Astro 310 HW2

September 14, 2024

1 ASTR 310 HW 2

For this assignment, your solutions should consist of a PDF file listing your answers and program code and showing your interaction with your program. Please follow the template below; clean up your code so that it runs straight through from the beginning and does not include extraneous material. In other words, if you try some code and it doesn't work, delete the bad code or at the very least comment it out so that the TA knows not to grade it. Otherwise you may find unexpected points taken off. Use Jupyter's export function to create a PDF file and submit that. If you don't have LaTeX installed (what are you waiting for?) you may have to create a html file and use your browser's print functionality to convert the html into a PDF.

1.0.1 1. Python code fragments, part 1

State in words what each of these Python code fragments does. As a reminder, you should do this yourself without using a code explainer or other AI tool. Feel free to execute the code and see what it does. [10 pts; 2 each]

a.)

```
x_comp = 34.897
y_comp = x_comp**2 + 12.
print("x_comp = ", x_comp, " y_comp = ", y_comp)
```

This code performs a calculation, taking `x_comp` as an input, squaring it and adding 12 and storing the result as `y_comp`, and then printing the results

b.)

```
tmp = A
A = B
B = tmp
```

This code swaps the variables `A` and `B`

c.)

```
from cmath import log
z = -4 + 2j
print((log(z)).imag)
```

This code creates a complex number `z`, then takes the natural log of `z` and prints out the imaginary component.

d.)

```
q = s = 3.3
q = s * q
```

This code assigns `q` and `s` the value 3.3. Then it reassigns `q` to `s` $*$ `q`.

e.)

```
s = float(input("enter a value: "))
print((s % 2) == 1)
```

This code takes an input and casts it to a float and assigns it to `s`. Then it prints out whether or not `s` is an odd number.

1.0.2 2. Python code fragments, part 2

Explain why each of the following broken Python statements is broken. Again, don't use a code explainer or other AI tool. Feel free to try executing the code and see what happens, but your answer will need more detail than just repeating the error messages. You'll have to give enough specifics that your reader will know how to fix the code. [10 pts; 2 each]

a.) `R = ((x-3.231)**2 + (y-7.325))**2)**0.5`

There are 2 problems with this code. One is that there is an extra unmatched parenthesis after `**2`, and the other is that the variables `x` and `y` were never defined. To fix it, remove the parenthesis and define `x` and `y`.

b.) `print("the result is " + 17.1)`

The problem with this code is that you cannot concatenate a float to a string, i.e. you cannot add together a string and a float. To fix it, convert 17.1 to a string first.

c.) `result#2 = 6.4*sin(13.8*pi*omega) - 4.8`

There are 3 problems with this code. You cannot include a `#` in a variable name, this symbol will turn everything after it into a comment. The `sin` function doesn't work because it was not imported, and the variable `omega` was not defined. To fix these problems, remove the `#`, import `math.sin`, and define `omega`.

d.)

```
for i in range(13):
j = i**2 + 1
print(i, j)
```

This code is not indented properly, `j = i**2 + 1` should be indented, and it is unclear whether or not the following line should also be indented, although it probably should be.

e.) `length_in_inches = 2.54 * input("enter length in cm: ")`

This code does not work because the `input` function returns a string, and you cannot multiply by a string. To fix it, cast the input to an int.

1.0.3 3. Writing a Python program that works with user input and calculates something

In Big Bang cosmology the expansion of the Universe changes the relationship between the apparent angular size of a distant object and its real physical size. Normally in a non-expanding spacetime, a distant object of physical size L at a distance d subtends an angle $\theta \approx L/d$ (in radians; approximate because of the small angle approximation). In an expanding universe this relationship does not hold, but we can define a quantity called the "angular diameter distance" d_A such that

$$\theta = L/d_A$$

or $L = d_A\ \theta$. The angular diameter distance is a function of cosmological parameters such as the rate of expansion and the matter content of the Universe as well as the redshift (a measure of the amount that the wavelength of light gets stretched out by the expansion). In this problem we'll fix the value of $\theta = 1''$, so d_A can be thought of as a conversion factor between angle in arcseconds ($''$) and size in kiloparsecs (kpc $= 3.0857 \times 10^{21}$ cm).

