Zadanie 1

Autor: Adrian Fabisiewicz

Nr indeksu: 328935

Nr gwiazdy z RA FK5: 168

Rektascencja: $4^h\ 37^m\ 16.316^s$

Deklinacja: $16°\ 33^m\ 16.570^s$

Cel ćwiczenia

Zadanie polegało na wyznaczeniu położenia danej gwiazdy w układzie współrzędnych horyzontalnych dla dwóch miejsc na Ziemi. Pierwsze miejsce znajduje się niedaleko Warszawy, jego przyjętą szerokością geograficzną było 52 stopnie, a długością - 21 stopni. Drugim rozpatrywanym miejscem był równik, miejsce o tej samej długości geograficznej, co pierwszy punkt. Obliczenia zostały wykonane dla całej doby, w godzinnych interwałach. Pierwsze wyliczone położenie gwiazdy miało miejsce 1 lipca 2023 roku o godzinie 02:00, a ostatnie - 2 lipca 2023 roku o godzinie 01:00. Wszystkie wyliczone położenia zwizualizowano na wykresach.

Dane do zadania

Rozpatrywaną gwiazdą była gwiazda nr 168 z RA FK5, której współrzędne w układzie równikowym ekwinokcjalnym, na epokę 2023.5, wynosiły:

- α: $4^h\ 37^m\ 16.316^s$
- δ: $16°\ 33^m\ 16.570^s$

Współrzędne rozpatrywanych miejsc:

- okolice Warszawy: $\varphi = 52^\circ$; $\lambda = 21^\circ$
- równik: $\varphi = 0^\circ$; $\lambda = 21^\circ$

Kolejność wykonywania ćwiczeń

Uwzględnienie strefy czasowej

Na początku należało zauważyć strefy czasowe punktów, biorąc pod uwagę, że znajdują się one w strefie UTC+2. W wyniku tego, w obliczeniach został uwzględniony okres od 1 lipca 2023, od godziny 2^{00} do 2 lipca 2023, godziny 2^{00} czasu UTC. Dzięki temu zabiegowi uzyskano położenia gwiazdy dla całej doby 1 lipca, od godziny 0 do 24, już w czasie UTC+2.

Obliczenie lokalnego czasu gwiazdowego

Kolejnym krokiem było obliczenie lokalnego czasu gwiazdowego dla wybranych miejsc. Aby to zrobić, najpierw należało zapisać interesującą nas datę, używając kalendarza juliańskiego. Zastosowanie daty juliańskiej w obliczeniach astronomicznych między innymi pozwoliło na uniknięcie problemu przy zmianie kalendarza z juliańskiego na gregoriański. Do konwersji użyto uproszczoną wersję algorytmu, dostarczoną wraz z zadaniem, która pozwala na wykonanie poprawnych obliczeń w przedziale od 1 Marca 1900 do 28 lutego 2100. Wykorzystano przy tym poniższą funkcję *julday*.

In [ ]:
```
def julday(y,m,d,h):
    '''
    Simplified Julian Date generator, valid only between
    1 March 1900 to 28 February 2100
    '''
    if m <= 2:
        y = y - 1
        m = m + 12

    jd = np.floor(365.25*(y+4716))+np.floor(30.6001*(m+1))+d+h/24-1537.5
    return jd
```

Następnie, na podstawie otrzymanej daty juliańskiej, można było obliczyć czas średni gwiazdowy Greenwich (GMST). Aby to zrobić, skorzystałem z dostarczonej funkcji *GMST*, która jako argument przyjmowała datę juliańską. Funkcja zwróciła czas średni gwiazdowy Greenwich, w godzinach.

In [ ]:
```
def GMST(jd):
    '''
    calculation of Greenwich Mean Sidereal Time - GMST in hours
    ----------
    jd : TYPE
        julian date
    '''
    T = (jd - 2451545) / 36525
    Tu = jd - 2451545
    g = 280.46061837 + 360.98564736629*(jd - 2451545.0) + 0.000387933*T**2-T**3/387100
    g = (g%360) / 15
    return g
```

Aby teraz obliczyć lokalny czas gwiazdowy (LST), do uzyskanego czasu GMST, dodałem długość geograficzną miejsc obserwacji, wynoszącą 21 stopni, podzieloną przez 15, uzyskując LST wyrażony w godzinach.

