Kulminasi
Journal of Falak and Sharia
P-ISSN : 3032-6362 | E-ISSN : 3046-6644

Kulminasi: Journal of Falak and Sharia
Vol. 3 No. 2 Desember 2025
DOI: 10.22373/kulminasi.v3i2.7252

Analysis of the Accuracy of Jean Meeus Ephemeris Algorithm Data on Android Applications Using Kotlin Language

Analisis Keakuratan Data Ephemeris Algoritma Jean Meeus Pada Aplikasi Android Menggunakan Bahasa Kotlin

Andi Hasan Ashari[1]

Institut Agama Islam Ngawi[1]
Email: andihasanelfalakiy@gmail.com[1]

Abstract: Ephemeris is data used to determine the position of celestial objects at a certain time. One of the methods commonly used in ephemeris calculations is the Jean Meeus algorithm. In this study, an analysis of the accuracy of ephemeris data generated by the Jean Meeus algorithm was carried out which was implemented in an Android application using the Kotlin programming language. This application is designed to calculate the position of the sun and moon at hourly intervals in a day based on parameters inputted by the user. The accuracy of the calculation results was compared with the Excel program Determining the Position of the Moon and Sun according to the Meeus Algorithm version 2 (Geocentric and Topocentric) compiled by DR. Rinto Anugraha (UGM Physics Lecturer) and the Hisab Astronomis application by Abu Sabda which uses VSOP87D and ELPMPP02 complete correction terms of 38.326 pieces. The results of the study show that the implementation of the Jean Meeus algorithm in the Android application is able to produce ephemeris data with a high level of accuracy, with errors that are still within the tolerance limits accepted for general astronomical applications. These findings show that the use of Jean Meeus' algorithm in mobile devices is reliable for everyday astronomical applications.
Keywords: *Algorithm, Android, Ephemeris, Jean Meeus, Kotlin*

Abstract: Ephemeris merupakan data yang digunakan untuk mengetahui posisi benda-benda langit pada waktu tertentu. Salah satu metode yang umum digunakan dalam perhitungan ephemeris adalah algoritma Jean Meeus. Pada penelitian ini, dilakukan analisis keakuratan data ephemeris yang dihasilkan oleh algoritma Jean Meeus yang diimplementasikan pada sebuah aplikasi Android menggunakan bahasa pemrograman Kotlin. Aplikasi ini dirancang untuk menghitung posisi matahari dan bulan dengan interval per jam dalam sehari berdasarkan parameter yang diinput oleh pengguna. Keakuratan hasil perhitungan dibandingkan dengan program Excel Program Menentukan Posisi Bulan dan Matahari menurut Algoritma Meeus versi 2 (Geosentrik dan Toposentrik) disusun oleh DR. Rinto Anugraha (Dosen Fisika UGM) dan aplikasi Hisab Astronomis oleh Abu Sabda yang menggunakan VSOP87D dan ELPMPP02 suku koreksi lengkap sebanyak 38.326 buah. Hasil penelitian menunjukkan bahwa implementasi algoritma Jean Meeus dalam aplikasi Android mampu menghasilkan data ephemeris dengan tingkat akurasi yang tinggi, dengan kesalahan yang masih berada dalam batas toleransi yang diterima untuk aplikasi astronomi umum. Temuan ini menunjukkan bahwa penggunaan algoritma Jean Meeus dalam perangkat mobile dapat diandalkan untuk aplikasi astronomi sehari-hari.
Kata Kunci: *Algoritma, Android, Ephemeris, Jean Meeus, Kotlin*

PENDAHULUAN

Data Ephemeris bisa didapatkan dari buku Ephemeris Hisab Rukyah Kementrian Agama RI, The Nautical Almanac, WinHisab 2.0, Hisab Astronomis, dan Accurate Times. Namun seiring dengan kemajuan ilmu pengetahuan dan teknologi, keperluan komputasi astronomi khususnya ilmu falak hisab bisa diimplementasikan lebih mudah dan moderen, salah satunya dengan

menjadikan sebagai aplikasi android menggunakan algoritma Jean Meeus, algoritma Jean Meeus sebenarnya merupakan reduksi dari algoritma VSOP87D dan ELP2000/82 yang lengkap. VSOP87D sebagai dasar perhitungan posisi matahari dan ELP2000/82 untuk dasar perhitungan posisi bulan. Jumlah total suku koreksi pada VSOP87D sebanyak 2.425 buah; 1.080 koreksi untuk bujur ekliptika, 348 koreksi untuk lintang ekliptika dan 997 koreksi untuk jarak matahari-bumi, sedangkan jumlah total suku koreksi pada ELP2000/82 sebanyak 37.862 periodic terms (suku koreksi), terdiri dari 20.560 koreksi bujur bulan, 7.684 koreksi lintang bulan, dan 9.618 koreksi Jarak bulan ke bumi. Dari ribuan suku koreksi pada VSOP87D dan ELP2000/82 tersebut Jean Meeus hanya mengambil suku-suku koreksi yang besar dan penting, sedangkan suku-suku koreksi yang kecil dan kurang penting tidak digunakan[1]. Meskipun Jean Meeus memangkas suku koreksi tersebut menjadi beberapa ratus buah, hasil perhitungan Meeus ini memiliki kesalahan tidak lebih dari 1" (satu detik busur) dalam rentang tahun -2000 sampai 6000 (sekitar 8000 tahun)[2]

Android adalah sistem operasi berbasis Linux dengan kode sumber terbuka dan berlisensi APACHE 2.0 yang dirancang untuk beragam perangkat bergerak (mobile) layar sentuh seperti telepon pintar (smartphone), televisi pintar (Android TV), jam pintar (Wear OS) dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, ponsel Android pertama mulai dijual pada bulan Oktober 2008[3]. Untuk membuat aplikasi android, Google telah merekomendasikan Kotlin sebagai bahasa pemrograman resmi kelas satu untuk android pada tahun 2017. Kotlin merupakan bahasa pemrograman yang berjalan diatas JVM (Java Virtual Machine) yang dikembangkan oleh JetBrains yang bermarkas di Rusia pada tahun 2011, Kotlin versi 1.0 dirilis pada 15 Februari 2016, nama Kotlin diambil dari nama sebuah pulau didekat St. Petersburg. Andrey Breslav (Perancang Bahasa Utama dan Pemimpin Proyek untuk Kotlin di JetBrains)[4] menyebutkan bahwa tim memutuskan untuk menamainya sebuah pulau, seperti halnya Java dinamai menurut pulau Jawa (Java) di Indonesia (meskipun bahasa pemrograman Java diartikan dengan nama kopi dari pulau Jawa)[5].

---

[1] Anugraha, Rinto. *Mekanika Benda Langit.* Yogyakarta: Lab. Fisika Material dan Instrumentasi Jurusan Fisika FMIPA UGM, 2012, hlm. 68.

[2] Tidak lebih dari satu detik jika dibandingkan VSOP87 dan ELP2000/82, lihat: Meeus, Jean. *Astronomical Algorithms.* 2nd ed. Virginia: Willmann-Bell, 1998, hlm. 166.

[3] https://id.m.wikipedia.org/wiki/Android_(sistem_operasi), diakses pada 1 Oktober 2024, pukul 09.02

[4] https://www.abreslav.com/bio, diakses pada 1 Oktober 2024, pukul 09.25

[5] https://id.m.wikipedia.org/wiki/Kotlin_(bahasa_pemrograman), diakses pada 1 Oktober 2024, pukul 09.30

METODE PENELITIAN

Metode penelitian ini termasuk metode engineering science sebab menggabungkan prinsip-prinsip ilmiah, matematika, dan penerapan ilmu komputasi astronomi pada teknologi pemrograman aplikasi android. Sedangkan untuk pengumpulan datanya menggunakan teknik library research, buku Jean Meeus "*Astronomical Algorithms*" edisi kedua (2nd) menjadi sumber rujukan utama (primer) dalam penelitian ini, sedangkan sumber sekunder diambil dari data pustaka berupa buku baik berbentuk e-book (electronic book) atau cetak, makalah-makalah, artikel dan literatur-literatur yang berkaitan dengan obyek penelitian ini.