a.) Write a Python program that obtains from the user the value of the Hubble constant (units $km\ s^{-1}\ Mpc^{-1}$), the matter density parameter Ω_m (dimensionless), and a redshift z (also dimensionless). Your code should then compute and print the conversion between arcseconds and kpc, i.e. the angular diameter distance, for a flat cosmology at this redshift. Have your program print the result with a friendly message such as "The conversion factor is X.XXX kpc/arcsec."

Use vanilla Python for this problem, no fancy stuff like astropy.units. (We'll get to that before long.)

An approximate analytical fit to the angular diameter distance for a flat cosmology is given by Pen, U.-L. (1999, ApJS, 120, 49):

$$d_A = \frac{1}{1+z}\frac{c}{H_0}\left[\eta(1,\Omega_m) - \eta\left(\frac{1}{1+z},\Omega_m\right)\right]$$

where

$$\eta(a,\Omega_m) \equiv 2\sqrt{s^3+1}\left(\frac{1}{a^4} - 0.1540\frac{s}{a^3} + 0.4304\frac{s^2}{a^2} + 0.19097\frac{s^3}{a} + 0.066941 s^4\right)^{-1/8} \text{ and}$$

$$s^3 \equiv \frac{1-\Omega_m}{\Omega_m}.$$

Watch the unit conversions. Recall that c has units of length/time and H_0 has units of 1/time, so d_A will come out with native units of length/radian and you can convert that to kpc/arcsec using the definition of a radian. [10 pts]

[90]:
```
from scipy import constants as const
import math


def conversion_factor(H, m, z):

    H = H * 3.24078e-20 # convert to 1/s
```

```
    def n(a, o):
        s = ((1 - o) / o) ** (1/3)
        return 2 * math.sqrt(s**3 + 1) * ((1/a**4) - (0.1540 * s/a**3) + (0.
↪4304*s**2/a**2) + (0.19097*s**3/a) + (0.066941*s**4))**(-1/8)


    da = 1/(1+z) * (2.998e8 / H) * (n(1,m) - n((1/(1+z)),m))
    da # m/radian

    factor = da / 206265 / 3.086e+19 # convert to kpc/arcsecond
    print(f"The conversion factor is {factor} kpc/arcsec")
    return factor
```

[91]:
```
H = float(input("Hubble Constant (km/s/Mpc)= "))
m = float(input("Matter Density = "))
z = float(input("Redshift = "))
conversion_factor(H, m, z)
```

```
The conversion factor is 6.035953865591871 kpc/arcsec
```

[91]:
```
6.035953865591871
```

b.) Run your program for $H_0 = 72$ km s^{-1} Mpc^{-1} and $\Omega_m = 0.26$, for redshifts 0.5 and 1.0. You should get 6.036 kpc/arcsec and 7.990 kpc/arcsec as answers (both less than 0.4% in error). See Reading 4 if you're not sure how to define a function or to write a loop or to use a standalone file so that you can re-run your own code without copying and pasting it. [10 pts]

[92]:
```
conversion_factor(72, 0.26, 0.5)
```

```
The conversion factor is 6.035953865591871 kpc/arcsec
```

[92]:
```
6.035953865591871
```

[93]:
```
conversion_factor(72, 0.26, 1)
```

```
The conversion factor is 7.989504846309262 kpc/arcsec
```

[93]:
```
7.989504846309262
```

c.) Try the same cosmological parameters for redshifts of 2.0 and 4.0. Notice that the conversion factor increases with redshift for a while and then decreases! In other words, objects that are more and more distant appear smaller and smaller, and then at some point they begin to look larger again. Weird. If you like you can plot the conversion factor as a function of redshift, but that's not required at this point. [5 pts]