In [ ]:
```
lst = gmst + warsaw[1]/15  #warsaw = [52, 21]
```

Obliczenie kąta godzinnego

Mając lokalny czas gwiazdowy oraz daną wcześniej rektascensję gwiazdy, mogłem przejść do obliczenia kąta godzinnego. Wykorzystałem do tego równanie:

$S = \alpha^* + t^*$,
gdzie
S - czas gwiazdowy miejscowy,
α^* - rektascensja,
t^* - kąt godzinny

Po przekształceniu otrzymać można było wzór na kąt godzinny:
$t^* = S - \alpha^*$

A więc:

In [ ]:
```
t_warsaw = lst_warsaw - alfa_h
```

Rozwiązanie trójkąta sferycznego

Kolejnym krokiem zbliżającym do celu zadania, było rozwiązanie trójkąta sferycznego. Aby móc obliczyć wartości wysokości oraz azymutów, należało wyrazić na jego podstawie związek między układem współrzędnych horyzontalnych a układem współrzędnych równikowych godzinnych.
Po przekształceniach można było dojść do poniższego równania:
$$\sin h = \sin\phi \sin\delta + \cos\phi \cos\delta \cos t$$

Obliczyć azymut pozwoliło kolejne równanie:
$$tgAz = \frac{-\cos\delta \sin t}{\cos\phi \sin\delta - \sin\phi \cos\delta \cos t}$$

Pozwoliło to na utworzenie funkcji, obliczających azymut oraz wysokość, na podstawie danej deklinacji, szerokości geograficznej oraz kątu godzinnego.

In [ ]:
```
# function to calculate height
def calculate_h(delta_r, fi_r, t_r):
    h = math.asin(math.sin(delta_r)*math.sin(fi_r) + math.cos(delta_r)*math.cos(fi_r)*
    return h

# function to calculate azimuth
def calculate_a(delta_r, fi_r, t_r):
    a = -math.cos(delta_r)*math.sin(t_r)
    b = math.sin(delta_r)*math.cos(fi_r) - math.cos(delta_r)*math.sin(fi_r)*math.cos(t
    az = math.atan2(a, b)
    return az
```

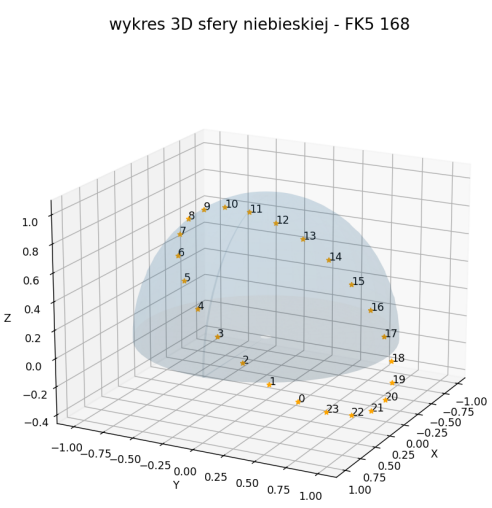
Obliczenie wysokości oraz azymutu gwiazdy - transformacja do współrzędnych równikowych horyzontalnych

