z0     - The "standard" zenith distance for the object
                  at rise or set.  This quantity has different
                  values for different objects according to the
                  following table:
                  z0 = 90d 50'          Sun.
                  z0 = 90d 34' + s - pi Moon.
                  z0 = 90d 34'          stars and planets.
                  z0 = 108d             astronomical twilight.
                  z0 = 102d         nautical twilight.
                  z0 =  96d             civil twilight.
                  z0 should be given by the program that calls RST.
                  z0 is expressed in radians.
         deltat - TDT-UT1 in seconds of time.
Outputs: ris[] - String containing rise time in
                 form hh.mm or an appropriate symbol.
         trn[] - String containing transit time
                 in form hh.mm or an appropriate symbol.
         set[] - String containing set time in
                 form hh.mm or an appropriate symbol.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void RST(double jed, double *ra, double *dec, double z0, double deltat,
  char *ris, char *trn, char *set) {

  int rsflag, c;
  double h0, cosh0, newm, oldm, m, m0, m1, m2;
  double ristime, settime, trntime, gast0;
  double d1, d2, d3, r1, r2, r3;

  /* Make sure the ra[]'s are in continuous order */
  if ((ra[1] < ra[0]) && (ra[2] > ra[1])) {
    ra[1] = ra[1] + TWOPI;
    ra[2] = ra[2] + TWOPI;
  }
  else if ((ra[1] > ra[0]) && (ra[2] < ra[1])) {
    ra[2] = ra[2] + TWOPI;
  }

  r1 = ra[1] - ra[0];
  r2 = ra[2] - ra[1];
  r3 = r2 - r1;
  d1 = dec[1] - dec[0];
  d2 = dec[2] - dec[1];
  d3 = d2 - d1;

  rsflag = -1;
```

```
  cosh0 = (cos(z0) - sin(obsr_lat) * sin(dec[1])) / (cos(obsr_lat) *
cos(dec[1]));

  if (cosh0 < -1.0) {
    /* Object is circumpolar */
    strcpy(ris, "***********");
    if ((z0 * R2D) >= 96.0) {
      strcpy(set, "**BRIGHT***");
    } else {
      strcpy(set, "**NO SET***");
    }
    rsflag = 0;
  }
  else if (cosh0 > 1.0) {
    /* Object never rises */
    if ((z0 * R2D) >= 96.0) {
      strcpy(ris, "---DARK----");
    } else {
      strcpy(ris, "--NO RISE--");
    }
    strcpy(set, "-----------");
    rsflag = 0;
  }

  GetGST(jed, 1, &gast0);

  m0 = (ra[1] - obsr_lon - gast0) / TWOPI;
  m0 = amodulo(m0, 1.0);

  if (rsflag) {
    h0 = acos(cosh0);
    h0 = amodulo(h0, PI);
    m1 = m0 - h0 / TWOPI;
    m1 = amodulo(m1, 1.0);
    m2 = m0 + h0 / TWOPI;
    m2 = amodulo(m2, 1.0);

    /* Rising */
    oldm = m1;
    c = 1;
    RST_Interpolate(c, z0, oldm, gast0, deltat, ra, dec, r1,
      r2, r3, d1, d2, d3, &newm);
    m = newm;
    ristime = 24.0 * m;
    if (ristime > 24.0) {
      ristime = ristime - 24.0;
      /* Event occurs the following day */
      FmtDms(ristime, 0, 1, ris);
      strcat(ris, "(f)");
    }
    else if (ristime < 0.0) {
      ristime = ristime + 24.0;
      /* Event occurs the previous day */
      FmtDms(ristime, 0, 1, ris);
      strcat(ris, "(p)");
    } else {
      FmtDms(ristime, 0, 1, ris);
    }

    /* Setting */
    oldm = m2;
    c = 1;
    RST_Interpolate(c, z0, oldm, gast0, deltat, ra, dec, r1,
```

```
      r2, r3, d1, d2, d3, &newm);
    m = newm;
    settime = 24.0 * m;
    if (settime > 24.0) {
      settime = settime - 24.0;
      FmtDms(settime, 0, 1, set);
      strcat(set, "(f)");
    }
    else if (settime < 0.0) {
      settime = settime + 24.0;
      FmtDms(settime, 0, 1, set);
      strcat(set, "(p)");
    } else {
      FmtDms(settime, 0, 1, set);
    }
  }

  /* Transiting */
  oldm = m0;
  c = 0;
  RST_Interpolate(c, z0, oldm, gast0, deltat, ra, dec, r1,
    r2, r3, d1, d2, d3, &newm);
  m = newm;
  trntime = 24.0 * m;
  if (trntime > 24.0) {
    trntime = trntime - 24.0;
    FmtDms(trntime, 0, 1, trn);
    strcat(trn, "(f)");
  }
  else if (trntime < 0.0) {
    trntime = trntime + 24.0;
    FmtDms(trntime, 0, 1, trn);
    strcat(trn, "(p)");
  } else {
    FmtDms(trntime, 0, 1, trn);
  }
}

/*****************************************************************
Name:    split
Purpose: Function to break a number into an integer part and a
         fractional part. For negative input numbers, 'ipart'
         contains the next more negative number and 'fpart' contains
         a positive fraction.
Inputs:  tt - Number to be split.
Outputs: ipart - Integer part.
         fpart - Fractional part.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void split(double tt, double *ipart, double *fpart) {

  *ipart = floor(tt);
  *fpart = tt - *ipart;

  if ((tt < 0) && (*fpart != 0)) {
    *ipart = *ipart - 1.0;
    *fpart = *fpart + 1.0;
  }
}

/*****************************************************************
```

```
Name:    SplitStateVector
Purpose: Subprogram to split a state vector into its position
         and velocity components.
Inputs:  pv[] - Zero-offset 6-d state vector.
Outputs: p[] - Zero-offset 3-d position vector.
         v[] - Zero-offset 3-d velocity vector.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
**************************************************************/
void SplitStateVector(double pv[], double p[], double v[]) {

  int i;