HASIL DAN PEMBAHASAN

Algoritma Jean Meeus

1. Menghitung Julian Day (JD)[6]

Dengan parameter input berupa tanggal, bulan, tahun masehi, waktu (jam, menit, detik), dan zona waktu:

- Jika bulan > 2, maka M = bulan, dan Y = tahun
- Jika bulan < 2, maka M = bulan + 12, dan Y = tahun - 1

$$A = INT(Y / 100)$$

$$B = 2 - A + INT(A / 4)$$

$$JD = INT(365{,}25 * (Y + 4716)) + INT(30{,}6001 * (M + 1)) + B +$$

$$tanggal + (jam + menit / 60 + detik / 3600) / 24 - 1524{,}5$$

berikut penerapan di bahasa pemrograman Kotlin:

Gambar 1. Fungsi Rumus Julian Day

Ada sedikit penyesuaian di parameter waktu (hourDouble) fungsi masehiToJD tersebut yakni jam, menit, dan detik diminta bertipe Double (desimal/float) agar lebih mudah saat inisialisasi fungsi, berikut rumus mengubah jam, menit, detik ke bentuk desimal:

$$hourDouble = (jam + menit / 60 + detik / 3600)$$

[6] Meeus, Jean. *Astronomical Algorithms.* 2nd ed. Virginia: Willmann-Bell, 1998, hlm. 61.

dan penambahan pengecekan jika tanggal yang dicari sebelum reformasi kalender Gregorian[7] yaitu:

- Jika (tahun + bulan / 100 + tanggal / 10000) ≥ 1582.1015 maka Gregorian, bila hasilnya < 1582.1015 maka Julian[8]
- Jika Julian, maka B = 0

2. Menghitung DeltaT (ΔT)

DeltaT merupakan selisih antara Waktu Dinamis (Dynamical Time TD) dengan Waktu Universal (Universal Time UT), nilai yang eksak dari DeltaT hanya dapat diperoleh dari pengamatan langsung (observasi)[9]. Ada banyak rumus untuk menghitung ΔT yang dijabarkan Jean Meeus dalam bukunya, namun terkadang menghasilkan kesalahan untuk beberapa tahun tertentu[10], berikut rumus pendekatan DeltaT yang penulis ambil dari website NASA yang terbaru (polynomial expressions for DeltaT ΔT) yang mana diadaptasi dari "Five Millennium Canon of Solar Eclipses" karya Jean Meeus dan Fred Espenak dengan cakupan tahun dari -1999 sampai +3000[11].

Definisikan tahun sebagai tahun desimal (y)[12]:

$$y = tahun + \frac{bulan - 1}{12} + \frac{tanggal}{365}$$

- Jika y ≤ -500, maka: $\Delta T = -20 + 32 * u^2$; dimana $u = \frac{(y - 1820)}{100}$
- Jika y > -500 sampai 500, maka:

$$\Delta T = 10583.6 - 1014.41 * u + 33.78311 * u^2 - 5.952053 * u^3$$
$$-0.1798452 * u^4 + 0.022174192 * u^5 + 0.0090316521 * u^6; \text{ dimana } u = \frac{y}{100}$$

- Jika y > 500 sampai 1600, maka:

$$\Delta T = 1574.2 - 556.01 * u + 71.23472 * u^2 + 0.319781 * u^3$$
$$-0.8503463 * u^4 - 0.005050998 * u^5 + 0.0083572073 * u^6; \text{ dimana } u = \frac{y-1000}{100}$$

- Jika y > 1600 sampai 1700, maka:

$$\Delta T = 120 - 0.9808 * u - 0.01532 * u^2 + \frac{u^3}{7129}; \text{ dimana } u = y - 1600$$

- Jika y > 1700 sampai 1800, maka:

$$\Delta T = 8.83 + 0.1603 * u - 0.0059285 * u^2 + 0.00013336 * u^3 - u^4/1174000 \text{ ; dimana } u = y - 1700$$

---

[7] Ibid. hlm. 59-60

[8] Sabda, Abu. *Ilmu Falak Rumusan Syar'i dan Astronomi Seri 03*. cetakan ke-1. DI Yogyakarta: Filosofis Indonesia Press, 2024, hlm. 10.

[9] Meeus, Jean. *Astronomical Algorithms*. 2nd ed. Virginia: Willmann-Bell, 1998, hlm. 77.

[10] Ibid. hlm. 77-78.

[11] https://eclipse.gsfc.nasa.gov/SEcat5/deltatpoly.html, diakses pada 5 Oktober 2024, pukul 11.09

[12] Sabda, Abu. *Ilmu Falak Rumusan Syar'i dan Astronomi Seri 03*. cetakan ke-1. DI Yogyakarta: Filosofis Indonesia Press, 2024, hlm. 12.

- Jika y > 1800 sampai 1860, maka:

$$\Delta T = 13.72 - 0.332447 * u + 0.0068612 * u^2 + 0.0041116 * u^3 - 0.00037436 * u^4 + 0.0000121272 * u^5 - 0.0000001699 * u^6 + 0.000000000875 * u^7; \text{ dimana } u = y - 1800$$

- Jika y > 1860 sampai 1900, maka:

$$\Delta T = 7.62 + 0.5737 * u - 0.251754 * u^2 + 0.01680668 * u^3 - 0.0004473624 * u^4 + \frac{u^5}{233174};$$
$$\text{dimana } u = y - 1860$$

- Jika y > 1900 sampai 1920, maka:

$$\Delta T = -2.79 + 1.494119 * u - 0.0598939 * u^2 + 0.0061966 * u^3 - 0.000197 * u^4;$$
$$\text{dimana } u = y - 1900$$

- Jika y > 1920 sampai 1941, maka:

$$\Delta T = 21.20 + 0.84493 * u - 0.076100 * u^2 + 0.0020936 * u^3; \text{ dimana } u = y - 1920$$

- Jika y > 1941 sampai 1961, maka:

$$\Delta T = 29.07 + 0.407 * u - \frac{u^2}{233} + \frac{u^3}{2547}; \text{ dimana } u = y - 1950$$

- Jika y > 1961 sampai 1986, maka:

$$\Delta T = 45.45 + 1.067 * u - \frac{u^2}{260} - \frac{u^3}{718}; \text{ dimana } u = y - 1975$$

- Jika y > 1986 sampai 2005, maka:

$$\Delta T = 63.86 + 0.3345 * u - 0.060374 * u^2 + 0.0017275 * u^3 + 0.000651814 * u^4 + 0.00002373599 * u^5;$$
$$\text{dimana } u = y - 2000$$

- Jika y > 2005 sampai 2015, maka:

$$\Delta T = 64.69 + 0.2930 * u; \text{ dimana } u = y - 2005$$

- Jika y > 2015 sampai 3000, naka:

$$\Delta T = 67.62 + 0.3645 * u + 0.0039755 * u^2; \text{ dimana } u = y - 2015$$

Untuk dua rumus terakhir (2005-2015 dan 2015-3000) diambil dari website EclipseWise milik Fred Espenak yang telah diupdate yang merujuk pada van der Sluys (2010) diambil dari "*Thousand Year Canon of Solar Eclipses 1501 to 2500*"[13]. Setelah nilai DeltaT didapatkan, DeltaT perlu ditambahkan dengan koreksi c jika y < 1955 atau y > 2005, selain itu tidak perlu ditambah koreksi c, berikut rumus koreksi c dengan penerapan pada kode Kotlin sebagai berikut:

$$c = -0.000012932 * (y - 1955)^2$$

[13] https://eclipsewise.com/help/deltatpoly2014.html, diakses pada 7 Oktober 2024, pukul 09.25

```
fun deltaT(date: Int, month: Int, year: Int): Double {
var u: Double ; var korC: Double ;  var deltaT: Double ; val y = year + (month - 1).toDouble() / 12 + (date).toDouble() / 365
when {  y <= -500 -> {  u = (y - 1820) / 100 ; deltaT = -20 + 32 * u.pow(2)}
        y > -500 && y <= 500 -> { u = y / 100
deltaT = 10583.6 - 1014.41 * u + 33.78311 * u.pow(2) - 5.952053 * u.pow(3) - 0.1798452 * u.pow(4) + 0.022174192 * u.pow(5)
+ 0.0090316521 * u.pow(6)}
        y > 500 && y <= 1600 -> { u = (y - 1000) / 100
deltaT = 1574.2 - 556.01 * u + 71.23472 * u.pow(2) + 0.319781 * u.pow(3) - 0.8503463 * u.pow(4) - 0.005050998 * u.pow(5)
+ 0.0083572073 * u.pow(6)}
        y > 1600 && y <= 1700 -> {  u = (y - 1600)
deltaT = 120 - 0.9808 * u - 0.01532 * u.pow(2) + u.pow(3) / 7129}
        y > 1700 && y <= 1800 -> { u = (y - 1700)
deltaT = 8.83 + 0.1603 * u - 0.0059285 * u.pow(2) + 0.00013336 * u.pow(3) - u.pow(4) / 1174000 }
        y > 1800 && y <= 1860 -> { u = (y - 1800)
deltaT = 13.72 - 0.332447 * u + 0.0068612 * u.pow(2) + 0.0041116 * u.pow(3) - 0.00037436 * u.pow(4) + 0.0000121272 *
u.pow(5) - 0.0000001699 * u.pow(6) + 0.000000000875 * u.pow(7) }
        y > 1860 && y <= 1900 -> { u = (y - 1860)
deltaT = 7.62 + 0.5737 * u - 0.251754 * u.pow(2) + 0.01680668 * u.pow(3) - 0.0004473624 * u.pow(4) + u.pow(5) / 233174}
        y > 1900 && y <= 1920 -> { u = (y - 1900)
deltaT = -2.79 + 1.494119 * u - 0.0598939 * u.pow(2) + 0.0061966 * u.pow(3) - 0.000197 * u.pow(4) }
        y > 1920 && y <= 1941 -> { u = (y - 1920)
deltaT = 21.20 + 0.84493 * u - 0.076100 * u.pow(2) + 0.0020936 * u.pow(3) }
        y > 1941 && y <= 1961 -> { u = (y - 1950)
deltaT = 29.07 + 0.407 * u - u.pow(2) / 233 + u.pow(3) / 2547 }
        y > 1961 && y <= 1986 -> { u = (y - 1975)
deltaT = 45.45 + 1.067 * u - u.pow(2) / 260 - u.pow(3) / 718 }
        y > 1986 && y <= 2005 -> { u = (y - 2000)
deltaT = 63.86 + 0.3345 * u - 0.060374 * u.pow(2) + 0.0017275 * u.pow(3) + 0.000651814 * u.pow(4) + 0.00002373599 *
u.pow(5) }
        y > 2005 && y <= 2015 -> { u = (y - 2005)
deltaT = 64.69 + 0.2930 * u }
        y > 2015 && y <= 3000 -> { u = (y - 2015)
deltaT = 67.62 + 0.3645 * u + 0.0039755 * u.pow(2)}
        else -> { deltaT = 0.0 } }
if (y < 1955 || y > 2005) { korC = -0.000012932 * (y - 1955).pow(2)
deltaT = deltaT + korC } else { deltaT = deltaT } return deltaT }
```

3. Menghitung Julian Day Ephemeris (JDE), Julian Century (JC), Nilai T dan Nilai tau

$$JDE = JD + \frac{DeltaT}{86400}; JC = \frac{JD - 2451545}{36525}; T = \frac{JDE - 2451545}{36525}; tau = \frac{T}{10}$$

kode Kotlin:

```
val jde = jd + deltaT / 86400.0
val jc = (jd - 2451545.0).toDouble() / 36525.0
val nilaiT = (jde - 2451545.0).toDouble() / 36525.0
val tau = nilaiT / 10.0
```

4. Menghitung Koordinat Matahari Ekliptik

Sebelum menghitung koordinat matahari ekliptik terlebih dahulu menghitung posisi bumi heliosentris (earth heliocentric), terdapat komponen periodik yang terbagi menjadi 3 bagian untuk menghitung koordinat heliosentris planet secara langsung[14], yaitu: L untuk bujur ekliptik, B untuk lintang ekliptik, dan R untuk vektor radius (jarak ke matahari).

Khusus utnuk menghitung posisi bumi suku koreksi L terbagi menjadi 6 kelompok, untuk suku koreksi B terbagi menjadi 2 kelompok, dan koreksi R terbagi menjadi 5 kelompok. Berikut rumus untuk menghitung masing-masing kelompok suku koreksi: $A * Cos(B + C * tau)$

a. Menghitung bujur heliosentris bumi dan bujur geosentris matahari

Terdapat 6 kelompok untuk komponen periodik L, yaitu: L0, L1, L2, L3, L4, dan L5, masing-masing kelompok dihitung dengan rumus diatas lalu dijumlahkan pada setiap kelompok, selanjutnya hitung bujur heliosentris bumi dengan rumus dibawah ini:

$$L = \frac{L0 + L1 * tau + L2 * tau^2 + L3 * tau^3 + L4 * tau^4 + L5 * tau^5}{100000000}$$

a.1. Menghitung bujur geosentris matahari dengan rumus: $\theta o = L + 180$,

a.2. Menghitung true geosentric matahari, dengan rumus: $\theta = \theta o - \frac{0.09033}{3600}$

b. Menghitung lintang heliocentric bumi dan lintang geosentris matahari

Suku koreksi lintang (B) terbagi 2 kelompok, yakni: B0 dan B1. Cara menghitungnya sama dengan suku periodik L.

b.1. Rumus menghitung lintang heliocentric bumi (B): $B = (\frac{B0 + B1 * tau}{100000000})$

b.2. Kemudian ubah ke bentuk derajat dengan rumus: $B(derajat) = B(radian) * \frac{180}{PI}$

b.3. Hitung lintang nyata geosentris matahari (β_o): $\beta o = -B$

b.4. Hitung koreksi bujur dengan rumus (λ'): $\lambda' = \theta o - 1.397 * T - 0.00031 * T^2$

b.5. Hitung delta beta (Δβ): $\Delta\beta = 0.03916 * (Cos\ \lambda' - Sin\ \lambda')$

b.6. Hitung apparent latitude matahari (β): $\beta = B + \Delta\beta$

[14] Meeus, Jean. *Astronomical Algorithms.* 2nd ed. Virginia: Willmann-Bell, 1998, hlm. 217.

c. Menghitung vektor radius/jarak matahari-bumi

Ada 5 kelompok suku koreksi untuk menghitung jarak matahari-bumi, yaitu: R0, R1, R2, R3, dan R4. Caranya sama seperti menghitung suku koreksi bujur L dan lintang B.

c.1. Hitung true geocentric distance/jarak matahari-bumi (R):

$R(au) = \frac{(R0+R1*tau+R2*tau^2+R3*tau^3+R4*tau^4)}{100000000}$, dalam satuan AU (astronomical unit)

c.2. Hitung true geocentric distance satuan kilometer KM: $R(km) = R(au) * 149597870.7$

c.3. Hitung true geocentric distance satuan earth radius ER: $R(er) = R(au) * \frac{149597870.7}{6371}$

5. Menghitung Nutasi dan Kemiringan Ekliptik

Nutasi dibagi menjadi dua komponen, yaitu komponen yang parallel terhadap ekliptika yang disebut nutation in longitude (ΔΨ) dan komponen yang tegak lurus terhadap ekliptika yang disebut obliquity of ecliptic (Δε)[15]. Sebelum menghitung ΔΨ dan Δε, terlebih dulu menghitung argumen multiple dengan rumus-rumus yang diambil dari International Astronomical Union (IAU), yang sedikit berbeda dengan yang digunakan dalam teori Lunar karya Chapront[16].

a. Elongasi rata-rata bulan dari matahari (d):

$d = 297.85036 + 445267.111480 * T - 0.001142 * T^2 + \frac{T^3}{189474}$

b. Anomali rata-rata matahari (m):

$m = 357.52772 + 35999.05034 * T - 0.0001603 * T^2 - \frac{T^3}{300000}$

c. Anomali rata-rata bulan (m1):

$m1 = 134.96298 + 477198.867398 * T + 0.0086972 * T^2 + \frac{T^3}{56250}$

d. Argumen lintang bulan (f):

$f = 93.27191 + 483202.017538 * T - 0.0036825 * T^2 + \frac{T^3}{327270}$

e. Bujur dari titik daki (ascending node) dari orbit rata-rata bulan pada ekliptika, diukur dari ekuinoks rata-rata pada tanggal tertentu (omega):

$\omega = 125.04452 - 1934.136261 * T + 0.0020708 * T^2 + \frac{T^3}{450000}$

Selanjutnya menghitung koreksi *nutation in longitude* (ΔΨ)[17] dengan rumus:

$$(Koefisien1 + Koefisien2 * T) * Sin(D * d + M * m + M' * m1 + F * f + \Omega * \omega)$$

dimana Koefisien1 dan Koefisien2 adalah tabel *Coeffisien of the sine of the argument*. Jumlahkan semua suku koreksi dengan rumus tersebut, lalu menghitung nution in longitude dengan rumus:

$$\Delta\Psi = akumulasi\ suku\ koreksi/10000/3600$$

[15] Sabda, Abu. *Ilmu Falak Rumusan Syar'i dan Astronomi Seri 03*. cetakan ke-1. DI Yogyakarta: Filosofis Indonesia Press, 2024, hlm 14.