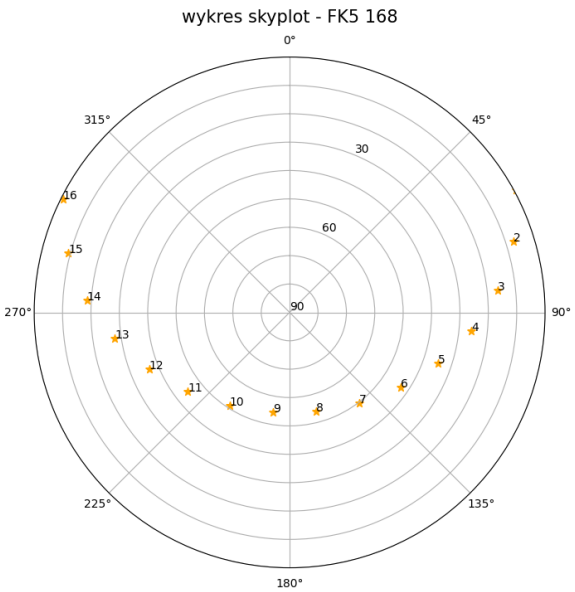
In [ ]:
```
# calculate h for warsaw
h_warsaw_r = calculate_h(delta_r, warsaw_r, t_warsaw_r)

# calculate azimuth for warsaw
a_warsaw_r = calculate_a(delta_r, warsaw_r, t_warsaw_r)

# calculate h for equator
h_equator_r = calculate_h(delta_r, equator_r, t_equator_r)

# calculate azimuth for equator
a_equator_r = calculate_a(delta_r, equator_r, t_equator_r)
```

Wykonanie wizualizacji i przedstawienie wyników

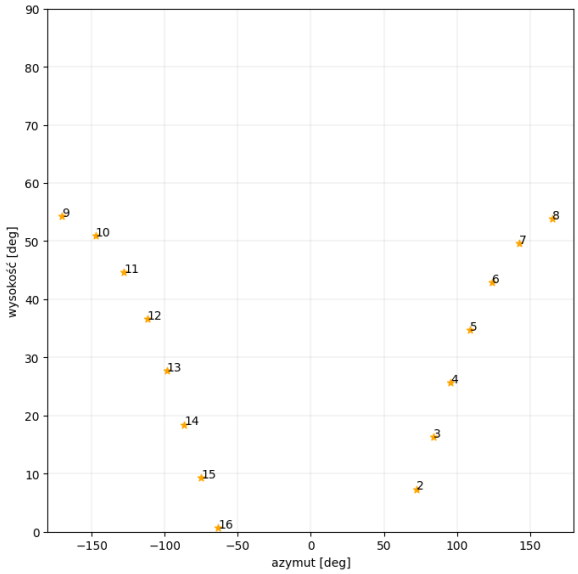
Do uprzednio utworzonego szablonu wykresu z biblioteki Matplotlib, pętla przechodząca przez każdą godzinę, dodała poszczególne punkty. Operacja została powtórzona dla każdego z wykonanych wykresów: 3d sfery niebieskiej, skyplot, panoramy oraz wykresu wysokości gwiazdy w zależności od czasu. Położenie gwiazdy zostało oznaczone symbolem ✯, a poszczególne godziny - odpowiednimi etykietami.

Warszawa

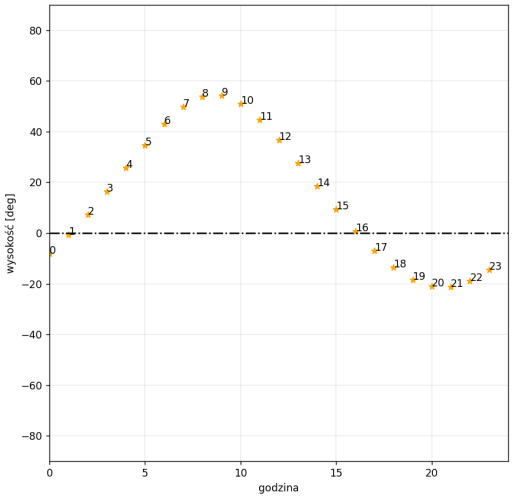




wykres panorama FK5 168



wykres wysokości gwiazdy FK5 168 w zależności od czasu- 1 lipca 2023



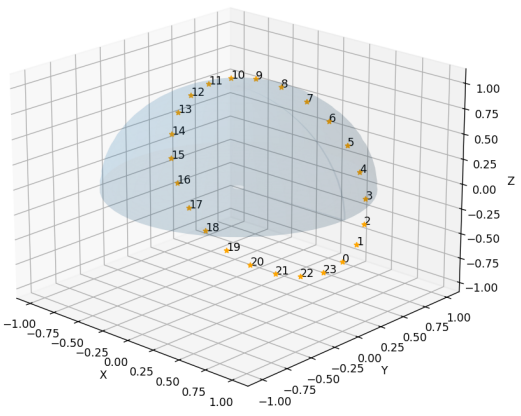
Wnioski

Analiza wykresów dla położenia nr 1 pozwoliła dojść do poniższych wniosków:

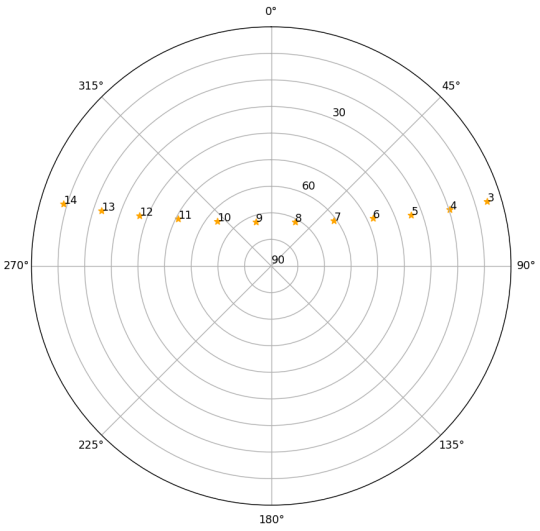
- Gwiazda nr 168 z RA FK5 wzeszła w tym miejscu 1 lipca 2023 r. krótko po godzinie 1^{00} oraz zaszła około godziny 16^{00}
- Najwyżej gwiazda znajdowała się krótko przed godziną 9^{00}. Wysokość wynosiła wtedy blisko 55 stopni.
- Gwiazda nie elongowała.
- Gwiazda nie przeszła przez I wertykał.

Równik

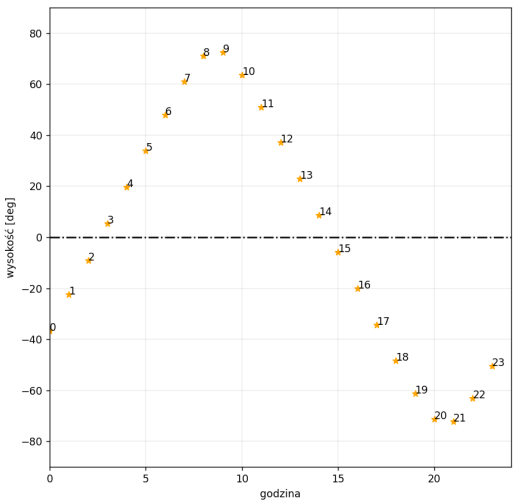
wykres 3D sfery niebieskiej - FK5 168



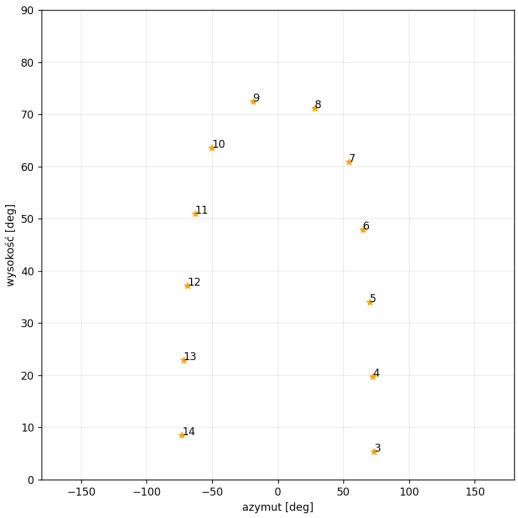
wykres skyplot - FK5 168



wykres wysokości gwiazdy FK5 168 w zależności od czasu- 1 lipca 2023



wykres panorama FK5 168



Wnioski

- Gwiazda FK5 168 w miejscu na równiku o współrzędnych φ = 0◦ oraz λ = 21◦ wzeszła 1 lipca 2023 r. około 2^{40} oraz zaszła około godziny 14^{40}, czyli była widoczna około 12 godzin.
- Najwyżej gwiazda znajdowała się krótko przed godziną 9^{00}. Wysokość wynosiła wtedy blisko 72 stopnie.
- Gwiazda nie elongowała.
- Gwiazda nie przeszła przez I wertykał.
- Gwiazda przez całą dobę była widoczna jedynie po północnej stronie nieba.