  for (i=0; i<3; i++) {
    p[i] = pv[i];
    v[i] = pv[i+3];
  }
}

/**************************************************************
Name:    state
Purpose: This subroutine reads and interpolates the JPL planetary
         ephemeris file.
Inputs:  jed[]   - 2 element array containing the JED epoch at which
                   interpolation is wanted. Any combination of
                   jed[0]+jed[1] which falls within the time span on
                   the file is a permissible epoch. For ease in
                   programming, the user may put the entire epoch in
                   jed[0] and set jed[1] = 0. For maximum accuracy,
                   set jed[0] = most recent midnight at or before
                   interpolation epoch and set jed[1] = fractional
                   part of a day elapsed between jed[0] and epoch.
                   As an alternative, it may prove convenient to set
                   jed[0] = some fixed epoch, such as start of
                   integration and jed[1] = elapsed interval between
                   interval between then and epoch.
         LList[] - 12 element array specifying what interpolation
                   is wanted for each of the bodies on the file.
                   LList[i] =0, no interpolation for body i,
                            =1, position only,
                            =2, position and velocity.
                   The designation of the astronomical bodies by i is:
                   i = 0: Mercury,
                     = 1: Venus,
                     = 2: Emb,
                     = 3: Mars,
                     = 4: Jupiter,
                     = 5: Saturn,
                     = 6: Uranus,
                     = 7: Neptune,
                     = 8: Pluto,
                     = 9: Moon (geocentric),
                     =10: Nutations in longitude and obliquity (if present),
                     =11: Lunar librations (if present).
Outputs: pv[]  - 6 x 13 array that will contain requested interpolated
                 quantities. the body specified by LList[i] will have its
                 state in the array starting at pv[0][i]. (On any given
                 call, only those words in pv[] which are affected by the
                 first 10 LList[] entries (and by LList[12] if librations
                 are on the file) are set. The rest of the pv[] array
                 is untouched.) the order of components starting in
                 pv[0][i] is: x,y,z,dx,dy,dz.
```

```
                All output vectors are referenced to the earth mean
                equator and equinox of epoch. The Moon state is always
                geocentric; the other nine states are either heliocentric
                or solar-system barycentric, depending on the setting of
                common flags (see below).

                Lunar librations, if in the data file, are put into
                pv[k][11] if LList[11] is 1 or 2.

        nut[] - 4-word array that will contain nutations and rates,
                depending on the setting of LList[10]. The order of
                quantities in nut[] is:
                dpsi     (nutation in longitude),
                depsilon (nutation in obliquity),
                dpsi dot,
                depsilon dot.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*****************************************************************/
void state(double *jed, int LList[], double pv[][13], double *nut) {
/*****************************************************************
* Other important variables:                                     *
*                                                                *
* km   Logical flag defining physical units of the output        *
*      states. km = TRUE,  units are km and km/sec,              *
*                 = FALSE, units are au and au/day.              *
*      default value = FALSE (km determines time unit            *
*      for nutations and librations. Angle unit is always radians.)
*                                                                *
* bary Logical flag defining output center.                      *
*      only the 9 planets are affected.                          *
*                    bary = TRUE  = center is SSB,               *
*                         = FALSE = center is Sun.               *
*           default value = FALSE.                               *
*                                                                *
* pvsun[] 6-word array containing the barycentric position and   *
*         velocity of the Sun.                                   *
*****************************************************************/

  static double t[2], jd[4], temp[6];
  static double dumpv[3][2]={{0.,0.},{0.,0.},{0.,0.}};
  static int sfirst;
  static long int nrl;

  static int buff, ncf, na, km = 0;
  static double aufac, s, ipart, fpart;

  int i, j, k, m;
  static long int nr;

  /* 1st time in, get pointer data, etc., from ephemeris file */
  if (!sfirst) {
    sfirst = TRUE;
    aufac = 1.0;
    nrl = 0L;
    ephopn("");
    if (km) {
      t[1] = SS[2]*86400.0;
    } else {
      t[1] = SS[2];
      aufac = 1.0 / au;
```

```
    }
  }

  /* main entry point -- check epoch and read right record */
  s = jed[0] - 0.5;
  split(s, &ipart, &fpart);
  jd[0] = ipart;
  jd[1] = fpart;
  split(jed[1], &ipart, &fpart);
  jd[2] = ipart;
  jd[3] = fpart;
  jd[0] = jd[0] + jd[2] + 0.5;
  jd[1] = jd[1] + jd[3];
  split(jd[1], &ipart, &fpart);
  jd[2] = ipart;
  jd[3] = fpart;
  jd[0] = jd[0] + jd[2];

  /* error return of epoch out of range */
  if ((jd[0] < SS[0]) || (jd[0]+jd[3]) > SS[1]) {
    LogMsg(stderr, "state: epoch out of range.\n");
    exit(1);
  }

  /* 'nr' is the byte index of the first coefficient */
  nr = (long) (floor((jd[0]-SS[0])/SS[2]));
  nr = LengthOfHeader + nr * BlockLength;
  /* use previous block if necessary */
  if (jd[0] == SS[1])
    nr = nr - BlockLength;
  if (nr < 1L) {
    LogMsg(stderr, "state: block not present.\n");
    exit(1);
  }

  /* calculate relative time in interval (0 <= t[0] <= 1) */
  t[0] = ((jd[0]-(((double)nr -
(double)(LengthOfHeader))/(double)BlockLength
    * SS[2] + SS[0])) + jd[3]) / SS[2];
  if (t[0] < 0.0)
    t[0] = t[0]+ 1.0;

  /* read correct record if not in core */
  if (nr != nrl) {
    nrl = nr;
    if (fseek(fpBinaryFile, (long) (nr), 0) != 0) {
      LogMsg(stderr, "state: fseek() failed.\n");
      exit(1);
    }
    k = 0;
    do {
      if (k < ncoeff) {
        fread(&tmpDouble, sizeof(double), 1, fpBinaryFile);
        convert_little_endian((char *) &tmpDouble, sizeof(double));
        db[k] = (double) tmpDouble;
      }
      k++;
    } while (!feof(fpBinaryFile) && k <= ncoeff);
  }

  /* interpolate SSBARY Sun */
  buff = ipt[0][10]-1; /* location of first coeff */
  ncf  = ipt[1][10]; /* number of coeffs per component */
```

```
na   = ipt[2][10]; /* number of sets of coeffs per 32day interval */

interp(buff, t, ncf, 3, na, 2, dumpv);

k = 0;
for (j=0; j<2; j++) {
  for (i=0; i<3; i++) {
    pvsun[k] = dumpv[i][j]*aufac;
    k++;
  }
}

/* check and interpolate whichever bodies are requested */
for (i=0; i<10; i++) {
  if (LList[i] <= 0)
    continue;
  if (ipt[1][i] <= 0) {
    errprt(i ,"th body requested - not on file.\n");
  }
  buff = ipt[0][i]-1; /* location of first coeff */
  ncf  = ipt[1][i]; /* number of coeffs per component */
  na   = ipt[2][i]; /* number of sets of coeffs per 32day interval */

  interp(buff, t, ncf, 3, na, LList[i], dumpv);

  /* need to re-map dumpv[1..3][1..2] --> temp[1..6] */
  k = 0;
  for (j=0; j<2; j++) {
    for (m=0; m<3; m++) {
      temp[k] = dumpv[m][j];
      k++;
    }
  }

  for (j=0; j<(LList[i]*3); j++) {
    if ((i <= 8) && (!bary)) {
      pv[j][i] = temp[j]*aufac-pvsun[j];
    } else {
      pv[j][i] = temp[j]*aufac;
    }
  }
}

/* do nutations if requested and if on file */
if ((LList[10] > 0) && (ipt[1][11] > 0)) {
  buff = ipt[0][11]-1; /* location of first coeff */
  ncf  = ipt[1][11]; /* number of coeffs per component */
  na   = ipt[2][11]; /* number of sets of coeffs per 32day interval */

  interp(buff, t, ncf, 2, na, LList[10], dumpv);

