In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\11_winequality-red.csv")
df.fillna(0,inplace=True)
df
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [3]:
```
df.head()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [5]:
```
import seaborn as sns
```

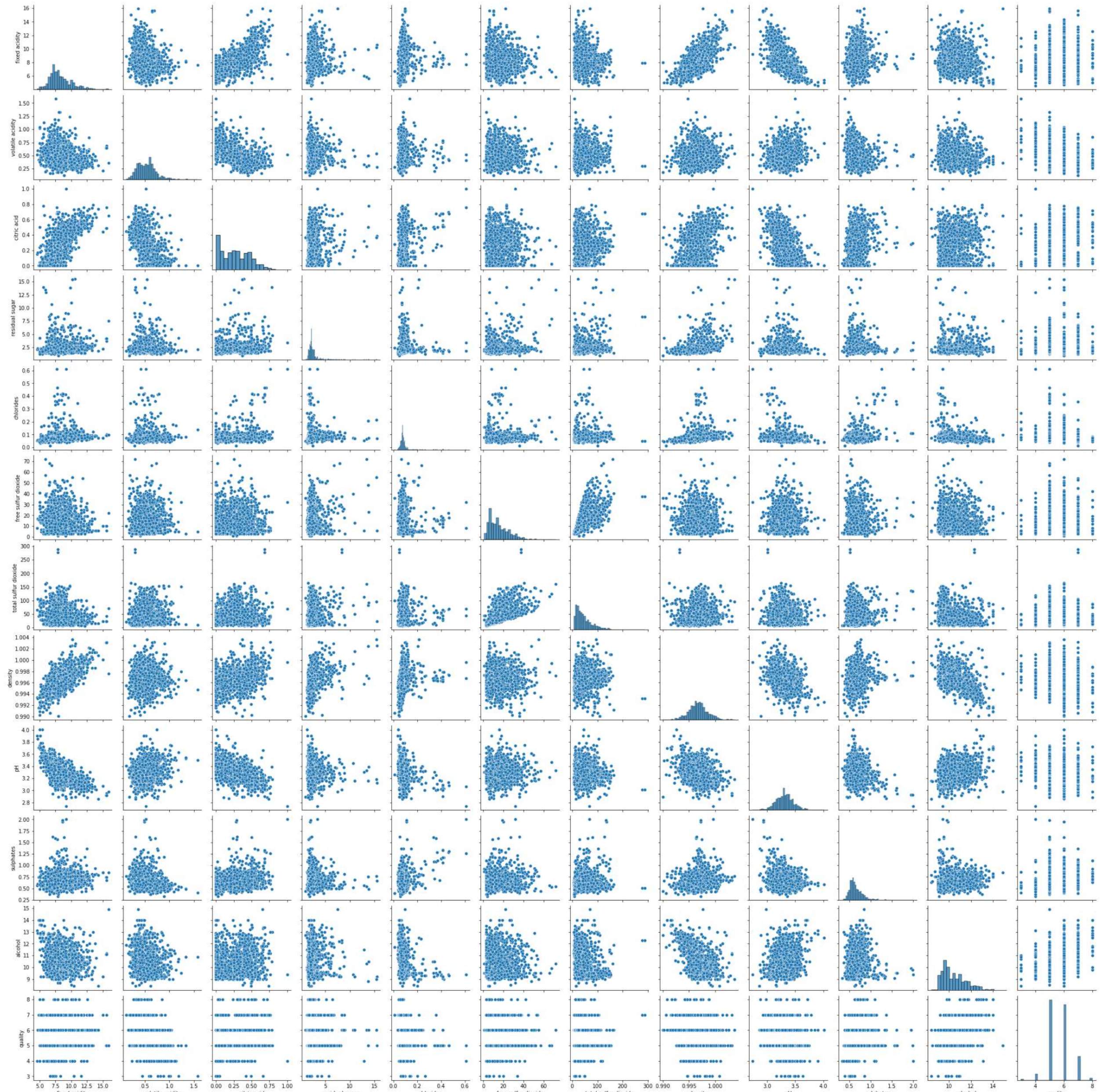
In [6]:
```
df.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	der
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003