Kod źródłowy

In [ ]:
```
import datetime
import math
import numpy as np
import matplotlib.pyplot as plt

def julday(y,m,d,h):
    if m <= 2:
        y = y - 1
        m = m + 12
    jd = np.floor(365.25*(y+4716))+np.floor(30.6001*(m+1))+d+h/24-1537.5;
    return jd

def GMST(jd):
    T = (jd - 2451545) / 36525
    Tu = jd - 2451545
    g = 280.46061837 + 360.98564736629*(jd - 2451545.0) + 0.000387933*T**2-T**3/387100
    g = (g%360) / 15
    return g

def dms2rad(dms):
    d = dms[0]
    m = dms[1]
    s = dms[2]
```

```
    deg = d+m/60+s/3600
    rad = np.deg2rad(deg)
    return rad

def hms2rad(dms):
    d = dms[0]
    m = dms[1]
    s = dms[2]

    deg = d+m/60+s/3600
    rad = np.deg2rad(deg*15)
    return rad

def hms2h(dms):
    return dms[0] + dms[1]/60 + dms[2]/3600

def calculate_h(delta_r, fi_r, t_r):
    h = math.asin(math.sin(delta_r)*math.sin(fi_r) + math.cos(delta_r)*math.cos(fi_r)*
    return h

def calculate_a(delta_r, fi_r, t_r):
    a = -math.cos(delta_r)*math.sin(t_r)
    b = math.sin(delta_r)*math.cos(fi_r) - math.cos(delta_r)*math.sin(fi_r)*math.cos(t
    az = math.atan2(a, b)
    return az

def add_point(a, h, r, ax):
    x = r * np.sin(a) * np.cos(h)
    y = r * np.cos(a) * np.cos(h)
    z = r * np.sin(h)

    ax.scatter(x, y, z, marker='*', color="orange")
    ax.text(x, y, z, hr)

# right ascension
alfa = [4, 37, 16.316]
alfa_h = hms2h(alfa)
alfa_r = hms2rad(alfa)

# declination
delta = [16, 33, 16.570]
delta_r = dms2rad(delta)

# warsaw coordinates
warsaw = [52, 21]
warsaw_r = math.radians(warsaw[0])

# equator coordinates
equator = [0, 21]
equator_r = math.radians(equator[0])

#WYKRES 3D
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(projection = '3d')
r = 1
u, v = np.mgrid[0:(2 * np.pi+0.1):0.1, 0:np.pi:0.1]
x = np.cos(u) * np.sin(v)
y = np.sin(u) * np.sin(v)
```

```
z = np.cos(v)
z[z<0] = 0
ax.plot_surface(x,y,z, alpha = 0.1)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('wykres 3D sfery niebieskiej - FK5 168', pad = 30, fontsize = 15)
ax.grid(color = 'grey', linestyle = '-', linewidth = 0.25, alpha = 0.8)

# WYKRES SKYPLOT
# fig = plt.figure(figsize = (8,8))
# ax = fig.add_subplot(polar = True)
# ax.set_theta_zero_location('N')
# ax.set_theta_direction(-1)
# ax.set_yticks(range(0, 90+10, 10))
# yLabel = ['90', '', '', '60', '', '', '30', '', '', '']
# ax.set_yticklabels(yLabel)
# ax.set_rlim(0,90)
# ax.set_title('wykres skyplot - FK5 168', pad = 30, fontsize = 15)

#WYKRES ZALEZNOSCI WYSOKOSCI OD CZASU
