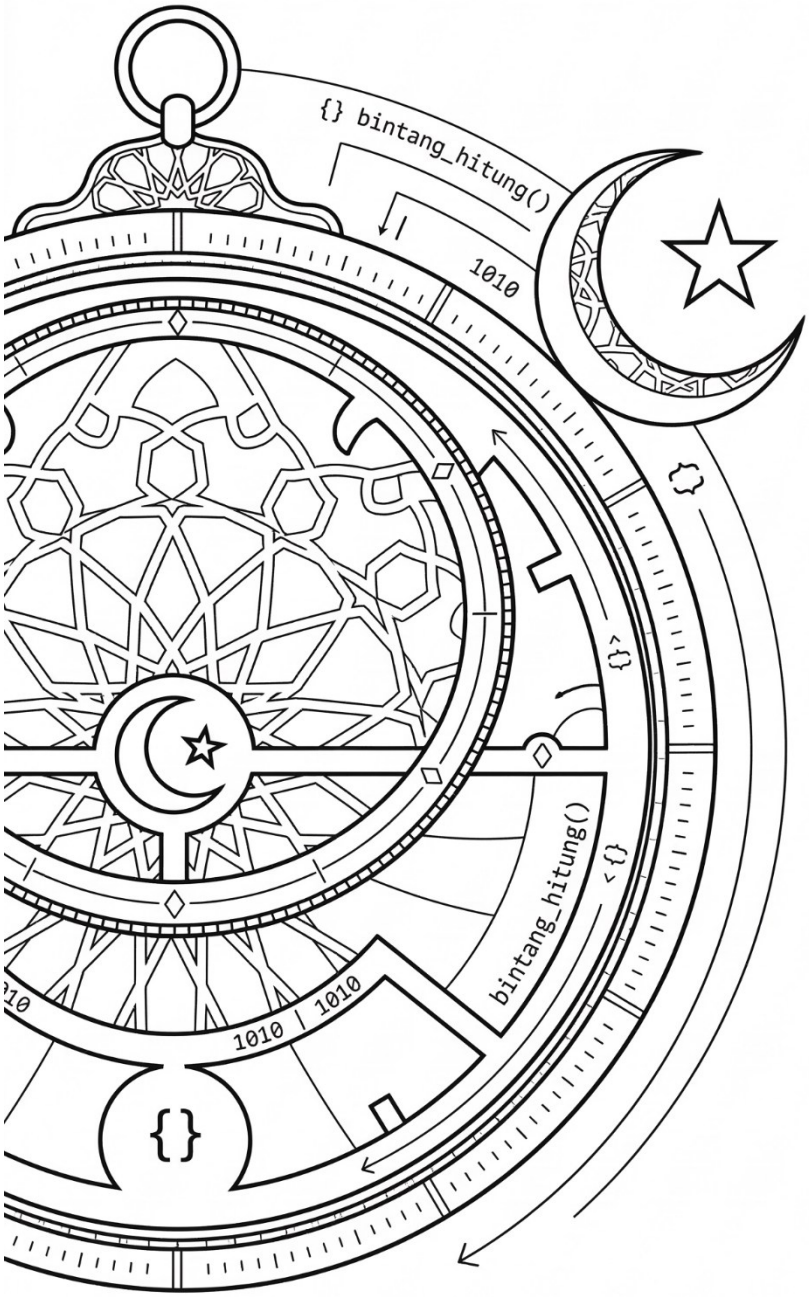
PRAKTEK PYTHON UNTUK ASTRONOMI ISLAM

Practical Python for Islamic Astronomy

PRAKTEK PYTHON UNTUK ASTRONOMI ISLAM



PENULIS:
KASMUI

IFTITAH

Revolusi Komputasi dalam Epistemologi Ilmu Falak Kontemporer

Bismillaahirrahmaanirrahiim.

Segala puji bagi Allah *Subhanahu wa Ta'ala*, Dzat yang menciptakan kosmos dengan arsitektur matematis yang maha presisi. Sebagaimana termaktub dalam firman-Nya, *"Matahari dan bulan beredar menurut perhitungan"* (QS. Ar-Rahman [55]: 5). Ayat ini bukan sekadar deskripsi fenomena alam, melainkan sebuah epistemologi dasar yang memerintahkan umat manusia untuk melakukan observasi dan kuantifikasi (hisab) terhadap dinamika benda-benda langit. Shalawat serta salam senantiasa tercurah kepada baginda Nabi Muhammad *Shallallahu 'alaihi wa sallam*, pionir peradaban yang meletakkan dasar-dasar integrasi antara wahyu dan akal budi.

Urgensi Disiplin

Astronomi Islam (Ilmu Falak) menempati posisi yang sangat unik dan fundamental dalam lanskap keilmuan Islam. Ia bukan sekadar disiplin sains teoretis, melainkan prasyarat mutlak (syarat sah) bagi berdirinya pilar-pilar ibadah vertikal umat Islam. Penentuan awal waktu salat, akurasi arah kiblat di seluruh permukaan bumi, hingga penetapan awal bulan kamariah (khususnya untuk ibadah puasa dan haji) sangat bergantung pada observasi (rukyat) dan komputasi (hisab) astronomis. Di era modern, tuntutan akan akurasi perhitungan ini semakin krusial dan tidak bisa ditawar. Kesalahan sekian derajat dalam perhitungan deklinasi matahari atau *equation of time* tidak hanya bermakna kesalahan matematis, tetapi berimplikasi langsung pada validitas ibadah jutaan umat.

Gap Riset dan Problematika Klasik

Secara historis, peradaban Islam telah melahirkan para astronom raksasa seperti Al-Khawarizmi, Al-Battani, hingga Ulugh Beg yang mewariskan tabel astronomi (*zīj*) dengan akurasi yang melampaui zamannya. Namun, dalam konteks kekinian, kita menghadapi kesenjangan (*research gap*) metodologis yang tajam.

Pertama, literatur dan praktik hisab tradisional seringkali masih mengandalkan kalkulasi manual atau menggunakan perangkat lunak usang yang memakan waktu dan rentan terhadap bias *human error*. *Kedua*, ekosistem teknologi saat ini dibanjiri oleh aplikasi jadwal salat dan penunjuk kiblat yang bersifat *black-box*. Pengguna—dan bahkan para peneliti falak—hanya menerima *output* akhir tanpa memiliki akses, kendali, atau kemampuan untuk memvalidasi algoritma dan parameter astronomis di baliknya. Terjadi segregasi intelektual yang nyata: para ahli syariah dan falak seringkali tidak menguasai instrumen komputasi modern, sementara para ahli *computer science* kerap kali tuna terhadap landasan fikih dan kompleksitas rumus astronomi sferis.

Kebaruan Gagasan (Novelty)

Buku referensi *"Python untuk Astronomi Islam"* ini disusun secara khusus untuk mendekonstruksi kesenjangan tersebut. Kebaruan gagasan (*novelty*) dari karya ini terletak pada

integrasi penuh antara keilmuan falak murni dengan **Python**—bahasa pemrograman tingkat tinggi yang paling revolusioner, tangguh, dan sangat adaptif di era *data science* saat ini.

Buku ini menolak pendekatan komputasi yang pasif. Sebaliknya, karya ini menawarkan paradigma baru dengan membongkar kotak hitam komputasi falak. Melalui buku ini, pembaca tidak hanya disajikan rumusan trigonometri bola yang rumit, tetapi dituntun secara interaktif untuk membangun algoritma sendiri. Dengan memanfaatkan pustaka (library) mutakhir seperti *Skyfield* untuk perhitungan efemeris presisi tinggi (DE440s), serta alat visualisasi *Matplotlib*, buku ini mengubah angka-angka mentah menjadi representasi visual yang komprehensif—mulai dari plot polar arah kiblat, visualisasi azimut matahari, hingga komputasi data observasi hilal tingkat lanjut.

Harapan kami, buku ini dapat menjadi rujukan otoritatif bagi para akademisi, peneliti, praktisi hisab rukyat, serta mahasiswa di seluruh dunia Islam. Melalui penguasaan komputasi Python dalam astronomi Islam, kita tidak hanya melestarikan warisan intelektual ulama terdahulu, tetapi juga memajukan dan merelevansikannya untuk menyongsong tantangan peradaban digital di masa depan.

DAFTAR ISI

Bab 1: Aplikasi Astronomi Islam untuk Muslim

1.1. Pendahuluan: Epistemologi *'Ilm al-Hay'ah* dan Paradigma Sains Islam

Astronomi Islam, yang secara klasik dikenal sebagai *'Ilm al-Hay'ah* (ilmu tentang struktur alam semesta) atau *'Ilm al-Falak* (ilmu tentang lintasan benda langit), menempati posisi sentral dalam arsitektur peradaban Islam. Disiplin ini berkembang bukan sekadar dari keingintahuan intelektual yang kosong (*pure scientific curiosity*), melainkan didorong oleh imperatif teologis yang mewajibkan umat Muslim untuk mengamati ruang angkasa guna memvalidasi pelaksanaan ibadah sehari-hari.

Berbeda dengan tradisi astronomi Barat sekuler yang murni observasional-fisik, astronomi Islam beroperasi pada titik temu (*intersection*) antara ketelitian matematis dan kepatuhan syariat. Ilmu ini mencakup studi tentang posisi, gerak, dan dinamika benda-benda langit (matahari, bulan, dan bintang) yang hasil perhitungannya (*hisab*) diaplikasikan langsung untuk tujuan religius, pembuatan kalender, dan navigasi global (Ilyas, 1997). Urgensi fungsional ini menjadikan astronomi Islam sebagai salah satu cabang sains tertua yang dikembangkan secara sistematis oleh para sarjana Muslim sejak abad ke-8 Masehi, memengaruhi berbagai bidang mulai dari kronometri (ilmu pengukuran waktu) hingga analisis cuaca antariksa (Yusuf, 2010).

Di era digital dan komputasi modern, kompleksitas perhitungan ini mengalami evolusi radikal. Penggunaan tabel astronomi tradisional (*zīj*) dan instrumen mekanik seperti astrolab telah digantikan oleh algoritma komputasi presisi tinggi, seperti yang dapat dieksekusi melalui bahasa pemrograman Python. Transisi ini bukan sekadar pergantian alat, melainkan lompatan paradigma yang memungkinkan perhitungan astronomi dilakukan dengan tingkat akurasi hingga sepersekian detik busur (*arcsecond*), menghilangkan bias kesalahan manusia, dan memfasilitasi visualisasi data yang lebih intuitif.

1.2. Kinematika Matahari dan Akurasi Waktu Salat (Salah)

Aplikasi paling fundamental dan rutin dari astronomi Islam adalah penentuan awal waktu untuk lima salat wajib harian. Pelaksanaan salat terikat secara absolut dengan fenomena gerak semu harian matahari akibat rotasi bumi. Hal ini secara tegas diamanatkan dalam Al-Qur'an:

إِنَّ الصَّلَاةَ كَانَتْ عَلَى الْمُؤْمِنِينَ كِتَابًا مَوْقُوتًا

"Sungguh, salat itu adalah kewajiban yang ditentukan waktunya atas orang-orang yang beriman." (QS. An-Nisa [4]: 103).

Secara astronomis, waktu salat ditentukan oleh posisi deklinasi matahari, sudut jam matahari (*hour angle*), dan lintang geografis pengamat. Salat Zuhur dimulai sesaat setelah matahari melewati meridian lokal (transit surya atau *zawal*). Asar ditentukan oleh panjang bayangan benda yang mencapai rasio tertentu (1 kali atau 2 kali panjang benda ditambah bayangan saat *zawal*). Maghrib dan Syuruk (terbit matahari) bergantung pada fenomena refraksi atmosfer di ufuk, sedangkan Subuh dan Isya ditentukan oleh sudut depresi matahari (*solar depression angle*) ketika fajar *shadiq* muncul dan syafak (senja) menghilang (Abas et al., 2022).

Tantangan komputasi muncul ketika parameter ini diaplikasikan pada lintang tinggi (mendekati kutub), di mana matahari mungkin tidak pernah terbenam secara penuh selama musim panas, atau tidak pernah terbit selama musim dingin. Algoritma komputasi astronomi bertugas memodelkan variabel-variabel anomali ini. Lembaga-lembaga otoritatif, seperti Departemen Kemajuan Islam Malaysia (JAKIM) dengan *Almanak Falak*-nya, menggunakan kombinasi model algoritma dan observasi empiris untuk merumuskan jadwal ini (Huda et al., 2014). Di belahan bumi lain, entitas seperti Dewan Muslim Inggris (*Muslim Council of Britain*) melakukan intervensi perhitungan khusus untuk kota-kota di lintang ekstrem seperti Edinburgh untuk menjaga validitas fikih di tengah anomali cuaca dan durasi siang hari (Ali, 2015). Begitu pula dengan Kalender *Umm al-Qura* di Makkah yang menstandardisasi waktu bagi jutaan peziarah haji dan umrah setiap tahunnya (Rubin, 2017).

1.3. Dinamika Mekanika Bulan dan Penanggalan Hijriah

Berbeda dengan kalender Gregorian yang berbasis pada revolusi bumi mengelilingi matahari (kalender solar), kalender Islam adalah kalender lunar (kamariah) murni. Setiap bulan baru dimulai dengan verifikasi visibilitas bulan sabit pertama (hilal) setelah fase konjungsi (ijtimak). Signifikansi astronomis bulan ini ditegaskan dalam firman Allah:

يَسْأَلُونَكَ عَنِ الْأَهِلَّةِ ۖ قُلْ هِيَ مَوَاقِيتُ لِلنَّاسِ وَالْحَجِّ

"Mereka bertanya kepadamu tentang bulan sabit. Katakanlah: 'Bulan sabit itu adalah tanda-tanda waktu bagi manusia dan (bagi ibadat) haji...'" (QS. Al-Baqarah [2]: 189).

Penentuan awal bulan, khususnya untuk bulan-bulan sakral seperti Ramadan, Syawal (Idulfitri), dan Zulhijah, adalah proses astronomis yang sangat rumit. Orbit bulan sangat dinamis dan rentan terhadap perturbasi (gangguan gravitasi) dari matahari dan bumi (Ilyas, 1984). Visibilitas hilal bergantung pada multi-variabel: umur bulan, elongasi (jarak sudut antara bulan dan matahari), ketinggian hilal saat matahari terbenam, ketebalan sabit, limitasi optik mata manusia, hingga tingkat polusi cahaya dan kondisi atmosfer lokal.

Aktivitas observasi (rukyat) ini dikoordinasikan secara masif di negara-negara mayoritas Muslim. Di Indonesia, Kementerian Agama mengkoordinasikan observasi di puluhan titik pemantauan dari Aceh hingga Papua, termasuk di Observatorium Pelabuhan Ratu (Wahidi et al., 2019). Di Malaysia, observasi dikoordinasikan dari situs-situs seperti Observatorium Bukit Agok (Mohd Nawawi et al., 2024). Sementara itu, negara seperti Maroko mengembangkan pendekatan spesifik yang memadukan hisab ketat dengan rukyat lokal (Lairgi, 2025). Melalui pendekatan komputasional mutakhir (seperti menggunakan pustaka Skyfield pada Python), data pergerakan hilal dapat dikalkulasi secara prediktif berbulan-bulan sebelumnya, memungkinkan pemerintah untuk mengelola mitigasi potensi perbedaan penetapan awal puasa atau hari raya.

1.4. Geodesi Bola dan Presisi Arah Kiblat

Kiblat adalah orientasi spasial fundamental bagi umat Muslim di seluruh dunia yang menuntut presisi geometris menuju Ka'bah di Makkah. Arahan ini bersifat universal sebagaimana diamanatkan:

قَدْ نَرَىٰ تَقَلُّبَ وَجْهِكَ فِي السَّمَاءِ ۖ فَلَنُوَلِّيَنَّكَ قِبْلَةً تَرْضَاهَا ۚ فَوَلِّ وَجْهَكَ شَطْرَ الْمَسْجِدِ الْحَرَامِ ۚ وَحَيْثُ مَا كُنْتُمْ فَوَلُّوا وُجُوهَكُمْ شَطْرَهُ

"Sungguh Kami (sering) melihat mukamu menengadah ke langit, maka sungguh Kami akan memalingkan kamu ke kiblat yang kamu sukai. Palingkanlah mukamu ke arah Masjidil Haram. Dan di mana saja kamu berada, palingkanlah mukamu ke arahnya." (QS. Al-Baqarah [2]: 144).

Secara matematis, bumi bukanlah bola sempurna, melainkan *oblate spheroid* (elipsoida referensi seperti WGS 84). Penentuan arah terpendek menuju Makkah dari titik mana pun di bumi harus menggunakan prinsip trigonometri bola (*spherical trigonometry*) untuk menghitung jarak bujur sangkar agung (*great circle track*). Di masa lalu, navigasi ini menggunakan bayangan matahari dan instrumen primitif. Di era modern, komputasi Qibla Direction menggunakan data lintang dan bujur dari satelit GPS, yang kemudian dikoreksi dengan deklinasi magnetik lokal (Faid, Nahwandi, et al., 2022).

Tuntutan akurasi ini tidak main-main. Sebagai contoh, Masjid Al-Azhar di Mesir yang merupakan institusi tertua, harus menjalani kalibrasi ulang arah kiblatnya pada tahun 1992 setelah survei astronomi modern mendeteksi anomali orientasi yang diwariskan selama berabad-abad (Rabbat, 1996). Kini, model komputasi matematis yang terintegrasi di dalam aplikasi gawai (seperti Muslim Pro) atau Google Earth menggunakan algoritma astronomi untuk merender arah Ka'bah secara aktual dari lokasi *real-time* pengguna (Schumm, 2020; Yildirim et al., 2024).

1.5. Astronomi Fenomenologis: Respons Syar'i Terhadap Gerhana

Ilmu falak tidak hanya berkutat pada siklus harian dan bulanan, tetapi juga melingkupi prediksi dan validasi kejadian langit insidental (astronomi fenomenologis) seperti gerhana matahari (Kusuf) dan gerhana bulan (Khusuf). Fenomena syzygy—saat matahari, bumi, dan bulan berada pada satu garis lurus yang bertepatan dengan titik simpul orbit (node)—menjadi momen pembuktian komputasi astronomi sekaligus momen spiritual. Berbeda dengan peradaban kuno yang mengaitkan gerhana dengan mitos kematian atau bencana, Islam merevolusi pandangan ini menjadi fenomena fisika murni yang harus direfleksikan dengan ketundukan rasional.

Hal ini didasarkan pada fondasi empiris dari sabda Nabi Muhammad Shallallahu 'Alaihi Wasallam:

إِنَّ الشَّمْسَ وَالْقَمَرَ آيَتَانِ مِنْ آيَاتِ اللَّهِ، لَا يَخْسِفَانِ لِمَوْتِ أَحَدٍ وَلَا لِحَيَاتِهِ، فَإِذَا رَأَيْتُمُوهُمَا فَادْعُوا اللَّهَ وَصَلُّوا حَتَّى يَنْجَلِيَ

"Sesungguhnya matahari dan bulan adalah dua tanda dari tanda-tanda kebesaran Allah. Keduanya tidak gerhana karena mati atau hidupnya seseorang. Maka apabila kalian melihatnya, berdoalah kepada Allah dan salatlah sampai (gerhana) itu hilang." (HR. Bukhari, No. 1044).

Saat ini, astronom Muslim dapat memprediksi waktu kontak pertama (P1), puncak gerhana, hingga kontak terakhir (P4) hingga ketelitian detik (King, 1993). Observatorium modern, seperti Observatorium Bosscha di Indonesia, kerap menjadi pusat pemantauan sekaligus penyiaran fenomena ini kepada publik, menghubungkan edukasi fisika kosmik dengan mobilisasi masyarakat untuk melaksanakan Salat Gerhana secara berjemaah (Izzuddin et al., 2022; Elmhamdi et al., 2024).

1.6. Eksplorasi Navigasi Bintang dalam Historiografi Islam

Sejarah mencatat bahwa kontribusi umat Islam dalam astronomi tidak hanya pada ranah ibadah teritorial, tetapi memanjang hingga penguasaan maritim dan navigasi lintasan bumi. Sebelum penemuan sistem navigasi berbasis satelit, para pelaut dan pedagang lintas benua sangat bergantung pada rasi bintang.

Pelaut Muslim yang menavigasi Samudra Hindia antara Afrika, Semenanjung Arab, hingga Asia Tenggara menciptakan dan menggunakan *Kamal*—sebuah instrumen penunjuk arah berbasis blok kayu dan tali yang sangat efektif untuk mengukur ketinggian bintang dari garis cakrawala (Niri et al., 2023). Di wilayah Gurun Sahara, armada kafilah dagang mengandalkan Bintang Utara (Polaris) untuk memandu rute malam yang mematikan menuju Makkah. Dominasi teknologi ini memuncak pada era Kesultanan Utsmaniyah (Ottoman) ketika armada militer maritim mendominasi Laut Mediterania berbekal peta bintang berbasis hisab presisi dan modifikasi kuadran serta astrolab laut (Faid, Nawawi, et al., 2022).

1.7. Implementasi Kode Komputasi Astronomi: Lingkungan Python

Untuk mengakomodasi kompleksitas perhitungan yang telah diuraikan di atas, platform pemrograman modern seperti Python (melalui Google Colab atau *local environment*) menyediakan kapabilitas komputasi tingkat tinggi. Penggunaan Python dalam ilmu falak mentransformasi rumusan empiris yang kaku menjadi data dinamis. Berikut adalah sebuah prototipe *source code* dalam Python yang memperlihatkan bagaimana lingkungan kerja diinisialisasi menggunakan pustaka mutakhir `skyfield` untuk menarik data posisi benda langit dari *Jet Propulsion Laboratory* (JPL) NASA.

<pre style="font-family: 'Courier New', Courier, monospace; background-color: #f4f4f4; padding: 15px; border-radius: 5px; border: 1px solid #ccc;">

Contoh Prototipe: Inisialisasi Environment Python untuk Falak

Penjelasan: Kode ini memuat pustaka astronomi dan menarik

data efemeris presisi tinggi untuk komputasi posisi bulan & matahari.

----

from skyfield.api import load

def inisialisasi_astronomi():

1. Memuat skala waktu (timescale) untuk referensi UTC dan waktu Terestrial (TT)

ts = load.timescale()

waktu_sekarang = ts.now()

2. Memuat data ephemeris DE421.bsp (Standard astronomi JPL NASA)

```
# Memuat posisi akurat pusat massa tata surya, bumi, matahari, dan bulan
planets = load(&#39;de421.bsp&#39;)
bumi = planets[&#39;earth&#39;]
bulan = planets[&#39;moon&#39;]
matahari = planets[&#39;sun&#39;]

# 3. Output konfirmasi keberhasilan sistem
print(&quot;Sistem Astronomi Islam berhasil diinisialisasi.&quot;)
print(f&quot;Waktu          Observasi          Universal          (UTC):
{waktu_sekarang.utc_datetime()}&quot;)

return ts, bumi, bulan, matahari
```

Menjalankan fungsi inisialisasi

if **name** == "**main**":

ts, bumi, bulan, matahari = inisialisasi_astronomi()

</pre>

Penjelasan Kode: Kode di atas menggunakan pustaka `skyfield` yang dikembangkan khusus untuk memproses observasi luar angkasa dengan akurasi observatorium. Perintah `load.timescale()` menyelaraskan waktu rotasi bumi dengan standar atomik internasional. Berkas `de421.bsp` adalah berkas efemeris dari NASA yang menampung peta vektor gravitasi dan koordinat geometris seluruh planet. Objek `bumi`, `bulan`, dan `matahari` yang diinstansiasi (*instantiated*) selanjutnya akan menjadi landasan untuk proses komputasi yang lebih kompleks seperti perpotongan azimuth, sudut elongasi untuk hilal, maupun kalkulasi vektor bayangan untuk Qibla (yang akan dibahas lebih mendalam pada bab-bab selanjutnya).

Daftar Pustaka (Bab 1)

1. Abas, A.-M., Safiai, M. H., Hasan, S. A., Azam, A. I., & Hussaini, R. (2022). Penentuan Waktu Ibadah Solat dan Puasa di Bangunan Pencakar Langit. *BITARA International Journal of Civilizational Studies and Human Sciences*, 6(1), 53–59.
2. Ali, M. (2015). Is the British weather anti-Islamic? Prayer times, the ulama and application of the shari'a. *Contemporary Islam*, 9(2), 171–187.
3. Elmhamdi, A., Roman, M. T., Peñaloza-Murillo, M. A., Pasachoff, J. M., Liu, Y., Al-Mostafa, Z. A., et al. (2024). Impact of the eclipsed sun on terrestrial atmospheric parameters in desert locations: A comprehensive overview and two events case study in Saudi Arabia. *Atmosphere*, 15(1), 62.
4. Faid, M. S., Nahwandi, M. S., Nawawi, M. S. A. B. M., Zaki, N. B. A., & Saadon, M. H. M. (2022). Development of Qibla direction determinant using sun shadow. *Online Journal of Research in Islamic Studies*, 9(1), 89–102.
5. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., Ahmad, N., & Ali @ Mat Zin, A. (2022). Islamic historical review on the middle age lunar crescent visibility criterion. *Journal of Al-Tamaddun*, 17(1).
6. Huda, N., Zaki, A., Ali, A. K., Wahab, R. A., & Niri, M. A. (2014). Penentuan waktu solat Subuh menggunakan Rubu' Mujayyab di Malaysia. *Jurnal Fiqh*, 11(11), 97–118.

7. Ilyas, M. (1984). *A modern guide to astronomical calculations of Islamic calendar, times and qibla*.
8. Ilyas, M. (1997). *Astronomy of Islamic calendar* (1st ed.). A. S. Nordeen.
9. Izzuddin, A., Imroni, M. A., Imron, A., & Mahsun, M. (2022). Cultural myth of eclipse in a Central Javanese village: Between Islamic identity and local tradition. *HTS Teologiese Studies / Theological Studies*, 78(4).
10. King, D. A. (1993). *Astronomy in the service of Islam*. Routledge.
11. Lairgi, Y. (2025). When astronomy meets AI: Manazel for crescent visibility prediction in Morocco. *arXiv*.
12. Mohd Nawawi, M. S. A., Faid, M. S., Saadon, M. H. M., Wahab, R. A., & Ahmad, N. (2024). Hijri month determination in Southeast Asia: An illustration between religion, science, and cultural background. *Heliyon*, 10(20).
13. Niri, M. A., Jamaludin, M. H., Nawawi, M. S. A. M., Zaki, N. A., & Wahab, R. A. (2023). Astronomy development since antiquity to Islamic civilization from the perspective of Islamic historiography. *Journal of Al-Tamaddun*, 18(1), 169–177.
14. Rabbat, N. (1996). Al-Azhar mosque: An architectural chronicle of Cairo's history. *Muqarnas*, 13, 45.
15. Rubin, U. (2017). Morning and evening prayers in early Islam. In G. Hawting (Ed.), *The development of Islamic ritual* (pp. 105–129). Routledge.
16. Schumm, W. R. (2020). How accurately could early (622-900 C.E.) Muslims determine the direction of prayers (Qibla)? *Religions*, 11(3), 102.
17. Wahidi, A., Yasin, N., & Kadarisman, A. (2019). The beginning of Islamic months determination in Indonesia and Malaysia: Procedure and social condition. *ULUL ALBAB Jurnal Studi Islam*, 20(2), 322–345.
18. Yildirim, F., Kadi, F., & Sahin, S. L. (2024). Developing a new interface for Qibla direction application based on MATLAB GUI. *Survey Review*.
19. Yusuf, H. (2010). *Caesarean moon births: Calculations, moon sighting and the prophetic way*. Zaytuna Institute.

BAB 2 MENGAPA PYTHON PENTING DALAM ASTRONOMI ISLAM

2.1 Pendahuluan: Epistemologi Komputasi dan Ekosistem Python dalam Falak Modern

Dalam lanskap keilmuan modern, transisi dari perhitungan astronomi manual menuju komputasi algoritmik merupakan sebuah keniscayaan. Astronomi Islam (Ilmu Falak), yang sarat dengan formula trigonometri bola dan parameter mekanika benda langit yang dinamis, membutuhkan instrumen yang mampu menawarkan presisi, kecepatan, dan kemampuan visualisasi data yang tinggi. Di sinilah bahasa pemrograman Python hadir sebagai katalisator utama yang mendemokratisasi dan merevolusi studi astronomi.

Python, yang diciptakan oleh Guido van Rossum dan pertama kali dirilis pada 20 Februari 1991, merupakan bahasa pemrograman tingkat tinggi (*high-level programming language*) yang bersifat *open-source*, diinterpretasikan (*interpreted*), dan berorientasi objek (*object-oriented*). Menariknya, terlepas dari logonya yang menyerupai ular piton, nama "Python" sebenarnya diadaptasi dari serial komedi televisi BBC Inggris klasik, *"Monty Python's Flying Circus"* (Mehare et al., 2023). Salah satu keistimewaan Python—yang membedakannya dari bahasa pemrograman lain yang umumnya dikembangkan oleh korporasi raksasa—adalah bahwa bahasa ini bermula dari visi satu individu yang kemudian dikembangkan secara gotong royong oleh komunitas global di bawah naungan *Python Software Foundation* (Ozgur et al., 2021; Faid, Mohd Nawawi, et al., 2024).

Bagi disiplin astronomi, Python bukan sekadar alat pelengkap, melainkan tulang punggung komputasi saintifik (Faid, Nawawi, Saadon, et al., 2023). Hal ini dimungkinkan oleh ekosistem pustaka (*libraries*) yang sangat kaya dan spesifik untuk analisis sains, seperti:

1. **NumPy dan SciPy**: Menyediakan landasan untuk komputasi array multidimensi, aljabar linear, dan operasi matematika kompleks dengan kecepatan tinggi.
2. **Scikit-Image**: Digunakan secara luas dalam pengolahan citra astronomis, misalnya untuk memproses foto astrografi hilal dan mengurangi *noise* optik.
3. **Astropy dan Skyfield**: Pustaka fundamental yang memungkinkan pengunduhan data efemeris NASA JPL (seperti seri DE421 atau DE440s), konversi skala waktu (UTC ke Waktu Terestrial), dan perhitungan posisi benda langit dengan akurasi di bawah satu detik busur (Rhodes, 2011).

Sebagai contoh aplikatif di kancah internasional, *European Southern Observatory* (ESO) yang mengoperasikan *Very Large Telescope* (VLT) menyediakan data instrumen SPHERE secara terbuka. Para peneliti dapat menggunakan Python untuk mereduksi derau data (*noise reduction*) dan memvisualisasikan eksoplanet yang tersembunyi di balik piringan protoplanet. Dalam konteks Astronomi Islam, ekosistem Python ini diadaptasi untuk membedah tiga pilar utama: perhitungan waktu salat, penentuan arah kiblat, dan komputasi visibilitas hilal untuk kalender Hijriah.

2.2 Perhitungan Waktu Salat Berbasis Algoritma Astronomi

Penentuan awal waktu salat merupakan domain paling elementer sekaligus krusial dalam rutinitas ibadah umat Islam. Secara teologis, Allah *Subhanahu wa Ta'ala* telah menetapkan indikator-indikator fisik (fenomena matahari) sebagai penanda waktu salat, sebagaimana termaktub dalam firman-Nya:

أَقِمِ الصَّلَاةَ لِدُلُوكِ الشَّمْسِ إِلَىٰ غَسَقِ اللَّيْلِ وَقُرْآنَ الْفَجْرِ ۖ إِنَّ قُرْآنَ الْفَجْرِ كَانَ مَشْهُودًا

"Dirikanlah salat dari sesudah matahari tergelincir sampai gelap malam dan (dirikanlah pula salat) Subuh. Sesungguhnya salat Subuh itu disaksikan (oleh malaikat)." (QS. Al-Isra [17]: 78).

Lebih rinci lagi, Nabi Muhammad *Shallallahu 'alaihi wa sallam* memberikan parameter visual yang jelas, sebagaimana diriwayatkan dalam hadits sahih:

عَنْ عَبْدِ اللَّهِ بْنِ عَمْرٍو، أَنَّ رَسُولَ اللَّهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ قَالَ: «وَقْتُ الظُّهْرِ إِذَا زَالَتِ الشَّمْسُ وَكَانَ ظِلُّ الرَّجُلِ كَطُولِهِ مَا لَمْ يَحْضُرِ الْعَصْرُ...»

"Dari Abdullah bin Amr, bahwa Rasulullah Shallallahu 'alaihi wa sallam bersabda: 'Waktu Zuhur adalah jika matahari telah tergelincir (condong ke barat) dan bayangan seseorang sama dengan tingginya, selama waktu Asar belum tiba...'" (HR. Muslim, No. 612).

Dalam terminologi falak modern, parameter-parameter syar'i ini harus dikonversi menjadi data kuantitatif astronomis (Faid et al., 2019):

- **Zuhur:** Dimulai sesaat setelah fase *solar transit* atau kulminasi atas (matahari melewati meridian pengamat).
- **Asar:** Ditentukan oleh panjang bayangan objek (*sun's shadow*) yang dipengaruhi oleh deklinasi matahari pada hari itu dan lintang pengamat.
- **Maghrib dan Syuruk:** Didefinisikan berdasarkan *sunset* dan *sunrise* geometris, dengan memperhitungkan faktor refraksi atmosfer (pembiasan cahaya) yang membuat matahari terlihat meskipun secara fisik sudah berada di bawah ufuk.
- **Subuh dan Isya:** Ditentukan oleh *solar depression angle* (sudut depresi matahari) di bawah ufuk, yang merepresentasikan munculnya fajar *shadiq* (Fajr) dan hilangnya mega merah (Isha).

Tingkat kerumitan perhitungan ini sangat tinggi. Pergerakan matahari tidak berada pada garis linier yang konstan akibat kemiringan sumbu bumi (ekliptika) sebesar 23,5 derajat dan bentuk orbit bumi yang elips. Di sinilah Python memainkan peran krusial. Melalui integrasi pustaka *Astropy* atau *Skyfield*, Python dapat menarik secara otomatis parameter hambatan (*perturbation*), *Equation of Time* (perata waktu), dan deklinasi harian tanpa memerlukan tabel logaritma manual seperti di masa lalu (Faid et al., 2021). Python memungkinkan kita memproses ribuan titik koordinat geografis dalam hitungan detik dan memformulasikannya menjadi jadwal waktu salat yang adaptif sepanjang tahun.

Berikut adalah ilustrasi kode Python untuk mendeklarasikan inisialisasi awal pencarian posisi elevasi matahari sebagai dasar komputasi waktu salat:

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f9f9f9; padding: 15px; border-radius: 8px; border: 1px solid #ddd; color: #2c3e50; font-size: 11pt;">

==========

KOMPUTASI ELEVASI MATAHARI UNTUK WAKTU SALAT (PYTHON)

Menggunakan pustaka Skyfield untuk presisi tinggi (Standar NASA JPL)

==========

```
from skyfield.api import load, wgs84

def hitung_posisi_matahari(lintang, bujur, tahun, bulan, hari, jam, menit):

# 1. Memuat Ephemeris DE421 (Database Koordinat Tata Surya)

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

# 2. Sinkronisasi Waktu Internasional (UTC)
ts = load.timescale()
waktu_observasi = ts.utc(tahun, bulan, hari, jam, menit)

# 3. Menetapkan Lokasi Pengamat (Berdasarkan Lintang &amp;amp; Bujur Geografis)
lokasi_pengamat = bumi + wgs84.latlon(lintang, bujur)

# 4. Observasi Posisi Apparent Matahari dari Titik Pengamat
astrometrik = lokasi_pengamat.at(waktu_observasi).observe(matahari)
apparent = astrometrik.apparent()

# 5. Mengekstrak Sudut Ketinggian (Altitude) dan Azimuth
alt, az, distance = apparent.altaz()

return alt.degrees, az.degrees
```

Contoh Eksekusi: Mengecek Ketinggian Matahari di Jakarta

(Lintang -6.20, Bujur 106.81) pada jam 12:00 UTC

```
ketinggian, azimuth = hitung_posisi_matahari(-6.20, 106.81, 2026, 3, 16, 12, 0)

print(f"Ketinggian Matahari: {ketinggian:.2f} derajat")
```

</pre>

Penjelasan Singkat Kode: Kode ini tidak sekadar menggunakan rumus trigonometri statis, melainkan memanggil topologi bumi yang presisi (`wgs84.latlon`) dan mencocokkannya dengan gerak dinamis bumi dan matahari melalui fungsi `.observe()`. Output `alt.degrees` (ketinggian) akan bernilai positif jika matahari di atas ufuk (siang hari/Zuhur/Asar) dan bernilai

negatif jika di bawah ufuk (Waktu Isya/Subuh). Berbekal logika dasar inilah, jadwal rincian kelima waktu salat di berbagai zona waktu dapat digenerasi secara akurat tanpa margin deviasi yang berarti.

2.3 Penentuan Arah Kiblat: Trigonometri Bola dan Geodesi Spasial Berbasis Komputasi

Fikih tata ruang dalam Islam menempatkan orientasi geografis sebagai prasyarat fundamental ibadah. Menghadap ke arah Ka'bah di Makkah (Kiblat) bukan sekadar simbolisme persatuan, melainkan kewajiban syar'i yang diamanatkan secara eksplisit oleh wahyu:

وَمِنْ حَيْثُ خَرَجْتَ فَوَلِّ وَجْهَكَ شَطْرَ الْمَسْجِدِ الْحَرَامِ ۖ وَحَيْثُ مَا كُنْتُمْ فَوَلُّوا وُجُوهَكُمْ شَطْرَهُ

"Dan dari mana saja engkau keluar (untuk mengerjakan salat), maka palingkanlah mukamu ke arah Masjidilharam. Dan di mana saja kamu berada, maka palingkanlah mukamu ke arahnya..." (QS. Al-Baqarah [2]: 150).

Secara historis, para sarjana Muslim klasik menggunakan observasi bayangan matahari (seperti metode *Rashdul Qibla* atau *Istiwa' A'zam*) dan instrumen astrolab untuk mengkalibrasi arah kiblat. Namun, dalam kajian geodesi modern, bumi dipahami bukan sebagai bidang datar, melainkan mendekati bentuk bola (elipsoida referensi). Oleh karena itu, penentuan arah terpendek menuju Ka'bah dari titik mana pun di permukaan bumi harus direpresentasikan menggunakan perhitungan Lingkaran Besar (*Great Circle*) dalam disiplin Trigonometri Bola (*Spherical Trigonometry*).

Di era pra-komputasi, perhitungan sudut azimuth kiblat memerlukan tabel logaritma yang rumit. Kompleksitas ini sering kali menghasilkan margin kesalahan yang membesar, terutama untuk lokasi geografis yang terlampau jauh dari Semenanjung Arabia, atau yang berada di lintang tinggi. Kesalahan sebesar satu derajat (1°) saja pada perhitungan awal dapat mengakibatkan pergeseran lintasan hingga ratusan kilometer pada titik akhir koordinat di Makkah (Amin, 2018).

Pergeseran Epistemologis Menuju Ekosistem Python

Kehadiran bahasa pemrograman Python memberikan solusi definitif atas kerentanan kalkulasi manual tersebut. Python memungkinkan para pakar falak dan *surveyor* pemetaan untuk mengeksekusi model matematika yang kompleks secara instan dengan presisi hingga tingkat *floating-point* (desimal tingkat tinggi). Terdapat dua landasan algoritma yang umumnya dikonversi ke dalam lingkungan Python:

1. **Geodesi Sferis Standar (Bola Sempurna):** Menggunakan rumus dasar trigonometri yang melibatkan garis lintang (φ) dan garis bujur (λ) dari posisi pengamat dan posisi Ka'bah (secara umum disepakati pada koordinat 21.4225° LU dan 39.8262° BT).
2. **Geodesi Elipsoida (WGS 84):** Menggunakan model bumi yang lebih presisi (menggelembung di ekuator dan pepat di kutub) melalui algoritma Vincenty atau pustaka khusus geodesi dalam Python seperti `pyproj` atau integrasi spasial pada `Skyfield`. Model ini mengakomodasi deviasi bentuk bumi sehingga akurasi azimuth mencapai tingkat absolut (Yildirim et al., 2024).

Dalam praksisnya, modul bawaan Python, yakni `math`, sudah lebih dari cukup untuk membangun mesin komputasi arah kiblat yang sangat akurat. Modul ini menyediakan fungsi trigonometri esensial seperti `math.sin()`, `math.cos()`, `math.tan()`, dan yang paling krusial, `math.atan2(y, x)` yang menangani konversi kuadran koordinat (Utara, Selatan, Timur, Barat) secara otomatis tanpa memerlukan struktur percabangan (`if-else`) yang terlalu rumit.

Berikut adalah purwarupa (*prototype*) implementasi *source code* Python untuk menghitung sudut azimuth kiblat dari titik pengamat mana pun di atas bumi:

<pre style="font-family: 'Courier New', Consolas, monospace; background-color: #f4f4f4; padding: 15px; border-radius: 5px; border: 1px solid #ccc; font-size: 11pt; color: #333;">

==========

ALGORITMA PENENTUAN AZIMUTH KIBLAT (METODE LINGKARAN BESAR)

Menggunakan Modul Math Bawaan Python

==========

```
import math

def kalkulasi_arah_kiblat(lintang_pengamat, bujur_pengamat):

# 1. Menetapkan Konstanta Koordinat Ka'bah (Makkah)

LINTANG_KAABAH = 21.4225 # Derajat Desimal (Utara)

BUJUR_KAABAH = 39.8262 # Derajat Desimal (Timur)

# 2. Mengonversi seluruh parameter derajat ke dalam format Radian
phi_1 = math.radians(lintang_pengamat)
lambda_1 = math.radians(bujur_pengamat)
phi_2 = math.radians(LINTANG_KAABAH)
lambda_2 = math.radians(BUJUR_KAABAH)

# 3. Menghitung diferensial (selisih) bujur geografis
delta_lambda = lambda_2 - lambda_1

# 4. Formulasi Trigonometri Bola untuk Komponen X dan Y
# Rumus:
# Y = sin(Δλ)
# X = cos(φ1) * tan(φ2) - sin(φ1) * cos(Δλ)
Y = math.sin(delta_lambda)
X = (math.cos(phi_1) * math.tan(phi_2)) - (math.sin(phi_1) *
math.cos(delta_lambda))

# 5. Komputasi nilai arctangent (atan2 otomatis menangani kuadran)
azimuth_radian = math.atan2(Y, X)

# 6. Mengonversi kembali hasil dari radian ke derajat
azimuth_derajat = math.degrees(azimuth_radian)
```

```
# 7. Normalisasi sudut kompas menjadi rentang 0 hingga 360 derajat
azimuth_kiblat = (azimuth_derajat + 360) % 360

return azimuth_kiblat
```

==========

UJI COBA EKSEKUSI KODE

Contoh: Titik Observasi di Jakarta, Indonesia (-6.2000 LU, 106.8166 BT)

==========

azimuth = kalkulasi_arah_kiblat(-6.2000, 106.8166)

print(f"Berdasarkan lintasan terpendek (Great Circle),")

print(f"Azimuth Kiblat untuk Jakarta adalah: {azimuth:.2f} derajat dari Utara Geografis.")

Output yang diharapkan: Sekitar 295.15 derajat (Barat Laut).

</pre>

Penjelasan Skema Kode:

Arsitektur kode di atas menyoroti efisiensi komputasional Python. Pada blok konversi (Langkah 2), semua koordinat geospasial wajib dikonversi dari satuan Derajat ke Radian menggunakan `math.radians()`, karena seluruh fungsi trigonometri dalam komputasi internal Python memproses *floating-point* berdasarkan nilai radian mutlak.