In [7]:
```
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x21fc2b00580>

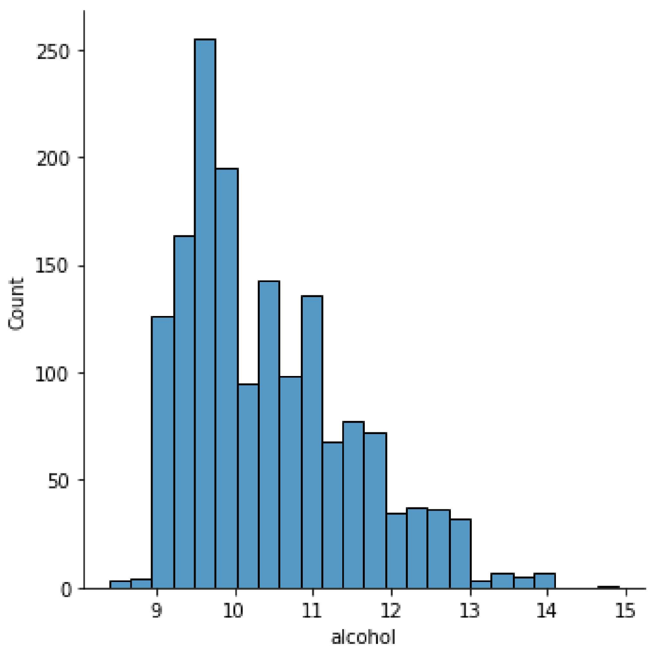


In [8]:
```
df1=df.drop(['citric acid'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

Out[8]:
```
fixed acidity           0
volatile acidity        0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

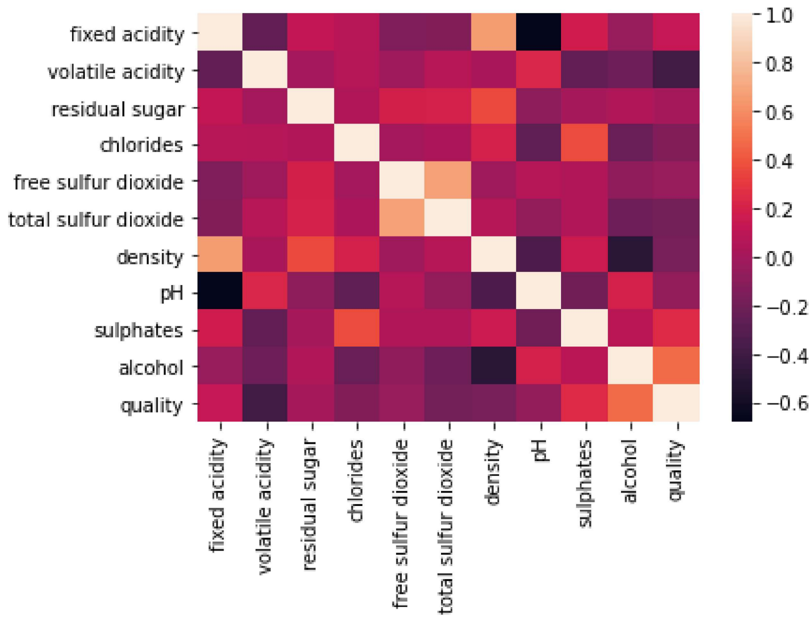
In [9]:
```
sns.displot(df['alcohol'])
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x21fc9aa6d60>



In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [12]:
```
df1.isna().sum()
```

Out[12]:
```
fixed acidity           0
volatile acidity        0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [13]:
```
y=df1['fixed acidity']
x=df1.drop(['chlorides','residual sugar'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

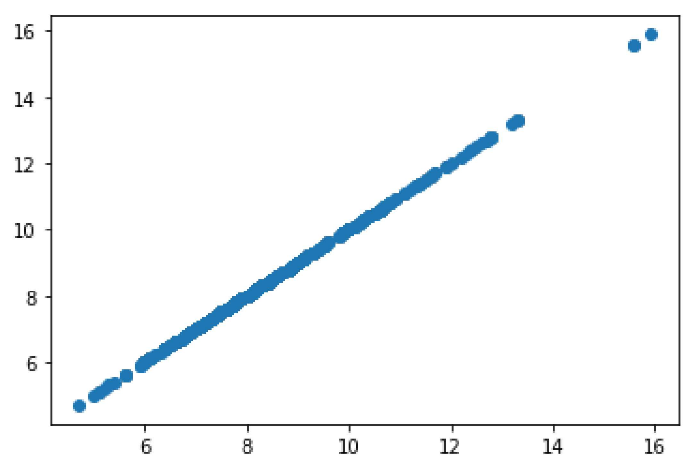
```
      fixed acidity  volatile acidity  free sulfur dioxide  \
573            10.5             0.590                 14.0
1078            8.6             0.370                  3.0
1256            7.5             0.590                 43.0
1309            7.0             0.620                 27.0
86              8.6             0.490                 20.0
...             ...               ...                  ...
154             7.1             0.430                 29.0
1008            8.9             0.350                 12.0
271            11.5             0.180                  4.0
65              7.2             0.725                  4.0
767             7.5             0.600                 13.0

      total sulfur dioxide  density    pH  sulphates  alcohol  quality
573                   47.0  0.99910  3.30       0.56      9.6        4
1078                   8.0  0.99817  3.27       0.58     11.0        5
1256                  60.0  0.99499  3.10       0.42      9.2        5
1309                  63.0  0.99600  3.28       0.61      9.2        5
86                   136.0  0.99720  2.93       1.95      9.9        6
...                    ...      ...   ...        ...      ...      ...
154                  129.0  0.99730  3.42       0.72     10.5        5
1008                  24.0  0.99549  3.23       0.70     12.0        7
271                   23.0  0.99960  3.28       0.97     10.1        6
65                    11.0  0.99620  3.41       0.39     10.9        5
767                   98.0  0.99938  3.45       0.62      9.5        5

[1075 rows x 9 columns]
```

