BST 263 HOMEWORK 2

Jenny Wang (71401898)

February 25, 2020

(0/7 late days used; 7/7 late days remaining)

For Questions 2.2 and 2.3, see attached Latex document at the end.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import random as random
%matplotlib inline
import math
import statsmodels.api as sm
import pandas as pd
from numpy import array
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import enet_path
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import lasso_path
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV
from sklearn.metrics import mean_squared_error
from itertools import cycle
from itertools import chain
from itertools import product
from scipy.io import wavfile
from pydub import AudioSegment
from pydub import playback
```

Question 1.1

In [2]:
```
# Load data
train = pd.read_csv("train.data.csv", sep=",")
test = pd.read_csv("test.data.csv", sep=",")

# Create categorical variables for zipcode
train_catzip = pd.get_dummies(train['zipcode'])
test_catzip = pd.get_dummies(test['zipcode'])

# Incorporate categorical zipcodes into original datasets
train = pd.concat([train, train_catzip], axis=1)
test = pd.concat([test, test_catzip], axis=1)
```

Converting the type of `zipcode` is necessary because there is no numerical significance to zipcodes. They are not integers and we do not use them to do computations. It would only make sense to build a regression model that treats zipcodes as nominal categories and not integers. Without this conversion, the prediction results of the regression would not be valid or have any meaning.

Question 1.2a

In [3]:
```
np.random.seed(263)

# Subset variables in the training set
X_train = train[["bedrooms", "bathrooms", "sqft_living", "sqft_lot"]]
y_train = train["price"]

# Build a linear regression model on training data
fit = LinearRegression().fit(X_train, y_train)

# Predict price using the linear model
#train_pred = fit.predict(X_train)

# Goodness of fit on training data
print("The R\u00b2 of the model on training data is {}.".format(fit.scor
e(X_train, y_train)))
```

```
The R² of the model on training data is 0.5101138530794578.
```

In [4]:
```
np.random.seed(263)

# Subset variables in the test set
X_test = test[["bedrooms", "bathrooms", "sqft_living", "sqft_lot"]]
y_test = test["price"]

# Predict price using the linear model
#test = fit.predict(X_test)

# Goodness of fit on test data
print("The R\u00b2 of the model on test data is {}.".format(fit.score(X_
test, y_test)))
```

```
The R² of the model on test data is 0.5049944614037092.
```

From the linear model regressing `price` on `bedrooms`, `bathrooms`, `sqft_living`, and `sqft_lot`, we see that the R^2 on the training data is higher than that on the test data by approximately 1%.

Question 1.2b

In [5]:
```
np.random.seed(263)

# Subset variables in the training set, this time including zipcode
X_train = train[["bedrooms", "bathrooms", "sqft_living", "sqft_lot"]]
X_train = pd.concat([X_train, train_catzip], axis=1)
y_train = train["price"]

# Build a linear regression model on training data
fit = LinearRegression().fit(X_train, y_train)

# Predict price using the linear model
train_pred = fit.predict(X_train)

# Goodness of fit on training data
print("The R\u00b2 of the model on training data is {}.".format(fit.scor
e(X_train, y_train)))
```

```
The R² of the model on training data is 0.7392754568193081.
```

In [6]:
```
np.random.seed(263)

# Subset variables in the test set, this time including zipcode
X_test = test[["bedrooms", "bathrooms", "sqft_living", "sqft_lot"]]
X_test = pd.concat([X_test, test_catzip], axis=1)
y_test = test["price"]

# Predict price using the linear model
#test_pred = fit.predict(X_test)

# Goodness of fit on test data
print("The R\u00b2 of the model on test data is {}.".format(fit.score(X_
test, y_test)))
```

```
The R² of the model on test data is 0.7378668448927312.
```

Adding zipcode improves the R^2 by approximately 45% and 46% for training and test data, respectively. The R^2 on the training data is higher than that on the test data by approximately 0.2%.

Question 1.3

In [7]:
```
# Generate polynomial variables
poly = PolynomialFeatures(3, include_bias=False)
train_polybed = poly.fit_transform(train[["bedrooms"]])
train_polybed = pd.DataFrame(train_polybed)
#train_polybed.columns = ["bedrooms", "bedrooms", "bedrooms", "bedroom
s"]
train_polybath = poly.fit_transform(train[["bathrooms"]])
train_polybath = pd.DataFrame(train_polybath)
#train_polybath.columns = ["bathrooms", "bathrooms", "bathrooms", "bathr
ooms"]

test_polybed = poly.fit_transform(test[["bedrooms"]])
test_polybed = pd.DataFrame(test_polybed)
#test_polybed.columns = ["bedrooms", "bedrooms", "bedrooms", "bedrooms"]
test_polybath = poly.fit_transform(test[["bathrooms"]])
test_polybath = pd.DataFrame(test_polybath)
#test_polybath.columns = ["bathrooms", "bathrooms", "bathrooms", "bathro
oms"]

# Variables to keep
train_variables = X_train.drop(["bedrooms", "bathrooms"], axis=1)
test_variables = X_test.drop(["bedrooms", "bathrooms"], axis=1)
```

In [8]:
```
# 2nd degree polynomial
## Training data
X_train_2 = pd.concat([train_polybed.iloc[:, 1], train_polybath.iloc[:,
1], train_variables], axis=1)
y_train_2 = train["price"]

## Build a linear regression model on training data
fit = LinearRegression().fit(X_train_2, y_train_2)

## Goodness of fit on training data
print("The R\u00b2 of the model on training data is {}.".format(fit.scor
e(X_train_2, y_train_2)))

## Test data
X_test_2 = pd.concat([test_polybed.iloc[:, 1], test_polybath.iloc[:, 1],
test_variables], axis=1)
y_test_2 = test["price"]

## Goodness of fit on test data
print("The R\u00b2 of the model on test data is {}.".format(fit.score(X_
test_2, y_test_2)))
```

```
The R² of the model on training data is 0.7362562392305858.
The R² of the model on test data is 0.7367450006593194.
```

In [9]:
```
# 3rd degree polynomial
## Training data
X_train_3 = pd.concat([train_polybed.iloc[:, 2], train_polybath.iloc[:,
2], train_variables], axis=1)
y_train_3 = train["price"]

## Build a linear regression model on training data
fit = LinearRegression().fit(X_train_3, y_train_3)

## Goodness of fit on training data
print("The R\u00b2 of the model on training data is {}.".format(fit.scor
e(X_train_3, y_train_3)))

## Test data
X_test_3 = pd.concat([test_polybed.iloc[:, 2], test_polybath.iloc[:, 2],
test_variables], axis=1)
y_test_3 = test["price"]

## Goodness of fit on test data
print("The R\u00b2 of the model on test data is {}.".format(fit.score(X_
test_3, y_test_3)))
```

```
The R² of the model on training data is 0.7401579725768732.
The R² of the model on test data is 0.7363867644616062.
```

Using polynomial features does not improve the goodness of the fit of the model by an appreciable amount. Using a second-degree linear model slightly increases test score compared to training score. Using a third-degree linear model actually produces a reduced test score compared to that from the first-degree model. Additionally, polynomial features generally reduce the parsimony of the model. Therefore, taking all of these factors into consideration, we do not need to include polynomial features in our model.

Question 1.4a

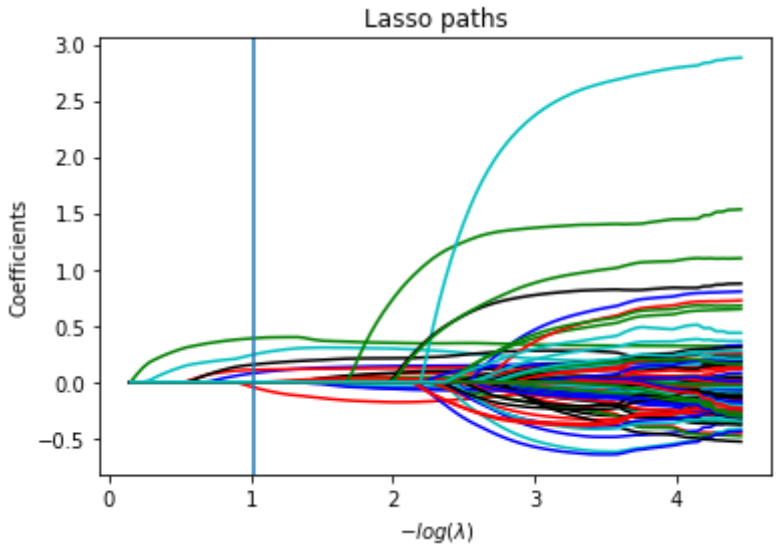
In [10]:

```
# Subset and standardize variables in the training set
## Predictors
X_train = pd.read_csv("train.data.csv", sep=",")
X_train = train.iloc[:, 4:23]
X_train -= X_train.mean(axis=0)
X_train /= X_train.std(axis=0)
X_train = pd.concat([X_train, train_catzip], axis=1)

## Outcome
y_train = train.iloc[:, 3]
y_train -= y_train.mean(axis=0)
y_train /= y_train.std(axis=0)

# Subset and standardize variables in the test set
## Predictors
test = pd.read_csv("test.data.csv", sep=",")
X_test = test.iloc[:, 4:23]
X_test -= X_test.mean(axis=0)
X_test /= X_test.std(axis=0)
X_test = pd.concat([X_test, test_catzip], axis=1)

## Outcome
y_test = test.iloc[:, 3]
y_test -= y_test.mean(axis=0)
y_test /= y_test.std(axis=0)
```

In [11]:
```
# Lasso regression on training data
random.seed(263)

eps = 5e-5
print("Computing regularization path using lasso...")
lambdas_lasso, coefs_lasso, _ = enet_path(
    X_train, y_train, n_alphas=100, eps=eps, l1_ratio=1., fit_intercept=
False)

plt.figure(1)
colors = cycle(["b", "r", "g", "c", "k"])
neg_log_lambdas_lasso = -np.log10(lambdas_lasso)
for coef_l, c in zip(coefs_lasso, colors):
    l1 = plt.plot(neg_log_lambdas_lasso, coef_l, c=c)

plt.axvline(x=-np.log10(0.09610779662375922))
plt.xlabel(r"$-log(\lambda)$")
plt.ylabel("Coefficients")
plt.title("Lasso paths")
#plt.legend((l1[-1]), ("Lasso', 'Elastic-Net'), loc='lower left')
plt.axis("tight")
```

```
Computing regularization path using lasso...
```

Out[11]:
```
(-0.06180494827423519,
 4.669328046956144,
 -0.8209226472436919,
 3.0565174245844506)
```



In [12]:
```
# Top 5 important features determined by lasso
lm_lasso = Lasso(alpha=0.09610779662375922)
lm_lasso.fit(X_train, y_train)
coef_dict = {}
for coef, feat in zip(lm_lasso.coef_, X_train.columns):
    coef_dict[feat] = coef

print("The most important features determined by lasso are:")
sorted(coef_dict, key=coef_dict.get)[0:5]
```

```
The most important features determined by lasso are:
```

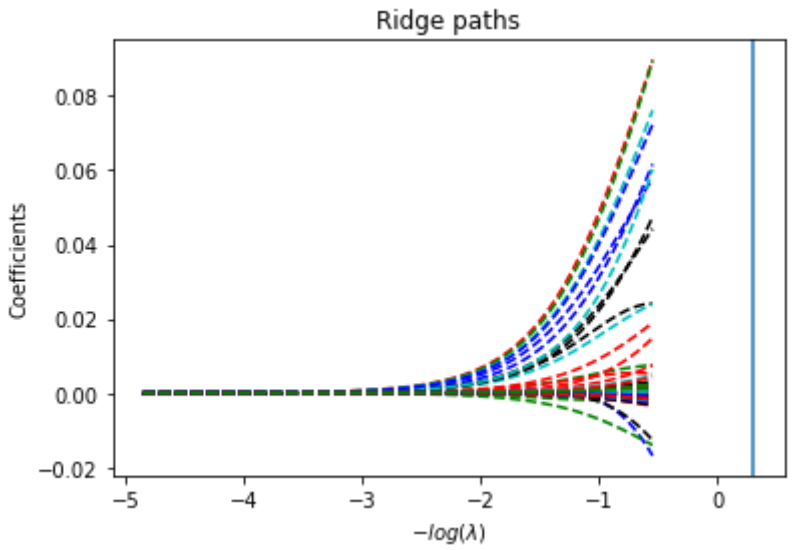
Out[12]:
```
['yr_built', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors']
```

The top five features determined by the lasso are `yr_built`, `bedrooms`, `bathrooms`, `sqft_lot`, and `floors`. These features makes a lot of sense because they directly have to do with the age, utility, and area of a house - all important things to consider when purchasing a house.

In [13]:
```
# Ridge regression on training data
random.seed(263)

print("Computing regularization path using ridge...")
lambdas_ridge, coefs_ridge, _ = enet_path(
    X_train, y_train, eps=eps, n_alphas=100, l1_ratio=0.00001, fit_inter
cept=False)

plt.figure(2)
neg_log_lambdas_ridge = -np.log10(lambdas_ridge)
for coef_r, c in zip(coefs_ridge, colors):
    l2 = plt.plot(neg_log_lambdas_ridge, coef_r, linestyle='--', c=c)


plt.axvline(x=-np.log10(0.49417133613238384))
plt.xlabel(r"$-log(\lambda)$")
plt.ylabel("Coefficients")
plt.title("Ridge paths")
plt.axis("tight")
```

```
Computing regularization path using ridge...
```

Out[13]:
```
(-5.104397243364567,
 0.5637662438531229,
 -0.022193146208053585,
 0.09514860014889637)
```



In [14]:
```
# Top 5 important features determined by ridge
lm_ridge = Ridge(alpha=0.35564803062231287)
lm_ridge.fit(X_train, y_train)
coef_dict = {}
for coef, feat in zip(lm_ridge.coef_, X_train.columns):
    coef_dict[feat] = coef

print("The most important features determined by ridge are:")
sorted(coef_dict, key=coef_dict.get)[0:5]
```

```
The most important features determined by ridge are:
```

Out[14]:
```
[98070, 98023, 98003, 98032, 98028]
```

The top 5 features determined by the ridge are all zipcodes. These features do not make sense.

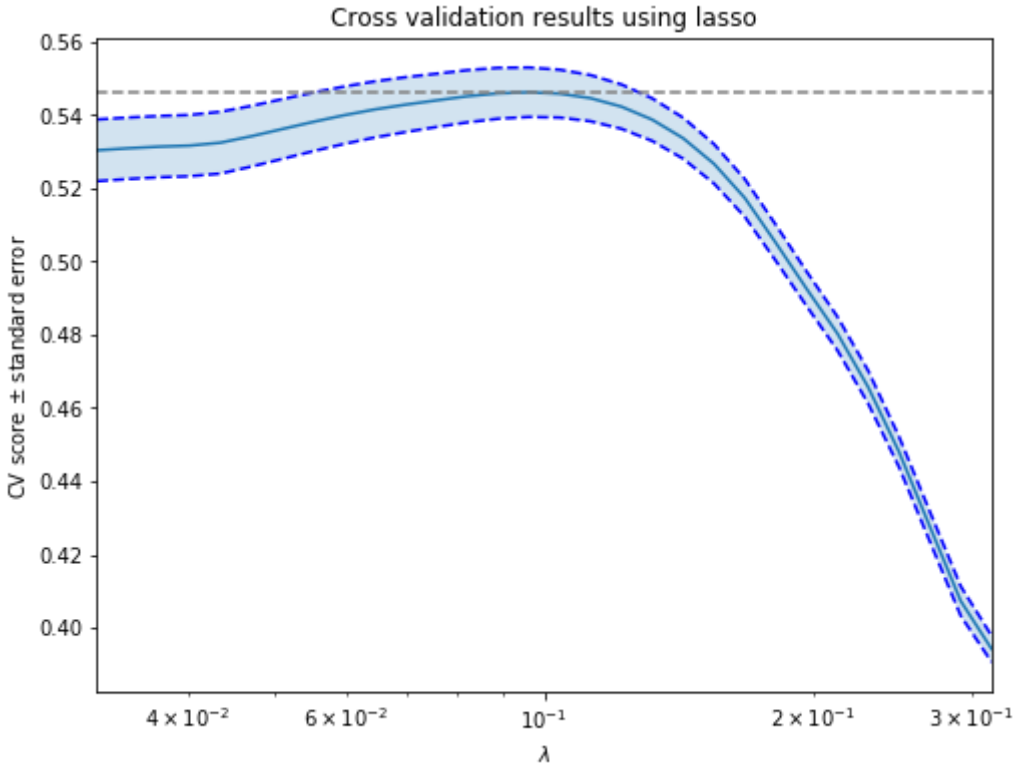
Question 1.4b

In [15]:
```
# 5-fold cross validation with lasso
np.random.seed(263)

lasso = Lasso(random_state=0, max_iter=100000, tol=10)
alphas = np.logspace(-1.5, -0.5, 30)

tuned_parameters = [{"alpha": alphas}]
n_folds = 5

clf = GridSearchCV(lasso, tuned_parameters, cv=n_folds)
clf.fit(X_train, y_train)
scores = clf.cv_results_["mean_test_score"]
scores_std = clf.cv_results_["std_test_score"]

# Best hyperparameter determined by lasso
print("The best hyperparameter chosen by lasso is {}.".format(clf.best_p
arams_["alpha"]))
```

```
The best hyperparameter chosen by lasso is 0.09610779662375922.
```

In [16]:
```
# Plot the cross valiadation results
plt.figure().set_size_inches(8, 6)
plt.semilogx(alphas, scores)

# Plot error lines showing +/- standard errors of the scores
std_error = scores_std / np.sqrt(n_folds)

plt.semilogx(alphas, scores + std_error, "b--")
plt.semilogx(alphas, scores - std_error, "b--")