  /* need to re-map dumpv(1:3,1:2) --> temp(1:6) */
  k = 0;
  for (j=0; j<2; j++) {
    for (m=0; m<2; m++) {
      nut[k] = dumpv[m][j];
      k++;
    }
  }
  nut[4] = 0.0;
  nut[5] = 0.0;
}
```

```
  /* get librations if requested and if on file */
  if ((lpt[1] > 0) && (LList[11] > 0)) {
    buff = lpt[0]-1; /* location of first coeff */
    ncf  = lpt[1]; /* number of coeffs per component */
    na   = lpt[2]; /* number of sets of coeffs per 32day interval */

    interp(buff, t, ncf, 3, na, LList[11], dumpv);

    pv[0][10] = dumpv[0][0];
    pv[1][10] = dumpv[1][0];
    pv[2][10] = dumpv[2][0];
    pv[3][10] = dumpv[0][1];
    pv[4][10] = dumpv[1][1];
    pv[5][10] = dumpv[2][1];
  }
}

/*************************************************************
Name:    Stat2Elem
Purpose: Subprogram to transform the components of a state vector into
         the universal orbital element set at a given time.
         Reference: Mansfield. 1986. AIAA Paper 86-2269-CP.
Inputs:  posvel[] - State vector.
         mu       - Gravitational const.
         jed      - Time.
Outputs: uelement[] - Array of universal elements:
                      uelement[0] = q.
                      uelement[1] = e.
                      uelement[2] = i.
                      uelement[3] = node.
                      uelement[4] = arg.peri.
                      uelement[5] = T.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Stat2Elem(double *posvel, double mu, double jed, double *uelement) {

  int i;
  double r[3], rdot[3], L[3], wvec[3], u[3], v[3], pvec[3], qvec[3];
  double magr, magL, p, vsquared, alpha, chi, psi, sini, cosi, sinn, cosn;
  double rnudot, rrdot, magrdot, esinnu, ecosnu, sinnu, cosnu;
  double z, w, h, s, x, c3, tspp;

  /* Get pos. and vel. vectors from state vector */
  r[0] = posvel[0];
  r[1] = posvel[1];
  r[2] = posvel[2];
  rdot[0] = posvel[3];
  rdot[1] = posvel[4];
  rdot[2] = posvel[5];

  /* Compute magr */
  magr = Vecmag(r);

  /* Compute angular momentum vector */
  Vcross(r, rdot, L);

  /* Compute magL */
  magL = Vecmag(L);

  /* Compute wvec[] */
  Uvector(L, wvec);
```

```
/* Compute u[] */
Uvector(r, u);

/* Compute v[] */
Vcross(wvec, u, v);

/* Compute semilatus rectum */
p = magL * magL / mu;

/* Compute square of velocity */
Vdot(3, rdot, rdot, &vsquared);

/* Compute alpha */
alpha = 2.0 * mu / magr - vsquared;

/* Compute eccentricity */
uelement[1] = sqrt(1.0 - alpha * magL * magL / (mu * mu));

/* Compute perihelion distance */
uelement[0] = p / (1.0 + uelement[1]);

/* Compute node and inclination */
/* First compute chi and psi */
chi = wvec[0] / (1.0 + wvec[2]);
psi = -wvec[1] / (1.0 + wvec[2]);
/* Now get inclination */
sini = 2.0 * sqrt(chi * chi + psi * psi) / (1.0 + chi * chi + psi * psi);
cosi = (1.0 - chi * chi - psi * psi) / (1.0 + chi * chi + psi * psi);
uelement[2] = atan2(sini, cosi);
if (uelement[2] < 0.0)
  uelement[2] += TWOPI;

/* Now get node */
sinn = chi / sqrt(chi * chi + psi * psi);
cosn = psi / sqrt(chi * chi + psi * psi);
uelement[3] = atan2(sinn, cosn);
if (uelement[3] < 0.0)
  uelement[3] += TWOPI;

/* Compute arg. peri. */
/* Compute rnudot */
Vdot(3, rdot, v, &rnudot);
/* Compute magrdot */
Vdot(3, r, rdot, &rrdot);
magrdot = rrdot / magr;
esinnu = magrdot * sqrt(p / mu);
ecosnu = rnudot * sqrt(p / mu) - 1.0;
/* Proceed to compute arg. peri. */
z = esinnu / (uelement[1] + ecosnu);
sinnu = 2.0 * z / (1.0 + z * z);
cosnu = (1.0 - z * z) / (1.0 + z * z);
/* Get the pvec[] and qvec[] vectors */
for (i = 0; i < 3; i++) {
  pvec[i] = u[i] * cosnu - v[i] * sinnu;
  qvec[i] = u[i] * sinnu + v[i] * cosnu;
}

/* Finally compute arg. peri. */
uelement[4] = atan2(pvec[2], qvec[2]);
if (uelement[4] < 0.0)
  uelement[4] += TWOPI;
```

```
  /* Compute time of peri. passage */
  w = sqrt(uelement[0] / (mu * (1.0 + uelement[1]))) * z;
  h = alpha * w * w;
  s = 2.0 * ((((((h / 13.0 - 1.0 / 11.0) * h + 1.0 / 9.0) * h - 1.0 / 7.0)
    * h + 1.0 / 5.0) * h - 1.0 / 3.0) * h + 1.0) * w;
  /* Compute Stumpff functions */
  x = alpha * s * s;
  c3 = StumpffN(x, 3);
  tspp = uelement[0] * s + mu * uelement[1] * s * s * s * c3;
  uelement[5] = jed - tspp;
}
/*****************************************************************
Name:    StumpffN
Purpose: Function to compute the nth order Stumpff function for any x.
         Reference: Danby. FUN, pp. 172-174.
Inputs:  x      - Argument.
         Norder - Order desired.
Outputs: None.
Returns: nth order Stumpff function.
Status:  Finished.
Errors:  None known.
*****************************************************************/
double StumpffN(double x, int Norder) {

  int n = 0;
  double a, b, c0, c1, c2, c3;

  do {
    n++;
    x = x / 4.0;
  } while (fabs(x) > 0.1);

  a = (1.0 - x * (1.0 - x / 182.0) / 132.0);
  b = (1.0 - x * (1.0 - x * a / 90.0) / 56.0);
  c2 = (1.0 - x * (1.0 - x * b / 30.0) / 12.0) / 2.0;
  a = (1.0 - x * (1.0 - x / 210.0) / 156.0);
  b = (1.0 - x * (1.0 - x * a / 110.0) / 72.0);
  c3 = (1.0 - x * (1.0  - x * b / 42.0)/ 20.0) / 6.0;

  c1 = 1.0 - x * c3;
  c0 = 1.0 - x * c2;

  do {
    n--;
    c3 = (c2 + c0 * c3) / 4.0;
    c2 = c1 * c1 / 2.0;
    c1 = c0 * c1;
    c0 = 2.0 * c0 * c0 - 1.0;
    x = x * 4.0;
  } while (n > 0);

  switch (Norder) {
    case 0:
      return (c0);
      case 1:
      return (c1);
    case 2:
      return (c2);
      case 3:
      return (c3);
    default:
      LogMsg(stderr, "StumpffN: Norder not 1, 2, or 3\n");
      exit(1);
```