In [14]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[14]:
```
1.6697754290362354e-13
```

In [15]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x21fcd353e50>
```



In [16]:
```
model.score(x_test,y_test)
```
Out[16]:
```
1.0
```

In [17]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [18]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```
Out[18]:
```
Ridge(alpha=10)
```

In [19]:
```
rr.score(x_test,y_test)
```
Out[19]:
```
0.999987820988224
```

In [20]:
```
la =Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[20]:
```
Lasso(alpha=10)
```

In [21]:
```
la.score(x_test,y_test)
```
Out[21]:
```
-0.00041081260831399929
```

In [22]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
[ 0.71067243 -0.         -0.         -0.00129119  0.         -0.
  0.         -0.          0.        ]
2.4811184505018398
[ 7.42612805  7.68586439  7.77371712  7.10312335  7.89519254  8.40820927
  7.90949912  7.56955372 10.95495756  7.54114752  8.21308422  8.06320259
  9.99100474  9.80367857  7.5101072   8.86425839  8.38884141  8.90950181
  8.32805196  9.70802696  7.79701029  7.27758934  7.49724703  7.77113473
  8.71566795  9.78555016 10.4510309   6.92736616  8.08644402  7.91843398
  7.57730087  7.19614084  8.47276882  8.84101695  7.70264987  7.96238621
  7.25558736  9.85145264  7.27108165  7.69232034  9.46512784 10.37221651
  6.60317374  8.34876275  8.36162292  9.90062137  7.47008028  6.71158205
 10.04141293  7.88357182  9.56331009  7.54760348  7.43129281  7.65358461
  6.70254371  7.36291142 10.90325818  8.05023894 10.6060773   8.71825033
  8.41859053  7.47911862  7.64712866 10.56083387  8.06315085  6.72320277
  9.036142    7.99347827  8.89922402  8.92757848 11.5234955   9.14599671
  9.98842236 10.75466774  7.09021144  8.61888035  9.50778888  7.81250458
 10.52984529  7.08499494  7.42747097  8.00892083  7.58246564  9.90578613
 10.04399531  7.79701029  7.34602247  7.99983075  7.8680258   9.40185948
  8.12388857  7.40810311  9.57498255  8.74412588  7.86942046  7.16122694
  7.92747232  6.6754287   9.41089782  6.98939506  6.64309719  7.24654902
 11.14357492  7.48820869 11.87754052  8.18462628  7.80475744  9.4366699
 10.70555074  9.91219035  7.32536341 11.81680281  8.91595776  7.91326922
  7.90040904  7.2646257   7.15089741  8.15363769  9.91219035  7.07466541
  7.83838014  7.19872322  7.58633921  8.61500678  6.63023701  8.64207006
  7.55023759  7.48820869  7.36285969  7.84225371  8.23503447  8.8474729
  8.68080579  7.47911862 10.13443042  9.0994621   8.03216227  8.64207006
  6.44668094  9.27258517  7.32278103 11.3826522   7.77247766  8.05798609
  9.01806533 10.12151851  8.35392751  8.60973854  7.96367741  9.49224285
  8.36420531  9.81013453  8.40170158  6.98299084  9.43150514  6.83305748
  7.56180658  6.65988267  7.54114752  9.01935652  7.64320335  6.98293911
  8.21695779  9.55685414  7.66133176  7.76596997 10.55566911  7.23880188
  8.6692368   8.00112195  8.04636537  8.86038481  7.48686576  8.32805196
  9.18473244  7.60705    10.1086066   8.79319114  8.79319114  7.15089741
  7.89777493  8.9586188   5.68441261  8.10710308  8.60070021  9.54652461
  8.23761685  7.80863101  7.25558736  8.74541708  8.42494302  7.43516639
  7.23105473 10.76628846  8.91337538  8.38104252  7.82670768  7.77113473
  6.98939506  9.79071492  6.8149808   9.52328317  7.69366327  9.63566853
  7.84607555  7.65482407  8.29313806  9.80238738  7.90815619  7.76080521
  8.47276882  7.15100089 11.20431263  7.49461291  7.79701029  7.55411117
  7.59672047  8.03732703 10.1706355   9.06847351  7.69619392  7.01392769
  8.07869688  7.95334788  7.01392769  7.92488994  7.79566736  7.1392767
  8.43924959  7.51279305  8.91466657  7.57213611  8.93661682  7.36931564
  8.24794638  7.31890746 11.54157217  8.93532563  8.39395443  6.33305613
  8.56965989  7.08370374  7.87195111  8.07348038  6.80459954  7.11990883
  7.82025173  7.36409914  7.46878909  8.26085829  8.5825718   7.80863101
  8.66407204  7.68973796  7.54114752  8.7699497   8.29835456  7.42354567
  7.96367741  8.67564102 10.06465437  8.446945    9.1911884   7.55669355
 11.54157217  7.30336143  6.79943477  7.04367682  8.42236064  7.80868274
  7.33827532  9.01935652  7.95463907  9.98713117  6.13276631  7.40810311
 10.20937123  7.41063376  7.34602247  9.1756941  10.50138735  6.78006691
  8.62394165  7.65482407  7.71427059  7.47911862  9.78425897  8.9070229
  9.2674204   8.29184687 10.70684193  9.14470552  8.62135926  7.38480993
  7.8293418   8.02575805  6.09527004  8.32686424  9.20802562  9.24557363
  7.41455906  9.625339    8.04765656  8.04001288  7.82670768  8.35904054
  9.91353328  7.79566736  7.35506081  9.73519371  7.76349106 10.88389031
  8.9249961   8.66143792  6.71158205  6.74133118  7.36931564  6.13276631
  6.44409856  6.33305613  7.63287382  7.69102915  9.04259796  8.54781311
  9.67703838  9.64212448  7.69237208  7.34989604 13.5120871  10.04399531
  7.5786438   8.46373049  7.50757655 13.53791092  8.78033097 11.23793534
  7.11350461  8.06702442  7.70270161  7.50757655  8.45598334  9.92381107
  8.53738011  8.7712409   7.30077905  6.61727337 10.62415397  8.35516697
```

