Exchange Rate Forecasting: Interest Rate Differential, Kalman Filter, Equity Curve and Analysis of Trading Strategy

In [224]:
```
import yfinance as yf
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import datetime
import io
import datetime
import matplotlib.lines as mlines
from fredapi import Fred
import statsmodels.formula.api as smf
import datetime
```

Question 1

a)

In [225]:
```
interest = pd.read_csv("/Users/snehilshandilya/Desktop/hw2_data.csv
```

In [226]:
```
interest
```

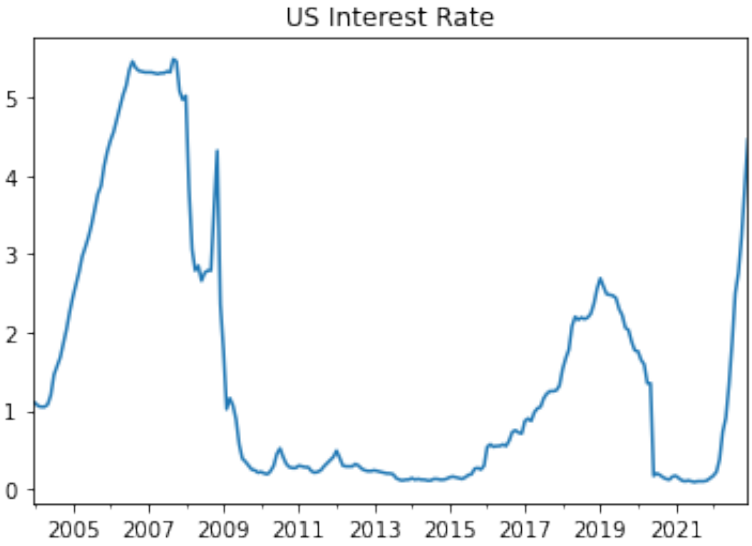
Out[226]:

	I_US	I_EU	Inf_US	Inf_EU	Euro
2003-12-01	1.11	2.1590	0.019284	0.021933	1.196501
2004-01-01	1.10	2.1463	0.020352	0.020207	1.258194
2004-02-01	1.06	2.0895	0.020263	0.018343	1.246805
2004-03-01	1.05	2.0706	0.016885	0.016654	1.244803
2004-04-01	1.05	2.0288	0.017401	0.017175	1.236507
...	...	...	...	...	...
2022-08-01	2.50	0.0366	0.084821	0.088662	1.020825
2022-09-01	2.76	0.3947	0.082492	0.091406	1.003905
2022-10-01	3.21	1.0109	0.082224	0.099272	0.982956
2022-11-01	3.85	1.4277	0.077631	0.106206	0.988631
2022-12-01	4.46	1.8252	0.071179	0.100546	1.042535

229 rows × 5 columns

In [227]:
```
interest["I_US"].dropna().plot()
plt.title("US Interest Rate")
```

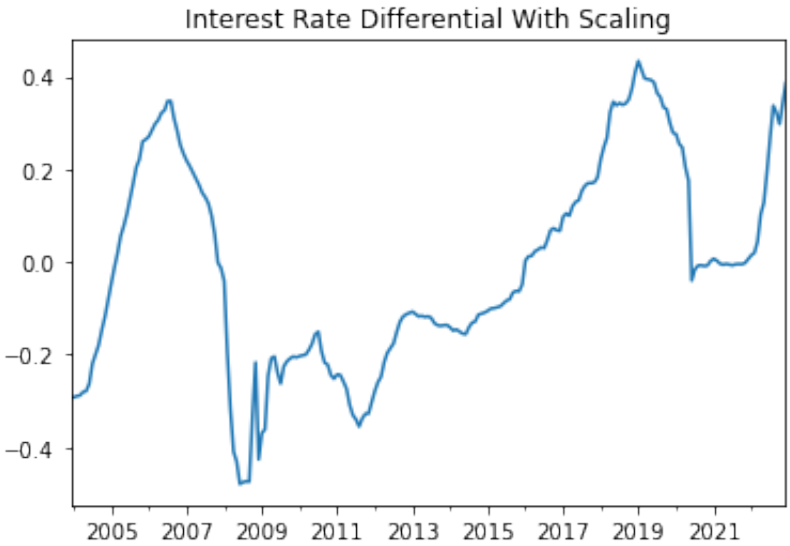
Out[227]:
```
Text(0.5, 1.0, 'US Interest Rate')
```