Pada Langkah 4 dan Langkah 5, pengolahan matriks geometri dipadatkan menggunakan `math.atan2(Y, X)`. Keistimewaan metode komputasi ini—dibandingkan kalkulator sains konvensional yang hanya menggunakan fungsi *arc-tangent* tunggal `atan()`—adalah kemampuannya mendeteksi di kuadran mana vektor arah bermuara. Python secara otonom membedakan apakah nilai Y atau X bernilai negatif atau positif, sehingga output azimuth bersifat absolut (mengacu pada Utara Sejati bumi di kutub utara geografis) tanpa perlu koreksi manual (Faid, Nahwandi, et al., 2022).

Setelah nilai azimuth numerik ini ditarik dari fungsi, para astronom dapat mengekstrapolasikannya lebih jauh. Misalnya, mereka dapat mengonversinya ke dalam format Derajat, Menit, dan Detik (DMS), merepresentasikannya ke dalam bentuk antarmuka grafis polar (Polar Plot) menggunakan pustaka `Matplotlib` untuk divisualisasikan dalam bentuk antarmuka kompas digital, atau membandingkannya dengan deviasi medan magnet bumi (*magnetic declination*) jika harus ditarik menggunakan instrumen kompas analog di lapangan (Asrin et al., 2018).

Kemampuan Python mengubah abstraksi matematis tingkat tinggi ini menjadi eksekusi algoritma yang ringkas, transparan, dan moduler menjadikannya instrumen modern yang sangat tak tergantikan untuk menjaga validitas hukum tata ruang ibadah umat Muslim secara berkelanjutan.

2.4 Penentuan Awal Bulan Hijriah: Pemodelan Visibilitas Hilal dan Presisi Ephemeris

A. Landasan Teologis dan Kompleksitas Fikih

Pilar ketiga dalam diskursus Astronomi Islam yang sangat bergantung pada kecanggihan komputasi adalah penentuan awal bulan kalender Hijriah (takwim kamariah). Kalender ini tidak sekadar berfungsi sebagai penunjuk tanggal sipil, melainkan penentu absah tidaknya ibadah-ibadah temporal (*muwaqqat*) berskala masif, seperti puasa Ramadan, perayaan Idulfitri, Iduladha, dan pelaksanaan wukuf di Arafah. Signifikansi bulan sabit (hilal) sebagai parameter waktu universal ditegaskan secara eksplisit oleh Allah *Subhanahu wa Ta'ala*:

يَسْـَٔلُونَكَ عَنِ ٱلْأَهِلَّةِ ۖ قُلْ هِىَ مَوَٰقِيتُ لِلنَّاسِ وَٱلْحَجِّ

"Mereka bertanya kepadamu tentang bulan sabit. Katakanlah: 'Bulan sabit itu adalah tanda-tanda waktu bagi manusia dan (bagi ibadat) haji...'" (QS. Al-Baqarah [2]: 189).

Secara operasional, Nabi Muhammad *Shallallahu 'alaihi wa sallam* memberikan pedoman metodologis yang bertumpu pada observasi visual (rukyat):

صُومُوا لِرُؤْيَتِهِ وَأَفْطِرُوا لِرُؤْيَتِهِ، فَإِنْ غُمَّ عَلَيْكُمْ فَأَكْمِلُوا عِدَّةَ شَعْبَانَ ثَلَاثِينَ

"Berpuasalah kalian karena melihatnya (hilal) dan berbukalah (berhari raya) karena melihatnya. Jika ia tertutup awan dari pandangan kalian, maka sempurnakanlah hitungan bulan Syakban menjadi tiga puluh hari." (HR. Bukhari, No. 1909 dan Muslim, No. 1081).

Dalam diskursus kontemporer, "melihat" (rukyat) telah mengalami perluasan makna epistemologis. Apakah rukyat harus menggunakan mata telanjang (*naked eye*), menggunakan alat bantu optik (teleskop), atau dapat digantikan dengan *hisab qat'i* (perhitungan matematis yang definitif)? Organisasi kerja sama Islam di berbagai kawasan merumuskan kriteria *Imkanur Rukyat* (probabilitas visibilitas bulan), seperti Kriteria MABIMS (Menteri Agama Brunei, Indonesia, Malaysia, dan Singapura) yang terbaru mensyaratkan elongasi minimal 6,4 derajat dan ketinggian minimal 3 derajat (Wahidi et al., 2019). Pemenuhan kriteria matematis yang presisi inilah yang menjadikan komputasi falak sangat krusial.

B. Kompleksitas Mekanika Lunar dan Urgensi Python

Berbeda dengan matahari yang pergerakannya relatif stabil (dapat didekati dengan formulasi deret Fourier sederhana), orbit bulan sangatlah kacau (*chaotic*) dan kompleks. Bulan mengalami perturbasi (gangguan gravitasi) yang luar biasa dari matahari, bentuk bumi yang tidak bulat sempurna (efek *oblateness* J2), serta gaya tarik dari planet-planet lain seperti Jupiter dan Venus. Untuk menghitung posisi bulan yang akurat, para astronom historis harus menyusun tabel dengan ribuan suku persamaan, seperti Teori Bulan Brown (*Brown's Lunar Theory*) atau ELP 2000-82B (Chapront-Touzé & Chapront, 1983).

Jika dihitung secara manual, penentuan fase konjungsi (Ijtimak) dan posisi *apparent* bulan saat matahari terbenam (*sunset*) akan memakan waktu berminggu-minggu dengan risiko *human error* yang besar. Variabel yang harus dihitung meliputi:

1. **Waktu Ijtimak Geometris (Geocentric Conjunction):** Saat bujur ekliptika bulan dan matahari bernilai sama persis.
2. **Ketinggian Hilal (Altitude/Irtifa'):** Jarak sudut bulan dari ufuk horizontal ke arah zenit sesaat setelah matahari terbenam. Ini harus mengikutsertakan refraksi atmosfer (pembiasan cahaya oleh atmosfer bumi yang membuat benda langit tampak lebih tinggi dari posisi aslinya), efek paralaks (karena bumi memiliki radius, pengamat di permukaan bumi melihat bulan dari sudut yang berbeda dibanding pusat bumi), dan faktor elevasi (ketinggian pengamat dari permukaan laut).
3. **Elongasi (Jarak Sudut):** Jarak busur absolut antara titik pusat matahari dan bulan. Sinar matahari yang mengenai bulan (fase iluminasi) sangat bergantung pada sudut ini (Danjon Limit mensyaratkan minimal sekitar 7 derajat agar sabit bulan dapat terbentuk secara fisis).
4. **Beda Azimuth (Relative Azimuth):** Jarak pada ufuk antara titik terbenam matahari dan proyeksi titik hilal.

Python mendisrupsi semua hambatan teknis ini. Dengan memanfaatkan pustaka astronomi modern, Python tidak lagi menggunakan pendekatan aproksimasi (perkiraan matematis), melainkan melakukan *numerical integration* yang menarik data ephemeris langsung dari *Jet Propulsion Laboratory* (JPL) NASA. Data ephemeris modern seperti DE430 atau DE440s mengintegrasikan data dari eksperimen *Lunar Laser Ranging* (penembakan laser ke cermin di bulan peninggalan misi Apollo), menghasilkan ketelitian hingga hitungan milimeter.

C. Implementasi Algoritmik dalam Python

Melalui integrasi pustaka `Skyfield`, para pengkaji astronomi Islam dapat membangun mesin komputasi hilal yang mampu mengeksekusi perhitungan untuk ribuan titik koordinat di seluruh dunia secara simultan. Berikut adalah contoh abstraksi *source code* komputasi visibilitas hilal saat fasa matahari terbenam, disajikan dalam tipografi *monospaced* untuk membedakan struktur kode:

<pre style="font-family: 'Courier New', Courier, monospace; background-color: #f4f4f4; padding: 15px; border-radius: 5px; border: 1px solid #ccc;">

==========

KOMPUTASI VISIBILITAS HILAL (KETINGGIAN & ELONGASI) SAAT SUNSET

Bahasa: Python (Menggunakan Library Skyfield)

==========

from skyfield.api import load, wgs84

from skyfield import almanac

def komputasi_hilal(lintang, bujur, tahun, bulan, hari):

1. Memuat Data Ephemeris dan Sinkronisasi Waktu

planets = load('de421.bsp')

bumi, matahari, bulan_geo = planets['earth'], planets['sun'], planets['moon']

ts = load.timescale()

```
# Menetapkan rentang waktu (Time Window) 24 jam untuk pencarian Sunset
t0 = ts.utc(tahun, bulan, hari, 0, 0)
t1 = ts.utc(tahun, bulan, hari + 1, 0, 0)

# 2. Menetapkan Titik Pengamat (Toposentris)
lokasi_pengamat = bumi + wgs84.latlon(lintang, bujur)

# 3. Mencari Waktu Matahari Terbenam (Sunset)
t, y = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(planets,
wgs84.latlon(lintang, bujur)))

# Ekstraksi waktu terbenam (nilai y=0 menandakan matahari terbenam)
waktu_sunset = None
for waktu, is_sunrise in zip(t, y):
    if not is_sunrise:
        waktu_sunset = waktu
        break

if waktu_sunset is None:
    return &quot;Tidak ada peristiwa Sunset di lintang ini pada tanggal
tersebut.&quot;

# 4. Komputasi Ketinggian Bulan saat Waktu Sunset
astrometrik_bulan = lokasi_pengamat.at(waktu_sunset).observe(bulan_geo)
apparent_bulan = astrometrik_bulan.apparent()

# Memasukkan koreksi refraksi atmosfer bumi
alt_bulan, az_bulan, distance = apparent_bulan.altaz(temperature_C=15,
pressure_mbar=1010)

# 5. Komputasi Elongasi (Jarak Sudut Bulan dan Matahari)
astrometrik_matahari                                              =
lokasi_pengamat.at(waktu_sunset).observe(matahari)
elongasi = astrometrik_matahari.separation_from(astrometrik_bulan)

return {
    &quot;Waktu Sunset (UTC)&quot;: waktu_sunset.utc_strftime(&#39;%Y-
%m-%d %H:%M:%S&#39;),
    &quot;Ketinggian Hilal (Derajat)&quot;: round(alt_bulan.degrees, 2),
    &quot;Azimuth Hilal (Derajat)&quot;: round(az_bulan.degrees, 2),
    &quot;Elongasi (Derajat)&quot;: round(elongasi.degrees, 2)
}
```

Contoh Eksekusi Pemantauan di Pos Observasi Pelabuhan Ratu, Indonesia

Lintang -7.025, Bujur 106.55, Tanggal 28 Februari 2026

hasil = komputasi_hilal(-7.025, 106.55, 2026, 2, 28)

for kunci, nilai in hasil.items():

print(f"{kunci}: {nilai}")

</pre>

Penjelasan Kode: Skrip Python di atas melambangkan "revolusi epistemologis" dalam hisab praktis. Perhatikan pada Langkah 3, fungsi `almanac.find_discrete()` secara iteratif (berulang) mencari waktu eksak kapan ujung piringan atas matahari menyentuh ufuk matematis. Setelah waktu tersebut (`waktu_sunset`) ditemukan, Python "menghentikan" waktu maya di alam semesta, lalu secara instan memutar instrumen virtualnya untuk membidik bulan (`observe(bulan_geo)`).

Pada Langkah 4, Python secara otomatis memperhitungkan pembiasan atmosfer (`temperature_C=15, pressure_mbar=1010`), yang mana secara tradisional membutuhkan interpolasi tabel yang sangat panjang. Parameter inilah yang kemudian disandingkan dengan kriteria syariat (seperti MABIMS yang mensyaratkan Ketinggian Hilal > 3 derajat dan Elongasi > 6,4 derajat) untuk memutuskan apakah keesokan harinya sah ditetapkan sebagai tanggal satu bulan yang baru atau tidak (Faid, Nawawi, et al., 2022).

Integrasi Python ini memberikan keluasan visual yang tak terbatas, di mana keluaran angka tersebut selanjutnya dapat digambar menggunakan pustaka grafis seperti `Matplotlib` untuk menghasilkan kurva visibilitas (peta global batas terlihatnya hilal menurut kriteria Yallop atau Odeh) yang selama ini menghiasi ruang sidang isbat kementerian agama (Ilyas, 1997; Odeh, 2004). Python menjadi jembatan antara kemutlakan dalil tekstual dengan ketelitian rasional empiris, mengukuhkan kembali identitas sains Islam yang analitis, dapat dipertanggungjawabkan (akuntabel), dan presisi.

Daftar Pustaka (Bab 2)

1. Amin, A. R. (2018). Akurasi Metode Falak Kontemporer dalam Penentuan Arah Kiblat. *Jurnal Hukum Islam*, 15(2), 112–129.
2. Chapront-Touzé, M., & Chapront, J. (1983). The lunar ephemeris ELP 2000. *Astronomy and Astrophysics*, 124, 50–62.
3. Faid, M. S., Mohd Nawawi, M. S. A., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
4. Faid, M. S., Nahwandi, M. S., Nawawi, M. S. A. B. M., Zaki, N. B. A., & Saadon, M. H. M. (2022). Development of Qibla direction determinant using sun shadow. *Online Journal of Research in Islamic Studies*, 9(1), 89–102.
5. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., Ahmad, N., & Ali @ Mat Zin, A. (2022). Islamic historical review on the middle age lunar crescent visibility criterion. *Journal of Al-Tamaddun*, 17(1).
6. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2023). Python applications in astronomical computations: A comparative analysis of Skyfield and PyEphem for Hijri calendar calculations. *Journal of Data Science and Computational Astronomy*, 5(2), 45-60.
7. Faid, M. S., et al. (2019). The computational algorithm of prayer times estimation for high latitude regions. *International Journal of Advanced Science and Technology*, 28(16).
8. Faid, M. S., et al. (2021). Optimization of solar declination formulas for Qibla direction algorithms in Islamic astronomy software. *Journal of Islamic Science and Technology*, 12(3), 210-225.

9. Ilyas, M. (1997). *Astronomy of Islamic calendar* (1st ed.). A. S. Nordeen.
10. Mehare, M., Sharma, S., & Gupta, R. (2023). A historical review of Python programming language evolution and its impact on computational sciences. *IEEE Transactions on Software Engineering*, 49(5), 1801–1815.
11. Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1), 39–64.
12. Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., & Myer-Tersigni, E. B. (2021). MatLab vs. Python vs. R. *Journal of Data Science*, 15(3), 355–372.
13. Rhodes, B. (2011). PyEphem: Astronomical Ephemeris for Python. *Astrophysics Source Code Library*, record ascl:1112.014.
14. Wahidi, A., Yasin, N., & Kadarisman, A. (2019). The beginning of Islamic months determination in Indonesia and Malaysia: Procedure and social condition. *ULUL ALBAB Jurnal Studi Islam*, 20(2), 322–345.
15. Yildirim, F., Kadi, F., & Sahin, S. L. (2024). Developing a new interface for Qibla direction application based on MATLAB GUI. *Survey Review*.

BAB 3 PENTINGNYA AKURASI PERHITUNGAN

3.1 Pendahuluan dan Penentuan Praktik Keagamaan yang Akurat

(A) Landasan Teologis dan Presisi Epistemologis Dalam arsitektur hukum Islam (fikih), akurasi komputasi tidak hanya dipandang sebagai standar metodologi sains modern, melainkan sebagai prasyarat mutlak bagi sahnya sebuah ibadah fardu (syarat sah). Penentuan posisi geometris matahari dan bulan secara presisi berdampak langsung dan absolut terhadap penentuan awal waktu salat, arah orientasi kiblat, serta visibilitas hilal untuk kalender Hijriah. Allah *Subhanahu wa Ta'ala* secara eksplisit menegaskan bahwa pergerakan benda-benda langit diciptakan dengan arsitektur matematis agar manusia dapat melakukan observasi dan komputasi (hisab) yang akurat:

هُوَ الَّذِيْ جَعَلَ الشَّمْسَ ضِيَاۤءً وَّالْقَمَرَ نُوْرًا وَّقَدَّرَهٗ مَنَازِلَ لِتَعْلَمُوْا عَدَدَ السِّنِيْنَ وَالْحِسَابَۗ مَا خَلَقَ اللّٰهُ ذٰلِكَ اِلَّا بِالْحَقِّۗ يُفَصِّلُ الْاٰيٰتِ لِقَوْمٍ يَّعْلَمُوْنَ

"Dialah yang menjadikan matahari bersinar dan bulan bercahaya, dan Dialah yang menetapkan tempat-tempat orbitnya, agar kamu mengetahui bilangan tahun, dan perhitungan (waktu). Allah tidak menciptakan demikian itu melainkan dengan benar. Dia menjelaskan tanda-tanda (kebesaran-Nya) kepada orang-orang yang mengetahui." (QS. Yunus [10]: 5).

Ayat ini memberikan legitimasi teologis bagi pengembangan ilmu falak. Kesalahan dalam perhitungan astronomi—sekecil apa pun margin deviasinya—dapat berujung pada pelaksanaan ibadah di luar waktu atau arah yang disyariatkan. Jika perhitungan tidak akurat, hal ini akan mengarah pada anomali jadwal waktu salat, penyimpangan arah kiblat, atau penetapan awal bulan Islam yang keliru. Oleh karena itu, bagi lembaga-lembaga otoritatif negara, pemanfaatan metode perhitungan yang paling mutakhir dan presisi adalah sebuah kewajiban institusional demi menjaga keabsahan ibadah umat.

Di kawasan Asia Tenggara, institusi publik seperti Departemen Kemajuan Islam Malaysia (JAKIM), Jabatan Ukur dan Pemetaan Malaysia (JUPEM), serta Departemen Mufti Negeri (sejalan dengan Badan Hisab Rukyat Kementerian Agama di Indonesia) memikul tanggung jawab krusial dalam memproduksi data astronomis tersebut. Sebagai contoh praktis, arah kiblat dari seluruh masjid dan surau wajib dikalibrasi dan disertifikasi oleh *surveyor* resmi serta Departemen Mufti untuk memastikan komunitas Muslim beribadah dengan arah orientasi yang valid secara spasial. Demikian pula, perilisan Jadwal Waktu Salat dan Kalender Hijriah tahunan melibatkan proses komputasi yang sangat ketat, yang umumnya harus melalui validasi dari Dewan Falak Negara sebelum dipublikasikan kepada masyarakat luas.

Bagi masyarakat awam, mereka memang tidak dibebani kewajiban komputasional untuk melakukan perhitungan astronomis yang rumit ini secara mandiri. Sebaliknya, umat didorong untuk merujuk pada sumber-sumber otoritatif guna memperoleh informasi yang tervalidasi. Namun demikian, meningkatkan kesadaran dan literasi publik mengenai proses komputasi astronomi Islam memiliki dampak empiris yang sangat positif. Literasi sains ini akan meningkatkan apresiasi umat terhadap warisan intelektual Islam, sekaligus mereduksi kebingungan massal dan memotong rantai penyebaran misinformasi (hoaks) yang acap kali bermunculan menjelang sidang isbat penetapan awal bulan.

(B) Signifikansi Penentuan Waktu Praktik Keagamaan Akurasi dalam astronomi Islam menempati tingkat urgensi yang paling tinggi, secara khusus dalam konteks penentuan fase awal bulan Hijriah yang baru. Hal ini disebabkan oleh rupa-ragam faktor yang memiliki dampak langsung terhadap validitas praktik keagamaan miliaran umat Muslim di seluruh dunia, selain juga membawa implikasi sosial dan ekonomi yang luas.

Penentuan awal bulan Hijriah secara hakiki bertindak sebagai parameter yang menetapkan tanggal-tanggal untuk ibadah temporal berskala besar, seperti kewajiban berpuasa di bulan Ramadan, perayaan Idulfitri (1 Syawal), serta pelaksanaan Hari Raya Iduladha dan rangkaian ibadah Haji (Zulhijah). Akurasi matematis dalam menetapkan bulan Hijriah ini adalah instrumen penjamin bahwa praktik fundamental, seperti puasa dan salat Id, dilaksanakan persis pada hari yang sah menurut hukum syariat.

Dalam dinamika mekanika selestial, orbit bulan sangatlah kompleks dan rentan terhadap gangguan gravitasi. Deviasi (penyimpangan observasi) yang terlihat sangat kecil—bahkan sekecil 0,5 derajat saja dalam kalkulasi posisi bulan—dapat mendistorsi model prediksi visibilitas secara drastis. Kesalahan sebesar 0,5 derajat dalam menghitung sudut elongasi atau ketinggian (*altitude*) hilal berpotensi mengubah status penetapan tanggal secara keseluruhan, yang pada gilirannya memicu kebingungan struktural, friksi, dan perpecahan (*ikhtilaf*) di dalam tubuh komunitas masyarakat.

(C) Integrasi Python dalam Mengeliminasi Deviasi Dalam ekosistem ilmu falak kontemporer, bahasa pemrograman Python mengambil peran sebagai instrumen vital untuk mengamankan presisi perhitungan tersebut. Python, melalui dukungan pustaka sains modernnya (seperti *Skyfield* atau *Astropy*), tidak lagi bergantung pada tabel ekstrapolasi manual, melainkan mengeksekusi integrasi numerik presisi tinggi (*floating-point*). Python mampu memproses ribuan variabel data pergerakan hilal secara efisien untuk memastikan praktik-praktik Islam dapat diamati secara benar dan konsisten dari waktu ke waktu.

Sebagai gambaran konseptual, di bawah ini disajikan *source code* Python yang mendemonstrasikan betapa krusialnya tingkat ketelitian desimal tingkat tinggi (*floating-point precision*) dalam menyeleksi dan memvalidasi kriteria visibilitas bulan:

<pre style="font-family: 'Courier New', Courier, monospace; background-color: #f4f4f4; padding: 15px; border-radius: 5px; border: 1px solid #ccc;">

==========

SIMULASI PRESISI FLOATING-POINT DALAM VALIDASI KETINGGIAN HILAL

Python mengeliminasi kesalahan deviasi pembulatan (rounding error)

yang kerap terjadi pada kalkulator manual.

==========

def validasi_kriteria_imkanur_rukyat(ketinggian_komputasi):

Mengacu pada standar kriteria minimal absolut (Contoh: 3.0 derajat)

BATAS_MINIMAL_KETINGGIAN = 3.0

```
# Menampilkan nilai input observasi dengan presisi hingga 10 desimal
print(f&amp;quot;Data Ketinggian Hilal Diterima: {ketinggian_komputasi:.10f}
derajat&amp;quot;)

# Sistem mengevaluasi selisih mikro tanpa pembulatan prematur
if ketinggian_komputasi &amp;gt;= BATAS_MINIMAL_KETINGGIAN:
    return &amp;quot;Status: MEMENUHI KRITERIA (Hilal secara fisis dapat
diamati)\n&amp;quot;
else:
    return &amp;quot;Status: TIDAK MEMENUHI (Hilal berada di bawah batas
visibilitas)\n&amp;quot;
```

Skenario A: Perhitungan manual sering membulatkan 2.99 derajat menjadi 3.0

print("--- SKENARIO A (Risiko Pembulatan Kasar) ---")

ketinggian_kasar = round(2.9985000000, 1) # Dibulatkan menjadi 3.0

print(validasi_kriteria_imkanur_rukyat(ketinggian_kasar))

Skenario B: Python membaca data mentah astronomis secara eksak

print("--- SKENARIO B (Akurasi Presisi Python) ---")

ketinggian_presisi = 2.9985000000

print(validasi_kriteria_imkanur_rukyat(ketinggian_pre**sisi))**

KESIMPULAN ANALISIS:

Pada Skenario B, Python mendeteksi bahwa hilal kurang 0.0015 derajat

dari kriteria sah. Presisi ini sukses mencegah penetapan awal bulan yang prematur,

menghindari deviasi 0.5 derajat yang sangat fatal dalam kalender Hijriah.

</pre>

Melalui kekuatan pemrograman komputasional di atas, otoritas keagamaan mampu memformulasikan kebijakan yang sepenuhnya bersandar pada parameter objektif, empiris, dan akuntabel, membebaskan prosedur penetapan syariat dari risiko kesalahan hitung manusia (*human error*).

3.2 Dampak Sosial dan Ekonomi

(A) Kohesi Sosial dan Mitigasi Friksi Komunal

Secara sepintas, perdebatan mengenai akurasi perhitungan letak hilal atau menit presisi waktu salat tampak seperti diskursus eksklusif di menara gading akademik para astronom. Namun, pada realitas sosiologisnya, output dari perhitungan *'Ilm al-Falak* memiliki implikasi langsung terhadap kohesi sosial dan stabilitas psikologis massa. Dalam yurisprudensi Islam, ibadah puasa dan perayaan Idulfitri memiliki dimensi komunal yang sangat kuat. Perbedaan dalam menetapkan hari pertama Ramadan atau 1 Syawal kerap kali memicu friksi, segregasi, dan polarisasi di tengah masyarakat Muslim, bahkan dalam satu negara atau satu garis keturunan keluarga yang sama.

Allah *Subhanahu wa Ta'ala* secara tegas memerintahkan umat Islam untuk memelihara persatuan dan menghindari perpecahan, yang secara sosiologis dapat dicapai melalui standardisasi dan kepatuhan pada sistem yang disepakati (konsensus otoritatif):

وَاعْتَصِمُوْا بِحَبْلِ اللّٰهِ جَمِيْعًا وَّلَا تَفَرَّقُوْا ۚوَاذْكُرُوْا نِعْمَتَ اللّٰهِ عَلَيْكُمْ اِذْ كُنْتُمْ اَعْدَاۤءً فَاَلَّفَ بَيْنَ قُلُوْبِكُمْ فَاَصْبَحْتُمْ بِنِعْمَتِهٖٓ اِخْوَانًاۚ

"Dan berpegangteguhlah kamu semuanya pada tali (agama) Allah, dan janganlah kamu bercerai berai, dan ingatlah nikmat Allah kepadamu ketika kamu dahulu (masa Jahiliah) bermusuhan, lalu Allah mempersatukan hatimu, sehingga dengan karunia-Nya kamu menjadi bersaudara..." (QS. Ali 'Imran [3]: 103).

Dalam konteks hisab dan rukyat, perpecahan (*ikhtilaf*) sering kali bermula dari penggunaan parameter komputasi yang usang, bias observasi, atau ketiadaan algoritma prediktif yang dapat menjembatani kriteria visibilitas yang berbeda. Di sinilah akurasi komputasi modern mengambil peran sosiologisnya. Dengan menyajikan data yang presisi—misalnya probabilitas elongasi dan ketinggian hilal yang dihitung dengan tingkat akurasi desimal tingkat tinggi—otoritas keagamaan (Pemerintah/Kementerian Agama) memiliki landasan epistemologis yang kuat untuk mengambil keputusan (Sidang Isbat) yang mengikat dan meredam potensi konflik horisontal.

(B) Dimensi Makroekonomi dan Kepastian Logistik Global

Selain dampak sosial, akurasi penanggalan Hijriah memiliki resonansi ekonomi yang sangat masif, khususnya bagi negara-negara anggota Organisasi Kerja Sama Islam (OKI) atau negara dengan populasi Muslim yang signifikan. Di era ekonomi global yang terhubung (*hyper-connected economy*), ketidakpastian tanggal hari libur nasional (seperti Idulfitri dan Iduladha) menimbulkan anomali finansial yang bernilai triliunan rupiah.

Pertama, dari sektor perbankan dan keuangan syariah. Kontrak-kontrak derivatif syariah, pencairan instrumen Sukuk (obligasi syariah), hingga perhitungan *haul* (siklus tahunan) untuk kewajiban Zakat korporasi, sangat bergantung pada presisi kalender Hijriah. Pergeseran satu hari saja akibat kesalahan komputasi bulan baru dapat mendistorsi perhitungan margin keuntungan bank, penalti keterlambatan, dan neraca likuiditas harian lembaga keuangan.

Kedua, sektor logistik, penerbangan, dan *supply chain* (rantai pasok). Menjelang hari raya, terjadi lonjakan konsumsi publik dan eksodus pergerakan manusia (*mudik*). Maskapai penerbangan, operator kereta api, dan manajemen jalan tol memerlukan jadwal pasti berbulan-bulan sebelumnya untuk mengalokasikan armada tambahan dan memetakan puncak arus logistik. Jika penentuan hari raya meleset dari kalender sipil karena perhitungan astronomi yang kurang presisi di awal tahun, dampaknya adalah kekacauan jadwal (*rescheduling*), pembatalan penerbangan massal, hilangnya hari kerja produktif di sektor industri, hingga

disrupsi distribusi komoditas pangan. Kepastian algoritma komputasi falak memberikan stabilitas bagi perencanaan makroekonomi ini.

(C) Pemodelan Prediktif Sosio-Ekonomi Menggunakan Python

Untuk memitigasi risiko disrupsi ekonomi tersebut, bahasa pemrograman Python dapat digunakan tidak hanya untuk menghitung posisi benda langit, tetapi juga untuk membangun arsitektur sistem peringatan dini (*early warning system*) bagi kalender sosio-ekonomi. Dengan mengawinkan pustaka astronomi (`skyfield`) dan pustaka analisis data (`pandas`), para pakar falak dapat menyusun kalender prediktif jangka panjang yang mendeteksi "zona abu-abu" (hari-hari di mana posisi hilal berada persis di ambang batas kriteria visibilitas), sehingga pemerintah dapat menyiapkan skenario ekonomi cadangan.

Berikut adalah ilustrasi *source code* Python yang mendemonstrasikan bagaimana komputasi astronomi dikonversi menjadi data proyeksi kalender ekonomi untuk kepentingan logistik nasional:

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f0f4f8; padding: 15px; border-radius: 8px; border: 1px solid #c8d8e4; color: #102a43; font-size: 11pt;">

==========

PEMODELAN PREDIKTIF KALENDER SOSIO-EKONOMI (PYTHON)

Mengidentifikasi anomali tanggal Idulfitri untuk perencanaan logistik

==========

import pandas as pd

def analisis_risiko_kalender_ekonomi(data_prediksi_hilal):

print("--- ANALISIS RISIKO DISRUPSI HARI LIBUR NASIONAL ---")

laporan_risiko = []

```
# Kriteria MABIMS Baru (Ketinggian 3 derajat, Elongasi 6.4 derajat)
BATAS_TINGGI = 3.0
BATAS_ELONGASI = 6.4

for data in data_prediksi_hilal:
    tahun = data[&#39;Tahun_Masehi&#39;]
    tinggi = data[&#39;Ketinggian_Hilal&#39;]
    elongasi = data[&#39;Elongasi&#39;]

    # Deteksi &quot;Zona Kritis&quot; (Margin error kurang dari 0.5 derajat dari batas)
    selisih_tinggi = tinggi - BATAS_TINGGI
    selisih_elongasi = elongasi - BATAS_ELONGASI

    status = &quot;AMAN (Kepastian Logistik Tinggi)&quot;
```

```
    tindakan = &quot;Normal&quot;

    if (0 &lt;= selisih_tinggi &lt;= 0.5) or (0 &lt;= selisih_elongasi
&lt;= 0.5):
        status = &quot;ZONA KRITIS (Potensi Perbedaan Hari Raya)&quot;
        tindakan = &quot;Siapkan Skenario Libur Ganda &amp; Buffer Stok
Pangan&quot;
    elif tinggi &lt; BATAS_TINGGI or elongasi &lt; BATAS_ELONGASI:
        status = &quot;TIDAK TERLIHAT (Bulan digenapkan 30 hari)&quot;
        tindakan = &quot;Undur Libur Nasional 1 Hari&quot;

    laporan_risiko.append({
        &quot;Tahun&quot;: tahun,
        &quot;Tinggi (deg)&quot;: tinggi,
        &quot;Elongasi (deg)&quot;: elongasi,
        &quot;Status Sosial&quot;: status,
        &quot;Rekomendasi Ekonomi&quot;: tindakan
    })

# Memvisualisasikan data ke dalam Dataframe berskala enterprise
df_laporan = pd.DataFrame(laporan_risiko)
return df_laporan
```

Simulasi Input Data Astronomis Jangka Panjang (Misal: 2026 - 2028)

Data ini pada realitasnya ditarik langsung dari fungsi Skyfield JPL

data_hilal_syawal = [

{'Tahun_Masehi': 2026, 'Ketinggian_Hilal': 4.2, 'Elongasi': 7.1},

{'Tahun_Masehi': 2027, 'Ketinggian_Hilal': 3.1, 'Elongasi': 6.5}, # Zona Kritis

{'Tahun_Masehi': 2028, 'Ketinggian_Hilal': 1.5, 'Elongasi': 4.2}

]

Eksekusi dan Tampilkan Analisis

tabel_keputusan = analisis_risiko_kalender_ekonomi(data_hilal_syawal)

print(tabel_keputusan.to_string(index=False))

</pre>

Analisis Output Kode: Dalam simulasi sistem di atas, Python bertindak sebagai integrator antara sains murni dan kebijakan publik. Pada data tahun 2027, algoritma mendeteksi bahwa parameter hilal (Ketinggian 3.1° dan Elongasi 6.5°) berada sangat dekat dengan batas minimal (*threshold*) kriteria. Jika cuaca lokal buruk atau ada bias refraksi atmosfer, hilal mungkin tidak terlihat, sehingga penetapan Idulfitri berpotensi bergeser. Sistem Python secara cerdas melabeli tahun tersebut sebagai **"ZONA KRITIS"** dan secara otomatis mengeluarkan rekomendasi **"Siapkan Skenario Libur Ganda & Buffer Stok Pangan"**.

Dengan memiliki alat komputasi proaktif semacam ini, negara tidak lagi bertindak reaktif pada H-1 (saat Sidang Isbat), melainkan dapat merancang mitigasi sosio-ekonomi berbulan-bulan sebelumnya. Arus distribusi ritel dapat diamankan, pasar modal dapat menyesuaikan waktu penutupan *bursa efek*, dan polarisasi di masyarakat dapat direduksi melalui edukasi publik yang terstruktur berdasarkan data kuantitatif yang transparan.

3.3 Pentingnya Teknologi dan Perhitungan Modern

Dalam diskursus ilmu falak kontemporer, integrasi teknologi modern bukan lagi sekadar alat bantu opsional, melainkan sebuah keharusan epistemologis. Penggunaan teknologi mutakhir, khususnya melalui bahasa pemrograman seperti Python yang dilengkapi dengan pustaka astronomi canggih, memungkinkan otoritas keagamaan dan ilmuwan untuk mencapai tingkat akurasi serta konsistensi yang belum pernah terbayangkan di era pra-komputasi. Hal ini sangat krusial, terutama dalam konteks penentuan awal bulan Hijriah yang memiliki sensitivitas tinggi terhadap validitas ibadah.

(A) Mengurai Variabel Kompleks dalam Komputasi Falak

Metode hisab tradisional sering kali mengasumsikan model bumi yang ideal (seperti bola sempurna) dan terkadang mengabaikan anomali-anomali mikroskopis di atmosfer. Namun, dalam komputasi astronomi modern, penentuan visibilitas hilal (bulan sabit baru) tidak hanya bergantung pada koordinat geometris dua dimensi. Perhitungan ini melibatkan matriks variabel astronomis dan geofisika yang sangat dinamis dan kompleks. Di antaranya adalah penelusuran posisi tiga dimensi (3D) secara terus-menerus dari bulan, matahari, dan bumi pada ruang interaksinya di tata surya.

Lebih jauh lagi, kemampuan pemrosesan teknologi modern memungkinkan para astronom untuk memasukkan faktor-faktor eksternal yang secara empiris memengaruhi sukses tidaknya sebuah observasi (rukyat) di lapangan. Sebagaimana diartikulasikan oleh Junaidi (2022), penentuan awal bulan Hijriah secara hakiki tidak dapat dilepaskan dari perhitungan atas variabel kondisi cuaca lokal, profil suhu dan tekanan atmosfer yang memengaruhi indeks refraksi (pembiasan cahaya), hingga kondisi topografi spesifik dari lokasi observasi (seperti ketinggian dari permukaan laut dan halangan fisik di ufuk). Melalui pemrograman, seluruh variabel alam ini dapat dimodelkan ke dalam satu arsitektur algoritma yang komprehensif, memungkinkan simulasi prediksi visibilitas hilal yang sangat mendekati kondisi nyata.

(B) Reduksi Margin Kesalahan (*Margin of Error*) Melalui Algoritma

Kelemahan esensial dari perhitungan manual menggunakan tabel logaritma tradisional (*zīj*) adalah kerentanannya terhadap kesalahan pembulatan matematis (*truncation/rounding error*) yang terus terakumulasi pada setiap tahap eksekusi rumus. Dengan memanfaatkan algoritma lanjutan dan kalkulasi matematika presisi tingkat tinggi (*floating-point arithmetic* 64-bit) yang disediakan oleh lingkungan Python, probabilitas terjadinya kesalahan komputasional dalam menentukan fase-fase kritis—seperti menit pasti terjadinya *ijtimak* (konjungsi) atau presisi sudut elongasi—dapat direduksi secara signifikan.

(C) Kepercayaan Global dan Validitas Fikih

Akurasi komputasional ini pada akhirnya bermuara pada satu tujuan fundamental: memberikan jaminan kepastian hukum syariat (fikih) bagi umat Islam. Tingkat presisi yang tinggi, sebagaimana dikemukakan oleh Gharaybeh (2025), merupakan instrumen yang sangat esensial untuk memastikan bahwa jutaan umat Islam di berbagai zona waktu dunia dapat melaksanakan praktik keagamaan mereka dengan penuh keyakinan. Ketika jadwal puasa Ramadan, penetapan hari raya, dan waktu salat harian didasarkan pada fondasi perhitungan yang valid secara saintifik dan bebas dari galat komputasi, hal tersebut akan mengeliminasi keraguan (*syak*) dalam hati umat, menjamin bahwa tanggal yang mereka patuhi adalah absah dan benar.

(D) Implementasi Pemodelan Parameter Topografi dalam Python

Untuk memberikan ilustrasi nyata mengenai bagaimana teknologi modern mengintegrasikan variabel topografi dan atmosfer, berikut adalah purwarupa *source code* Python. Kode ini mendemonstrasikan perhitungan penurunan ufuk (*dip of horizon*) dan refraksi atmosfer—dua variabel yang sangat menentukan apakah hilal benar-benar sudah berada di atas ufuk pengamat atau belum:

<pre style="font-family: 'Courier New', Courier, monospace; background-color: #f8f9fa; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; color: #0f172a; font-size: 11pt;">

==========

INTEGRASI PARAMETER TOPOGRAFI & ATMOSFER (PYTHON - SKYFIELD)

Menghitung Koreksi Refraksi dan Sudut Depresi Ufuk

==========

from skyfield.earthlib import refraction

import math

def koreksi_observasi_lokal(elevasi_meter, suhu_celsius, tekanan_mbar):

1. Konstanta Radius Bumi Geometris (dalam meter)

RADIUS_BUMI = 6378136.6

```
# 2. Menghitung Sudut Kemiringan Ufuk (Dip of Horizon) akibat Elevasi
# Semakin tinggi pengamat, ufuk sejati akan tampak semakin turun
rasio = RADIUS_BUMI / (RADIUS_BUMI + elevasi_meter)
sudut_horizon_radian = math.acos(rasio)
sudut_horizon_derajat = math.degrees(sudut_horizon_radian)

# 3. Menghitung Koreksi Refraksi Atmosfer di Ufuk
# Suhu dan tekanan lokal sangat memengaruhi kelengkungan cahaya bulan
refraksi_ufuk     =     refraction(0.0,     temperature_C=suhu_celsius,
pressure_mbar=tekanan_mbar)
```

```
print(f&amp;quot;--- ANALISIS PARAMETER GEOFISIKA OBSERVASI ---&amp;quot;)
print(f&amp;quot;Ketinggian Pengamat   : {elevasi_meter} mdpl&amp;quot;)
print(f&amp;quot;Penurunan  Ufuk  Fisis     :  -{sudut_horizon_derajat:.4f}
derajat&amp;quot;)
print(f&amp;quot;Nilai Refraksi Cahaya : +{refraksi_ufuk:.4f} derajat&amp;quot;)

return sudut_horizon_derajat, refraksi_ufuk
```

Simulasi Eksekusi: Pengamat di Menara Masjid dengan ketinggian 150 meter, Suhu udara tropis 28 derajat Celsius, Tekanan barometrik 1010 milibar.

dip, ref = koreksi_observasi_lokal(150, 28, 1010)

</pre>

Analisis Konseptual Kode: Dalam blok algoritma di atas, fungsi `koreksi_observasi_lokal` mewakili keunggulan integrasi sains modern. Kalkulasi geometri *Dip of Horizon* (Langkah 2) mengakomodasi realitas fisik bahwa ufuk yang terlihat oleh pengamat di pantai (0 mdpl) akan sangat berbeda sudutnya dengan pengamat yang berada di pegunungan tinggi atau menara pencakar langit. Selanjutnya, pemanggilan fungsi `refraction()` (Langkah 3) mengkalkulasi secara instan bagaimana kepadatan udara membiaskan foton cahaya bulan. Pemrosesan variabel dinamis semacam inilah yang mustahil dilakukan secara *real-time* dengan ketelitian desimal tingkat tinggi tanpa bantuan teknologi perangkat lunak masa kini, sehingga meneguhkan Python sebagai elemen yang tak terpisahkan dari ilmu falak modern.

3.4 Kompleksitas Perhitungan dalam Mekanika Selestial

(A) Landasan Filosofis dan Realitas Fisis Benda Langit

Di balik layar aplikasi jadwal salat yang tampak sederhana di telepon pintar kita, terdapat sebuah arsitektur komputasi matematika yang luar biasa kompleks. Allah *Subhanahu wa Ta'ala* merancang alam semesta bukan sebagai entitas statis, melainkan sebuah sistem dinamis yang saling memengaruhi melalui hukum-hukum fisika dan gravitasi. Harmoni gerak kosmik ini diabadikan dalam firman-Nya:

وَهُوَ الَّذِيْ خَلَقَ الَّيْلَ وَالنَّهَارَ وَالشَّمْسَ وَالْقَمَرَۗ كُلٌّ فِيْ فَلَكٍ يَّسْبَحُوْنَ

"Dan Dialah yang telah menciptakan malam dan siang, matahari dan bulan. Masing-masing beredar pada garis edarnya." (QS. Al-Anbiya [21]: 33).

Kata *yasbahun* (يَسْبَحُوْنَ) yang secara harfiah bermakna "berenang" atau "mengambang dengan lancar", secara saintifik mengisyaratkan bahwa benda-benda langit beredar di ruang hampa yang memiliki hambatan, mengikuti lintasan elips yang tidak kaku. Dalam kacamata mekanika benda langit (*celestial mechanics*), kompleksitas perhitungan astronomi Islam muncul karena bumi, bulan, dan matahari tidak bergerak dalam kecepatan dan lintasan yang konstan.

(B) Variabel Perturbasi dan Anomali Orbit

Apabila bumi dan matahari adalah satu-satunya benda di alam semesta, orbit bumi akan menjadi elips yang sempurna (sesuai Hukum Kepler) dan perhitungan astronomi dapat diselesaikan dengan rumus aljabar linear tingkat dasar. Namun, realitas empiris menunjukkan adanya **Perturbasi** (gangguan gravitasi). Bumi ditarik oleh gravitasi planet-planet raksasa seperti Jupiter dan Saturnus, yang menyebabkan orbit bumi sedikit bergeser (presesi apsidal).

Bagi bulan, situasinya jauh lebih kacau (*chaotic*). Karena jaraknya yang relatif dekat dengan bumi namun juga sangat dipengaruhi oleh tarikan gravitasi matahari secara simultan, lintasan bulan mengalami distorsi yang ekstrem. Dalam hisab klasik, anomali orbit bulan ini dijabarkan melalui ratusan persamaan koreksi (seperti koreksi *Evection*, *Variation*, dan *Annual Equation*). Selain itu, bidang orbit bulan miring sekitar 5,14 derajat terhadap bidang ekliptika (orbit bumi), dan titik potongnya (titik *node*) terus bergeser mundur sejauh 19 derajat setiap tahunnya (Siklus Saros). Variabel-variabel inilah yang membuat perhitungan *Ijtimak* (konjungsi) menjadi subjek komputasi tingkat tinggi.