```
  8.93016087  9.68349433  7.50881601  8.70533842  9.90831678  9.9057344
  6.98810387  9.70415339  9.47416618  9.57363962 11.86462861  8.21561486
  7.19872322  7.66913064  9.06196582  9.46254546  9.50644595 11.53253383
  7.56180658  7.57213611  8.91208419  5.9750858   9.18215006  7.36673326
  9.98454879  7.91197803 10.10731541  7.23105473  7.97405867  9.5775132
  6.96744482  8.23116089  8.08644402 11.16562864  7.72201774  7.48299219
  8.27893496  7.2349283   8.27247901  8.26865717  9.2674204   7.77888188
 10.55825149  9.78296778  7.88878832  8.7609631   8.91208419  7.01144878
  7.33698413  7.46104194  8.11355904  8.71577142 10.49234901  7.6897897
  8.00499552  7.91326922  9.70420512  6.19866879  7.35118723  6.96098886
  7.65224169  8.81261074  8.05803782  8.05411251 11.34515593  9.10333567
 10.2391721   8.04636537  7.45716837 10.02204506  9.13561545  7.70781464
  9.56847486  8.05550718  8.43785493  9.07616892  8.91337538  8.32417838
  9.77398118  9.96259854  7.34602247  8.58515418  7.28270237  7.46620671
 10.0789092   8.63168879  6.74133118  8.1045207   8.32417838 11.88141409
 10.65906786  8.02312393  6.6754287   7.15477099  7.95593026  7.86415222
  8.621411   11.47825208  7.98433646  8.79060876  7.58251737 13.68913548
 10.35667048  6.04873542  8.22604786  7.79308498  7.92488994  8.1187238
  5.97513753  8.00112195  7.77113473  8.22207082  7.90293969  8.42117292]
0.9175337096091217
Mean Absolute Error: 7.130419391055768e-15
Mean Squared Error: 8.221399074780627e-29
Root Mean Squared Error: 9.067193101936578e-15
```