```
  }

  return(-999);
}

/*************************************************************
Name:    Transpose
Purpose: Function to compute the transpose of a matrix.
Inputs:  a[][] - Zero-offset matrix a (n rows by n columns).
Outputs: b[][] - Zero-offset matrix transpose (n rows by n columns).
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Transpose(DMatrix a, DMatrix b, int n) {

  int i, j;

  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      b[i][j] = a[j][i];
    }
  }
}

/*************************************************************
Name:    Uvector
Purpose: Unit vector subroutine.
Inputs:  a[] - Zero-offset column vector (3 rows by 1 column).
Outputs: unita[] - Zero-offset unit vector (3 rows by 1 column).
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Uvector(double a[], double unita[]) {

  double maga;
  int i;

  maga = sqrt(a[0]*a[0] + a[1]*a[1] + a[2]*a[2]);
  for (i=0; i<3; i++) {
    if (maga != 0) {
      unita[i] = a[i]/maga;
    } else {
      unita[i] = 0.0;
    }
  }
}

/*************************************************************
Name:    Vcross
Purpose: Vector cross product subroutine.
Inputs:  a - Zero-offset vector a (3 rows by 1 column).
         b - Zero-offset vector b (3 rows by 1 column).
Outputs: acrossb - a X b (3 rows by 1 column).
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Vcross(double a[], double b[], double acrossb[]) {

  acrossb[0] = a[1] * b[2] - a[2] * b[1];
  acrossb[1] = a[2] * b[0] - a[0] * b[2];
```

```
  acrossb[2] = a[0] * b[1] - a[1] * b[0];
}

/*************************************************************
Name:    Vdot
Purpose: Vector dot product function.
Inputs:  n   - Number of rows.
         a[] - Zero-offset column vector with N rows.
         b[] - Zero-offset column vector with N rows.
Outputs: adotb - Dot product of a and b.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*************************************************************/
void Vdot(int n, double a[], double b[], double *adotb) {

  int i;

  *adotb = 0.0;
  for (i = 0; i<n; i++) {
    *adotb += a[i] * b[i];
  }
}

/*************************************************************
Name:    Vecmag
Purpose: Vector magnitude function.
Inputs:  a[] - Zero-offset column vector (3 rows by 1 column).
Outputs: None.
Returns: Magnitude of vector.
Status:  Finished.
Errors:  None known.
*************************************************************/
double Vecmag (double a[]) {

  double x;

  x = sqrt(a[0] * a[0] + a[1] * a[1] + a[2] * a[2]);

  return (x);
}

/* End Of File - astrolib.c ***************************************/
```

Lampiran 7.
Source code file*: asc2eph.c*

```
/*****************************************************************
Project:  asc2eph
Filename: asc2eph.c
Author:   Ported to C by Joe Heafner and Varian Swieter.
Purpose:  This program converts a JPL ephemeris ASCII data file
          into a binary data file. Note that the binary data files
          created with this program are not identical to those
          created with the JPL ephemeris software.
Thanks to Charles Gamble for extensive modifications.
*****************************************************************/

/* Header Files ******************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <errno.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/param.h>
#include "astrolib.h"
#include "support.h"

/* Function Prototypes ***********************************************/
void ParseCmdLine(int argc, char *argv[], char *InFile, char *OutFile);
void PrintBanner(void);
void PrintUsage(void);
void ERRPRT(int group, char *message);
void NXTGRP(FILE *fptr, char *outstr);

/* Globals ***********************************************************/
char szVersion[] = "ASC2EPH v1.060300c";
char szLogFile[] = "asc2eph.log";

#define MAX_KSIZE       2048
#define MAX_TTL         66
#define MAX_CNAME       7
#define DEFAULT_OUTPUT  "JPLEPH"

int DeletefpLogFile = FALSE,
    PromptForDates  = FALSE,
    HeaderIncluded  = FALSE,
    FIRST           = TRUE,
    KSIZE, NCON, tmpInt;

FILE *fpInputFile, *fpOutputFile, *fpHFile;

short IPT[3][13], LPT[3], NUMDE, tmpShort;

char InFile[MAX_NAME_SIZE+1]  = "",
     OutFile[MAX_NAME_SIZE+1] = "",
     HFile[MAX_NAME_SIZE+1]   = "",
     HEADER[MAX_NAME_SIZE+1]  = "";

double T1, T2, tmpDouble;

char Body[13][4] = { "MER", "VEN", "EMB", "MAR",
                     "JUP", "SAT", "URA", "NEP",
                     "PLU", "MOO", "SUN", "NUT",
```

```
                    "LIB" };

/*****************************************************************
Name:    main
Purpose: Main routine for asc2eph.
Inputs:  argc - Number of command-line arguments.
         argv - Pointer to array of command-line arguments.
Outputs: None.
Returns: 0 if execution successful.
Status:  Finished.
Errors:  None known.
*****************************************************************/
int main(int argc, char *argv[]) {

  char Progress[50] = "Searching for first requested record ",
       TTL[3][MAX_TTL] = { "", "", "" },
       CNAM[400][MAX_CNAME], right_buffer[4], line[1024];

  double DB[MAX_KSIZE], SS[3], CVAL[400], AU, EMRAT, DB2Z = 0;

  int DB_size, i, k, jrow, jcol, N, NROUT, NRW, NCOEFF,
      exponent, LeftOver, NumZeros, LengthOfHeader,
      LengthOfRecord, loop;

  long int fpeof, LengthOfFile;

  /* Open the log file */
  if (LogOpen(szLogFile) == FALSE)
    {
      fprintf(stdout, "Could not open log file '%s': %s\n\n",
        szLogFile, strerror(errno));
      exit(1); /* Exit with an error code */
    }

  /* Write a fingerprint to the screen and to log file */
  PrintBanner();

  ParseCmdLine(argc, argv, InFile, OutFile);

  /*
      If you don't want all the data, set T1 and T2 to the begin
      and end times of the span you desire. Units are JED.
  */
  if (PromptForDates) {
    do {
      do {
        fprintf(stdout, "Enter start JED: ");
        fflush(stdout);
        fgets(line, sizeof(line), stdin);
      } while (sscanf(line, "%lf", &T1) != 1);

      do {
        fprintf(stdout, "Enter final JED: ");
        fflush(stdout);
        fgets(line, sizeof(line), stdin);
      } while (sscanf(line, "%lf", &T2) != 1);

      if (T2 < T1) {
        fprintf(stdout, "Final JED must be later than start JED!\n");
      }
    } while (T2 < T1);
```