[16] Meeus, Jean. *Astronomical Algorithms*. 2nd ed. Virginia: Willmann-Bell, 1998, hlm. 143.

[17] Ibid. hlm. 145.

(hasil ΔΨ diatas dalam satuan derajat). Lalu menghitung *obliquity of ecliptic* (Δε) dengan rumus:

$$(Koefisien1 + Koefisien2 * T) * Cos(D * d + M * m + M' * m1 + F * f + \Omega * \omega)$$

dimana Koefisien1 dan Koefisien2 adalah tabel *Coeffisien of the cosine of the argument*. akumulasikan semua suku koreksi tersebut, lalu menghitung obliquity of ecliptic dengan rumus dibawah ini:

$$\varepsilon o \ = \ akumulasi\ suku\ koreksi \ / \ 10000 \ / \ 3600$$

(hasil ε_o diatas dalam satuan derajat). Selanjutnya menghitung mean obliquity of ecliptic (ε_o)dan true obliquity of ecliptic (ε), dengan rumus:

$$\varepsilon o \ = 23 + \frac{26.0}{60} + \frac{21.448}{3600} + (-4680.93 * u - 1.55 * u^2 + 1999.25 * u^3 - 51.38 * u^4 - 249.67 * u^5$$

$$-39.05 * u^6 + 7.12 * u^7 + 27.87 * u^8 + 5.79 * u^9 + 2.45 * u^{10})/3600 \text{ ; dimana } u \ = \frac{T}{100}$$

$$\varepsilon \ = \ \varepsilon o \ + \ \Delta\varepsilon$$

6. Menghitung Apparent Longitude Matahari Geosentris (λ):

$$\lambda \ = \ \theta \ + \ \Delta\Psi \ + \ abr$$

dimana $abr \ = \ -20.4898 \ / \ (3600 \ * \ R(au))$, lalu jadikan dalam rentang 360°

```
val sunApparentGeoLongitude = (sunTrueGeocentricLonJ2000Degrees + deltaPsiDegrees + abrasiDegree).mod(360.0)
```

7. Menghitung Right Ascension dan Sun Apparent Declination (Deklinasi Matahari)[18]:

a. Rumus apparent geocentric right ascension matahari (α):

$$\alpha = Atan(\frac{Sin\ \lambda * Cos\ \varepsilon - Tan\ \beta * Sin\ \varepsilon}{Cos\lambda})$$

*h*asil dikonversi ke satuan derajat dan dijadikan dalam rentang 360 derajat dengan modulus, implementasi dalam bahasa Kotlin:

```
valsunApparentGeoRightAscension = (Math.toDegrees(atan2((sin(Math.toRadians(sunApparentGeoLongitude)) * cos(Math.toRadians(trueObliquityOfEcliptic)) - tan(Math.toRadians(sunApparentGeoLatitude)) * sin(Math.toRadians(trueObliquityOfEcliptic))), cos(Math.toRadians(sunApparentGeoLongitude))))).mod(360.0)
```

$$Sin\ \delta \ = \ Sin\ \beta \ * \ Cos\ \varepsilon \ + \ Cos\ \beta \ * \ Sin\ \varepsilon \ * \ Sin\ \lambda$$

hasil dikonversi ke satuan derajat, implementasi bahasa Kotlin:

```
valsunApparentGeoDeclination = (Math.toDegrees(asin(sin(Math.toRadians(sunApparentGeoLatitude)) * cos(Math.toRadians(trueObliquityOfEcliptic)) + cos(Math.toRadians(sunApparentGeoLatitude)) * sin(Math.toRadians(trueObliquityOfEcliptic)) * sin(Math.toRadians(sunApparentGeoLongitude)))))
```

8. Menghitung Equation of Time (e)[19]:

Sebelum menghitung equation of time, terlebih dahulu menghitung bujur rata-rata matahari dengan rumus:

[18] Ibid. hlm. 93.
[19] Ibid. hlm. 183-187.

$$Lo = 280.4664567 + 360007.6982779 * tau + 0.03032028 * tau^2 + \frac{tau^3}{49931} - \frac{tau^4}{15299} - \frac{tau^5}{1988000}$$

hasil dikonversi ke dalam derajat lalu dijadikan dalam rentang 360° dengan modulus, lalu menghitung equation of time satuan derajat dengan rumus:

$$e = Lo - 0.0057183 - \alpha + \Delta\Psi * Cos\,\varepsilon$$

ubah ke satuan menit jam dengan rumus $e = e * 4$, sedangkan untuk mengubah ke satuan jam menggunakan rumus ketentuan berikut[20]:

- Jika $|e * 4| < 20$, maka $\frac{e}{15}$
- Jika $|e * 4| \geq 20$ dan $e > 0$, maka $\frac{e}{15} - 24$
- Jika $|e * 4| \geq 20$ dan $e < 0$, maka $\frac{e}{15} + 24$

Keterangan: |...| merupakan simbol nilai mutlak (absolut), nilai mutlak ialah hasil nilainya harus dijadikan positif walau nilainya negatif.Kode kotlin equation of time:

```
// equation of time
        val _lo = (280.4664567 + 360007.6982779 * tau + 0.03032028 * tau.pow(2) + tau.pow(3) / 49931 - tau.pow(4) / 15300 - tau.pow(5) / 2000000).mod(360.0)
        val _e = (_lo - 0.0057183 - sunApparentGeoRightAscension + deltaPsiDegrees * cos(Math.toRadians(trueObliquityOfEcliptic)))
/*** equation of time satuan menit, e*/
        val equationOfTimeMinute = _e * 4
/*** equation of time satuan jam, e*/
        val equationOfTimeHour = when { abs(equationOfTimeMinute) < 20.0 -> {_e / 15 }
          abs(equationOfTimeMinute) >= 20.0 && _e > 0.0 -> { _e / 15 – 24 }
          abs(equationOfTimeMinute) >= 20.0 && _e < 0.0 -> { _e / 15 + 24 }
          else -> { _e / 15 } }
```

9. Menghitung Semidiameter Matahari Geosentris (s)[21]:

$s = 15'\,59{,}63''/R(au)$ hasil dalam derajat, kode kotlin:

```
valsunApparentGeocentricSemidiameter = 0.266563888889 /sunTrueGeocentricDistanceAU
```

10. Menghitung Koordinat Bulan Ekliptik

a. Menghitung koreksi posisi bulan[22]

Pertama hitung bujur rata-rata bulan dengan rumus:

$$L' = 218.3164591 + 481267.88134236 * T - 0.0013268 * T^2 + \frac{T^3}{538841} - \frac{T^4}{65194000}$$

[20] Sabda, Abu. *Ilmu Falak Rumusan Syar'i dan Astronomi Seri 03*. cetakan ke-1. DI Yogyakarta: Filosofis Indonesia Press, 2024, hlm 31.

[21] Meeus, Jean. *Astronomical Algorithms*. 2nd ed. Virginia: Willmann-Bell, 1998, hlm. 389.

[22] Ibid. hlm. 337-338

jadikan dalam rentang 360° dengan modulus, implimentasi kode Kotlin:

```
val bujur Rata2 Bulan = (218.3164591 + 481267.88134236 * nilaiT - 0.0013268 * nilaiT.pow(2) + nilaiT.pow(3) / 538841 - nilaiT.pow(4) / 65194000).mod(360.0)
```

kemudian menghitung rumus-rumus argumen berikut:

a.1. Hitung elongasi rata-rata bulan (d):

$$d = 297.8502042 + 445267.1115168 * T - 0.00163 * T^2 + \frac{T^3}{545868} - \frac{T^4}{113065000}$$

jadikan dalam rentang 360° dengan modulus, kode Kotlin:

```
val d = (297.8502042 + 445267.1115168 * t - 0.00163 * t.pow(2) + t.pow(3) / 545868 - t.pow(4) / 113065000).mod(360.0)
```

a.2. Hitung anomali rata-rata matahari (m):

$$m = 357.5291092 + 35999.0502909 * T - 0.0001536 * T^2 + \frac{T^3}{24490000}$$

jadikan dalam rentang 360° dengan modulus, kode Kotlin:

```
val m = (357.5291092 + 35999.0502909 * t - 0.0001536 * t.pow(2) + t.pow(3) / 24490000).mod(360.0)
```

a.3. Hitung anomali rata-rata bulan (ma):

$$ma = 134.9634114 + 477198.8676313 * T + 0.008997 * T^2 + \frac{T^3}{69699} - \frac{T^4}{14712000}$$

jadikan dalam rentang 360° dengan modulus, kode Kotlin:

```
val ma = (134.9634114 + 477198.8676313 * t + 0.008997 * t.pow(2) + t.pow(3) / 69699 - t.pow(4) / 14712000).mod(360.0)
```

a.4. Hitung argumen bujur bulan (jarak rata-rata bulan yang dihitung dari titik perpotongan lintasan bulan dengan ekliptika):

$$f = 93.2720993 + 483202.0175273 * T - 0.0034029 * T^2 - \frac{T^3}{3526000} + \frac{T^4}{863310000}$$

jadikan dalam rentang 360° dengan modulus, kode Kotlin:

```
val f = (93.2720993 + 483202.0175273 * t - 0.0034029 * t.pow(2) - t.pow(3) / 3526000 + t.pow(4) / 863310000).mod(360.0)
```

a.5. selanjutnya menghitung 3 argumen berikut:

$A1 = 119{,}75 + 131{,}849 * T$; $A2 = 53{,}09 + 479264{,}29 * T$; $A3 = 313{,}45 + 481266{,}484 * T$

masing-masing dijadikan dalam rentang 360°, kode Kotlin:

```
val argumenA1 = (119.75 + 131.849 * t).mod(360.0)
val argumenA2 = (53.09 + 479264.29 * t).mod(360.0)
val argumenA3 = (313.45 + 481266.484 * t).mod(360.0)
```

Kemudian menghitung suku koreksi bujur, lintang, dan jarak bulan-bumi

b. Rumus menghitung suku koreksi bujur dan lintang bulan dengan tabel 47.A dan tabel 47.B[23]:

$$Coeffisienn * Sin(Dn * d + Mn * m + M'n * ma + Fn * f)$$; dimana n = data nilai ke n.

Jika nilai M pada tabel tidak sama dengan 0 (nol), yaitu: M < 0 atau M > 0, maka:

$$Coeffisienn * E * Sin(Dn * d + Mn * m + M'n * ma + Fn * f)$$

implementasi pada kode Kotlin:

```
var bujur_bulan = 0.0
    bujur_bulan += koefisien1 * Math.pow(e, Math.abs(M1)) * Math.sin(D1 * d_r + M1 * m_r + MA1 * ma_r + F1 * f_r)
```

dan seterusnya, dimana E adalah eksentrisitas orbit bumi

$$E = 1 - 0.002516 * T - 0.0000074 * T\text{^}2$$

```
val e = 1 - 0.002516 * t - 0.0000074 * t.pow(2)
```

c. Rumus mengitung jarak bulan-bumi dengan tabel 47.A24 (lampiran V):

$$Coefffisienn * Cos(Dn * d + Mn * m + M'n * ma + Fn * f)$$

tapi jika nilai M pada tabel tidak sama dengan 0 (nol), yaitu: M < 0 atau M > 0, maka:

$$Coefffisienn * E * Cos(Dn * d + Mn * m + M'n * ma + Fn * f)$$

```
var lintang_bulan = 0.0
    lintang_bulan += koefisien1 * Math.pow(e, Math.abs(M1)) * Math.sin(D1 * d_r + M1 * m_r + MA1 * ma_r + F1 * f_r)
```

dan seterusnya, dimana E adalah eksentrisitas orbit bumi. Selanjutnya semua hasil masing-masing suku koreksi dijumlahkan, yang mana menghasilkan data:

1. jumlah koreksi bujur bulan (Σl)
2. jumlah koreksi lintang bulan (Σb)
3. jumlah koreksi jarak bulan-bumi (Σr)

d. Hitung koreksi tambahan untuk bujur bulan, lintang bulan, dan jarak bulan-bumi:

d.1. Koreksi tambahan bujur bulan[25]

$$kor\ tambahan\ \Sigma l = jumlah\ koreksi\ bujur\ bulan + 3958 * Sin(A1) + 1962 * Sin(L' - f) + 318 * Sin(A2) / 1000000$$

d.2. Koreksi tambahan lintang bulan[26]

$$kor\ tambahan\ \Sigma b = (jumlah\ koreksi\ lintang\ bulan\ \Sigma b - 2235 * Sin(L') + 382 * Sin(A3) + 175 * Sin(A1 - f) + 175 * Sin(A1 + f) + 127 * Sin(L' - ma) - 115 * Sin(L' + ma) / 1000000$$

[23] Ibid. hlm. 339-341.
[24] Ibid. hlm. 339.
[25] Ibid. hlm. 342.
[26] Ibid.

d.3. Koreksi tambahan jarak bulan-bumi[27]

$$kor\ tambahan\ \Sigma r\ =\ jumlah\ koreksi\ jarak\ bulan-bumi\ \Sigma r\ /\ 1000$$

e. Hitung bujur nyata bulan (moon true longitude), lintang nyata bulan (moon true latitude), dan jarak nyata bulan-bumi (moon true distance)[28]:

e.1. Rumus moon true geocentric longitude (λ'): $\lambda'\ =\ L'\ +\ kor\ tambahan\ \Sigma l$

hasil dijadikan dalam rentang 360°, dengan modulus

e.2. Rumus moon true geocentric latitude (β'): $\beta'\ =\ kor\ tambahan\ \Sigma b$

e.3. Rumus moon true geocentric distance (R'):

R' = 385000,56 + kor tambahan Σr, hasil dalam satuan kilometer

f. Hitung bujur nampak geosentris bulan/moon apparent geocentric longitude (λ):

Rumus moon apparent geocentric longitude[29]: $\lambda\ =\ \lambda'\ +\ \Delta\Psi$

g. Hitung kenaikan kanan nampak bulan/ moon apparent geocentric right ascension (α)[30]:

$\alpha\ =\ Atan((Sin\ \lambda\ *\ Cos\ \varepsilon\ -\ Tan\ \beta\ *\ Sin\ \varepsilon)\ /\ Cos\ \lambda)$

h. Hitung deklinasi nampak bulan/ moon apparent declination (δ)[31]:

$Sin\ \delta\ =\ Sin\ \beta\ *\ Cos\ \varepsilon\ +\ Cos\ \beta\ *\ Sin\ \varepsilon\ *\ Sin\ \lambda$

i. Hitung horizontal pallax bulan (π)[32]: $\pi\ =\ 6378{,}14\ /\ R$, hasil dalam derajat

dimana R = moon true geocentric distance dalam satuan km

j. Menghitung semi diameter bulan (s)[33]: $Sin\ s\ =\ k\ *\ Sin\ \pi$; dimana $k\ =\ 0{,}272481$

k. Hitung elongasi nampak geosentris bulan-matahari/moon-sun apparent geocentric elongation (Ψ)[34]: $Cos\ \Psi\ =\ Sin\ \delta s\ *\ Sin\ \delta m\ +\ Cos\ \delta s\ *\ Cos\ \delta m\ *\ Cos(\alpha s\ -\ \alpha m)$; dimana δs, αs = sun apparent geocentric declination, sun apparent geocentric right ascension; δm, αm = moon apparent geocentric declination, moon apparent geocentric right ascension.

l. Hitung sudut fase bulan nampak geosentris/moon apparent geocentric phase angle (i)[35]: Untuk menghitung sudut fase bulan diperlukan data elongasi bulan-matahari (Ψ), jarak bumi-matahari dan jarak bumi-bulan dalam satuan kilometer, berikut rumus sudut fase bulan:

$$i\ =\ Atan2(Rs\ *\ Sin\ \Psi\ /\ (Rm\ -\ Rs\ *\ Cos\ \Psi))$$

dimana:

[27] Ibid.
[28] Ibid.
[29] Ibid. hlm. 343.
[30] Ibid.
[31] Ibid.
[32] Ibid. hlm. 279.
[33] Ibid. hlm. 390.
[34] Ibid. hlm. 345.
[35] Ibid. hlm. 345-346.

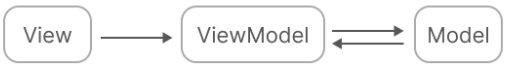
Rs = jarak bumi-matahari dalam kilometer

Rm = jarak bumi-bulan dalam kilometer

m. Hitung apparent geocentric fraction illumination bulan (k)[36]: $k = \frac{1 + Cos\ i}{2}$.

Implementasi Aplikasi Android

Setelah kode algoritma untuk menghitung data ephemeris matahari dan bulan selesai ditulis dalam bahasa Kotlin, selanjutnya mengimplementasikannya menjadi aplikasi android dengan teknik design pattern (pola desain) MVVM (Model-View-ViewModel) yang ditambah teknologi Kotlin Coroutines, MVVM adalah salah satu arsitektur pembuatan aplikasi berbasis GUI yang berfokus pada pemisahan antara kode untuk logika bisnis dan tampilan aplikasi[37].