# Fill in
plt.fill_between(alphas, scores + std_error, scores - std_error, alpha=
0.2)

plt.title("Cross validation results using lasso")
plt.xlabel(r"$\lambda$")
plt.ylabel(r"CV score $\pm$ standard error")
plt.axhline(np.max(scores), linestyle="--", color="0.5")
plt.xlim([alphas[0], alphas[-1]])
```

Out[16]:
```
(0.03162277660168379, 0.31622776601683794)
```

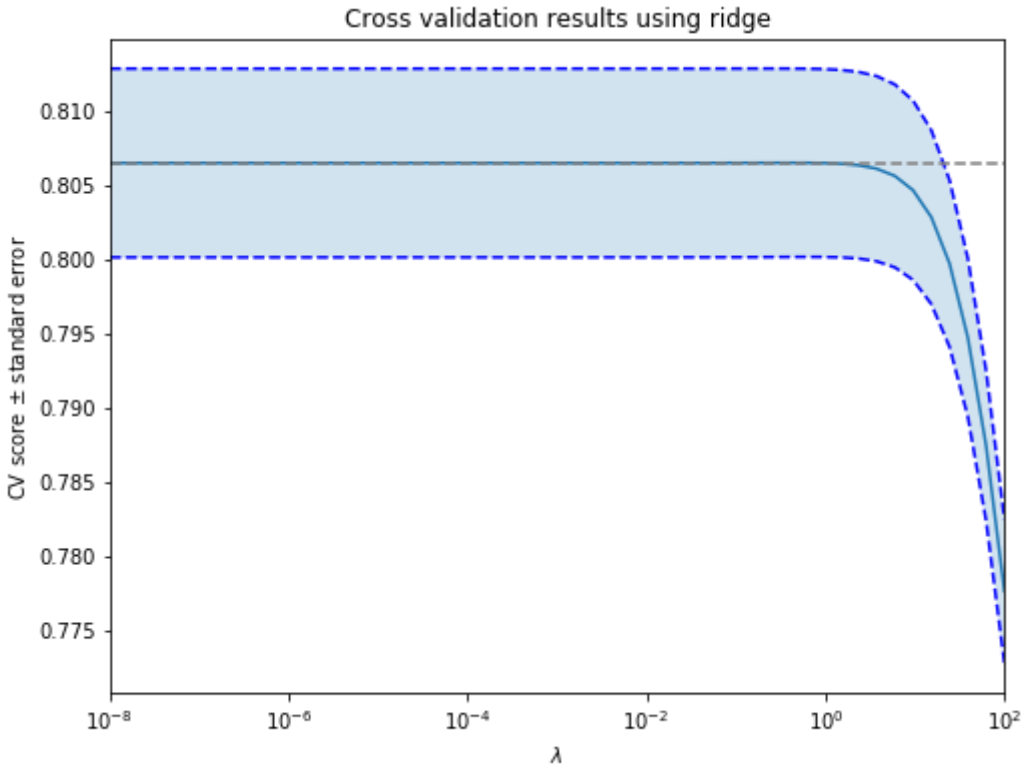


In [17]:
```
# 5-fold cross validation with ridge
np.random.seed(263)

ridge = Ridge(random_state=0, max_iter=100000, tol=100)
alphas = np.logspace(-8, 2, 50)

tuned_parameters = [{"alpha": alphas}]
n_folds = 5

clf = GridSearchCV(ridge, tuned_parameters, cv=n_folds)
clf.fit(X_train, y_train)
scores = clf.cv_results_['mean_test_score']
scores_std = clf.cv_results_['std_test_score']

# Best hyperparameter determined by ridge
print("The best hyperparameter chosen by ridge is {}.".format(clf.best_p
arams_["alpha"]))
```

```
The best hyperparameter chosen by ridge is 0.35564803062231287.
```

In [18]:
```
# Plot the cross valiadation results
plt.figure().set_size_inches(8, 6)
plt.semilogx(alphas, scores)

# Plot error lines showing +/- standard errors of the scores
std_error = scores_std / np.sqrt(n_folds)

plt.semilogx(alphas, scores + std_error, "b--")
plt.semilogx(alphas, scores - std_error, "b--")

# Fill in
plt.fill_between(alphas, scores + std_error, scores - std_error, alpha=
0.2)

plt.title("Cross validation results using ridge")
plt.xlabel(r"$\lambda$")
plt.ylabel(r"CV score $\pm$ standard error")
plt.axhline(np.max(scores), linestyle="--", color="0.5")
plt.xlim([alphas[0], alphas[-1]])
```

Out[18]: (1e-08, 100.0)



Question 1.4c

In [19]:
```
# Best hyperparameter determined by lasso
lasso_lambda = 0.09610779662375922

# Predict on test data
best_mod = Lasso(alpha=lasso_lambda)
lasso_mod = best_mod.fit(X_test, y_test)
preds = lasso_mod.predict(X_test)

# Calculate MSE
print("The MSE obtained from the lasso predictions are:")
mean_squared_error(y_test, preds)
```

```
The MSE obtained from the lasso predictions are:
```

Out[19]:
```
0.3594596105326516
```

In [20]:
```
# Best hyperparameter determined by ridge
ridge_lambda = 0.49417133613238384

# Predict on test
best_mod = Ridge(alpha=ridge_lambda)
ridge_mod = best_mod.fit(X_test, y_test)
preds = ridge_mod.predict(X_test)

# Calculate MSE
print("The MSE obtained from the ridge predictions are:")
mean_squared_error(y_test, preds)
```

```
The MSE obtained from the ridge predictions are:
```

Out[20]:
```
0.19379058442921912
```

Question 1.5

In [21]:
```
# Donald Trump's features
donald_trump = {"bedrooms": [8], "bathrooms": [25],
    "sqft_living": [50000], "sqft_lot": [225000], "floors": [4],
    "zipcode": [98039], "condition": [10], "grade": [10], "waterfront":
[1], "view": [4], "sqft_above": [37500],
 "sqft_basement": [12500], "yr_built": [1994], "yr_renovated": [2010],
"lat": [47.627606],
 "long": [-122.242054], "sqft_living15": [5000], "sqft_lot15": [40000]}

# Zipcode
zip = train_catzip.iloc[2082:2083]
zip = zip.reset_index(drop=True)

# Standardize and concatenate
donald_trump = pd.DataFrame(data=donald_trump)
donald_trump -= X_test.mean(axis=0)
donald_trump /= X_test.std(axis=0)
donald_trump = pd.concat([donald_trump, zip], axis=1)


# Predictions by lasso
lasso_lambda = 0.00041753189365604
best_mod = Lasso(alpha=lasso_lambda)
lasso_mod = best_mod.fit(X_test, y_test)
preds = lasso_mod.predict(donald_trump)
print("The lasso model predicts Donald Trump's house prices to be:"); pr
int(preds)
```

```
The lasso model predicts Donald Trump's house prices to be:
[37462.62960981]
```

The prediction from the lasso does not look reasonable for Donald Trump's house. This could be due to the fact that the lasso has made the coefficients too sparse.

Question 2.1a

In [22]:
```
# Standardize the training data
X_train = train[["bedrooms", "sqft_living"]]
X_train -= X_train.mean(axis=0)
X_train /= X_train.std(axis=0)

y_train = train[["price"]]
y_train -= y_train.mean(axis=0)
y_train /= y_train.std(axis=0)
```

In [23]:
```
# MLE coefficients (same as OLS coefficients)
fit = LinearRegression().fit(X_train, y_train)

print("The OLS coefficients: {}".format(fit.coef_))
print("R\u00b2: {}".format((fit.score(X_train, y_train))))
```

```
The OLS coefficients: [[-0.14579081  0.78615901]]
R²: 0.5081078864280749
```

Therefore, the MLE coefficients are $\hat{\beta}_1 = -0.1458$ and $\hat{\beta}_2 = 0.7862$. The R^2 is 0.5081.

In [24]:
```
# Use the function below to find the penalized coefficients and correspo
nding hyperparameters
def find_nearest(array, value):
    array = np.asarray(array)
    idx = (np.abs(array - value)).argmin()
    print("The index of lambda that best fits our criteria is:"); print(
idx)
    return array[idx]
```

In [25]:
```
# Lasso coefficients
np.random.seed(263)

eps = 5e-3
print("Computing regularization path using lasso...")
lambdas_lasso, coefs_lasso, _ = enet_path(X_train, y_train,
                                          n_alphas=100, eps=eps,
                                          l1_ratio=1., fit_intercept=Fal
se)

# Define the lasso criteria
l1_norm = 1/2*(abs(fit.coef_[0][0]) + abs(fit.coef_[0][1]))

b1_lasso = coefs_lasso[0][0]
b2_lasso = coefs_lasso[0][1]
b_lasso = abs(b1_lasso) + abs(b2_lasso)

# Find the lasso coefficients that fit the criteria
find_nearest(b_lasso, l1_norm)

print("The lasso coefficients are:"); print(b1_lasso[20]); print(b2_lass
o[20])
print("The corresponding lambda is:"); print(lambdas_lasso[20])
```

```
Computing regularization path using lasso...
The index of lambda that best fits our criteria is:
20
The lasso coefficients are:
0.0
0.4617693659846561
The corresponding lambda is:
0.24093445105402134
```

Therefore, the lasso coefficients are $\hat{\beta}_1 = 0$ and $\hat{\beta}_2 = 0.4618$. The hyperparameter associated with these estimates is $\lambda = 0.2409$.

In [26]:
```
# Ridge coefficients
np.random.seed(263)

eps = 5e-9
print("Computing regularization path using ridge...")
lambdas_ridge, coefs_ridge, _ = enet_path(X_train, y_train,
                                          n_alphas=100, eps=eps,
                                          l1_ratio=1e-05, fit_intercept=
False)

# Define the ridge criteria
l2_norm = 1/2*np.sqrt(fit.coef_[0][0]**2 + fit.coef_[0][1]**2)

b1_ridge = coefs_ridge[0][0]
b2_ridge = coefs_ridge[0][1]
b_ridge = np.sqrt(b1_ridge**2 + b2_ridge**2)

# Find the ridge coefficients that fit the criteria
find_nearest(b_ridge, l2_norm)

print("The ridge coefficients are:"); print(b1_ridge[99]); print(b2_ridg
e[99])
print("The corresponding lambda is:"); print(lambdas_ridge[99])
```

```
Computing regularization path using ridge...
The index of lambda that best fits our criteria is:
60
The ridge coefficients are:
-0.14547982861752207
0.7857049630464638
The corresponding lambda is:
0.0003513366474523075
```

Therefore, the ridge coefficients are $\hat{\beta}_1 = -0.1455$ and $\hat{\beta}_2 = 0.7857$. The hyperparameter associated with these estimates is $\lambda = 0.0004$.

Question 2.1b

In [27]:
```
# Generate the grid
beta1 = np.linspace(-0.5, 1, num=100)
beta2 = np.linspace(-0.5, 1, num=100)

# Every combination of beta1 and beta2
betas = list(product(beta1, beta2))
```

In [28]:

```
# Calculate the mean-squared error for the discretized coefficient grid
def find_mse(b1, b2, print_betas):
    X1 = X_train["bedrooms"].apply(lambda x: x*b1)
    X2 = X_train["sqft_living"].apply(lambda x: x*b2)
    fhat = X1 + X2
    mse = pd.DataFrame.sum((y_train["price"] - fhat)**2, axis=0)/len(y_t
rain)

    if print_betas==True:
        return (b1, b2, mse)
    else:
        return mse

mse = [find_mse(*beta, print_betas=True) for beta in betas]

print("Below is the discretized coefficient grid with their correspondin
g MSE values")
mse
```

Below is the discretized coefficient grid with their corresponding MSE values

Out[28]:
```
[(-0.5, -0.5, 2.7928412330039376),
 (-0.5, -0.48484848484848486, 2.7479561723261074),
 (-0.5, -0.4696969696969697, 2.7035302181229235),
 (-0.5, -0.45454545454545453, 2.6595633703943844),
 (-0.5, -0.4393939393939394, 2.616055629140491),
 (-0.5, -0.42424242424242425, 2.5730069943612435),
 (-0.5, -0.40909090909090906, 2.5304174660566408),
 (-0.5, -0.3939393939393939, 2.4882870442266842),
 (-0.5, -0.3787878787878788, 2.446615728871373),
 (-0.5, -0.3636363636363636, 2.4054035199907076),
 (-0.5, -0.3484848484848485, 2.3646504175846874),
 (-0.5, -0.3333333333333333, 2.324356421653312),
 (-0.5, -0.3181818181818182, 2.2845215321965835),
 (-0.5, -0.30303030303030304, 2.2451457492145),
 (-0.5, -0.28787878787878785, 2.2062290727070617),
 (-0.5, -0.2727272727272727, 2.1677715026742694),
 (-0.5, -0.25757575757575757, 2.1297730391161225),
 (-0.5, -0.24242424242424243, 2.0922336820326217),
 (-0.5, -0.2272727272727273, 2.0551534314237654),
 (-0.5, -0.2121212121212121, 2.0185322872895552),
 (-0.5, -0.19696969696969696, 1.9823702496299906),
 (-0.5, -0.18181818181818182, 1.9466673184450713),
 (-0.5, -0.16666666666666663, 1.9114234937347976),
 (-0.5, -0.1515151515151515, 1.8766387754991698),
 (-0.5, -0.13636363636363635, 1.842313163738187),
 (-0.5, -0.12121212121212122, 1.8084466584518504),
 (-0.5, -0.10606060606060608, 1.7750392596401587),
 (-0.5, -0.09090909090909088, 1.7420909673031129),
 (-0.5, -0.07575757575757575, 1.7096017814407125),
 (-0.5, -0.06060606060606061, 1.6775717020529577),
 (-0.5, -0.045454545454545414, 1.6460007291398484),
 (-0.5, -0.030303030303030276, 1.6148888627013844),
 (-0.5, -0.015151515151515138, 1.5842361027375664),
 (-0.5, 0.0, 1.5540424492483937),
 (-0.5, 0.015151515151515138, 1.5243079022338666),
 (-0.5, 0.030303030303030276, 1.495032461693985),
 (-0.5, 0.045454545454545414, 1.466216127628749),
 (-0.5, 0.06060606060606066, 1.4378589000381583),
 (-0.5, 0.0757575757575758, 1.4099607789222135),
 (-0.5, 0.09090909090909094, 1.382521764280914),
 (-0.5, 0.10606060606060608, 1.3555418561142603),
 (-0.5, 0.12121212121212122, 1.3290210544222518),
 (-0.5, 0.13636363636363635, 1.3029593592048891),
 (-0.5, 0.1515151515151515, 1.277356770462172),
 (-0.5, 0.16666666666666674, 1.2522132881941),
 (-0.5, 0.18181818181818188, 1.2275289124006739),
 (-0.5, 0.19696969696969702, 1.2033036430818933),
 (-0.5, 0.21212121212121215, 1.1795374802377583),
 (-0.5, 0.2272727272727273, 1.1562304238682688),
 (-0.5, 0.24242424242424243, 1.1333824739734248),
 (-0.5, 0.25757575757575757, 1.1109936305532262),
 (-0.5, 0.2727272727272727, 1.0890638936076735),
 (-0.5, 0.28787878787878785, 1.067593263136766),
 (-0.5, 0.3030303030303031, 1.0465817391405041),
 (-0.5, 0.31818181818181823, 1.0260293216188878),
 (-0.5, 0.33333333333333337, 1.005936010571917),
 (-0.5, 0.3484848484848485, 0.9863018059995919),
```