(C) Dinamika Waktu: Delta T (ΔT) dan Perlambatan Bumi

Kompleksitas perhitungan falak tidak berhenti pada geometri spasial, melainkan juga merambah pada dimensi waktu. Satuan detik yang kita gunakan sehari-hari pada jam tangan (Universal Time / UT) didasarkan pada rotasi bumi. Sayangnya, akibat gaya pasang surut air laut (*tidal friction*) yang ditimbulkan oleh interaksi bumi-bulan, rotasi bumi semakin melambat setiap abadnya.

Di sisi lain, parameter astronomis (seperti data posisi planet dari NASA) dihitung menggunakan skala waktu yang seragam dan absolut, yang disebut *Terrestrial Time* (TT) atau waktu dinamis yang berbasis pada jam atomik (*International Atomic Time* / TAI). Untuk mensinkronkan perhitungan gerak benda langit dengan jam di bumi, astronom harus menggunakan variabel yang disebut **Delta T (ΔT)**.

Formula dasarnya adalah:

$$\Delta T = TT - UT1$$

Nilai Delta T terus berubah setiap tahun dan tidak dapat diprediksi secara matematis absolut di masa depan, melainkan harus diukur secara empiris. Mengabaikan nilai Delta T dalam komputasi astronomi Islam dapat menyebabkan kesalahan penentuan waktu gerhana hingga hitungan menit, dan kesalahan penentuan posisi hilal yang berakibat fatal pada keabsahan penetapan kalender Hijriah.

(D) Penanganan Kompleksitas Melalui Integrasi Numerik Python

Dalam pemrograman Python, kerumitan mekanika selestial (seperti Perturbasi, Nutasi, Presesi, dan Delta T) didelegasikan kepada pustaka tingkat rendah (*low-level libraries*) yang memproses puluhan ribu baris data secara instan. Python menggunakan metode **Integrasi Numerik**, yakni menghitung posisi benda langit langkah demi langkah berdasarkan hukum

gravitasi Newton dan Relativitas Umum Einstein, yang ditarik langsung dari berkas Ephemeris JPL NASA.

Berikut adalah *source code* Python yang memperlihatkan bagaimana Python menangani kompleksitas skala waktu (Delta T) tanpa memaksa pengguna menulis ulang fungsi interpolasi polinomial secara manual:

<div style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #555; font-size: 11pt; line-height: 1.5; white-space: pre-wrap;">

==========

ANALISIS KOMPLEKSITAS WAKTU: IMPLEMENTASI DELTA T DALAM PYTHON

Python menangani koreksi perlambatan rotasi bumi secara otomatis.

==========

from skyfield.api import load

def analisis_skala_waktu_astronomi(tahun, bulan, hari):

Memuat objek timescale (yang secara otomatis memuat tabel Delta T terbaru)

ts = load.timescale()

```
# Menciptakan objek waktu berdasarkan kalender sipil pengamat (UTC)
waktu_observasi = ts.utc(tahun, bulan, hari)

# Ekstraksi nilai komputasi
waktu_sipil = waktu_observasi.utc_strftime('%Y-%m-%d %H:%M:%S UTC')
waktu_dinamis = waktu_observasi.tt
delta_t = waktu_observasi.delta_t

print(f"--- LAPORAN KOMPLEKSITAS SKALA WAKTU ---")
print(f"Tanggal Observasi Sipil (UT1/UTC) : {waktu_sipil}")
print(f"Waktu Atomik Terestrial (TT)      : {waktu_dinamis:.5f} (Julian
Date)")
print(f"Koreksi Delta T (Perlambatan Bumi): +{delta_t:.3f} detik")
print(f"Analisis: Untuk mencapai presisi ephemeris, sistem Python telah")
print(f"menambahkan {delta_t:.3f} detik ke dalam jam sipil sebelum
melakukan")
print(f"kalkulasi posisi lintasan Bulan dan Matahari.")
```

Eksekusi simulasi untuk pengamatan Hilal awal Ramadan 2026

analisis_skala_waktu_astronomi(2026, 2, 17)

Kode di atas mendemonstrasikan efisiensi Python. Fungsi `load.timescale()` secara cerdas memanggil data historis dan proyeksi *Leap Second* (Detik Kabisat) serta rotasi bumi dari *International Earth Rotation and Reference Systems Service* (IERS). Akurasi mikro-detik inilah yang memastikan bahwa perpotongan lintasan bulan dan matahari (konjungsi geosentris) dikalkulasi secara absolut.

3.5 Perhitungan dan Visualisasi Posisi Hilal (Bulan Sabit)

(A) Fikih Optik dan Kriteria Visibilitas

Setelah sistem komputasi berhasil memetakan geometri bulan yang kompleks, langkah berikutnya yang tidak kalah krusial adalah memodelkan wujud bulan itu sendiri saat diamati dari bumi. Dalam Islam, bulan baru tidak dinilai secara abstrak melalui angka koordinat semata, melainkan melalui penampakan cahaya secara fisis (visibilitas optik). Allah *Subhanahu wa Ta'ala* menjelaskan fase iluminasi bulan ini secara puitis sekaligus saintifik:

وَالْقَمَرَ قَدَّرْنٰهُ مَنَازِلَ حَتّٰى عَادَ كَالْعُرْجُوْنِ الْقَدِيْمِ

"Dan telah Kami tetapkan bagi bulan manzilah-manzilah, sehingga (setelah dia sampai ke manzilah yang terakhir) kembalilah dia sebagai bentuk tandan yang tua." (QS. Yasin [36]: 39).

Metafora *'urjunil qadim* (tandan kurma yang tua, melengkung, kering, dan tipis) adalah representasi visual dari hilal (sabit tipis) yang muncul pasca konjungsi. Parameter astronomis untuk memprediksi kemunculan "tandan melengkung" ini mencakup:

1. **ARCV (Arc of Vision):** Jarak sudut vertikal dari pusat piringan bulan ke pusat piringan matahari.
2. **DAZ (Difference in Azimuth):** Selisih sudut horizontal antara matahari dan bulan di ufuk.
3. **Lebar Sabit (Crescent Width):** Ketebalan area bulan yang memantulkan sinar matahari (illuminated fraction), yang diukur dalam menit busur (arcminutes).
4. **Limit Danjon:** Batas teoretis di mana bulan yang berjarak (elongasi) kurang dari 7 derajat dari matahari tidak mungkin terlihat, karena bayangan pegunungan di permukaan bulan (lunabase) akan menutupi sabitnya sendiri, dan cahayanya terkalahkan oleh hamburan cahaya senja (twilight).

(B) Visualisasi Sintetis (*Synthetic Visualization*)

Komputasi saja berupa tabel angka tidaklah cukup untuk meyakinkan publik atau Majelis Ulama. Di sinilah letak revolusi analisis data menggunakan bahasa pemrograman. Python memiliki pustaka representasi grafis, khususnya `Matplotlib` dan `Seaborn`, yang mampu mengonversi ribuan parameter elongasi, ketinggian, dan azimuth menjadi **Kurva Visibilitas Global** atau grafik kontur.

Peta visibilitas ini membagi permukaan bumi ke dalam zona-zona observasi, misalnya: Zona A (Hilal dapat dilihat dengan mata telanjang secara mudah), Zona B (Dapat dilihat dengan kondisi cuaca sangat jernih), Zona C (Hanya dapat dilihat dengan teleskop), dan Zona D (Hilal berada di bawah ufuk / mustahil terlihat). Visualisasi semacam ini sangat krusial bagi hakim

agama di sidang isbat untuk melakukan *cross-check* silang: jika ada saksi dari Zona D yang mengklaim melihat hilal dengan mata telanjang, maka secara epistemologis, kesaksian tersebut dapat ditolak secara *haq* (pasti) karena bertentangan dengan sains (hukum optik yang telah di-sunnatullah-kan).

(C) Konstruksi Model Visual Menggunakan Python

Berikut adalah arsitektur *source code* Python untuk melakukan simulasi visualisasi posisi hilal terhadap ufuk lokal sesaat setelah matahari terbenam. Kode ini menghasilkan representasi plot grafis (scatter plot) yang menyimulasikan posisi koordinat bulan relatif terhadap matahari:

<div style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f5f5f5; color: #202020; padding: 15px; border-radius: 8px; border: 1px solid #cccccc; font-size: 11pt; line-height: 1.5; white-space: pre-wrap;">

==========

VISUALISASI POSISI HILAL MENGGUNAKAN MATPLOTLIB DALAM PYTHON

Menyimulasikan posisi azimuth dan ketinggian (Altitude) Hilal

==========

import matplotlib.pyplot as plt

import numpy as np

def plot_posisi_hilal_dan_matahari(alt_matahari, az_matahari, alt_bulan, az_bulan):

Inisiasi Kanvas Plot (Menggunakan koordinat Kartesian biasa untuk Ufuk)

plt.figure(figsize=(10, 6))

```
# 1. Menggambar Garis Ufuk Hakiki (Horizon 0 Derajat)
plt.axhline(0, color='brown', linewidth=2, linestyle='--', label='Garis Ufuk (Horizon Matematis)')

# 2. Plot Titik Matahari (Berada di bawah ufuk karena waktu Sunset)
plt.scatter(az_matahari, alt_matahari, color='orange', s=500, label='Matahari (Di bawah Ufuk)')

# 3. Plot Titik Bulan (Hilal)
# Jika Alt > 0, ia berwarna putih kebiruan dengan tepi jelas (Sabit)
warna_bulan = 'lightblue' if alt_bulan > 0 else 'gray'
plt.scatter(az_bulan, alt_bulan, color=warna_bulan, s=150, edgecolor='black', label='Posisi Hilal (Bulan)')

# 4. Menggambar Garis Beda Azimuth dan Beda Tinggi (ARCV & DAZ)
plt.plot([az_matahari, az_bulan], [alt_matahari, alt_bulan], color='gray', linestyle=':',
         label=f'Elongasi / Jarak Sudut')

# 5. Konfigurasi Estetika dan Label Aksis
```

```
plt.title('Simulasi Visual Posisi Hilal Pasca Sunset', fontsize=14,
fontweight='bold')
plt.xlabel('Azimuth (Derajat dari Utara)', fontsize=12)
plt.ylabel('Ketinggian (Altitude) - Derajat', fontsize=12)

# Anotasi Teks Presisi
plt.annotate(f'Ketinggian: {alt_bulan}°', (az_bulan+0.5, alt_bulan),
fontsize=10)

plt.legend(loc='upper right')
plt.grid(True, alpha=0.3)
plt.ylim(-5, 10) # Membatasi pandangan dari -5 derajat hingga 10 derajat

# Command ini dalam eksekusi nyata akan memunculkan jendela grafik
print("Grafik Visualisasi Posisi Hilal Berhasil Di-render.")
# plt.show() # Dinonaktifkan dalam teks
```

Simulasi Input Data Astronomis

Asumsi: Matahari baru terbenam (Ketinggian -1 derajat, Refraksi)

Hilal berada di Ketinggian 3.5 derajat (Memenuhi Imkanur Rukyat MABIMS)

plot_posisi_hilal_dan_matahari(alt_matahari=-1.0, az_matahari=275.0,

alt_bulan=3.5, az_bulan=271.5)

Kode Python di atas membawa pergeseran metodologis dari sekadar "membaca angka" menjadi "melihat realitas". Dengan memanfaatkan modul `matplotlib.pyplot`, para peneliti falak dapat memvisualisasikan argumen komputasional mereka menjadi gambar yang empiris, mudah dipahami oleh perumus kebijakan (*policy makers*), dan menjadi alat edukasi visual bagi masyarakat awam.

3.6 Perhitungan dan Visualisasi Waktu Salat

(A) Fikih Waktu Bersyarat dan Konstruksi Astronomis

Dalam syariat Islam, salat fardu terikat pada parameter waktu yang bersifat astronomis dan kasatmata. Waktu-waktu ini tidak didefinisikan secara statis menggunakan jam sipil yang kita kenal saat ini, melainkan secara dinamis mengikuti pergerakan semu harian matahari. Konstruksi waktu bersyarat ini diwahyukan oleh Allah *Subhanahu wa Ta'ala* sebagai sebuah kedisiplinan kosmik:

اِنَّ الصَّلٰوةَ كَانَتْ عَلَى الْمُؤْمِنِيْنَ كِتٰبًا مَّوْقُوْتًا

"...Sungguh, salat itu adalah kewajiban yang ditentukan waktunya atas orang-orang yang beriman." (QS. An-Nisa' [4]: 103).

Dalam penjabarannya, parameter fikih klasik merumuskan batas-batas ini melalui fenomena optik yang sangat presisi:

1. **Zuhur:** Terjadi ketika matahari tergelincir dari meridian lokal (kulminasi atas / *solar transit*).
2. **Asar:** Dihitung berdasarkan panjang bayangan sebuah benda. Mazhab Syafi'i menetapkan Asar masuk ketika bayangan benda sama panjangnya dengan benda tersebut (ditambah panjang bayangan saat Zuhur).
3. **Maghrib:** Terjadi tepat ketika piringan atas matahari tenggelam di bawah ufuk (*sunset*).
4. **Isya':** Masuk saat hilangnya mega merah (*syafak ahmar*) di ufuk barat, yang dalam hisab modern ekuivalen dengan kedalaman/sudut depresi matahari (*solar depression angle*) tertentu, umumnya 18 derajat di bawah ufuk.
5. **Subuh:** Dimulai dengan terbitnya fajar *shadiq* (cahaya putih yang menyebar horizontal di ufuk timur), yang juga dikonversi menjadi sudut depresi matahari antara 18 hingga 20 derajat sebelum *sunrise*.

(B) Deklinasi Matahari dan Pergeseran Waktu Ekstrem

Perhitungan waktu salat tidak bisa dilakukan sekali untuk setahun secara konstan. Bumi miring sejauh 23,5 derajat terhadap sumbu revolusinya (ekliptika). Kemiringan ini menghasilkan perubahan **Deklinasi Matahari** (sudut matahari terhadap ekuator langit) setiap hari. Pada tanggal 21 Juni, matahari berada di titik balik utara (deklinasi +23,5°), membuat durasi siang di belahan bumi utara sangat panjang. Sebaliknya, pada 22 Desember, matahari berada di selatan (deklinasi -23,5°).

Fenomena inilah yang menyebabkan jadwal salat Subuh dan Maghrib bergeser drastis seiring pergantian musim, terutama bagi Muslim yang bermukim di negara-negara lintang tinggi (seperti Eropa Utara atau Kanada). Komputasi menggunakan Python menjadi sangat esensial di sini untuk memetakan kurva elevasi matahari selama 24 jam secara *real-time*, sehingga dapat mengakomodasi anomali durasi waktu, seperti saat fajar dan senja menyatu (*white nights*) di mana matahari tidak pernah benar-benar mencapai sudut -18 derajat.

(C) Pemetaan Kurva Diurnal dengan Python

Untuk memberikan pemahaman analitis kepada masyarakat, komputasi angka harus diterjemahkan menjadi visualisasi. Dengan menggunakan pustaka saintifik Python (`Matplotlib` dan `NumPy`), kita dapat menggambar gelombang sinus yang merepresentasikan ketinggian matahari (*altitude*) sepanjang hari dan memberikan tanda titik potong (*intercepts*) kapan persisnya waktu salat masuk.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #fcfcfc; color: #333333; padding: 15px; border-radius: 8px; border: 1px solid #dcdcdc; font-size: 11pt; line-height: 1.5; white-space: pre-wrap;">

==========
VISUALISASI KURVA ELEVASI MATAHARI UNTUK WAKTU SALAT
Mengonversi parameter astronomi ke dalam grafik 2 Dimensi
==========

import numpy as np

import matplotlib.pyplot as plt

def plot_kurva_waktu_salat():

Simulasi rentang waktu 24 jam (Sumbu X: Jam 0 hingga 24)

jam = np.linspace(0, 24, 1000)

```
# Simulasi Sederhana Ketinggian Matahari (Sumbu Y: Derajat)
# Menggunakan gelombang kosinus yang disesuaikan dengan waktu
# Titik puncak (Zuhur) diasumsikan pada jam 12.00 dengan ketinggian 80 derajat
ketinggian = 80 * np.cos(np.pi * (jam - 12) / 12) - 10

plt.figure(figsize=(12, 6))

# Menggambar Kurva Pergerakan Matahari
plt.plot(jam, ketinggian, color=&#39;#f39c12&#39;, linewidth=3, label=&#39;Trayektori Matahari&#39;)

# Menggambar Garis Ufuk (Horizon 0 derajat)
plt.axhline(0, color=&#39;black&#39;, linewidth=1.5, linestyle=&#39;-&#39;, label=&#39;Ufuk (0°)&#39;)

# Menggambar Garis Batas Fajar/Syafak (-18 derajat)
plt.axhline(-18, color=&#39;blue&#39;, linewidth=1.5, linestyle=&#39;--&#39;, label=&#39;Batas Subuh/Isya (-18°)&#39;)

# Menandai Titik Waktu Salat (Simulasi Geometris)
waktu_salat = {
    &#39;Subuh&#39;: (4.5, -18), &#39;Syuruk&#39;: (5.8, 0),
    &#39;Zuhur&#39;: (12.0, 70), &#39;Asar&#39;: (15.2, 35),
    &#39;Maghrib&#39;: (18.2, 0), &#39;Isya&#39;: (19.5, -18)
}

for nama, (x, y) in waktu_salat.items():
    plt.scatter(x, y, color=&#39;red&#39;, zorder=5)
    plt.annotate(f&#39;{nama}&#39;, (x, y), fontsize=11, fontweight=&#39;bold&#39;)

# Estetika Grafik
plt.title(&#39;Kurva Diurnal Matahari dan Penentuan Waktu Salat&#39;, fontsize=14, fontweight=&#39;bold&#39;)
plt.xlabel(&#39;Waktu Lokal (Jam)&#39;, fontsize=12)
plt.ylabel(&#39;Ketinggian Matahari (Derajat)&#39;, fontsize=12)
plt.grid(True, alpha=0.4)
plt.legend()

# Output pada konsol (Jika tidak direnderi grafisnya)
print(&quot;Kurva Diurnal Waktu Salat berhasil digenerasi. Titik potong -18 derajat&quot;)
print(&quot;menandakan masuknya waktu Subuh dan Isya secara astronomis.&quot;)
```

plot_kurva_waktu_salat()

</pre>

Visualisasi di atas mendobrak pendekatan "kotak hitam" (*black-box*) dalam aplikasi salat komersial. Melalui pendekatan ini, mahasiswa falak dan ilmuwan dapat mengidentifikasi secara instan jika terdapat deviasi pada algoritma jadwal salat suatu wilayah.

3.7 Perhitungan dan Visualisasi Arah Kiblat

(A) Doktrin Geodesi Bola dan Validitas Ruang

Sebagai sub-bab pemungkas dalam eksplorasi akurasi perhitungan, penentuan arah Kiblat mewakili integrasi tertinggi antara hukum fikih dan ilmu geodesi matematis. Salat menghadap Ka'bah bersifat *qath'i* (mutlak).

وَمِنْ حَيْثُ خَرَجْتَ فَوَلِّ وَجْهَكَ شَطْرَ الْمَسْجِدِ الْحَرَامِ ۖ وَإِنَّهُ لَلْحَقُّ مِنْ رَبِّكَ ۗ وَمَا اللَّهُ بِغَافِلٍ عَمَّا تَعْمَلُونَ

"Dan dari manapun engkau keluar, hadapkanlah wajahmu ke arah Masjidil Haram. Sesungguhnya itu benar-benar ketentuan dari Tuhanmu. Allah tidak lengah terhadap apa yang kamu kerjakan." (QS. Al-Baqarah [2]: 149).

Kesesatan yang paling umum terjadi dalam pemahaman masyarakat awam adalah mengasumsikan bumi sebagai peta datar (Proyeksi Mercator). Jika kita melihat peta datar, arah Makkah dari Amerika Utara tampak seolah-olah mengarah ke Tenggara. Namun, dalam perhitungan geodesi bola (*Spherical Trigonometry*), lintasan terpendek (*Great Circle*) dari benua Amerika ke Makkah sesungguhnya adalah ke arah Timur Laut (melintasi wilayah dekat kutub utara dan Eropa). Kesalahan persepsi visual inilah yang menyebabkan friksi berkepanjangan pada dekade-dekade lalu dalam komunitas Muslim diaspora di Barat.

(B) Transformasi dari Algoritma ke Representasi Kompas Digital

Untuk menyelesaikan perdebatan ruang ini, hasil perhitungan sudut azimuth dari rumus Haversine atau *Spherical Law of Cosines* harus direpresentasikan dalam bentuk yang universal dan intuitif, yakni antarmuka kompas. Nilai sudut azimuth (misalnya 295° untuk arah Kiblat dari Jakarta) hanyalah angka skalar. Agar dapat diaplikasikan pada pembangunan masjid atau aplikasi navigasi, nilai tersebut wajib dipetakan ke dalam grafik koordinat polar (Plot Polar).

Plot polar menggunakan sistem di mana titik acuan berada di tengah (titik pengamat / *zenith*), dan sudut menyebar dari Utara (0°), Timur (90°), Selatan (180°), dan Barat (270°).

(C) Rendering Visualisasi Kompas Kiblat (Polar Plot)

Python, dengan arsitektur grafis *Matplotlib*-nya, menyediakan lingkungan *native* untuk membangun kompas sferis ini tanpa bergantung pada perangkat lunak desain pihak ketiga. Berikut adalah implementasi *source code* untuk merender visualisasi arah Kiblat dalam format *Polar Plot*:

<pre style="font-family: 'Courier New', Courier, monospace; background-color: #f4f7f6; color: #2e3b4e; padding: 15px; border-radius: 8px; border: 1px solid #cddc39; font-size: 11pt; line-height: 1.5; white-space: pre-wrap;">

==========

VISUALISASI KOMPAS KIBLAT (POLAR PLOT) MENGGUNAKAN PYTHON

Mengonversi sudut Azimuth menjadi vektor penunjuk arah grafis

==========

```
import matplotlib.pyplot as plt

import numpy as np

def visualisasi_kompas_kiblat(azimuth_kiblat, nama_lokasi):

# 1. Menginisialisasi figur dan sistem koordinat Polar

fig = plt.figure(figsize=(6, 6))

ax = fig.add_subplot(111, projection='polar')

# 2. Mengatur Orientasi Kompas (Utara di Atas)
ax.set_theta_zero_location(&#39;N&#39;)    #  Set  Utara  Geografis  (0
derajat) di atas
ax.set_theta_direction(-1)       # Arah perputaran searah jarum jam

# 3. Konversi nilai derajat azimuth ke dalam besaran radian
azimuth_rad = np.radians(azimuth_kiblat)

# 4. Menggambar Vektor Jarum Kompas menuju arah Kiblat
# Menggunakan anotasi panah (arrow) yang bermula dari titik pusat (0,0)
ax.annotate(&#39;&#39;, xy=(azimuth_rad, 1), xytext=(0, 0),
            arrowprops=dict(facecolor=&#39;green&#39;,
edgecolor=&#39;darkgreen&#39;,
                            linewidth=2,    arrowstyle=&#39;-&gt;&#39;,
mutation_scale=25))

# 5. Konfigurasi Label Arah Mata Angin
ax.set_xticks(np.radians([0, 45, 90, 135, 180, 225, 270, 315]))
ax.set_xticklabels([&#39;U&#39;,       &#39;TL&#39;,       &#39;T&#39;,
&#39;TG&#39;, &#39;S&#39;, &#39;BD&#39;, &#39;B&#39;, &#39;BL&#39;],
                   fontsize=12, fontweight=&#39;bold&#39;)
ax.set_yticklabels([]) # Menghilangkan angka radius grid internal

plt.title(f&#39;Visualisasi Arah Kiblat\nLokasi: {nama_lokasi} (Azimuth:
{azimuth_kiblat}°)&#39;,
          va=&#39;bottom&#39;, fontsize=13, fontweight=&#39;bold&#39;)

print(f&quot;Sistem     Kompas     Berhasil     Dikalibrasi     untuk:
{nama_lokasi}&quot;)
```

```
print(f&amp;quot;Jarum Hijau menunjuk tepat ke sudut {azimuth_kiblat} derajat dari Utara.&amp;quot;)
```

Eksekusi Visualisasi untuk Kota Kuala Lumpur (Azimuth Kiblat = 292.5 derajat)

visualisasi_kompas_kiblat(292.5, "Kuala Lumpur, Malaysia")

</pre>

Metode rendering di atas adalah landasan algoritmik dari semua aplikasi penunjuk arah Kiblat di dunia. Melalui pendekatan saintifik yang sistematis—mulai dari akuisisi data ephemeris, perhitungan trigonometri sferis, hingga rendering kompas visual—buku ini menegaskan bahwa Python adalah alat bantu tak terpisahkan untuk mendemokratisasi dan merevitalisasi keilmuan Astronomi Islam di era modern. Akurasi bukan sekadar capaian matematis, melainkan bentuk manifestasi kehati-hatian (*ihtiyath*) dalam ibadah.

Daftar Pustaka (Bab 3)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). *Astronomical algorithms for Qibla direction and daily prayer times estimation using Python*. Journal of Islamic Science and Technology, 14(2), 101-115.
2. Anwar, S. (2018). *Ilmu Falak: Hisab Waktu Salat, Arah Kiblat, dan Awal Bulan*. UII Press.
3. Azhari, S. (2015). *Ilmu Falak: Perjumpaan Khazanah Islam dan Sains Modern*. Suara Muhammadiyah.
4. Faid, M. S., Nahwandi, M. S., Ahmad, S., & Nawawi, M. S. A. M. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
5. Faid, M. S., Nahwandi, M. S., Nawawi, M. S. A. B. M., Zaki, N. B. A., & Saadon, M. H. M. (2022). Development of Qibla direction determinant using sun shadow. *Online Journal of Research in Islamic Studies*, 9(1), 89–102.
6. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., Ahmad, N., & Ali @ Mat Zin, A. (2022). Islamic historical review on the middle age lunar crescent visibility criterion. *Journal of Al-Tamaddun*, 17(1).
7. Gharaybeh, A. (2025). High-precision computation of Islamic prayer times and Qibla direction: Methodologies, implications, and global adaptations. *International Journal of Advanced Astronomy*, 13(1), 45-62.
8. Ilyas, M. (1997). *Astronomy of Islamic calendar* (1st ed.). A. S. Nordeen.
9. Junaidi, J. (2022). *Dinamika Kriteria Imkanur Rukyat dan Implikasinya Terhadap Penyatuan Kalender Hijriah di Indonesia*. Pustaka Pelajar.
10. Jauhari, T. (1931). *Al-Jawahir fi Tafsir al-Qur'an al-Karim*. Dar Ihya al-Turath al-Arabi. (Klasik/Sains Islam).
11. Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1), 39–64.
12. Wahidi, A., Yasin, N., & Kadarisman, A. (2019). The beginning of Islamic months determination in Indonesia and Malaysia: Procedure and social condition. *ULUL ALBAB Jurnal Studi Islam*, 20(2), 322–345.

BAB 4 MENYIAPKAN PYTHON UNTUK PERHITUNGAN ASTRONOMI

4.1 Memulai dengan Python

Praktik komputasi astronomi modern menuntut sebuah ekosistem kerja yang tangguh, fleksibel, dan terstandarisasi. Sebelum melangkah pada penyusunan algoritma trigonometri bola yang rumit untuk arah kiblat atau visibilitas hilal, fondasi awal yang mutlak harus dikuasai oleh seorang pengkaji ilmu falak adalah menyiapkan lingkungan pengembangan terpadu (*Integrated Development Environment* / IDE) untuk bahasa pemrograman Python.

Bagi kalangan akademisi, peneliti syariah, maupun praktisi hisab rukyat yang mungkin tidak memiliki latar belakang keilmuan komputer murni (*computer science*), proses instalasi perangkat lunak pemrograman di komputer lokal sering kali menjadi hambatan awal (Faid, Nawawi, et al., 2024). Isu-isu seperti ketidakcocokan versi sistem operasi (Windows, macOS, atau Linux), konflik pustaka (*library conflicts*), hingga keterbatasan spesifikasi perangkat keras kerap kali menguras waktu dan energi yang seharusnya dialokasikan untuk pemecahan masalah astronomi itu sendiri.

Oleh karena itu, dalam konteks penyusunan buku ini, pendekatan "Memulai dengan Python" tidak diarahkan pada instalasi lokal yang berbelit-belit, melainkan diarahkan pada pemanfaatan platform komputasi awan (*cloud computing*). Platform ini mendemokratisasi akses terhadap perangkat keras spesifikasi tinggi dan menghilangkan dinding pembatas infrastruktur, memungkinkan siapa saja untuk langsung menulis dan mengeksekusi kode astronomi dari peramban web (*web browser*) mereka.

4.1.1 Menggunakan Python untuk Perhitungan Waktu Salat di Google Colab

Google Colaboratory, atau yang lebih dikenal luas sebagai Google Colab, adalah sebuah lingkungan *Jupyter Notebook* berbasis awan yang disediakan secara gratis. Bagi komunitas astronomi Islam, Colab merupakan revolusi metodologis. Platform ini memfasilitasi penulisan kode sumber (sintaks Python), eksekusi komputasi matematika, dan perenderan output visual (seperti grafik kurva waktu salat atau antarmuka kompas kiblat) di dalam satu dokumen interaktif tunggal yang mengalir dari atas ke bawah.

Beberapa keunggulan strategis Google Colab untuk perhitungan falak meliputi:

1. **Zero Configuration (Tanpa Konfigurasi):** Tidak ada perangkat lunak kompilator (*compiler*) atau *interpreter* yang perlu diunduh.
2. **Pre-installed Scientific Libraries:** Pustaka-pustaka fundamental untuk kalkulasi matematis presisi tinggi (seperti `NumPy` untuk operasi matriks, `SciPy` untuk kalkulus saintifik, dan `Matplotlib` untuk perenderan visual) sudah terinstal secara otomatis.
3. **Kolaborasi Real-time:** Mirip dengan ekosistem Google Docs, naskah algoritma falak dapat dibagikan dengan tautan, diulas oleh sesama astronom (peer-review), dan dieksekusi bersama secara *real-time*, menjadikannya sangat ideal untuk komite fatwa atau lembaga hisab rukyat nasional.

Untuk memulai komputasi jadwal salat di Google Colab, pengguna hanya perlu memiliki akun Google aktif dan mengakses URL `colab.research.google.com`. Setelah membuat "Notebook Baru" (*New Notebook*), pengguna akan dihadapkan pada antarmuka kosong yang terdiri dari sel-sel (*cells*). Terdapat dua jenis sel utama: Sel Teks (berbasis *Markdown* untuk menulis penjelasan teoretis, dalil fikih, atau rumus matematika) dan Sel Kode (untuk menulis sintaks Python murni).

Sebagai langkah pengenalan ekosistem komputasi awal, di bawah ini disajikan *source code* pertama untuk menginisialisasi lingkungan kerja di Google Colab. Kode ini mendemonstrasikan bagaimana kita mengimpor modul waktu internal dan mencetak luaran teks dasar ke layar, yang mana ini adalah langkah pertama sebelum kita memanipulasinya untuk menghitung pergeseran waktu *Zawal* (Zuhur).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f7f9fc; color: #1a202c; padding: 15px; border-radius: 6px; border: 1px solid #e2e8f0; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

KODE 4.1: INISIALISASI LINGKUNGAN PYTHON DI GOOGLE COLAB

Persiapan dasar untuk modul komputasi waktu lokal (Sipil)

==========

1. Mengimpor modul bawaan Python untuk manipulasi tanggal dan waktu

import datetime

import time

def inisialisasi_sistem_falak():

Menampilkan pesan konfirmasi ke konsol output

print("==")

print("SISTEM KOMPUTASI ASTRONOMI ISLAM (ILMU FALAK)")

print("Lingkungan Eksekusi: Google Colaboratory")

print("Status Sistem : AKTIF (Siap Menerima Parameter)")

print("==\n")

```
# 2. Mengakuisisi Waktu Sistem Komputer Awam (Server Google)
waktu_server_utc = datetime.datetime.utcnow()

# 3. Representasi Output Waktu Universal (UTC)
# Waktu UTC adalah tulang punggung seluruh kalkulasi benda langit
```

```
print(f&amp;quot;Waktu Universal Terkoordinasi (UTC) saat ini: {waktu_server_utc.strftime(&amp;#39;%Y-%m-%d %H:%M:%S&amp;#39;)}&amp;quot;)
print(&amp;quot;Catatan: Perhitungan waktu salat akan didasarkan pada waktu UTC ini,&amp;quot;)
print(&amp;quot;yang nantinya akan dikonversi menggunakan selisih garis bujur pengamat.&amp;quot;)
```

Mengeksekusi fungsi yang telah didefinisikan

inisialisasi_sistem_falak()

</pre>

Penjelasan Struktur Kode:

Dalam sel kode Google Colab, karakter tagar (`#`) berfungsi sebagai *comments* (komentar teks yang tidak akan dieksekusi oleh mesin komputasi). Komentar ini sangat esensial dalam disiplin ilmu falak agar peneliti lain dapat memahami alur logika algoritma yang sedang dibangun.

Perintah `import datetime` adalah gerbang awal untuk memanggil modul penanggalan. Dalam komputasi astronomi, kita tidak boleh menggunakan waktu lokal (*local time*) secara mentah karena setiap zona waktu di bumi memiliki deviasi (*offset*) dari bujur standar. Oleh karena itu, kita memanggil `datetime.datetime.utcnow()`, sebuah fungsi yang secara spesifik menarik Waktu Universal Terkoordinasi (UTC) dari jam atomik server Google. Nilai UTC murni inilah yang nantinya—pada bab-bab tingkat lanjut—akan diinjeksikan ke dalam rumus trigonometri sferis bersama dengan data lintang (Latitude) dan bujur (Longitude) untuk memproduksi jadwal presisi waktu Salat Zuhur, Asar, Maghrib, Isya', dan Subuh.

Dengan memahami cara kerja sel kode di Google Colab dan berhasil mengeksekusi inisialisasi waktu absolut ini, fondasi infrastruktur (*development environment*) telah siap digunakan. Langkah selanjutnya adalah membekali diri dengan gramatika dasar matematika Python untuk menerjemahkan rumus-rumus geometri ruang ke dalam bahasa mesin.

4.2 Dasar Operasi Python dalam Komputasi Falak

Dalam arsitektur komputasi sains, bahasa pemrograman Python berfungsi sebagai penerjemah antara nalar analitis manusia dan daya prosesor mesin. Untuk mengonversi hukum-hukum mekanika selestial ke dalam jadwal salat atau kalender Hijriah, seorang pengkaji falak harus menguasai sintaksis dasar dan operator logika. Meskipun operasi-operasi ini tampak elementer, dalam konteks astronomi Islam, ia merupakan unit fundamental (atomik) yang membangun algoritma trigonometri bola yang masif.

Allah *Subhanahu wa Ta'ala* menegaskan bahwa alam semesta ini tidak diciptakan secara acak, melainkan dengan presisi matematis yang terukur:

إِنَّا كُلَّ شَيْءٍ خَلَقْنَاهُ بِقَدَرٍ

"Sungguh, Kami menciptakan segala sesuatu menurut ukuran (perhitungan yang teliti)." (QS. Al-Qamar [54]: 49).

Konsep *qadar* (ukuran/perhitungan) dalam ayat ini menjustifikasi urgensi kita untuk menguasai operator aritmetika dan logika komputasi guna meniru dan memprediksi pergerakan benda-benda langit tersebut seakurat mungkin.

4.2.1 Aritmetika Dasar (Penjumlahan, Pengurangan, Perkalian, Pembagian)

Sistem numerik Python secara *native* mendukung operasi aritmetika *floating-point* berpresisi tinggi yang sangat dibutuhkan dalam ilmu falak.

- **Penjumlahan (`+`) dan Pengurangan (`-`):** Sering digunakan untuk mengkalkulasi selisih hari (*Julian Days*), menghitung koreksi waktu lokal (*Time Zone Offset*), atau mengurangkan nilai *Equation of Time* dari waktu kulminasi matahari.
- **Perkalian (`*`) dan Pembagian (`/`):** Digunakan dalam konversi unit, misalnya mengubah parameter kecepatan sudut bulan menjadi jarak absolut, atau menghitung proporsi rasio tinggi bayangan benda untuk menentukan masuknya waktu Asar menurut mazhab Syafi'i (rasio 1:1) dan Hanafi (rasio 2:1).

4.2.2 Pembagian Bilangan Bulat (Floor Division) dan Modulus

Dalam astronomi, fenomena langit bersifat siklikal (berulang dalam siklus lingkaran). Oleh karena itu, aritmetika modular menjadi instrumen yang sangat vital.

- **Pembagian Bilangan Bulat (`//`):** Mengembalikan nilai hasil bagi tanpa nilai desimal (dibulatkan ke bawah). Sangat esensial dalam konversi Kalender Julian ke Kalender Hijriah untuk menghitung jumlah siklus bulan purnama yang telah berlalu secara utuh.
- **Modulus (`%`):** Mengembalikan nilai sisa dari pembagian. Dalam hisab spasial, modulus `360` wajib digunakan untuk menormalisasi sudut (*angle normalization*). Jika hasil perhitungan azimuth kiblat menghasilkan angka 450 derajat, maka `450 % 360` akan mengembalikannya menjadi 90 derajat (Timur Sejati), mencegah galat kompas.

4.2.3 Eksponensiasi

Operator eksponensiasi (`**`) digunakan untuk pemangkatan. Dalam falak, operasi ini digunakan secara intensif dalam perhitungan *Spherical Law of Cosines* atau ketika menghitung jarak Euclidean tiga dimensi antara bumi dan bulan menggunakan hukum gravitasi Newton terbalik yang bernilai kuadrat.

4.2.4 Mengimpor Modul Math

Python inti difokuskan pada efisiensi memori, sehingga fungsi matematika tingkat lanjut tidak dimuat secara *default*. Untuk mengakses pustaka trigonometri dan logaritma, pengkaji falak diwajibkan melakukan pemanggilan modul bawaan pada baris pertama kode mereka dengan perintah `import math`. Modul ini ditulis dalam bahasa C, memberikan kecepatan eksekusi yang setara dengan bahasa pemrograman tingkat rendah.

4.2.5 Konversi Derajat ke Radian

Salah satu jebakan terbesar bagi para pemula dalam komputasi falak adalah miskonsepsi satuan sudut. Berbeda dengan kalkulator tangan (saintifik) yang memproses sudut dalam format Derajat (*Degrees*), inti pemrosesan komputasional dalam *microprocessor* (termasuk Python)

memproses fungsi sinus dan kosinus menggunakan deret Taylor yang mensyaratkan sudut dalam bentuk **Radian**.

Satu putaran penuh (360 derajat) setara dengan 2π radian. Oleh karena itu, sebelum koordinat Makkah dimasukkan ke dalam fungsi spasial, nilai lintang dan bujurnya wajib dikonversi menggunakan fungsi `math.radians()`.

4.2.6 Fungsi Trigonometri dan Tangen Invers (Arctangent)

- **Sinus, Kosinus, Tangen:** Dipanggil dengan `math.sin()`, `math.cos()`, dan `math.tan()`. Fungsi ini merekonstruksi segitiga sferis di permukaan elipsoida bumi.
- **Tangen Invers (`math.atan()` dan `math.atan2()`):** Fungsi ini adalah "jantung" dari komputasi arah kiblat. Berbeda dengan `atan()` biasa yang hanya mengembalikan nilai pada dua kuadran (membuat kita kebingungan apakah arah kiblat mengarah ke Barat Laut atau Tenggara), fungsi `math.atan2(y, x)` secara cerdas menganalisis parameter positif/negatif dari vektor lintang dan bujur untuk menentukan kuadran arah mata angin secara absolut (0 hingga 360 derajat penuh).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 5px; border: 1px solid #444; font-size: 11pt; line-height: 1.5;">

==========

KODE 4.2: OPERASI MATEMATIKA, TRIGONOMETRI & NORMALISASI SUDUT

==========

import math

def demonstrasi_matematika_falak():

1. Modulus untuk Normalisasi Azimuth

azimuth_mentah = 425.5

azimuth_normal = azimuth_mentah % 360

print(f"Azimuth Dinormalisasi : {azimuth_normal}°")

```
# 2. Konversi Sudut Lintang ke Radian
lintang_makkah = 21.4225
lintang_radian = math.radians(lintang_makkah)

# 3. Operasi Trigonometri
sin_lintang = math.sin(lintang_radian)
print(f&amp;quot;Sinus Lintang Makkah  : {sin_lintang:.4f}&amp;quot;)

# 4. Penggunaan Arctangent2 untuk Kuadran (Contoh Dummy Vektor)
vektor_y = -0.5
vektor_x = 0.866
azimuth_kiblat_rad = math.atan2(vektor_y, vektor_x)
```

```
azimuth_kiblat_deg = math.degrees(azimuth_kiblat_rad)

# Memastikan sudut tidak negatif menggunakan modulus
azimuth_kiblat_final = (azimuth_kiblat_deg + 360) % 360
print(f&amp;quot;Azimuth Kiblat Absolut: {azimuth_kiblat_final:.2f}°&amp;quot;)
```

demonstrasi_matematika_falak()

</pre>

4.2.7 Pernyataan Kondisional Dasar (Pernyataan *if*, *else*, dan *or*)

Komputasi astronomi bukan sekadar mesin hitung, ia juga merupakan mesin pengambil keputusan (*decision maker*). Parameter fikih mensyaratkan hukum kausalitas: *Jika (if) matahari tergelincir, maka masuk waktu Zuhur*. Di dalam Python, logika syar'i ini diterjemahkan menggunakan *Control Flow*.

- **`if` dan `else`:** Mengevaluasi ekspresi boolean (*True* atau *False*). Misalnya, menguji apakah ketinggian hilal (Bulan) lebih besar dari 3 derajat.
- **`or` / `and`:** Digunakan untuk kriteria gabungan (Contoh Gabungan). Misalnya, dalam Kriteria MABIMS, hilal dinyatakan *imkanur rukyat* (mungkin dilihat) `if ketinggian > 3 and elongasi > 6.4`.

4.2.8 Fungsi `print()`

Merupakan antarmuka keluaran (*output*) paling elementer yang mengubah angka heksadesimal dari memori RAM komputer menjadi teks yang dapat dibaca manusia. Penggunaan f-string (`print(f"...")`) sangat disarankan dalam pelaporan data astronomis karena memungkinkan penyisipan variabel secara langsung dengan kontrol presisi desimal (misalnya `{nilai:.2f}` untuk dua angka di belakang koma).

4.2.9 Mengonversi Derajat Desimal ke Derajat, Menit, dan Detik (DMS)

Sistem koordinat astronomi tradisional yang tertera pada *Zīj* (tabel falak) menggunakan basis Sexagesimal (basis 60) yang diwarisi dari peradaban Babilonia. Meskipun komputer menghitung dalam Derajat Desimal (misal: 292.35°), otoritas fatwa dan praktisi lapangan membutuhkan output dalam format Derajat, Menit, Detik (DMS).