Gambar 2. Design Pattern MVVM Layer[38]

Komponen layer MVVM terdiri dari 3 bagian[39]:

1. Model; Layer ini merupakan model atau entitas yang merepresentasikan data yang akan digunakan pada logika bisnis.

2. View; Layer ini berisi UI dari aplikasi untuk mengatur bagaimana informasi akan ditampilkan. Layer ini berisi kelas-kelas, seperti Activity dan Fragment.

3. ViewModel; Layer ViewModel bertugas untuk berinteraksi dengan model dimana data yang ada akan diteruskan ke layer view.

Manfaat menggunakan MVVM:

a. Mempertahankan data yang ditampilkan, penulisan kode dengan pola MVVM dengan baik akan membuat data tetap dipertahankan walau terjadi perubahan behaviour pada aplikasi seperti perubahan orientasi layar (orientation changed)[40].

b. Mencegah membuat perubahan besar pada kode model, jika implementasi model yang ada merangkum logika bisnis yang ada, mungkin sulit atau berisiko untuk mengubahnya. Dalam

[36] Ibid.

[37] *https://www.dicoding.com/blog/tips-design-pattern-mvvm/*, diakses pada 24 Oktober 2024, pukul 12.51

[38] Ibid.

[39] Ibid.

[40] Ibid.

skenario ini, model bertindak sebagai adaptor untuk kelas model dan mencegah pengembang membuat perubahan besar pada kode model[41].

c. Pengembang dapat membuat pengujian unit untuk model tampilan dan model itu sendiri, tanpa menggunakan tampilan tersebut[42].

Sedangkan Kotlin Coroutines adalah alat untuk menulis kode secara asinkron yang efisien dan sederhana. Coroutine dapat digunakan dalam berbagai cara dan bersifat multiplatform[43]. Adapun manfaat dan fungsi Kotlin Coroutines salah satunya adalah mencegah kebocoran memori (memory leaks)[44]. Contoh penerapan MVVM dan Kotlin Coroutines:

```
Pada file UserViewModel.kt
    class UserViewModel(repository: UserRepository): ViewModel() {
        val _users = MutableLiveData<Users>() ; val users : LiveData<Users> ; get() = _users
        fun getAllUsers() { viewModelScope.launch(Dispatchers.IO) {
        val tempUsers = repository.getAllUsers() ; withContext(Dispatchers.Main) {_users.postValue(tempUsers) } } } }
Pada file UserActivity.kt
    class UserActivity : AppCompatActivity() {
        override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        btn.setOnClickListener { userViewModel.getAllUser() }
        userViewModel.users.observe(this){ renderUsers(it) }
// atau userViewModel.users.observe(this){users ->
        renderUsers(users) } } }
```

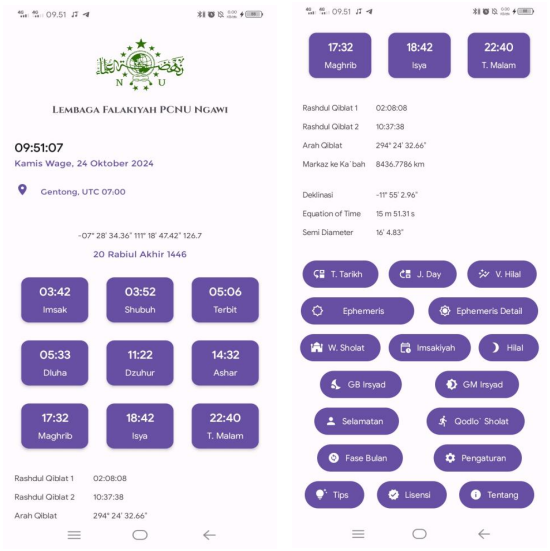
[41] https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm, diakses pada 27 Oktober 2024, pukul 10.05

[42] Ibid.

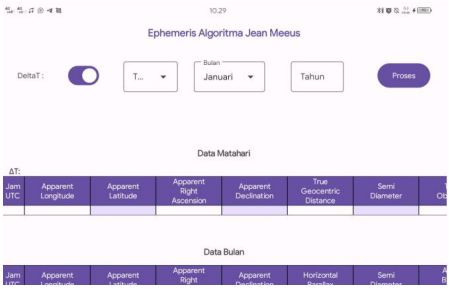
[43] https://kt.academy/article/cc-why, diakses pada 27 Oktober 2024, pukul 10.45

[44] https://developer.android.com/kotlin/coroutines?hl=id, diakses pada 27 Oktober 2024, pukul 11.03

Kemudian aplikasi tersebut penulis beri nama "Kanzul Falak", berikut tampilan antarmuka (UI) aplikasi Kanzul Falak[45]:



Gambar 3. Tampilan halaman depan aplikasi Kanzul Falak dan **Gambar 4.** Tampilan menu aplikasi Kanzul Falak

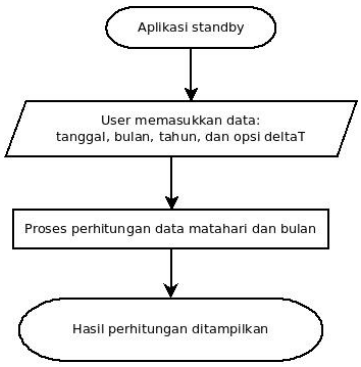


Gambar 5. Tampilan halaman untuk menghitung data ephemeris matahari dan bulan

Gambar 6. Tampilan saat sudah menghitung ditampilkan menjadi tabel

[45] Aplikasi Kanzul Falak dapat diunduh di: *https://github.com/hasanelfalakiy/kanzul-falak-page/releases/latest*, versi terbaru saat artikel ini ditulis adalah v2.7.0

Adapun diagram alur (flowchart) proses perhitungan pada aplikasi Kanzul Falak saat menghitung data ephemeris matahari dan bulan secara umum adalah sebagai berikut:



Gambar 7. Flowchart proses kerja aplikasi Kanzul Falak saat menghitung data ephemeris secara umum

Tampilan aplikasi standby di menu Ephemeris, pengguna memasukkan data yang dibutuhkan di antaranya: tanggal, bulan, tahun, dan opsi untuk memakai deltaT atau tidak, lalu pengguna menekan tombol Proses dan data hasil perhitungan akan muncul sebagai tabel yang bisa digeser kesamping (horizontal) dan layar digeser keatas dan bawah (vertikal).

Analisis, Temuan, dan Pembahasan

Sebagai bahan perbandingan untuk analisis keakuratan hasil data ephemeris di aplikasi Kanzul Falak, disini penulis menggunakan dua sumber program pembanding, yaitu: pertama program excel yang ditulis oleh Rinto Anugraha yang berjudul "Program Menentukan Posisi Bulan dan Matahari menurut Algoritma Meeus versi 2 (Geosentrik dan Toposentrik)[46]" dan yang kedua aplikasi Hisab Astronomis[47] yang diprogram oleh Abu Sabda, menggunakan algoritma VSOP87D dan ELPMPP02[48] suku koreksi lengkap sebanyak 38.326 (tiga puluh delapan ribu tiga ratus dua puluh enam) buah suku koreksi[49].

Sedangkan waktu untuk uji coba akan menggunakan 3 sampel waktu:

[46] Program excel bisa diunduh di: *https://drive.google.com/drive/mobile/folders/14w3FEi0JaeFSbShvQa7ofDHpiRsA8lma?usp=share_link&fbclid=IwY2xjawFIjT5leHRuA2FlbQIxMQABHaHlEHoipAsTB3X2dcxRu7L3W9wTSsLu2DE9ardcDUsvOaaKqm_m1hYupA_aem_6V1zCdgRJaDmZDaNSevF5A*

[47] Aplikasi Hisab Astronomis update v1.0 bisa diunduh di: *https://drive.google.com/file/d/1Ltv4qThg1BSWghzU2GHk2-ptOsjl5LQy/view*

[48] ELPMPP02 adalah teori semi-analitik bulan yang merupakan peningkatan dari teori ELP sebelumnya yakni: ELPMPP01, ELP2000-96, ELP2000-82, ELP2000-82B, dan ELP2000-85. lihat: *https://www.aanda.org/articles/aa/full/2003/23/aa3101/aa3101.html*, diakses pada 31 Oktober 2024.