```
(-0.5, 0.36363636363636365, 0.9671267079019122),
(-0.5, 0.3787878787878788, 0.9484107162788783),
(-0.5, 0.3939393939393939, 0.9301538311304896),
(-0.5, 0.40909090909090917, 0.9123560524567464),
(-0.5, 0.4242424242424243, 0.895017380257649),
(-0.5, 0.43939393939393945, 0.8781378145331971),
(-0.5, 0.4545454545454546, 0.8617173552833907),
(-0.5, 0.4696969696969697, 0.8457560025082298),
(-0.5, 0.48484848484848486, 0.8302537562077145),
(-0.5, 0.5, 0.8152106163818448),
(-0.5, 0.5151515151515151, 0.8006265830306205),
(-0.5, 0.5303030303030303, 0.7865016561540418),
(-0.5, 0.5454545454545454, 0.7728358357521087),
(-0.5, 0.5606060606060606, 0.759629121824821),
(-0.5, 0.5757575757575757, 0.746881514372179),
(-0.5, 0.5909090909090908, 0.7345930133941824),
(-0.5, 0.6060606060606062, 0.7227636188908313),
(-0.5, 0.6212121212121213, 0.7113933308621259),
(-0.5, 0.6363636363636365, 0.7004821493080661),
(-0.5, 0.6515151515151516, 0.6900300742286518),
(-0.5, 0.6666666666666667, 0.6800371056238829),
(-0.5, 0.6818181818181819, 0.6705032434937596),
(-0.5, 0.696969696969697, 0.6614284878382819),
(-0.5, 0.7121212121212122, 0.6528128386574498),
(-0.5, 0.7272727272727273, 0.6446562959512631),
(-0.5, 0.7424242424242424, 0.636958859719722),
(-0.5, 0.7575757575757576, 0.6297205299628265),
(-0.5, 0.7727272727272727, 0.6229413066805763),
(-0.5, 0.7878787878787878, 0.6166211898729718),
(-0.5, 0.803030303030303, 0.6107601795400128),
(-0.5, 0.8181818181818181, 0.6053582756816995),
(-0.5, 0.8333333333333335, 0.6004154782980315),
(-0.5, 0.8484848484848486, 0.5959317873890093),
(-0.5, 0.8636363636363638, 0.5919072029546325),
(-0.5, 0.8787878787878789, 0.5883417249949012),
(-0.5, 0.893939393939394, 0.5852353535098156),
(-0.5, 0.9090909090909092, 0.5825880884993755),
(-0.5, 0.9242424242424243, 0.5803999299635809),
(-0.5, 0.9393939393939394, 0.5786708779024318),
(-0.5, 0.9545454545454546, 0.5774009323159284),
(-0.5, 0.9696969696969697, 0.5765900932040703),
(-0.5, 0.9848484848484849, 0.5762383605668578),
(-0.5, 1.0, 0.576345734404291),
(-0.48484848484848486, -0.5, 2.7600333912012216),
(-0.48484848484848486, -0.48484848484848486, 2.7154110871931865),
(-0.48484848484848486, -0.4696969696969697, 2.671247889659796),
(-0.48484848484848486, -0.45454545454545453, 2.6275437986010526),
(-0.48484848484848486, -0.4393939393939394, 2.584298814016954),
(-0.48484848484848486, -0.42424242424242425, 2.541512935907501),
(-0.48484848484848486, -0.40909090909090906, 2.499186164272693),
(-0.48484848484848486, -0.3939393939393939, 2.4573184991125308),
(-0.48484848484848486, -0.3787878787878788, 2.415909940427014),
(-0.48484848484848486, -0.36363636363636365, 2.3749604882161437),
(-0.48484848484848486, -0.3484848484848485, 2.334470142479918),
(-0.48484848484848486, -0.3333333333333333, 2.2944389032183383),
(-0.48484848484848486, -0.3181818181818182, 2.254866770431404),
(-0.48484848484848486, -0.30303030303030304, 2.215753744119115),
```

(-0.48484848484848486, -0.28787878787878785, 2.1770998242814716),
(-0.48484848484848486, -0.2727272727272727, 2.138905010918474),
(-0.48484848484848486, -0.25757575757575757, 2.101169304030122),
(-0.48484848484848486, -0.24242424242424243, 2.0638927036164154),
(-0.48484848484848486, -0.2272727272727273, 2.0270752096773546),
(-0.48484848484848486, -0.2121212121212121, 1.9907168222129386),
(-0.48484848484848486, -0.19696969696969696, 1.9548175412231685),
(-0.48484848484848486, -0.18181818181818182, 1.9193773667080443),
(-0.48484848484848486, -0.16666666666666663, 1.884396298667565),
(-0.48484848484848486, -0.1515151515151515, 1.8498743371017319),
(-0.48484848484848486, -0.13636363636363635, 1.815811482010544),
(-0.48484848484848486, -0.12121212121212122, 1.7822077333940018),
(-0.48484848484848486, -0.10606060606060608, 1.7490630912521048),
(-0.48484848484848486, -0.09090909090909088, 1.7163775555848537),
(-0.48484848484848486, -0.07575757575757575, 1.684151126392248),
(-0.48484848484848486, -0.06060606060606061, 1.6523838036742882),
(-0.48484848484848486, -0.045454545454545414, 1.6210755874309732),
(-0.48484848484848486, -0.030303030303030276, 1.5902264776623043),
(-0.48484848484848486, -0.015151515151515138, 1.5598364743682807),
(-0.48484848484848486, 0.0, 1.5299055775489028),
(-0.48484848484848486, 0.015151515151515138, 1.5004337872041702),
(-0.48484848484848486, 0.030303030303030276, 1.4714211033340836),
(-0.48484848484848486, 0.045454545454545414, 1.4428675259386423),
(-0.48484848484848486, 0.06060606060606066, 1.4147730550178463),
(-0.48484848484848486, 0.0757575757575758, 1.3871376905716961),
(-0.48484848484848486, 0.09090909090909094, 1.3599614326001914),
(-0.48484848484848486, 0.10606060606060608, 1.3332442811033323),
(-0.48484848484848486, 0.12121212121212122, 1.3069862360811184),
(-0.48484848484848486, 0.13636363636363635, 1.2811872975335508),
(-0.48484848484848486, 0.1515151515151515, 1.2558474654606282),
(-0.48484848484848486, 0.16666666666666674, 1.2309667398623507),
(-0.48484848484848486, 0.18181818181818188, 1.2065451207387197),
(-0.48484848484848486, 0.19696969696969702, 1.1825826080897335),
(-0.48484848484848486, 0.21212121212121215, 1.1590792019153933),
(-0.48484848484848486, 0.2272727272727273, 1.1360349022156986),
(-0.48484848484848486, 0.24242424242424243, 1.113449708990649),
(-0.48484848484848486, 0.25757575757575757, 1.0913236222402454),
(-0.48484848484848486, 0.2727272727272727, 1.069656641964487),
(-0.48484848484848486, 0.28787878787878785, 1.0484487681633745),
(-0.48484848484848486, 0.3030303030303031, 1.0277000008369073),
(-0.48484848484848486, 0.31818181818181823, 1.0074103399850858),
(-0.48484848484848486, 0.33333333333333337, 0.9875797856079097),
(-0.48484848484848486, 0.3484848484848485, 0.9682083377053792),
(-0.48484848484848486, 0.36363636363636365, 0.9492959962774943),
(-0.48484848484848486, 0.3787878787878788, 0.9308427613242549),
(-0.48484848484848486, 0.3939393939393939, 0.912848632845661),
(-0.48484848484848486, 0.40909090909090917, 0.8953136108417126),
(-0.48484848484848486, 0.4242424242424243, 0.8782376953124098),
(-0.48484848484848486, 0.43939393939393945, 0.8616208862577526),
(-0.48484848484848486, 0.4545454545454546, 0.845463183677741),
(-0.48484848484848486, 0.4696969696969697, 0.8297645875723747),
(-0.48484848484848486, 0.48484848484848486, 0.8145250979416542),
(-0.48484848484848486, 0.5, 0.7997447147855791),
(-0.48484848484848486, 0.5151515151515151, 0.7854234381041495),
(-0.48484848484848486, 0.5303030303030303, 0.7715612678973656),
(-0.48484848484848486, 0.5454545454545454, 0.7581582041652271),
(-0.48484848484848486, 0.5606060606060606, 0.7452142469077341),

(-0.48484848484848486, 0.5757575757575757, 0.7327293961248869),
(-0.48484848484848486, 0.5909090909090908, 0.720703651816685),
(-0.48484848484848486, 0.6060606060606062, 0.7091370139831287),
(-0.48484848484848486, 0.6212121212121213, 0.6980294826242178),
(-0.48484848484848486, 0.6363636363636365, 0.6873810577399527),
(-0.48484848484848486, 0.6515151515151516, 0.677191739330333),
(-0.48484848484848486, 0.6666666666666667, 0.6674615273953589),
(-0.48484848484848486, 0.6818181818181819, 0.6581904219350303),
(-0.48484848484848486, 0.696969696969697, 0.6493784229493473),
(-0.48484848484848486, 0.7121212121212122, 0.6410255304383098),
(-0.48484848484848486, 0.7272727272727273, 0.6331317444019179),
(-0.48484848484848486, 0.7424242424242424, 0.6256970648401715),
(-0.48484848484848486, 0.7575757575757576, 0.6187214917530707),
(-0.48484848484848486, 0.7727272727272727, 0.6122050251406154),
(-0.48484848484848486, 0.7878787878787878, 0.6061476650028055),
(-0.48484848484848486, 0.803030303030303, 0.6005494113396413),
(-0.48484848484848486, 0.8181818181818181, 0.5954102641511226),
(-0.48484848484848486, 0.8333333333333335, 0.5907302234372493),
(-0.48484848484848486, 0.8484848484848486, 0.5865092891980218),
(-0.48484848484848486, 0.8636363636363638, 0.5827474614334398),
(-0.48484848484848486, 0.8787878787878789, 0.5794447401435032),
(-0.48484848484848486, 0.893939393939394, 0.5766011253282123),
(-0.48484848484848486, 0.9090909090909092, 0.5742166169875668),
(-0.48484848484848486, 0.9242424242424243, 0.5722912151215668),
(-0.48484848484848486, 0.9393939393939394, 0.5708249197302124),
(-0.48484848484848486, 0.9545454545454546, 0.5698177308135037),
(-0.48484848484848486, 0.9696969696969697, 0.5692696483714403),
(-0.48484848484848486, 0.9848484848484849, 0.5691806724040227),
(-0.48484848484848486, 1.0, 0.5695508029112505),
(-0.4696969696969697, -0.5, 2.7276846558731513),
(-0.4696969696969697, -0.48484848484848486, 2.6833251085349112),
(-0.4696969696969697, -0.4696969696969697, 2.639424667671315),
(-0.4696969696969697, -0.45454545454545453, 2.595983333282366),
(-0.4696969696969697, -0.4393939393939394, 2.553001105368062),
(-0.4696969696969697, -0.42424242424242425, 2.5104779839284035),
(-0.4696969696969697, -0.40909090909090906, 2.468413968963391),
(-0.4696969696969697, -0.3939393939393939, 2.4268090604730235),
(-0.4696969696969697, -0.3787878787878788, 2.3856632584573014),
(-0.4696969696969697, -0.36363636363636365, 2.344976562916225),
(-0.4696969696969697, -0.3484848484848485, 2.304748973849795),
(-0.4696969696969697, -0.3333333333333333, 2.2649804912580094),
(-0.4696969696969697, -0.3181818181818182, 2.22567111514087),
(-0.4696969696969697, -0.30303030303030304, 2.1868208454983757),
(-0.4696969696969697, -0.28787878787878785, 2.148429682330527),
(-0.4696969696969697, -0.2727272727272727, 2.1104976256373242),
(-0.4696969696969697, -0.25757575757575757, 2.073024675418767),
(-0.4696969696969697, -0.24242424242424243, 2.036010831674855),
(-0.4696969696969697, -0.2272727272727273, 1.9994560944055884),
(-0.4696969696969697, -0.2121212121212121, 1.9633604636109676),
(-0.4696969696969697, -0.1969696969696969, 1.9277239392909922),
(-0.4696969696969697, -0.18181818181818182, 1.8925465214456625),
(-0.4696969696969697, -0.16666666666666663, 1.857828210074978),
(-0.4696969696969697, -0.1515151515151515, 1.8235690051789395),
(-0.4696969696969697, -0.13636363636363635, 1.7897689067575464),
(-0.4696969696969697, -0.12121212121212122, 1.7564279148107986),
(-0.4696969696969697, -0.10606060606060608, 1.7235460293386968),
(-0.4696969696969697, -0.09090909090909088, 1.6911232503412401),

(-0.4696969696969697, -0.07575757575757575, 1.6591595778184294),
(-0.4696969696969697, -0.06060606060606061, 1.6276550117702637),
(-0.4696969696969697, -0.045454545454545414, 1.59660955219674 4),
(-0.4696969696969697, -0.030303030303030276, 1.5660231990978697),
(-0.4696969696969697, -0.015151515151515138, 1.5358959524736406),
(-0.4696969696969697, 0.0, 1.5062278123240576),
(-0.4696969696969697, 0.015151515151515138, 1.4770187786491198),
(-0.4696969696969697, 0.030303030303030276, 1.4482688514488278),
(-0.4696969696969697, 0.045454545454545414, 1.419978030723181),
(-0.4696969696969697, 0.06060606060606066, 1.39214631647218),
(-0.4696969696969697, 0.0757575757575758, 1.3647737086958243),
(-0.4696969696969697, 0.09090909090909094, 1.3378602073941146),
(-0.4696969696969697, 0.10606060606060608, 1.3114058125670498),
(-0.4696969696969697, 0.12121212121212122, 1.285410524214631),
(-0.4696969696969697, 0.13636363636363635, 1.2598743423368575),
(-0.4696969696969697, 0.1515151515151515, 1.23479726693373),
(-0.4696969696969697, 0.16666666666666674, 1.2101792980052473),
(-0.4696969696969697, 0.18181818181818188, 1.1860204355514108),
(-0.4696969696969697, 0.19696969696969702, 1.1623206795722194),
(-0.4696969696969697, 0.21212121212121215, 1.139080030067674),
(-0.4696969696969697, 0.2272727272727273, 1.1162984870377737),
(-0.4696969696969697, 0.24242424242424243, 1.0939760504825191),
(-0.4696969696969697, 0.25757575757575757, 1.07211272040191),
(-0.4696969696969697, 0.2727272727272727, 1.0507084967959466),
(-0.4696969696969697, 0.28787878787878785, 1.0297633796646286),
(-0.4696969696969697, 0.3030303030303031, 1.0092773690079562),
(-0.4696969696969697, 0.31818181818181823, 0.9892504648259293),
(-0.4696969696969697, 0.33333333333333337, 0.969682667118548),
(-0.4696969696969697, 0.3484848484848485, 0.950573975885812),
(-0.4696969696969697, 0.36363636363636365, 0.9319243911277219),
(-0.4696969696969697, 0.3787878787878788, 0.9137339128442772),
(-0.4696969696969697, 0.3939393939393939, 0.896002541035478),
(-0.4696969696969697, 0.40909090909090917, 0.8787302757013243),
(-0.4696969696969697, 0.4242424242424243, 0.8619171168418163),
(-0.4696969696969697, 0.43939393939393945, 0.8455630644569538),
(-0.4696969696969697, 0.4545454545454546, 0.8296681185467368),
(-0.4696969696969697, 0.4696969696969697, 0.8142322791111652),
(-0.4696969696969697, 0.48484848484848486, 0.7992555461502393),
(-0.4696969696969697, 0.5, 0.784737919663959),
(-0.4696969696969697, 0.5151515151515151, 0.7706793996523241),
(-0.4696969696969697, 0.5303030303030303, 0.7570799861153349),
(-0.4696969696969697, 0.5454545454545454, 0.7439396790529911),
(-0.4696969696969697, 0.5606060606060606, 0.731258478465293),
(-0.4696969696969697, 0.5757575757575757, 0.7190363843522404),
(-0.4696969696969697, 0.5909090909090908, 0.7072733967138332),
(-0.4696969696969697, 0.6060606060606062, 0.6959695155500716),
(-0.4696969696969697, 0.6212121212121213, 0.6851247408609554),
(-0.4696969696969697, 0.6363636363636365, 0.6747390726464849),
(-0.4696969696969697, 0.6515151515151516, 0.6648125109066599),
(-0.4696969696969697, 0.6666666666666667, 0.6553450556414806),
(-0.4696969696969697, 0.6818181818181819, 0.6463367068509468),
(-0.4696969696969697, 0.696969696969697, 0.6377874645350584),
(-0.4696969696969697, 0.7121212121212122, 0.6296973286938157),
(-0.4696969696969697, 0.7272727272727273, 0.6220662993272184),
(-0.4696969696969697, 0.7424242424242424, 0.6148943764352667),
(-0.4696969696969697, 0.7575757575757576, 0.6081815600179604),
(-0.4696969696969697, 0.7727272727272727, 0.6019278500752999),

(-0.4696969696969697, 0.7878787878787878, 0.5961332466072848),
(-0.4696969696969697, 0.803030303030303, 0.5907977496139153),
(-0.4696969696969697, 0.8181818181818181, 0.5859213590951913),
(-0.4696969696969697, 0.8333333333333335, 0.5815040750511128),
(-0.4696969696969697, 0.8484848484848486, 0.5775458974816798),
(-0.4696969696969697, 0.8636363636363638, 0.5740468263868925),
(-0.4696969696969697, 0.8787878787878789, 0.5710068617667506),
(-0.4696969696969697, 0.893939393939394, 0.5684260036212544),
(-0.4696969696969697, 0.9090909090909092, 0.5663042519504037),
(-0.4696969696969697, 0.9242424242424243, 0.5646416067541985),
(-0.4696969696969697, 0.9393939393939394, 0.5634380680326389),
(-0.4696969696969697, 0.9545454545454546, 0.5626936357857247),
(-0.4696969696969697, 0.9696969696969697, 0.5624083100134561),
(-0.4696969696969697, 0.9848484848484849, 0.5625820907158331),
(-0.4696969696969697, 1.0, 0.5632149778928556),
(-0.45454545454545453, -0.5, 2.695795027019726),
(-0.45454545454545453, -0.48484848484848486, 2.65169823635128),
(-0.45454545454545453, -0.4696969696969697, 2.60806055215748),
(-0.45454545454545453, -0.45454545454545453, 2.5648819744383244),
(-0.45454545454545453, -0.4393939393939394, 2.522162503193816),
(-0.45454545454545453, -0.42424242424242425, 2.479902138423952),
(-0.45454545454545453, -0.40909090909090906, 2.4381008801287334),
(-0.45454545454545453, -0.3939393939393939, 2.3967587283081615),
(-0.45454545454545453, -0.3787878787878788, 2.3558756829622345),
(-0.45454545454545453, -0.36363636363636365, 2.315451744090953),
(-0.45454545454545453, -0.3484848484848485, 2.275486911694317),
(-0.45454545454545453, -0.3333333333333333, 2.235981185772326),
(-0.45454545454545453, -0.3181818181818182, 2.1969345663249813),
(-0.45454545454545453, -0.30303030303030304, 2.1583470533522817),
(-0.45454545454545453, -0.28787878787878785, 2.120218646854228),
(-0.45454545454545453, -0.2727272727272727, 2.08254934683082),
(-0.45454545454545453, -0.25757575757575757, 2.045339153282057),
(-0.45454545454545453, -0.24242424242424243, 2.0085880662079396),
(-0.45454545454545453, -0.2272727272727273, 1.972296085608468),
(-0.45454545454545453, -0.2121212121212121, 1.9364632114836418),
(-0.45454545454545453, -0.19696969696969696, 1.9010894438334611),
(-0.45454545454545453, -0.18181818181818182, 1.866174782657926),
(-0.45454545454545453, -0.16666666666666663, 1.8317192279570365),
(-0.45454545454545453, -0.1515151515151515, 1.7977227797307926),
(-0.45454545454545453, -0.13636363636363635, 1.7641854379791941),
(-0.45454545454545453, -0.12121212121212122, 1.7311072027022414),
(-0.45454545454545453, -0.10606060606060608, 1.6984880738999342),
(-0.45454545454545453, -0.09090909090909088, 1.6663280515722723),
(-0.45454545454545453, -0.07575757575757575, 1.634627135719256),
(-0.45454545454545453, -0.06060606060606061, 1.6033853263408853),
(-0.45454545454545453, -0.045454545454545414, 1.57260262343716),
(-0.45454545454545453, -0.030303030303030276, 1.5422790270080802),
(-0.45454545454545453, -0.015151515151515138, 1.5124145370536461),
(-0.45454545454545453, 0.0, 1.4830091535738579),
(-0.45454545454545453, 0.015151515151515138, 1.4540628765687147),
(-0.45454545454545453, 0.030303030303030276, 1.4255757060382173),
(-0.45454545454545453, 0.045454545454545414, 1.3975476419823651),
(-0.45454545454545453, 0.06060606060606066, 1.3699786844011588),
(-0.45454545454545453, 0.0757575757575758, 1.3428688332945982),
(-0.45454545454545453, 0.09090909090909094, 1.3162180886626826),
(-0.45454545454545453, 0.10606060606060608, 1.290026450505413),
(-0.45454545454545453, 0.12121212121212122, 1.264293918822789),

(-0.45454545454545453, 0.13636363636363635, 1.2390204936148101),
(-0.45454545454545453, 0.1515151515151515, 1.2142061748814768),
(-0.45454545454545453, 0.16666666666666674, 1.189850962622789),
(-0.45454545454545453, 0.18181818181818188, 1.1659548568387474),
(-0.45454545454545453, 0.19696969696969702, 1.1425178575293506),