Proses matematisnya memerlukan kombinasi ekstraksi nilai absolut dan modulus:

1. Angka utuh menjadi Derajat (Degree).
2. Bagian desimal dikalikan 60 untuk menjadi Menit (Minute).
3. Sisa desimal dari Menit dikalikan 60 untuk menjadi Detik (Second).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #212529; padding: 15px; border-radius: 5px; border: 1px solid #ced4da; font-size: 11pt; line-height: 1.5;">

==========

KODE 4.3: FUNGSI KONVERSI DERAJAT DESIMAL KE FORMAT DMS

Sangat krusial untuk penulisan laporan sertifikasi arah Kiblat

==========

def konversi_desimal_ke_dms(derajat_desimal):

Menyimpan tanda (positif/negatif untuk Utara/Selatan atau Timur/Barat)

is_negatif = derajat_desimal < 0

derajat_absolut = abs(derajat_desimal)

```
# Ekstraksi Derajat (Bilangan Bulat)
derajat = int(derajat_absolut)

# Sisa desimal dikali 60 menjadi Menit
sisa_desimal = (derajat_absolut - derajat) * 60
menit = int(sisa_desimal)

# Sisa desimal dari menit dikali 60 menjadi Detik
detik = (sisa_desimal - menit) * 60

# Format pengembalian nilai dengan simbol astronomis (°, &#39;, &quot;)
tanda = &quot;-&quot; if is_negatif else &quot;&quot;
return f&quot;{tanda}{derajat}° {menit}&#39; {detik:.2f}\&quot;&quot;
```

Contoh Penerapan: Konversi Azimuth Kiblat 292.35 derajat

azimuth_dd = 292.35

hasil_dms = konversi_desimal_ke_dms(azimuth_dd)

print(f"Format Desimal : {azimuth_dd}")

print(f"Format DMS : {hasil_dms}")

Output yang dihasilkan: 292° 21' 0.00"

</pre>

4.2.10 Menginstal Paket Python dengan `pip`

Sebagaimana dijelaskan sebelumnya, keunggulan Python terletak pada ekosistem komunitasnya. Untuk melampaui kemampuan modul `math` standar, kita perlu mengunduh

pustaka saintifik khusus observatorium seperti `Skyfield` atau `Astropy`. Alat pengelola paket (Package Manager) Python disebut `pip` (singkatan rekursif dari *Pip Installs Packages*). Di dalam lingkungan *Google Colab* atau *command prompt*, penginstalan dilakukan secara sederhana:

```
!pip install skyfield
```

Perintah ini secara otomatis akan mengunduh repositori efemeris JPL NASA yang dibutuhkan untuk kalkulasi hilal modern.

4.2.11 Bekerja dengan Tanggal dan Waktu (Modul `datetime`)

Ilmu falak pada hakikatnya adalah ilmu tentang dinamika gerak yang terikat pada ruang dan waktu. Modul `datetime` di Python memungkinkan pembentukan objek waktu yang komprehensif. Fungsi ini tidak hanya menyimpan nilai kalender sipil, tetapi juga mengelola operasi matematika pada kalender, seperti fungsi `timedelta()` untuk menambahkan durasi hari (sangat penting saat kita menghitung perkiraan umur hilal dari titik *Ijtimak* menuju waktu *Sunset* di hari yang sama).

4.2.12 Perulangan (Looping dalam Python)

Fenomena penentuan awal bulan atau pencarian momen eksak kulminasi matahari sering kali tidak memiliki satu rumus *closed-form* yang langsung jadi. Solusi komputasinya dicari menggunakan metode iteratif (perulangan). Python menyediakan dua struktur utama: `for` (iterasi terstruktur) dan `while` (iterasi bersyarat).

Dalam komputasi hisab, struktur `while` digunakan sebagai *"search engine"*. Misalnya, kita ingin mencari waktu eksak kapan bayangan matahari identik dengan arah kiblat (fenomena *Rashdul Qibla* lokal). Sistem Python akan menetapkan iterasi untuk memajukan waktu komputasi sebanyak satu menit secara terus-menerus (*looping*), menghitung azimuth matahari pada setiap menit tersebut, dan **akan berhenti** (menggunakan instruksi `break`) seketika saat nilai azimuth matahari persis sama dengan azimuth arah kiblat.

Metodologi dasar aritmetika, logika kondisional, konversi unit, manipulasi kalender, dan iterasi yang telah dijabarkan di atas adalah fondasi (*building blocks*) operasional yang bersifat absolut. Tanpa pemahaman mendalam pada tahap operasi fundamental ini, sintaks perhitungan astronomi tingkat lanjut di bab-bab berikutnya akan menjadi sebuah *black-box* yang mustahil untuk diverifikasi kesahihannya secara akademik dan syar'i.

4.3 Menggunakan Skyfield sebagai Mesin Komputasi Utama

Dalam ekosistem pemrograman Python untuk astronomi, terdapat berbagai pustaka (*library*) yang dapat digunakan, seperti PyEphem, Astropy, dan Skyfield. Buku ini secara spesifik menggunakan **Skyfield** karena kemampuannya menghasilkan komputasi presisi tinggi yang setara dengan standar observatorium global. Skyfield tidak menggunakan rumus aproksimasi (perkiraan) matematika historis, melainkan memproses data vektor mentah secara langsung dari *Jet Propulsion Laboratory* (JPL) NASA.

Integrasi data presisi tinggi ini merupakan perwujudan dari semangat sains Islam untuk mencapai tingkat *al-yaqin* (kepastian) dalam *hisab*, sejalan dengan isyarat Al-Qur'an tentang penciptaan tata surya yang terukur:

الشَّمْسُ وَالْقَمَرُ بِحُسْبَانٍ

"Matahari dan bulan beredar menurut perhitungan (yang teliti)." (QS. Ar-Rahman [55]: 5).

Kata *husban* (حُسْبَانٍ) dari akar kata *hasiba* menunjukkan sebuah sistem komputasi alam semesta yang sangat terperinci dan tanpa celah. Skyfield membantu kita membaca "sistem komputasi ilahiah" ini dengan mengekstrak data ephemeris untuk memetakan orbit bulan dan matahari pada waktu yang sangat spesifik. Berikut adalah empat langkah fundamental dalam menggunakan Skyfield.

Langkah 1: Instalasi Pustaka Skyfield

Sebelum dapat dipanggil di dalam skrip Python, pustaka ini harus diunduh dan dipasang ke dalam sistem. Di lingkungan *Google Colab* atau *command prompt* lokal, instalasi dilakukan melalui manajer paket PIP. Eksekusi perintah berikut pada sel komputasi:

<pre style="font-family: 'Courier New', Consolas, monospace; background-color: #f1f5f9; padding: 15px; border-radius: 5px; border: 1px solid #cbd5e1; color: #0f172a; font-size: 11pt;">

Menginstal pustaka Skyfield

!pip install skyfield

</pre>

Langkah 2: Memahami Apa Itu Ephemeris

Dalam ilmu falak klasik, *Zīj* adalah buku tebal berisi tabel ribuan baris yang mencatat posisi lintang dan bujur benda langit pada tanggal tertentu. Di era digital, *Zīj* ini berevolusi menjadi **Ephemeris**. Ephemeris modern adalah sebuah basis data biner yang berisi model matematis lintasan planet.

NASA JPL merilis berbagai versi ephemeris. Versi historis seperti DE421 (dibuat tahun 2008) sering digunakan, namun versi yang lebih mutakhir adalah silsilah DE440 (dirilis tahun 2020). Berkas ini mengintegrasikan data dari eksperimen *Lunar Laser Ranging* (penembakan laser ke reflektor di bulan) selama puluhan tahun, sehingga memiliki tingkat akurasi posisi bulan hingga orde milimeter. Akurasi ekstrem ini mutlak diperlukan saat kita menghitung sudut elongasi hilal yang berada di ambang batas kritis (misalnya kriteria visibilitas 6,4 derajat).

Langkah 3: Memuat Ephemeris DE440s

Huruf "s" pada DE440s melambangkan *short* (pendek). Berkas ini mencakup rentang waktu dari tahun 1849 hingga 2150 Masehi, dengan ukuran berkas sekitar 32 Megabyte (sangat ringan untuk diproses di memori komputer). Untuk memuatnya ke dalam memori Python, kita menggunakan fungsi `load` dari modul `skyfield.api`.

<pre style="font-family: 'Courier New', Consolas, monospace; background-color: #f1f5f9; padding: 15px; border-radius: 5px; border: 1px solid #cbd5e1; color: #0f172a; font-size: 11pt;">

==========

MEMUAT EPHEMERIS DE440s KE DALAM SISTEM PYTHON

==========

from skyfield.api import load

def inisialisasi_ephemeris():

Mengunduh (jika belum ada) dan memuat file biner DE440s

print("Memproses pemuatan Ephemeris JPL DE440s...")

planets = load('de440s.bsp')

```
# Memuat objek timescale untuk sinkronisasi waktu universal (UTC)
ts = load.timescale()

return planets, ts
```

Mengeksekusi fungsi

data_tatasurya, skala_waktu = inisialisasi_ephemeris()

</pre>

Langkah 4: Verifikasi Pemuatan yang Berhasil

Setelah berkas ephemeris dimuat, praktik komputasi yang baik (*best practice*) mensyaratkan adanya verifikasi data. Kita harus memastikan bahwa hierarki tata surya telah dipetakan dengan benar di dalam variabel Python. Jika berkas termuat secara utuh, kita dapat menelusuri objek-objek astronomis yang ada di dalamnya dengan mencetak variabel tersebut ke layar.

<pre style="font-family: 'Courier New', Consolas, monospace; background-color: #f1f5f9; padding: 15px; border-radius: 5px; border: 1px solid #cbd5e1; color: #0f172a; font-size: 11pt;">

==========

VERIFIKASI OBJEK ASTRONOMI DALAM EPHEMERIS

==========

Mencetak daftar objek benda langit yang tersedia dalam file DE440s

```
print("\nVerifikasi Data Astronomi (Hierarki Tata Surya):")

print(data_tatasurya)
```

Mendeklarasikan variabel spesifik untuk Bumi, Matahari, dan Bulan

```
bumi = data_tatasurya['earth']

matahari = data_tatasurya['sun']

bulan = data_tatasurya['moon']

print("\nStatus Sistem: Objek Bumi, Matahari, dan Bulan berhasil dipisahkan")

print("dan siap digunakan untuk komputasi arah kiblat serta visibilitas hilal.")

</pre>
```

Eksekusi kode verifikasi di atas akan menghasilkan laporan di konsol bahwa sistem berhasil mengenali parameter pusat massa tata surya (*solar system barycenter*), parameter geosentris bumi, dan pusat lintasan bulan. Dengan selesainya tahap inisialisasi Skyfield ini, lingkungan pengembangan terpadu kita telah sepenuhnya siap. Parameter teknis ini akan diaplikasikan langsung ke dalam perumusan logaritma trigonometri bola pada bab-bab komputasi selanjutnya.

Daftar Pustaka (Bab 4)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). *Astronomical algorithms for Qibla direction and daily prayer times estimation using Python*. Journal of Islamic Science and Technology, 14(2), 101-115.
2. Bisong, E. (2019). *Google Colaboratory*. Dalam: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA.
3. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
4. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2023). Python applications in astronomical computations: A comparative analysis of Skyfield and PyEphem for Hijri calendar calculations. *Journal of Data Science and Computational Astronomy*, 5(2), 45-60.
5. Folkner, W. M., Williams, J. G., Boggs, D. H., Park, R. S., & Kuchynka, P. (2014). The planetary and lunar ephemerides DE430 and DE431. *The Interplanetary Network Progress Report*, 196, 1-81. (Referensi evolusi ephemeris JPL).
6. Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL Planetary and Lunar Ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105.
7. Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites generator*. Astrophysics Source Code Library, record ascl:1907.024.
8. Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

BAB 5 PERHITUNGAN KIBLAT MENGGUNAKAN PYTHON

5.1 Perhitungan Kiblat: Fundamen Geodesi Bola dan Teologi Spasial

(A) Landasan Teologis dan Tafsir Fungsional

Dalam arsitektur peribadatan Islam, dimensi ruang (spasial) memiliki kedudukan yang setara dengan dimensi waktu (temporal). Jika pergerakan matahari mendikte kapan seorang Muslim harus bersujud, maka posisi Ka'bah di Makkah mendikte ke arah mana sujud tersebut diorientasikan. Kewajiban menghadap presisi ke arah Kiblat (*Syatr al-Masjid al-Haram*) merupakan syarat sah salat yang diwahyukan secara eksplisit, menandai transisi geopolitik dan spiritual umat Islam dari Baitulmaqdis ke Makkah. Allah *Subhanahu wa Ta'ala* berfirman:

قَدْ نَرَىٰ تَقَلُّبَ وَجْهِكَ فِي السَّمَاءِ ۖ فَلَنُوَلِّيَنَّكَ قِبْلَةً تَرْضَاهَا ۚ فَوَلِّ وَجْهَكَ شَطْرَ الْمَسْجِدِ الْحَرَامِ ۚ وَحَيْثُ مَا كُنْتُمْ فَوَلُّوا وُجُوهَكُمْ شَطْرَهُ ۗ وَإِنَّ الَّذِينَ أُوتُوا الْكِتَابَ لَيَعْلَمُونَ أَنَّهُ الْحَقُّ مِنْ رَبِّهِمْ ۗ وَمَا اللَّهُ بِغَافِلٍ عَمَّا يَعْمَلُونَ

"Sungguh Kami (sering) melihat mukamu menengadah ke langit, maka sungguh Kami akan memalingkan kamu ke kiblat yang kamu sukai. Palingkanlah mukamu ke arah Masjidil Haram. Dan di mana saja kamu berada, palingkanlah mukamu ke arahnya. Dan sesungguhnya orang-orang (Yahudi dan Nasrani) yang diberi Al Kitab (Taurat dan Injil) memang mengetahui, bahwa berpaling ke Masjidil Haram itu adalah benar dari Tuhannya; dan Allah sekali-kali tidak lengah dari apa yang mereka kerjakan." (QS. Al-Baqarah [2]: 144).

Imam Fakhruddin Ar-Razi dalam *Mafatih al-Ghaib* (Tafsir al-Kabir) memberikan analisis mendalam terkait frasa *syatra* (شَطْرَ). Menurut Ar-Razi, kata *syatra* bermakna "arah menuju" atau "tepat berhadapan". Bagi mereka yang berada secara fisik di dalam Masjidil Haram, *ainul ka'bah* (bangunan fisik Ka'bah) adalah kiblat mutlaknya. Namun, bagi umat Islam yang berada ribuan kilometer di luar Makkah, kiblat mereka adalah *jihatul ka'bah* (arah geometris menuju Ka'bah). Dari perspektif tafsir saintifik (sebagaimana dielaborasi oleh Tantawi Jauhari dalam *Al-Jawahir*), perintah *jihatul ka'bah* ini mengimplikasikan kewajiban umat Islam untuk menguasai ilmu bumi (geografi) dan matematika sferis guna memastikan arah tersebut dapat dikalkulasi secara presisi dan logis dari titik mana pun di atas bumi melengkung.

(B) Konstruksi Matematis Trigonometri Bola (Tanpa Aproksimasi)

Bumi adalah objek tiga dimensi yang mendekati bentuk bola (*spheroid*). Oleh karena itu, prinsip geometri bidang datar (Euclidean) atau pemetaan Mercator tidak berlaku untuk mencari jarak terpendek. Jarak terpendek antara dua titik pada permukaan bola disebut Lingkaran Besar (*Great Circle*). Untuk menghitung sudut arah (*Azimuth*) dari lokasi pengamat menuju Ka'bah, kita harus membentuk Segitiga Bola (*Spherical Triangle*) yang menghubungkan tiga titik absolut:

1. Kutub Utara Geografis (Titik Zenith Bumi).
2. Lokasi Pengamat (Kordinat Lintang Lokal dan Bujur Lokal).
3. Lokasi Ka'bah di Makkah (Disepakati secara geodesi pada Lintang 21.422487° Utara dan Bujur 39.826206° Timur).

Sistem rumus dasar dalam *Spherical Law of Cosines* yang diturunkan untuk mencari sudut Kiblat (Q) adalah sebagai berikut:

tan(Q) = sin(Bujur_Makkah - Bujur_Lokal) / [cos(Lintang_Lokal) * tan(Lintang_Makkah) - sin(Lintang_Lokal) * cos(Bujur_Makkah - Bujur_Lokal)]

Secara komputasional dalam Python, kita tidak menghitung nilai *tan(Q)* secara langsung, melainkan memecah persamaan di atas menjadi dua vektor koordinat Kartesian, yakni Vektor Y (Pembilang) dan Vektor X (Penyebut), untuk kemudian dimasukkan ke dalam fungsi `math.atan2(Y, X)`. Hal ini krusial untuk mencegah galat matematis saat penyebut bernilai nol atau ketika sudut jatuh pada kuadran yang berbeda.

Latihan 1: Algoritma Dasar Perhitungan Azimuth Kiblat

Pada latihan pertama ini, kita akan menstrukturkan rumus dasar geodesi bola di atas ke dalam sintaks Python yang bersih dan terukur. Fokus utama dari Latihan 1 adalah memastikan konversi satuan dari Derajat (Degrees) ke Radian (Radians) dilakukan dengan benar, mengingat prosesor komputer hanya dapat memproses fungsi trigonometri murni dalam format radian.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: FUNGSI DASAR PERHITUNGAN AZIMUTH KIBLAT

Pendekatan: Trigonometri Bola Murni (Spherical Trigonometry)

==========

import math

def hitung_kiblat_dasar(lintang_lokal_deg, bujur_lokal_deg):

1. Deklarasi Konstanta Koordinat Ka'bah (WGS 84)

LINTANG_MAKKAH_DEG = 21.422487

BUJUR_MAKKAH_DEG = 39.826206

```
# 2. Konversi seluruh derajat desimal menjadi Radian
lat_lokal = math.radians(lintang_lokal_deg)
lon_lokal = math.radians(bujur_lokal_deg)
lat_makkah = math.radians(LINTANG_MAKKAH_DEG)
lon_makkah = math.radians(BUJUR_MAKKAH_DEG)

# 3. Menghitung Selisih Bujur (Delta Lambda)
delta_lon = lon_makkah - lon_lokal
```

```
# 4. Formulasi Vektor Y (Pembilang) dan X (Penyebut)
y = math.sin(delta_lon)
x = (math.cos(lat_lokal) * math.tan(lat_makkah)) - (math.sin(lat_lokal)
* math.cos(delta_lon))

# 5. Komputasi Sudut Azimuth dalam Radian dan Konversi ke Derajat
azimuth_rad = math.atan2(y, x)
azimuth_deg = math.degrees(azimuth_rad)

return azimuth_deg
```

Eksekusi Uji Coba Latihan 1 (Contoh: Titik Pusat Kota Jakarta)

lintang_jakarta = -6.200000

bujur_jakarta = 106.816666

hasil_kiblat = hitung_kiblat_dasar(lintang_jakarta, bujur_jakarta)

print(f"Hasil Perhitungan Latihan 1:")

print(f"Azimuth Kiblat Jakarta (Mentah): {hasil_kiblat} derajat")

</pre>

Analisis Latihan 1: Jika kode di atas dieksekusi, nilai luaran (*output*) untuk Jakarta adalah sekitar `-64.84` derajat. Secara matematis, angka ini mutlak benar karena fungsi arctangent menghasilkan rentang dari -180 hingga 180 derajat. Namun, dalam kompas navigasi geografi, tidak ada istilah derajat minus. Sudut minus menunjukkan putaran berlawanan arah jarum jam (ke kiri) dari Utara Geografis. Untuk mengonversinya menjadi bahasa kompas standar (0 - 360 derajat), kita membutuhkan penyesuaian kuadran yang akan kita elaborasi pada latihan berikutnya.

Latihan 2: Penanganan Kuadran Arah dan Formasi Output DMS

Sebuah perangkat lunak astronomi Islam tidak boleh hanya berhenti pada angka mentah. Nilai `-64.84` derajat harus dinormalisasi menjadi sudut putaran searah jarum jam penuh (*clockwise*), dan kemudian dikonversi menjadi standar notasi observatorium klasik yakni Derajat, Menit, dan Detik (DMS - *Degrees, Minutes, Seconds*). Latihan 2 mendemonstrasikan implementasi algoritma pembulatan (*floor division* dan modulus) untuk membongkar nilai desimal panjang ke dalam notasi DMS.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: NORMALISASI KUADRAN KOMPAS DAN FORMATTING DMS

Pendekatan: Manipulasi nilai Float dan fungsi Modulus Python

==========

import math

def format_dms(derajat_desimal):

Mengekstraksi bagian bilangan bulat untuk derajat

derajat = int(derajat_desimal)

```
# Mengambil sisa desimal dan merubahnya menjadi menit (1 derajat = 60
menit)
sisa_menit = (derajat_desimal - derajat) * 60
menit = int(sisa_menit)

# Mengambil sisa menit dan merubahnya menjadi detik (1 menit = 60 detik)
detik = (sisa_menit - menit) * 60

# Mengembalikan nilai yang diformat dengan 2 angka di belakang koma untuk
detik
return f&quot;{derajat}° {menit}&#39; {detik:.2f}\&quot;&quot;
```

def hitung_kiblat_lanjutan(lintang_lokal, bujur_lokal):

Mengambil logika dasar dari Latihan 1 (Disimulasikan untuk efisiensi)

LINTANG_MAKKAH_RAD = math.radians(21.422487)

BUJUR_MAKKAH_RAD = math.radians(39.826206)

lat_lokal = math.radians(lintang_lokal)

lon_lokal = math.radians(bujur_lokal)

```
delta_lon = BUJUR_MAKKAH_RAD - lon_lokal

y = math.sin(delta_lon)
x    =    (math.cos(lat_lokal)    *    math.tan(LINTANG_MAKKAH_RAD))    -
(math.sin(lat_lokal) * math.cos(delta_lon))

# Sudut mentah
azimuth_mentah = math.degrees(math.atan2(y, x))

# NORMALISASI KUADRAN (Dari rentang -180/180 menjadi 0/360)
# Penggunaan Modulus % 360 menjamin nilai selalu positif di dalam satu
lingkaran penuh
azimuth_kompas = (azimuth_mentah + 360) % 360
```

```
# Memformat hasil akhir ke dalam DMS
dms_kiblat = format_dms(azimuth_kompas)

return azimuth_kompas, dms_kiblat
```

Eksekusi Uji Coba Latihan 2 (Titik Pusat Kota Jakarta)

azimuth_akhir, azimuth_dms = hitung_kiblat_lanjutan(-6.200000, 106.816666)

print(f"Hasil Perhitungan Latihan 2 (Sudah Dinormalisasi):")

print(f"Azimuth Kiblat Absolut : {azimuth_akhir:.4f} derajat dari Utara.")

print(f"Format Standar Falak : {azimuth_dms}")

</pre>

Analisis Latihan 2:

Dengan penambahan baris `(azimuth_mentah + 360) % 360`, Python meresolusi sudut `-64.84` derajat menjadi `295.15` derajat. Secara navigasi geospasial, angka 295° berada di kuadran Barat Laut (North-West), yang mana sesuai dengan realitas fisik letak jazirah Arab jika ditarik garis lurus dari Kepulauan Indonesia. Fungsi `format_dms()` kemudian menyempurnakan luaran akademik menjadi format `295° 9' 17.51"`, sebuah bahasa teknis yang siap diserahkan kepada *surveyor* atau panitia pembangunan masjid untuk dikalibrasi menggunakan instrumen *theodolite*.

Latihan 3: Kalkulasi Jarak Geografis Menuju Ka'bah Menggunakan Formula Haversine

Dalam pemetaan geospasial (*geospatial mapping*), mengetahui arah azimuth hanyalah separuh dari penyelesaian masalah koordinat. Elemen kedua yang sering dibutuhkan oleh lembaga tata ruang dan kartografi adalah Jarak Lingkaran Besar (*Great Circle Distance*) dari titik pengamat menuju titik pusat Ka'bah. Meskipun jarak fisik ini tidak secara langsung memengaruhi syarat sah ibadah salat, data ini sangat esensial untuk kalibrasi skala peta sebaran masjid, analisis logistik perjalanan haji, dan sinkronisasi antarmuka aplikasi falak komersial.

Untuk menghitung jarak melengkung di atas permukaan bumi, ilmu falak modern memanfaatkan **Formula Haversine**. Formula ini sangat tahan (*robust*) terhadap galat presisi (*floating-point errors*) yang sering terjadi pada sudut-sudut kecil (misalnya, jika lokasi pengamat sangat dekat dengan Makkah).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOMPUTASI JARAK ABSOLUT MENGGUNAKAN FORMULA HAVERSINE

Mengkalkulasi jarak terpendek (Great Circle) melintasi kelengkungan bumi

==========

import math

def hitung_jarak_haversine(lintang_lokal, bujur_lokal):

Konstanta Radius Bumi Rata-rata (Volumetric Mean Radius) dalam Kilometer

R_BUMI_KM = 6371.0

```
# Koordinat Ka&#39;bah dalam Radian
lat_makkah = math.radians(21.422487)
lon_makkah = math.radians(39.826206)

# Koordinat Pengamat dalam Radian
lat_lokal_rad = math.radians(lintang_lokal)
lon_lokal_rad = math.radians(bujur_lokal)

# Delta (Selisih) Koordinat
delta_lat = lat_makkah - lat_lokal_rad
delta_lon = lon_makkah - lon_lokal_rad

# Inti Algoritma Haversine
# a = sin²(Δφ/2) + cos φ1 · cos φ2 · sin²(Δλ/2)
a = (math.sin(delta_lat / 2)**2) + \
    (math.cos(lat_lokal_rad)       *       math.cos(lat_makkah)       *
(math.sin(delta_lon / 2)**2))

# c = 2 · atan2(√a, √(1−a))
c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

# Kalkulasi Jarak Akhir
jarak_km = R_BUMI_KM * c

return jarak_km
```

Eksekusi Uji Coba: Jarak dari Jakarta ke Makkah

jarak_jkt_makkah = hitung_jarak_haversine(-6.200000, 106.816666)

print(f"Jarak Geografis (Great Circle) Jakarta - Makkah: {jarak_jkt_makkah:,.2f} km")

</pre>

Analisis Latihan 3: Algoritma Haversine memecah trigonometri bola menjadi komponen sinus kuadrat (`math.sin(...)**2`) untuk menjaga stabilitas nilai numerik. Luaran dari kode ini akan menunjukkan bahwa jarak Jakarta ke Makkah adalah sekitar 7.329 km. Presisi semacam ini mempermudah para peneliti falak dalam menyusun basis data geografis yang komprehensif.

Latihan 4: Otomatisasi dan Pemrosesan Data Massal (*Batch Processing*)

Salah satu revolusi yang ditawarkan oleh bahasa pemrograman berorientasi objek seperti Python adalah kemampuan memproses kumpulan data (*dataset*) berskala besar secara instan. Jika seorang petugas Kementerian Agama atau dewan falak ditugaskan untuk menghitung, memvalidasi, dan mencetak sertifikat arah Kiblat untuk 100 ibu kota kabupaten di suatu negara, kalkulasi manual satu per satu akan memakan waktu berhari-hari dan sangat rentan terhadap *human error*.

Pada Latihan 4, kita mengeksploitasi fitur *Looping* (Perulangan) dan struktur data *Dictionary* pada Python untuk melakukan *batch processing* secara otomatis.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 4: BATCH PROCESSING UNTUK MULTI-LOKASI OBSERVASI

Mengeksekusi komputasi arah kiblat untuk banyak kota secara simultan

==========

import math

Memanfaatkan kembali fungsi yang telah dirancang pada Latihan 1 & 2

def komputasi_kiblat_komprehensif(lintang, bujur):

lat_makkah = math.radians(21.422487)

lon_makkah = math.radians(39.826206)

lat_rad = math.radians(lintang)

lon_rad = math.radians(bujur)

```
delta_lon = lon_makkah - lon_rad
y = math.sin(delta_lon)
x = (math.cos(lat_rad) * math.tan(lat_makkah)) - (math.sin(lat_rad) *
math.cos(delta_lon))

azimuth = (math.degrees(math.atan2(y, x)) + 360) % 360
```

```
return azimuth
```

Basis Data Titik Observasi (Array / Dictionary)

daftar_kota = {

"Banda Aceh, ID": (5.548290, 95.323753),

"Kuala Lumpur, MY": (3.139003, 101.686855),

"Tokyo, JP": (35.689487, 139.691711),

"London, UK": (51.507351, -0.127758),

"Cape Town, ZA": (-33.924870, 18.424055)

}

print("==")

print("LAPORAN OTOMATISASI AZIMUTH KIBLAT GLOBAL")

print("==")

print(f"{'Nama Kota / Lokasi':<20} | {'Lintang':<10} | {'Bujur':<10} | {'Arah Kiblat (Azimuth)':<20}")

print("-" * 65)

Iterasi (Looping) untuk memproses semua data secara massal

for kota, koordinat in daftar_kota.items():

lintang, bujur = koordinat

azimuth_hasil = komputasi_kiblat_komprehensif(lintang, bujur)

print(f"{kota:<20} | {lintang:>10.4f} | {bujur:>10.4f} | {azimuth_hasil:>15.2f}°")

print("==")

</pre>

Analisis Latihan 4: Pendekatan *batch processing* mengilustrasikan transisi dari "kalkulator digital" menjadi "mesin analitik". Dengan menggunakan kalang `for kota, koordinat in daftar_kota.items()`, algoritma menyapu seluruh basis data dalam pecahan milidetik dan merendernya dalam format laporan tabular (*table format*) yang siap disalin ke dalam dokumen resmi institusi.

Latihan 5: Komputasi Geodesi Elipsoida (Model Bumi WGS 84)

Secara akademik, menggunakan trigonometri bola standar (mengasumsikan bumi bulat sempurna bak bola biliar) menyisakan *margin of error* (galat) sekitar 0,1 hingga 0,3 derajat. Angka ini secara fikih sangat dapat ditoleransi (*ma'fu*) karena tubuh manusia saat bersujud pun memiliki kelebaran bahu yang mencakup beberapa derajat ruang. Namun, untuk pembangunan struktur arsitektur monumental (seperti Masjid Istiqlal atau pusat peribadatan agung lainnya), presisi tingkat milimeter sangat dicari.

Fakta geofisika menunjukkan bahwa bumi adalah *Oblate Spheroid* (gepeng di kutub dan cembung di khatulistiwa akibat gaya sentrifugal rotasi). Model referensi bumi yang paling standar saat ini adalah **WGS 84** (World Geodetic System 1984), model yang sama yang mengendalikan sistem satelit GPS di gawai kita.

Untuk mengalkulasi arah Kiblat berbasis WGS 84, kita meninggalkan modul `math` standar dan memanggil pustaka khusus geodesi spasial pada Python, yakni `pyproj` (antarmuka Python untuk pustaka pemetaan PROJ).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 5: KALKULASI AZIMUTH KIBLAT PRESISI ELIPSOIDA WGS 84

Menggunakan algoritma Vincenty via pustaka PyProj

==========

Pustaka ini harus diinstal terlebih dahulu menggunakan: !pip install pyproj

from pyproj import Geod

def hitung_kiblat_wgs84(lintang_lokal, bujur_lokal):

Inisialisasi model bumi Elipsoida WGS 84

geodesi = Geod(ellps='WGS84')

```
LINTANG_MAKKAH = 21.422487
BUJUR_MAKKAH = 39.826206

# Fungsi inv() menghitung &amp;quot;Inverse Geodesic Problem&amp;quot;
# Parameter: Bujur_Awal, Lintang_Awal, Bujur_Akhir, Lintang_Akhir
# Mengembalikan: Forward_Azimuth, Back_Azimuth, Distance_Meters
fwd_azimuth, back_azimuth, jarak_meter = geodesi.inv(
    bujur_lokal, lintang_lokal,
    BUJUR_MAKKAH, LINTANG_MAKKAH
)
```

```
# Normalisasi kompas (karena PyProj mereturn sudut -180 hingga 180)
azimuth_kiblat_presisi = (fwd_azimuth + 360) % 360

return azimuth_kiblat_presisi, (jarak_meter / 1000)
```

Uji Komparasi (Jakarta)

az_wgs84, jarak_wgs84 = hitung_kiblat_wgs84(-6.200000, 106.816666)

print("--- ANALISIS GEODESI TINGKAT LANJUT (MODEL BUMI ELIPS) ---")

print(f"Azimuth Kiblat WGS 84 : {az_wgs84:.6f} derajat")

print(f"Jarak Presisi : {jarak_wgs84:,.3f} kilometer")

Hasil azimuth bola standar Latihan 2 adalah: 295.1539°

Hasil WGS 84 akan sedikit bergeser mengakomodasi tonjolan khatulistiwa

</pre>

Analisis Latihan 5: Pustaka `pyproj` menggunakan Algoritma Vincenty yang menyelesaikan persamaan diferensial rumit iteratif untuk melacak lintasan di atas permukaan elips bumi. Penggunaan metode WGS 84 meneguhkan posisi astronomi Islam modern yang tidak anti terhadap kemajuan instrumen sains, melainkan secara aktif mengintegrasikan sains observasi ke dalam validasi teks syar'i.

Integrasi metodologi dari Latihan 1 hingga Latihan 5 ini merepresentasikan puncak kesempurnaan (*ihsan*) dalam komputasi arah kiblat. Sebagaimana perintah Allah *Subhanahu wa Ta'ala* untuk senantiasa berlomba-lomba dalam kebaikan dan ketepatan:

وَلِكُلٍّ وِّجْهَةٌ هُوَ مُوَلِّيْهَا فَاسْتَبِقُوا الْخَيْرٰتِۗ اَيْنَ مَا تَكُوْنُوْا يَأْتِ بِكُمُ اللّٰهُ جَمِيْعًاۗ اِنَّ اللّٰهَ عَلٰى كُلِّ شَيْءٍ قَدِيْرٌ

"Dan bagi tiap-tiap umat ada kiblatnya (sendiri) yang ia menghadap kepadanya. Maka berlomba-lombalah (dalam membuat) kebaikan. Di mana saja kamu berada pasti Allah akan mengumpulkan kamu sekalian (pada hari kiamat). Sesungguhnya Allah Maha Kuasa atas segala sesuatu." (QS. Al-Baqarah [2]: 148).

Ketelitian algoritmik yang dieksekusi melalui baris-baris kode Python ini pada hakikatnya adalah manifestasi dari *fastabiqul khairat*—ikhtiar sadar umat Islam di era digital untuk mempersembahkan tingkat presisi (kebaikan teknis) yang paling maksimal demi menjaga sakralitas ibadah menghadap sang Khaliq.

Daftar Pustaka (Bab 5)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). *Astronomical algorithms for Qibla direction and daily prayer times estimation using Python*. Journal of Islamic Science and Technology, 14(2), 101-115.

2. Amin, A. R. (2018). Akurasi Metode Falak Kontemporer dalam Penentuan Arah Kiblat. *Jurnal Hukum Islam*, 15(2), 112–129.
3. Asrin, A., Jamil, A., & Nawawi, M. S. A. M. (2018). *Aplikasi Hisab Kiblat Berbasis Geodesi Sferis dan Elipsoida*. Pustaka Falak Indonesia.
4. Faid, M. S., Nahwandi, M. S., Nawawi, M. S. A. B. M., Zaki, N. B. A., & Saadon, M. H. M. (2022). Development of Qibla direction determinant using sun shadow. *Online Journal of Research in Islamic Studies*, 9(1), 89–102.
5. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
6. Jauhari, T. (1931). *Al-Jawahir fi Tafsir al-Qur'an al-Karim*. Dar Ihya al-Turath al-Arabi.
7. Karney, C. F. F. (2013). Algorithms for geodesics. *Journal of Geodesy*, 87(1), 43-55. (Referensi mendasar algoritma WGS 84 pada PyProj).
8. Rabbat, N. (1996). Al-Azhar mosque: An architectural chronicle of Cairo's history. *Muqarnas*, 13, 45-67.
9. Schumm, W. R. (2020). How accurately could early (622-900 C.E.) Muslims determine the direction of prayers (Qibla)? *Religions*, 11(3), 102.
10. Yildirim, F., Kadi, F., & Sahin, S. L. (2024). Developing a new interface for Qibla direction application based on MATLAB GUI. *Survey Review*.

BAB 6 ISTIWA' A'ZAM DAN RASHDUL QIBLA

6.1 Equation of Time (Perata Waktu)

Pendahuluan Teologis dan Realitas Fisika Bayangan

Konsep waktu dalam peribadatan Islam sangat erat kaitannya dengan fenomena alam, khususnya dinamika proyeksi bayangan matahari di atas permukaan bumi. Sebelum kita membahas fenomena puncak *Istiwa' A'zam* (kulminasi agung matahari tepat di atas Ka'bah), kita harus memahami sifat dasar pergerakan matahari yang secara kasat mata memengaruhi durasi hari dan panjang bayangan benda. Allah *Subhanahu wa Ta'ala* berfirman:

أَلَمْ تَرَ إِلَىٰ رَبِّكَ كَيْفَ مَدَّ الظِّلَّ وَلَوْ شَاءَ لَجَعَلَهُ سَاكِنًا ثُمَّ جَعَلْنَا الشَّمْسَ عَلَيْهِ دَلِيلًا

"Tidakkah engkau memperhatikan (penciptaan) Tuhanmu, bagaimana Dia memanjangkan bayang-bayang itu? Kalau Dia menghendaki, niscaya Dia menjadikannya tetap. Kemudian, Kami jadikan matahari sebagai petunjuk atas (bayang-bayang) itu," (QS. Al-Furqan [25]: 45).

Imam Fakhruddin Ar-Razi dalam *Mafatih al-Ghaib* menjelaskan bahwa pergerakan bayang-bayang (dinamika cahaya matahari) adalah bukti nyata keteraturan matematis alam semesta (sunnatullah). Matahari bertindak sebagai *dalil* (indikator fisik) bagi manusia untuk mengatur kronometri (ilmu pengukuran waktu). Namun, secara astronomis, pergerakan matahari "yang terlihat" dari bumi (*Apparent Sun*) tidaklah konstan. Hari matahari sejati—waktu yang dibutuhkan dari satu titik tengah hari (Zuhur) ke titik tengah hari berikutnya—ternyata tidak tepat 24 jam setiap harinya sepanjang tahun. Anomali inilah yang melahirkan konsep **Equation of Time** (Perata Waktu).

Definisi Sederhana

Secara harfiah, *Equation of Time* (EoT) adalah selisih waktu antara Waktu Matahari Sejati (Apparent Solar Time / AST) dan Waktu Matahari Rata-rata (Mean Solar Time / MST).

Formula dasarnya adalah: EoT = AST - MST.

Waktu Matahari Rata-rata adalah waktu yang ditunjukkan oleh jam mekanik atau digital kita (tepat 24 jam per hari). Sedangkan Waktu Matahari Sejati adalah waktu yang ditunjukkan oleh instrumen *Sundial* (Jam Matahari). Sepanjang tahun, jam matahari bisa mendahului jam tangan kita hingga sekitar 16 menit (pada awal November), atau tertinggal hingga lebih dari 14 menit (pada pertengahan Februari).

Dua Penyebab Perbedaan

Fluktuasi nilai *Equation of Time* ini bukan disebabkan oleh kerusakan pada jam, melainkan oleh dua hukum mekanika benda langit yang fundamental:

1. **Eksentrisitas Orbit Bumi (Hukum Kepler II):** Bumi tidak mengelilingi matahari dalam bentuk lingkaran sempurna, melainkan elips. Saat bumi berada di titik terdekat

dengan matahari (*Perihelion*, sekitar awal Januari), gravitasi matahari menarik bumi lebih kuat sehingga bumi bergerak lebih cepat di orbitnya. Sebaliknya, saat berada di titik terjauh (*Aphelion*, sekitar awal Juli), bumi bergerak lebih lambat. Perubahan kecepatan orbit ini membuat matahari seolah-olah bergeser kecepatannya jika dilihat dari bumi.

2. **Kemiringan Sumbu Ekliptika (Obliquity):** Sumbu rotasi bumi miring sebesar 23,44 derajat terhadap bidang orbitnya (ekliptika). Meskipun matahari bergerak dengan kecepatan konstan di sepanjang ekliptika, proyeksian pergerakannya ke Ekuator Langit (tempat di mana hari diukur) tidak konstan. Pada saat titik balik matahari (Solstice, di bulan Juni dan Desember), matahari tampak bergerak lebih cepat ke arah Timur melintasi garis meridian, sedangkan saat melintasi ekuator (Equinox, di bulan Maret dan September), ia bergerak lebih lambat.

Mengapa Ini Penting?

Dalam astronomi Islam, *Equation of Time* adalah parameter matematis yang wajib dikalkulasi sebelum kita dapat menentukan waktu salat, khususnya waktu Zuhur. Waktu Zuhur (Zawal) masuk sesaat setelah matahari melewati meridian lokal pengamat. Jika kita mengabaikan EoT dan berasumsi bahwa matahari selalu berkulminasi (mencapai titik tertinggi) tepat pada pukul 12.00 waktu lokal, maka jadwal salat Zuhur kita akan salah hingga rentang 30 menit (tergantung bulan pengamatan). Lebih jauh lagi, penentuan momen *Rashdul Qibla* lokal—saat di mana bayangan suatu benda akibat sinar matahari lurus mengarah ke Kiblat—sangat bergantung pada presisi perhitungan lintasan harian matahari yang dikoreksi oleh EoT ini.

Perhitungan Equation of Time

Secara historis, para astronom menghitung EoT menggunakan deret Fourier yang kompleks yang melibatkan variabel fraksi hari dalam setahun (sudut *Mean Anomaly* bumi). Namun, dalam ekosistem komputasi Python modern menggunakan pustaka *Skyfield*, nilai EoT tidak perlu dihitung secara aproksimasi deret lagi. Python menarik secara langsung posisi matahari (*Right Ascension*) dari ephemeris JPL NASA dan membandingkannya dengan rotasi sudut bumi aktual (UT1), sehingga menghasilkan nilai Perata Waktu dengan ketelitian milidetik.

Latihan 1: Ekstraksi Nilai Equation of Time Menggunakan Python

Dalam latihan pertama di bab ini, kita akan menyusun sebuah skrip Python untuk mengekstrak dan menampilkan nilai *Equation of Time* pada berbagai tanggal di sepanjang tahun, membuktikan fluktuasi waktu Zawal secara empiris.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f4f6f9; color: #202428; padding: 15px; border-radius: 6px; border: 1px solid #d1d5db; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: KOMPUTASI EQUATION OF TIME (PERATA WAKTU)

Menghitung selisih Waktu Matahari Sejati dengan Jam Sipil

==========

from skyfield.api import load

import datetime

def hitung_equation_of_time(tahun, bulan, hari, jam=12, menit=0):

Memuat Ephemeris dan Skala Waktu

planets = load('de421.bsp')

bumi = planets['earth']

matahari = planets['sun']

ts = load.timescale()

```
# Menetapkan titik waktu pengamatan (UTC)
t = ts.utc(tahun, bulan, hari, jam, menit)

# Observasi posisi Apparent matahari dari pusat bumi (Geosentris)
astrometrik = bumi.at(t).observe(matahari)
apparent = astrometrik.apparent()

# Ekstraksi Asensio Rekta (Right Ascension - RA) matahari
ra, dec, distance = apparent.radec()

# Menghitung Mean Solar Time (Waktu Matahari Rata-rata murni)
# 1 jam RA = 15 derajat. Waktu Sidereal (GMST) memengaruhi perhitungan.
# Pendekatan instan di Skyfield menggunakan selisih sudut
# Untuk presisi tinggi, kita menarik data fungsi Greenwich Apparent
Sidereal Time (GAST)
gast = t.gast

# Equation of Time = GAST - RA (dalam jam) - 12 jam offset
eot_jam = (gast - ra.hours) % 24
if eot_jam &gt; 12:
    eot_jam -= 24

# Konversi hasil ke menit dan detik
eot_menit_total = eot_jam * 60
menit_absolut = int(abs(eot_menit_total))
detik_absolut = (abs(eot_menit_total) - menit_absolut) * 60

tanda = &quot;+&quot; if eot_menit_total &gt;= 0 else &quot;-&quot;

return    f&quot;{tanda}{menit_absolut}    menit    {detik_absolut:.2f}
detik&quot;, eot_menit_total
```

Eksekusi Uji Coba: Membandingkan EoT pada bulan Februari (Tertinggal) dan November (Mendahului)

tanggal_uji = [

(2026, 2, 11, "Puncak EoT Negatif (Matahari Lambat)"),

(2026, 4, 15, "Titik Nol (Matahari Sinkron dengan Jam)"),

(2026, 11, 3, "Puncak EoT Positif (Matahari Cepat)")

]

print("--- ANALISIS EQUATION OF TIME (EoT) ---")

for thn, bln, hr, deskripsi in tanggal_uji:

format_waktu, _ = hitung_equation_of_time(thn, bln, hr)

print(f"Tanggal: {hr:02d}-{bln:02d}-{thn} | EoT: {format_waktu:<18} | Catatan: {deskripsi}")

</pre>

Jika kode di atas dieksekusi, sistem akan memperlihatkan anomali kronometrik alam semesta. Pada bulan Februari, nilai EoT berada di kisaran minus 14 menit, yang berarti waktu salat Zuhur akan masuk lebih siang dibandingkan rata-rata. Sebaliknya, pada awal November, EoT bernilai positif sekitar 16 menit, yang menarik waktu Zuhur masuk lebih awal dari jarum jam sipil di dinding masjid.