[94]:
```
conversion_factor(72, 0.26, 2)
```

```
The conversion factor is 8.432661979423127 kpc/arcsec
```

[94]: 8.432661979423127

[95]:
```
conversion_factor(72, 0.26, 4)
```

```
The conversion factor is 7.064349074935276 kpc/arcsec
```

[95]: 7.064349074935276

[96]:
```
import matplotlib.pyplot as plt
import seaborn
import pandas as pd
import numpy as np

x = np.linspace(0, 10, 1000)
y = conversion_factor(72, 0.26, x)

df = pd.DataFrame({"x":x, "y":y})
seaborn.lineplot(x='x',y='y',data=df)
plt.xlabel("Redshift")
plt.ylabel("Conversion Factor")
```

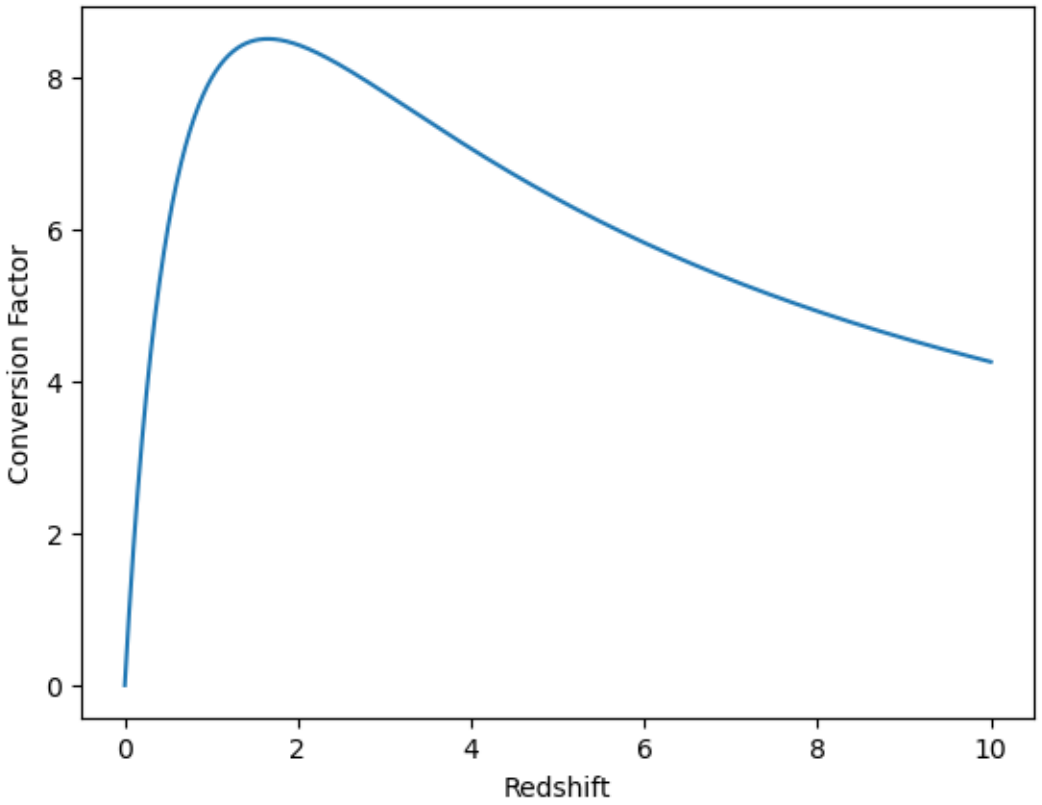
```
The conversion factor is [0.         0.19980989 0.39497632 0.58562464 0.7718757
0.95384605
 1.13164815 1.30539056 1.47517812 1.64111216 1.80329061 1.96180818
 2.11675652 2.26822431 2.41629745 2.56105912 2.70258992 2.84096801
 2.97626915 3.10856685 3.23793243 3.36443511 3.48814212 3.60911873
 3.72742838 3.84313269 3.95629157 4.06696329 4.17520448 4.28107026
 4.38461425 4.48588863 4.58494421 4.68183044 4.77659548 4.86928626
 4.95994847 5.04862665 5.1353642  5.22020341 5.30318554 5.38435078
 5.46373836 5.54138652 5.61733256 5.69161288 5.764263   5.83531757
