In [1]:
```
import pandas as pd

# Load the data
file_path = "/Users/macbookpro/Desktop/ECON331 HW1/TÜİK 2013-2023 TR-DEU/Export.xlsx"
sheet_name = "Use this"
trade_data = pd.read_excel(file_path, sheet_name=sheet_name)

# Display the first few rows of the dataframe to understand its structure
print(trade_data.head())
```

```
   Year    TOTAL USD
0  2013  40430998510
1  2014  39757516843
2  2015  37226299584
3  2016  37812506839
4  2017  38637760895
```

In [2]:
```
# Load Turkey GDP data
turkey_gdp_file_path = "/Users/macbookpro/Desktop/ECON331 HW1/TurkeyGDP.xls"
turkey_gdp_sheet_name = "Turkey GDP"
turkey_gdp_data = pd.read_excel(turkey_gdp_file_path, sheet_name=turkey_gdp_sheet_name)
```

In [3]:
```
# Load Germany GDP data
germany_gdp_file_path = "/Users/macbookpro/Desktop/ECON331 HW1/GermanyGDP.xls"
germany_gdp_sheet_name = "Germany GDP"
germany_gdp_data = pd.read_excel(germany_gdp_file_path, sheet_name=germany_gdp_sheet_name)
```

In [4]:
```
# Merge trade data with Turkey GDP data
merged_data = pd.merge(trade_data, turkey_gdp_data, on='Year', how='inner')

# Merge merged data with Germany GDP data
merged_data = pd.merge(merged_data, germany_gdp_data, on='Year', how='inner')
```

In [5]:
```
# Rearrange columns
merged_data = merged_data[['Year', 'TOTAL USD', 'Germany GDP', 'Turkey GDP']]
```

In [6]:
```
# Display the resulting dataframe
merged_data
```

Out[6]:

	Year	TOTAL USD	Germany GDP	Turkey GDP
0	2013	40430998510	3.733805e+12	9.577991e+11
1	2014	39757516843	3.889093e+12	9.389346e+11
2	2015	37226299584	3.357586e+12	8.643138e+11
3	2016	37812506839	3.469853e+12	8.696829e+11
4	2017	38637760895	3.690849e+12	8.589885e+11
5	2018	38888666525	3.974443e+12	7.789722e+11
6	2019	35897642733	3.889178e+12	7.610059e+11
7	2020	37711457138	3.887727e+12	7.203385e+11
8	2021	41037327626	4.278504e+12	8.198653e+11
9	2022	45174857021	4.082469e+12	9.071184e+11
10	2023	49763197518	4.509469e+12	1.112118e+12

In [7]:
```
import numpy as np
import statsmodels.api as sm
```

In [8]:
```
# Add Distance column with constant value of 2350 kilometers for the years 2013 to 2023
merged_data['Distance'] = 2350
```

In [9]:
```
# Initial guess for parameters
initial_guess = [0.5, 1, 1]
```

In [10]:
```
# Extract independent variables
X = merged_data[['Germany GDP', 'Turkey GDP', 'Distance']].values
```

In [11]:
```
# Extract dependent variable
y = merged_data['TOTAL USD'].values
```

In [12]:
```
from scipy.optimize import curve_fit
import numpy as np

# Define the gravity model function
def gravity_model(X, beta, n, constant):
    return constant + (beta * (X[:, 0] * X[:, 1])) / (X[:, 2] ** n)

# Rescale the variables
X_rescaled = X / np.max(X, axis=0)
y_rescaled = y / np.max(y)

# Initial guess for parameters
initial_guess = [0.5, 1, 1]

# Perform nonlinear least squares (NLS) regression
popt, _ = curve_fit(gravity_model, X_rescaled, y_rescaled, initial_guess)

# Extract estimated parameters
beta, n, constant = popt

# Print the estimated parameters
print("Estimated beta:", beta)
print("Estimated n:", n)
print("Estimated constant:", constant)
```

```
Estimated beta: 0.6076928959513533
Estimated n: 1.0
Estimated constant: 0.39608622960928297

/opt/anaconda3/lib/python3.9/site-packages/scipy/optimize/minpack.py:833: OptimizeWarning: Covariance of the parameters could not be estimated
  warnings.warn('Covariance of the parameters could not be estimated',
```