```
  if (T1 == T2) {
    fprintf(stdout,
      "Start and final JED's cannot be the same. Using Defaults.\n");
    T1 = 0.0;
    T2 = 9999999.0;
  }
} else {
  T1 = 0.0;
  T2 = 9999999.0;
}

if (strlen(InFile) == 0) {
  /* Prompt user for file name */
  do {
    fprintf(stdout, "File name to convert (XXXX to end): ");
    fflush(stdout);
    fgets(InFile, MAX_NAME_SIZE+1, stdin);

    /* Remove whitespace from either end */
    Trim(InFile);
  } while (strlen(InFile) == 0);
}

/* We have a filename now */
/* ucase(InFile); */  /* May not be used for OS/2 and UNIX compiles */
                      /* if you want mixed-case filenames.          */

/* Test for user exit request */
if (strcmp(InFile, "XXXX") == 0) {
  LogMsg(stdout, "\nOK\n");
  LogClose();
  remove(szLogFile);
  return (0);
}

/* Test to see if filename exists */
if (!fexist(InFile)) {
  LogMsg(stdout, "Requested input file does not exist.\n");
  LogMsg(stdout, "Specify another file or move to another directory.\n");
  LogMsg(stdout, "\nOK\n");
  LogClose();
  exit(1);
}

if (strlen(OutFile) == 0) strcpy(OutFile, DEFAULT_OUTPUT);

LogMsg(stdout, "Input will be read from %s\n", InFile);
LogMsg(stdout, "Output will be written to %s\n", OutFile);

/* Form the header file name */
if (!HeaderIncluded) {
  right(InFile, 3, right_buffer);
  strcpy(HFile, "header.");
  strcat(HFile, right_buffer);

  if (!fexist(HFile)) {
    LogMsg(stdout, "Header file not present.\n");
    LogMsg(stdout, "\nOK\n");
    LogClose();
    exit(1);
  }
}
```

```
if (HeaderIncluded) strcpy(HFile, InFile);

/* Open the header file for input */
if ((fpHFile = fopen(HFile, "r")) == NULL) {
  LogMsg(stdout, "Header file not present.\n");
  LogMsg(stdout, "\nOK\n");
  LogClose();
  exit(1);
}

/*
    Get KSIZE, throwing away 'KSIZE=', 'NCOEFF=', and
    NCOEFF decimal constant.
*/
if (fscanf(fpHFile, " %*s %d %*s %*d", &KSIZE) != 1) {
  LogMsg(stderr, "Error reading KSIZE\n");
  LogClose();
  fclose(fpHFile);
  exit(1);
}
LogMsg(stdout, "KSIZE=  %d\n", KSIZE);

/*
    Set max size of coeficient array for this particular
    ephemeris file. If KSIZE is an odd number, then
    DB_size = (KSIZE / 2) + (KSIZE % 2).
*/
DB_size = KSIZE / 2;

if (DB_size > MAX_KSIZE) {
  /* Array is not large enough */
  LogMsg(stdout,
    "Adjust #define MAX_KSIZE to larger value & recompile.\n");
  LogMsg(stdout, "\nOK\n");
  LogClose();
  exit(1);
}

/* Initialize DB through DB_size */
NXTGRP(fpHFile, HEADER);
if (strcmp(HEADER, "GROUP   1010") != 0) {
  ERRPRT(1010, "NOT HEADER");
}

for (i = 0; i < 3; i++) {
  fscanf(fpHFile, " %65[^\n]", TTL[i]);
  LogMsg(stdout, "%s\n",   TTL[i]);
}

/* Read start, end and record span (GROUP 1030) */
NXTGRP(fpHFile, HEADER);
if (strcmp(HEADER, "GROUP   1030") != 0) {
  ERRPRT(1030, "NOT HEADER");
}

/* Read in values of ss[0], ss[1], ss[2] */
for (i = 0; i < 3; i++) {
  if (fscanf(fpHFile, " %lf", &SS[i]) != 1) {
    LogMsg(stderr, "Error reading SS[%d].\n", i);
    LogClose();
    fclose(fpHFile);
    exit(1);
```

```
  }
}

/* Read number of constants and names of constants (GROUP 1040/4) */
NXTGRP(fpHFile, HEADER);
if (strcmp(HEADER, "GROUP   1040") != 0) {
  ERRPRT(1040, "NOT HEADER");
}

if (fscanf(fpHFile, " %d", &N) != 1) {
  LogMsg(stderr, "Error reading N.\n");
  LogClose();
  fclose(fpHFile);
  exit(1);
}

/* Now parse the constant names from each line of input */
for (i = 0; i < N; i++) {
  if (fscanf(fpHFile, " %s", CNAM[i]) != 1) {
    LogMsg(stderr, "Error reading CNAM[%d].\n", i);
    LogClose();
    fclose(fpHFile);
    exit(1);
  }
}

NCON = N;

/* Read number of values and values (GROUP 1041/4) */
NXTGRP(fpHFile, HEADER);
if (strcmp(HEADER, "GROUP   1041") != 0) {
  ERRPRT(1041, "NOT HEADER");
}

if (fscanf(fpHFile, " %d", &N) != 1) {
  LogMsg(stderr, "Error reading N.\n");
  LogClose();
  fclose(fpHFile);
  exit(1);
}

LogMsg(stdout, "\n"
  "Ephemeris Constants\n"
  "-------------------\n");

for (i = 0; i < N; i++) {
  /*
      Read cval mask out D and read exponent
      then convert cval to reflect exponent.
  */
  if (fscanf(fpHFile, " %lfD%d", &CVAL[i], &exponent) != 2) {
    LogMsg(stderr, "Error reading CVAL[%d] and exponent.\n", i);
    LogClose();
    fclose(fpHFile);
    exit(1);
  }

  CVAL[i] *= pow(10, exponent);
  if (strcmp(CNAM[i], "AU")    == 0) AU    = CVAL[i];
  if (strcmp(CNAM[i], "EMRAT") == 0) EMRAT = CVAL[i];
  if (strcmp(CNAM[i], "DENUM") == 0) NUMDE = (int) CVAL[i];

  LogMsg(stdout, "%-6s   %+.15E\t", CNAM[i], CVAL[i]);
```

```
  if (i % 2) {
    LogMsg(stdout, "\n");
  }
}

LeftOver = N % 3;
switch (LeftOver) {
  case 0:
          NumZeros = 0;
          break;
  case 1:
  case 2:
          NumZeros = 3 - LeftOver;
          break;
}

for (i = 0; i < LeftOver; i++) {
  /* Throw away padded values */
  fscanf(fpHFile, " %*fD%*d");
}

NXTGRP(fpHFile, HEADER);
if (strcmp(HEADER, "GROUP   1050") != 0) {
  ERRPRT (1050, "NOT HEADER");
}

