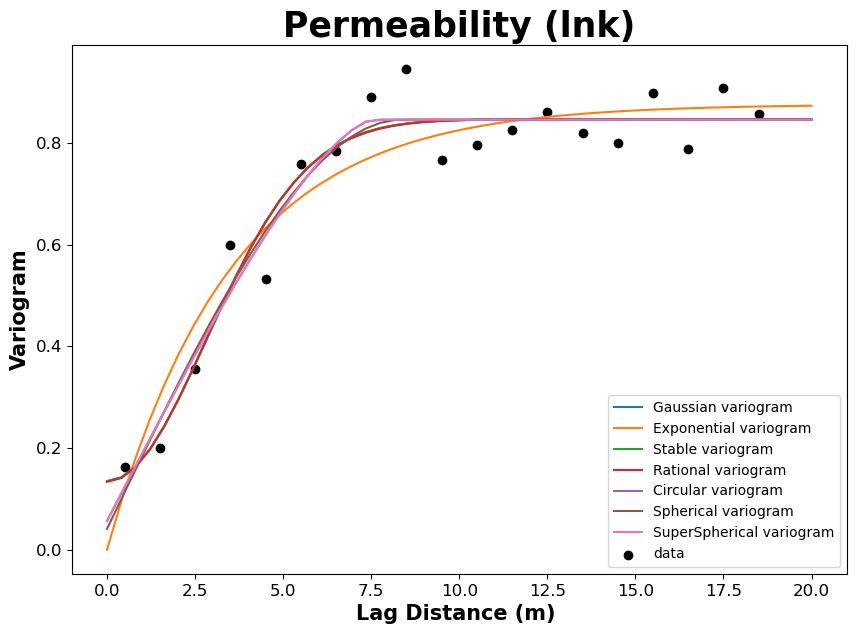
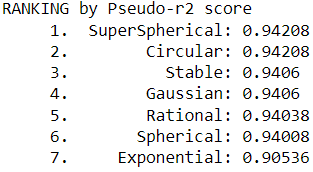
**Cokriging Permeability**

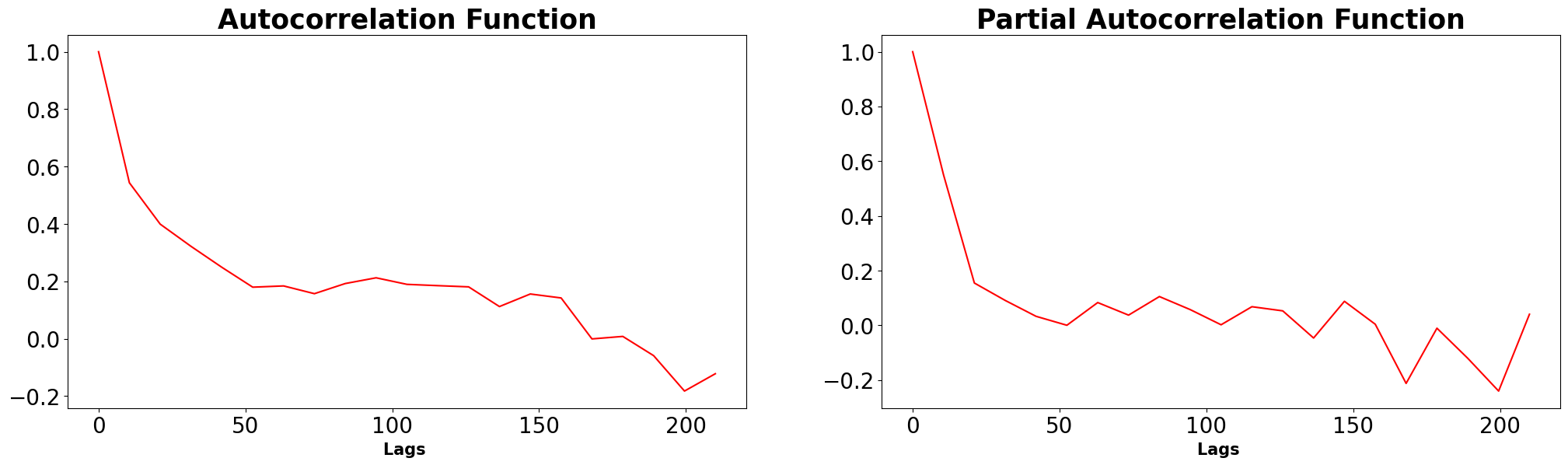
We also estimated permeability by simple kriging. First, we calculated bulk density from grain density values from core using the equation below