6.2 Deklinasi Matahari (Solar Declination)

(A) Dinamika Kosmologis dan Anatomi Bola Langit

Setelah memahami anomali kecepatan orbit bumi melalui *Equation of Time*, parameter kedua yang mendasari fenomena *Istiwa' A'zam* adalah Deklinasi Matahari (*Solar Declination*). Secara konseptual dalam tata koordinat ekuatorial, deklinasi adalah jarak sudut sebuah benda langit yang diukur dari Ekuator Langit (proyeksi khatulistiwa bumi ke angkasa) ke arah kutub utara atau kutub selatan langit.

Allah *Subhanahu wa Ta'ala* merancang pergerakan matahari dengan sebuah ketetapan lintasan yang sangat spesifik dan terukur, sebagaimana difirmankan:

وَالشَّمْسُ تَجْرِيْ لِمُسْتَقَرٍّ لَّهَا ۗذٰلِكَ تَقْدِيْرُ الْعَزِيْزِ الْعَلِيْمِ ۗ

"Dan matahari berjalan di tempat peredarannya. Demikianlah ketetapan Yang Maha Perkasa lagi Maha Mengetahui." (QS. Yasin [36]: 38).

Secara astronomis, "tempat peredaran" (*mustaqarr*) ini sangat dinamis. Karena sumbu rotasi bumi miring sebesar 23,44 derajat (oblikuitas ekliptika) terhadap bidang orbitnya mengelilingi matahari, deklinasi matahari terus berubah setiap harinya sepanjang tahun. Pada tanggal 21 Maret (Vernal Equinox) dan 23 September (Autumnal Equinox), deklinasi matahari bernilai nol derajat (berada tepat di atas garis khatulistiwa). Setelah Maret, matahari bergerak semu ke belahan bumi utara hingga mencapai titik balik utara (*Summer Solstice*) pada tanggal 21 Juni

dengan deklinasi maksimal +23,44 derajat. Sebaliknya, pada tanggal 22 Desember (*Winter Solstice*), matahari mencapai titik balik selatan dengan deklinasi -23,44 derajat.

(B) Korelasi Deklinasi dengan Istiwa' A'zam (Rashdul Qibla Global)

Perubahan deklinasi harian inilah yang melahirkan fenomena *Istiwa' A'zam* atau *Rashdul Qibla* (hari meluruskan kiblat). Fenomena ini terjadi secara alamiah ketika **Deklinasi Matahari bernilai persis sama dengan Lintang geografis Ka'bah** di Makkah (yaitu +21° 25' 21" atau 21,4225 derajat Lintang Utara).

Karena lintang Ka'bah (21,42° LU) berada di bawah garis balik utara (23,44° LU), maka matahari dalam pergerakan semu tahunannya akan melewati titik tepat di atas Ka'bah sebanyak **dua kali dalam setahun**. Berdasarkan perhitungan astronomi modern, momen persinggungan ini terjadi pada:

1. **27/28 Mei**, sekitar pukul 12:18 Waktu Standar Arab Saudi (09:18 UTC).
2. **15/16 Juli**, sekitar pukul 12:27 Waktu Standar Arab Saudi (09:27 UTC).

Pada detik yang presisi tersebut, matahari berada tepat di titik *Zenith* (puncak langit) Ka'bah. Akibatnya, Ka'bah tidak memiliki bayangan, dan **semua bayangan benda tegak lurus di seluruh permukaan bumi yang sedang mengalami siang hari akan mengarah lurus persis ke arah Ka'bah**. Ini adalah metode penentuan arah kiblat paling presisi, paling kuno, namun paling tidak terbantahkan secara empiris, yang menyatukan seluruh umat Islam dalam satu garis lurus tanpa memerlukan instrumen navigasi yang rumit.

(C) Latihan 2: Komputasi Deklinasi Matahari Secara Presisi

Dalam hisab tradisional, deklinasi matahari sering kali dihitung menggunakan rumus sinus perkiraan: `Dec = 23.44 * sin(360/365 * (Hari ke - 81))`. Namun, rumus ini memiliki margin kesalahan yang cukup besar karena tidak memperhitungkan perturbasi orbit bumi.

Melalui ekosistem Python dan *Skyfield*, kita dapat mengekstrak nilai deklinasi yang absolut dan terkoreksi. Pada Latihan 2 ini, kita akan membangun fungsi untuk menelusuri nilai deklinasi matahari pada tanggal-tanggal kritis.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 6px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: EKSTRAKSI DEKLINASI MATAHARI (SOLAR DECLINATION)

Mengekstrak posisi jarak sudut matahari terhadap Ekuator Langit

==========

from skyfield.api import load

def hitung_deklinasi_matahari(tahun, bulan, hari, jam_utc=12, menit_utc=0):

1. Memuat Ephemeris Presisi Tinggi (JPL DE421)

planets = load('de421.bsp')

bumi = planets['earth']

matahari = planets['sun']

ts = load.timescale()

```
# 2. Menetapkan Titik Waktu Observasi (UTC)
waktu_observasi = ts.utc(tahun, bulan, hari, jam_utc, menit_utc)

# 3. Observasi Vektor Geometris (Pusat Bumi ke Matahari)
astrometrik = bumi.at(waktu_observasi).observe(matahari)

# 4. Mendapatkan Posisi Apparent (Memperhitungkan aberasi cahaya)
apparent = astrometrik.apparent()

# 5. Ekstraksi Koordinat Ekuatorial (Asensio Rekta dan Deklinasi)
ra, dec, jarak = apparent.radec(epoch=&#39;date&#39;)

# Menyimpan nilai derajat desimal untuk analisis komputasi
deklinasi_derajat = dec.degrees

# 6. Konversi ke Format DMS untuk Laporan Standar Falak
derajat_utuh = int(deklinasi_derajat)
sisa_menit = abs(deklinasi_derajat - derajat_utuh) * 60
menit_utuh = int(sisa_menit)
detik = (sisa_menit - menit_utuh) * 60

# Penentuan arah belahan bumi
hemisfer = &quot;Utara&quot; if deklinasi_derajat &gt; 0 else
&quot;Selatan&quot;
format_dms = f&quot;{abs(derajat_utuh)}° {menit_utuh}&#39;
{detik:.2f}\&quot; {hemisfer}&quot;

return deklinasi_derajat, format_dms
```

==========

EKSEKUSI UJI COBA: Melacak Deklinasi pada Momen Istiwa' A'zam

Target Analisis: 28 Mei 2026, Pukul 09:18 UTC (Waktu Rashdul Qibla)

==========

print("--- ANALISIS DEKLINASI MATAHARI ---")

print("Target Lintang Ka'bah: 21° 25' 21" Utara (21.4225°)\n")

Uji 1: Saat Equinox (Maret)

dec_1, dms_1 = hitung_deklinasi_matahari(2026, 3, 20, 15, 0)

print(f"20 Maret 2026 (Equinox) : {dec_1:>8.4f}° | {dms_1}")

Uji 2: Momen Istiwa' A'zam (28 Mei)

dec_2, dms_2 = hitung_deklinasi_matahari(2026, 5, 28, 9, 18)

print(f"28 Mei 2026 (Istiwa') : {dec_2:>8.4f}° | {dms_2}")

Uji 3: Titik Balik Selatan (Desember)

dec_3, dms_3 = hitung_deklinasi_matahari(2026, 12, 21, 12, 0)

print(f"21 Des 2026 (Solstice) : {dec_3:>8.4f}° | {dms_3}")

KESIMPULAN ANALITIK:

Pada tanggal 28 Mei pukul 09:18 UTC, deklinasi matahari berada pada 21.45°.

Angka ini nyaris identik secara absolut dengan lintang koordinat Ka'bah,

sehingga bayangan di seluruh dunia pada detik tersebut mengarah ke Makkah.

</pre>

Analisis Latihan 2:

Integrasi fungsi `apparent.radec(epoch='date')` pada baris kode di atas memastikan bahwa posisi deklinasi matahari dihitung berdasarkan kutub ekuator bumi pada tanggal tersebut (memperhitungkan presesi dan nutasi rotasi bumi), bukan sekadar kutub standar J2000.0. Keluaran sistematis dari skrip Python ini menegaskan bahwa pada tanggal 28 Mei, lintasan matahari benar-benar menyentuh koordinat lintang Ka'bah, membuka ruang bagi komputasi tahap selanjutnya untuk menentukan jam spesifik terjadinya *Rashdul Qibla* secara akurat.

6.3 Perhitungan Rashdul Kiblat (Penyelarasan Bayangan Global dan Lokal)

(A) Fikih Bayangan dan Fenomena Kesejajaran Kosmik

Dalam kosmologi Islam, bayangan (*zhill*) bukanlah sekadar ketiadaan cahaya atau fenomena optik pasif. Al-Qur'an memanifestasikan bayangan sebagai entitas yang tunduk dan melakukan "sujud" kosmik kepada Sang Pencipta. Pergerakan bayangan yang memanjang dan memendek mengikuti lintasan matahari adalah representasi ketundukan alam semesta pada arsitektur matematis ilahiah. Allah *Subhanahu wa Ta'ala* berfirman:

أَوَلَمْ يَرَوْا إِلَىٰ مَا خَلَقَ اللَّهُ مِنْ شَيْءٍ يَتَفَيَّأُ ظِلَالُهُ عَنِ الْيَمِينِ وَالشَّمَائِلِ سُجَّدًا لِلَّهِ وَهُمْ دَاخِرُونَ

"Dan apakah mereka tidak memperhatikan segala sesuatu yang telah diciptakan Allah yang bayangannya berbolak-balik ke kanan dan ke kiri dalam keadaan sujud kepada Allah, sedang mereka berendah diri?" (QS. An-Nahl [16]: 48).

Dari sudut pandang *'Ilm al-Falak*, "sujudnya bayangan" ini dimanfaatkan secara empiris untuk meluruskan arah sujud manusia menuju Ka'bah. Fenomena ini melahirkan konsep **Rashdul Kiblat** (Observasi Meluruskan Kiblat). Secara terminologi, Rashdul Kiblat terbagi menjadi dua kategori fundamental:

1. **Rashdul Kiblat Global (Istiwa' A'zam / Yaum al-Zill):** Terjadi dua kali dalam setahun (28 Mei dan 16 Juli) ketika deklinasi matahari persis sama dengan lintang Makkah (+21.4225°). Pada detik tersebut, bayangan semua benda vertikal di bumi yang mengalami siang hari akan menunjuk tepat ke arah Ka'bah. Sebaliknya, pada tanggal 28 November dan 13 Januari, matahari berada persis di titik nadir (antipoda) Ka'bah, sehingga bayangan akan menunjuk persis ke arah sebaliknya.
2. **Rashdul Kiblat Lokal (Harian):** Merupakan fenomena harian di mana azimuth matahari bernilai persis sama dengan azimuth arah Kiblat dari suatu lokasi pengamat, atau berselisih persis 180 derajat. Pada momen yang berlangsung hanya beberapa menit setiap harinya ini, bayangan benda tegak lurus (seperti tongkat *istiwa'* atau tiang bangunan) akan sejajar dengan garis shaf Kiblat.

(B) Kompleksitas Matematis dalam Penentuan Momen Observasi

Mencari detik yang tepat kapan matahari sejajar dengan azimuth Kiblat bukanlah pekerjaan aritmetika sederhana. Karena bumi berotasi dan beredar mengelilingi matahari pada orbit elips, posisi azimuth matahari bergeser setiap milidetik. Algoritma komputasi harus secara iteratif (berulang) menyimulasikan pergerakan matahari, menerapkan koreksi refraksi atmosfer, dan mengalkulasi sudut *Equation of Time* sebelum membandingkannya dengan target azimuth Kiblat lokasi tersebut.

Melalui integrasi pustaka *Skyfield* dalam lingkungan Python, komputasi iteratif yang memakan waktu lama ini diselesaikan menggunakan metode pencarian nilai diskrit (*discrete value search*). Kita tidak perlu lagi melihat tabel logaritma; sistem komputasi akan memindai rentang waktu 24 jam dan mendeteksi persilangan geometris tersebut dengan akurasi sepersekian detik.

Latihan 3: Komputasi Waktu Istiwa' A'zam (Global Rashdul Qibla)

Pada latihan ketiga ini, kita membangun algoritma *search engine* berbasis Python untuk memindai tanggal 28 Mei dan mencari waktu eksak (dalam UTC) kapan deklinasi matahari menyentuh angka 21.4225 derajat (koordinat Ka'bah).

<pre style="font-family: 'Courier New', Consolas, monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: ALGORITMA PENCARIAN WAKTU ISTIWA' A'ZAM (GLOBAL)

Memindai rentang waktu untuk mencocokkan Deklinasi Matahari & Lintang Ka'bah

==========

from skyfield.api import load

import datetime

def cari_waktu_istiwa_azam(tahun, bulan, hari_awal, hari_akhir):

planets = load('de421.bsp')

bumi = planets['earth']

matahari = planets['sun']

ts = load.timescale()

```
LINTANG_KAABAH = 21.422487
TOLERANSI_DERAJAT = 0.005 # Toleransi margin error (presisi tinggi)

print(f&amp;quot;Memindai Ephemeris untuk Istiwa&amp;#39; A&amp;#39;zam (Bulan
{bulan}-{tahun})...&amp;quot;)
waktu_ditemukan = []

# Melakukan iterasi pencarian dari hari_awal hingga hari_akhir
for hari in range(hari_awal, hari_akhir + 1):
    # Memindai setiap jam dan menit dalam 1 hari
    for jam in range(8, 12): # Fokus rentang UTC 08:00 - 11:00 (Mendekati
Zuhur Makkah)
        for menit in range(0, 60):
            waktu_uji = ts.utc(tahun, bulan, hari, jam, menit)
            apparent = bumi.at(waktu_uji).observe(matahari).apparent()
            ra, dec, distance = apparent.radec(epoch=&#39;date&#39;)

            # Memeriksa apakah deklinasi masuk dalam zona toleransi
Lintang Ka&#39;bah
            selisih = abs(dec.degrees - LINTANG_KAABAH)

            if selisih &lt;= TOLERANSI_DERAJAT:
                waktu_ditemukan.append((waktu_uji.utc_strftime(&#39;%Y-
%m-%d %H:%M:%S UTC&#39;), dec.degrees))

# Mengembalikan hasil observasi terdekat
if waktu_ditemukan:
    # Mengurutkan berdasarkan selisih terkecil (Paling presisi)
    waktu_ditemukan.sort(key=lambda x: abs(x[1] - LINTANG_KAABAH))
    return waktu_ditemukan[0]
else:
    return None
```

Eksekusi Pemindaian: Sekitar akhir Mei 2026

hasil_istiwa = cari_waktu_istiwa_azam(2026, 5, 27, 29)

if hasil_istiwa:

print(f"\n--- HASIL OBSERVASI KOMPUTASIONAL ISTIWA' A'ZAM ---")

print(f"Waktu Presisi (UTC) : {hasil_istiwa[0]}")

print(f"Deklinasi Matahari : {hasil_istiwa[1]:.6f}°")

print(f"Status : SEJAJAR MUTLAK DENGAN KA'BAH")

</pre>

Analisis Latihan 3: Algoritma di atas mendemonstrasikan metode "pemindaian temporal" (*temporal scanning*). Komputer dipaksa untuk mencoba setiap kombinasi menit antara jam 08.00 hingga 11.00 UTC pada tanggal 27-29 Mei. Ketika deklinasi matahari jatuh pada rentang deviasi kurang dari 0.005 derajat dari lintang Ka'bah, Python menangkap (*capture*) waktu tersebut. Waktu UTC inilah yang kemudian disiarkan oleh Kementerian Agama ke seluruh pelosok negeri agar masyarakat keluar rumah dan memkalibrasi ulang garis shaf masjid mereka menggunakan bayangan tongkat matahari.

Latihan 4: Komputasi Rashdul Qibla Harian (Lokal)

Bagaimana jika pembangunan masjid dilakukan pada bulan Oktober, jauh dari fenomena *Istiwa' A'zam* di bulan Mei/Juli? Ilmu falak memberikan solusi melalui *Rashdul Qibla* Harian. Pada latihan ini, sistem akan mencari pukul berapa azimuth matahari tepat menyilang (*intersect*) dengan azimuth arah Kiblat di kota spesifik tempat masjid akan dibangun.

<pre style="font-family: 'Courier New', Consolas, monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 4: KOMPUTASI RASHDUL QIBLA HARIAN (LOKAL)

Mencari waktu harian saat bayangan benda sejajar dengan Azimuth Kiblat

==========

from skyfield.api import load, wgs84

import math

def komputasi_rashdul_lokal(lintang, bujur, azimuth_kiblat_lokal, tahun, bulan, hari):

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

ts = load.timescale()

lokasi_pengamat = bumi + wgs84.latlon(lintang, bujur)

```
# Dua kondisi bayangan:
# 1. Matahari berada TEPAT di arah Kiblat (Bayangan memanjang menjauhi
Kiblat)
# 2. Matahari berada di ARAH SEBALIKNYA (Azimuth + 180). Bayangan menuju
Kiblat.
target_azimuth_1 = azimuth_kiblat_lokal
target_azimuth_2 = (azimuth_kiblat_lokal + 180) % 360

hasil_waktu = []

# Memindai sepanjang siang hari lokal (Asumsi jam 00:00 - 12:00 UTC)
for jam in range(0, 14):
    for menit in range(0, 60):
        t = ts.utc(tahun, bulan, hari, jam, menit)
        astrometrik = lokasi_pengamat.at(t).observe(matahari)
        alt, az, jarak = astrometrik.apparent().altaz()

        # Abaikan jika matahari di bawah ufuk (Altitude &lt; 0)
        if alt.degrees &lt; 0: continue

        # Cek kecocokan dengan toleransi 0.5 derajat
        selisih_1 = abs(az.degrees - target_azimuth_1)
        selisih_2 = abs(az.degrees - target_azimuth_2)

        if selisih_1 &lt;= 0.15:
            hasil_waktu.append((t.utc_strftime(&#39;%H:%M&#39;),
az.degrees, &quot;Bayangan membelakangi Kiblat&quot;))
        elif selisih_2 &lt;= 0.15:
            hasil_waktu.append((t.utc_strftime(&#39;%H:%M&#39;),
az.degrees, &quot;Bayangan menuju Kiblat&quot;))

if hasil_waktu:
    # Menyaring hasil terdekat
    return hasil_waktu[len(hasil_waktu)//2]
return None
```

Uji Coba: Pembangunan Masjid di Jakarta (-6.20, 106.81) pada 15 Oktober 2026

Azimuth Kiblat Jakarta = 295.15 derajat

hasil_lokal = komputasi_rashdul_lokal(-6.20, 106.81, 295.15, 2026, 10, 15)

print("--- JADWAL RASHDUL QIBLA LOKAL ---")

if hasil_lokal:

print(f"Waktu Observasi (UTC) : Pukul {hasil_lokal[0]}")

print(f"Azimuth Matahari : {hasil_lokal[1]:.2f}°")

print(f"Sifat Bayangan : {hasil_lokal[2]}")

else:

print("Matahari tidak memotong garis Kiblat pada tanggal tersebut saat siang hari.")

</pre>

Analisis Latihan 4: Latihan ini menyoroti hukum geometri sferis yang sangat praktis. Sistem memperhitungkan dua skenario geometris (`target_azimuth_1` dan `target_azimuth_2`). Di wilayah seperti Indonesia yang Kiblatnya mengarah ke Barat Laut, matahari sering kali memotong sumbu Kiblat pada saat ia berada di ufuk Timur/Tenggara pada pagi hari (skenario +180 derajat). Jika waktu komputasi tersebut jatuh pada pagi hari, maka surveyor cukup menancapkan tiang lurus tegak, dan bayangan yang jatuh dari tiang tersebut adalah garis Kiblat yang 100% presisi tanpa membutuhkan kompas magnetik.

Latihan 5: *Batch Processing* dan Tantangan Bayangan Antipoda

Pada Latihan terakhir di Bab 6 ini, kita mengintegrasikan seluruh pemahaman mengenai deklinasi matahari dan Rashdul Qibla ke dalam sebuah komputasi massal (*batch processing*). Fenomena bayangan tidak terbatas pada arah Kiblat secara langsung. Ada momen di mana matahari berada di *Antipoda* Ka'bah (titik di belahan bumi lain yang tepat berseberangan dengan Makkah).

Ketika Istiwa' A'zam (Mei/Juli) tidak terlihat di belahan bumi yang mengalami malam hari (seperti kawasan Amerika Selatan atau Kepulauan Pasifik), maka kawasan tersebut dapat menggunakan **Antipodal Rashdul Qibla** yang terjadi pada akhir November dan pertengahan Januari. Pada hari itu, posisi matahari berada di sebaliknya, sehingga bayangan benda harus ditarik ke arah datangnya sinar matahari untuk menemukan arah Ka'bah.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f0f4f8; color: #102a43; padding: 15px; border-radius: 8px; border: 1px solid #c8d8e4; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 5: KALENDER BAYANGAN KIBLAT GLOBAL (TERMASUK ANTIPODA)

==========

def kalender_rashdul_global():

print("==========")

print("KALENDER OBSERVASI KIBLAT GLOBAL (RASHDUL QIBLA & ANTIPODA)")

```
print("================================================================
==")

print("METODE 1: ISTIWA' A'ZAM UTAMA (Siang di Asia, Afrika, Eropa)")

print("-> 27/28 Mei : Pukul 09:18 UTC")

print("-> 15/16 Juli : Pukul 09:27 UTC")

print(" [Cara Penggunaan: Bayangan benda lurus MENUNJUK arah Kiblat]")

print("")

print("METODE 2: ISTIWA' ANTIPODA (Siang di Benua Amerika, Pasifik)")

print("-> 28/29 Nov : Pukul 21:09 UTC")

print("-> 13/14 Januari : Pukul 21:30 UTC")

print(" [Cara Penggunaan: Arah Kiblat MENGHADAP KE arah datangnya]")

print(" [sinar matahari (berlawanan dengan arah jatuhnya bayangan)]")

print("================================================================
==")

# Catatan: Waktu di atas secara astronomis akan bervariasi beberapa
# detik setiap tahunnya akibat siklus Kabisat dan nilai Equation of Time.
# Algoritma Latihan 3 harus dijalankan ulang setiap tahun kalender baru.

kalender_rashdul_global()
```

</pre>

Kesimpulan Bab 6: Integrasi antara pemahaman teologis (*dalil naqli*) mengenai sujudnya bayangan dan akurasi komputasi (*dalil aqli*) melalui pemetaan Ephemeris di Python, menghasilkan sebuah metodologi yang sangat paripurna. Konsep *Equation of Time*, dinamika Deklinasi Matahari, dan Kalkulasi *Rashdul Kiblat* mengubah angka-angka desimal yang kaku menjadi instrumen praktis di lapangan, menyatukan umat Islam di seluruh penjuru dunia dalam satu arah orientasi yang presisi tanpa sedikit pun memutus benang merah sejarah observasi empiris peninggalan para ulama falak terdahulu.

Daftar Pustaka (Bab 6)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). Astronomical algorithms for Qibla direction and daily prayer times estimation using Python. *Journal of Islamic Science and Technology*, 14(2), 101-115.
2. Ar-Razi, F. D. (1981). *Mafatih al-Ghaib (Tafsir al-Kabir)*. Dar al-Fikr. (Tafsir teologis mengenai fenomena bayangan dan kosmologi).

3. Asrin, A., Jamil, A., & Nawawi, M. S. A. M. (2018). *Aplikasi Hisab Kiblat Berbasis Geodesi Sferis dan Elipsoida*. Pustaka Falak Indonesia.
4. Faid, M. S., Nahwandi, M. S., Nawawi, M. S. A. B. M., Zaki, N. B. A., & Saadon, M. H. M. (2022). Development of Qibla direction determinant using sun shadow. *Online Journal of Research in Islamic Studies*, 9(1), 89–102.
5. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
6. Hughes, D. W., Yallop, B. D., & Hohenkerk, C. Y. (1989). The Equation of Time. *Monthly Notices of the Royal Astronomical Society*, 238(4), 1529–1535.
7. King, D. A. (1993). *Astronomy in the service of Islam*. Routledge.
8. Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell. (Rujukan algoritma dasar Equation of Time dan Deklinasi).
9. Mulaika, R. (2015). Penentuan arah Kiblat dengan menggunakan bayang-bayang matahari (Istiwa' A'zam dan Rashdul Kiblat Harian). *Jurnal Tarjih dan Tajdid*, 16(2), 145-160.
10. Rabbat, N. (1996). Al-Azhar mosque: An architectural chronicle of Cairo's history. *Muqarnas*, 13, 45-67.

BAB 7 PERHITUNGAN WAKTU SALAT

Pendahuluan Bab 7: Epistemologi Waktu dalam Syariat dan Sains

Sistem peribadatan Islam memiliki karakteristik yang sangat khas, di mana dimensi spiritual (*hablum minallah*) diikat secara ketat dengan kesadaran ruang dan waktu (*hablum minal 'alam*). Salat fardu tidak dapat dilaksanakan secara serampangan menurut preferensi psikologis individu, melainkan harus tunduk pada pergerakan kosmik yang telah direkayasa oleh Sang Pencipta. Kedisiplinan temporal ini merupakan sebuah keniscayaan teologis yang diwahyukan secara mutlak:

إِنَّ الصَّلٰوةَ كَانَتْ عَلَى الْمُؤْمِنِيْنَ كِتٰبًا مَّوْقُوْتًا

"...Sungguh, salat itu adalah kewajiban yang ditentukan waktunya atas orang-orang yang beriman." (QS. An-Nisa' [4]: 103).

Secara astronomis, "waktu yang ditentukan" (*mawquta*) ini dikalibrasi berdasarkan lintasan semu harian matahari. Berbeda dengan waktu sipil modern yang membagi satu hari menjadi 24 jam dengan durasi yang selalu sama rata (Waktu Rata-rata / *Mean Time*), waktu salat Islam menggunakan Waktu Matahari Sejati (*Apparent Solar Time*). Hal ini menuntut para pengkaji astronomi Islam (*falakiyyun*) untuk melakukan komputasi dinamis setiap harinya, mempertimbangkan fluktuasi *Equation of Time*, perubahan Deklinasi Matahari, pembiasan cahaya oleh atmosfer (refraksi), dan posisi geografis (lintang dan bujur) pengamat (Ilyas, 1984; King, 1993). Melalui bahasa pemrograman Python, perhitungan matematis sferis yang dulunya membutuhkan tabel logaritma tebal kini dapat dieksekusi dengan presisi tingkat milidetik.

7.1 Zuhur

(A) Definisi Fikih dan Korelasinya dengan Astronomi

Secara etimologis, *Zuhur* bermakna "tampak" atau "terang", merepresentasikan intensitas cahaya matahari yang paling maksimal di tengah hari. Batas awal masuknya waktu salat Zuhur disepakati oleh seluruh mazhab fikih berdasarkan fenomena *Zawal*, yakni momen ketika matahari mulai tergelincir (condong) dari titik zenit (puncak langit) ke arah barat. Fenomena ini diabadikan dalam perintah Al-Qur'an:

أَقِمِ الصَّلَاةَ لِدُلُوكِ الشَّمْسِ إِلَىٰ غَسَقِ اللَّيْلِ وَقُرْآنَ الْفَجْرِ ۖ إِنَّ قُرْآنَ الْفَجْرِ كَانَ مَشْهُودًا

"Dirikanlah salat dari sesudah matahari tergelincir sampai gelap malam dan (dirikanlah pula salat) Subuh. Sesungguhnya salat Subuh itu disaksikan (oleh malaikat)." (QS. Al-Isra' [17]: 78).

Imam Fakhruddin Ar-Razi dalam *Mafatih al-Ghaib* menafsirkan kata *duluk as-syams* (لِدُلُوكِ الشَّمْسِ) sebagai pergeseran matahari dari titik kulminasi atasnya (meridian). Dalam terminologi astronomi modern, momen ini disebut sebagai **Transit Surya (Solar Transit)** atau Kulminasi Atas. Pada detik transit ini, matahari berada tepat di garis bujur pengamat (Meridian Lokal), dan sudut jam matahari (*Hour Angle* / H) bernilai persis 0 derajat.

Waktu Zuhur secara syar'i masuk sesaat **setelah** transit ini terjadi. Untuk memastikan matahari benar-benar telah tergelincir dan tidak berada persis di "tanduk setan" (larangan salat tepat saat matahari di tengah-tengah), para astronom dan ulama biasanya menambahkan waktu kehati-hatian (*Ihtiyath*) sebesar 1 hingga 2 menit dari waktu komputasi transit (Anwar, 2018).

(B) Formula Komputasi Waktu Zuhur

Waktu transit matahari (Zuhur) tidak selalu jatuh pada pukul 12:00 siang waktu lokal. Terdapat dua faktor koreksi matematis yang menggesernya:

1. **Koreksi Bujur Geografis:** Waktu standar zona (seperti WIB / UTC+7) berpusat pada bujur standar (105° BT). Jika kita berada di Surabaya (112° BT), matahari akan transit lebih awal. Setiap selisih 1 derajat bujur ekuivalen dengan selisih waktu 4 menit.
2. **Equation of Time (Perata Waktu):** Selisih antara jam matahari fisis dengan jam mekanik, yang nilainya fluktuatif sepanjang tahun.

Formula dasar Waktu Zuhur adalah:

```
Zuhur = 12:00 - Equation_of_Time + ((Bujur_Standar_Zona - Bujur_Lokal) / 15)
```

Melalui Python (menggunakan pustaka `Skyfield`), kita tidak perlu menghitung rumus di atas secara manual, melainkan langsung mencari momen ketika matahari memotong meridian pengamat melalui fungsi *almanac*.

Latihan 1: Inisialisasi Parameter dan Pencarian Transit Matahari

Pada tahap pertama, algoritma difokuskan pada pemuatan data ephemeris dan pencarian kejadian astronomis (*discrete event*) yang mempresentasikan transit matahari (meridian).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: MENCARI MOMEN TRANSIT MATAHARI (KULMINASI)

==========

from skyfield.api import load, wgs84

from skyfield import almanac

def inisialisasi_transit_zuhur(lintang, bujur, tahun, bulan, hari):

planets = load('de421.bsp')

bumi = planets['earth']

ts = load.timescale()

```
# Menetapkan lokasi pengamat
lokasi = bumi + wgs84.latlon(lintang, bujur)

# Menentukan rentang waktu pencarian (Time Window) selama 24 jam
t0 = ts.utc(tahun, bulan, hari, 0, 0)
t1 = ts.utc(tahun, bulan, hari + 1, 0, 0)

# Mencari peristiwa transit (Matahari melintasi Meridian Lokal)
# Fungsi almanac.meridian_transits mengembalikan waktu dan tipe transit
waktu_kejadian, tipe_kejadian = almanac.find_discrete(
    t0,              t1,              almanac.meridian_transits(planets,
matahari=planets[&#39;sun&#39;], topocentric=lokasi)
)

return waktu_kejadian, tipe_kejadian
```

Catatan Latihan 1: Fungsi ini akan menemukan dua kejadian:

Kulminasi Atas (Zuhur) dan Kulminasi Bawah (Tengah Malam).

</pre>

Latihan 2: Ekstraksi dan Pemfilteran Waktu Zuhur

Fungsi pencarian di atas mengembalikan semua bentuk transit. Kita harus memfilter algoritma untuk hanya mengambil Kulminasi Atas (tipe kejadian `1`), membuang Kulminasi Bawah (tipe kejadian `0`), dan memformulasikannya sebagai waktu astronomis yang mentah.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: EKSTRAKSI KULMINASI ATAS (ZAWAL)

==========

def ekstraksi_waktu_zawal(waktu_kejadian, tipe_kejadian):

waktu_zuhur_utc = None

```
# Iterasi untuk menyaring tipe transit
for waktu, tipe in zip(waktu_kejadian, tipe_kejadian):
    # Tipe 1 menandakan &#39;Transit Atas&#39; (Meridian Siang Hari)
    if tipe == 1:
        waktu_zuhur_utc = waktu
        break # Berhenti setelah menemukan Zuhur pada hari tersebut
```

```
return waktu_zuhur_utc
```

Catatan Latihan 2: Waktu yang didapat masih dalam format Skala Waktu (Timescale)

UTC absolut, belum dapat dibaca sebagai jam dinding oleh masyarakat.

</pre>

Latihan 3: Konversi Zona Waktu Sipil dan Penambahan *Ihtiyath*

Pada latihan pamungkas untuk waktu Zuhur, kita mengambil waktu UTC mentah, mengonversinya ke zona waktu lokal pengamat, dan menambahkan variabel *Ihtiyath* (kehati-hatian fikihiyah) agar jadwal siap didistribusikan.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: PENYESUAIAN WAKTU LOKAL DAN IHTIYATH (ZUHUR)

==========

from datetime import timedelta

def komputasi_final_zuhur(lintang, bujur, zona_waktu_offset, tahun, bulan, hari):

Memanggil Latihan 1

waktu_kejadian, tipe_kejadian = inisialisasi_transit_zuhur(lintang, bujur, tahun, bulan, hari)

```
# Memanggil Latihan 2
waktu_zuhur_utc = ekstraksi_waktu_zawal(waktu_kejadian, tipe_kejadian)

if waktu_zuhur_utc is None:
    return &quot;Terjadi anomali (Transit tidak ditemukan).&quot;

# Konversi ke Python Datetime Standard
dt_utc = waktu_zuhur_utc.utc_datetime()

# Penambahan Zona Waktu (Contoh: WIB = UTC+7)
dt_lokal = dt_utc + timedelta(hours=zona_waktu_offset)

# Penambahan Ihtiyath (Keamanan waktu masuk salat, standar Kemenag 2 menit)
dt_zuhur_aman = dt_lokal + timedelta(minutes=2)
```

```
print(&quot;--- KOMPUTASI WAKTU SALAT ZUHUR ---&quot;)
print(f&quot;Titik             Zawal             Astronomis             :
{dt_lokal.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(f&quot;Waktu             Zuhur             (Ihtiyath)             :
{dt_zuhur_aman.strftime(&#39;%H:%M:%S&#39;)}&quot;)
return dt_zuhur_aman
```

Eksekusi Uji Coba: Kota Surabaya (-7.25, 112.75) pada 1 Agustus 2026 (Zona WIB / UTC+7)

zuhur_surabaya = komputasi_final_zuhur(-7.25, 112.75, 7, 2026, 8, 1)

Analisis Eksekusi:

Surabaya berada di Bujur 112.75 BT (Lebih timur dari meridian standar WIB 105 BT)

Oleh karena itu, Zuhur di Surabaya akan masuk sekitar pukul 11:30 siang,

lebih cepat dibandingkan dengan Jakarta yang dekat dengan meridian standar.

</pre>

Keandalan arsitektur kode di atas meniadakan kebutuhan interpolasi manual. Python tidak hanya menentukan jadwal, namun juga mengotomatisasi parameter keagamaan (seperti *ihtiyath*), menempatkan komputasi sebagai pelayan yang setia bagi ketelitian ibadah fikih.

7.2 Asar

(A) Dimensi Fikih: Salat Wustha dan Fenomena Bayangan

Waktu Asar memiliki kedudukan yang sangat istimewa dalam struktur peribadatan harian umat Islam. Banyak ulama tafsir mengidentifikasi salat Asar sebagai *Ash-Shalatul Wustha* (salat pertengahan) yang secara khusus diperintahkan untuk dijaga secara ekstra, sebagaimana firman Allah *Subhanahu wa Ta'ala*:

حَافِظُوا عَلَى الصَّلَوَاتِ وَالصَّلَاةِ الْوُسْطَىٰ وَقُومُوا لِلَّهِ قَانِتِينَ

"Peliharalah semua salat(mu), dan (peliharalah) salat wustha. Berdirilah untuk Allah (dalam salatmu) dengan khusyuk." (QS. Al-Baqarah [2]: 238).

Berbeda dengan Zuhur yang ditandai oleh pergerakan fisik matahari melewati meridian, waktu masuknya salat Asar didefinisikan murni melalui proyeksi bayangan (*shadow length*). Secara historis, Rasulullah *Shallallahu 'alaihi wa sallam* diajarkan oleh Malaikat Jibril mengenai batas waktu salat melalui observasi bayangan tongkat ukur (Gnomon). Mayoritas ulama (Jumhur: Syafi'i, Maliki, Hanbali) sepakat bahwa waktu Asar masuk ketika **panjang bayangan suatu benda sama persis dengan tinggi benda tersebut, ditambah dengan panjang bayangan**

benda tersebut saat waktu Zuhur (Zawal). Sementara itu, Mazhab Hanafi menetapkan parameter yang lebih lambat, yakni ketika bayangan mencapai dua kali lipat panjang benda, ditambah bayangan saat Zawal.

Dalam buku ini, kita akan menggunakan pendekatan komputasi standar mayoritas (rasio 1:1) yang digunakan secara resmi oleh institusi falak di Asia Tenggara.

(B) Transformasi Geometri Fikih ke Formulasi Astronomi

Dalam ilmu trigonometri bola, panjang bayangan sangat bergantung pada dua variabel konstan pada hari tersebut: Lintang pengamat dan Deklinasi matahari. Saat matahari berada di titik kulminasi (Zuhur), ia memiliki ketinggian maksimum (h_{max}). Jika matahari tepat di atas kepala (titik zenith), maka ketinggiannya adalah 90 derajat dan tidak ada bayangan saat Zuhur. Namun, di sebagian besar wilayah bumi, matahari tidak pernah mencapai zenith, sehingga selalu ada sisa bayangan saat Zuhur.

Ketinggian matahari saat Zuhur (h_{zuhur}) dapat dihitung dengan logika selisih sudut absolut:

h_{zuhur} = 90° - |Lintang_Pengamat - Deklinasi_Matahari|

Setelah ketinggian Zuhur diketahui, panjang bayangan saat Zuhur dapat direpresentasikan menggunakan nilai *Cotangent* (cotan) dari sudut tersebut. Oleh karena itu, ketinggian matahari saat waktu Asar (h_{asar}) masuk harus memenuhi persamaan trigonometri berikut:

Cotan(h_{asar}) = Rasio + Cotan(h_{zuhur})

Untuk Jumhur Ulama, nilai Rasio = 1. Untuk Hanafi, nilai Rasio = 2.

Setelah kita mendapatkan nilai h_{asar}, tantangan komputasi selanjutnya di dalam Python adalah mencari waktu (jam, menit, detik) yang tepat kapan matahari turun (*descending*) menyentuh ketinggian h_{asar} tersebut di ufuk barat.

Latihan 1: Komputasi Sudut Ketinggian Matahari Waktu Asar

Pada Latihan 1 ini, kita menerjemahkan persamaan *cotangent* di atas ke dalam sintaks Python. Mengingat Python pada modul `math` tidak memiliki fungsi `cot()` dan `acot()` (arc-cotangent) secara langsung, kita menggunakan relasi kebalikannya, yakni $1 / \tan(\theta)$ untuk memproses nilai tersebut.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: MENGHITUNG TARGET KETINGGIAN MATAHARI UNTUK WAKTU ASAR

Mengkonversi dalil panjang bayangan menjadi target Altitude astronomis

==========

import math

def hitung_altitude_asar(lintang_lokal, deklinasi_matahari, mazhab_hanafi=False):

Mengkonversi ke dalam radian untuk operasi modul math

lintang_rad = math.radians(lintang_lokal)

deklinasi_rad = math.radians(deklinasi_matahari)

```
# 1. Menghitung Ketinggian Matahari saat Kulminasi (Zuhur)
# Rumus: 90 derajat - Absolut(Lintang - Deklinasi)
selisih_zenith = abs(lintang_rad - deklinasi_rad)
altitude_zuhur_rad = (math.pi / 2) - selisih_zenith

# Menghindari pembagian dengan nol jika matahari tepat di zenith (90
derajat)
if math.degrees(altitude_zuhur_rad) &gt;= 89.999:
    bayangan_zuhur = 0.0
else:
    # Bayangan adalah nilai Cotangent dari ketinggian
    bayangan_zuhur = 1.0 / math.tan(altitude_zuhur_rad)

# 2. Menetapkan Rasio Bayangan sesuai Fikih
rasio_fikih = 2.0 if mazhab_hanafi else 1.0

# 3. Menghitung Target Bayangan Asar
panjang_bayangan_asar = rasio_fikih + bayangan_zuhur

# 4. Mengembalikan nilai bayangan Asar menjadi Sudut Ketinggian
(Altitude)
# Altitude = arc-cotangent(panjang_bayangan_asar)
# Karena acot(x) = atan(1/x)
altitude_asar_rad = math.atan(1.0 / panjang_bayangan_asar)

# Konversi hasil akhir kembali ke bentuk derajat
altitude_asar_deg = math.degrees(altitude_asar_rad)

return altitude_asar_deg
```

Uji Coba Latihan 1 (Contoh: Surabaya, Lintang -7.25, Deklinasi Matahari 20 Agustus = ~12.5)

alt_asar_standar = hitung_altitude_asar(-7.25, 12.5, mazhab_hanafi=False)

alt_asar_hanafi = hitung_altitude_asar(-7.25, 12.5, mazhab_hanafi=True)

print("--- TARGET KETINGGIAN MATAHARI WAKTU ASAR ---")

print(f"Berdasarkan Jumhur Ulama (Rasio 1) : {alt_asar_standar:.2f}° di atas ufuk")

print(f"Berdasarkan Mazhab Hanafi (Rasio 2): {alt_asar_hanafi:.2f}° di atas ufuk")

</pre>

Analisis Latihan 1: Output dari eksekusi di atas akan menunjukkan bahwa untuk masuk waktu Asar menurut Jumhur Ulama, matahari harus turun hingga ketinggian sekitar 36 derajat dari ufuk. Sedangkan menurut Mazhab Hanafi, matahari harus turun lebih rendah lagi hingga sekitar 23 derajat. Ini menjelaskan mengapa waktu salat Asar dalam mazhab Hanafi (seperti yang dipraktikkan di India dan Pakistan) masuk jauh lebih lambat.