/* Read pointers from file */
for (jrow = 0; jrow < 3; jrow++) {
  for (jcol = 0; jcol < 13; jcol++) {
    if (fscanf(fpHFile, " %hd", &IPT[jrow][jcol]) != 1) {
      LogMsg(stderr, "Error reading IPT[%d][%d].\n", jrow, jcol);
      LogClose();
      fclose(fpHFile);
      exit(1);
    }
  }
}

LPT[0] = IPT[0][12];
LPT[1] = IPT[1][12];
LPT[2] = IPT[2][12];

LogMsg(stdout,
  "\n\n"
  "  Body    1st coeff      coefs/component      sets of coefs\n"
  "-------------------------------------------------------------\n");

for (jcol = 0; jcol < 13; jcol++) {
  LogMsg(stdout,
    " %2d %3s     %3d               %3d                  %3d\n",
  jcol+1, Body[jcol], IPT[0][jcol], IPT[1][jcol], IPT[2][jcol]);
}
LogMsg(stdout, "\n");

/* Open direct-access output file ('JPLEPH' by default) */
if (strlen(OutFile) == 0) {
  if (fexist(DEFAULT_OUTPUT)) remove(DEFAULT_OUTPUT);
    strcpy(OutFile, DEFAULT_OUTPUT);
  }

  if ((fpOutputFile = fopen(OutFile, "wb")) == NULL) {
```

```
  LogMsg(stdout, "Can't create binary data file.\n");
  LogMsg(stdout, "\nOK\n");
  LogClose();
  fclose(fpHFile);
  exit(1);
}

/******  BEGINNING OF HEADER  *******/
for (k = 0; k < 3; k++)
  fprintf (fpOutputFile, "%-65s", TTL[k]); /* 195 bytes */

tmpShort = (short) NCON;
make_little_endian((char *)&tmpShort, sizeof(short));
fwrite(&tmpShort, sizeof(short), 1, fpOutputFile); /* 2 bytes */

for (k = 0; k < NCON; k++)
  fprintf(fpOutputFile, "%-6s", CNAM[k]); /* NCON*6 bytes */

for (loop = 0; loop < 3; loop++) {
  tmpDouble = (double) SS[loop];
  make_little_endian((char *)&tmpDouble, sizeof(double));
  fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
} /* 24 bytes */

tmpDouble = (double) AU;
make_little_endian((char *)&tmpDouble, sizeof(double));
fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile); /* 8 bytes */

tmpDouble = (double) EMRAT;
make_little_endian((char *)&tmpDouble, sizeof(double));
fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile); /* 8 bytes */

for (k = 0; k < 3; k++) {
  for (loop = 0; loop < 12; loop++) {
    tmpShort = (short)IPT[k][loop];
    make_little_endian((char *)&tmpShort, sizeof(short));
    fwrite(&tmpShort, sizeof(short), 1, fpOutputFile);
  } /* 72 bytes */
}

tmpShort = (short) NUMDE;
make_little_endian((char *)&tmpShort, sizeof(short));
fwrite(&tmpShort, sizeof(short), 1, fpOutputFile); /* 2 bytes */

for (loop = 0; loop < 3; loop++) {
  tmpShort = (short) LPT[loop];
  make_little_endian((char *)&tmpShort, sizeof(short));
  fwrite(&tmpShort, sizeof(short), 1, fpOutputFile);
} /* 6 bytes */

for (loop = 0; loop < NCON; loop++) {
  tmpDouble = (double) CVAL[loop];
  make_little_endian((char *)&tmpDouble, sizeof(double));
  fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
} /* NCON*8 bytes */

/* Length of header = 317+NCON*14 bytes */
/*******  END OF HEADER  *******/

/* Read and write the ephemeris data records (GROUP 1070) */
NXTGRP(fpHFile, HEADER);
if (strcmp(HEADER, "GROUP   1070") != 0) {
  ERRPRT(1070, "NOT HEADER");
```

```
}

/*
    Close the header file and open the actual data file
    if the header is in a seperate file.
*/
if (!HeaderIncluded) {
  fclose(fpHFile);
  if ((fpInputFile = fopen(InFile, "rt")) == NULL) {
    LogMsg(stdout, "Can't open ascii data file '%s'.\n", InFile);
    LogMsg(stdout, "\nOK\n");
    LogClose();
    exit(1);
  }
}

NROUT = 0;

if (HeaderIncluded) fpInputFile = fpHFile;

/* Read the very first record in */
if (!feof(fpInputFile)) {
  if (fscanf(fpInputFile, " %d %d", &NRW, &NCOEFF) != 2) {
    LogMsg(stderr, "Error reading NRW and NCOEFF.\n");
    LogClose();
    fclose(fpInputFile);
    exit(1);
   }

   for (k = 0; k < NCOEFF; k++) {
     /* Read DB mask out D and read exponent */
     /* and convert DB to reflect exponent.  */
     if (fscanf(fpInputFile, " %lfD%d", &DB[k], &exponent) != 2) {
       LogMsg(stderr, "Error reading DB[%d] and exponent.\n", k);
       LogClose();
       fclose(fpInputFile);
       exit(1);
     }
     DB[k] *= pow(10, exponent);
   }

   LeftOver = NCOEFF % 3;
   switch (LeftOver) {
     case 0:
               NumZeros = 0;
               break;
     case 1:
     case 2:
               NumZeros = 3 - LeftOver;
               break;
   }

   /* Read in padded values and discard them */
   for (k = 0; k < NumZeros; k++) {
     fscanf(fpInputFile, " %*fD%*d");
   }
}

while (!feof(fpInputFile) && (DB[1] < T2)) {
  if ((2 * NCOEFF) != KSIZE) {
    ERRPRT(NCOEFF, " 2*NCOEFF not equal to KSIZE");
  }
```

```
/*
    Skip this data block if the end of the interval is less
    than the specified start time or if it does not begin
    where the previous block ended.
*/
if ((DB[1] > T1) && (DB[0] >= DB2Z)) {
  if (FIRST) {
    /*
        Don't worry about the intervals overlapping or abutting
        if this is the first applicable interval.
    */
    DB2Z = DB[0];
    FIRST = FALSE;
  }
  if (DB[0] != DB2Z) {
    /*
        Beginning of current interval is past the end
        of the previous one.
    */
    ERRPRT(NRW, "Records do not overlap or abut.");
  }

  DB2Z = DB[1];
  NROUT++;

  /* Write the numbers to binary file */
  for (loop = 0; loop < NCOEFF; loop++) {
    tmpDouble = (double) DB[loop];
    make_little_endian((char *)&tmpDouble, sizeof(double));
    fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
  }

  /*
      Save this block's starting date, it's interval span,
      and its end date.
  */
  if (NROUT == 1) {
    SS[0] = DB[0];
    SS[2] = DB[1] - DB[0];
  }
  SS[1] = DB[1];