(-0.45454545454545453, 0.21212121212121215, 1.1195399646945998),
(-0.45454545454545453, 0.2272727272727273, 1.0970211783344943),
(-0.45454545454545453, 0.24242424242424243, 1.0749614984490345),
(-0.45454545454545453, 0.25757575757575757, 1.0533609250382203),
(-0.45454545454545453, 0.2727272727272727, 1.0322194581020514),
(-0.45454545454545453, 0.28787878787878785, 1.0115370976405282),
(-0.45454545454545453, 0.3030303030303031, 0.9913138436536504),
(-0.45454545454545453, 0.31818181818181823, 0.9715496961414183),
(-0.45454545454545453, 0.33333333333333337, 0.9522446551038315),
(-0.45454545454545453, 0.3484848484848485, 0.9333987205408905),
(-0.45454545454545453, 0.36363636363636365, 0.915011892452595),
(-0.45454545454545453, 0.3787878787878788, 0.8970841708389449),
(-0.45454545454545453, 0.3939393939393939, 0.8796155556999405),
(-0.45454545454545453, 0.40909090909090917, 0.8626060470355814),
(-0.45454545454545453, 0.4242424242424243, 0.8460556448458681),
(-0.45454545454545453, 0.43939393939393945, 0.8299643491308002),
(-0.45454545454545453, 0.4545454545454546, 0.814332159890378),
(-0.45454545454545453, 0.4696969696969697, 0.7991590771246013),
(-0.45454545454545453, 0.48484848484848486, 0.7844451008334701),
(-0.45454545454545453, 0.5, 0.7701902310169845),
(-0.45454545454545453, 0.5151515151515151, 0.7563944676751444),
(-0.45454545454545453, 0.5303030303030303, 0.7430578108079496),
(-0.45454545454545453, 0.5454545454545454, 0.7301802604154006),
(-0.45454545454545453, 0.5606060606060606, 0.7177618164974972),
(-0.45454545454545453, 0.5757575757575757, 0.7058024790542392),
(-0.45454545454545453, 0.5909090909090908, 0.6943022480856267),
(-0.45454545454545453, 0.6060606060606062, 0.6832611235916598),
(-0.45454545454545453, 0.6212121212121213, 0.6726791055723383),
(-0.45454545454545453, 0.6363636363636365, 0.6625561940276627),
(-0.45454545454545453, 0.6515151515151516, 0.6528923889576325),
(-0.45454545454545453, 0.6666666666666667, 0.6436876903622477),
(-0.45454545454545453, 0.6818181818181819, 0.6349420982415086),
(-0.45454545454545453, 0.696969696969697, 0.626655612595415),
(-0.45454545454545453, 0.7121212121212122, 0.6188282334239669),
(-0.45454545454545453, 0.7272727272727273, 0.6114599607271645),
(-0.45454545454545453, 0.7424242424242424, 0.6045507945050073),
(-0.45454545454545453, 0.7575757575757576, 0.5981007347574959),
(-0.45454545454545453, 0.7727272727272727, 0.5921097814846299),
(-0.45454545454545453, 0.7878787878787878, 0.5865779346864096),
(-0.45454545454545453, 0.803030303030303, 0.5815051943628348),
(-0.45454545454545453, 0.8181818181818181, 0.5768915605139054),
(-0.45454545454545453, 0.8333333333333335, 0.5727370331396215),
(-0.45454545454545453, 0.8484848484848486, 0.5690416122399834),
(-0.45454545454545453, 0.8636363636363638, 0.5658052978149908),
(-0.45454545454545453, 0.8787878787878789, 0.5630280898646437),
(-0.45454545454545453, 0.893939393939394, 0.560709988388942),
(-0.45454545454545453, 0.9090909090909092, 0.5588509933878861),
(-0.45454545454545453, 0.9242424242424243, 0.5574511048614756),
(-0.45454545454545453, 0.9393939393939394, 0.5565103228097106),
(-0.45454545454545453, 0.9545454545454546, 0.5560286472325913),
(-0.45454545454545453, 0.9696969696969697, 0.5560060781301173),
(-0.45454545454545453, 0.9848484848484849, 0.5564426155022891),

(-0.45454545454545453, 1.0, 0.5573382593491062),
(-0.4393939393939394, -0.5, 2.6643645046409463),
(-0.4393939393939394, -0.48484848484848486, 2.620530470642296),
(-0.4393939393939394, -0.4696969696969697, 2.5771555431182906),
(-0.4393939393939394, -0.45454545454545453, 2.53423972206893),
(-0.4393939393939394, -0.4393939393939394, 2.4917830074942158),
(-0.4393939393939394, -0.42424242424242425, 2.4497853993941465),
(-0.4393939393939394, -0.40909090909090906, 2.4082468977687226),
(-0.4393939393939394, -0.3939393939393939, 2.367167502617945),
(-0.4393939393939394, -0.3787878787878788, 2.326547213941813),
(-0.4393939393939394, -0.36363636363636365, 2.286386031740326),
(-0.4393939393939394, -0.3484848484848485, 2.2466839560134844),
(-0.4393939393939394, -0.3333333333333333, 2.2074409867612883),
(-0.4393939393939394, -0.3181818181818182, 2.1686571239837384),
(-0.4393939393939394, -0.30303030303030304, 2.130332367680834),
(-0.4393939393939394, -0.28787878787878785, 2.092466717852574),
(-0.4393939393939394, -0.2727272727272727, 2.055060174498961),
(-0.4393939393939394, -0.25757575757575757, 2.018112737619993),
(-0.4393939393939394, -0.24242424242424243, 1.9816244072156703),
(-0.4393939393939394, -0.2272727272727273, 1.9455951832859932),
(-0.4393939393939394, -0.2121212121212121, 1.9100250658309617),
(-0.4393939393939394, -0.19696969696969696, 1.874914054850576),
(-0.4393939393939394, -0.18181818181818182, 1.8402621503448358),
(-0.4393939393939394, -0.16666666666666663, 1.8060693523137408),
(-0.4393939393939394, -0.1515151515151515, 1.7723356607572913),
(-0.4393939393939394, -0.13636363636363635, 1.7390610756754878),
(-0.4393939393939394, -0.12121212121212122, 1.7062455970683295),
(-0.4393939393939394, -0.10606060606060608, 1.673889224935817),
(-0.4393939393939394, -0.09090909090909088, 1.6419919592779502),
(-0.4393939393939394, -0.07575757575757575, 1.6105538000947284),
(-0.4393939393939394, -0.06060606060606061, 1.5795747473861521),
(-0.4393939393939394, -0.045454545454545414, 1.5490548011522218),
(-0.4393939393939394, -0.030303030303030276, 1.5189939613929366),
(-0.4393939393939394, -0.015151515151515138, 1.4893922281082976),
(-0.4393939393939394, 0.0, 1.4602496012983037),
(-0.4393939393939394, 0.015151515151515138, 1.4315660809629551),
(-0.4393939393939394, 0.030303030303030276, 1.4033416671022525),
(-0.4393939393939394, 0.045454545454545414, 1.3755763597161956),
(-0.4393939393939394, 0.06060606060606066, 1.3482701588047834),
(-0.4393939393939394, 0.0757575757575758, 1.3214230643680174),
(-0.4393939393939394, 0.09090909090909094, 1.295035076405897),
(-0.4393939393939394, 0.10606060606060608, 1.2691061949184217),
(-0.4393939393939394, 0.12121212121212122, 1.2436364199055923),
(-0.4393939393939394, 0.13636363636363635, 1.2186257513674084),
(-0.4393939393939394, 0.1515151515151515, 1.19407418930387),
(-0.4393939393939394, 0.16666666666666674, 1.169981733714977),
(-0.4393939393939394, 0.18181818181818188, 1.1463483846007294),
(-0.4393939393939394, 0.19696969696969702, 1.1231741419611279),
(-0.4393939393939394, 0.21212121212121215, 1.1004590057961716),
(-0.4393939393939394, 0.2272727272727273, 1.0782029761058607),
(-0.4393939393939394, 0.24242424242424243, 1.0564060528901957),
(-0.4393939393939394, 0.25757575757575757, 1.035068236149176),
(-0.4393939393939394, 0.2727272727272727, 1.014189525882802),
(-0.4393939393939394, 0.28787878787878785, 0.9937699220910735),
(-0.4393939393939394, 0.3030303030303031, 0.9738094247739904),
(-0.4393939393939394, 0.31818181818181823, 0.9543080339315528),
(-0.4393939393939394, 0.33333333333333337, 0.935265749563761),

(-0.4393939393939394, 0.3484848484848485, 0.9166825716706145),
(-0.4393939393939394, 0.36363636363636365, 0.8985585002521137),
(-0.4393939393939394, 0.3787878787878788, 0.8808935353082584),
(-0.4393939393939394, 0.3939393939393939, 0.8636876768390486),
(-0.4393939393939394, 0.40909090909090917, 0.8469409248444844),
(-0.4393939393939394, 0.4242424242424243, 0.8306532793245657),
(-0.4393939393939394, 0.43939393939393945, 0.8148247402792925),
(-0.4393939393939394, 0.4545454545454546, 0.799455307708665),
(-0.4393939393939394, 0.4696969696969697, 0.7845449816126828),
(-0.4393939393939394, 0.48484848484848486, 0.7700937619913465),
(-0.4393939393939394, 0.5, 0.7561016488446555),
(-0.4393939393939394, 0.5151515151515151, 0.7425686421726101),
(-0.4393939393939394, 0.5303030303030303, 0.7294947419752102),
(-0.4393939393939394, 0.5454545454545454, 0.7168799482524558),
(-0.4393939393939394, 0.5606060606060606, 0.7047242610043469),
(-0.4393939393939394, 0.5757575757575757, 0.6930276802308838),
(-0.4393939393939394, 0.5909090909090908, 0.6817902059320662),
(-0.4393939393939394, 0.6060606060606062, 0.6710118381078939),
(-0.4393939393939394, 0.6212121212121213, 0.6606925767583672),
(-0.4393939393939394, 0.6363636363636365, 0.650832421883486),
(-0.4393939393939394, 0.6515151515151516, 0.6414313734832504),
(-0.4393939393939394, 0.6666666666666667, 0.6324894315576605),
(-0.4393939393939394, 0.6818181818181819, 0.6240065961067162),
(-0.4393939393939394, 0.696969696969697, 0.6159828671304172),
(-0.4393939393939394, 0.7121212121212122, 0.6084182446287638),
(-0.4393939393939394, 0.7272727272727273, 0.601312728601756),
(-0.4393939393939394, 0.7424242424242424, 0.5946663190493937),
(-0.4393939393939394, 0.7575757575757576, 0.5884790159716768),
(-0.4393939393939394, 0.7727272727272727, 0.5827508193686057),
(-0.4393939393939394, 0.7878787878787878, 0.5774817292401799),
(-0.4393939393939394, 0.803030303030303, 0.5726717455863999),
(-0.4393939393939394, 0.8181818181818181, 0.5683208684072654),
(-0.4393939393939394, 0.8333333333333335, 0.5644290977027762),
(-0.4393939393939394, 0.8484848484848486, 0.5609964334729327),
(-0.4393939393939394, 0.8636363636363638, 0.5580228757177348),
(-0.4393939393939394, 0.8787878787878789, 0.5555084244371822),
(-0.4393939393939394, 0.893939393939394, 0.5534530796312754),
(-0.4393939393939394, 0.9090909090909092, 0.551856841300014),
(-0.4393939393939394, 0.9242424242424243, 0.5507197094433983),
(-0.4393939393939394, 0.9393939393939394, 0.5500416840614281),
(-0.4393939393939394, 0.9545454545454546, 0.5498227651541033),
(-0.4393939393939394, 0.9696969696969697, 0.5500629527214241),
(-0.4393939393939394, 0.9848484848484849, 0.5507622467633906),
(-0.4393939393939394, 1.0, 0.5519206472800026),
(-0.42424242424242425, -0.5, 2.633393088736813),
(-0.42424242424242425, -0.48484848484848486, 2.589821811407957),
(-0.42424242424242425, -0.4696969696969697, 2.546709640553746),
(-0.42424242424242425, -0.45454545454545453, 2.5040565761741806),
(-0.42424242424242425, -0.4393939393939394, 2.4618626182692607),
(-0.42424242424242425, -0.42424242424242425, 2.4201277668389864),
(-0.42424242424242425, -0.40909090909090906, 2.3788520218833575),
(-0.42424242424242425, -0.3939393939393939, 2.338035383402374),
(-0.42424242424242425, -0.3787878787878788, 2.297677851396036),
(-0.42424242424242425, -0.36363636363636365, 2.2577794258643444),
(-0.42424242424242425, -0.3484848484848485, 2.218340106807297 7),
(-0.42424242424242425, -0.3333333333333333, 2.1793598942248966),
(-0.42424242424242425, -0.3181818181818182, 2.140838788117141),

(-0.42424242424242425, -0.30303030303030304, 2.102776788484031),
(-0.42424242424242425, -0.28787878787878785, 2.0651738953255663),
(-0.42424242424242425, -0.2727272727272727, 2.0280301086417474),
(-0.42424242424242425, -0.25757575757575757, 1.9913454284325742),
(-0.42424242424242425, -0.24242424242424243, 1.9551198546980466),
(-0.42424242424242425, -0.2272727272727273, 1.9193533874381643),
(-0.42424242424242425, -0.2121212121212121, 1.8840460266529275),
(-0.42424242424242425, -0.19696969696969696, 1.8491977723423363),
(-0.42424242424242425, -0.18181818181818182, 1.8148086245063904),
(-0.42424242424242425, -0.16666666666666663, 1.7808785831450904),
(-0.42424242424242425, -0.1515151515151515, 1.7474076482584358),
(-0.42424242424242425, -0.13636363636363635, 1.714395819846427),
(-0.42424242424242425, -0.12121212121212122, 1.6818430979090637),
(-0.42424242424242425, -0.10606060606060608, 1.6497494824463454),
(-0.42424242424242425, -0.09090909090909088, 1.618114973458273),
(-0.42424242424242425, -0.07575757575757575, 1.5869395709448464),
(-0.42424242424242425, -0.06060606060606061, 1.5562232749060652),
(-0.42424242424242425, -0.045454545454545414, 1.525966085341929),
(-0.42424242424242425, -0.030303030303030276, 1.4961680022524388),
(-0.42424242424242425, -0.015151515151515138, 1.4668290256375942),
(-0.42424242424242425, 0.0, 1.4379491554973949),
(-0.42424242424242425, 0.015151515151515138, 1.4095283918318415),
(-0.42424242424242425, 0.030303030303030276, 1.3815667346409333),
(-0.42424242424242425, 0.045454545454545414, 1.3540641839246708),
(-0.42424242424242425, 0.06060606060606066, 1.3270207396830538),
(-0.42424242424242425, 0.0757575757575758, 1.3004364019160823),
(-0.42424242424242425, 0.09090909090909094, 1.2743111706237567),
(-0.42424242424242425, 0.10606060606060608, 1.2486450458060763),
(-0.42424242424242425, 0.12121212121212122, 1.2234380274630412),
(-0.42424242424242425, 0.13636363636363635, 1.1986901155946523),
(-0.42424242424242425, 0.1515151515151515, 1.1744013102009085),
(-0.42424242424242425, 0.16666666666666674, 1.1505716112818103),
(-0.42424242424242425, 0.18181818181818188, 1.1272010188373573),
(-0.42424242424242425, 0.19696969696969702, 1.1042895328675504),
(-0.42424242424242425, 0.21212121212121215, 1.0818371533723887),
(-0.42424242424242425, 0.2272727272727273, 1.0598438803518728),
(-0.42424242424242425, 0.24242424242424243, 1.0383097138060025),
(-0.42424242424242425, 0.25757575757575757, 1.0172346537347774),
(-0.42424242424242425, 0.2727272727272727, 0.9966187001381981),
(-0.42424242424242425, 0.28787878787878785, 0.9764618530162642),
(-0.42424242424242425, 0.3030303030303031, 0.9567641123689757),
(-0.42424242424242425, 0.31818181818181823, 0.937525478196333),
(-0.42424242424242425, 0.33333333333333337, 0.9187459504983359),
(-0.42424242424242425, 0.3484848484848485, 0.9004255292749841),
(-0.42424242424242425, 0.3636363636363636, 0.882564214526278),
(-0.42424242424242425, 0.3787878787878788, 0.8651620062522174),
(-0.42424242424242425, 0.3939393939393939, 0.8482189044528025),
(-0.42424242424242425, 0.40909090909090917, 0.8317349091280327),
(-0.42424242424242425, 0.4242424242424243, 0.8157100202779088),
(-0.42424242424242425, 0.43939393939393945, 0.8001442379024304),
(-0.42424242424242425, 0.4545454545454546, 0.7850375620015975),
(-0.42424242424242425, 0.4696969696969697, 0.7703899925754101),
(-0.42424242424242425, 0.48484848484848486, 0.7562015296238683),
(-0.42424242424242425, 0.5, 0.7424721731469721),
(-0.42424242424242425, 0.5151515151515151, 0.7292019231447214),
(-0.42424242424242425, 0.5303030303030303, 0.7163907796171163),
(-0.42424242424242425, 0.5454545454545454, 0.7040387425641565),

(-0.42424242424242425, 0.5606060606060606, 0.6921458119858426),
(-0.42424242424242425, 0.5757575757575757, 0.6807119878821739),
(-0.42424242424242425, 0.5909090909090908, 0.6697372702531509),
(-0.42424242424242425, 0.6060606060606062, 0.6592216590987733),
(-0.42424242424242425, 0.6212121212121213, 0.6491651544190413),
(-0.42424242424242425, 0.6363636363636365, 0.639567756213955),
(-0.42424242424242425, 0.6515151515151516, 0.6304294644835143),
(-0.42424242424242425, 0.6666666666666667, 0.6217502792277189),
(-0.42424242424242425, 0.6818181818181819, 0.6135302004465691),
(-0.42424242424242425, 0.696969696969697, 0.605769228140065),
(-0.42424242424242425, 0.7121212121212122, 0.5984673623082062),
(-0.42424242424242425, 0.7272727272727273, 0.591624602950993),
(-0.42424242424242425, 0.7424242424242424, 0.5852409500684255),
(-0.42424242424242425, 0.7575757575757576, 0.5793164036605035),
(-0.42424242424242425, 0.7727272727272727, 0.5738509637272269),
(-0.42424242424242425, 0.7878787878787878, 0.568844630268596),
(-0.42424242424242425, 0.803030303030303, 0.5642974032846105),
(-0.42424242424242425, 0.8181818181818181, 0.5602092827752706),
(-0.42424242424242425, 0.8333333333333335, 0.5565802687405763),
(-0.42424242424242425, 0.8484848484848486, 0.5534103611805274),
(-0.42424242424242425, 0.8636363636363638, 0.5506995600951241),
(-0.42424242424242425, 0.8787878787878789, 0.5484478654843665),
(-0.42424242424242425, 0.893939393939394, 0.5466552773482544),
(-0.42424242424242425, 0.9090909090909092, 0.5453217956867876),
(-0.42424242424242425, 0.9242424242424243, 0.5444474204999667),
(-0.42424242424242425, 0.9393939393939394, 0.5440321517877911),
(-0.42424242424242425, 0.9545454545454546, 0.544075989550261),
(-0.42424242424242425, 0.9696969696969697, 0.5445789337873765),
(-0.42424242424242425, 0.9848484848484849, 0.5455409844991377),
(-0.42424242424242425, 1.0, 0.5469621416855441),
(-0.40909090909090906, -0.5, 2.6028807793073248),
(-0.40909090909090906, -0.48484848484848486, 2.5595722586482634),
(-0.40909090909090906, -0.4696969696969697, 2.5167228444638474),
(-0.40909090909090906, -0.45454545454545453, 2.4743325367540763),
(-0.40909090909090906, -0.4393939393939394, 2.432401335518951),
(-0.40909090909090906, -0.42424242424242425, 2.3909292407584712),
(-0.40909090909090906, -0.40909090909090906, 2.3499162524726374),
(-0.40909090909090906, -0.3939393939393939, 2.309362370661449),
(-0.40909090909090906, -0.3787878787878788, 2.269267595324906),
(-0.40909090909090906, -0.3636363636363636, 2.229631926463008),
(-0.40909090909090906, -0.3484848484848485, 2.1904553640757563),
(-0.40909090909090906, -0.3333333333333333, 2.15173790816315),
(-0.40909090909090906, -0.3181818181818182, 2.113479558725189),
(-0.40909090909090906, -0.30303030303030304, 2.0756803157618737),
(-0.40909090909090906, -0.28787878787878785, 2.038340179273204),
(-0.40909090909090906, -0.2727272727272727, 2.00145914925918),
(-0.40909090909090906, -0.25757575757575757, 1.9650372257198014),
(-0.40909090909090906, -0.24242424242424243, 1.9290744086550682),
(-0.40909090909090906, -0.2272727272727273, 1.8935706980649807),
(-0.40909090909090906, -0.2121212121212121, 1.8585260939495385),
(-0.40909090909090906, -0.1969696969696696, 1.8239405963087423),
(-0.40909090909090906, -0.18181818181818182, 1.7898142051425912),
(-0.40909090909090906, -0.16666666666666663, 1.7561469204510858),
(-0.40909090909090906, -0.1515151515151515, 1.722938742234226),
(-0.40909090909090906, -0.13636363636363635, 1.6901896704920116),
(-0.40909090909090906, -0.12121212121212122, 1.657899705224443),
(-0.40909090909090906, -0.10606060606060608, 1.6260688464315196),