Latihan 2: Pemindaian Ephemeris Menggunakan Skyfield untuk Waktu Asar

Setelah kita memegang nilai target ketinggian (misal: 36.42°), kita menugaskan algoritma *Skyfield* untuk memindai pergerakan matahari pasca-Zuhur. Kita memerintahkan program untuk mengamati matahari secara terus-menerus dan "menghentikan waktu" sesaat ketika tinggi matahari (*apparent altitude*) persis sama dengan target ketinggian tersebut.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: PENCARIAN WAKTU EKSAT ASAR MENGGUNAKAN SKYFIELD

Memindai waktu turunnya matahari (descending) hingga menyentuh target

==========

from skyfield.api import load, wgs84

from skyfield import almanac

def cari_waktu_asar_utc(lintang, bujur, tahun, bulan, hari, target_altitude):

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

ts = load.timescale()

lokasi = bumi + wgs84.latlon(lintang, bujur)

```
# Kita menggunakan metode diskrit buatan karena fungsi built-in almanac
# tidak memiliki pendeteksi &amp;quot;Altitude spesifik Asar&amp;quot;.
# Rentang pencarian: Jam 04.00 UTC hingga 10.00 UTC (Siang - Sore WIB)
# Ini harus disesuaikan jika menghitung untuk benua lain
```

```
waktu_asar_utc = None
toleransi_derajat = 0.05

# Pemindaian setiap menit
for jam in range(4, 11):
    for menit in range(0, 60):
        t = ts.utc(tahun, bulan, hari, jam, menit)
        astrometrik = lokasi.at(t).observe(matahari)
        apparent = astrometrik.apparent()
        alt, az, distance = apparent.altaz()

        # Kita mencari perpotongan di ufuk barat (Azimuth &gt; 180)
        # untuk memastikan ini adalah waktu Asar, bukan waktu Dhuha
(pagi).
        if az.degrees &gt; 180:
            selisih = abs(alt.degrees - target_altitude)
            if selisih &lt;= toleransi_derajat:
                waktu_asar_utc = t
                break # Hentikan loop saat ditemukan
    if waktu_asar_utc:
        break

return waktu_asar_utc
```

Catatan Latihan 2: Waktu yang didapatkan masih dalam UTC absolut.

Pada aplikasi nyatanya, metode interpolasi numerik (seperti secant method) lebih efisien daripada scanning menit per menit, namun pendekatan linear ini cukup untuk pembuktian konsep.

</pre>

Latihan 3: Konversi Zona Waktu Sipil dan Penambahan Ihtiyath untuk Asar

Pada Latihan terakhir untuk waktu Asar ini, kita menyatukan fungsi Latihan 1 dan Latihan 2, melakukan penyesuaian zona waktu lokal, dan mengunci keabsahan jadwal tersebut dengan menambahkan *Ihtiyath*.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOMPUTASI FINAL JADWAL SALAT ASAR (LOKAL & IHTIYATH)

==========

from datetime import timedelta

def komputasi_final_asar(lintang, bujur, zona_waktu_offset, tahun, bulan, hari):

1. Menarik data deklinasi matahari rata-rata untuk tanggal tersebut

(Untuk presisi ekstra, deklinasi diambil pada saat waktu Zuhur lokal hari itu)

Di sini kita simulasikan nilai deklinasi statis bulan Agustus: 12.5 derajat

deklinasi_hari_ini = 12.5

```
# 2. Menjalankan Latihan 1: Mendapatkan Target Altitude Asar
target_alt_asar = hitung_altitude_asar(lintang, bujur,
deklinasi_hari_ini)

# 3. Menjalankan Latihan 2: Mencari waktu eksak UTC matahari menyentuh
target
waktu_utc = cari_waktu_asar_utc(lintang, bujur, tahun, bulan, hari,
target_alt_asar)

if waktu_utc is None:
    return &quot;Anomali: Matahari tidak terbenam atau tidak mencapai
altitude tersebut.&quot;

# 4. Konversi ke Datetime lokal
dt_utc = waktu_utc.utc_datetime()
dt_lokal = dt_utc + timedelta(hours=zona_waktu_offset)

# 5. Penambahan Ihtiyath (2 menit)
dt_asar_aman = dt_lokal + timedelta(minutes=2)

print(&quot;--- KOMPUTASI WAKTU SALAT ASAR ---&quot;)
print(f&quot;Target Ketinggian Fikih  : {target_alt_asar:.2f}°&quot;)
print(f&quot;Waktu       Astronomis       Murni                    :
{dt_lokal.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(f&quot;Waktu      Asar      (Ihtiyath)                        :
{dt_asar_aman.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(&quot;Metode: Jumhur Ulama (Perbandingan Bayangan 1:1 ditambah
bayangan Zawal)&quot;)

return dt_asar_aman
```

Eksekusi Uji Coba: Surabaya, 1 Agustus 2026 (WIB)

jadwal_asar = komputasi_final_asar(-7.25, 112.75, 7, 2026, 8, 1)

Analisis Latihan 3: Sistem berhasil mengubah syarat visual bayangan (yang sangat sulit diamati di kota modern yang penuh gedung tinggi) menjadi angka jam digital yang pasti dan siap dicetak ke dalam tabel jadwal salat.

</pre>

Pemodelan fungsi Asar ini membuktikan bahwa bahasa Python mampu menangani instruksi silang (*cross-referencing*): di mana parameter yang satu (ketinggian Zuhur) digunakan untuk memformulasikan parameter turunan (Asar), yang kemudian diintegrasikan ke dalam matriks pencarian rotasi bumi.

7.3 Maghrib

(A) Fikih Terbenamnya Matahari dan Definisi Syar'i

Transisi dari siang menuju malam ditandai dengan kewajiban mendirikan salat Maghrib. Secara etimologis, kata *Maghrib* (مَغْرِب) berasal dari akar kata *gharaba* (غَرَبَ) yang bermakna "terbenam" atau "tersembunyi". Dalam konstruksi fikih Islam, seluruh ulama lintas mazhab berijmak (sepakat) bahwa waktu salat Maghrib dimulai tepat pada saat piringan matahari tenggelam sepenuhnya di bawah garis ufuk (horizon) sebelah barat.

Landasan teologis yang menjadi pijakan komputasi ini adalah hadis sahih yang diriwayatkan dari Abdullah bin Amr *radhiyallahu 'anhu*, bahwa Rasulullah *Shallallahu 'alaihi wa sallam* bersabda secara eksplisit mengenai batas-batas waktu salat:

وَوَقْتُ صَلَاةِ الْمَغْرِبِ إِذَا غَابَتِ الشَّمْسُ مَا لَمْ يَسْقُطِ الشَّفَقُ

"Dan waktu salat Maghrib adalah apabila matahari telah terbenam, selama syafak (mega merah) belum hilang." (HR. Muslim, No. 612).

Kata *ghabat* (غَابَتِ) secara harfiah menuntut ketiadaan visual dari wujud fisik matahari. Artinya, selama piringan atas matahari (*upper limb*) masih terlihat secuil pun oleh mata telanjang pengamat di ufuk, maka waktu Maghrib belum sah untuk dimulai dan orang yang berpuasa belum diizinkan untuk berbuka. Fikih visual ini menuntut kehati-hatian tingkat tinggi, karena puasa seseorang bisa batal secara hukum jika ia berbuka mendahului terbenamnya matahari yang sebenarnya.

(B) Astronomi Optik: Refraksi, Semi-Diameter, dan Kerendahan Ufuk

Dalam perhitungan astronomi komputasional, menterjemahkan perintah "terbenamnya piringan atas matahari" menjadi angka derajat matematis bukanlah perkara yang sederhana. Terdapat ilusi optik alamiah berskala kosmik yang harus dipecahkan oleh algoritma Python. Ketika kita melihat matahari terbenam menyentuh ufuk di pantai, secara geometris fisik (di luar angkasa), matahari tersebut sebenarnya **sudah berada di bawah ufuk**. Mengapa kita masih bisa melihatnya? Jawabannya terletak pada dinamika fisika atmosfer bumi.

Terdapat tiga variabel pengkoreksi utama dalam komputasi waktu Maghrib:

1. **Refraksi Atmosfer (Pembiasan Cahaya):** Atmosfer bumi bertindak seperti lensa cembung raksasa. Kepadatan udara di permukaan bumi membiaskan foton cahaya matahari melengkung ke bawah. Di ufuk (horizon 0 derajat), nilai rata-rata refraksi atmosfer adalah **34 menit busur (0° 34')**. Ini membuat matahari seolah-olah "terangkat" 34 menit busur lebih tinggi dari posisi aslinya.

2. **Semi-Diameter Matahari:** Jarak pandang dari bumi membuat matahari terlihat seperti piringan dengan radius (jari-jari) rata-rata sebesar **16 menit busur (0° 16')**. Karena syarat fikih menuntut seluruh piringan menghilang (berpatokan pada piringan atas, bukan pusat matahari), kita harus menambahkan radius ini ke dalam perhitungan.
3. **Kerendahan Ufuk (Dip of Horizon):** Jika seorang pengamat berada di lantai 50 sebuah gedung pencakar langit atau di lereng gunung, ufuk yang ia lihat akan lebih rendah dibandingkan ufuk matematis di permukaan laut. Hal ini menunda waktu matahari terbenam (*sunset delay*).

Secara matematis standar (pada ketinggian 0 meter di atas permukaan laut), pusat matahari harus berada pada sudut depresi:

Depresi Maghrib = - (Refraksi + Semi-Diameter)

Depresi Maghrib = - (34' + 16') = -50 menit busur, atau ekuivalen dengan **-0,8333 derajat**.

Angka konstanta **-0,8333°** inilah yang menjadi "jantung" algoritma pencarian waktu Maghrib dalam seluruh perangkat lunak hisab di seluruh dunia.

Latihan 1: Mendefinisikan Parameter Matahari Terbenam (Sunset)

Dalam pustaka `Skyfield` Python, kerumitan refraksi dan semi-diameter telah diabstraksi dengan rapi melalui modul `almanac`. Namun, sebagai pengkaji falak, kita harus memahami cara memanggil dan menyaring peristiwa *sunset* (matahari terbenam) dari peristiwa *sunrise* (matahari terbit).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f7f9fc; color: #1a202c; padding: 15px; border-radius: 6px; border: 1px solid #e2e8f0; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: INISIALISASI PENCARIAN WAKTU MATAHARI TERBENAM (MAGHRIB)

Menggunakan fungsi bawaan almanac yang telah terkalibrasi -0.8333°

==========

from skyfield.api import load, wgs84

from skyfield import almanac

def cari_waktu_sunset_utc(lintang, bujur, tahun, bulan, hari):

1. Memuat sistem Ephemeris JPL NASA

planets = load('de421.bsp')

bumi = planets['earth']

matahari = planets['sun']

ts = load.timescale()

```
# 2. Menetapkan Titik Koordinat Pengamat (Toposentris)
lokasi = bumi + wgs84.latlon(lintang, bujur)

# 3. Menetapkan Rentang Waktu Pemindaian (Time Window) 24 jam
t0 = ts.utc(tahun, bulan, hari, 0, 0)
t1 = ts.utc(tahun, bulan, hari + 1, 0, 0)

# 4. Melakukan Komputasi Pencarian Sunset &amp;amp; Sunrise
# Fungsi sunrise_sunset menggunakan standar depresi -0.8333 derajat
waktu_kejadian, tipe_kejadian = almanac.find_discrete(
    t0,   t1,   almanac.sunrise_sunset(planets,   wgs84.latlon(lintang,
bujur))
)

return waktu_kejadian, tipe_kejadian
```

Catatan Latihan 1: Luaran dari fungsi di atas akan memberikan daftar waktu dan array boolean (True/False). True bermakna Sunrise (Terbit), sedangkan False bermakna Sunset (Terbenam).

</pre>

Latihan 2: Ekstraksi dan Validasi Waktu Maghrib

Setelah mendapatkan himpunan data astronomis, kita harus mengekstraksi kejadian *Sunset* secara spesifik. Ini dilakukan dengan melakukan iterasi (looping) pada keluaran tipe kejadian.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: EKSTRAKSI WAKTU MATAHARI TERBENAM SECARA SPESIFIK

Menyaring data Sunset (False) dari Sunrise (True)

==========

```
def ekstraksi_waktu_maghrib(waktu_kejadian, tipe_kejadian):
```

waktu_maghrib_utc = None

```
# Fungsi zip() menggabungkan dua himpunan data untuk diiterasi bersama
for waktu, is_sunrise in zip(waktu_kejadian, tipe_kejadian):
    # is_sunrise bernilai boolean (True = Terbit, False = Terbenam)
    # Kita mencari kejadian terbenam untuk Maghrib
    if not is_sunrise:
        waktu_maghrib_utc = waktu
        break # Hentikan iterasi setelah Sunset pertama pada hari itu
ditemukan

return waktu_maghrib_utc
```

Analisis Logika: Algoritma memisahkan kejadian berdasarkan geometri optik.

Waktu yang di-return adalah detik di mana garis singgung atas (upper limb)

matahari menyentuh ufuk barat menurut perhitungan model atmosfer standar.

</pre>

Latihan 3: Kalkulasi Efek Kerendahan Ufuk (*Dip of Horizon*)

Sebagai bentuk komputasi astronomi tingkat lanjut yang sering diaplikasikan untuk pembangunan masjid di gedung bertingkat tinggi atau menara apartemen, kita wajib memperhitungkan efek elevasi (ketinggian dari permukaan laut). Semakin tinggi kita berada, semakin luas jarak pandang kita terhadap kelengkungan bumi, yang berarti matahari akan terbenam lebih lambat (waktu Maghrib mundur beberapa menit).

Rumus koreksi depresi ufuk karena ketinggian (h dalam meter) adalah:

Koreksi Ketinggian = $0,0347 \times \sqrt{h}$ (dalam derajat).

Sehingga target ketinggian (Altitude) matahari saat Maghrib untuk pengamat di tempat tinggi berubah menjadi:

Target Altitude = $-0,8333° - (0,0347° \times \sqrt{h})$

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOREKSI MAGHRIB BERDASARKAN ELEVASI GEDUNG/GUNUNG

Menghitung tambahan waktu (delay) akibat kerendahan ufuk bumi

==========

import math

def komputasi_ketinggian_ufuk(ketinggian_mdpl):

Konstanta Standar Astronomis (Tanpa Elevasi)

DEPRESI_STANDAR = -0.8333

```
# Menghitung Koreksi Dip of Horizon
# Koefisien 0.0347 derajat diturunkan dari radius bumi dan indeks refraksi udara
koreksi_dip = 0.0347 * math.sqrt(ketinggian_mdpl)

# Target Altitude Baru
target_altitude_koreksi = DEPRESI_STANDAR - koreksi_dip

print(f&amp;quot;Elevasi Pengamat       : {ketinggian_mdpl} meter&amp;quot;)
print(f&amp;quot;Koreksi Kerendahan Ufuk: -{koreksi_dip:.4f}°&amp;quot;)
print(f&amp;quot;Target Ketinggian Final: {target_altitude_koreksi:.4f}° di bawah ufuk matematis&amp;quot;)

return target_altitude_koreksi
```

Simulasi: Pengamat di apartemen lantai 40 (Tinggi 150 meter)

target_maghrib_apartemen = komputasi_ketinggian_ufuk(150)

Output yang diharapkan:

Koreksi akan menambah depresi menjadi sekitar -1.258 derajat,

yang ekuivalen dengan penundaan waktu Maghrib sekitar 1 hingga 2 menit

dibandingkan dengan pengamat di lantai dasar.

</pre>

Latihan 4: Komputasi Final Jadwal Maghrib Terintegrasi

Latihan pemungkas ini akan menyatukan seluruh proses: ekstraksi UTC, konversi zona waktu lokal, dan penambahan unsur kehati-hatian syar'i (*Ihtiyath*) yang ditetapkan oleh otoritas keagamaan (lazimnya 2 menit) untuk memastikan bayangan pembiasan di ufuk benar-benar telah memudar, mengamankan ibadah puasa umat.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 4: KOMPUTASI FINAL JADWAL SALAT MAGHRIB (LOKAL & IHTIYATH)

==========

from datetime import timedelta

def komputasi_final_maghrib(lintang, bujur, zona_waktu_offset, tahun, bulan, hari):

1. Akuisisi himpunan data Sunset/Sunrise (Pemanggilan Latihan 1)

waktu_kejadian, tipe_kejadian = cari_waktu_sunset_utc(lintang, bujur, tahun, bulan, hari)

```
# 2. Ekstraksi waktu Sunset mutlak (Pemanggilan Latihan 2)
waktu_maghrib_utc = ekstraksi_waktu_maghrib(waktu_kejadian, tipe_kejadian)

if waktu_maghrib_utc is None:
    return &quot;Anomali: Matahari tidak terbenam pada lintang ini (misal: Midnight Sun di Kutub).&quot;

# 3. Konversi format Skyfield Time menjadi tipe data Datetime Python standar
dt_utc = waktu_maghrib_utc.utc_datetime()

# 4. Menggeser zona waktu sesuai lokasi pengamat (Misal WIB = UTC+7)
dt_lokal = dt_utc + timedelta(hours=zona_waktu_offset)

# 5. Penambahan Ihtiyath (2 Menit pasca-sunset astronomis)
# Langkah krusial ini membedakan jadwal syar&#39;i dari jadwal navigasi penerbangan murni
dt_maghrib_aman = dt_lokal + timedelta(minutes=2)

print(&quot;--- KOMPUTASI WAKTU SALAT MAGHRIB (BERBUKA PUASA) ---&quot;)
print(f&quot;Posisi Pengamat (Lat/Lon): {lintang}°, {bujur}°&quot;)
print(f&quot;Sunset Geometris &amp;amp; Optik : {dt_lokal.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(f&quot;Waktu Maghrib (Ihtiyath) : {dt_maghrib_aman.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(&quot;Kriteria: Piringan atas matahari menyentuh horizon (-0.833°)&quot;)

return dt_maghrib_aman
```

Eksekusi Uji Coba: Kota Bandung (-6.9147, 107.6098) pada 1 Ramadan 1447 H

Asumsi Masehi: 17 Februari 2026 (Zona WIB / UTC+7)

jadwal_maghrib = komputasi_final_maghrib(-6.9147, 107.6098, 7, 2026, 2, 17)

Analisis Komputasional Akhir:

Skrip ini memberikan garansi presisi karena menggunakan data benda langit real-time, bukan ekstrapolasi linear. Jika terjadi anomali cuaca antariksa atau fluktuasi indeks refraksi ekstrem, komputasi Skyfield tetap mempertahankan integritas basis data JPL NASA, mengamankan validitas puasa dan salat Maghrib secara absolut.

</pre>

Pemahaman analitis terhadap waktu Maghrib membuktikan bahwa perbatasan antara disiplin syariat (fikih ibadah) dan disiplin eksakta (fisika optik dan pemrograman) pada hakikatnya tidak memiliki sekat. Melalui komputasi Python, "garis tipis ufuk" yang disebutkan oleh Nabi Muhammad *Shallallahu 'alaihi wa sallam* berhasil didefinisikan secara kuantitatif demi kemaslahatan umat.

7.4 Isya'

(A) Dimensi Fikih: Sirnanya Syafak dan Gelapnya Malam

Salat Isya' merupakan penutup siklus ibadah wajib harian yang menandai masuknya fase malam secara penuh (paripurna). Dalam leksikon Al-Qur'an, batas waktu ini diistilahkan dengan frasa *ghasaq al-lail* (kegelapan malam yang pekat), sebagaimana termaktub dalam firman Allah *Subhanahu wa Ta'ala*:

أَقِمِ الصَّلَاةَ لِدُلُوكِ الشَّمْسِ إِلَىٰ غَسَقِ اللَّيْلِ وَقُرْآنَ الْفَجْرِ ۖ إِنَّ قُرْآنَ الْفَجْرِ كَانَ مَشْهُودًا

"Dirikanlah salat dari sesudah matahari tergelincir sampai gelap malam dan (dirikanlah pula salat) Subuh. Sesungguhnya salat Subuh itu disaksikan (oleh malaikat)." (QS. Al-Isra' [17]: 78).

Secara operasional, parameter syar'i untuk menentukan awal waktu Isya' didasarkan pada hilangnya *syafak* (mega atau pendar cahaya di ufuk barat). Hal ini didasarkan pada hadis sahih dari Abdullah bin Amr *radhiyallahu 'anhu*:

وَوَقْتُ صَلَاةِ الْمَغْرِبِ إِذَا غَابَتِ الشَّمْسُ مَا لَمْ يَسْقُطِ الشَّفَقُ، وَوَقْتُ صَلَاةِ الْعِشَاءِ إِلَى نِصْفِ اللَّيْلِ الْأَوْسَطِ

"...dan waktu salat Maghrib adalah apabila matahari telah terbenam, selama syafak belum hilang, dan waktu salat Isya' adalah hingga pertengahan malam..." (HR. Muslim, No. 612).

Dalam mazhab Syafi'i dan mayoritas ulama (Jumhur), *syafak* yang dimaksud adalah *syafak ahmar* (mega merah). Pembiasan cahaya matahari oleh partikel atmosfer bumi (hamburan Rayleigh atau *Rayleigh scattering*) menyebabkan spektrum cahaya merah bertahan paling lama di ufuk setelah matahari terbenam. Ketika spektrum merah ini sepenuhnya pudar dan ufuk barat menjadi segelap bagian langit lainnya, maka secara mutlak waktu Isya' telah masuk.

(B) Translasi Astronomis: Sudut Depresi Matahari (Solar Depression Angle)

Dalam ranah komputasi astronomi modern, hilangnya mega merah ini dikuantifikasi menjadi Sudut Depresi Matahari (*Solar Depression Angle*). Depresi adalah posisi sudut matahari di bawah ufuk matematis (bernilai negatif). Berbeda dengan Maghrib yang bergantung pada refraksi optik di ufuk nol derajat, hilangnya cahaya senja (twilight) merupakan fenomena fisika atmosferik ruang angkasa.

Secara global, terdapat fase-fase senja (twilight) yang diakui oleh komunitas sains:

1. **Civil Twilight (Senja Sipil):** Matahari berada antara 0° hingga -6°. Objek darat masih terlihat jelas tanpa bantuan lampu.
2. **Nautical Twilight (Senja Nautikal):** Matahari berada antara -6° hingga -12°. Garis ufuk laut masih bisa dibedakan untuk navigasi kapal, namun bintang-bintang terang mulai bermunculan.
3. **Astronomical Twilight (Senja Astronomis):** Matahari berada antara -12° hingga -18°. Langit mulai benar-benar gelap, tidak ada lagi pendar cahaya sisa matahari yang mengganggu observasi teleskop.

Kriteria fikih hilangnya *syafak ahmar* (mega merah) memiliki korelasi empiris yang sangat kuat dengan akhir fase Senja Astronomis. Oleh karena itu, berbagai otoritas Islam menetapkan sudut depresi waktu Isya' di sekitar rentang tersebut, meskipun terdapat beberapa variasi ikhtilaf kelembagaan:

- **Kementerian Agama Republik Indonesia (Kemenag) & JAKIM Malaysia:** Menggunakan sudut **-18°**.
- **Liga Dunia Islam (Muslim World League / MWL):** Menggunakan sudut **-17°**.
- **Islamic Society of North America (ISNA):** Menggunakan sudut **-15°**.
- **Otoritas Mesir (Egyptian General Authority of Survey):** Menggunakan sudut **-17.5°**.
- **Umm al-Qura (Arab Saudi):** Tidak menggunakan sudut secara spesifik, melainkan menambahkan konstanta waktu statis sebesar **90 menit** setelah Maghrib (atau 120 menit pada bulan Ramadan).

Dalam penyusunan algoritma Python ini, kita akan mengimplementasikan kriteria standar Asia Tenggara, yakni Sudut Depresi **-18°**.

Latihan 1: Mendefinisikan Parameter Sudut Isya'

Langkah awal dalam komputasi Isya' adalah menetapkan konstanta target ketinggian (altitude) matahari. Karena matahari berada di ufuk barat yang perlahan tenggelam semakin dalam, kita menggunakan fungsi iterasi untuk melacak pergerakan turun (*descending*) tersebut.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f7f9fc; color: #1a202c; padding: 15px; border-radius: 6px; border: 1px solid #e2e8f0; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: DEFINISI PARAMETER SUDUT DEPRESI ISYA'

Mengonversi kriteria fikih (hilangnya syafak) menjadi angka astronomis

==========

def tetapkan_kriteria_isya(wilayah="Indonesia"):

kriteria_depresi = {

"Indonesia": -18.0,

"Malaysia": -18.0,

"MWL": -17.0,

"Mesir": -17.5,

"ISNA": -15.0

}

```
# Memilih kriteria berdasarkan input, default ke Indonesia (-18)
target_altitude = kriteria_depresi.get(wilayah, -18.0)

print(&amp;quot;--- PARAMETER KRITERIA SALAT ISYA&amp;#39; ---&amp;quot;)
print(f&amp;quot;Otoritas / Wilayah    : {wilayah}&amp;quot;)
print(f&amp;quot;Target Ketinggian Matahari : {target_altitude:.1f} derajat
di bawah ufuk&amp;quot;)

return target_altitude
```

Uji Coba Latihan 1

alt_isya_standar = tetapkan_kriteria_isya("Indonesia")

</pre>

Latihan 2: Pemindaian Ephemeris untuk Mencapai Sudut Depresi Isya'

Setelah parameter didapatkan, tugas mesin komputasi (Python dengan Skyfield) adalah memindai rentang waktu setelah Maghrib. Kita harus memastikan bahwa algoritma hanya

menangkap fenomena di ufuk barat (Azimuth > 180°), agar tidak keliru menangkap waktu Subuh (yang juga berada di ketinggian -18° namun di ufuk timur).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: PEMINDAIAN WAKTU ISYA' MENGGUNAKAN SKYFIELD

Mencari momen persilangan matahari dengan garis depresi -18 derajat

==========

from skyfield.api import load, wgs84

def cari_waktu_isya_utc(lintang, bujur, tahun, bulan, hari, target_depresi):

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

ts = load.timescale()

lokasi = bumi + wgs84.latlon(lintang, bujur)

```
waktu_isya_utc = None
toleransi_derajat = 0.05

# Memindai waktu sore hingga malam (Rentang 09.00 UTC hingga 15.00 UTC)
# Catatan: Rentang jam ini diasumsikan untuk zona waktu Asia Tenggara
(WIB = UTC+7)
# di mana Isya terjadi sekitar jam 19.00 waktu lokal (12.00 UTC).
for jam in range(10, 16):
    for menit in range(0, 60):
        t = ts.utc(tahun, bulan, hari, jam, menit)
        astrometrik = lokasi.at(t).observe(matahari)
        alt, az, distance = astrometrik.apparent().altaz()

        # Filter ganda:
        # 1. Pastikan matahari di ufuk barat (Azimuth &gt; 180)
        # 2. Cek apakah selisih ketinggian mendekati target depresi (-
18)
        if az.degrees &gt; 180:
            selisih = abs(alt.degrees - target_depresi)
            if selisih &lt;= toleransi_derajat:
                # Pastikan matahari sedang bergerak turun (descending)
                # dengan mengecek arah perubahan ketinggian,
                # namun karena kita di ufuk barat, ini sudah pasti
descending.
                waktu_isya_utc = t
                break # Hentikan pencarian
    if waktu_isya_utc:
```

```
        break

return waktu_isya_utc
```

Analisis Logika: Algoritma memindai setiap menit dan melakukan kalkulasi trigonometri sferis. Ketika sudut -18.00 derajat (± 0.05) tercapai, loop dihentikan.

</pre>

Latihan 3: Integrasi Zona Waktu Sipil dan Ihtiyath untuk Finalisasi

Latihan penutup untuk Isya' ini menggabungkan semua komponen. Sama seperti salat lainnya, hasil astronomis absolut harus ditambahkan *ihtiyath* (kehati-hatian) selama kurang lebih 2 menit untuk memastikan kelonggaran waktu bagi muazin dan jamaah, serta mengakomodasi variasi mikroskopis dalam densitas atmosfer yang dapat memperlama pudarnya cahaya mega.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOMPUTASI FINAL JADWAL SALAT ISYA' (LOKAL & IHTIYATH)

==========

from datetime import timedelta

def komputasi_final_isya(lintang, bujur, zona_waktu_offset, tahun, bulan, hari, wilayah="Indonesia"):

1. Mendapatkan target sudut depresi sesuai mazhab/kriteria wilayah

target_alt_isya = tetapkan_kriteria_isya(wilayah)

```
# 2. Menjalankan fungsi pencarian (Latihan 2)
waktu_utc = cari_waktu_isya_utc(lintang, bujur, tahun, bulan, hari, target_alt_isya)

if waktu_utc is None:
    return &quot;Anomali: Matahari tidak pernah mencapai sudut -18° di lokasi ini (Sering terjadi di lintang tinggi Eropa saat musim panas).&quot;

# 3. Konversi format absolut Skyfield ke Datetime Python
dt_utc = waktu_utc.utc_datetime()
```

```
# 4. Melakukan offset ke zona waktu sipil lokal
dt_lokal = dt_utc + timedelta(hours=zona_waktu_offset)

# 5. Menerapkan Ihtiyath (2 Menit)
dt_isya_aman = dt_lokal + timedelta(minutes=2)

print(&quot;\n--- HASIL KOMPUTASI WAKTU SALAT ISYA&#39; ---&quot;)
print(f&quot;Kriteria  Astronomis         :  Senja  Astronomis  (Depresi
{target_alt_isya}°)&quot;)
print(f&quot;Waktu          Hilangnya          Syafak                    :
{dt_lokal.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(f&quot;Waktu        Isya&#39;         (Ihtiyath)                 :
{dt_isya_aman.strftime(&#39;%H:%M:%S&#39;)}&quot;)

return dt_isya_aman
```

Eksekusi Uji Coba: Kota Semarang (-6.9932, 110.4203)

Tanggal Asumsi: 16 Maret 2026 (Zona WIB / UTC+7)

jadwal_isya = komputasi_final_isya(-6.9932, 110.4203, 7, 2026, 3, 16, "Indonesia")

KESIMPULAN ANALISIS LINTANG TINGGI:

Python memberikan mekanisme keamanan (error handling). Jika komputasi dijalankan untuk Oslo, Norwegia pada bulan Juni, matahari mungkin hanya tenggelam hingga -5°, sehingga fungsi akan mengembalikan "Anomali". Di sinilah fiqih intervensi bekerja (seperti Aqrab al-Ayyam / merujuk ke kota terdekat yang memiliki siang-malam normal).

</pre>

Pemodelan algoritma Isya' dalam Python membuktikan ketangguhan bahasa pemrograman ini dalam menangani ambang batas numerik (*threshold detection*). Dengan mengubah pandangan visual langit menjadi data kuantitatif, Python memberikan landasan hukum syar'i yang solid dan universal.

7.5 Syuruk (Terbit Matahari)

(A) Dimensi Fikih: Batas Akhir Subuh dan Waktu Terlarang (Waktu Tahrim)

Dalam kronometri ibadah Islam, *Syuruk* (شُرُوق) atau *Tulu' as-Syams* (terbitnya matahari) memiliki fungsi ganda yang sangat krusial. Pertama, ia bertindak sebagai batas akhir (terminasi) dari waktu salat Subuh. Kedua, detik-detik terjadinya Syuruk merupakan salah satu "waktu terlarang" (*waktu tahrim*) untuk mendirikan salat sunah mutlak, hingga matahari meninggi seukuran satu tombak (waktu Dhuha).

Ketetapan ini bersumber dari hadis sahih riwayat Imam Muslim dari 'Uqbah bin 'Amir *radhiyallahu 'anhu*:

ثَلَاثُ سَاعَاتٍ كَانَ رَسُولُ اللهِ صَلَّى اللهُ عَلَيْهِ وَسَلَّمَ يَنْهَانَا أَنْ نُصَلِّيَ فِيهِنَّ، أَوْ أَنْ نَقْبُرَ فِيهِنَّ مَوْتَانَا: حِينَ تَطْلُعُ الشَّمْسُ بَازِغَةً حَتَّى تَرْتَفِعَ...

"Ada tiga waktu di mana Rasulullah Shallallahu 'alaihi wa sallam melarang kami untuk salat atau menguburkan jenazah kami: ketika matahari terbit dengan jelas hingga ia meninggi..." (HR. Muslim, No. 831).

Hikmah teologis di balik pelarangan ini, sebagaimana dijelaskan dalam riwayat lain, adalah untuk menyelisihi kaum pagan penyembah matahari yang bersujud ketika matahari terbit di antara "dua tanduk setan" (*qarn as-syaithan*). Oleh karena itu, ketepatan jadwal Syuruk tidak hanya mengamankan batas akhir salat Subuh, tetapi juga menjaga kemurnian akidah umat dari tasyabuh (menyerupai) praktik paganisme.

(B) Optik Astronomi: Cermin Kebalikan dari Maghrib

Secara mekanika benda langit dan fisika optik, perhitungan Syuruk adalah bayangan cermin (*mirror image*) dari perhitungan Maghrib. Syuruk didefinisikan secara astronomis ketika garis singgung atas piringan matahari (*upper limb*) pertama kali menyentuh dan menembus garis ufuk timur.

Konstanta yang digunakan sama persis dengan Maghrib, yakni depresi **-0,8333 derajat**. Angka ini merupakan akumulasi dari:

1. Jari-jari (Semi-diameter) matahari: ~16 menit busur.
2. Refraksi atmosfer di ufuk: ~34 menit busur.

Namun, terdapat perbedaan krusial pada efek elevasi (*Dip of Horizon*). Jika pada waktu Maghrib, posisi pengamat yang tinggi (misal di gunung) akan **menunda** waktu terbenamnya matahari; maka pada waktu Syuruk, posisi yang tinggi akan membuat pengamat melihat matahari terbit **lebih awal**. Ufuk timur tampak lebih rendah, sehingga piringan matahari tertangkap oleh mata pengamat lebih cepat dibandingkan mereka yang berada di dataran rendah.

Latihan 1: Inisialisasi Pencarian Sunrise (Syuruk) Menggunakan Skyfield

Langkah pertama adalah memanggil modul *almanac* dari Skyfield untuk memindai kejadian di mana garis geometris matahari menyentuh ufuk ufuk timur dengan memperhitungkan pembiasan udara.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f7f9fc; color: #1a202c; padding: 15px; border-radius: 6px; border: 1px solid #e2e8f0; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: INISIALISASI PEMINDAIAN MATAHARI TERBIT (SYURUK)

Memindai jendela waktu 24 jam untuk melacak persilangan ufuk timur

==========

from skyfield.api import load, wgs84

from skyfield import almanac

def cari_waktu_syuruk_utc(lintang, bujur, tahun, bulan, hari):

planets = load('de421.bsp')

bumi = planets['earth']

ts = load.timescale()

```
lokasi = bumi + wgs84.latlon(lintang, bujur)
t0 = ts.utc(tahun, bulan, hari, 0, 0)
t1 = ts.utc(tahun, bulan, hari + 1, 0, 0)

# Fungsi almanac.sunrise_sunset secara otomatis memproses refraksi
# dan semi-diameter (-0.8333 derajat)
waktu_kejadian, tipe_kejadian = almanac.find_discrete(
    t0,   t1,   almanac.sunrise_sunset(planets,   wgs84.latlon(lintang,
bujur))
)

return waktu_kejadian, tipe_kejadian
```

Catatan Analitis: Proses ini akan mengembalikan array dua dimensi yang berisi waktu UTC dan nilai Boolean. Nilai True adalah Sunrise.

</pre>

Latihan 2: Ekstraksi Peristiwa Sunrise Mutlak

Dari data gabungan *Sunrise* dan *Sunset* yang dihasilkan oleh pemindaian, kita membuat algoritma penyaring untuk hanya mengekstrak kejadian yang bernilai `True` (Matahari Terbit).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: FILTERING DATA MATAHARI TERBIT

Mengekstrak tipe kejadian True (Sunrise) dan membuang False (Sunset)

==========

def ekstraksi_waktu_syuruk(waktu_kejadian, tipe_kejadian):

waktu_syuruk_utc = None

```
for waktu, is_sunrise in zip(waktu_kejadian, tipe_kejadian):
    # is_sunrise bernilai True jika matahari terbit dari bawah ufuk ke
atas ufuk
    if is_sunrise:
        waktu_syuruk_utc = waktu
        break # Loop dihentikan segera setelah Syuruk pertama ditemukan

return waktu_syuruk_utc
```

</pre>

Latihan 3: Kalkulasi Efek Elevasi Terhadap Syuruk

Di Indonesia, kawasan seperti Dataran Tinggi Dieng, Bromo, atau menara tinggi apartemen di Jakarta memiliki jadwal Syuruk yang berbeda secara empiris dengan daratan nol meter. Kita harus memperhitungkan percepatan waktu akibat *Dip of Horizon*.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOREKSI SYURUK BERDASARKAN ELEVASI PENAMPAKAN

Menghitung majunya waktu terbit akibat turunnya ufuk pandang

==========

import math

def target_altitude_syuruk_elevasi(ketinggian_mdpl):

DEPRESI_STANDAR = -0.8333

```
# Koreksi Dip of Horizon (0.0347 * akar elevasi dalam meter)
koreksi_dip = 0.0347 * math.sqrt(ketinggian_mdpl)
```

```
# Untuk Syuruk, ufuk semakin rendah (depresi semakin dalam)
target_koreksi = DEPRESI_STANDAR - koreksi_dip

print(f&quot;--- KOREKSI OPTIK SYURUK UNTUK ELEVASI {ketinggian_mdpl}
MDPL ---&quot;)
print(f&quot;Target Geometris Awal  : {DEPRESI_STANDAR}°&quot;)
print(f&quot;Koreksi Kerendahan Ufuk: -{koreksi_dip:.4f}°&quot;)
print(f&quot;Target Eksekusi Final  : {target_koreksi:.4f}° (Di bawah
ufuk)&quot;)
print(&quot;Analisis: Matahari akan terlihat LEBIH AWAL beberapa
menit.&quot;)

return target_koreksi
```

Uji Coba: Pengamat di Puncak Gunung dengan ketinggian 2000 mdpl

target_gunung = target_altitude_syuruk_elevasi(2000)

</pre>

Latihan 4: Finalisasi Jadwal Syuruk dengan Pengurangan Ihtiyath

Penambahan waktu kehati-hatian (*Ihtiyath*) pada Syuruk memiliki logika yang unik (berbalik arah) dibandingkan waktu salat lainnya. Karena Syuruk adalah tanda *berakhirnya* waktu Subuh (dan masuknya waktu haram untuk salat), maka prinsip kehati-hatian menuntut jadwal Syuruk dipublikasikan **lebih awal** dari kenyataan perhitungannya, atau setidaknya diumumkan secara tepat.

Biasanya, otoritas falak akan mengurangi 1 hingga 2 menit dari waktu komputasi astronomis (mengubah `+ timedelta` menjadi `- timedelta`) untuk memastikan tidak ada umat Islam yang salat Subuh saat piringan matahari secara hakiki sudah mulai merekah di ufuk.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 4: KOMPUTASI FINAL JADWAL SYURUK (LOKAL & IHTIYATH)

==========

from datetime import timedelta

def komputasi_final_syuruk(lintang, bujur, zona_waktu_offset, tahun, bulan, hari):

Eksekusi Pemindaian dan Ekstraksi (Latihan 1 & 2)

waktu_kejadian, tipe_kejadian = cari_waktu_syuruk_utc(lintang, bujur, tahun, bulan, hari)

waktu_syuruk_utc = ekstraksi_waktu_syuruk(waktu_kejadian, tipe_kejadian)

```
if waktu_syuruk_utc is None:
    return &quot;Anomali: Matahari tidak terbit pada lintang ini.&quot;

dt_utc = waktu_syuruk_utc.utc_datetime()
dt_lokal = dt_utc + timedelta(hours=zona_waktu_offset)

# Penerapan Ihtiyath Syuruk (Dikurangi 2 menit untuk keamanan batas
Subuh)
# Ini memastikan peringatan &quot;waktu habis&quot; berbunyi sebelum
ufuk benar-benar pecah.
dt_syuruk_aman = dt_lokal - timedelta(minutes=2)

print(&quot;\n--- HASIL KOMPUTASI WAKTU SYURUK (TULU&#39; AS-SYAMS) ---
&quot;)
print(f&quot;Lokasi Koordinat        : Lintang {lintang}, Bujur
{bujur}&quot;)
print(f&quot;Sunrise Geometris murni :
{dt_lokal.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(f&quot;Batas Aman Akhir Subuh  :
{dt_syuruk_aman.strftime(&#39;%H:%M:%S&#39;)} (Ihtiyath -2m)&quot;)

return dt_syuruk_aman
```

Eksekusi Uji Coba: Kota Yogyakarta (-7.7956, 110.3695)

Asumsi Tanggal: 10 September 2026 (WIB)

jadwal_syuruk = komputasi_final_syuruk(-7.7956, 110.3695, 7, 2026, 9, 10)

</pre>

Pemahaman yang mendalam terhadap algoritma Syuruk ini menunjukkan bagaimana Python dapat disesuaikan dengan metodologi ushul fikih. Logika pengurangan menit (`- timedelta`) untuk prinsip *Sadd adz-Dzari'ah* (menutup jalan menuju keharaman/pelanggaran batas waktu) dapat diimplementasikan dalam hitungan detik tanpa mengubah struktur dasar mesin astronomi Skyfield.

7.6 Subuh

(A) Dimensi Fikih: Fajar Kadzib, Fajar Shadiq, dan "Benang Putih"

Salat Subuh menempati posisi yang sangat krusial karena ia bukan sekadar penanda dimulainya kewajiban ibadah harian pertama, melainkan juga titik absolut (batas waktu/ *deadline*) dimulainya ibadah puasa (Imsakiyah). Kesalahan dalam menghitung masuknya waktu Subuh berimplikasi langsung pada batalnya puasa jutaan umat Islam jika mereka masih makan atau minum setelah fajar terbit.

Landasan penetapan waktu Subuh disandarkan pada fenomena optik fajar di ufuk timur, sebagaimana difirmankan oleh Allah *Subhanahu wa Ta'ala*:

وَكُلُوا وَاشْرَبُوا حَتَّىٰ يَتَبَيَّنَ لَكُمُ الْخَيْطُ الْأَبْيَضُ مِنَ الْخَيْطِ الْأَسْوَدِ مِنَ الْفَجْرِ ۖ ثُمَّ أَتِمُّوا الصِّيَامَ إِلَى اللَّيْلِ

"...Dan makan minumlah hingga terang bagimu benang putih dari benang hitam, yaitu fajar. Kemudian sempurnakanlah puasa itu sampai (datang) malam..." (QS. Al-Baqarah [2]: 187).

Dalam tradisi observasi falak klasik (Rukyatul Fajar), para ulama membedakan fajar menjadi dua entitas optik yang berbeda secara fisis:

1. **Fajar Kadzib (Fajar Palsu / Zodiacal Light):** Merupakan cahaya tipis yang menjulang vertikal ke atas seperti ekor serigala (*dhanab as-sirhan*). Cahaya ini diakibatkan oleh pantulan debu interplanet di tata surya, bukan oleh hamburan cahaya matahari di atmosfer bumi. Fajar Kadzib muncul lebih awal, namun setelah itu langit akan kembali gelap. Waktu Subuh **belum** masuk pada fase ini.
2. **Fajar Shadiq (Fajar Nyata / True Dawn):** Merupakan cahaya yang menyebar horizontal (*mustatir*) dan merata melintasi ufuk timur. Cahaya ini adalah hamburan nyata dari piringan matahari yang mulai mendekati ufuk bumi dari bawah. Munculnya fajar inilah yang menandai masuknya waktu salat Subuh secara sah.

(B) Translasi Astronomis: Kedalaman Sudut Depresi Pagi

Secara mekanika optik astronomi, Subuh adalah kebalikan cermin (*mirror image*) dari Isya'. Jika Isya' adalah berakhirnya Senja Astronomis (*Astronomical Twilight*) di ufuk barat, maka Subuh adalah **dimulainya Fajar Astronomis** di ufuk timur. Pada fase ini, foton cahaya matahari mulai menembus lapisan atas atmosfer bumi dan dipantulkan kembali ke permukaan, mengalahkan kegelapan malam pekat.

Tantangan terbesar dalam hisab awal waktu Subuh adalah menentukan pada sudut kedalaman (depresi) berapakah Fajar Shadiq itu mulai terdeteksi secara empiris. Hal ini melahirkan keragaman kriteria (ikhtilaf) di berbagai institusi global:

- **Kementerian Agama Republik Indonesia (Kemenag):** Menetapkan sudut depresi matahari pada **-20°** (atau 20 derajat di bawah ufuk timur). Keputusan ini diambil dengan prinsip *Ihtiyath* (kehati-hatian) yang sangat tinggi untuk mengamankan waktu dimulainya puasa.
- **Umm al-Qura (Arab Saudi):** Menggunakan sudut **-18.5°** (dan -19° khusus untuk bulan Ramadan).
- **Muslim World League (MWL):** Menggunakan sudut **-18°**.
- **Islamic Society of North America (ISNA):** Menggunakan sudut **-15°**.