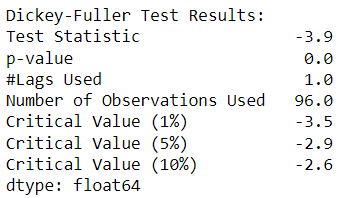
Then we trained the kriging model with input features from the core bulk density calculated from the equation above and core permeability to predict the logarithm of permeability at log depths. We fit a variogram and selected the best model based on the ranking produced.





We also performed stationarity test by analyzing the autocorrelation and Dickey-Fuller test results.

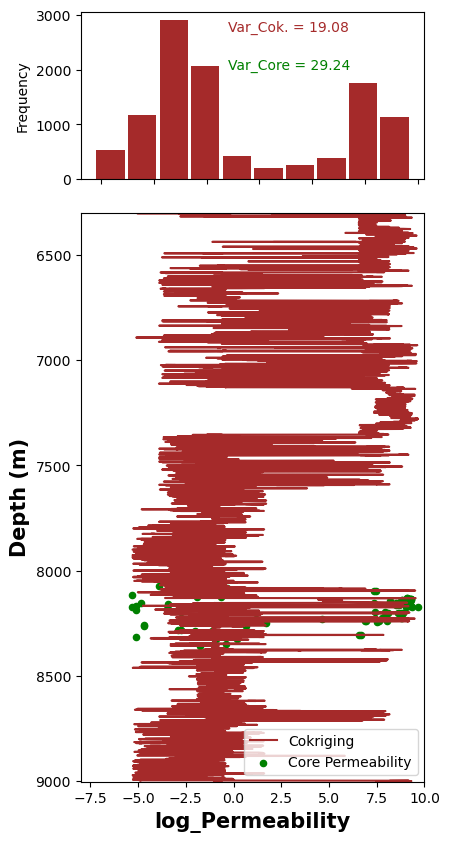




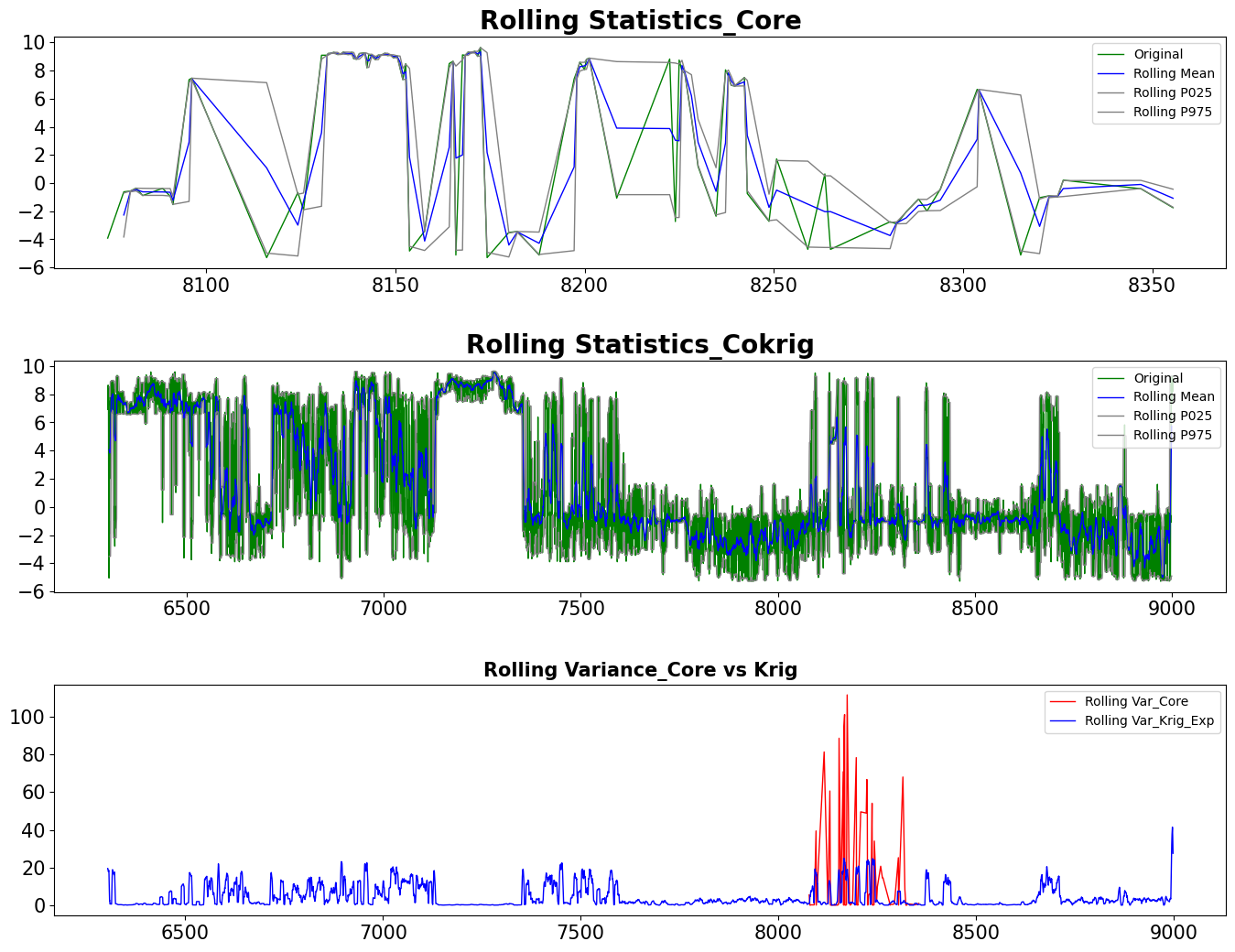
The test statistic is less than the critical value, therefore the data is stationary. Then, we adopted the cokriging equation below

+

where is the estimate at the log depths, is the value secondary variable at the estimation location and the λ’s are the kriging weights. The prediction results are shown in the diagram.



To ensure that the variance of the core permeability is preserved, we evaluate the prediction results by comparing the rolling statistics of the core and log permeability.



**Python Code**

### Selecting the Variogram type

import numpy as np

from gstools import Gaussian, SRF, Spherical

### Fit Variogram

import numpy as np

from gstools import SRF, Exponential, Stable, vario\_estimate\_unstructured

x = np.array(LN1\_core["DEPTH"])

y = np.array(LN1\_core["lnk"])

model = Exponential(dim=2, var=2, len\_scale=7)

srf = SRF(model, mean=0, seed=19970221)

field = srf((x, y))

bins = np.arange(20)

bin\_center, gamma = vario\_estimate((x, y), field, bins)

models = {

"Gaussian": gs.Gaussian,

"Exponential": gs.Exponential,

"Stable": gs.Stable,

"Rational": gs.Rational,

"Circular": gs.Circular,

"Spherical": gs.Spherical,

"SuperSpherical": gs.SuperSpherical,

}

scores = {}

from matplotlib.pyplot import figure

figure(figsize=(10, 7), dpi=100)

# plot the estimated variogram

plt.scatter(bin\_center, gamma, color="k", label="data")

ax = plt.gca()