In [13]:
```
from scipy.stats import t
```

In [14]:
```
# Calculate standard errors for parameters
beta_std_err = np.sqrt(np.diag(np.linalg.inv(np.dot(X_rescaled.T, X_rescaled))))
t_values = beta / beta_std_err
```

In [15]:
```
# Degrees of freedom
df = len(y_rescaled) - len(initial_guess)
```

In [16]:
```
# Calculate p-values for variables
p_values = 2 * (1 - t.cdf(np.abs(t_values), df))
```

In [17]:
```
# Print p-values for variables
print("P-values for Variables:")
print("Product of GDPs:", p_values[0])
print("Distance:", p_values[1])
```

```
P-values for Variables:
Product of GDPs: 0.8973610826324747
Distance: 0.8644004376758858
```

In [18]:
```
# Compute predicted values
y_pred_rescaled = gravity_model(X_rescaled, beta, n, constant)
```

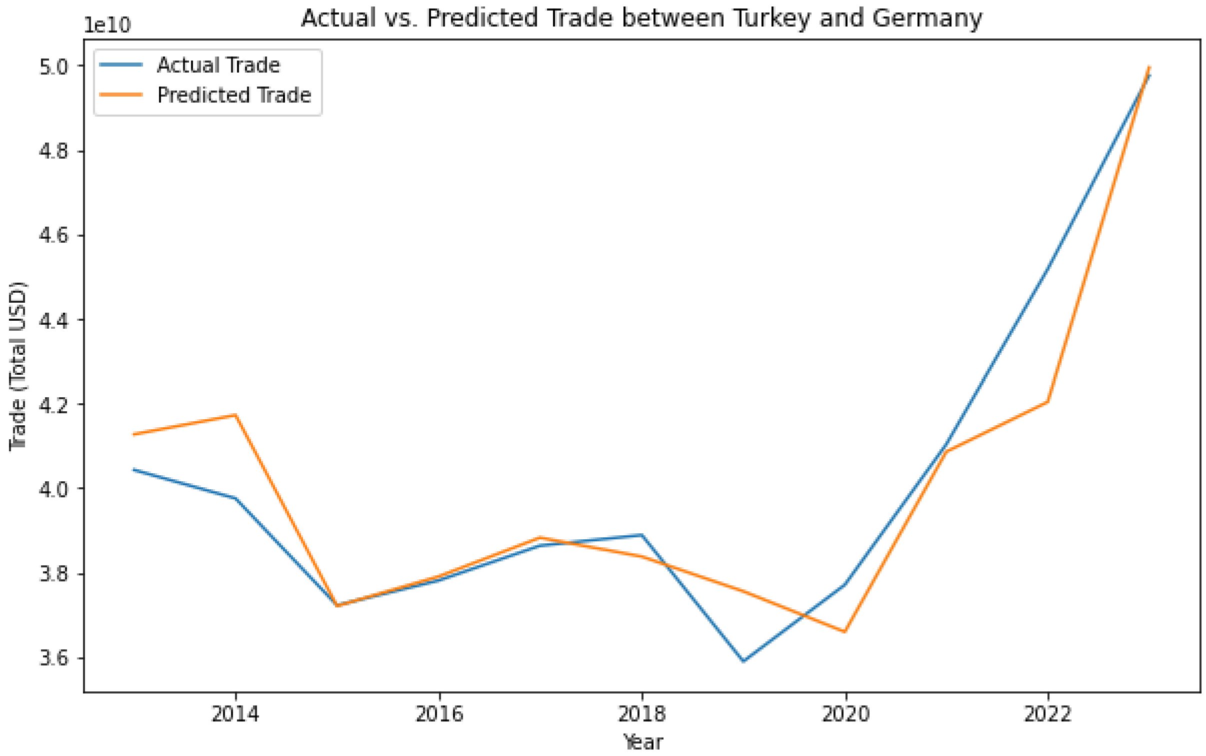
In [19]:
```
# Compute R-squared (R^2)
ss_total = np.sum((y_rescaled - np.mean(y_rescaled)) ** 2)
ss_residual = np.sum((y_rescaled - y_pred_rescaled) ** 2)
rsquared = 1 - (ss_residual / ss_total)
print("R-squared (R^2):", rsquared)
```

```
R-squared (R^2): 0.8829831023533649
```

In [20]:
```
# Compute Mean Squared Error (MSE)
mse = np.mean((y_rescaled - y_pred_rescaled) ** 2)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 0.0006897210443292836
```