[49] Keterangan didalam aplikasi tersebut

1. 1 Januari 1900 pukul 18.00 UTC atau 01.00 WIB (2 Januari 1900)
2. 20 Maret 2100 pukul 14.00 UTC atau 21.00 WIB (hari yang sama)
3. 21 Desember 2024 pukul 09.00 UTC atau 16.00 WIB (hari yang sama)

Alasan memilih tahun 1900 dan 2100 karena kedua tahun tersebut merupakan tahun keseratus sebelum dan sesudah epoch pada sistem J2000, selain itu pada tahun 1900, DeltaT bernilai negatif, dan pada tanggal 20 Maret 2100, Matahari berada pada posisi equinox yaitu saat nilai deklinasi mendekati 0°. Sedangkan tanggal 21 Desember 2024, merupakan waktu terdekat yang mana saat itu nilai deklinasi matahari mencapai puncak tertinggi pada utara equator.

Berikut ini hasil uji komparasi antara perhitungan aplikasi Kanzul Falak dengan program excel Program Menentukan Posisi Bulan dan Matahari menurut Algoritma Meeus versi 2 (Geosentrik dan Toposentrik) oleh Rinto Anugraha dan aplikasi Hisab Astronomis oleh Abu Sabda, pada ketiga sampel waktu tersebut:

Jenis Data	**Ephemeris Jean Meeus Kanzul Falak**	**Program Excel Ephemeris Jean Meeus Rinto Anugraha**	**Selisih**
Data Matahari			
DeltaT	-2,825022″	-2,8″	0,025022″ (detik)
Apparent Longitude	280° 55′ 5,37″	280° 55′ 5″	0,37″ (detik)
Apparent Latitude	0,44″	0,44″	sama
Apparent Right Ascension	281° 52′ 27,96″	281° 52′ 27,9″	0,06″
Apparent Declination	-23° 00′ 08,58″	-23° 00′ 09″	0,42″
True Geocentric Distance	0,98326445	0,9832644762	0,000000026
Semi Diameter	16′ 15,96″	16′ 16″	0,04″
True Obliquity	23° 27′ 05,98″	23° 27′ 06″	0,02″
Equation of Time	-03 m 47,21 s	-03 m 47 s	0,21 s
Data Bulan			
Apparent Longitude	283° 14′ 36,79″	283° 14′ 36″	0,79″
Apparent Latitude	02° 03′ 17,1″	02° 03′ 17″	0,1″
Apparent Right Ascension	284° 10′ 16,28″	284° 10′ 15,6″	0,68″
Apparent Declination	-20° 44′ 50,79″	-20° 44′ 51″	0,21″
True Geocentric	365823,73756	365823,8	0,0625

Distance			
Semi Diameter	16′ 19,91″	16′ 20″	0,09″
Horizontal Parallax	00° 59′ 56,41″	00° 59′ 56″	0,41″
Fraction Illumination	0,00074	0,000737	0,000003

Tabel 1.1. Perbandingan Hasil Perhitungan Ephemeris Aplikasi Kanzul Falak dengan Program Excel Ephemeris Jean Meeus Rinto Anugraha pada tanggal 1 Januari 1900 pukul 18.00 UTC atau 01.00 WIB (2 Januari 1900)

Jenis Data	**Ephemeris Jean Meeus Kanzul Falak**	**Aplikasi Hisab Astronomis Abu Sabda**	**Selisih**
Data Matahari			
DeltaT	-2,825022″	-2,76690862726091″	0,058113373″
Apparent Longitude	280° 55′ 05,37″	280° 55′ 05,39″	0,02″
Apparent Latitude	0,44″	0,34″	0,10″
Apparent Right Ascension	281° 52′ 27,96″	281° 52′ 27,98″	0,02″
Apparent Declination	-23° 00′ 08,58″	-23° 00′ 08,68″	0,10″
True Geocentric Distance	0,98326445	0,983263844533723	0,000000605
Semi Diameter	16′ 15,96″	16′ 15,96″	sama
True Obliquity	23° 27′ 05,98″	23° 27′ 05,98″	sama
Equation of Time	-03m 47,21s	-03m 47s	0,21s
Data Bulan			
Apparent Longitude	283° 14′ 36,79″	283° 14′ 35,10″	1,69″
Apparent Latitude	02° 03′ 17,1″	02° 03′ 16,25″	0,85″
Apparent Right Ascension	284° 10′ 16,28″	284° 10′ 14,57″	1,71″
Apparent Declination	-20° 44′ 50,79″	-20° 44′ 51,79″	1″
True Geocentric Distance	365823,73756	-	-
Semi Diameter	16′ 19,91″	16′ 19,92″	0,01″
Horizontal Parallax	00° 59′ 56,41″	00° 59′ 56,44″	0,03″
Fraction Illumination	0,00074	0,0007363537	0,000003646

Tabel 1.2. Perbandingan Hasil Perhitungan Ephemeris Aplikasi Kanzul Falak dengan Aplikasi Hisab Astronomis Abu Sabda pada tanggal 1 Januari 1900 pukul 18.00 UTC atau 01.00 WIB (2 Januari 1900)

Jenis Data	Ephemeris Jean Meeus Kanzul Falak	Program Excel Ephemeris Jean Meeus Rinto Anugraha	Selisih
Data Matahari			
DeltaT	127.28335″	203,3″	76,01665″
Apparent Longitude	00° 02′ 17,95″	00° 02′ 21″	3,05″
Apparent Latitude	0,32″	0,32″	sama
Apparent Right Ascension	00° 02′ 06,45″	00° 02′ 09,34″	2,89″
Apparent Declination	00° 00′ 55,14″	00° 00′ 56″	0,86″
True Geocentric Distance	0,9955366	0,9955368759	0,000000276
Semi Diameter	16′ 03,93″	16′ 04″	0,07″
True Obliquity	23° 25′ 43,9″	23° 25′ 44″	0,1″
Equation of Time	-07m 21,39s	52m 38s	00h 59m 59,39s
Data Bulan			
Apparent Longitude	107° 21′ 26,54″	107° 22′ 07″	40,46″
Apparent Latitude	04° 34′ 35,89″	04° 34′ 34″	1,89″
Apparent Right Ascension	109° 28′ 05,88″	109° 28′ 50,2″	44,32″
Apparent Declination	26° 50′ 25,68″	26° 50′ 18″	7,68″
True Geocentric Distance	383974,198333	383969,0	5,198333
Semi Diameter	15′ 33,59″	15′ 34″	0,41″
Horizontal Parallax	00° 57′ 06,39″	00° 57′ 06″	0,39″
Fraction Illumination	0,64955	0,649631	0,000081

Tabel 2.1. Perbandingan Hasil Perhitungan Ephemeris Aplikasi Kanzul Falak dengan Program Excel Ephemeris Jean Meeus Rinto Anugraha pada tanggal 20 Maret 2100 pukul 14.00 UTC atau 21.00 WIB (hari yang sama)

Jenis Data	Ephemeris Jean Meeus Kanzul Falak	Aplikasi Hisab Astronomis Abu Sabda	Selisih
Data Matahari			
DeltaT	127,28335″	127,269719336459″	0,013630664″
Apparent Longitude	00° 02′ 17,95″	00° 02′ 17,82″	0,13″
Apparent Latitude	0,32″	00,29″	0,03″
Apparent Right Ascension	00° 02′ 06,45″	00° 02′ 06,34″	0,11″
Apparent Declination	00° 00′ 55,14″	00° 00′ 55,06″	0,08″
True Geocentric Distance	0,9955366	0,995537	0,0000004
Semi Diameter	16′ 03,93″	16′ 03,93″	sama
True Obliquity	23° 25′ 43,90″	23° 25′ 43,90″	sama
Equation of Time	-07m 21,39s	-07m 21s	0,39s
Data Bulan			
Apparent Longitude	107° 21′ 26,54″	107° 21′ 23,87″	2,67″
Apparent Latitude	04° 34′ 35,89″	04° 34′ 34,80″	1,09″
Apparent Right Ascension	109° 28′ 05,88″	109° 28′ 02,77″	3,11″
Apparent Declination	26° 50′ 25,68″	26° 50′ 24,96″	0,72″
True Geocentric Distance	383974,198333	-	-
Semi Diameter	15′ 33,59″	15′ 33,60″	0,01″
Horizontal Parallax	00° 57′ 06,39″	00° 57′ 06,43″	0,04″
Fraction Illumination	0,64955	0.649539	0,000011

Tabel 2.2. Perbandingan Hasil Perhitungan Ephemeris Aplikasi Kanzul Falak dengan Aplikasi Hisab Astronomis Abu Sabda pada tanggal 20 Maret 2100 pukul 14.00 UTC atau 21.00 WIB (hari yang sama)