 5.90481041 5.97277454 6.0392422  6.10424483 6.16781318 6.22997724
 6.29076631 6.35020903 6.40833333 6.46516655 6.52073535 6.57506582
 6.62818343 6.68011308 6.73087911 6.78050529 6.82901488 6.87643062
 6.9227747  6.96806888 7.01233437 7.05559196 7.09786196 7.13916423
 7.17951821 7.2189429  7.25745688 7.29507834 7.33182507 7.36771448
 7.40276359 7.43698906 7.4704072  7.50303396 7.53488496 7.56597547
 7.59632044 7.62593453 7.65483204 7.68302701 7.71053317 7.73736395
 7.76353251 7.78905174 7.81393425 7.83819238 7.86183824 7.88488366
 7.90734025 7.92921935 7.9505321  7.97128939 7.99150188 8.01118003
 8.03033409 8.04897407 8.0671098  8.08475091 8.10190683 8.1185868
 8.13479987 8.15055491 8.16586061 8.18072549 8.1951579  8.20916601
 8.22275785 8.23594126 8.24872395 8.26111347 8.2731172  8.28474242
 8.29599621 8.30688556 8.31741728 8.32759809 8.33743454 8.34693308
 8.35610001 8.36494153 8.37346369 8.38167246 8.38957368 8.39717305
 8.40447619 8.41148862 8.41821572 8.4246628  8.43083503 8.43673753
 8.44237528 8.44775319 8.45287606 8.45774862 8.46237548 8.46676119
 8.4709102  8.47482689 8.47851553 8.48198034 8.48522545 8.4882549
 8.49107268 8.49368267 8.49608871 8.49829456 8.50030391 8.50212036
```