```
(-0.40909090909090906, -0.09090909090909088, 1.5946970941132417),
(-0.40909090909090906, -0.07575757575757575, 1.5637844482696097),
(-0.40909090909090906, -0.06060606060606061, 1.533330908900623),
(-0.40909090909090906, -0.045454545454545414, 1.5033364760062817),
(-0.40909090909090906, -0.030303030303030276, 1.4738011495865866),
(-0.40909090909090906, -0.015151515151515138, 1.4447249296415365),
(-0.40909090909090906, 0.0, 1.4161078161711318),
(-0.40909090909090906, 0.015151515151515138, 1.387949809175373),
(-0.40909090909090906, 0.030303030303030276, 1.3602509086542596),
(-0.40909090909090906, 0.045454545454545414, 1.333011114607792),
(-0.40909090909090906, 0.06060606060606066, 1.3062304270359693),
(-0.40909090909090906, 0.0757575757575758, 1.2799088459387928),
(-0.40909090909090906, 0.09090909090909094, 1.2540463713162615),
(-0.40909090909090906, 0.10606060606060608, 1.2286430031683762),
(-0.40909090909090906, 0.12121212121212122, 1.2036987414951361),
(-0.40909090909090906, 0.13636363636363635, 1.1792135862965414),
(-0.40909090909090906, 0.1515151515151515, 1.1551875375725924),
(-0.40909090909090906, 0.16666666666666674, 1.1316205953232887),
(-0.40909090909090906, 0.18181818181818188, 1.1085127595486308),
(-0.40909090909090906, 0.19696969696969702, 1.0858640302486182),
(-0.40909090909090906, 0.21212121212121215, 1.0636744074232518),
(-0.40909090909090906, 0.2272727272727273, 1.0419438910725303),
(-0.40909090909090906, 0.24242424242424243, 1.0206724811964545),
(-0.40909090909090906, 0.25757575757575757, 0.9998601777950241),
(-0.40909090909090906, 0.2727272727272727, 0.9795069808682397),
(-0.40909090909090906, 0.28787878787878785, 0.9596128904161005),
(-0.40909090909090906, 0.3030303030303031, 0.9401779064386069),
(-0.40909090909090906, 0.31818181818181823, 0.9212020289357588),
(-0.40909090909090906, 0.33333333333333337, 0.9026852579075563),
(-0.40909090909090906, 0.3484848484848485, 0.8846275933539992),
(-0.40909090909090906, 0.36363636363636365, 0.8670290352750878),
(-0.40909090909090906, 0.3787878787878788, 0.8498895836708219),
(-0.40909090909090906, 0.3939393939393939, 0.8332092385412015),
(-0.40909090909090906, 0.40909090909090917, 0.8169879998862266),
(-0.40909090909090906, 0.4242424242424243, 0.8012258677058974),
(-0.40909090909090906, 0.43939393939393945, 0.7859228420002138),
(-0.40909090909090906, 0.4545454545454546, 0.7710789227691756),
(-0.40909090909090906, 0.4696969696969697, 0.7566941100127829),
(-0.40909090909090906, 0.48484848484848486, 0.7427684037310357),
(-0.40909090909090906, 0.5, 0.7293018039239342),
(-0.40909090909090906, 0.5151515151515151, 0.7162943105914782),
(-0.40909090909090906, 0.5303030303030303, 0.7037459237336677),
(-0.40909090909090906, 0.5454545454545454, 0.6916566433505027),
(-0.40909090909090906, 0.5606060606060606, 0.6800264694419833),
(-0.40909090909090906, 0.5757575757575757, 0.6688554020081097),
(-0.40909090909090906, 0.5909090909090908, 0.6581434410488813),
(-0.40909090909090906, 0.6060606060606062, 0.6478905865642983),
(-0.40909090909090906, 0.6212121212121213, 0.6380968385543612),
(-0.40909090909090906, 0.6363636363636365, 0.6287621970190694),
(-0.40909090909090906, 0.6515151515151516, 0.6198866619584233),
(-0.40909090909090906, 0.6666666666666667, 0.6114702333724228),
(-0.40909090909090906, 0.6818181818181819, 0.6035129112610678),
(-0.40909090909090906, 0.696969696969697, 0.5960146956243583),
(-0.40909090909090906, 0.7121212121212122, 0.5889755864622943),
(-0.40909090909090906, 0.7272727272727273, 0.5823955837748758),
(-0.40909090909090906, 0.7424242424242424, 0.5762746875621029),
(-0.40909090909090906, 0.7575757575757576, 0.5706128978239755),
```

(-0.40909090909090906, 0.7727272727272727, 0.5654102145604938),
(-0.40909090909090906, 0.7878787878787878, 0.5606666377716576),
(-0.40909090909090906, 0.803030303030303, 0.5563821674574668),
(-0.40909090909090906, 0.8181818181818181, 0.5525568036179216),
(-0.40909090909090906, 0.8333333333333335, 0.5491905462530219),
(-0.40909090909090906, 0.8484848484848486, 0.5462833953627677),
(-0.40909090909090906, 0.8636363636363638, 0.5438353509471592),
(-0.40909090909090906, 0.8787878787878789, 0.5418464130061962),
(-0.40909090909090906, 0.893939393939394, 0.5403165815398787),
(-0.40909090909090906, 0.9090909090909092, 0.5392458565482068),
(-0.40909090909090906, 0.9242424242424243, 0.5386342380311805),
(-0.40909090909090906, 0.9393939393939394, 0.5384817259887996),
(-0.40909090909090906, 0.9545454545454546, 0.5387883204210643),
(-0.40909090909090906, 0.9696969696969697, 0.5395540213279745),
(-0.40909090909090906, 0.9848484848484849, 0.5407788287095303),
(-0.40909090909090906, 1.0, 0.5424627425657317),
(-0.3939393939393939, -0.5, 2.5728275763524824),
(-0.3939393939393939, -0.48484848484848486, 2.529781812363215),
(-0.3939393939393939, -0.4696969696969697, 2.487195154848594),
(-0.3939393939393939, -0.45454545454545453, 2.445067603808618),
(-0.3939393939393939, -0.4393939393939394, 2.4033991592432873),
(-0.3939393939393939, -0.42424242424242425, 2.3621898211526027),
(-0.3939393939393939, -0.40909090909090906, 2.321439589536563),
(-0.3939393939393939, -0.3939393939393939, 2.2811484643951694),
(-0.3939393939393939, -0.3787878787878788, 2.241316445728421),
(-0.3939393939393939, -0.36363636363636365, 2.201943533536318),
(-0.3939393939393939, -0.3484848484848485, 2.163029727818861),
(-0.3939393939393939, -0.3333333333333333, 2.124575028576049),
(-0.3939393939393939, -0.3181818181818182, 2.086579435807883),
(-0.3939393939393939, -0.30303030303030304, 2.0490429495143627),
(-0.3939393939393939, -0.28787878787878785, 2.0119655696954872),
(-0.3939393939393939, -0.2727272727272727, 1.975347296351258),
(-0.3939393939393939, -0.25757575757575757, 1.9391881294816742),
(-0.3939393939393939, -0.24242424242424243, 1.9034880690867357),
(-0.3939393939393939, -0.2272727272727273, 1.8682471151664428),
(-0.3939393939393939, -0.2121212121212121, 1.8334652677207952),
(-0.3939393939393939, -0.19696969696969696, 1.7991425267497936),
(-0.3939393939393939, -0.18181818181818182, 1.7652788922534375),
(-0.3939393939393939, -0.16666666666666663, 1.7318743642317262),
(-0.3939393939393939, -0.1515151515151515, 1.6989289426846614),
(-0.3939393939393939, -0.13636363636363635, 1.6664426276122417),
(-0.3939393939393939, -0.12121212121212122, 1.6344154190144675),
(-0.3939393939393939, -0.10606060606060608, 1.6028473168913393),
(-0.3939393939393939, -0.09090909090909088, 1.5717383212428562),
(-0.3939393939393939, -0.07575757575757575, 1.5410884320690186),
(-0.3939393939393939, -0.06060606060606061, 1.5108976493698272),
(-0.3939393939393939, -0.045454545454545414, 1.4811659731452806),
(-0.3939393939393939, -0.030303030303030276, 1.4518934033953796),
(-0.3939393939393939, -0.015151515151515138, 1.4230799401201242),
(-0.3939393939393939, 0.0, 1.3947255833195147),
(-0.3939393939393939, 0.015151515151515138, 1.3668303329935503),
(-0.3939393939393939, 0.030303030303030276, 1.3393941891422319),
(-0.3939393939393939, 0.045454545454545414, 1.3124171517655587),
(-0.3939393939393939, 0.06060606060606066, 1.285899220863531),
(-0.3939393939393939, 0.0757575757575758, 1.2598403964361489),
(-0.3939393939393939, 0.09090909090909094, 1.2342406784834126),
(-0.3939393939393939, 0.10606060606060608, 1.2091000670053218),

(-0.3939393939393939, 0.12121212121212122, 1.1844185620018761),
(-0.3939393939393939, 0.13636363636363635, 1.1601961634730764),
(-0.3939393939393939, 0.1515151515151515, 1.1364328714189218),
(-0.3939393939393939, 0.16666666666666674, 1.1131286858394132),
(-0.3939393939393939, 0.18181818181818188, 1.0902836067345498),
(-0.3939393939393939, 0.19696969696969702, 1.0678976341043322),
(-0.3939393939393939, 0.21212121212121215, 1.04597076794876),
(-0.3939393939393939, 0.2272727272727273, 1.0245030082678335),
(-0.3939393939393939, 0.24242424242424243, 1.0034943550615525),
(-0.3939393939393939, 0.25757575757575757, 0.9829448083299169),
(-0.3939393939393939, 0.2727272727272727, 0.9628543680729269),
(-0.3939393939393939, 0.28787878787878785, 0.9432230342905825),
(-0.3939393939393939, 0.3030303030303031, 0.9240508069828834),
(-0.3939393939393939, 0.31818181818181823, 0.90533768614983),
(-0.3939393939393939, 0.33333333333333337, 0.8870836717914222),
(-0.3939393939393939, 0.3484848484848485, 0.86928876390766),
(-0.3939393939393939, 0.36363636363636365, 0.8519529624985432),
(-0.3939393939393939, 0.3787878787878788, 0.835076267564072),
(-0.3939393939393939, 0.3939393939393939, 0.8186586791042464),
(-0.3939393939393939, 0.40909090909090917, 0.8027001971190663),
(-0.3939393939393939, 0.4242424242424243, 0.7872008216085316),
(-0.3939393939393939, 0.43939393939393945, 0.7721605525726426),
(-0.3939393939393939, 0.4545454545454546, 0.7575793900113992),
(-0.3939393939393939, 0.4696969696969697, 0.7434573339248012),
(-0.3939393939393939, 0.48484848484848486, 0.7297943843128488),
(-0.3939393939393939, 0.5, 0.716590541175542),
(-0.3939393939393939, 0.5151515151515151, 0.7038458045128807),
(-0.3939393939393939, 0.5303030303030303, 0.6915601743248649),
(-0.3939393939393939, 0.5454545454545454, 0.6797336506114947),
(-0.3939393939393939, 0.5606060606060606, 0.66836623337277),
(-0.3939393939393939, 0.5757575757575757, 0.657457922608691),
(-0.3939393939393939, 0.5909090909090908, 0.6470087183192571),
(-0.3939393939393939, 0.6060606060606062, 0.6370186205044691),
(-0.3939393939393939, 0.6212121212121213, 0.6274876291643267),
(-0.3939393939393939, 0.6363636363636365, 0.6184157442988296),
(-0.3939393939393939, 0.6515151515151516, 0.6098029659079781),
(-0.3939393939393939, 0.6666666666666667, 0.6016492939917721),
(-0.3939393939393939, 0.6818181818181819, 0.5939547285502119),
(-0.3939393939393939, 0.696969696969697, 0.5867192695832971),
(-0.3939393939393939, 0.7121212121212122, 0.5799429170910277),
(-0.3939393939393939, 0.7272727272727273, 0.5736256710734041),
(-0.3939393939393939, 0.7424242424242424, 0.5677675315304259),
(-0.3939393939393939, 0.7575757575757576, 0.5623684984620932),
(-0.3939393939393939, 0.7727272727272727, 0.557428571868406),
(-0.3939393939393939, 0.7878787878787878, 0.5529477517493646),
(-0.3939393939393939, 0.803030303030303, 0.5489260381049685),
(-0.3939393939393939, 0.8181818181818181, 0.5453634309352181),
(-0.3939393939393939, 0.8333333333333335, 0.542259930240113),
(-0.3939393939393939, 0.8484848484848486, 0.5396155360196536),
(-0.3939393939393939, 0.8636363636363638, 0.5374302482738398),
(-0.3939393939393939, 0.8787878787878789, 0.5357040670026716),
(-0.3939393939393939, 0.893939393939394, 0.5344369922061488),
(-0.3939393939393939, 0.9090909090909092, 0.5336290238842716),
(-0.3939393939393939, 0.9242424242424243, 0.5332801620370399),
(-0.3939393939393939, 0.9393939393939394, 0.5333904066644537),
(-0.3939393939393939, 0.9545454545454546, 0.533959757766513),
(-0.3939393939393939, 0.9696969696969697, 0.534988215343218),

(-0.3939393939393939, 0.9848484848484849, 0.5364757793945686),
(-0.3939393939393939, 1.0, 0.5384224499205645),
(-0.3787878787878788, -0.5, 2.5432334798722853),
(-0.3787878787878788, -0.48484848484848486, 2.5004504725528127),
(-0.3787878787878788, -0.4696969696969697, 2.458126571707986),
(-0.3787878787878788, -0.45454545454545453, 2.416261777337805),
(-0.3787878787878788, -0.4393939393939394, 2.374856089442269),
(-0.3787878787878788, -0.42424242424242425, 2.3339095080213785),
(-0.3787878787878788, -0.40909090909090906, 2.2934220330751343),
(-0.3787878787878788, -0.3939393939393939, 2.2533936646035353),
(-0.3787878787878788, -0.3787878787878788, 2.213824402606581),
(-0.3787878787878788, -0.36363636363636365, 2.1747142470842733),
(-0.3787878787878788, -0.3484848484848485, 2.136063198036611),
(-0.3787878787878788, -0.3333333333333333, 2.097871255463594),
(-0.3787878787878788, -0.3181818181818182, 2.060138419365223),
(-0.3787878787878788, -0.30303030303030304, 2.0228646897414966),
(-0.3787878787878788, -0.28787878787878785, 1.9860500665924163),
