Neal Ma
YSPA Problem Set #1
Dr. Michael Faison
6 July 2018

The code for each of the problems can be found at the bottom of this document. The raw python files will also be attached.

Problem 1. Astronomy Basics

a) The Spring Equinox occurs when the Sun's path intersects the Equator and Prime Meridian, so the declination is 00°00'00" and the right ascension is 00h 00m 00s.
b) The Summer Solstice occurs when the Sun reaches its highest declination of the year, 23°30'00" and this occurs at a right ascension of 06h 00m 00s.
c) To determine the altitude of the Sun above the horizon as it transits on the Summer Solstice, I used the equation:

$$90 - \text{Latitude} + \text{Declination} = \text{Altitude}$$

With a latitude of 41.3° N in New Haven and the Sun's declination of 23.5° N on the Summer Solstice, the altitude of the Sun would be 72.2° N.

d) The right ascension and declination of an object directly opposite of the Sun in the Celestial Sphere ion the evening of the Summer Solstice would be the RA and Dec of the Sun + half a rotation (Dec * -1 and RA ± 12 hours). This would mean a RA of 18 hours and a Dec of -23.5° N of 23.5° S. Because the Sun transits at around noon and this object is 12 hours away from the Sun, it would transit at about midnight. It's maximum altitude (determined the same was as that of the Sun was earlier) would be 25.2° N.
e) Of the three stars, Deneb is both the most North and the most East as it has the greatest declination (indicating that it is the most North) and the greatest right ascension (indicating that it is the most East).
f) I chose to attempt to find the latitude at which Vega is circumpolar intuitively rather than using an equation. I first imagined the Celestial Sphere with Polaris at the top and Vega tracing out a circular orbit around Polaris. I figured that the latitude that would allow Vega to be circumpolar would have to be tangent to the path that Vega followed. This transformed into a simple calculation of (90 – Dec) and ended up being 51.2° N. To find the latitude at which Vega would never rise I imagined the same scenario but I had Vega in the opposite hemisphere as my plane of reference. This would lead to the calculation (Dec – 90) to find the latitude at which Vega reaches the horizon, but never crosses above it. This was -51.2° N or 51.2° S.
g) Altair is at nearly a right ascension of 20 hours. To find when Altair would be on the Local Meridian at midnight, the Sun must have a right ascension of ± 12 hours of that of Altair. In this case that would be a right ascension of 8 hours. The Summer Solstice

(when the right ascension of the Sun is 6 hours) occurs in June. This means we need the Sun to be 2 hours farther in its cycle. 2 hours corresponds to 1 month (24 hours/12 months), so the point where Altair would transit at midnight would be approximately one month after June. Altair would be on the Local Meridian at midnight in July.

Problem 2. Introduction to Stellarium

March 20, 2018:	June 21, 2018:	July 8, 2018:
Sunrise:	Sunrise:	Sunrise:
Time:0 5:56:32	Time: 04:20:24	Time: 04:28:08
Az: 89°37’10.1”	Az: 57°26’45.0”	RA: 07h 09m 07.2s
RA: 23h 58m 16.77s	RA: 05h 58m 45.84s	Dec: 22°29’05.3”
Dec: -00°11’21.5”	Dec: -23°26’05.8”	Noon:
Noon:	Noon:	Az: 182°14’37.8”
Az: 180°19’36.5”	Az: 184°47’54.1”	Alt: 71°06’09.5”
Alt: 48°43’09.7”	Alt: 72°04’51.4”	RA: 07h 10m 23.92s
RA: 23h 59m 11.57s	RA: 06h 00m 05.04s	Dec: 22°27’00.3”
Dec: -00°05’22.3”	Dec: 23°26’10.5”	Sunset:
Sunset:	Sunset:	Time: 19:25:14
Time: 18:02:28	Time: 19:26:55	Az: 301°01’59.3”
Az: 270°38’45.0”	Az: 302°33’08.5”	RA: 07h 11m 39.48s
RA: 00h 00m 06.21s	RA: 06h 01m 22.04s	Dec: 22°24’48.2”
Dec: 00°00’35.9”	Dec: 23°26’05.6”	

At noon on July 8, 2018, the azimuth of the Sun is 182°15’ E and the altitude of the Sun is 71°06’ N. The Sun is not exactly on the Meridian at noon EST on July 8th. This is expected as it the Sun only transits the Meridian at noon 4 times a year, and this date is not one of the 4 dates. The right ascension and declination of the Sun at this point in time are 07h 10m 23.92s E and 22°27’00.3” N, respectively. This location does make sense as the declination is positive and so the Sun appears above the celestial equator. The Sun is in the constellation Gemini on July 8, 2018. The sunset that day is at 19:25 EST.