  /* Update the user as to our progress every 10th block */
  if ((NROUT % 10) == 1) {
    if (DB[0] >= T1) {
      fprintf(stdout,
        "  %4d EPHEMERIS RECORD(S) WRITTEN.  LAST JED = %9.1f\r",
        NROUT, DB[1]);
    }
  }
} else {
  fprintf(stdout, "%s\r", Progress);
  strcat(Progress, ".");
  if (strlen(Progress) == 47) {
    strcpy(Progress, "Searching for first requested record ");
    fprintf(stdout,
      "Searching for first requested record           \r");
  }
}

/*
    Read next block of coefficients unless EOF has
    been reached.
```

```
  */
  if (!feof(fpInputFile)) {
    if (fscanf(fpInputFile, " %d %d", &NRW, &NCOEFF) != 2) {
      LogMsg(stderr, "Error reading NRW and NCOEFF.\n");
      LogClose();
      fclose(fpInputFile);
      exit(1);
    }

    for (k = 0; k < NCOEFF; k++) {
      /*
          Read DB mask out D and read exponent
          and convert DB to reflect exponent.
      */
      if (fscanf(fpInputFile, " %lfD%d", &DB[k], &exponent) != 2) {
        LogMsg(stderr, "Error reading DB[%d] and exponent.\n", k);
        LogClose();
        fclose(fpInputFile);
        exit(1);
      }
      DB[k] *= pow(10, exponent);
    }
    LeftOver = NCOEFF % 3;
    switch (LeftOver) {
      case 0:
                NumZeros = 0;
                break;
      case 1:
      case 2:
                NumZeros = 3 - LeftOver;
                break;
    }

    /* Read in padded values and discard them */
    for (k = 0; k < NumZeros; k++) {
      fscanf(fpInputFile, " %*fD%*d "); /* Need trailing space */
    }                                   /* to test eof.        */
  }
}

/*
    End of file, but no records yet written OR
    just no records yet written.
*/
if ((feof(fpInputFile) && (NROUT == 0)) || (NROUT == 0)) {
  NROUT++;
  SS[0] = DB[0];
  SS[1] = DB[1];
  for (loop = 0; loop < NCOEFF; loop++) {
    tmpDouble = (double) DB[loop];
    make_little_endian((char *)&tmpDouble, sizeof(double));
    fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
  }
}
/* End of file but T2 lies within most recently read record */
else if (feof(fpInputFile)) {
  NROUT++;
  SS[1] = DB[1];
  for (loop = 0; loop < NCOEFF; loop++) {
    tmpDouble = (double) DB[loop];
    make_little_endian((char *)&tmpDouble, sizeof(double));
    fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
  }
```

```
    }
    else if (DB[1] > T2) {
      NROUT++;
      SS[1] = DB[1];
      for (loop = 0; loop < NCOEFF; loop++) {
        tmpDouble = (double) DB[loop];
        make_little_endian((char *)&tmpDouble, sizeof(double));
        fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
      }
    }

    /* Reset file pointer from beginning of file */
    /* and write start and final JED to header.  */
    fpeof = ftell(fpOutputFile);
    fseek(fpOutputFile, (197 + NCON * 6), SEEK_SET);

    for (loop = 0; loop < 2; loop++) {
      tmpDouble = (double) SS[loop];
      make_little_endian((char *)&tmpDouble, sizeof(double));
      fwrite(&tmpDouble, sizeof(double), 1, fpOutputFile);
    }

    LengthOfHeader = 317 + NCON * 14;
    LengthOfRecord =   8 * NCOEFF;
    LengthOfFile = (long)LengthOfHeader + (long)LengthOfRecord *
(long)NROUT;
    /* LengthOfFile should equal fpeof */

    LogMsg(stdout,
      " %4d EPHEMERIS RECORD(S) WRITTEN.  LAST JED = %9.1lf\n", NROUT,
DB[1]);
    LogMsg(stdout, "\nStart JED for this file is %9.1lf\n", SS[0]);
    LogMsg(stdout, "Final JED for this file is %9.1lf\n\n", SS[1]);
    LogMsg(stdout, "Header is %d bytes\n", LengthOfHeader);
    LogMsg(stdout, "Record is %d bytes\n", LengthOfRecord);
    LogMsg(stdout, " %ld bytes written\n", LengthOfFile);
    LogMsg(stdout, " %ld bytes actual file size\n", fpeof);

    LogMsg(stdout, "\nOK\n");

    /* Close all files */
    LogClose();
    fclose(fpOutputFile);
    fclose(fpInputFile);

    if (DeletefpLogFile) {
      remove(szLogFile);
    }

    return(0);
}

/*********************************************************************
Name:    ParseCmdLine
Purpose: Routine to parse the command line.
Inputs:  argc - Number of command-line arguments.
         argv - Pointer to array of command-line arguments.
Outputs: InFile  - Filename given by user for input file.
         OutFile - Filename given by user for output file.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*********************************************************************/
```

```
void ParseCmdLine(int argc, char *argv[], char *InFile, char *OutFile) {
  /*
      Command line parser ported from Basic and optimized for C
       by Varian Swieter.
  */

  int i;

  for (i = 1; i < argc; i++) {
    /*
        Find next occurance of "/" or "-" by looking at
        the first character of each argument.
    */
    if (argv[i][0] != '-' && argv[i][0] != '/') {
      /* Found an argument that did not begin with "/" or "-" */
      LogMsg(stdout, "Command line arguments must begin with '-' or
'/'.\n");
      PrintUsage();
      LogClose();
      remove(szLogFile);
      exit(1);
    }

    switch (argv[i][1]) {
      case '\0':
                  LogMsg(stdout, "Space not allowed between - and
option.\n");
                  LogMsg(stdout, "Type asc2eph -h for help.\n");
                  LogClose();
                  remove(szLogFile);
                  exit(1);
      case 'I':
      case 'i':
                  if (strlen(argv[i]) == 2) {
                    /* We were given nothing after the "-I"
                       so return empty string */
                    strcpy(InFile, "");
                  } else {
                    if (argv[i][2] != ':') {
                      /* User missed out colon so return empty string */
                      strcpy(InFile, "");
                     } else {
                       strcpy(InFile, &(argv[i][3]));
                     }
                  }
                  break;
      case 'O':
      case 'o':
                  if (strlen(argv[i]) == 2) {
                    /* We were given nothing after the "-O"
                       so return empty string */
                    strcpy(OutFile, "");
                  } else {
                    if (argv[i][2] != ':') {
                      /* User missed out colon so return empty string */
                      strcpy(OutFile, "");
                    } else {
                      strcpy(OutFile, &(argv[i][3]));
                    }
                  }

                  /* If OutFile is an empty string then return "JPLEPH"
                     as a default */
```