(-0.3787878787878788, -0.2727272727272727, 1.9496945499179812),
(-0.3787878787878788, -0.25757575757575757, 1.913798139718192),
(-0.3787878787878788, -0.24242424242424243, 1.8783608359930486),
(-0.3787878787878788, -0.2272727272727273, 1.8433826387425503),
(-0.3787878787878788, -0.2121212121212121, 1.8088635479666975),
(-0.3787878787878788, -0.19696969696969696, 1.7748035636654909),
(-0.3787878787878788, -0.18181818181818182, 1.7412026858389291),
(-0.3787878787878788, -0.16666666666666663, 1.708060914487013),
(-0.3787878787878788, -0.1515151515151515, 1.6753782496097427),
(-0.3787878787878788, -0.13636363636363635, 1.6431546912071175),
(-0.3787878787878788, -0.12121212121212122, 1.6113902392791382),
(-0.3787878787878788, -0.10606060606060608, 1.5800848938258043),
(-0.3787878787878788, -0.09090909090909088, 1.5492386548471162),
(-0.3787878787878788, -0.07575757575757575, 1.5188515223430734),
(-0.3787878787878788, -0.06060606060606061, 1.4889234963136764),
(-0.3787878787878788, -0.045454545454545414, 1.4594545767589244),
(-0.3787878787878788, -0.030303030303030276, 1.4304447636788182),
(-0.3787878787878788, -0.015151515151515138, 1.4018940570733576),
(-0.3787878787878788, 0.0, 1.3738024569425429),
(-0.3787878787878788, 0.015151515151515138, 1.3461699632863733),
(-0.3787878787878788, 0.030303030303030276, 1.3189965761048494),
(-0.3787878787878788, 0.045454545454545414, 1.2922822953979711),
(-0.3787878787878788, 0.06060606060606066, 1.2660271211657381),
(-0.3787878787878788, 0.0757575757575758, 1.2402310534081507),
(-0.3787878787878788, 0.09090909090909094, 1.214894092125209),
(-0.3787878787878788, 0.10606060606060608, 1.1900162373169128),
(-0.3787878787878788, 0.12121212121212122, 1.165597488983262),
(-0.3787878787878788, 0.13636363636363635, 1.1416378471242568),
(-0.3787878787878788, 0.1515151515151515, 1.118137311739897),
(-0.3787878787878788, 0.16666666666666674, 1.0950958828301827),
(-0.3787878787878788, 0.18181818181818188, 1.0725135603951144),
(-0.3787878787878788, 0.19696969696969702, 1.0503903444346914),
(-0.3787878787878788, 0.21212121212121215, 1.0287262349489141),
(-0.3787878787878788, 0.2272727272727273, 1.007521231937782),
(-0.3787878787878788, 0.24242424242424243, 0.9867753354012957),
(-0.3787878787878788, 0.25757575757575757, 0.9664885453394549),
(-0.3787878787878788, 0.2727272727272727, 0.9466608617522597),
(-0.3787878787878788, 0.28787878787878785, 0.9272922846397099),
(-0.3787878787878788, 0.3030303030303031, 0.9083828140018055),
(-0.3787878787878788, 0.31818181818181823, 0.889932449838547),

(-0.3787878787878788, 0.33333333333333337, 0.8719411921499339),
(-0.3787878787878788, 0.3484848484848485, 0.8544090409359663),
(-0.3787878787878788, 0.36363636363636365, 0.8373359961966442),
(-0.3787878787878788, 0.3787878787878788, 0.8207220579319677),
(-0.3787878787878788, 0.3939393939393939, 0.8045672261419369),
(-0.3787878787878788, 0.40909090909090917, 0.7888715008265513),
(-0.3787878787878788, 0.4242424242424243, 0.7736348819858114),
(-0.3787878787878788, 0.43939393939393945, 0.7588573696197172),
(-0.3787878787878788, 0.4545454545454546, 0.7445389637282684),
(-0.3787878787878788, 0.4696969696969697, 0.7306796643114651),
(-0.3787878787878788, 0.48484848484848486, 0.7172794713693076),
(-0.3787878787878788, 0.5, 0.7043383849017953),
(-0.3787878787878788, 0.5151515151515151, 0.6918564049089287),
(-0.3787878787878788, 0.5303030303030303, 0.6798335313907077),
(-0.3787878787878788, 0.5454545454545454, 0.6682697643471323),
(-0.3787878787878788, 0.5606060606060606, 0.6571651037782021),
(-0.3787878787878788, 0.5757575757575757, 0.6465195496839177),
(-0.3787878787878788, 0.5909090909090908, 0.6363331020642787),
(-0.3787878787878788, 0.6060606060606062, 0.6266057609192852),
(-0.3787878787878788, 0.6212121212121213, 0.6173375262489375),
(-0.3787878787878788, 0.6363636363636365, 0.6085283980532352),
(-0.3787878787878788, 0.6515151515151516, 0.6001783763321784),
(-0.3787878787878788, 0.6666666666666667, 0.5922874610857672),
(-0.3787878787878788, 0.6818181818181819, 0.5848556523140017),
(-0.3787878787878788, 0.696969696969697, 0.5778829500168816),
(-0.3787878787878788, 0.7121212121212122, 0.5713693541944069),
(-0.3787878787878788, 0.7272727272727273, 0.565314864846578),
(-0.3787878787878788, 0.7424242424242424, 0.5597194819733945),
(-0.3787878787878788, 0.7575757575757576, 0.5545832055748566),
(-0.3787878787878788, 0.7727272727272727, 0.5499060356509641),
(-0.3787878787878788, 0.7878787878787878, 0.5456879722017172),
(-0.3787878787878788, 0.803030303030303, 0.5419290152271159),
(-0.3787878787878788, 0.8181818181818181, 0.53862916472716),
(-0.3787878787878788, 0.8333333333333335, 0.53578842070185),
(-0.3787878787878788, 0.8484848484848486, 0.5334067831511852),
(-0.3787878787878788, 0.8636363636363638, 0.5314842520751661),
(-0.3787878787878788, 0.8787878787878789, 0.5300208274737924),
(-0.3787878787878788, 0.893939393939394, 0.5290165093470643),
(-0.3787878787878788, 0.9090909090909092, 0.5284712976949819),
(-0.3787878787878788, 0.9242424242424243, 0.5283851925175449),
(-0.3787878787878788, 0.9393939393939394, 0.5287581938147534),
(-0.3787878787878788, 0.9545454545454546, 0.5295903015866076),
(-0.3787878787878788, 0.9696969696969697, 0.5308815158331072),
(-0.3787878787878788, 0.9848484848484849, 0.5326318365542523),
(-0.3787878787878788, 1.0, 0.534841263750043),
(-0.36363636363636365, -0.5, 2.5140984898667336),
(-0.36363636363636365, -0.48484848484848486, 2.4715782392170564),
(-0.36363636363636365, -0.4696969696969697, 2.429517095042024),
(-0.36363636363636365, -0.45454545454545453, 2.3879150573416372),
(-0.36363636363636365, -0.4393939393939394, 2.346772126115896 5),
(-0.36363636363636365, -0.42424242424242425, 2.3060883013648006),
(-0.36363636363636365, -0.40909090909090906, 2.265863583088351),
(-0.36363636363636365, -0.3939393939393939, 2.226097971286546),
(-0.36363636363636365, -0.3787878787878788, 2.1867914659593874),
(-0.36363636363636365, -0.3636363636363636 5, 2.1479440671068746),
(-0.36363636363636365, -0.3484848484848485, 2.109555774729006),
(-0.36363636363636365, -0.3333333333333333, 2.071626588825784),

(-0.36363636363636365, -0.3181818181818182, 2.0341565093972074),
(-0.36363636363636365, -0.30303030303030304, 1.9971455364432762),
(-0.36363636363636365, -0.28787878787878785, 1.9605936699639905),
(-0.36363636363636365, -0.2727272727272727, 1.9245009099593504),
(-0.36363636363636365, -0.25757575757575757, 1.8888672564293558),
(-0.36363636363636365, -0.24242424242424243, 1.853692709374007),
(-0.36363636363636365, -0.2272727272727273, 1.8189772687933035),
(-0.36363636363636365, -0.2121212121212121, 1.7847209346872457),
(-0.36363636363636365, -0.1969696969696969, 1.7509237070558332),
(-0.36363636363636365, -0.18181818181818182, 1.7175855858990665),
(-0.36363636363636365, -0.16666666666666663, 1.6847065712169447),
(-0.36363636363636365, -0.1515151515151515, 1.6522866630094692),
(-0.36363636363636365, -0.13636363636363635, 1.6203258612766391),
(-0.36363636363636365, -0.12121212121212122, 1.5888241660184543),
(-0.36363636363636365, -0.10606060606060608, 1.5577815772349153),
(-0.36363636363636365, -0.09090909090909088, 1.5271980949260215),
(-0.36363636363636365, -0.07575757575757575, 1.4970737190917738),
(-0.36363636363636365, -0.06060606060606061, 1.4674084497321709),
(-0.36363636363636365, -0.045454545454545414, 1.4382022868472142),
(-0.36363636363636365, -0.030303030303030276, 1.4094552304369024),
(-0.36363636363636365, -0.015151515151515138, 1.3811672805012367),
(-0.36363636363636365, 0.0, 1.3533384370402166),
(-0.36363636363636365, 0.015151515151515138, 1.3259687000538418),
(-0.36363636363636365, 0.030303030303030276, 1.2990580695421126),
(-0.36363636363636365, 0.045454545454545414, 1.2726065455050286),
(-0.36363636363636365, 0.06060606060606066, 1.2466141279425904),
(-0.36363636363636365, 0.0757575757575758, 1.2210808168547977),
(-0.36363636363636365, 0.09090909090909094, 1.1960066122416508),
(-0.36363636363636365, 0.10606060606060608, 1.1713915141031492),
(-0.36363636363636365, 0.12121212121212122, 1.1472355224392934),
(-0.36363636363636365, 0.13636363636363635, 1.1235386372500826),
(-0.36363636363636365, 0.1515151515151515, 1.1003008585355176),
(-0.36363636363636365, 0.16666666666666674, 1.0775221862955984),
(-0.36363636363636365, 0.18181818181818188, 1.0552026205303244),
(-0.36363636363636365, 0.19696969696969702, 1.0333421612396965),
(-0.36363636363636365, 0.21212121212121215, 1.0119408084237136),
(-0.36363636363636365, 0.2272727272727273, 0.9909985620823764),
(-0.36363636363636365, 0.24242424242424243, 0.9705154222156848),
(-0.36363636363636365, 0.25757575757575757, 0.9504913888236385),
(-0.36363636363636365, 0.2727272727272727, 0.9309264619062381),
(-0.36363636363636365, 0.28787878787878785, 0.911820641463483),
(-0.36363636363636365, 0.3030303030303031, 0.8931739274953733),
(-0.36363636363636365, 0.31818181818181823, 0.8749863200019095),
(-0.36363636363636365, 0.33333333333333337, 0.8572578189830911),
(-0.36363636363636365, 0.3484848484848485, 0.8399884244389182),
(-0.36363636363636365, 0.36363636363636365, 0.8231781363693909),
(-0.36363636363636365, 0.3787878787878788, 0.806826954774509),
(-0.36363636363636365, 0.3939393939393939, 0.7909348796542728),
(-0.36363636363636365, 0.40909090909090917, 0.7755019110086819),
(-0.36363636363636365, 0.4242424242424243, 0.760528048837737),
(-0.36363636363636365, 0.43939393939393945, 0.7460132931414373),
(-0.36363636363636365, 0.4545454545454546, 0.7319576439197834),
(-0.36363636363636365, 0.4696969696969697, 0.7183611011727746),
(-0.36363636363636365, 0.48484848484848486, 0.7052236649004117),
(-0.36363636363636365, 0.5, 0.6925453351026942),
(-0.36363636363636365, 0.5151515151515151, 0.6803261117796222),
(-0.36363636363636365, 0.5303030303030303, 0.6685659949311958),

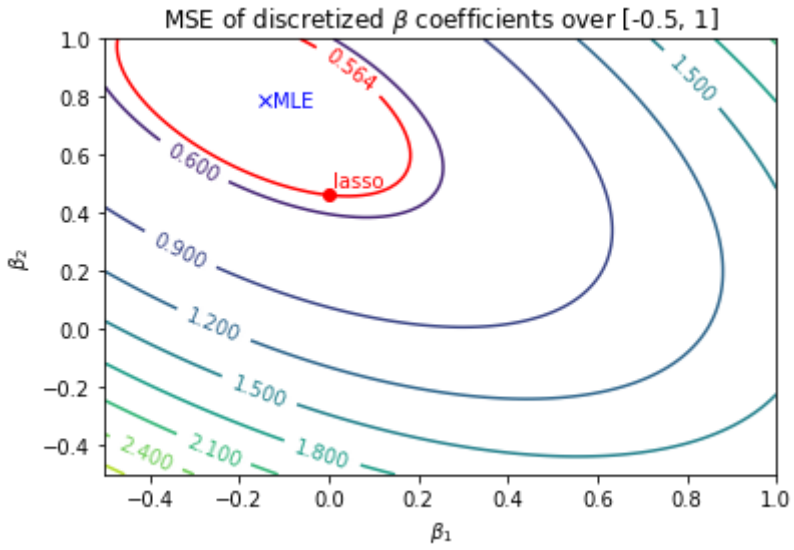
```
(-0.36363636363636365, 0.5454545454545454, 0.6572649845574151),
(-0.36363636363636365, 0.5606060606060606, 0.6464230806582797),
(-0.36363636363636365, 0.5757575757575757, 0.6360402832337902),
(-0.36363636363636365, 0.5909090909090908, 0.6261165922839459),
(-0.36363636363636365, 0.6060606060606062, 0.6166520078087472),
(-0.36363636363636365, 0.6212121212121213, 0.6076465298081941),
(-0.36363636363636365, 0.6363636363636365, 0.5991001582822866),
(-0.36363636363636365, 0.6515151515151516, 0.5910128932310243),
(-0.36363636363636365, 0.6666666666666667, 0.5833847346544079),
(-0.36363636363636365, 0.6818181818181819, 0.576215682552437),
(-0.36363636363636365, 0.696969696969697, 0.5695057369251116),
(-0.36363636363636365, 0.7121212121212122, 0.5632548977724317),
(-0.36363636363636365, 0.7272727272727273, 0.5574631650943974),
(-0.36363636363636365, 0.7424242424242424, 0.5521305388910086),
(-0.36363636363636365, 0.7575757575757576, 0.5472570191622653),
(-0.36363636363636365, 0.7727272727272727, 0.5428426059081677),
(-0.36363636363636365, 0.7878787878787878, 0.5388872991287155),
(-0.36363636363636365, 0.803030303030303, 0.5353910988239088),
(-0.36363636363636365, 0.8181818181818181, 0.5323540049937477),
(-0.36363636363636365, 0.8333333333333335, 0.5297760176382322),
(-0.36363636363636365, 0.8484848484848486, 0.5276571367573621),
(-0.36363636363636365, 0.8636363636363638, 0.5259973623511378),
(-0.36363636363636365, 0.8787878787878789, 0.5247966944195589),
(-0.36363636363636365, 0.893939393939394, 0.5240551329626255),
(-0.36363636363636365, 0.9090909090909092, 0.5237726779803377),
(-0.36363636363636365, 0.9242424242424243, 0.5239493294726955),
(-0.36363636363636365, 0.9393939393939394, 0.5245850874396988),
(-0.36363636363636365, 0.9545454545454546, 0.5256799518813474),
(-0.36363636363636365, 0.9696969696969697, 0.5272339227976418),
(-0.36363636363636365, 0.9848484848484849, 0.5292470001885817),
(-0.36363636363636365, 1.0, 0.5317191840541672),
...]
```

Question 2.1c

In [29]:
```
# Generate coordinates
X, Y, = np.meshgrid(beta1, beta2)