8.50374748 8.50518875 8.50644759 8.50752736 8.50843136 8.50916282
8.50972493 8.51012079 8.51035347 8.51042598 8.51034127 8.51010223
8.50971172 8.50917252 8.50848738 8.50765899 8.50668999 8.50558299
8.50434052 8.50296511 8.5014592 8.49982521 8.49806552 8.49618245
8.49417829 8.4920553 8.48981567 8.48746158 8.48499515 8.48241848
8.47973363 8.47694261 8.4740474 8.47104996 8.4679522 8.464756
8.46146321 8.45807564 8.45459507 8.45102327 8.44736195 8.4436128
8.43977748 8.43585764 8.43185487 8.42777075 8.42360683 8.41936464
8.41504568 8.41065141 8.40618327 8.4016427 8.39703109 8.39234981
8.38760021 8.38278361 8.37790133 8.37295463 8.36794479 8.36287304
8.35774059 8.35254864 8.34729837 8.34199093 8.33662746 8.33120908
8.32573688 8.32021195 8.31463535 8.30900811 8.30333127 8.29760583
8.29183279 8.28601313 8.28014779 8.27423772 8.26828385 8.2622871
8.25624835 8.25016849 8.24404839 8.23788889 8.23169083 8.22545505
8.21918234 8.21287351 8.20652934 8.2001506 8.19373804 8.18729242
8.18081447 8.1743049 8.16776442 8.16119374 8.15459353 8.14796448
8.14130724 8.13462247 8.12791081 8.12117289 8.11440933 8.10762074
8.10080772 8.09397086 8.08711074 8.08022794 8.07332301 8.06639652
8.05944899 8.05248097 8.04549298 8.03848554 8.03145916 8.02441434
8.01735158 8.01027134 8.00317412 7.99606039 7.98893059 7.98178519
7.97462463 7.96744935 7.96025978 7.95305635 7.94583948 7.93860957
7.93136703 7.92411225 7.91684563 7.90956756 7.90227841 7.89497855
7.88766835 7.88034817 7.87301836 7.86567928 7.85833127 7.85097465
7.84360977 7.83623695 7.82885652 7.82146878 7.81407405 7.80667264
7.79926484 7.79185095 7.78443127 7.77700607 7.76957564 7.76214026
7.75470019 7.74725571 7.73980708 7.73235456 7.72489841 7.71743886
7.70997618 7.7025106 7.69504237 7.68757171 7.68009886 7.67262404
7.66514748 7.6576694 7.65019001 7.64270953 7.63522816 7.62774611
7.62026358 7.61278077 7.60529788 7.5978151 7.59033261 7.58285061
7.57536927 7.56788877 7.5604093 7.55293102 7.5454541 7.53797872
7.53050504 7.52303322 7.51556342 7.5080958 7.50063051 7.4931677
7.48570752 7.47825012 7.47079565 7.46334424 7.45589603 7.44845116
7.44100976 7.43357198 7.42613793 7.41870774 7.41128154 7.40385946
7.3964416 7.38902811 7.38161908 7.37421463 7.36681488 7.35941995
7.35202992 7.34464493 7.33726506 7.32989042 7.32252112 7.31515725
7.30779892 7.30044621 7.29309923 7.28575806 7.27842279 7.27109352
7.26377034 7.25645332 7.24914256 7.24183813 7.23454013 7.22724862
7.21996369 7.21268541 7.20541386 7.19814912 7.19089125 7.18364033
7.17639643 7.16915962 7.16192996 7.15470752 7.14749237 7.14028457
7.13308418 7.12589126 7.11870587 7.11152808 7.10435794 7.0971955
7.09004082 7.08289396 7.07575497 7.0686239 7.06150079 7.05438572
7.04727871 7.04017982 7.03308909 7.02600658 7.01893232 7.01186637
7.00480875 6.99775953 6.99071873 6.98368639 6.97666256 6.96964728
6.96264058 6.9556425 6.94865308 6.94167235 6.93470034 6.92773709
6.92078263 6.91383699 6.90690022 6.89997232 6.89305334 6.88614331
6.87924224 6.87235018 6.86546714 6.85859316 6.85172825 6.84487245
6.83802578 6.83118825 6.82435991 6.81754075 6.81073082 6.80393012
6.79713869 6.79035653 6.78358367 6.77682013 6.77006593 6.76332107