```
                if (strlen(OutFile) == 0) {
                  strcpy(OutFile, DEFAULT_OUTPUT);
                }
                break;
      case '?':  /* Using this for help will cause problems under UNIX */
                /* because it is used by the shell for substitution.  */
      case 'h':
                PrintUsage();
                LogClose();
                remove(szLogFile);
                exit(0);
      case 'H':
                LogMsg(stdout ,   "Assuming included header.\n");
                HeaderIncluded = TRUE;
                break;
      case 'D':
                PromptForDates = TRUE;
                break;
      case 'K':
                DeletefpLogFile = TRUE;
                break;
      case 'v':  /* Undocumented command line option */
                /* to print version information.    */
                LogMsg(stdout, "OK\n");
                LogClose();
                remove(szLogFile);
                exit(0);
      default:
                LogMsg(stdout, "Option not recognized.\n");
                LogMsg(stdout, "Type asc2eph -h for help.\n");
                LogClose();
                remove(szLogFile);
                exit(1);
    }
  }
}

/*********************************************************************
Name:    NXTGRP
Purpose: Finds the next data group in an ephemeris file.
Inputs:  fptr - File pointer to read from.
Outputs: outstr - Destination for the output string.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
*********************************************************************/
void NXTGRP(FILE *fptr, char *outstr) {
  char string[50];
  char str[6];
  char num[5];

  fscanf(fptr, " %s %s", str, num);
  strcpy(string, str);
  strcat(string, "   ");
  strcat(string, num);

  strcpy(outstr, string);
}

/*********************************************************************
Name:    ERRPRT
Purpose: Prints error messages to stdout and log file.
Inputs:  group   - Error number.
```

```
         message - Error message.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
**********************************************************************/
void ERRPRT(int group, char *message) {
  LogMsg(stdout, "\nERROR #%d %s\n", group, message);
  LogClose();
  fclose(fpHFile);
  fclose(fpOutputFile);
  exit(1);
}
/*********************************************************************
Name:    PrintBanner
Purpose: Prints a banner to stdout and log file.
Inputs:  None.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
**********************************************************************/
void PrintBanner() {
  LogMsg(stdout, "\n");
  LogMsg(stdout, "***Program %s  ", szVersion);
  LogMsg(stdout, "                          Written by Joe Heafner\n");
  LogMsg(stdout, "***Converts JPL ASCII ephemeris data files to binary\n");
  LogMsg(stdout, "***The author can be reached via Internet:");
  LogMsg(stdout, "             heafnerj@interpath.com\n");
  LogMsg(stdout, "\n");
}
/*********************************************************************
Name:    PrintUsage
Purpose: Prints usage information.
Inputs:  None.
Outputs: None.
Returns: Nothing.
Status:  Finished.
Errors:  None known.
**********************************************************************/
void PrintUsage() {
  printf("Usage: ASC2EPH [-i:FILE][-o:FILE][-D][-H][-K][-h]\n");
  printf("Valid command line options are:\n");
  printf("\n");
  printf("    -i:FILE   Use FILE as input assuming separate header file\n");
  printf("              Will prompt if this option is not used\n");
  printf("    -o:FILE   Use FILE as name of binary file\n");
  printf("              Default binary file name is %s\n", DEFAULT_OUTPUT);
  printf("    -D        Program will prompt for initial and final dates\n");
  printf("              Defaults are full range of data file\n");
  printf("    -H        Assume data file has header included\n");
  printf("              Default is separate header in HEADER.XXX\n");
  printf("    -K        Delete log file ASC2EPH.LOG\n");
  printf("              Log file kept by default\n");
  printf("    -h        Display this help screen\n");
  printf("\n");
  printf("Command line options may be in any order.\n");
}
/* End Of File - asc2eph.c ***************************************/
```

Lampiran 8.
Source code file*: makefile.unx*

```
INCLUDE=
LIBS= -lm
CC=gcc

CFLAGS=-ansi -pedantic -Wall -g

TARGETS = asc2eph ephinfo ephtest sephem jdtest


all: $(TARGETS)

clean:
         -rm *.o
         -rm *.log
         -rm $(TARGETS)

asc2eph: asc2eph.o astrocon.o support.o
         $(CC) $(CFLAGS) -o asc2eph asc2eph.o astrocon.o support.o $(LIBS)

ephinfo: ephinfo.o astrolib.o astrocon.o support.o
         $(CC) $(CFLAGS) -o ephinfo ephinfo.o astrolib.o astrocon.o \
         support.o $(LIBS)

ephtest: ephtest.o astrolib.o astrocon.o support.o
         $(CC) $(CFLAGS) -o ephtest ephtest.o astrolib.o astrocon.o \
         support.o $(LIBS)

sephem: sephem.o astrolib.o astrocon.o support.o
         $(CC) $(CFLAGS) -o sephem sephem.o astrolib.o astrocon.o \
         support.o $(LIBS)

jdtest: jdtest.o astrolib.o astrocon.o support.o
         $(CC) $(CFLAGS) -o jdtest jdtest.o astrolib.o astrocon.o \
         support.o $(LIBS)

asc2eph.o: asc2eph.c astrolib.h support.h
         $(CC) $(CFLAGS) -c asc2eph.c

ephinfo.o: ephinfo.c astrolib.h support.h
         $(CC) $(CFLAGS) -c ephinfo.c

ephtest.o: ephtest.c astrolib.h support.h
         $(CC) $(CFLAGS) -c ephtest.c

sephem.o: sephem.c astrolib.h support.h
         $(CC) $(CFLAGS) -c sephem.c

astrolib.o: astrolib.c astrolib.h astrocon.h support.h
         $(CC) $(CFLAGS) -c astrolib.c

astrocon.o: astrocon.c astrocon.h
         $(CC) $(CFLAGS) -c astrocon.c

support.o: support.c support.h
         $(CC) $(CFLAGS) -c support.c

jdtest.o: jdtest.c astrolib.h support.h
         $(CC) $(CFLAGS) -c jdtest.c
```

BIODATA PENULIS

A. Indentitas Diri

1. Nama Lengkap : M. Basthoni
2. Tempat & Tgl. Lahir : Nganjuk, 16 November 1977
3. Alamat Rumah : Jl. Kyai Gilang No. 45 Mangkangkulon Kec. Tugu Kota Semarang
4. HP : 085641016622
5. E-mail : m.basthoni@gmail.com

B. Riwayat Pendidikan

1. Pendidikan Formal
   a. SD Negeri Lambangkuning II Kec. Kertosono Kab. Nganjuk
   b. SMP Negeri I Kec. Kertosono Kab. Nganjuk
   c. MAN Kec. Kertosono Kab. Nganjuk
   d. Jurusan Muamalah Fakultas Syaria'ah IAIN Walisongo Semarang
   e. Jurusan Teknik Komputer Fakultas Teknologi Informasi Universitas Stikubank Semarang
2. Pendidikan Non-Formal
   a. Pondok Pesantren Darul Muta'allimin Kec. Kertosono Kab. Nganjuk
   b. Pondok Pesantran Al-Ishlah Mangkangkulon Kec. Tugu Kota Semarang

Semarang, 08 Juni 2017

M. Basthoni
NIM. 1500028006