# MSE
Z = find_mse(X, Y, print_betas=False)

# Lasso MSE
lasso_mse = find_mse(b1_lasso[20], b2_lasso[20], print_betas=False)
```

In [30]:
```
# Create the contour plot
plt.figure()

cs = plt.contour(X, Y, Z)
cs_lasso = plt.contour(X, Y, Z, levels=[lasso_mse],
                       colors=("r",), linestyles=("-",))

plt.clabel(cs)
plt.clabel(cs_lasso)

# MLE and lasso coefficients
plt.plot(fit.coef_[0][0], fit.coef_[0][1], "bx")
plt.annotate(s="MLE", xy=(fit.coef_[0][0], fit.coef_[0][1]), xytext=(fit
.coef_[0][0]+0.02, fit.coef_[0][1]-0.02), color="b")
plt.plot(b1_lasso[20], b2_lasso[20], "ro")
plt.annotate(s="lasso", xy=(b1_lasso[20], b2_lasso[20]), xytext=(b1_lass
o[20]+0.01, b2_lasso[20]+0.03), color="r")
plt.title(r"MSE of discretized $\beta$ coefficients over [-0.5, 1]")
plt.xlabel(r"$\beta_{1}$")
plt.ylabel(r"$\beta_{2}$")
```

Out[30]:
```
Text(0, 0.5, '$\\beta_{2}$')
```

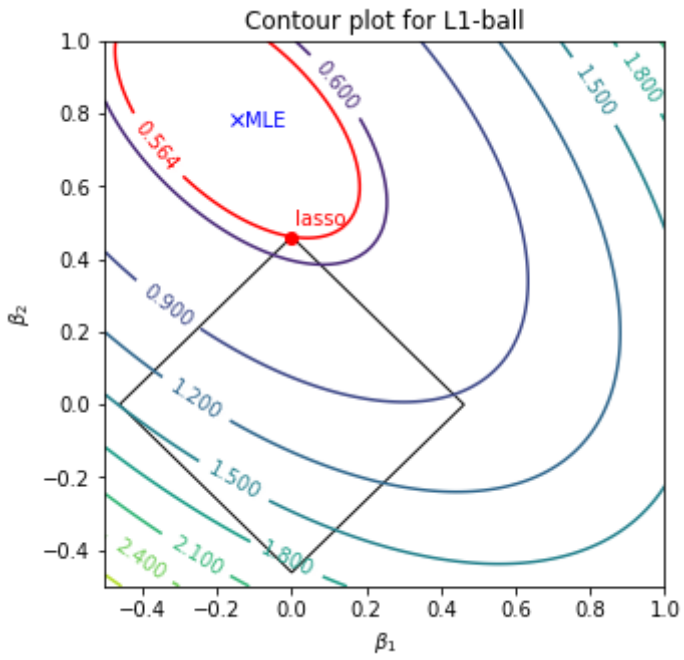


Question 2.1d

In [31]:
```
# Generate a new coefficient grid
beta1 = np.linspace(-1, 1, num=100)
beta2 = np.linspace(-1, 1, num=100)

# Every combination of beta1 and beta2
betas = list(product(beta1, beta2))
```

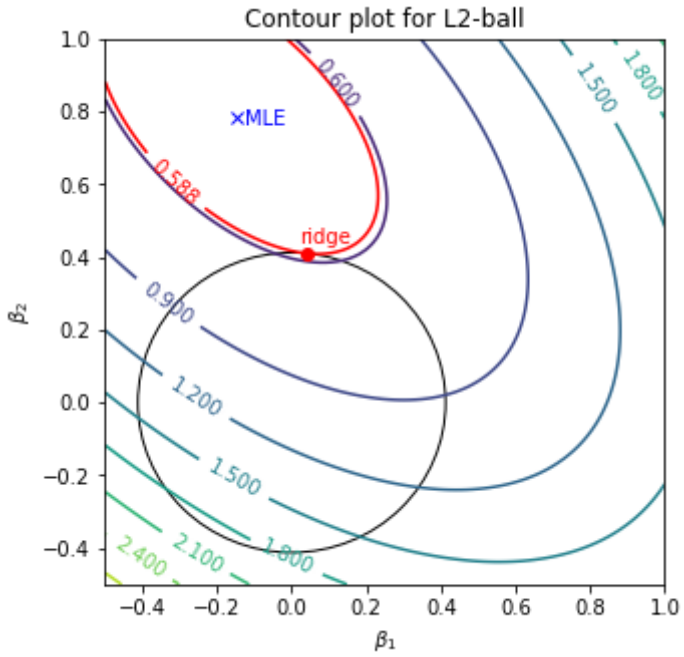
In [32]:
```
# Create the contour plot for lasso
fig, ax = plt.subplots(figsize=(5, 5))

cs = ax.contour(X, Y, Z)
cs_lasso = ax.contour(X, Y, Z, levels=[lasso_mse], colors=("r",), linest
yles=("-",))
#cs_beta = ax.contour(X, Y, 1/2*(abs(X) + abs(Y)), levels=[l1_norm])

ax.clabel(cs)
ax.clabel(cs_lasso)

# MLE and lasso coefficients
ax.plot(fit.coef_[0][0], fit.coef_[0][1], "bx")
ax.annotate(s="MLE", xy=(fit.coef_[0][0], fit.coef_[0][1]),
            xytext=(fit.coef_[0][0]+0.02, fit.coef_[0][1]-0.02), color=
"b")
ax.plot(b1_lasso[20], b2_lasso[20], "ro")
ax.annotate(s="lasso", xy=(b1_lasso[20], b2_lasso[20]),
            xytext=(b1_lasso[20]+0.01, b2_lasso[20]+0.03), color="r")
ax.set_title("Contour plot for L1-ball")
ax.set_xlabel(r"$\beta_{1}$")
ax.set_ylabel(r"$\beta_{2}$")

# Draw the square
s = 2*b2_lasso[20]/np.sqrt(2)
rect = patches.Rectangle((b1_lasso[20], -b2_lasso[20]), s, s, angle=45,
linewidth=1, edgecolor="k", facecolor="none")
ax.add_patch(rect)
```

Out[32]: <matplotlib.patches.Rectangle at 0x1345b2250>



In [33]:
```
# Ridge MSE
ridge_mse = find_mse(b1_ridge[60], b2_ridge[60], print_betas=False)

# Create the contour plot for ridge
fig, ax = plt.subplots(figsize=(5, 5))

cs = ax.contour(X, Y, Z)
cs_ridge = ax.contour(X, Y, Z, levels=[ridge_mse], colors=("r",), linest
yles=("-",))

ax.clabel(cs)
ax.clabel(cs_ridge)

# MLE and ridge coefficients
ax.plot(fit.coef_[0][0], fit.coef_[0][1], "bx")
ax.annotate(s="MLE", xy=(fit.coef_[0][0], fit.coef_[0][1]),
            xytext=(fit.coef_[0][0]+0.02, fit.coef_[0][1]-0.02), color=
"b")
ax.plot(b1_ridge[60], b2_ridge[60], "ro")
ax.annotate(s="ridge", xy=(b1_ridge[60], b2_ridge[60]),
            xytext=(b1_ridge[60]-0.02, b2_ridge[60]+0.03), color="r")
ax.set_title("Contour plot for L2-ball")
ax.set_xlabel(r"$\beta_{1}$")
ax.set_ylabel(r"$\beta_{2}$")

# Draw the circle
r = np.sqrt(b1_ridge[60]**2 + b2_ridge[60]**2)
circle = patches.Circle((0, 0), r, linewidth=1, edgecolor="k", facecolor
="none")
ax.add_patch(circle)
```

Out[33]: <matplotlib.patches.Circle at 0x12ef96850>



Question 2.1e

From the contour plots above, we can see that the $\hat{\beta}_{\text{MLE}}$ coefficients sit in the valley of the decreasing contours, indicating that these are the parameter estimates that minimize the MSE.

The $\hat{\beta}_{\text{lasso}}$ and $\hat{\beta}_{\text{ridge}}$ coefficients sit on the intersection between a contour and the L1-/L2-ball, respectively. Since these coefficients do not sit in the valley of the contours, they are not the parameter estimates that minimize the MSE. This is expected because of the penalty we place on the L1-/L2-norms: we introduce some bias but reduce the variance of the model through regularization.

The contours have a higher probability of hitting the corners on the axes compared to elsewhere on the L1-ball (a polytope), while they have an equal chance of hitting the axes and everywhere else on the L2-ball (a circle), thus decreasing the overall probability of intersecting on an axis. Since it is easier for the contours to hit an axis with the L1-ball, we are more likely to see zero coefficients with lasso; by definition, this is how sparsity comes about.

Question 3.1

In [34]:
```
# Load data
ligo = pd.read_csv("LIGO.Hanford.Data.txt", sep=" ", header=0)
ligo.columns = ["Time (seconds)", "Strain*1e21"]

# Amplitude
y = ligo["Strain*1e21"]
y = array(y)

# Number of features
T = len(ligo["Strain*1e21"])
```

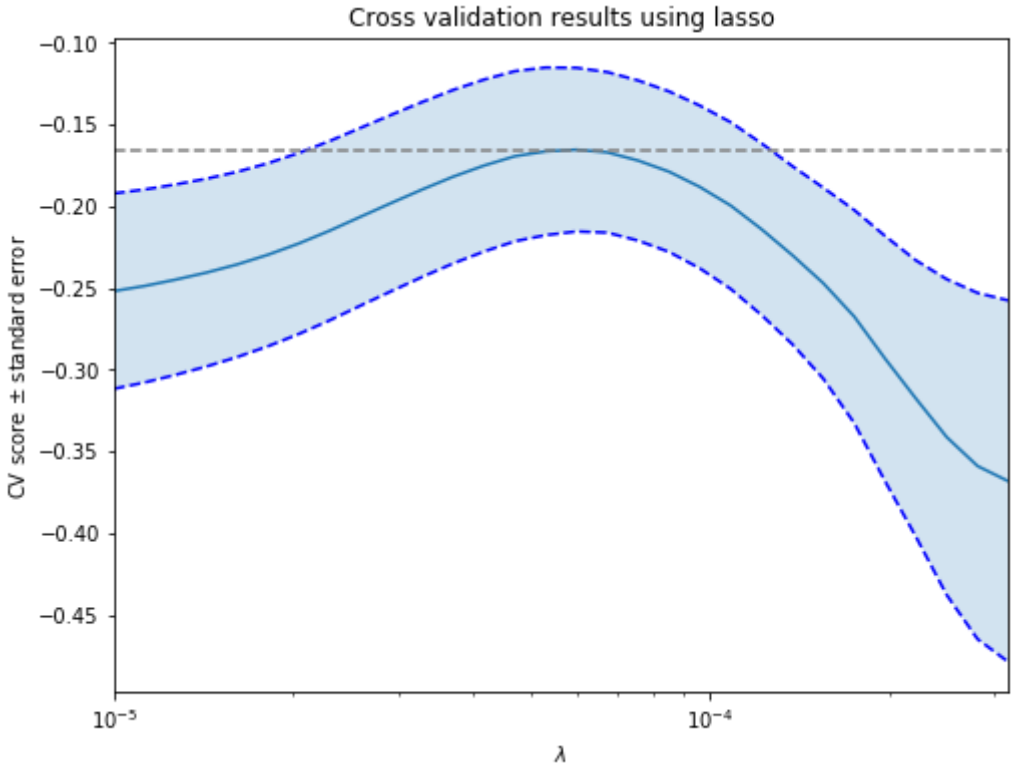
In [35]:
```
# Row 1 of matrix C
C1 = [np.sqrt(1/T)]*T

# Rows 2 through T of matrix C
## Generate the grid of indices
j = list(range(2, T+1))
k = list(range(1, T+1))
grids = list(product(j, k))

## Find the matrix elements
def waves(j, k):
    result = np.sqrt(2/T)*math.cos(math.pi*(2*k-1)*(j-1)/(2*T))

    return result

C2 = [waves(*pair) for pair in grids]
C2 = array(C2)

## Reshape array into a (T-1)*T matrix
shape = (T-1, T)
C2 = C2.reshape(shape)

# Final matrix C
C = np.vstack((C1, C2))
```

In [36]:
```
random.seed(10)

# Denoise using lasso
lasso = Lasso(random_state=0, max_iter=100000, tol=1)
alphas = np.logspace(-5, -3.5, 30)

tuned_parameters = [{"alpha": alphas}]
n_folds = 10

clf = GridSearchCV(lasso, tuned_parameters, cv=n_folds, refit=False)
clf.fit(C, y)
scores = clf.cv_results_["mean_test_score"]
scores_std = clf.cv_results_["std_test_score"]

# Best hyperparameter determined by lasso
best_lambda = [clf.best_params_["alpha"]]
print("The best hyperparameter chosen by lasso is 5.9684569951223044e-0
5.")
```

```
The best hyperparameter chosen by lasso is 5.9684569951223044e-05.
```

In [37]:
```
# Plot the cross valiadation results
plt.figure().set_size_inches(8, 6)
plt.semilogx(alphas, scores)

# Plot error lines showing +/- standard errors of the scores
std_error = scores_std / np.sqrt(n_folds)

plt.semilogx(alphas, scores + std_error, "b--")
plt.semilogx(alphas, scores - std_error, "b--")

# Fill in
plt.fill_between(alphas, scores + std_error, scores - std_error, alpha=
0.2)

plt.title("Cross validation results using lasso")
plt.xlabel(r"$\lambda$")
plt.ylabel(r"CV score $\pm$ standard error")
plt.axhline(np.max(scores), linestyle="--", color="0.5")
plt.xlim([alphas[0], alphas[-1]])
```

Out[37]:
```
(1e-05, 0.00031622776601683794)
```



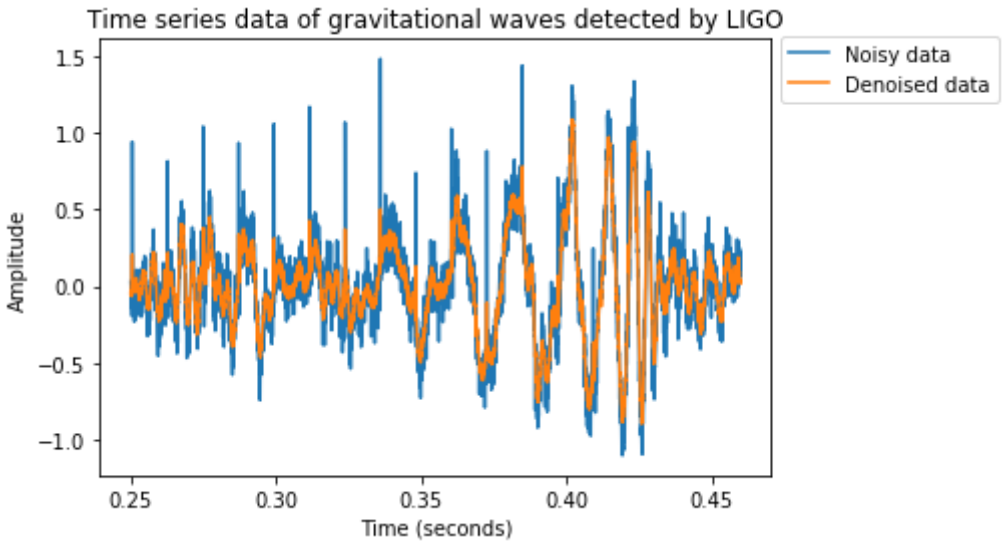
In [38]:
```
# Find the coefficients that optimizes the lasso regression
clf = Lasso(alpha=best_lambda)
clf.fit(C, y)
w = clf.coef_
print("The coefficients determined by this denoising procedure are {}.".
format(w))

# Number of non-zero coefficients
print("The number of non-zero coefficients determined by this denoising
 procedure is {}.".
      format(np.count_nonzero(clf.coef_)))
```

```
The coefficients determined by this denoising procedure are [-0.
-0.         -0.09353316 ... -0.          0.
 -0.11186574].
The number of non-zero coefficients determined by this denoising proced
ure is 417.
```

In [39]:
```
# Find the denoised amplitudes
yhat1 = C@w
```

In [40]:
```
# Plot the signals with and without noise superimposed on each other
plt.plot(ligo["Time (seconds)"], y)
plt.plot(ligo["Time (seconds)"], yhat1)

plt.title("Time series data of gravitational waves detected by LIGO")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.legend(["Noisy data", "Denoised data"], bbox_to_anchor=(1, 1.03))
```

Out[40]: <matplotlib.legend.Legend at 0x16cf5bed0>



From the plot above, we can see that the denoising procedure reduced a considerable amount of noise.

In [41]:
```
# Number of non-zero coefficients
print("The number of non-zero coefficients determined by this denoising
 procedure is {}.".
      format(np.count_nonzero(w)))

print("The number of non-zero coefficients without denoising is {}.".for
mat(np.count_nonzero(np.linalg.inv(C)@y)))
```

```
The number of non-zero coefficients determined by this denoising proced
ure is 417.
The number of non-zero coefficients without denoising is 3441.
```

From here, we can see that $\hat{w}$ is appreciably more sparse than $C^{-1}y$. This is because $\hat{w}$ has been regularized with many multicollinear terms reduced to zero, while $C^{-1}y$ still carries the noise without any manipulation on correlated features.

Question 3.2

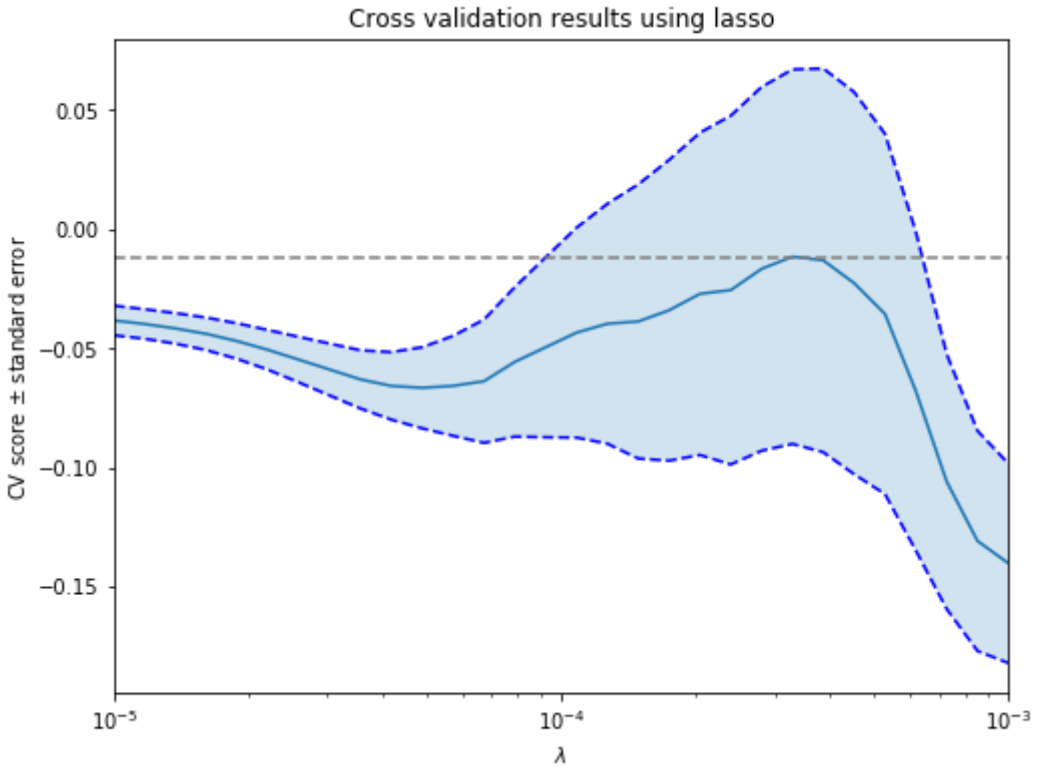
In [42]:
```
# Create the new matrix
I = np.identity(T)
psi = np.vstack((I, C)).T
```

In [43]:
```
random.seed(10)

# Denoise using lasso
lasso = Lasso(random_state=0, max_iter=100000, tol=1)
alphas = np.logspace(-5, -3, 30)

tuned_parameters = [{"alpha": alphas}]
n_folds = 10

clf = GridSearchCV(lasso, tuned_parameters, cv=n_folds, refit=False)
clf.fit(psi, y)
scores = clf.cv_results_["mean_test_score"]
scores_std = clf.cv_results_["std_test_score"]

# Best hyperparameter determined by lasso
best_lambda = [clf.best_params_["alpha"]]
print("The best hyperparameter chosen by lasso is 0.000329034456231266
6.")
```

```
The best hyperparameter chosen by lasso is 0.0003290345623126676.