6.75658559 6.7498595 6.7431428 6.73643553 6.72973769 6.72304929
6.71637036 6.7097009 6.70304093 6.69639046 6.6897495 6.68311806
6.67649616 6.66988381 6.66328101 6.65668777 6.65010411 6.64353003
6.63696555 6.63041066 6.62386538 6.61732972 6.61080368 6.60428726
6.59778048 6.59128334 6.58479584 6.578318   6.5718498 6.56539127
6.5589424 6.5525032 6.54607366 6.5396538 6.53324361 6.5268431
6.52045227 6.51407112 6.50769965 6.50133787 6.49498577 6.48864335
6.48231062 6.47598757 6.46967421 6.46337053 6.45707653 6.45079221
6.44451757 6.43825261 6.43199732 6.42575171 6.41951577 6.4132895
6.40707289 6.40086595 6.39466866 6.38848103 6.38230304 6.37613471
6.36997602 6.36382696 6.35768754 6.35155775 6.34543758 6.33932702
6.33322608 6.32713475 6.32105302 6.31498088 6.30891832 6.30286535
6.29682196 6.29078813 6.28476386 6.27874915 6.27274398 6.26674836
6.26076226 6.25478568 6.24881863 6.24286108 6.23691302 6.23097446
6.22504538 6.21912578 6.21321563 6.20731495 6.20142371 6.1955419
6.18966953 6.18380657 6.17795302 6.17210887 6.1662741 6.16044872
6.15463271 6.14882605 6.14302874 6.13724077 6.13146212 6.12569279
6.11993277 6.11418204 6.1084406 6.10270843 6.09698551 6.09127185
6.08556743 6.07987223 6.07418625 6.06850948 6.06284189 6.05718349
6.05153426 6.04589418 6.04026324 6.03464144 6.02902876 6.02342519
6.01783071 6.01224532 6.00666899 6.00110172 5.9955435 5.98999432
5.98445415 5.97892298 5.97340082 5.96788763 5.96238341 5.95688815
5.95140183 5.94592444 5.94045596 5.93499638 5.9295457 5.92410389
5.91867094 5.91324684 5.90783157 5.90242513 5.89702749 5.89163865
5.88625859 5.88088729 5.87552475 5.87017095 5.86482587 5.8594895
5.85416183 5.84884284 5.84353253 5.83823086 5.83293784 5.82765345
5.82237766 5.81711048 5.81185188 5.80660185 5.80136038 5.79612745
5.79090304 5.78568715 5.78047976 5.77528085 5.77009041 5.76490842
5.75973488 5.75456976 5.74941306 5.74426475 5.73912483 5.73399327
5.72887006 5.7237552 5.71864866 5.71355043 5.7084605 5.70337884
5.69830546 5.69324032 5.68818342 5.68313474 5.67809427 5.67306199
5.66803789 5.66302196 5.65801417 5.65301451 5.64802298 5.64303955
5.6380642 5.63309694 5.62813773 5.62318657 5.61824344 5.61330832
5.60838121 5.60346208 5.59855092 5.59364772 5.58875246 5.58386513
5.57898571 5.57411419 5.56925055 5.56439478 5.55954686 5.55470679
5.54987453 5.54505008 5.54023343 5.53542456 5.53062345 5.5258301
5.52104447 5.51626657 5.51149638 5.50673387 5.50197905 5.49723188
5.49249236 5.48776048 5.48303621 5.47831954 5.47361047 5.46890896
5.46421502 5.45952862 5.45484975 5.4501784 5.44551455 5.44085818
5.43620929 5.43156785 5.42693385 5.42230729 5.41768813 5.41307638
5.40847201 5.40387501 5.39928537 5.39470307 5.39012809 5.38556043
5.38100006 5.37644698 5.37190117 5.36736261 5.36283129 5.3583072
5.35379032 5.34928064 5.34477814 5.34028281 5.33579464 5.3313136
5.3268397 5.3223729 5.3179132 5.31346059 5.30901505 5.30457656
5.30014511 5.29572069 5.29130329 5.28689288 5.28248946 5.27809301
5.27370351 5.26932096 5.26494534 5.26057664 5.25621484 5.25185992
5.24751188 5.2431707 5.23883636 5.23450886 5.23018817 5.22587429
5.2215672 5.21726689 5.21297334 5.20868654 5.20440648 5.20013314

```
5.19586651 5.19160657 5.18735332 5.18310673 5.1788668  5.17463351
5.17040685 5.1661868  5.16197335 5.15776649 5.1535662  5.14937248
5.1451853  5.14100465 5.13683053 5.13266291 5.12850179 5.12434715
5.12019897 5.11605725 5.11192197 5.10779312 5.10367069 5.09955465