In [21]:
```
import matplotlib.pyplot as plt

# Set figure size
plt.figure(figsize=(10, 6))  # Adjust width and height as needed

# Denormalize the predicted trade values
y_pred = y_pred_rescaled * np.max(y)

# Plot actual trade values
plt.plot(merged_data['Year'], merged_data['TOTAL USD'], label='Actual Trade')

# Plot predicted trade values
plt.plot(merged_data['Year'], y_pred, label='Predicted Trade')

# Set plot labels and title
plt.xlabel('Year')
plt.ylabel('Trade (Total USD)')
plt.title('Actual vs. Predicted Trade between Turkey and Germany')
plt.legend()

# Display the plot
plt.show()
```



Including MATR on the RHS

In [22]:
```
# Load the MATR index data
matr_data = pd.read_excel("/Users/macbookpro/Desktop/ECON331 HW1/econ331 matr.xlsx")
```

In [23]:
```
# Encode "yes" as 1 and "no" as 0
matr_data['Status'] = matr_data['Status'].map({'yes': 1, 'no': 0})
```

In [24]:
```
# Sum "yes" for each country for each year
matr_summary = matr_data.groupby(['Year', 'Country'])['Status'].sum().reset_index()
```

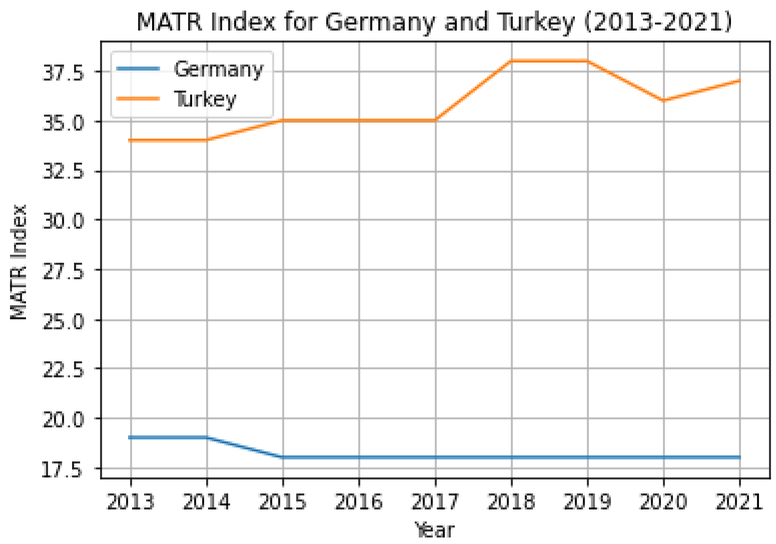
In [25]:
```
# Pivot the data to get MATR index for both countries throughout the years
matr_pivot = matr_summary.pivot(index='Year', columns='Country', values='Status').reset_index()
```

In [26]:
```
# Rename the columns for clarity
matr_pivot.columns.name = None  # Remove the 'Country' label from the columns
matr_pivot.columns = ['Year', 'MATR_Germany', 'MATR_Turkey']
```

In [27]:
```
# Output the table
print(matr_pivot)
```

```
   Year  MATR_Germany  MATR_Turkey
0  2013            19           34
1  2014            19           34
2  2015            18           35
3  2016            18           35
4  2017            18           35
5  2018            18           38
6  2019            18           38
7  2020            18           36
8  2021            18           37
```

In [28]:
```
import matplotlib.pyplot as plt

# Extract data
years = matr_pivot['Year']
matr_index_germany = matr_pivot['MATR_Germany']
matr_index_turkey = matr_pivot['MATR_Turkey']

# Plot MATR index values for Germany
plt.plot(years, matr_index_germany, label='Germany')

# Plot MATR index values for Turkey
plt.plot(years, matr_index_turkey, label='Turkey')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('MATR Index')
plt.title('MATR Index for Germany and Turkey (2013-2021)')

# Add legend
plt.legend()

# Show plot
plt.grid(True)
plt.show()
```



In [29]:
```
# Merge MATR index data with merged data
matr_merged_data = pd.merge(merged_data, matr_pivot, on='Year', how='inner')
```

In [30]:
```
matr_merged_data
```

Out[30]:

	Year	TOTAL USD	Germany GDP	Turkey GDP	Distance	MATR_Germany	MATR_Turkey
0	2013	40430998510	3.733805e+12	9.577991e+11	2350	19	34
1	2014	39757516843	3.889093e+12	9.389346e+11	2350	19	34
2	2015	37226299584	3.357586e+12	8.643138e+11	2350	18	35
3	2016	37812506839	3.469853e+12	8.696829e+11	2350	18	35
4	2017	38637760895	3.690849e+12	8.589885e+11	2350	18	35
5	2018	38888666525	3.974443e+12	7.789722e+11	2350	18	38
6	2019	35897642733	3.889178e+12	7.610059e+11	2350	18	38
7	2020	37711457138	3.887727e+12	7.203385e+11	2350	18	36
8	2021	41037327626	4.278504e+12	8.198653e+11	2350	18	37

In [31]:
```
# Calculate MATR index product
matr_merged_data['MATR_Product'] = matr_merged_data['MATR_Germany'] * matr_merged_data['MATR_Turkey']
```

In [32]:
```
# Extract independent variables
X = matr_merged_data[['Germany GDP', 'Turkey GDP', 'Distance', 'MATR_Product']].values

# Extract dependent variable
y = matr_merged_data['TOTAL USD'].values
```

In [33]:
```
# Define the gravity model function
def gravity_model_with_matr(X, beta1, n, constant, beta2):
    return constant + (beta1 * (X[:, 0] * X[:, 1]) / (X[:, 2] ** n)) + beta2 * X[:, 3]
```

In [34]:
```
# Initial guess for parameters
initial_guess = [0.5, 1, 1, 0.5]
```

In [35]:
```
# Perform nonlinear least squares (NLS) estimation using curve_fit
params, covariance = curve_fit(gravity_model_with_matr, X, y, p0=initial_guess)

# Extract estimated parameters
beta1, n, constant, beta2 = params
```

In [36]:
```
# Print the estimated parameters
print("Estimated beta1:", beta1)
print("Estimated n:", n)
print("Estimated constant:", constant)
print("Estimated beta2:", beta2)
```

```
Estimated beta1: 1.15406745732168e-05
Estimated n: 2.7933607126181954
Estimated constant: 27192357824.325413
Estimated beta2: -4120659.767225214
```

In [37]:
```
# Calculate standard errors for parameters
beta_std_err = np.sqrt(np.diag(np.linalg.inv(np.dot(X.T, X))))
t_values = params / beta_std_err
```

In [38]:
```
# Degrees of freedom
df = len(y) - len(params)
```

In [39]:
```
# Calculate p-values for parameters
p_values = 2 * (1 - t.cdf(np.abs(t_values), df))
```

In [40]:
```
# Print p-values for parameters with 3 decimals
print("P-values for Parameters (with 3 decimals):")
print("Estimated beta1:", format(p_values[0], '.3f'))
print("Estimated n:", format(p_values[1], '.3f'))
print("Estimated constant:", format(p_values[2], '.3f'))
print("Estimated beta2:", format(p_values[3], '.3f'))
```

```
P-values for Parameters (with 3 decimals):
Estimated beta1: 0.000
Estimated n: 0.000
Estimated constant: 0.000
Estimated beta2: 0.000
```

In [41]:
```
# Predicted values
predicted_values = gravity_model_with_matr(X, *params)
```

In [42]:
```
from sklearn.metrics import r2_score, mean_squared_error

# Calculate R-squared
r_squared = r2_score(y, predicted_values)

# Calculate mean squared error
mse = mean_squared_error(y, predicted_values)

# Print R-squared and MSE
print("R-squared (R^2):", r_squared)
print("Mean Squared Error (MSE):", mse)
```

```
R-squared (R^2): 0.7256402921164997
Mean Squared Error (MSE): 6.49841164470292e+17
```

In [43]:
```
# Set figure size
plt.figure(figsize=(10, 6))  # Adjust width and height as needed

# Plot actual trade data
plt.plot(matr_merged_data['Year'], matr_merged_data['TOTAL USD'], label='Actual Trade Data', marker='o')

# Plot fitted values
plt.plot(matr_merged_data['Year'], predicted_values, label='Fitted Values', marker='x')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Trade (Total USD)')
plt.title('Fit of Gravity Model with MATR Index to Actual Trade Data')
plt.legend()

# Show plot
plt.grid(True)
plt.show()
```