/opt/anaconda3/lib/python3.7/site-packages/sklearn/model_selection/_sea
rch.py:814: DeprecationWarning: The default of the `iid` parameter will
change from True to False in version 0.22 and will be removed in 0.24.
This will change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

In [44]:
```
# Plot the cross valiadation results
plt.figure().set_size_inches(8, 6)
plt.semilogx(alphas, scores)

# Plot error lines showing +/- standard errors of the scores
std_error = scores_std / np.sqrt(n_folds)

plt.semilogx(alphas, scores + std_error, "b--")
plt.semilogx(alphas, scores - std_error, "b--")

# Fill in
plt.fill_between(alphas, scores + std_error, scores - std_error, alpha=
0.2)

plt.title("Cross validation results using lasso")
plt.xlabel(r"$\lambda$")
plt.ylabel(r"CV score $\pm$ standard error")
plt.axhline(np.max(scores), linestyle="--", color="0.5")
plt.xlim([alphas[0], alphas[-1]])
```

Out[44]: (1e-05, 0.001)



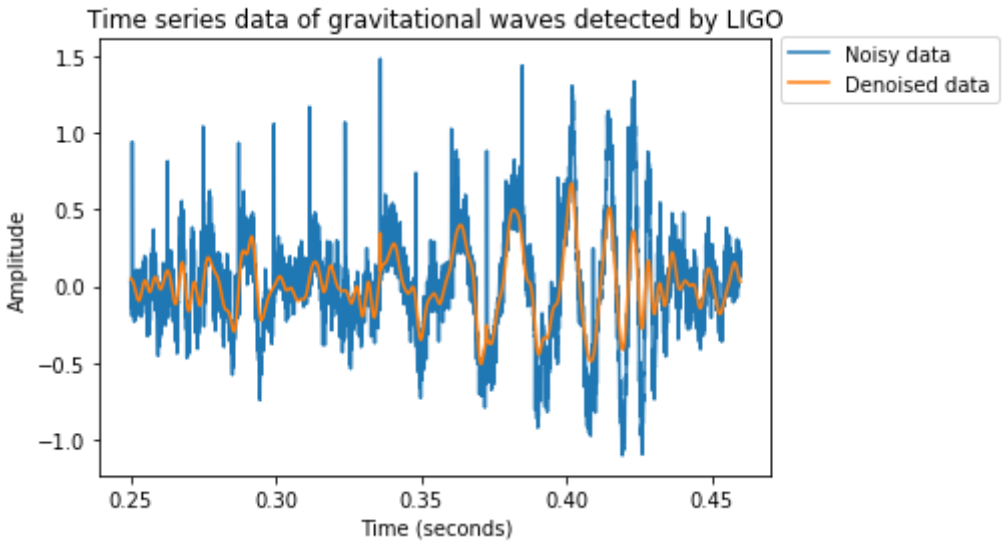
In [45]:
```
# Find the coefficients that optimizes the lasso regression
clf = Lasso(alpha=best_lambda)
clf.fit(psi, y)
w = clf.coef_
print("The coefficients determined by this denoising procedure are {}.".
format(w))

# Number of non-zero coefficients
print("The number of non-zero coefficients determined by this denoising
 procedure is {}.".
      format(np.count_nonzero(w)))
```

```
The coefficients determined by this denoising procedure are [-0. -0. -
0. ... -0.  0. -0.].
The number of non-zero coefficients determined by this denoising proced
ure is 57.
```

In [46]:
```
# Find the denoised amplitudes
yhat2 = psi@w
```

In [47]:
```
# Plot the signals with and without noise superimposed on each other
plt.plot(ligo["Time (seconds)"], y)
plt.plot(ligo["Time (seconds)"], yhat2)

plt.title("Time series data of gravitational waves detected by LIGO")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.legend(["Noisy data", "Denoised data"], bbox_to_anchor=(1, 1.03))
```

Out[47]: <matplotlib.legend.Legend at 0x132a93350>



From the plot above, we can see that the denoising procedure reduced a considerable amount of noise - even more so than the method in Question 3.1. The amplitudes are also a lot cleaner with appreciably fewer spikes. This indicates that the denoising process is working as intended.

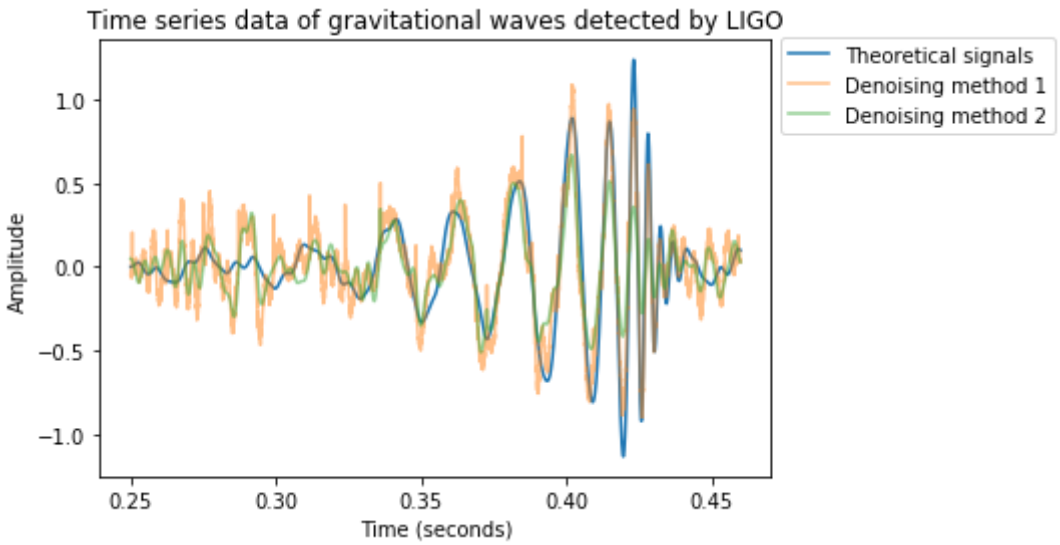
In [48]:
```
# Number of non-zero coefficients
print("The number of non-zero coefficients determined by this denoising
 procedure is {}.".
      format(np.count_nonzero(w)))

print("The number of non-zero coefficients without denoising is {}.".for
mat(np.count_nonzero(np.linalg.inv(C)@y)))
```

```
The number of non-zero coefficients determined by this denoising proced
ure is 57.
The number of non-zero coefficients without denoising is 3441.
```

From here, we can see that $\tilde{w}$ is appreciably more sparse than $C^{-1}y$ - even more so than the method in Question 3.1. This is because $\tilde{w}$ has been regularized with many multicollinear terms reduced to zero, while $C^{-1}y$ still carries the noise without any manipulation on correlated features.

In [49]:
```
# Compare results
theoretical = pd.read_csv("LIGO.Hanford.Theory.txt", sep=" ")
theoretical.columns = ["Time (seconds)", "Strain*1e21"]

# Create the plot
plt.plot(theoretical["Time (seconds)"], theoretical["Strain*1e21"])
plt.plot(theoretical["Time (seconds)"], yhat1[0:3440], alpha=0.5)
plt.plot(theoretical["Time (seconds)"], yhat2[0:3440], alpha=0.5)

plt.title("Time series data of gravitational waves detected by LIGO")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.legend(["Theoretical signals", "Denoising method 1", "Denoising meth
od 2"], bbox_to_anchor=(1, 1.03))
```

Out[49]:
```
<matplotlib.legend.Legend at 0x131e3a610>
```



From the plot above, we can see that the signal denoised using the method in Question 3.2 is closer to the theoretical signal at smaller amplitudes. At larger amplitudes, the signal denoised using the method in Question 3.1 matches the theoretical values more with more prominent spikes. However, denoising method 2 produces cleaner and steadier signals without losing the integrity of the overall trajectory. Therefore, I think the denoising method from Question 3.2 is more meaningful and easier to interpret.

In [50]:
```
# Write sound
# scaled1 = np.int16(yhat1/np.max(np.abs(yhat1)) * 32767)
# write("waves1.wav", 3441, scaled1)

# scaled2 = np.int16(yhat2/np.max(np.abs(yhat2)) * 32767)
# write("waves2.wav", 3441, scaled2)
```

In [51]:
```
# Import sound
waves1 = AudioSegment.from_wav("waves1.wav")
waves2 = AudioSegment.from_wav("waves2.wav")
```

In [52]:
```
# Prepare to be amazed (!!!)
playback.play(waves1)
```

In [53]:
```
# Prepare to be amazed again
playback.play(waves2)
```

From the audio clips, it is clear that the denoising method from Question 3.2 produced a distinct "wave" sound, while the method from Question 3.1 sounded more like drumrolls. This confirms again that the second method is more meaningful.

BST 263 HOMEWORK 2

Jenny Wang (71401898)

2/25/20

Question 2.2

Zou and Hastie (2005) stated that the naïve elastic net problem can be turned into an equivalent lasso problem on augmented data[1]. We demonstrate its proof here.

Given:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}_{n\times 1} \quad ; \quad \mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}_{n\times p} \quad ; \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta p \end{pmatrix}_{p\times 1}$$

The elastic net regularizer is defined as:

$$\min_{\boldsymbol{\beta}} \Big\{ ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda[\alpha||\boldsymbol{\beta}||_2^2 + (1-\alpha)||\boldsymbol{\beta}||_1] \Big\}.$$

We can rewrite this as:

$$\min_{\boldsymbol{\beta}} \Big\{ ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda_2||\boldsymbol{\beta}||_2^2 + \lambda_1||\boldsymbol{\beta}||_1 \Big\}, \text{ where } \lambda_1 = \lambda(1-\alpha) \text{ and } \lambda_2 = \lambda\alpha.$$

Generate augmented data (interestingly, this is equivalent to adding p more observations to the original dataset):

$$\text{Let } \tilde{\mathbf{X}}_{(n+p)\times p} = \tfrac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I} \end{pmatrix}; \qquad \tilde{\mathbf{y}}_{(n+p)\times 1} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}.$$

We can write:

$$\tilde{\mathbf{X}}\boldsymbol{\beta} = \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} \mathbf{X}\boldsymbol{\beta} \\ \sqrt{\lambda_2}\beta_1 \\ \sqrt{\lambda_2}\beta_2 \\ \vdots \\ \sqrt{\lambda_2}\beta_p \end{pmatrix}$$

$$\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta} = \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \\ -\sqrt{\lambda_2}\beta_1 \\ -\sqrt{\lambda_2}\beta_2 \\ \vdots \\ -\sqrt{\lambda_2}\beta_p \end{pmatrix}$$

$$\begin{aligned} ||\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}||_2^2 &= \frac{1}{\sqrt{1+\lambda_2}} \Big(||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda_2\beta_1^2 + \lambda_2\beta_2^2 + \cdots + \lambda_2\beta_p^2 \Big) \\ &= \frac{1}{\sqrt{1+\lambda_2}} \Big(\underbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda_2||\boldsymbol{\beta}||_2^2}_{\text{ridge regularizer}} \Big) \end{aligned}$$

Now, we can represent the ridge regularizer using the augmented $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$:

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda_2||\boldsymbol{\beta}||_2^2 = \sqrt{1+\lambda_2}||\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}||_2^2.$$

The elastic net regularizer now becomes:

$$\min_{\boldsymbol{\beta}}\left\{\sqrt{1+\lambda_2}||\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}||_2^2 + \lambda_1||\boldsymbol{\beta}||_1\right\}.$$

We can divide out the constants and obtain:

$$\min_{\boldsymbol{\beta}}\left\{\underbrace{||\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}||_2^2 + \tilde{\lambda}||\boldsymbol{\beta}||_1}_{\text{lasso regularizer}}\right\}, \text{ where } \tilde{\lambda} = \frac{\lambda_1}{\sqrt{1+\lambda_2}}.$$

Therefore, the elastic net regularization is a special case of lasso regularization with augmented data. QED.

Question 2.3

A few axioms that will come in handy; color-coded for ease of visualization:

- $\text{Var}[Z] = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2$
- $Y_0 = \beta_0 + \epsilon$
- $\mathbb{E}[\epsilon] = 0$; $\mathbb{E}[\epsilon\hat{\beta}(\lambda)] = \mathbb{E}[\epsilon]\mathbb{E}[\hat{\beta}(\lambda)] = 0$ given $\epsilon \perp\!\!\!\perp \hat{\beta}(\lambda)$
- $\text{Cov}[Z, W] = \mathbb{E}[ZW] - \mathbb{E}[Z]\mathbb{E}[W]$
- $\text{Var}[Z, W] = \text{Var}[Z] + \text{Var}[W] - 2\text{Cov}[Z, W]$

$$\begin{aligned}
\text{Tested MSE} &= \mathbb{E}[Y_0 - \hat{\beta}(\lambda)]^2 \\
&= \mathbb{E}[Y_0^2] + \mathbb{E}[\hat{\beta}(\lambda)^2] - 2\mathbb{E}[Y_0\hat{\beta}(\lambda)] \\
&= \text{Var}[Y_0] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[Y_0\hat{\beta}(\lambda)] \\
&= \text{Var}[Y_0] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[(\beta_0 + \epsilon)\hat{\beta}(\lambda)] \\
&= \text{Var}[Y_0] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[\beta_0\hat{\beta}(\lambda)] \cancelto{0}{-2\mathbb{E}[\epsilon\hat{\beta}(\lambda)]} \\
&= \text{Var}[Y_0] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[\beta_0]\mathbb{E}[\hat{\beta}(\lambda)] \\
&= \text{Var}[Y_0] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2(\mathbb{E}[\beta_0\hat{\beta}(\lambda)] - \text{Cov}[\beta_0, \hat{\beta}(\lambda)]) \\
&= \text{Var}[Y_0] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[\beta_0]\mathbb{E}[\hat{\beta}(\lambda)] - 2\text{Cov}[\beta_0, \hat{\beta}(\lambda)] \\
&= \text{Var}[\beta_0] + \text{Var}[\epsilon] + \mathbb{E}[Y_0]^2 + \text{Var}[\hat{\beta}(\lambda)] + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[\beta_0]\mathbb{E}[\hat{\beta}(\lambda)] - 2\text{Cov}[\beta_0, \hat{\beta}(\lambda)] \\
&= \text{Var}[\beta_0 - \hat{\beta}(\lambda)] + \text{Var}[\epsilon] + \mathbb{E}[\beta_0] \cancelto{0}{+\mathbb{E}[\epsilon]} + \mathbb{E}[\hat{\beta}(\lambda)]^2 - 2\mathbb{E}[\beta_0]\mathbb{E}[\hat{\beta}(\lambda)] \\
&= \underbrace{\text{Var}[\beta_0 - \hat{\beta}(\lambda)]}_{\text{Var}[\hat{\beta}(\lambda)]} + \underbrace{(\mathbb{E}[\beta_0] - \mathbb{E}[\hat{\beta}(\lambda)])^2}_{\mathbb{E}[Y_0 - \hat{\beta}(\lambda)]^2} + \underbrace{\text{Var}[\epsilon]}_{\text{noise}}
\end{aligned}$$

From here, we have:

$$\begin{aligned}
\text{Bias} &= \mathbb{E}[Y_0 - \hat{\beta}(\lambda)] = \mathbb{E}[\beta_0] - \mathbb{E}[\hat{\beta}(\lambda)]] \\
\text{Variance} &= \text{Var}[\hat{\beta}(\lambda)] = \text{Var}[\beta_0 - \hat{\beta}(\lambda)] \\
\text{Tested MSE} &= \mathbb{E}[Y_0 - \hat{\beta}(\lambda)]^2 = \underbrace{\text{Var}[\beta_0 - \hat{\beta}(\lambda)]}_{\text{variance}} + \underbrace{(\mathbb{E}[\beta_0] - \mathbb{E}[\hat{\beta}(\lambda)])^2}_{\text{bias}^2} + \underbrace{\text{Var}[\epsilon]}_{\text{noise}}
\end{aligned}$$

We can also see that as $\lambda \to \infty$, the bias would increase while the variance would decrease. The tested MSE would also increase because the rate of increase of the bias (second order) is greater compared to the rate of decrease of the variance (first order).

References

1. Zou, H., and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B.* **67**(2):301-320.