5.09544501 5.09134174 5.08724483 5.08315427 5.07907005 5.07499215
5.07092057 5.06685528 5.06279628 5.05874354 5.05469707 5.05065685
5.04662286 5.04259508 5.03857352 5.03455815 5.03054897 5.02654595
5.0225491  5.01855838 5.0145738  5.01059534 5.00662299 5.00265673
4.99869655 4.99474245 4.9907944  4.98685239 4.98291642 4.97898646
4.97506252 4.97114457 4.9672326  4.9633266  4.95942656 4.95553247
4.95164431 4.94776208 4.94388575 4.94001532 4.93615077 4.9322921
4.92843929 4.92459233 4.92075121 4.91691591 4.91308642 4.90926274
4.90544485 4.90163273 4.89782638 4.89402578 4.89023092 4.8864418
4.88265839 4.87888069 4.87510869 4.87134236 4.86758171 4.86382672
4.86007738 4.85633367 4.85259559 4.84886313 4.84513626 4.84141499
4.8376993  4.83398917 4.8302846  4.82658557 4.82289208 4.81920411
4.81552165 4.81184469 4.80817322 4.80450723 4.8008467  4.79719163
4.793542   4.7898978  4.78625902 4.78262566 4.77899769 4.77537511
4.77175791 4.76814607 4.76453959 4.76093845 4.75734265 4.75375217
4.75016699 4.74658712 4.74301254 4.73944324 4.73587921 4.73232043
4.7287669  4.72521861 4.72167554 4.71813768 4.71460503 4.71107757
4.7075553  4.70403819 4.70052625 4.69701946 4.69351781 4.69002129
4.68652989 4.6830436  4.6795624  4.6760863  4.67261527 4.66914931
4.66568841 4.66223255 4.65878173 4.65533594 4.65189516 4.64845939
4.64502862 4.64160283 4.63818202 4.63476617 4.63135528 4.62794933
4.62454832 4.62115224 4.61776107 4.6143748  4.61099343 4.60761695
4.60424534 4.6008786  4.59751671 4.59415967 4.59080747 4.58746009
4.58411753 4.58077977 4.57744681 4.57411865 4.57079525 4.56747663
4.56416277 4.56085365 4.55754927 4.55424963 4.5509547  4.54766449
4.54437897 4.54109815 4.53782201 4.53455054 4.53128374 4.52802159
4.52476409 4.52151122 4.51826298 4.51501935 4.51178033 4.50854592
4.50531608 4.50209083 4.49887015 4.49565404 4.49244247 4.48923545
4.48603296 4.482835   4.47964155 4.47645261 4.47326817 4.47008822
4.46691274 4.46374174 4.4605752  4.45741312 4.45425547 4.45110227
4.44795349 4.44480912 4.44166917 4.43853362 4.43540246 4.43227568
4.42915327 4.42603523 4.42292155 4.41981222 4.41670722 4.41360655
4.41051021 4.40741818 4.40433046 4.40124703 4.39816789 4.39509303
4.39202244 4.38895611 4.38589403 4.3828362  4.37978261 4.37673325
4.37368811 4.37064718 4.36761045 4.36457792 4.36154958 4.35852541
4.35550542 4.35248958 4.3494779  4.34647037 4.34346698 4.34046771
4.33747257 4.33448154 4.33149462 4.32851179 4.32553305 4.3225584
4.31958781 4.31662129 4.31365883 4.31070042 4.30774605 4.30479572
4.3018494  4.29890711 4.29596883 4.29303454 4.29010426 4.28717795
4.28425563 4.28133728 4.27842289 4.27551245 4.27260597 4.26970342
4.26680481 4.26391012 4.26101934 4.25813248] kpc/arcsec
```

[96]: Text(0, 0.5, 'Conversion Factor')



d.) You're writing a proposal to the ALMA telescope, to study a galaxy at a redshift of 0.5. If the telescope gives you 0.4″ resolution, what's the linear size (in kpc) corresponding to that resolution? You can stick with the cosmological parameters noted above, as they are fairly standard and will not surprise or shock any of your readers. [5 pts]

[97]:
```
conversion_factor(72, 0.26, 0.5) * 0.4 # kpc/arcsecond * arcsecond
```

```
The conversion factor is 6.035953865591871 kpc/arcsec
```

[97]:
```
2.414381546236749
```

The linear size would be 2.414 kpc, obtained simply by applying the conversion factor .