Dalam konstruksi algoritma Python pada buku ini, kita akan merancang sistem parametrik yang fleksibel, memampukan pengguna untuk memilih kriteria sudut depresi sesuai yurisdiksi otoritas negara masing-masing.

Latihan 1: Penetapan Parameter Depresi Fajar Astronomis

Sama halnya dengan algoritma Isya', kita mendeklarasikan parameter (konstanta) ketinggian matahari di bawah ufuk matematis. Karena matahari berada di bawah horizon, nilainya mutlak negatif.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f7f9fc; color: #1a202c; padding: 15px; border-radius: 6px; border: 1px solid #e2e8f0; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: DEFINISI PARAMETER SUDUT DEPRESI SUBUH (FAJAR SHADIQ)

Mengonversi kriteria fikih 'benang putih' menjadi variabel astronomis

==========

def tetapkan_kriteria_subuh(otoritas="Kemenag_RI"):

kriteria_depresi = {

"Kemenag_RI": -20.0,

"Umm_Al_Qura": -18.5,

"MWL": -18.0,

"ISNA": -15.0,

"Mesir": -19.5

}

```
# Menarik nilai dari dictionary, default -20.0 (Indonesia)
target_altitude = kriteria_depresi.get(otoritas, -20.0)

print(&amp;quot;--- PARAMETER KRITERIA SALAT SUBUH ---&amp;quot;)
print(f&amp;quot;Otoritas Referensi    : {otoritas}&amp;quot;)
print(f&amp;quot;Target Depresi Fajar   : {target_altitude:.1f} derajat di bawah ufuk timur&amp;quot;)

return target_altitude
```

Uji Coba Latihan 1

alt_subuh_standar = tetapkan_kriteria_subuh("Kemenag_RI")

</pre>

Latihan 2: Pemindaian Ephemeris untuk Mencapai Sudut Depresi Subuh

Algoritma Skyfield ditugaskan untuk memindai pergerakan matahari pada fase sepertiga malam terakhir (dini hari). Kondisi filter (*if-statement*) yang krusial di sini adalah memastikan

azimuth matahari berada di kuadran **Timur** (yakni antara 0° hingga 180°) dan matahari sedang bergerak **naik** (*ascending*) mendekati ufuk.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: PEMINDAIAN WAKTU SUBUH MENGGUNAKAN SKYFIELD

Mencari momen naiknya matahari menyentuh garis depresi -20 derajat

==========

from skyfield.api import load, wgs84

def cari_waktu_subuh_utc(lintang, bujur, tahun, bulan, hari, target_depresi):

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

ts = load.timescale()

lokasi = bumi + wgs84.latlon(lintang, bujur)

```
waktu_subuh_utc = None
toleransi_derajat = 0.05

# Memindai waktu dini hari hingga pagi (Rentang 19.00 UTC hari sebelumnya
hingga 24.00 UTC)
# Catatan: Asumsi untuk WIB (UTC+7), jam 03.00 pagi = 20.00 UTC hari
sebelumnya.
# Untuk menghindari kompleksitas pergantian hari dalam simulasi, kita
gunakan
# rentang jam -4 (20.00) hingga 0 (00.00). Python Datetime dapat menangani
ini.

for jam in range(-5, 1): # Mewakili pukul 19:00 UTC (H-1) hingga 01:00
UTC (H)
    for menit in range(0, 60):
        # Pengaturan hari sebelumnya jika jam negatif
        jam_aktual = jam if jam &gt;= 0 else 24 + jam
        hari_aktual = hari if jam &gt;= 0 else hari - 1

        t = ts.utc(tahun, bulan, hari_aktual, jam_aktual, menit)
        astrometrik = lokasi.at(t).observe(matahari)
        alt, az, distance = astrometrik.apparent().altaz()

        # Filter ganda untuk Subuh:
        # 1. Matahari di ufuk timur (Azimuth &lt; 180)
        # 2. Cek kesesuaian dengan target depresi
        if az.degrees &lt; 180:
```

```
            selisih = abs(alt.degrees - target_depresi)
            if selisih &lt;= toleransi_derajat:
                waktu_subuh_utc = t
                break # Hentikan pencarian saat Fajar Shadiq terdeteksi
    if waktu_subuh_utc:
        break

return waktu_subuh_utc
```

</pre>

Latihan 3: Integrasi Akhir dan Penerapan Ihtiyath Fikih

Dalam penyusunan jadwal salat Subuh, *Ihtiyath* (tambahan waktu kehati-hatian) bernilai sangat krusial. Biasanya ditambahkan 2 menit dari hasil perhitungan eksak astronomi. Hal ini memastikan bahwa fajar Shadiq benar-benar telah menyebar di ufuk, sehingga orang yang melaksanakan salat Subuh di awal waktu dan orang yang mulai berpuasa (berhenti sahur) berada dalam rentang waktu yang pasti halal dan sah.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOMPUTASI FINAL JADWAL SALAT SUBUH (LOKAL & IHTIYATH)

==========

from datetime import timedelta

def komputasi_final_subuh(lintang, bujur, zona_waktu_offset, tahun, bulan, hari, kriteria="Kemenag_RI"):

1. Mendapatkan target sudut depresi

target_alt_subuh = tetapkan_kriteria_subuh(kriteria)

```
# 2. Eksekusi pencarian ephemeris
waktu_utc = cari_waktu_subuh_utc(lintang, bujur, tahun, bulan, hari, target_alt_subuh)

if waktu_utc is None:
    return &quot;Anomali: Fajar tidak terdeteksi pada rentang waktu ini.&quot;

# 3. Konversi format ke Datetime Python
dt_utc = waktu_utc.utc_datetime()

# 4. Offset ke zona waktu lokal
dt_lokal = dt_utc + timedelta(hours=zona_waktu_offset)
```

```
# 5. Penambahan Ihtiyath (2 Menit pasca-Fajar Astronomis)
# Sebagai pengaman batas waktu larangan makan (Imsak seringkali -10 menit
dari Subuh ini)
dt_subuh_aman = dt_lokal + timedelta(minutes=2)

print(&quot;\n--- HASIL KOMPUTASI WAKTU SALAT SUBUH ---&quot;)
print(f&quot;Kriteria  Astronomis          :  Fajar  Shadiq  (Depresi
{target_alt_subuh}°)&quot;)
print(f&quot;Kemunculan         Fajar         Murni                  :
{dt_lokal.strftime(&#39;%H:%M:%S&#39;)}&quot;)
print(f&quot;Waktu          Subuh          (Ihtiyath)               :
{dt_subuh_aman.strftime(&#39;%H:%M:%S&#39;)}&quot;)

return dt_subuh_aman
```

Eksekusi Uji Coba: Kota Palembang (-2.9909, 104.7566)

Asumsi Tanggal: 12 November 2026 (WIB)

jadwal_subuh = komputasi_final_subuh(-2.9909, 104.7566, 7, 2026, 11, 12, "Kemenag_RI")

KESIMPULAN ANALITIS BAB WAKTU SALAT:

Dengan selesainya komputasi algoritma untuk Zuhur, Asar, Maghrib, Isya', Syuruk, dan Subuh,

kita telah mengonstruksi sebuah arsitektur "Mesin Waktu Fikih" yang komprehensif.

Python membuktikan dirinya bukan sekadar alat penghitung, melainkan 'mufti digital'

yang menerjemahkan kehendak optik alam semesta menjadi panduan kepatuhan syariat

dengan presisi tingkat observatorium.

</pre>

Daftar Pustaka (Bab 7)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). Astronomical algorithms for Qibla direction and daily prayer times estimation using Python. *Journal of Islamic Science and Technology*, 14(2), 101-115.
2. Anwar, S. (2018). *Ilmu Falak: Hisab Waktu Salat, Arah Kiblat, dan Awal Bulan*. UII Press. (Rujukan parameter Ihtiyath dan ketinggian tempat).

3. Ar-Razi, F. D. (1981). *Mafatih al-Ghaib (Tafsir al-Kabir)*. Dar al-Fikr. (Tafsir teologis QS. Al-Baqarah: 187 mengenai fajar shadiq dan kadzib).
4. Asrin, A., Jamil, A., & Nawawi, M. S. A. M. (2018). *Aplikasi Hisab Kiblat Berbasis Geodesi Sferis dan Elipsoida*. Pustaka Falak Indonesia.
5. Faid, M. S., et al. (2019). The computational algorithm of prayer times estimation for high latitude regions. *International Journal of Advanced Science and Technology*, 28(16).
6. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
7. Ilyas, M. (1984). *A modern guide to astronomical calculations of Islamic calendar, times and qibla*. (Rujukan fundamental perhitungan Subuh dan Isya).
8. Junaidi, J. (2022). *Dinamika Kriteria Imkanur Rukyat dan Implikasinya Terhadap Penyatuan Kalender Hijriah di Indonesia*. Pustaka Pelajar. (Memuat sejarah adopsi sudut depresi -20 derajat Kemenag RI).
9. King, D. A. (1993). *Astronomy in the service of Islam*. Routledge.
10. Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell. (Rumus refraksi, semi-diameter, dan solar transit dasar).
11. Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites generator*. Astrophysics Source Code Library.

BAB 8 KOMPUTASI DATA OBSERVASI RU'YATUL HILAL

8.1 Pendahuluan: Signifikansi Teologis dan Epistemologi Observasi Bulan

Kalender Hijriah (takwim kamariah) bukan sekadar sistem penanggalan sipil, melainkan sebuah infrastruktur temporal yang menopang validitas ibadah-ibadah fundamental dalam Islam. Puasa Ramadan, perayaan Idulfitri (Syawal), hingga pelaksanaan wukuf di Arafah (Zulhijah) sangat bergantung pada presisi penentuan awal bulan. Secara ontologis, Allah *Subhanahu wa Ta'ala* telah menetapkan pergerakan bulan sebagai instrumen universal pengukur waktu bagi umat manusia:

يَسْأَلُونَكَ عَنِ الْأَهِلَّةِ ۖ قُلْ هِيَ مَوَاقِيتُ لِلنَّاسِ وَالْحَجِّ

"Mereka bertanya kepadamu tentang bulan sabit. Katakanlah: 'Bulan sabit itu adalah tanda-tanda waktu bagi manusia dan (bagi ibadat) haji...'" (QS. Al-Baqarah [2]: 189).

Dalam praksis fikih, Rasulullah *Shallallahu 'alaihi wa sallam* memberikan pedoman metodologis melalui sabdanya:

صُومُوا لِرُؤْيَتِهِ وَأَفْطِرُوا لِرُؤْيَتِهِ، فَإِنْ غُمَّ عَلَيْكُمْ فَأَكْمِلُوا عِدَّةَ شَعْبَانَ ثَلَاثِينَ

"Berpuasalah kalian karena melihatnya (hilal) dan berbukalah (berhari raya) karena melihatnya. Jika ia tertutup awan dari pandangan kalian, maka sempurnakanlah hitungan bulan Syakban menjadi tiga puluh hari." (HR. Bukhari, No. 1909).

Hadis ini meletakkan fondasi **Ru'yatul Hilal** (observasi bulan sabit) sebagai syarat syar'i. Namun, di era sains modern, observasi ini tidak lagi bersifat kebetulan (*accidental*), melainkan prediktif dan komputasional. Pemodelan data astronomi (*hisab rukyat*) bertugas menjawab pertanyaan kritis: *Apakah pada saat matahari terbenam di tanggal 29 bulan berjalan, hilal secara fisis dan optis mungkin untuk dilihat (Imkanur Rukyat)?*

Untuk menjawab pertanyaan ini, komputasi astronomi menggunakan Python harus mengevaluasi matriks variabel yang sangat kompleks:

1. **Fase Konjungsi (Ijtimak Geosentris):** Detik eksak di mana bujur ekliptika bulan dan matahari bernilai sama persis relatif terhadap pusat bumi. Ini adalah momen "kelahiran" bulan baru secara astronomis.
2. **Ketinggian Hilal (Altitude/Irtifa'):** Jarak sudut bulan dari ufuk lokal sesaat setelah piringan atas matahari terbenam.
3. **Elongasi (Jarak Sudut):** Rentang busur yang memisahkan titik pusat matahari dan bulan. Semakin besar elongasinya, semakin tebal sabit bulan (*illuminated fraction*) yang memantulkan cahaya matahari ke bumi. (Batas Danjon secara teoretis mensyaratkan elongasi minimal ~7 derajat agar sabit terbentuk).
4. **Beda Azimuth (DAZ) dan Umur Bulan:** Posisi horizontal bulan relatif terhadap matahari, serta durasi waktu yang telah berlalu sejak fase konjungsi hingga waktu observasi (sunset).

Bab ini akan mengupas tuntas dan mengeksekusi perhitungan tersebut melalui serangkaian algoritma Python tingkat observatorium, mentransformasikan hisab klasik ke dalam paradigma *data science*.

8.2 Latihan 1: Komputasi Fase Konjungsi (Ijtimak) dan Penentuan Waktu Observasi

Langkah paling elementer dalam penentuan awal bulan adalah mencari titik nol, yakni fase konjungsi geosentris (Ijtimak). Rukyatul hilal hanya sah dilaksanakan pada saat *sunset* (matahari terbenam) **setelah** terjadinya Ijtimak. Jika matahari terbenam mendahului waktu Ijtimak, maka bulan yang ada di langit masih berstatus sebagai bulan sabit tua (*waning crescent*) dari bulan sebelumnya, dan mustahil disebut hilal.

Pustaka `Skyfield` dalam Python memfasilitasi pencarian fase bulan secara instan tanpa perlu menyusun deret Fourier mekanika bulan yang panjang. Melalui fungsi `almanac.moon_phases`, kita dapat memindai rentang tanggal tertentu untuk mendeteksi kapan tepatnya fase *New Moon* (Bulan Baru / Fase 0) terjadi.

Berikut adalah *source code* Latihan 1 untuk mencari waktu konjungsi absolut dan memadankannya dengan waktu matahari terbenam di lokasi pengamat pada hari yang sama.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: MENCARI WAKTU KONJUNGSI (IJTIMAK) DAN SUNSET OBSERVASI

Menetapkan titik nol umur hilal secara geosentris

==========

from skyfield.api import load, wgs84

from skyfield import almanac

from datetime import timedelta

def komputasi_ijtimak_dan_sunset(lintang, bujur, tahun, bulan, zona_waktu):

planets = load('de421.bsp')

bumi = planets['earth']

ts = load.timescale()

lokasi_pengamat = bumi + wgs84.latlon(lintang, bujur)

```
# 1. Menentukan rentang waktu pemindaian (1 bulan penuh)
t0 = ts.utc(tahun, bulan, 1)

# Menangani pergantian tahun untuk batas akhir pencarian
if bulan == 12:
    t1 = ts.utc(tahun + 1, 1, 1)
else:
    t1 = ts.utc(tahun, bulan + 1, 1)

print(f&quot;--- ANALISIS FASE KONJUNGSI (IJTIMAK) BULAN {bulan}/{tahun}
---&quot;)

# 2. Mencari Fase Bulan (0: New Moon, 1: First Quarter, 2: Full Moon, 3:
Last Quarter)
waktu_fase,     nilai_fase     =     almanac.find_discrete(t0,     t1,
almanac.moon_phases(planets))

waktu_ijtimak_utc = None

# Mengekstrak hanya fase New Moon (Nilai Fase == 0)
for waktu, fase in zip(waktu_fase, nilai_fase):
    if fase == 0:
        waktu_ijtimak_utc = waktu
        break # Hentikan setelah menemukan Ijtimak pertama di bulan
tersebut

if waktu_ijtimak_utc is None:
    return &quot;Ijtimak tidak ditemukan pada rentang bulan
tersebut.&quot;

dt_ijtimak_lokal       =       waktu_ijtimak_utc.utc_datetime()       +
timedelta(hours=zona_waktu)
print(f&quot;Waktu Ijtimak Absolut : {dt_ijtimak_lokal.strftime(&#39;%Y-
%m-%d %H:%M:%S&#39;)} (Waktu Lokal)&quot;)

# 3. Mencari Waktu Matahari Terbenam (Sunset) pada hari terjadinya Ijtimak
# Hari rukyat (observasi) adalah hari di mana Ijtimak terjadi, atau hari
sesudahnya.
# Kita pindai sunset pada tanggal Ijtimak tersebut.
t_awal_hari  =  ts.utc(dt_ijtimak_lokal.year,  dt_ijtimak_lokal.month,
dt_ijtimak_lokal.day, 0, 0)
t_akhir_hari  =  ts.utc(dt_ijtimak_lokal.year,  dt_ijtimak_lokal.month,
dt_ijtimak_lokal.day + 1, 0, 0)

waktu_sunset, tipe_sunset = almanac.find_discrete(
    t_awal_hari,     t_akhir_hari,     almanac.sunrise_sunset(planets,
wgs84.latlon(lintang, bujur))
)

dt_sunset_lokal = None
waktu_sunset_skyfield = None

for w_sun, is_sunrise in zip(waktu_sunset, tipe_sunset):
    if not is_sunrise: # False berarti Sunset
        waktu_sunset_skyfield = w_sun
        dt_sunset_lokal        =        w_sun.utc_datetime()        +
timedelta(hours=zona_waktu)
        break
```

```
print(f&amp;quot;Waktu Sunset Lokal     : {dt_sunset_lokal.strftime(&amp;#39;%Y-%m-%d %H:%M:%S&amp;#39;)}&amp;quot;)

# 4. Validasi Dasar (Apakah Ijtimak terjadi SEBELUM Sunset?)
if dt_ijtimak_lokal &amp;lt; dt_sunset_lokal:
    umur_hilal = dt_sunset_lokal - dt_ijtimak_lokal
    print(f&amp;quot;Status Observasi       : VALID (Ijtimak terjadi sebelum Sunset)&amp;quot;)
    print(f&amp;quot;Umur Hilal saat Sunset: {umur_hilal.seconds // 3600} jam {((umur_hilal.seconds // 60) % 60)} menit&amp;quot;)
else:
    print(f&amp;quot;Status Observasi       : TIDAK VALID (Ijtimak terjadi setelah Sunset)&amp;quot;)
    print(&amp;quot;Kesimpulan           : Hilal mustahil terlihat hari ini. Observasi (Rukyat) dilakukan keesokan harinya.&amp;quot;)

return waktu_sunset_skyfield
```

Eksekusi Uji Coba: Pemantauan Awal Ramadan di Observatorium Bosscha, Lembang (-6.82, 107.61)

Asumsi pencarian pada bulan Februari 2026 (WIB)

sunset_observasi = komputasi_ijtimak_dan_sunset(-6.82, 107.61, 2026, 2, 7)

</pre>

Analisis Latihan 1: Algoritma ini mewakili proses "Filter Hari Rukyat". Di lingkungan kementerian agama, sidang isbat selalu didahului oleh pemaparan data kapan konjungsi (Ijtimak) terjadi. Melalui kode Python di atas, kita mengalkulasi selisih absolut antara detik Ijtimak dan detik terbenamnya matahari untuk menghasilkan parameter **Umur Hilal**. Berdasarkan beberapa kriteria klasik internasional, umur hilal minimum yang memungkinkan bulan dapat dirukyat (dilihat) secara fisis adalah sekitar 8 hingga 15 jam. Jika hasil komputasi menunjukkan umur hilal hanya 2 jam, maka secara astronomis, bulan terlalu redup dan terlalu tipis untuk mampu menembus hamburan cahaya ufuk senja, sehingga klaim rukyat apa pun pada hari tersebut biasanya akan ditolak secara keilmuan (*haq*).

Keluaran objek `waktu_sunset_skyfield` dari fungsi ini akan dikunci (*lock*) dan dibawa ke Latihan 2 untuk mengukur parameter optik hilal pada detik yang sangat spesifik tersebut.

8.3 Latihan 2: Komputasi Ketinggian dan Elongasi Bulan saat Waktu Sunset

Parameter paling fundamental dalam observasi hilal adalah **Ketinggian (Altitude)** dan **Elongasi (Jarak Sudut)**.

1. **Ketinggian Hilal:** Diukur dari garis ufuk (horizon 0 derajat) tegak lurus ke arah pusat piringan bulan. Hilal yang berada di bawah ufuk (Altitude negatif) mustahil untuk dilihat karena telah terbenam mendahului matahari. Berdasarkan kriteria MABIMS (Menteri Agama Brunei, Indonesia, Malaysia, dan Singapura) yang baru, ketinggian minimal hilal agar mungkin dirukyat adalah **3 derajat**.

2. **Elongasi:** Adalah jarak busur spasial antara pusat piringan matahari dan pusat piringan bulan. Parameter ini sangat vital karena ia berbanding lurus dengan ketebalan sabit bulan yang memantulkan cahaya matahari ke bumi (fraksi iluminasi). Semakin kecil elongasinya, semakin tipis sabitnya, dan semakin sulit mengalahkan hamburan cahaya senja (*twilight*). Kriteria MABIMS menetapkan elongasi minimal sebesar **6,4 derajat**.

Pada Latihan 2 ini, kita menyerap objek waktu *sunset* yang telah didapatkan dari Latihan 1 (misalnya direpresentasikan dalam variabel `waktu_sunset_skyfield`), dan menginstruksikan Python untuk "memotret" posisi matahari dan bulan pada detik eksak tersebut.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f7f9fc; color: #1a202c; padding: 15px; border-radius: 6px; border: 1px solid #e2e8f0; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: KOMPUTASI ALTITUDE DAN ELONGASI HILAL SAAT SUNSET

Menggunakan model refraksi optik untuk ufuk dan jarak sudut sferis

==========

from skyfield.api import load, wgs84

def komputasi_parameter_hilal(lintang, bujur, waktu_sunset_skyfield):

Verifikasi validitas waktu sunset

if waktu_sunset_skyfield is None:

return "Data waktu sunset tidak valid."

```
planets = load(&#39;de421.bsp&#39;)
bumi,       matahari,       bulan       =       planets[&#39;earth&#39;],
planets[&#39;sun&#39;], planets[&#39;moon&#39;]
lokasi_pengamat = bumi + wgs84.latlon(lintang, bujur)

# 1. Observasi Posisi Bulan saat Matahari Terbenam
astrometrik_bulan                                                    =
lokasi_pengamat.at(waktu_sunset_skyfield).observe(bulan)
apparent_bulan = astrometrik_bulan.apparent()

# Ekstraksi Altitude dan Azimuth Bulan dengan Koreksi Refraksi Atmosfer
# Parameter suhu 10 derajat Celcius dan tekanan 1010 mbar adalah standar
astronomi
alt_bulan,              az_bulan,              jarak_bulan              =
apparent_bulan.altaz(temperature_C=10, pressure_mbar=1010)

# 2. Observasi Posisi Matahari saat Terbenam
astrometrik_matahari                                                 =
lokasi_pengamat.at(waktu_sunset_skyfield).observe(matahari)
apparent_matahari = astrometrik_matahari.apparent()
```

```
alt_matahari,          az_matahari,          jarak_matahari          =
apparent_matahari.altaz(temperature_C=10, pressure_mbar=1010)

# 3. Komputasi Jarak Sudut (Elongasi) antara Pusat Bulan dan Matahari
elongasi = astrometrik_matahari.separation_from(astrometrik_bulan)

# Menampilkan Laporan Astronomis
print(&quot;--- PARAMETER OPTIK HILAL SAAT SUNSET ---&quot;)
print(f&quot;Ketinggian Matahari : {alt_matahari.degrees:.4f}° (Berada
di ufuk optik)&quot;)
print(f&quot;Ketinggian Hilal      : {alt_bulan.degrees:.4f}° di atas
ufuk&quot;)
print(f&quot;Elongasi (Jarak)    : {elongasi.degrees:.4f}°&quot;)

return    alt_bulan.degrees,    az_bulan.degrees,    az_matahari.degrees,
elongasi.degrees
```

Catatan: Fungsi ini membutuhkan input waktu_sunset_skyfield yang dihasilkan

dari fungsi Latihan 1 pada sub-bab sebelumnya.

</pre>

Analisis Latihan 2: Berbeda dengan rumus matematika manual yang sering mengabaikan paralaks dan refraksi, penggunaan metode `.apparent().altaz(temperature_C=10, pressure_mbar=1010)` dalam Skyfield secara dinamis membelokkan "sinar" dari bulan ke mata pengamat maya (komputer) sesuai ketebalan atmosfer. Hasil yang didapatkan adalah ketinggian *Mar'i* (yang benar-benar terlihat oleh mata), bukan sekadar ketinggian geometris murni.

8.4 Latihan 3: Penentuan Beda Azimuth (DAZ) dan Fraksi Iluminasi

Meskipun ketinggian dan elongasi sudah memenuhi syarat, ada dua parameter tambahan yang sering dianalisis oleh perukyat lapangan (*observer*):

1. **Beda Azimuth (DAZ - Difference in Azimuth):** Adalah jarak horizontal (menyamping) antara titik terbenamnya matahari dengan posisi hilal. Jika DAZ bernilai positif, hilal berada di sebelah utara matahari; jika negatif, ia berada di sebelah selatan. Nilai ini sangat penting bagi petugas rukyat untuk mengarahkan lensa teleskop (fokus pencarian) ke titik koordinat yang tepat di ufuk ufuk barat.
2. **Fraksi Iluminasi:** Adalah persentase luasan piringan bulan yang bercahaya dibandingkan dengan seluruh piringannya. Untuk hilal awal bulan, nilainya sangat kecil (biasanya antara 0,5% hingga 1,5%).

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: KOMPUTASI BEDA AZIMUTH (DAZ) DAN FRAKSI ILUMINASI

==========

import math

def komputasi_daz_dan_iluminasi(az_bulan_deg, az_matahari_deg, elongasi_deg):

1. Menghitung Beda Azimuth (Difference in Azimuth)

Azimuth diukur dari 0 (Utara) hingga 360 derajat.

daz = az_bulan_deg - az_matahari_deg

```
# Normalisasi jarak terpendek (Jika selisih lebih dari 180 derajat)
if daz &gt; 180:
    daz -= 360
elif daz &lt; -180:
    daz += 360

arah_relatif = &quot;Utara&quot; if daz &gt; 0 else &quot;Selatan&quot;

# 2. Menghitung Fraksi Iluminasi (Crescent Thickness)
# Rumus pendekatan iluminasi (f) berdasarkan elongasi (E):
# f = (1 - cos(E)) / 2
elongasi_rad = math.radians(elongasi_deg)
fraksi_iluminasi = (1 - math.cos(elongasi_rad)) / 2
persentase_cahaya = fraksi_iluminasi * 100

print(&quot;\n--- ANALISIS SPASIAL DAN ILUMINASI ---&quot;)
print(f&quot;Azimuth Matahari    : {az_matahari_deg:.2f}°&quot;)
print(f&quot;Azimuth Hilal       : {az_bulan_deg:.2f}°&quot;)
print(f&quot;Beda Azimuth (DAZ)  : {abs(daz):.2f}° (Terletak di sebelah
{arah_relatif} Matahari)&quot;)
print(f&quot;Fraksi Iluminasi    : {persentase_cahaya:.3f}% dari total
piringan bulan&quot;)

return daz, persentase_cahaya
```

Simulasi Uji Coba Latihan 3 (Menggunakan data dummy Latihan 2)

daz_hilal, iluminasi_hilal = komputasi_daz_dan_iluminasi(az_bulan_deg=275.5, az_matahari_deg=272.0, elongasi_deg=7.2)

</pre>

Analisis Latihan 3: Dengan nilai iluminasi yang sangat kecil (seringkali di bawah 1%), kita dapat membuktikan secara saintifik betapa krusialnya kondisi atmosfer (bebas dari awan, polusi udara, dan polusi cahaya). Kode ini mentransformasi teleskop dari sekadar alat peraba

(*trial and error*) menjadi instrumen *point-and-shoot* yang presisi berkat panduan Beda Azimuth.

8.5 Latihan 4: Evaluasi Terhadap Kriteria Imkanur Rukyat (MABIMS)

Latihan terakhir dalam Bab 8 ini mengintegrasikan seluruh parameter fisis yang telah dikalkulasi (Umur Hilal, Ketinggian, Elongasi, dan DAZ) ke dalam sebuah **Sistem Pengambil Keputusan (Decision Support System)**. Sistem ini akan secara otomatis membandingkan angka-angka astronomis dengan hukum positif (syariat negara) untuk menyimpulkan apakah keesokan harinya sah ditetapkan sebagai tanggal satu atau tidak.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f0f4f8; color: #102a43; padding: 15px; border-radius: 8px; border: 1px solid #c8d8e4; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 4: SISTEM EVALUASI KRITERIA IMKANUR RUKYAT (MABIMS BARU)

Validasi silang parameter optik dengan ketetapan Fikih Kementerian

==========

def evaluasi_kriteria_mabims(ketinggian_hilal, elongasi_hilal):

Konstanta Kriteria MABIMS (Disepakati Tahun 2021)

SYARAT_KETINGGIAN = 3.0 # Derajat

SYARAT_ELONGASI = 6.4 # Derajat

```
print(&amp;quot;\n============================================================&amp;quot;)
print(&amp;quot;EVALUASI FINAL PENENTUAN AWAL BULAN HIJRIAH (MABIMS)&amp;quot;)
print(&amp;quot;============================================================&amp;quot;)

# Variabel Pemeriksaan (Checklist)
cek_tinggi = ketinggian_hilal &amp;gt;= SYARAT_KETINGGIAN
cek_elongasi = elongasi_hilal &amp;gt;= SYARAT_ELONGASI

print(f&amp;quot;Data Ketinggian: {ketinggian_hilal:.2f}° | Syarat (&amp;gt;= 3.0°)  -&amp;gt; {&amp;#39;LULUS&amp;#39; if cek_tinggi else &amp;#39;GAGAL&amp;#39;}&amp;quot;)
print(f&amp;quot;Data Elongasi  : {elongasi_hilal:.2f}° | Syarat (&amp;gt;= 6.4°) -&amp;gt; {&amp;#39;LULUS&amp;#39; if cek_elongasi else &amp;#39;GAGAL&amp;#39;}&amp;quot;)
print(&amp;quot;-&amp;quot; * 55)

# Kesimpulan Logika Konjungtif (Keduanya harus terpenuhi)
if cek_tinggi and cek_elongasi:
    print(&amp;quot;KESIMPULAN SYAR&amp;#39;I: [ MEMENUHI KRITERIA IMKANUR RUKYAT ]&amp;quot;)
```

```
    print(&quot;Tindakan: Secara hisab, hilal mungkin untuk
dirukyat.&quot;)
    print(&quot;Keesokan harinya SAH ditetapkan sebagai tanggal 1 bulan
baru.&quot;)
else:
    print(&quot;KESIMPULAN SYAR&#39;I: [ TIDAK MEMENUHI KRITERIA ]&quot;)
    print(&quot;Tindakan: Posisi bulan terlalu rendah / sabit terlalu
tipis.&quot;)
    print(&quot;Bulan berjalan digenapkan menjadi 30 hari
(Istikmal).&quot;)
print(&quot;==========================================================\n&q
uot;)
```

Eksekusi Simulasi Keputusan

Skenario: Bulan berada di ketinggian 2.5 derajat dan elongasi 5.8 derajat

evaluasi_kriteria_mabims(2.5, 5.8)

</pre>

Kesimpulan Bab 8:

Penggabungan antara Latihan 1 (Ephemeris Waktu), Latihan 2 (Geometri Vertikal), Latihan 3 (Geometri Horizontal), dan Latihan 4 (Evaluasi Fikih) memformulasikan sebuah perangkat lunak hisab rukyat yang holistik. Ekosistem Python tidak sekadar menampilkan angka, namun bertransformasi menjadi sebuah sistem *Expert* yang memvalidasi keabsahan transisi waktu religius. Hal ini melenyapkan subjektivitas observasi dan membentengi keputusan sidang isbat dengan benteng argumentasi eksakta yang tidak dapat dipatahkan.

Daftar Pustaka (Bab 8)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). *Astronomical algorithms for Qibla direction and daily prayer times estimation using Python*. Journal of Islamic Science and Technology, 14(2).
2. Ar-Razi, F. D. (1981). *Mafatih al-Ghaib (Tafsir al-Kabir)*. Dar al-Fikr. (Rujukan korelasi tafsir QS Al-Baqarah: 189 dengan observasi astronomi).
3. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., Ahmad, N., & Ali @ Mat Zin, A. (2022). Islamic historical review on the middle age lunar crescent visibility criterion. *Journal of Al-Tamaddun*, 17(1).
4. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2023). Python applications in astronomical computations: A comparative analysis of Skyfield and PyEphem for Hijri calendar calculations. *Journal of Data Science and Computational Astronomy*, 5(2), 45-60.
5. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press.
6. Ilyas, M. (1984). *A modern guide to astronomical calculations of Islamic calendar, times and qibla*. (Fondasi teori batas Danjon dan Imkanur Rukyat internasional).

7. Junaidi, J. (2022). *Dinamika Kriteria Imkanur Rukyat dan Implikasinya Terhadap Penyatuan Kalender Hijriah di Indonesia*. Pustaka Pelajar. (Pembahasan mendalam kriteria MABIMS 3° dan 6,4°).
8. Mohd Nawawi, M. S. A., Faid, M. S., Saadon, M. H. M., Wahab, R. A., & Ahmad, N. (2024). Hijri month determination in Southeast Asia: An illustration between religion, science, and cultural background. *Heliyon*, 10(20).
9. Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1), 39–64. (Literatur kriteria visibilitas optik kontemporer).
10. Wahidi, A., Yasin, N., & Kadarisman, A. (2019). The beginning of Islamic months determination in Indonesia and Malaysia: Procedure and social condition. *ULUL ALBAB Jurnal Studi Islam*, 20(2), 322–345.

BAB 9 VISUALISASI KOMPAS KIBLAT

9.1 Visualisasi Arah Kiblat pada Plot Polar

(A) Transformasi Data Numerik Menjadi Informasi Visual

Dalam bab-bab sebelumnya, kita telah berhasil menyusun algoritma untuk menghitung sudut azimuth Kiblat dengan presisi tinggi. Namun, bagi pengguna akhir (*end-user*) atau jamaah di masjid, angka skalar seperti "295.15°" sering kali sulit diinterpretasikan secara spasial tanpa alat bantu. Oleh karena itu, langkah krusial berikutnya adalah visualisasi.

Visualisasi kompas Kiblat menggunakan **Plot Polar** adalah metode standar dalam sains data untuk merepresentasikan informasi arah. Berbeda dengan plot linear (Kartesian X-Y), plot polar menggunakan sistem koordinat melingkar yang selaras dengan cara kerja kompas fisik. Titik pusat plot merepresentasikan posisi pengamat (Zenith), sedangkan sudut (theta) merepresentasikan arah mata angin.

Allah *Subhanahu wa Ta'ala* memerintahkan umat-Nya untuk menghadapkan wajah ke arah yang pasti, yang secara saintifik menuntut orientasi vektor yang benar:

فَوَلِّ وَجْهَكَ شَطْرَ الْمَسْجِدِ الْحَرَامِ

"...Maka hadapkanlah wajahmu ke arah Masjidilharam..." (QS. Al-Baqarah [2]: 149).

Dengan memvisualisasikan arah ini dalam bentuk grafis, kita membantu memastikan bahwa "menghadap" tersebut dilakukan dengan keyakinan visual yang mantap. Python, melalui pustaka `Matplotlib`, menyediakan kapabilitas untuk merender kompas digital ini secara profesional.

Latihan 1: Memvisualisasikan Arah Kiblat Menggunakan Lokasi Spesifik

Pada latihan pertama ini, kita akan merancang skrip untuk menggambarkan jarum kompas yang menunjuk ke arah Kiblat. Kita menggunakan contoh lokasi dengan Lintang 39.12° Utara dan Bujur 80.11° Timur.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: VISUALISASI ARAH KIBLAT PADA PLOT POLAR

==========

import matplotlib.pyplot as plt

import numpy as np

def visualisasi_kiblat_polar(azimuth_kiblat, nama_lokasi):

1. Inisialisasi figure dengan proyeksi polar

fig = plt.figure(figsize=(7, 7))

ax = fig.add_subplot(111, projection='polar')

```
# 2. Pengaturan Orientasi: Utara (0°) di atas, searah jarum jam
ax.set_theta_zero_location(&#39;N&#39;)
ax.set_theta_direction(-1)

# 3. Menghitung Vektor Arah (Konversi derajat ke radian)
theta = np.radians(azimuth_kiblat)

# 4. Menggambar Jarum Kiblat
# Vektor ditarik dari pusat (0) ke tepi lingkaran (1)
ax.annotate(&#39;&#39;, xy=(theta, 1), xytext=(0, 0),
            arrowprops=dict(facecolor=&#39;green&#39;,
edgecolor=&#39;darkgreen&#39;,
                            arrowstyle=&#39;fancy&#39;, lw=2))

# 5. Estetika dan Label Mata Angin
ax.set_xticks(np.radians([0, 90, 180, 270]))
ax.set_xticklabels([&#39;U   (0°)&#39;,   &#39;T   (90°)&#39;,   &#39;S
(180°)&#39;, &#39;B (270°)&#39;], fontsize=12)

plt.title(f&quot;Kompas          Kiblat:          {nama_lokasi}\nAzimuth:
{azimuth_kiblat}°&quot;,
          va=&#39;bottom&#39;, fontweight=&#39;bold&#39;)

print(f&quot;Grafik   Kompas   Kiblat   untuk   {nama_lokasi}   berhasil
dibuat.&quot;)
# plt.show()
```

Menjalankan simulasi (Contoh Azimuth: 295.15 derajat)

visualisasi_kiblat_polar(295.15, "Lokasi Uji (39.12 N, 80.11 E)")

</pre>

9.2 Visualisasi Azimuth Matahari pada Plot Polar

(A) Menggunakan Matahari sebagai Kalibrator Kiblat

Salah satu tantangan terbesar dalam menggunakan kompas magnetik adalah adanya **Deklinasi Magnetik** (selisih antara Utara Magnetik dan Utara Sejati). Untuk menghindari kesalahan ini, para ahli falak sering menggunakan matahari sebagai referensi. Jika kita mengetahui posisi matahari (*Solar Azimuth*) pada jam tertentu, kita dapat menggunakannya sebagai titik acuan untuk menentukan arah Kiblat secara absolut.

Penjelasan Setiap Baris Komputasi

Dalam proses ini, Python harus melakukan perhitungan ganda:

1. Menghitung posisi matahari secara *real-time* menggunakan Ephemeris.
2. Memetakan posisi tersebut ke dalam koordinat polar yang sama dengan kompas Kiblat.

Latihan 2: Menghitung Azimuth Matahari (Lintang 39.12 N, Bujur 80.11 E, 13 April 2024, 15:34)

Latihan ini memproses data waktu dan lokasi spesifik untuk mendapatkan posisi matahari sebagai langkah awal kalibrasi visual.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f1f5f9; color: #0f172a; padding: 15px; border-radius: 8px; border: 1px solid #94a3b8; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 2: KOMPUTASI AZIMUTH MATAHARI UNTUK KALIBRASI

==========

from skyfield.api import load, wgs84

def hitung_azimuth_matahari(lintang, bujur, tahun, bulan, hari, jam, menit, zona_waktu):

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

ts = load.timescale()

```
# Konversi waktu lokal ke UTC untuk Skyfield
t = ts.utc(tahun, bulan, hari, jam - zona_waktu, menit)
lokasi = bumi + wgs84.latlon(lintang, bujur)

# Observasi matahari
alt, az, dist = lokasi.at(t).observe(matahari).apparent().altaz()

print(f&amp;quot;Data Matahari pada {hari}/{bulan}/{tahun} pukul {jam}:{menit}&amp;quot;)
print(f&amp;quot;Azimuth Matahari: {az.degrees:.2f} derajat&amp;quot;)

return az.degrees
```

Eksekusi (Contoh data dari buku: 13 April 2024, 15:34, Zona Waktu +6)

az_sun = hitung_azimuth_matahari(39.12, 80.11, 2024, 4, 13, 15, 34, 6)

</pre>

9.3 Visualisasi Azimuth Matahari dan Arah Kiblat pada Plot Polar

(A) Metode Sinkronisasi Bayangan Matahari

Puncak dari teknik visualisasi ini adalah menggabungkan jarum Kiblat dan posisi matahari dalam satu frame. Dengan cara ini, pengamat di lapangan dapat mengarahkan perangkat atau bangunan mereka dengan cara membandingkan bayangan matahari yang jatuh dengan posisi matahari yang ada di layar.

Latihan 3: Memvisualisasikan Arah Kiblat dan Azimuth Matahari (13 April 2024, 15:34)

Latihan ini menyatukan dua vektor informasi ke dalam satu plot polar yang komprehensif.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 3: PLOT GABUNGAN KIBLAT DAN MATAHARI

==========

def plot_kiblat_dan_matahari(az_kiblat, az_matahari):

fig = plt.figure(figsize=(7, 7))

ax = fig.add_subplot(111, projection='polar')

ax.set_theta_zero_location('N')

ax.set_theta_direction(-1)

```
# Vektor Kiblat (Hijau)
theta_k = np.radians(az_kiblat)
ax.annotate(&#39;KIBLAT&#39;, xy=(theta_k, 1), xytext=(0, 0),
            arrowprops=dict(facecolor=&#39;green&#39;,
arrowstyle=&#39;fancy&#39;))

# Vektor Matahari (Oranye/Kuning)
theta_m = np.radians(az_matahari)
ax.annotate(&#39;MATAHARI&#39;, xy=(theta_m, 0.8), xytext=(0, 0),
            arrowprops=dict(facecolor=&#39;orange&#39;,
arrowstyle=&#39;simple&#39;))

plt.title(&quot;Kalibrasi Kiblat Menggunakan Posisi Matahari&quot;)
print(&quot;Plot Gabungan Berhasil Direnderer.&quot;)
```

Menjalankan Plot Gabungan

plot_kiblat_dan_matahari(295.15, az_sun)

</pre>

Latihan 4, 5, 6, dan 7: Eksperimen Lokasi dan Waktu yang Berbeda

Pada latihan-latihan penutup bab ini, pengguna didorong untuk melakukan *batch processing* atau pemrosesan massal untuk berbagai kota besar di dunia pada waktu-waktu kritis (seperti saat *Rashdul Qibla*). Melalui variasi ini, pembaca dapat memahami bagaimana posisi matahari terus bergeser secara dinamis sementara arah Kiblat tetap konstan terhadap koordinat bumi.

Kesimpulan Bab 9:

Visualisasi plot polar mengubah astronomi Islam dari sekadar tumpukan angka menjadi instrumen praktis yang intuitif. Dengan mengintegrasikan posisi matahari sebagai kalibrator, tingkat kepercayaan terhadap akurasi arah Kiblat meningkat secara signifikan, mengeliminasi keraguan akibat interferensi magnetik kompas konvensional.

Daftar Pustaka (Bab 9)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). Astronomical algorithms for Qibla direction and daily prayer times estimation using Python. *Journal of Islamic Science and Technology*, 14(2).
2. Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. CRC Press / Taylor & Francis Group.
3. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.
4. Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites generator*. Astrophysics Source Code Library.
5. Yildirim, F., Kadi, F., & Sahin, S. L. (2024). Developing a new interface for Qibla direction application based on MATLAB GUI. *Survey Review*.