# fig = plt.figure(figsize = (8,8))
# ax = fig.add_subplot()
# ax.set_xlabel('godzina')
# ax.set_ylabel('wysokość [deg]')
# ax.set_ylim(-90,90)
# ax.set_xlim(0,24)
# ax.axhline(0, linestyle = '-.', color = 'black')
# ax.set_title('wykres wysokości gwiazdy FK5 168 w zależności od czasu- 1 lipca 2023',
# ax.grid(color = 'grey', linestyle = '-', linewidth = 0.25, alpha = 0.5)


# WYKRES PANORAMA
# fig = plt.figure(figsize = (8,8))
# ax = fig.add_subplot()
# ax.set_xlabel('azymut [deg]')
# ax.set_ylabel('wysokość [deg]')
# ax.set_ylim(0,90)
# ax.set_xlim(-180,180)
# ax.set_title('wykres panorama FK5 168', pad = 30, fontsize = 15)
# ax.grid(color = 'grey', linestyle = '-', linewidth = 0.25, alpha = 0.5)

for hr in range(2,26):
    if hr >= 24:
        hr = hr - 24
        gd = datetime.datetime(2023, 7, 2, hr)
    else:
        gd = datetime.datetime(2023, 7, 1, hr)
    # julian date
    jd = julday(gd.year, gd.month, gd.day, gd.hour)
    # greenwich mean sidereal time
    gmst = GMST(jd)

    # local sidereal time
    lst_warsaw = gmst + warsaw[1]/15

    # hour angle
    t_warsaw = lst_warsaw - alfa_h
    t_warsaw_r = math.radians(t_warsaw * 15)
```

```
    t_equator_r = t_warsaw_r


    # calculate h and a for warsaw
    h_warsaw_r = calculate_h(delta_r, warsaw_r, t_warsaw_r)
    a_warsaw_r = calculate_a(delta_r, warsaw_r, t_warsaw_r)

    # calculate h and a for equator
    h_equator_r = calculate_h(delta_r, equator_r, t_equator_r)
    a_equator_r = calculate_a(delta_r, equator_r, t_equator_r)

# adding points to SPHERE
    add_point(a_equator_r, h_equator_r, r, ax)
    # add_point(a_warsaw_r, h_warsaw_r, r, ax)

# adding points to SKYPLOT
    # if (np.rad2deg(h_warsaw_r) >= 0):
    #     ax.scatter(a_warsaw_r, 90-np.rad2deg(h_warsaw_r), marker = "*", color = 'ora
    #     ax.text(a_warsaw_r, 90-np.rad2deg(h_warsaw_r), hr, color = 'black')
    #
    # if (np.rad2deg(h_equator_r) >= 0):
    #     ax.scatter(a_equator_r, 90-np.rad2deg(h_equator_r), marker = '*', color = 'o
    #     ax.text(a_equator_r, 90-np.rad2deg(h_equator_r), hr)

# adding points to WYKRES ZALEZNOSCI WYSOKOSCI OD CZASU
    # ax.scatter(hr, np.rad2deg(h_warsaw_r), marker = "*", color = 'orange')
    # ax.text(hr, np.rad2deg(h_warsaw_r), hr)

    # ax.scatter(hr, np.rad2deg(h_equator_r), marker = '*', color = 'orange')
    # ax.text(hr, np.rad2deg(h_equator_r), hr)

# adding points to PANORAMA
    # if (np.rad2deg(h_warsaw_r) >= 0):
    #     ax.scatter(np.rad2deg(a_warsaw_r), np.rad2deg(h_warsaw_r), marker = "*", col
    #     ax.text(np.rad2deg(a_warsaw_r), np.rad2deg(h_warsaw_r), hr)

    # if (np.rad2deg(h_equator_r) >= 0):
    #     ax.scatter(np.rad2deg(a_equator_r), np.rad2deg(h_equator_r), marker = '*', c
    #     ax.text(np.rad2deg(a_equator_r), np.rad2deg(h_equator_r), hr)

# plt.show()
```