# fit all models to the estimated variogram

for model in models:

fit\_model = models[model](dim=2)

para, pcov, r2 = fit\_model.fit\_variogram(bin\_center, gamma, return\_r2=True)

fit\_model.plot(x\_max=20, ax=ax)

scores[model] = r2

#plt.plot(xx,yy,color='red',linewidth=4)

plt.xlabel("Lag Distance (m) ", fontweight='bold',fontsize=15)

plt.ylabel("Variogram", fontweight='bold',fontsize=15)

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.title("Permeability (lnk)", fontweight='bold',fontsize=25)

print(fit\_model)

plt.legend()

ranking = sorted(scores.items(), key=lambda item: item[1], reverse=True)

print("RANKING by Pseudo-r2 score")

for i, (model, score) in enumerate(ranking, 1):

print(f"{i:>6}. {model:>15}: {score:.5}")

plt.show()

### Detecting Stationarity

from statsmodels.tsa.stattools import acf, pacf

nlags = 20

lag\_acf = acf(y, nlags=nlags)

lag\_pacf = pacf(y, nlags=nlags, method='ols')

lags = np.linspace(0,(nlags+1)\*10,nlags+1)

#Plot ACF:

plt.subplot(121)

plt.plot(lags,lag\_acf,color='red')

plt.xlabel("Lags ", fontweight='bold',fontsize=15)

plt.xticks(fontsize=20)

plt.yticks(fontsize=20)

plt.title('Autocorrelation Function',fontweight='bold',fontsize=25)

#Plot PACF:

plt.subplot(122)

plt.plot(lags,lag\_pacf,color='red')

plt.xlabel("Lags ", fontweight='bold',fontsize=15)

plt.xticks(fontsize=20)

plt.yticks(fontsize=20)

plt.title('Partial Autocorrelation Function',fontweight='bold',fontsize=25)

plt.tight\_layout()

plt.subplots\_adjust(left=0.0, bottom=0.0, right=3.0, top=1.0, wspace=0.2, hspace=0.2)

print('Dickey-Fuller Test Results:')

dftest = adfuller(y, autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

for key,value in dftest[4].items():

dfoutput['Critical Value (%s)'%key] = value

print(dfoutput[0:7])

### Kriging Permeability

import matplotlib.pyplot as plt

import numpy as np

from pykrige import SimpleKriging

plt.style.use("default")

X\_D = np.array(LN1\_core["DEPTH"])

X\_pred\_D = np.array(LN1\_log["DEPTH"])

X = np.array(LN1\_core["CRHOB"])

y = np.array(LN1\_core["lnk"])

X\_pred = np.array(LN1\_log["RHOZ"])

#uk = SimpleKriging(X, np.zeros(X.shape), y, variogram\_model="linear",variogram\_parameters = {'slope': 2, 'nugget': 0})

uk = SimpleKriging(X, np.zeros(X.shape), y, variogram\_model="gaussian",variogram\_parameters = {'psill': 0.4, 'range': 0.4, 'nugget': 0})

uk\_d = SimpleKriging(X, np.zeros(X.shape), y, variogram\_model="gaussian",variogram\_parameters = {'psill': 0.7, 'range': 0.7, 'nugget': 0})

uk\_s = SimpleKriging(X, np.zeros(X.shape), y, variogram\_model="exponential")

y\_pred, y\_std = uk.execute("grid", X\_pred, np.array([0.0]))

y\_pred\_d, y\_std\_d = uk\_d.execute("grid", X\_pred, np.array([0.0]))

y\_pred\_s, y\_std\_s = uk\_s.execute("grid", X\_pred, np.array([0.0]))

y\_pred = np.squeeze(y\_pred)

y\_std = np.squeeze(y\_std)

y\_pred\_d = np.squeeze(y\_pred\_d)

y\_std\_d = np.squeeze(y\_std\_d)

y\_pred\_s = np.squeeze(y\_pred\_s)

y\_std\_s = np.squeeze(y\_std\_s)

fig, ax = plt.subplots(1, 1, figsize=(4, 10))