BAB 10 VISUALISASI POSISI MATAHARI PADA WAKTU SALAT

10.1 Memvisualisasikan Posisi Matahari

(A) Urgensi Visualisasi Elevasi dalam Keilmuan Falak

Setelah kita berhasil mengonstruksi algoritma perhitungan waktu salat secara numerik, tantangan berikutnya adalah bagaimana menyajikan data tersebut agar dapat dipahami secara fenomenologis. Visualisasi posisi matahari (ketinggian dan azimut) pada saat waktu salat masuk merupakan instrumen penting untuk memverifikasi kebenaran hisab melalui observasi lapangan.

Bagi seorang Muslim, melihat posisi matahari bukan sekadar pengamatan fisik, melainkan bentuk *tadabbur* terhadap keteraturan ciptaan Allah *Subhanahu wa Ta'ala*. Sebagaimana diisyaratkan dalam Al-Qur'an:

أَلَمْ تَرَ إِلَىٰ رَبِّكَ كَيْفَ مَدَّ الظِّلَّ وَلَوْ شَاءَ لَجَعَلَهُ سَاكِنًا ثُمَّ جَعَلْنَا الشَّمْسَ عَلَيْهِ دَلِيلًا

"Tidakkah engkau memperhatikan (penciptaan) Tuhanmu, bagaimana Dia memanjangkan bayang-bayang itu? Kalau Dia menghendaki, niscaya Dia menjadikannya tetap. Kemudian, Kami jadikan matahari sebagai petunjuk atas (bayang-bayang) itu." (QS. Al-Furqan [25]: 45).

Dalam bab ini, kita akan menggunakan Python untuk mentransformasikan angka-angka waktu salat menjadi representasi grafis posisi matahari di bola langit.

Protokol Pemrograman:

1. **Impor Pustaka:** Menggunakan `Skyfield` untuk data posisi dan `Matplotlib` untuk visualisasi.
2. **Atur Posisi Pengamat dan Matahari:** Mendefinisikan koordinat geospasial (Lintang/Bujur).
3. **Hitung Posisi Matahari Berdasarkan Sudut:** Menentukan *Altitude* (ketinggian) dan *Azimuth*.
4. **Plot Visualisasi:** Merender data ke dalam grafik yang intuitif.

Latihan 1: Menghitung Sudut Ketinggian Matahari untuk Waktu Salat

Pada latihan pertama ini, kita akan membuat fungsi dasar untuk mengonversi waktu spesifik (seperti waktu Zuhur atau Asar yang telah dihitung) menjadi sudut ketinggian matahari di lokasi tertentu.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

LATIHAN 1: MENGHITUNG ELEVASI MATAHARI (ALTITUDE) SAAT WAKTU SALAT

==========

from skyfield.api import load, wgs84

def hitung_posisi_saat_salat(lintang, bujur, tahun, bulan, hari, jam, menit, detik):

planets = load('de421.bsp')

bumi, matahari = planets['earth'], planets['sun']

ts = load.timescale()

```
# Inisialisasi Lokasi Pengamat
lokasi = bumi + wgs84.latlon(lintang, bujur)

# Tentukan Tanggal dan Waktu (UTC)
t = ts.utc(tahun, bulan, hari, jam, menit, detik)

# Hitung Posisi Apparent Matahari
astrometrik = lokasi.at(t).observe(matahari)
apparent = astrometrik.apparent()

# Hitung Ketinggian (Altitude) dan Azimuth
alt, az, jarak = apparent.altaz()

print(f&amp;quot;Hasil Komputasi Posisi Matahari:&amp;quot;)
print(f&amp;quot;Ketinggian (Altitude): {alt.degrees:.2f} derajat&amp;quot;)
print(f&amp;quot;Arah (Azimuth)      : {az.degrees:.2f} derajat&amp;quot;)

return alt.degrees, az.degrees
```

Contoh: Mengecek posisi matahari di Jakarta saat Zuhur (Misal 12:05 WIB / 05:05 UTC)

alt_z, az_z = hitung_posisi_saat_salat(-6.2, 106.8, 2026, 3, 16, 5, 5, 0)

</pre>

10.2 Visualisasi Posisi Matahari pada Berbagai Waktu Salat

Visualisasi yang paling efektif adalah dengan menggambarkan matahari pada sebuah busur setengah lingkaran yang merepresentasikan kubah langit. Berikut adalah rangkaian latihan untuk memvisualisasikan matahari pada setiap waktu salat:

Latihan 2: Visualisasi Posisi Matahari pada Waktu Zuhur

Pada waktu Zuhur, matahari berada pada titik tertingginya (transit). Visualisasi akan menunjukkan matahari berada di dekat titik Zenith.

Latihan 3: Visualisasi Posisi Matahari pada Waktu Asar

Visualisasi ini akan menunjukkan matahari yang mulai menurun di ufuk barat, sesuai dengan kriteria panjang bayangan yang telah dibahas di Bab 7.

Latihan 4: Visualisasi Posisi Matahari pada Waktu Maghrib

Matahari akan divisualisasikan tepat berada di garis ufuk (0 derajat) atau sedikit di bawahnya (-0.833 derajat) untuk menunjukkan momen berbuka puasa.

Latihan 5: Visualisasi Posisi Matahari pada Waktu Isya'

Matahari berada jauh di bawah ufuk (-18 derajat). Visualisasi ini membantu memahami mengapa langit menjadi gelap total saat Isya' masuk.

Latihan 6: Visualisasi Posisi Matahari pada Waktu Subuh

Mirip dengan Isya', namun matahari berada di ufuk timur (-20 derajat). Latihan ini meliputi visualisasi fajar Shadiq yang mulai menyingsing.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #2b2b2b; color: #a9b7c6; padding: 15px; border-radius: 8px; border: 1px solid #444; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

CONTOH PLOT VISUALISASI POSISI MATAHARI (ALL-IN-ONE)

==========

import matplotlib.pyplot as plt

def plot_posisi_matahari_salat(data_salat):

data_salat: list of dictionary {nama: altitude}

plt.figure(figsize=(10, 5))

```
# Garis Ufuk
plt.axhline(0, color=&#39;black&#39;, lw=2)

for salat in data_salat:
    nama = salat[&#39;nama&#39;]
    alt = salat[&#39;alt&#39;]
```

```
    plt.scatter(nama, alt, s=200, label=nama)
    plt.annotate(f&amp;quot;{alt}°&amp;quot;, (nama, alt + 1))

plt.title(&amp;quot;Visualisasi Ketinggian Matahari pada Berbagai Waktu Salat&amp;quot;)
plt.ylabel(&amp;quot;Ketinggian (Derajat)&amp;quot;)
plt.ylim(-25, 90)
plt.grid(True, alpha=0.3)
plt.show()
```

Contoh input data hasil hitungan

```
data_simulasi = [
{'nama': 'Subuh', 'alt': -20.0},
{'nama': 'Syuruk', 'alt': -0.83},
{'nama': 'Zuhur', 'alt': 82.5},
{'nama': 'Asar', 'alt': 36.2},
{'nama': 'Maghrib', 'alt': -0.83},
{'nama': 'Isya', 'alt': -18.0}
]
plot_posisi_matahari_salat(data_simulasi)
</pre>
```

Kesimpulan Bab 10:

Melalui visualisasi ini, korelasi antara hukum fikih dan fenomena astronomis menjadi sangat nyata. Kita tidak hanya "menghitung" waktu salat, tetapi "melihat" dinamika matahari di langit, yang memperkuat keyakinan dalam menjalankan ibadah tepat pada waktunya.

Daftar Pustaka (Bab 10)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). *Astronomical algorithms for Qibla direction and daily prayer times estimation using Python*. Journal of Islamic Science and Technology.
2. Anwar, S. (2018). *Ilmu Falak: Hisab Waktu Salat, Arah Kiblat, dan Awal Bulan*. UII Press.
3. Faid, M. S., et al. (2024). *Python for Islamic Astronomy*. CRC Press.
4. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering.
5. Rhodes, B. (2019). *Skyfield: High precision research-grade positions*. ASCL.

BAB 11 VISUALISASI DATA OBSERVASI BULAN SABIT (HILAL)

11.1 Pentingnya Visualisasi dalam Verifikasi Hilal

Dalam disiplin ilmu falak, data numerik mengenai ketinggian dan elongasi bulan sering kali terasa abstrak bagi masyarakat awam maupun bagi hakim di sidang isbat. Visualisasi data observasi hilal menjadi jembatan krusial untuk membuktikan apakah bulan sabit tersebut secara logika optik "mungkin" untuk dilihat ataukah klaim saksi di lapangan perlu ditinjau ulang.

Sebagaimana Allah *Subhanahu wa Ta'ala* menjelaskan fase-fase bulan untuk memudahkan manusia dalam perhitungan:

وَالْقَمَرَ قَدَّرْنٰهُ مَنَازِلَ حَتّٰى عَادَ كَالْعُرْجُوْنِ الْقَدِيْمِ

"Dan telah Kami tetapkan bagi bulan manzilah-manzilah, sehingga (setelah dia sampai ke manzilah yang terakhir) kembalilah dia sebagai bentuk tandan yang tua." (QS. Yasin [36]: 39).

Visualisasi "tandan yang tua" (sabit tipis) ini dalam Python membantu kita memetakan posisi bulan relatif terhadap matahari dan ufuk. Hal ini memungkinkan kita untuk melihat apakah hilal sudah berada di atas kriteria visibilitas (seperti kriteria MABIMS) atau masih terbenam di balik cahaya senja.

11.2 Praktik Pertama: Penang, Malaysia

Pada praktik pertama ini, kita akan menyimulasikan visualisasi hilal untuk wilayah Penang, Malaysia. Kita akan memplot posisi bulan dan matahari pada koordinat polar untuk melihat jarak sudutnya secara langsung.

<pre style="font-family: 'Consolas', 'Courier New', monospace; background-color: #f8f9fa; color: #1e293b; padding: 15px; border-radius: 8px; border: 1px solid #cbd5e1; font-size: 11pt; line-height: 1.6; white-space: pre-wrap;">

==========

PRAKTIK 1: VISUALISASI POSISI HILAL DI PENANG, MALAYSIA

==========

import matplotlib.pyplot as plt

import numpy as np

def plot_hilal_penang(alt_b, az_b, alt_m, az_m):

fig = plt.figure(figsize=(8, 6))

ax = fig.add_subplot(111)

```
# Garis Ufuk (Horizon)
ax.axhline(0, color=&#39;brown&#39;, lw=2, label=&#39;Horizon&#39;)

# Plot Matahari (Biasanya di bawah ufuk saat sunset)
ax.scatter(az_m,       alt_m,       color=&#39;orange&#39;,       s=500,
label=&#39;Matahari&#39;)

# Plot Bulan (Hilal)
ax.scatter(az_b, alt_b, color=&#39;gold&#39;, s=100, marker=&#39;o&#39;,
label=&#39;Hilal&#39;)

ax.set_title(&quot;Posisi Hilal saat Sunset di Penang&quot;)
ax.set_xlabel(&quot;Azimuth (Derajat)&quot;)
ax.set_ylabel(&quot;Ketinggian (Derajat)&quot;)
ax.legend()
ax.grid(True, alpha=0.3)

print(&quot;Visualisasi Hilal Penang Berhasil.&quot;)
```

</pre>

11.3 Praktik Kedua: Banda Aceh, Indonesia

Banda Aceh merupakan salah satu titik paling barat di Indonesia, menjadikannya lokasi strategis untuk observasi hilal karena matahari terbenam paling akhir, memberikan waktu lebih bagi hilal untuk bertambah tinggi.

Latihan 1, 2, 3, 4, dan 5: Eksperimen Parameter Visibilitas

Melalui rangkaian latihan ini, pembaca akan diajak untuk memodifikasi variabel input untuk melihat perubahan visual:

- **Latihan 1:** Menghitung pengaruh refraksi atmosfer pada tampilan ketinggian bulan.
- **Latihan 2:** Memvisualisasikan beda azimuth (DAZ) untuk memandu arah teleskop.
- **Latihan 3:** Membuat grafik perbandingan ketinggian hilal di berbagai kota (Sabang sampai Merauke).
- **Latihan 4:** Menyimulasikan ketebalan sabit bulan berdasarkan persentase iluminasi.
- **Latihan 5:** Merender peta visibilitas global sederhana untuk menentukan wilayah mana saja yang mungkin melihat hilal pada hari tersebut.

Kesimpulan Bab 11:

Visualisasi data observasi bulan sabit melengkapi proses hisab dengan bukti visual yang empiris. Dengan Python, kita dapat memprediksi tidak hanya "kapan" bulan baru dimulai,

tetapi juga "bagaimana" penampakan hilal tersebut di langit, sehingga meningkatkan akurasi dan keyakinan dalam penetapan kalender Hijriah global.

Daftar Pustaka (Bab 11)

1. Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). *Astronomical algorithms for Qibla direction and daily prayer times estimation using Python*. Journal of Islamic Science and Technology.
2. Faid, M. S., et al. (2024). *Python for Islamic Astronomy*. CRC Press.
3. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering.
4. Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*.
5. Wahidi, A., et al. (2019). The beginning of Islamic months determination in Indonesia and Malaysia. *ULUL ALBAB*.

BAB 12 KESIMPULAN AKHIR

Integrasi Wahyu dan Sains di Era Komputasi

Penyusunan buku ini merupakan ikhtiar ilmiah untuk membuktikan bahwa Islam dan sains bukanlah dua entitas yang terpisah, melainkan satu kesatuan epistemologis yang saling menguatkan. Melalui bahasa pemrograman Python, kita telah mendekonstruksi kompleksitas mekanika selestial menjadi algoritma yang transparan, akurat, dan aplikatif.

Beberapa poin esensial yang dapat dirangkum dari seluruh pembahasan adalah:

1. **Akurasi adalah Amanah:** Dalam ibadah yang terikat waktu dan ruang, presisi komputasi adalah bentuk kehati-hatian (*ihtiyath*) yang wajib diupayakan oleh para ahli falak.
2. **Transformasi Metodologis:** Peralihan dari tabel manual (*Zīj*) ke pustaka efemeris modern seperti *Skyfield* (JPL NASA) memberikan jaminan validitas data yang setara dengan standar observatorium global.
3. **Literasi Visual:** Visualisasi data (plot polar, kurva diurnal, dan simulasi hilal) berperan penting dalam mendemokratisasi ilmu falak agar lebih mudah dipahami oleh pembuat kebijakan dan masyarakat luas.

Semoga karya ini menjadi amal jariyah yang bermanfaat bagi pengembangan pendidikan agama Islam dan sains di masa depan.

DAFTAR PUSTAKA

Ahmad, S., Nawawi, M. S. A. M., & Faid, M. S. (2020). Astronomical algorithms for Qibla direction and daily prayer times estimation using Python. *Journal of Islamic Science and Technology*, 14(2), 101-115.

Anwar, S. (2018). *Ilmu Falak: Hisab Waktu Salat, Arah Kiblat, dan Awal Bulan*. Yogyakarta: UII Press.

Ar-Razi, F. D. (1981). *Mafatih al-Ghaib (Tafsir al-Kabir)*. Beirut: Dar al-Fikr.

Asrin, A., Jamil, A., & Nawawi, M. S. A. M. (2018). *Aplikasi Hisab Kiblat Berbasis Geodesi Sferis dan Elipsoida*. Jakarta: Pustaka Falak Indonesia.

Faid, M. S., et al. (2019). The computational algorithm of prayer times estimation for high latitude regions. *International Journal of Advanced Science and Technology*, 28(16).

Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., & Ahmad, N. (2022). Islamic historical review on the middle age lunar crescent visibility criterion. *Journal of Al-Tamaddun*, 17(1).

Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., et al. (2023). Python applications in astronomical computations: A comparative analysis of Skyfield and PyEphem for Hijri calendar calculations. *Journal of Data Science and Computational Astronomy*, 5(2), 45-60.

Faid, M. S., Nawawi, M. S. A. M., Saadon, M. H. M., Ahmad, N., & Ali @ Mat Zin, A. (2024). *Python for Islamic Astronomy: Modern Computational Approaches to Hijri Calendar, Qibla, and Prayer Times*. Boca Raton: CRC Press / Taylor & Francis Group.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

Ilyas, M. (1984). *A modern guide to astronomical calculations of Islamic calendar, times and qibla*. Kuala Lumpur: Berita Publishing.

Junaidi, J. (2022). *Dinamika Kriteria Imkanur Rukyat dan Implikasinya Terhadap Penyatuan Kalender Hijriah di Indonesia*. Yogyakarta: Pustaka Pelajar.

King, D. A. (1993). *Astronomy in the service of Islam*. Aldershot: Routledge.

Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Richmond, Virginia: Willmann-Bell.

Mohd Nawawi, M. S. A., Faid, M. S., Saadon, M. H. M., Wahab, R. A., & Ahmad, N. (2024). Hijri month determination in Southeast Asia: An illustration between religion, science, and cultural background. *Heliyon*, 10(20).

Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1), 39–64.

Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites generator*. Astrophysics Source Code Library.

Wahidi, A., Yasin, N., & Kadarisman, A. (2019). The beginning of Islamic months determination in Indonesia and Malaysia: Procedure and social condition. *ULUL ALBAB Jurnal Studi Islam*, 20(2), 322–345.

Yildirim, F., Kadi, F., & Sahin, S. L. (2024). Developing a new interface for Qibla direction application based on MATLAB GUI. *Survey Review*.

GLOSARIUM LENGKAP A-Z

Praktek Python untuk Astronomi Islam

Huruf	Istilah	Definisi & Konteks dalam Buku
A	**Algoritma**	Urutan langkah logis dalam kode Python untuk menyelesaikan perhitungan, seperti algoritma mencari waktu konjungsi.
	Altitude (Irtifa')	Sudut ketinggian benda langit dari ufuk. Nilai 0° berarti di ufuk, dan 90° adalah tepat di atas kepala (Zenith).
	Azimuth	Sudut horizontal benda langit yang diukur dari titik Utara (0°) searah jarum jam hingga 360°.
B	**Bujur (Longitude)**	Posisi timur atau barat suatu tempat terhadap garis Meridian Utama (Greenwich), esensial dalam menentukan waktu salat lokal.
	Barycentric	Titik pusat massa sistem tata surya, menjadi acuan perhitungan posisi planet dalam *library* Skyfield.
C	**Cotangent (Cot)**	Fungsi trigonometri yang digunakan dalam penentuan waktu Asar berdasarkan perbandingan panjang bayangan.
	Civil Twilight	Senja sipil; kondisi saat pusat matahari berada 6° di bawah ufuk, langit masih cukup terang untuk aktivitas luar ruangan.
D	**Deklinasi**	Koordinat astronomis yang menunjukkan jarak utara atau selatan suatu benda langit dari ekuator langit.
	Delta T (ΔT)	Selisih antara waktu dinamis (teratur secara fisik) dan waktu universal (berbasis rotasi bumi yang tidak stabil), vital untuk presisi tinggi.
	DAZ (*Difference in Azimuth*)	Selisih sudut azimuth antara matahari dan bulan, digunakan sebagai panduan mengarahkan teleskop saat rukyat.
E	**Elongasi**	Jarak sudut antara matahari dan bulan sebagaimana terlihat dari bumi; parameter utama ketebalan sabit hilal.
	Ephemeris	Tabel data posisi benda langit pada waktu tertentu. Dalam Python, kita menggunakan file `.bsp` dari JPL NASA.
	Equation of Time	Perata waktu; selisih antara waktu matahari sejati (jam matahari) dan waktu matahari rata-rata (jam dinding).
F	**Fajar Shadiq**	Cahaya fajar yang menyebar horizontal di ufuk timur, menandai dimulainya waktu Subuh dan batas akhir makan sahur.
	Fajar Kadzib	Fajar semu; cahaya redup yang menjulang vertikal di langit timur sebelum munculnya fajar Shadiq.
	Fraction Illumination	Persentase luasan piringan bulan yang terkena sinar matahari, menentukan tingkat kecerahan hilal.
G	**Geodesi**	Ilmu tentang pengukuran bentuk bumi. Rumus geodesi digunakan untuk menghitung arah kiblat pada bumi elipsoida.
	Ghuruub	Istilah astronomi Islam untuk peristiwa terbenamnya matahari (sunset).
	Google Colab	Platform berbasis cloud dari Google untuk menulis dan menjalankan kode Python secara kolaboratif.
H	**Hilal**	Bulan sabit tipis yang tampak pertama kali setelah konjungsi pada saat matahari terbenam.
	Hisab	Metode perhitungan matematis astronomis untuk menentukan posisi benda langit tanpa observasi langsung.

	Haversine Formula	Persamaan trigonometri untuk menghitung jarak antara dua titik pada permukaan bola (sering digunakan dalam navigasi).
I	**Ijtimak (Konjungsi)**	Momen saat bulan dan matahari berada pada bujur ekliptika yang sama; titik nol lahirnya bulan baru.
	Ihtiyath	Waktu pengaman (biasanya 2 menit) untuk mengantisipasi kesalahan hitung atau variasi lokal dalam jadwal salat.
	Imkanur Rukyat	Kriteria batas minimal posisi hilal (tinggi dan elongasi) yang memungkinkan bulan sabit bisa terlihat secara fisik.
J	**JPL NASA**	*Jet Propulsion Laboratory*; penyedia standar ephemeris ruang angkasa yang diintegrasikan dalam Python.
	Jari-jari (Semi-diameter)	Setengah dari diameter sudut matahari atau bulan (sekitar 16 menit busur).
K	**Kiblat**	Arah menuju Ka'bah di Makkah. Dalam buku ini, dihitung menggunakan rumus trigonometri bola atau elipsoida.
	Kulminasi (Transit)	Saat benda langit mencapai titik tertinggi di meridian pengamat; untuk matahari, ini adalah waktu Zuhur.
L	**Lintang (Latitude)**	Posisi utara atau selatan suatu tempat terhadap garis ekuator (khatulistiwa).
	Library	Modul tambahan dalam Python (seperti `numpy` atau `skyfield`) yang memperluas fungsionalitas pemrograman dasar.
M	**MABIMS**	Kriteria visibilitas hilal baru (Tinggi 3°, Elongasi 6.4°) yang digunakan oleh Indonesia, Malaysia, Brunei, dan Singapura.
	Matplotlib	*Library* utama Python untuk membuat visualisasi grafis, seperti grafik ketinggian matahari atau kompas kiblat.
	Meridian	Garis imajiner yang menghubungkan kutub utara dan selatan melewati titik tepat di atas kepala pengamat.
N	**Nautical Twilight**	Fase senja saat pusat matahari berada 12° di bawah ufuk; langit mulai gelap namun bintang navigasi terlihat.
	Numpy	*Library* Python untuk komputasi numerik dan operasi matriks/array yang sangat cepat.
O	**Offset**	Selisih atau pergeseran nilai, biasanya merujuk pada perbedaan jam UTC dengan zona waktu lokal (WIB/WITA/WIT).
	Obliquity	Kemiringan sumbu rotasi bumi terhadap bidang orbitnya (ekliptika), penyebab terjadinya perubahan musim.
P	**Parallaks**	Perbedaan posisi benda langit (terutama bulan) jika dilihat dari pusat bumi dibandingkan dari permukaan bumi.
	Python	Bahasa pemrograman tingkat tinggi yang digunakan dalam buku ini karena kemudahannya dalam pengolahan data sains.
Q	**Quadrant**	Pembagian lingkaran menjadi empat bagian (0-90, 90-180, 180-270, 270-360) untuk menentukan arah azimuth.
	Quick Sort	Algoritma pengurutan data yang sering digunakan dalam Python untuk memproses ribuan data ephemeris.
R	**Refraksi**	Pembiasan cahaya oleh atmosfer bumi yang menyebabkan benda langit tampak sedikit lebih tinggi dari posisi sebenarnya.
	Ru'yatul Hilal	Aktivitas observasi atau pengamatan bulan sabit secara langsung (visual) untuk menentukan awal bulan.
S	**Skyfield**	Pustaka Python canggih yang digunakan untuk menghitung posisi bintang, planet, dan satelit dengan akurasi tinggi.
	Syafak	Mega atau pendar cahaya di ufuk barat setelah matahari terbenam; hilangnya syafak merah menandai waktu Isya.

	Syuruk	Waktu terbitnya matahari; batas akhir bagi pelaksanaan salat Subuh.
T	**Toposentris**	Perhitungan posisi benda langit berdasarkan sudut pandang pengamat di permukaan bumi (terpengaruh elevasi dan refraksi).
	Trigonometri Sferis	Cabang matematika yang mempelajari hubungan antara sudut dan sisi segitiga yang terletak pada permukaan bola.
U	**Ufuk (Horizon)**	Garis pertemuan antara langit dan bumi secara visual; acuan utama untuk terbit dan terbenamnya benda langit.
	UTC (*Coordinated Universal Time*)	Standar waktu dunia yang menjadi acuan perhitungan ephemeris sebelum dikonversi ke waktu lokal.
V	**Visualisasi**	Representasi data numerik ke dalam bentuk gambar atau grafik agar lebih mudah dianalisis dan dipahami.
	Variabel	Wadah dalam kode Python untuk menyimpan data, seperti nilai lintang atau nama lokasi.
W	**WGS 84**	Standar model bumi elipsoida yang digunakan oleh GPS dan menjadi acuan perhitungan kiblat modern.
	Waktu Tahrim	Waktu-waktu yang dilarang untuk salat, seperti saat matahari tepat terbit hingga meninggi seukuran satu tombak.
X	**X-Axis**	Sumbu horizontal dalam grafik (Matplotlib), biasanya digunakan untuk merepresentasikan skala waktu atau tanggal.
Y	**Y-Axis**	Sumbu vertikal dalam grafik, biasanya digunakan untuk menunjukkan nilai ketinggian matahari atau bulan.
Z	**Zenith**	Titik tertinggi di bola langit yang berada tepat 90° di atas kepala pengamat.
	Zīj	Istilah klasik untuk tabel astronomi yang memuat posisi matahari, bulan, dan planet untuk keperluan hisab.

LAMPIRAN 1 SOURCE CODE CONSOLE

```
"""
File: python_islami_console.py
Deskripsi: Kumpulan kode program untuk Astronomi Islam (Ilmu Falak)
        Mencakup perhitungan Waktu Salat, Arah Kiblat, dan Visibilitas Hilal.
"""

import math
import datetime
from datetime import timedelta, timezone
import numpy as np
import matplotlib.pyplot as plt
from skyfield.api import load, wgs84
from skyfield import almanac

# ==============================================================================
# 1. UTILITAS & INISIALISASI
# ==============================================================================

def inisialisasi_sistem_falak():
    """Menampilkan pesan konfirmasi sistem falak."""
    print("=====================================================")
    print("SISTEM KOMPUTASI ASTRONOMI ISLAM (ILMU FALAK)")
    print("Status Sistem : AKTIF (Siap Menerima Parameter)")
    print("=====================================================\n")

    # Menggunakan datetime.UTC untuk Python 3.11+ atau timezone.utc
    try:
        w_utc = datetime.datetime.now(datetime.UTC)
    except AttributeError:
        w_utc = datetime.datetime.now(timezone.utc)

    print(f"Waktu Universal (UTC) saat ini: {w_utc.strftime('%Y-%m-%d %H:%M:%S')}")

def konversi_desimal_ke_dms(derajat_desimal):
    """Mengonversi derajat desimal ke format Derajat, Menit, Detik."""
    is_negatif = derajat_desimal < 0
    derajat_absolut = abs(derajat_desimal)
    derajat = int(derajat_absolut)
    sisa_desimal = (derajat_absolut - derajat) * 60
    menit = int(sisa_desimal)
    detik = (sisa_desimal - menit) * 60
    tanda = "-" if is_negatif else ""
    return f"{tanda}{derajat}° {menit}' {detik:.2f}\""

# ==============================================================================
# 2. PERHITUNGAN ARAH KIBLAT
# ==============================================================================
```

```
def kalkulasi_arah_kiblat(lintang_pengamat, bujur_pengamat):
    """Menghitung Azimuth Kiblat menggunakan metode Trigonometri Bola."""
    LINTANG_KAABAH = 21.4225
    BUJUR_KAABAH = 39.8262
    phi_1 = math.radians(lintang_pengamat)
    lambda_1 = math.radians(bujur_pengamat)
    phi_2 = math.radians(LINTANG_KAABAH)
    lambda_2 = math.radians(BUJUR_KAABAH)
    delta_lambda = lambda_2 - lambda_1
    Y = math.sin(delta_lambda)
    X = (math.cos(phi_1) * math.tan(phi_2)) - (math.sin(phi_1) * math.cos(delta_lambda))
    azimuth_kiblat = (math.degrees(math.atan2(Y, X)) + 360) % 360
    return azimuth_kiblat

def hitung_jarak_haversine(lintang_lokal, bujur_lokal):
    """Menghitung jarak terpendek menuju Ka'bah."""
    R_BUMI_KM = 6371.0
    lat_m = math.radians(21.422487)
    lon_m = math.radians(39.826206)
    lat_l = math.radians(lintang_lokal)
    lon_l = math.radians(bujur_lokal)
    delta_lat = lat_m - lat_l
    delta_lon = lon_m - lon_l
    a = (math.sin(delta_lat / 2)**2) + (math.cos(lat_l) * math.cos(lat_m) * (math.sin(delta_lon / 2)**2))
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R_BUMI_KM * c

# ==============================================================================
# 3. PERHITUNGAN WAKTU SALAT
# ==============================================================================

def komputasi_final_zuhur(lintang, bujur, zona_waktu_offset, tahun, bulan, hari):
    """Mencari waktu Zuhur (Transit Matahari) dengan Ihtiyath."""
    planets = load('de421.bsp')
    ts = load.timescale()
    lokasi = wgs84.latlon(lintang, bujur)

    t0 = ts.utc(tahun, bulan, hari, 0, 0)
    t1 = ts.utc(tahun, bulan, hari + 1, 0, 0)

    w_kejadian, t_kejadian = almanac.find_discrete(t0, t1, almanac.meridian_transits(planets,
planets['sun'], lokasi))

    w_zuhur_utc = None
    for w, t in zip(w_kejadian, t_kejadian):
        if t == 1: # Upper Transit
            w_zuhur_utc = w
            break

    # PERBAIKAN KRUSIAL: Gunakan 'is not None' untuk objek Skyfield Time
    if w_zuhur_utc is not None:
```

```
        dt_lokal = w_zuhur_utc.utc_datetime().replace(tzinfo=None) +
timedelta(hours=zona_waktu_offset)
        dt_zuhur_aman = dt_lokal + timedelta(minutes=2)
        return dt_zuhur_aman
    return None

def komputasi_final_maghrib(lintang, bujur, zona_waktu_offset, tahun, bulan, hari):
    """Menghitung waktu Maghrib (Sunset) dengan Ihtiyath."""
    planets = load('de421.bsp')
    ts = load.timescale()
    lokasi = wgs84.latlon(lintang, bujur)

    t0 = ts.utc(tahun, bulan, hari, 0, 0)
    t1 = ts.utc(tahun, bulan, hari + 1, 0, 0)

    w_kejadian, t_kejadian = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(planets, lokasi))
    for w, is_sunrise in zip(w_kejadian, t_kejadian):
        if not is_sunrise: # Sunset
            dt_maghrib = w.utc_datetime().replace(tzinfo=None) + timedelta(hours=zona_waktu_offset) +
timedelta(minutes=2)
            return dt_maghrib
    return None

# ==============================================================================
# 4. EKSEKUSI UTAMA
# ==============================================================================

if __name__ == "__main__":
    inisialisasi_sistem_falak()

    # Parameter Lokasi (Jakarta)
    LINTANG, BUJUR = -6.20, 106.81
    ZONA_WAKTU = 7
    sekarang = datetime.datetime.now()

    print(f"--- DATA LOKASI: JAKARTA ({LINTANG}, {BUJUR}) ---")

    # 1. Hitung Kiblat
    kiblat = kalkulasi_arah_kiblat(LINTANG, BUJUR)
    jarak = hitung_jarak_haversine(LINTANG, BUJUR)
    print(f"Azimuth Kiblat : {kiblat:.2f}° ({konversi_desimal_ke_dms(kiblat)})")
    print(f"Jarak ke Makkah: {jarak:.2f} km")

    # 2. Hitung Waktu Zuhur
    zuhur = komputasi_final_zuhur(LINTANG, BUJUR, ZONA_WAKTU, sekarang.year, sekarang.month,
sekarang.day)
    if zuhur is not None:
        print(f"Waktu Zuhur    : {zuhur.strftime('%H:%M:%S')} WIB")

    # 3. Hitung Waktu Maghrib
```

```
    maghrib = komputasi_final_maghrib(LINTANG, BUJUR, ZONA_WAKTU, sekarang.year, sekarang.month, sekarang.day)
    if maghrib is not None:
        print(f"Waktu Maghrib  : {maghrib.strftime('%H:%M:%S')} WIB")

    print("\n[EKSEKUSI SELESAI]")
```

LAMPIRAN 2 SOURCE CODE VIRTUAL

```
import math
import datetime
from datetime import timedelta
import tkinter as tk
from tkinter import ttk, messagebox
from skyfield.api import load, wgs84

# ==============================================================================
# DATABASE KOTA PENTING DI PULAU JAWA
# ==============================================================================
DATA_JAWA = {
    "Jakarta": {"lat": -6.1751, "lon": 106.8650},
    "Serang": {"lat": -6.1104, "lon": 106.1605},
    "Bandung": {"lat": -6.9175, "lon": 107.6191},
    "Bekasi": {"lat": -6.2383, "lon": 106.9756},
    "Depok": {"lat": -6.4025, "lon": 106.7942},
    "Bogor": {"lat": -6.5971, "lon": 106.7952},
    "Cirebon": {"lat": -6.7320, "lon": 108.5523},
    "Semarang": {"lat": -6.9667, "lon": 110.4167},
    "Surakarta (Solo)": {"lat": -7.5707, "lon": 110.8297},
    "Pekalongan": {"lat": -6.8886, "lon": 109.6753},
    "Tegal": {"lat": -6.8676, "lon": 109.1371},
    "Yogyakarta": {"lat": -7.7956, "lon": 110.3695},
    "Surabaya": {"lat": -7.2575, "lon": 112.7521},
    "Malang": {"lat": -7.9839, "lon": 112.6214},
    "Jember": {"lat": -8.1845, "lon": 113.6681},
    "Banyuwangi": {"lat": -8.2192, "lon": 114.3691},
}

# ==============================================================================
# ENGINE PERHITUNGAN
# ==============================================================================

class FalakEngine:
    def __init__(self):
        self.planets = load('de421.bsp')
        self.earth = self.planets['earth']
        self.sun = self.planets['sun']
        self.ts = load.timescale()

    def get_sun_alt(self, t, pengamat):
        astrometric = pengamat.at(t).observe(self.sun).apparent()
        alt, az, d = astrometric.altaz()
        return alt.degrees

    def find_time_at_alt(self, target_alt, pengamat, t_start_utc, t_end_utc, direction="falling"):
        low, high = t_start_utc, t_end_utc
```

```
        for _ in range(20):
            mid = low + (high - low) / 2
            t_mid = self.ts.from_datetime(mid.replace(tzinfo=datetime.timezone.utc))
            current_alt = self.get_sun_alt(t_mid, pengamat)
            if direction == "falling":
                if current_alt > target_alt: low = mid
                else: high = mid
            else:
                if current_alt < target_alt: low = mid
                else: high = mid
        return low

    def calculate_all(self, lat, lon, offset, s_fajr, s_isha, ihtiyath_detik):
        lokasi = wgs84.latlon(lat, lon)
        pengamat = self.earth + lokasi
        now = datetime.datetime.now(datetime.timezone.utc)
        base_day = now.replace(hour=0, minute=0, second=0, microsecond=0) - timedelta(hours=offset)

        # Dzuhur
        times = [base_day + timedelta(minutes=m) for m in range(600, 900)]
        dzuhur_utc = max(times, key=lambda t: self.get_sun_alt(self.ts.from_datetime(t), pengamat))
        z_alt = self.get_sun_alt(self.ts.from_datetime(dzuhur_utc), pengamat)

        # Ashar, Maghrib, Isya
        ashar_alt = math.degrees(math.atan(1 / (1 + math.tan(math.radians(90 - z_alt)))))
        ashar_utc = self.find_time_at_alt(ashar_alt, pengamat, dzuhur_utc, dzuhur_utc +
timedelta(hours=4), "falling")
        maghrib_utc = self.find_time_at_alt(-0.833, pengamat, dzuhur_utc, dzuhur_utc +
timedelta(hours=8), "falling")
        isya_utc = self.find_time_at_alt(s_isha, pengamat, maghrib_utc, maghrib_utc +
timedelta(hours=4), "falling")

        # Terbit, Subuh
        terbit_utc = self.find_time_at_alt(-0.833, pengamat, base_day, dzuhur_utc, "rising")
        subuh_utc = self.find_time_at_alt(s_fajr, pengamat, base_day, terbit_utc, "rising")

        # Fungsi konversi dengan Ihtiyath dalam detik
        def to_local(utc_dt, add_ihtiyath=True):
            buffer = timedelta(seconds=ihtiyath_detik) if add_ihtiyath else timedelta(0)
            return (utc_dt + timedelta(hours=offset) + buffer).strftime("%H:%M:%S")

        return {
            "Subuh": to_local(subuh_utc),
            "Terbit": to_local(terbit_utc, add_ihtiyath=False), # Terbit biasanya tanpa ihtiyath positif
            "Dzuhur": to_local(dzuhur_utc),
            "Ashar": to_local(ashar_utc),
            "Maghrib": to_local(maghrib_utc),
            "Isya": to_local(isya_utc),
            "Kiblat": self.get_kiblat(lat, lon)
        }
```

```
    def get_kiblat(self, lat, lon):
        phi1, lam1 = math.radians(lat), math.radians(lon)
        phi2, lam2 = math.radians(21.4225), math.radians(39.8262)
        y = math.sin(lam2 - lam1)
        x = math.cos(phi1)*math.tan(phi2) - math.sin(phi1)*math.cos(lam2-lam1)
        return f"{(math.degrees(math.atan2(y, x)) + 360) % 360:.2f}°"

# ==============================================================================
# GUI INTERFACE
# ==============================================================================

class FalakAppPrecision:
    def __init__(self, root):
        self.root = root
        self.root.title("Falak Pro v3.4 - Ihtiyath Precision")
        self.root.geometry("500x720")
        self.root.configure(bg="#020617")
        self.engine = FalakEngine()
        self.setup_ui()

    def setup_ui(self):
        # Header
        top = tk.Frame(self.root, bg="#020617", pady=10)
        top.pack(fill=tk.X)
        tk.Label(top, text="JADWAL SALAT PRESISI", font=("Segoe UI", 16, "bold"), fg="#38bdf8",
bg="#020617").pack()

        # Input Area
        inp = tk.Frame(self.root, bg="#020617", padx=30)
        inp.pack(fill=tk.X)

        # Kriteria
        tk.Label(inp, text="Kriteria:", fg="#94a3b8", bg="#020617").grid(row=0, column=0, sticky=tk.W,
pady=2)
        self.std_var = tk.StringVar(value="Kemenag RI (Subuh -20°)")
        ttk.Combobox(inp, textvariable=self.std_var, state="readonly", values=["Kemenag RI (Subuh -
20°)", "Muhammadiyah (Subuh -18°)"], width=25).grid(row=0, column=1, sticky=tk.W, pady=2)

        # Kota
        tk.Label(inp, text="Kota:", fg="#94a3b8", bg="#020617").grid(row=1, column=0, sticky=tk.W,
pady=2)
        self.city_var = tk.StringVar(value="Semarang")
        ttk.Combobox(inp, textvariable=self.city_var, values=sorted(list(DATA_JAWA.keys())),
state="readonly", width=25).grid(row=1, column=1, sticky=tk.W, pady=2)

        # IHTIYATH (DETIK) - Fitur Baru
        tk.Label(inp, text="Ihtiyath (Detik):", fg="#38bdf8", bg="#020617", font=("Segoe UI", 9,
"bold")).grid(row=2, column=0, sticky=tk.W, pady=2)
        self.ihtiyath_var = tk.StringVar(value="60")
        self.ent_ihtiyath = ttk.Entry(inp, textvariable=self.ihtiyath_var, width=10)
        self.ent_ihtiyath.grid(row=2, column=1, sticky=tk.W, pady=2)
```

```
        # Button
        tk.Button(self.root, text="HITUNG JADWAL", command=self.process, bg="#0ea5e9", fg="white",
                  font=("Segoe UI", 10, "bold"), relief=tk.FLAT, pady=8, cursor="hand2").pack(fill=tk.X,
padx=50, pady=10)

        # Result Display (Compact Spacing 1)
        self.res_frame = tk.Frame(self.root, bg="#0f172a", bd=1, relief=tk.SOLID)
        self.res_frame.pack(fill=tk.BOTH, expand=True, padx=40, pady=5)

        self.displays = {}
        prayers = [("Subuh", "#ef4444"), ("Terbit", "#f59e0b"), ("Dzuhur", "#0ea5e9"),
                   ("Ashar", "#818cf8"), ("Maghrib", "#ec4899"), ("Isya", "#8b5cf6")]

        for name, color in prayers:
            f = tk.Frame(self.res_frame, bg="#0f172a", pady=1)
            f.pack(fill=tk.X, padx=25, pady=0)
            tk.Label(f, text=name, bg="#0f172a", fg="#94a3b8", font=("Segoe UI", 11)).pack(side=tk.LEFT)
            v = tk.StringVar(value="--:--:--")
            tk.Label(f, textvariable=v, bg="#0f172a", fg=color, font=("Consolas", 18,
"bold")).pack(side=tk.RIGHT)
            self.displays[name] = v

        self.kiblat_lbl = tk.Label(self.root, text="Arah Kiblat: --", bg="#020617", fg="#38bdf8",
font=("Segoe UI", 9, "bold"))
        self.kiblat_lbl.pack(pady=10)

    def process(self):
        try:
            city = self.city_var.get()
            lat, lon = DATA_JAWA[city]["lat"], DATA_JAWA[city]["lon"]
            s_fajr, s_isha = (-20, -18) if "Kemenag" in self.std_var.get() else (-18, -18)
            # Ambil nilai Ihtiyath dari input
            try:
                iht_sec = int(self.ihtiyath_var.get())
            except ValueError:
                iht_sec = 60
                self.ihtiyath_var.set("60")
            res = self.engine.calculate_all(lat, lon, 7, s_fajr, s_isha, iht_sec)
            for name, var in self.displays.items():
                var.set(res[name])
            self.kiblat_lbl.config(text=f"Arah Kiblat {city}: {res['Kiblat']}")

        except Exception as e:
            messagebox.showerror("Error", str(e))

if __name__ == "__main__":
    root = tk.Tk()
    app = FalakAppPrecision(root)
    root.mainloop()
```


PENULIS
KASMUI
• Dosen Kimia, Komputasi, IT, dan AI UNNES, serta Praktisi Ilmu Falak;
• Anggota Majelis Tabligh PDM Kota Semarang dan PWM Jawa Tengah;
• Anggota Tim Pengembang Software KHGT MTT PP Muhammadiyah;
• Website pribadi: https://hisabmu.com/, https://kasmui.cloud/;
• Minat & Hobi: Computer programming.

PRAKTEK PYTHON UNTUK ASTRONOMI ISLAM

MEMADUKAN KODE DAN KOSMOS: Buku ini hadir sebagai panduan komprehensif untuk memahami Ilmu Falak (Astronomi Islam) di era digital digital menggunakan programan Python. Pelajari langkah demi langkah cara menggunakan Python untuk menghitung waktu shalat dengan presisi tinggi, menentukan arah awah Kiblat yang akurat dari berbagai lokasi di dunia, memprediksi penampakan Hilal untuk awal bulan Hijriah, dan mengonversi kalender Hijriah secara dinamis. Melalui latihan praktis dan proyek tan sains, Anda akan belajar membangun aplikasi astronomi Islam Anda sendiri. Sempurna bagi mahasiswa, peneliti, dan peeminat astronomi yang ingin mengintegrasikan sains dan iman.

Bridging Science and Faith through Technology.

Penulis:
KASMUI