The difference in times of the sunset at various points in the year can be attributed to the changing right ascension and declination of the Sun throughout the year. With a lower declination, less of the path of the Sun will be above the horizon (shorter days) while a higher declination means that more of the path of the Sun will be above the horizon (longer days).

	Altitude	Azimuth	Right Ascension	Declination
Puerto Montt	26°08’35.9”	359°12’42.1”	7h 10m 23.92s	22°27’10.8”

The right ascension and declination of the Sun stay (relatively) constant as they are positions fixed on the celestial sphere while altitude and azimuth significantly change because they are relative positions in the sky that are dependent on the observer’s location.

Problem 3. Local Sidereal Time and Local Mean Solar Time

I chose to approach this problem by looking at the original definition of Sidereal and Solar Time. Sidereal Time is dependent on the Earth's orientation to the first point of Aries, or where the equator intersects the prime meridian at 0.0 hours. Solar Time is determined by Earth's orientation to the Sun. Because of these two facts, it could be concluded that the point at which Local Sidereal Time equals Local Mean Solar Time is the point where the Earth, the Sun, and the first point of Aries align (in that order too). This only occurs at the Autumnal Equinox where the first point of Aries and Earth directly oppose each other across the sun, so the Autumnal Equinox is where Local Sidereal Time equals Local Mean Solar Time.

Problem 4. Julian Day to Local Sidereal Time

This function took in two parameters: longitude and the Julian date. A preset reference date was subtracted from the given Julian date and multiplied by a conversion factor. I then took only the decimal of that number as it represented how far in the sidereal day had passed. I then added on the reference time from the reference date. I passed this into a different function along with the longitude. From this function, I calculated the time offset due to the longitude. I then returned an integer array containing the universal time in hours, minutes, and seconds. Finally, I printed out the time using leading zero formatting.

Problem 5. Asteroid Tracking

To calculate the hour angle, I found the difference between the Local Sidereal Time and the right ascension of the asteroid at that time. This came out to be -2h 27m 59s. To find what time it would be in transit I simply added the hour angle and the current time (4:00 UTC) to find that it would transit at about 1:32 am. I used the altitude equation from problem 1 of this problem set to determine that it would appear about 63.2° above the horizon. This asteroid would be best observed in the 12:45-2 am timeslot.

```
#Problem Set 2 - Problem 4


#1  solar  day  equals  1.0027379 sidereal days

#julian day 0 started at noon

#reference date: January 1, 2016 - 2457388.5

#reference time: 06:40:21


def find_LST(longitude, julian_days):

    reference_date = 2457388.5 #a reference date to reduce accumulated error

    reference_time = [6, 40, 21] #a reference time that corresponds with the
refernce date

    julian_days = (julian_days - reference_date)* 1.0027379 #tells how many
Sidereal days have passed since the reference date

    julian_days -= int(julian_days) #tells what percentage of a siderial date
the given julian date is

    reference_hours = reference_time[0] + reference_time[1] / 60.0 +
reference_time[2] / 3600.0 #totals the number of hours needed to add because
of the reference time

    UT = dec_hours_to_time(julian_days * 24 + reference_hours, longitude)
#returns a time based off of the reference time and the Julian date passed
into the function

    return "%02i:%02i:%02i LST" % ((UT[0]) % 24, UT[1], UT[2]) #prints out
the LST in a nice format




def dec_hours_to_time(time, longitude):

    offset = longitude / 15 #finds the offset time needed due to longitudal
differences

    time += offset #applies the offset time

    UT = []

    UT.append(int(time)) #adds the hours place of the current time

    time -= int(time)

    if UT[0] < 0: #ensures the hours place is greater than 0 but less than 24

        UT[0] += 24

    UT.append(int(time * 60)) #adds the minutes place of the current time

    time = (time * 60.0) - UT[1]

    UT.append(time * 60.0) #adds the seconds place of the current time

    return UT
```

```
print find_LST(-72.9279, 2458310.666673)

long = float(input("What longitude: "))
j_days = float(input("What Julian day: "))
print find_LST(long, j_days)
```