#ax.scatter(y, X, s=40, label="Core Permeability", color = 'green')

ax.scatter(y, X\_D, s=20,label="Core Permeability", color = 'green')

#ax.plot(y\_pred,X\_pred\_D, label="Gaussian", color = 'black')

#ax.plot(y\_pred\_d,X\_pred, label="Kriged Perm\_Gaussian", color = 'purple')

ax.plot(y\_pred\_s,X\_pred\_D, label="Exponential", color = 'blue')

#ax.fill\_betweenx(X\_pred,y\_pred - 2 \* y\_std,y\_pred +2 \* y\_std,alpha=0.3,label="Confidence interval",)

ax.legend(loc=4)

ax.set\_xlabel("log\_Permeability")

ax.set\_ylabel("Depth (m)")

ax.set\_ylim(6300,9002)

ax.xaxis.set\_label\_position('top')

ax.xaxis.tick\_top()

ax.invert\_yaxis()

ax.set\_xlim(-8, 10)

#ax.semilogy()

plt.show()

### Evaluating Results

window\_size = 2 # assume window size of 10 days

#Determing rolling statistics

rolling\_mean = df.rolling(window = window\_size, center = True).mean()

rolling\_std = df.rolling(window = window\_size, center = True).var()

rolling\_P025 = df.rolling(window = window\_size, center = True).quantile(.025)

rolling\_P975 = df.rolling(window = window\_size, center = True).quantile(.975)

window\_size2 = 30

#Determing rolling statistics

rolling\_mean2 = df2.rolling(window = window\_size2, center = True).mean()

rolling\_std2 = df2.rolling(window = window\_size2, center = True).var()

rolling\_P0252 = df2.rolling(window = window\_size2, center = True).quantile(.025)

rolling\_P9752 = df2.rolling(window = window\_size2, center = True).quantile(.975)

#Plot rolling statistics:

plt.subplot(311)

orig = plt.plot(df["lnk"], color='green',linewidth = 1, label='Original')

mean = plt.plot(rolling\_mean["lnk"], color='blue', linewidth = 1, label='Rolling Mean')

P025 = plt.plot(rolling\_P025, color='grey', linewidth = 1, label='Rolling P025')

P975 = plt.plot(rolling\_P975, color='grey', linewidth = 1, label='Rolling P975')

plt.title('Rolling Statistics\_Core', fontweight='bold',fontsize=20); plt.legend(loc='best')

plt.xticks(fontsize=15)

plt.yticks(fontsize=15)

plt.subplot(312)

orig2 = plt.plot(df2["Pred"], color='green',linewidth = 1, label='Original')

mean2 = plt.plot(rolling\_mean2["Pred"], color='blue', linewidth = 1, label='Rolling Mean')

P0252 = plt.plot(rolling\_P0252, color='grey', linewidth = 1, label='Rolling P025')

P9752 = plt.plot(rolling\_P9752, color='grey', linewidth = 1, label='Rolling P975')

plt.title('Rolling Statistics\_Cokrig', fontweight='bold',fontsize=20); plt.legend(loc='best')

plt.xticks(fontsize=15)

plt.yticks(fontsize=15)

plt.subplot(313)

std = plt.plot(rolling\_std["lnk"], color='red', linewidth = 1, label = 'Rolling Var\_Core')

std = plt.plot(rolling\_std2["Pred"], color='blue', linewidth = 1, label = 'Rolling Var\_Krig\_Exp')

plt.legend(loc='best'); plt.title('Rolling Variance\_Core vs Krig', fontweight='bold',fontsize=15)

plt.xticks(fontsize=15)

plt.yticks(fontsize=15)

plt.subplots\_adjust(left=0.0, bottom=0.0, right=2.0, top=2.0, wspace=0.5, hspace=0.4)