Jenis Data	Ephemeris Jean Meeus Kanzul Falak	Program Excel Ephemeris Jean Meeus Rinto Anugraha	Selisih
Data Matahari			
DeltaT	71,587777″	74,5″	2,912223″
Apparent Longitude	269° 59′ 07,95″	269° 59′ 08″	0,05″
Apparent Latitude	0,25″	0,25″	sama
Apparent Right Ascension	269° 59′ 03,27″	269° 59′ 03,41″	0,14″
Apparent Declination	-23° 26′ 18,13″	-23° 26′ 18″	0,13″
True Geocentric Distance	0,9837313	0,9837313457	0,000000046
Semi Diameter	16′ 15,50″	16′ 16″	0,5″
True Obliquity	23° 26′ 18,38″	23° 26′ 18″	0,38″
Equation of Time	01m 46,46s	01m 46s	0,46s
Data Bulan			
Apparent Longitude	162° 58′ 32,97″	162° 58′ 34″	1,03″
Apparent Latitude	01° 39′ 26,69″	01° 39′ 27″	0,31″
Apparent Right Ascension	164° 56′ 58,86″	164° 57′ 00″	1,14″
Apparent Declination	08° 13′ 05,52″	08° 13′ 05″	0,52″
True Geocentric Distance	398966,360766	398966,5	0,139234
Semi Diameter	14′ 58,51″	14′ 59″	0,49″
Horizontal Parallax	00° 54′ 57,63″	00° 54′ 58″	0,37″
Fraction Illumination	0,64744	0,647441	0,000001

Tabel 3.1. Perbandingan Hasil Perhitungan Ephemeris Aplikasi Kanzul Falak dengan Program Excel Ephemeris Jean Meeus Rinto Anugraha pada tanggal 21 Desember 2024 pukul 09.00 UTC atau 16.00 WIB (hari yang sama)

Jenis Data	Ephemeris Jean Meeus Kanzul Falak	Aplikasi Hisab Astronomis Abu Sabda	Selisih
Data Matahari			
DeltaT	71,587777″	71,5807650994583″	0,007011901″
Apparent Longitude	269° 59′ 07,95″	269° 59′ 07,79″	0,16″
Apparent Latitude	0,25″	0,15″	0,10″
Apparent Right Ascension	269° 59′ 03,27″	269° 59′ 03,10″	0,17″
Apparent Declination	-23° 26′ 18,13″	-23° 26′ 18,23″	0,10″
True Geocentric Distance	0,9837313	0,983732	0,0000007
Semi Diameter	16′ 15,50″	16′ 15,50″	sama
True Obliquity	23° 26′ 18,38″	23° 26′ 18,39″	0,01″
Equation of Time	01m 46,46s	01m 46s	0,46s
Data Bulan			
Apparent Longitude	162° 58′ 32,97″	162° 58′ 31,67″	1,3″
Apparent Latitude	01° 39′ 26,69″	01° 39′ 25,13″	1,56″
Apparent Right Ascension	164° 56′ 58,86″	164° 56′ 57,04″	1,82″
Apparent Declination	08° 13′ 05,52″	08° 13′ 04,59″	0,93″
True Geocentric Distance	398966,360766	-	-
Semi Diameter	14′ 58,51″	14′ 58,51″	sama
Horizontal Parallax	00° 54′ 57,63″	00° 54′ 57,64″	0,01″
Fraction Illumination	0,64744	0,647446	0,000006

Tabel 3.2. Perbandingan Hasil Perhitungan Ephemeris Aplikasi Kanzul Falak dengan Aplikasi Hisab Astronomis Abu Sabda pada tanggal 21 Desember 2024 pukul 09.00 UTC atau 16.00 WIB (hari yang sama)

Dari hasil perbandingan tabel 1.1. diatas selisih nilai antara hasil perhitungan ephemeris Jean Meeus aplikasi Kanzul Falak dengan program excel Jean Meeus Rinto Anugraha tidak terlalu jauh hanya nol koma sekian detik baik data matahari maupun bulan, sedangkan untuk tabel 1.2. perbandingan hasil aplikasi Kanzul Falak dengan aplikasi Hisab Astronomis Abu Sabda, selisih untuk data matahari juga tidak terlalu jauh hanya sekian angka dibelakang koma pada orde detik

busur bahkan ada yang sama, namun untuk data bulan terdapat selisih satu koma sekian detik busur. Hasil perbandingan dari tabel 2.1. aplikasi Kanzul Falak dengan program excel Jean Meeus Rinto Anugraha terdapat beberapa jenis data yang selisihnya terbilang jauh baik data matahari maupun bulan, hal ini dapat terjadi karena beberapa faktor diantaranya: terdapat kesalahan pada algoritma program dan atau rumus yang dipakai pada program excel tersebut, namun sebagai perbandingan, pada tabel 2.2. yang mana hasil perbandingan aplikasi Kanzul Falak dengan aplikasi Hisab Astronomis oleh Abu Sabda yang menggunakan VSOP87D dan ELPMPP02 dengan suku koreksi penuh sebanyak 38.326 buah, terdapat selisih yang relatif kecil, untuk data matahari tidak sampai satu detik busur bahkan beberapa ada yang sama, sedangkan untuk data bulan sebagian besar data selisih sedikit hanya dibelakang koma orde detik kecuali untuk data apparent longitude, apparent latitude, dan apparent right ascension bulan yang selisihnya sampai 2-3 detik busur. Sedangkan untuk tabel 3.1. perbandingan aplikasi Kanzul Falak dengan program excel Rinto Anugraha selisih data kembali relatif kecil begitu juga perbandingan antara aplikasi Kanzul Falak dengan aplikasi Hisab Astronomis Abu Sabda seperti yang ditampilkan pada tabel 3.2. selisih data tidak terlalu signifikan baik data matahari ataupun data bulan.

KESIMPULAN

Setelah melakukan analisis perbandingan dan pembahasan terkait hasil perhitungan antara algoritma Jean Meeus pada aplikasi Kanzul Falak (total koreksi 360 buah suku koreksi) dengan program excel Jean Meeus Rinto Anugraha dan aplikasi Hisab Astronomis Abu Sabda (total koreksi 38.326 buah suku koreksi), dapat disimpulkan bahwa data ephemeris hasil perhitungan pada aplikasi Kanzul Falak dapat dijadikan rujukan ataupun sumber data untuk keperluan komputasi data astronomis khususnya keperluan Falak Hisab baik menghitung arah qiblat, awal waktu sholat, rukyatul hilal, dll.

BIBLIOGRAPHY

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

https://developer.android.com/kotlin/coroutines?hl=id, diakses pada 27 Oktober 2024.

https://eclipse.gsfc.nasa.gov/SEcat5/deltatpoly.html, diakses pada 5 Oktober 2024.

https://eclipsewise.com/help/deltatpoly2014.html, diakses pada 7 Oktober 2024.

https://en.m.wikipedia.org/wiki/Jean_Meeus, diakses pada 1 Oktober 2024.

https://github.com/hasanelfalakiy/kanzul-falak-page/releases/latest, diakses pada 27 Oktober 2024.

https://github.com/hasanelfalakiy/lib-ephemeris-jeanmeeus.git, diakses pada 5 Oktober 2024.

https://id.m.wikipedia.org/wiki/Android_(sistem_operasi), diakses pada 1 Oktober 2024.

https://id.m.wikipedia.org/wiki/Jean_Meeus, diakses pada 1 Oktober 2024.

https://id.m.wikipedia.org/wiki/Kotlin_(bahasa_pemrograman), diakses pada 1 Oktober 2024.

https://kt.academy/article/cc-why, diakses pada 27 Oktober 2024.

https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm, diakses pada 27 Oktober 2024.

https://www.abreslav.com/bio, diakses pada 1 Oktober 2024.

https://www.aanda.org/articles/aa/full/2003/23/aa3101/aa3101.html, diakses pada 31 Oktober 2024.

https://www.dicoding.com/blog/tips-design-pattern-mvvm/, diakses pada 24 Oktober 2024.

Inshafi, Zul Amri Fathinul. *Aplikasi data ephemeris matahari dan bulan berdasarkan perhitungan Jean Meeus pada smartphone Android.* Skripsi. Semarang: Universitas Islam Negeri Walisongo. 2016.

Meeus, Jean. *Astronomical Algorithms.* 2nd ed. Virginia: Willmann-Bell, 1998.

Sabda, Abu. *Ilmu Falak Rumusan Syar'i dan Astronomi Seri 03.* cetakan ke-1. DI Yogyakarta: Filosofis Indonesia Press, 2024.