In [228]:
```
def scale(x):
    return (x-x.min())/(x.max()-x.min())

# Apply Scaling
interest["ir_diff"] = scale(interest["I_US"]) - scale(interest["I_E
interest.dropna(inplace= True)
```

In [229]:
```
plt.title("Interest Rate Differential With Scaling")
interest["ir_diff"].plot()
```

Out[229]:
```
<AxesSubplot:title={'center':'Interest Rate Differential With Scal
ing'}>
```



b)

In [230]:
```
k = 0.05
w = 10
z = 1.006667

interest["Filter"] = interest.ir_diff.ewm(alpha = k, adjust = False

# Compute the filter error
interest["Filter Error"] = interest.ir_diff - interest["Filter"]

# compute the rolling standard deviation
interest["std"] = interest["Filter Error"].rolling(w).std()

# create our confidence intervals or "boundaries of inaction"
# these are scaled by teh number of standard deviations "z"
interest["Upper"] = interest["Filter"] + z*interest["std"]
interest["Lower"] = interest["Filter"] - z*interest["std"]

# Create signal that evaluates whether we are outside the threshold
# then multiply by the direction of the mistake
# (we use economic theory to decide which direction is long or shor
interest["test"] = np.where(interest["Filter Error"].abs()>z*intere
```

In [231]:
```
interest["Filter Error"]
```

Out[231]:
```
2003-12-01    0.000000
2004-01-01    0.000359
2004-02-01    0.002779
2004-03-01    0.004034
2004-04-01    0.010805
                ...
2022-08-01    0.224920
2022-09-01    0.199681
2022-10-01    0.166075
2022-11-01    0.200837
2022-12-01    0.231803
Name: Filter Error, Length: 229, dtype: float64
```

In [232]:
```
interest["test"]
```

Out[232]:
```
2003-12-01    0.0
2004-01-01    0.0
2004-02-01    0.0
2004-03-01    0.0
2004-04-01    0.0
              ...
2022-08-01   -1.0
2022-09-01   -1.0
2022-10-01   -1.0
2022-11-01   -1.0
2022-12-01   -1.0
Name: test, Length: 229, dtype: float64
```

c)

In [233]:
```
drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
daily  = pd.DataFrame(index = drange)

# Integrate the monthly dta into the daily data
daily["test"] = interest["test"]

daily["Upper"] = interest["Upper"]
daily["Lower"] = interest["Lower"]
daily["Filter"] = interest["Filter"]
daily["ir_diff"] = interest["ir_diff"]

# Fill NA values with the last available value
daily["Upper"] = daily["Upper"].ffill()
daily["Lower"] = daily["Lower"].ffill()
daily["Filter"] = daily["Filter"].ffill()
daily["ir_diff"] = daily["ir_diff"].ffill()

# fill the remaining NA values with 0's
# also populates the test column
daily = daily.fillna(0)
```

In [234]:
```
daily
```

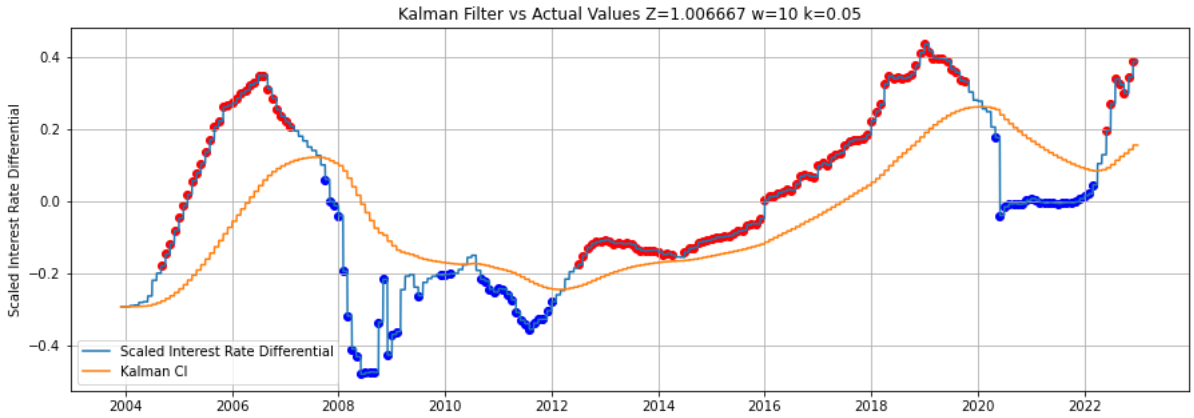
Out[234]:

	test	Upper	Lower	Filter	ir_diff
2003-12-01	0.0	0.000000	0.000000	-0.292402	-0.292402
2003-12-02	0.0	0.000000	0.000000	-0.292402	-0.292402
2003-12-03	0.0	0.000000	0.000000	-0.292402	-0.292402
2003-12-04	0.0	0.000000	0.000000	-0.292402	-0.292402
2003-12-05	0.0	0.000000	0.000000	-0.292402	-0.292402
...	...	...	...	...	...
2022-12-28	0.0	0.250421	0.059133	0.154777	0.386580
2022-12-29	0.0	0.250421	0.059133	0.154777	0.386580
2022-12-30	0.0	0.250421	0.059133	0.154777	0.386580
2022-12-31	0.0	0.250421	0.059133	0.154777	0.386580
2023-01-01	0.0	0.250421	0.059133	0.154777	0.386580

6972 rows × 5 columns

In [235]:
```
i = 171
daily.loc[:, str(i)+"_signal"] = 0
for j in daily.index:
    if daily.loc[j, "test"] != 0:
        daily.loc[j:j+datetime.timedelta(i), str(i)+"_signal"] = da
```

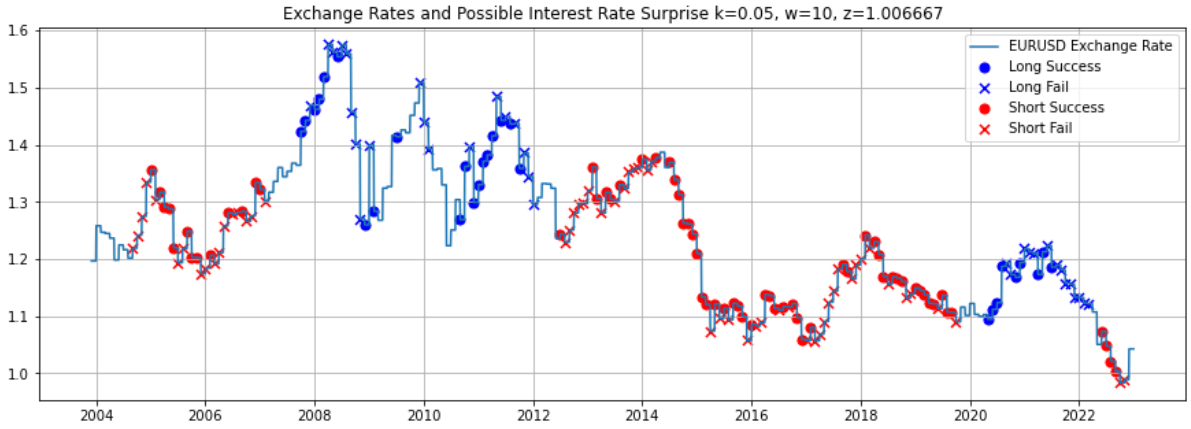
In [236]:
```
data2 = daily.dropna()
fig, ax = plt.subplots(figsize = (15, 5))
ax.set_title("Kalman Filter vs Actual Values " + "Z="+str(z) + " w=

ax.set_ylabel("Scaled Interest Rate Differential")

# Plot the actual series and the filter
ax.plot(data2["ir_diff"])
ax.plot(data2["Filter"])

# This code block is used to add confidence intervals when z > 0
#ax.fill_between(data2.index, data2.Lower, data2.Upper, color='b',

# add scatterplots using boolean indexing
# We change the colors and shapes based on the conditions
ax.scatter(data2[data2.test == 1].index, data2[data2.test == 1]["ir
ax.scatter(data2[data2.test == -1].index, data2[data2.test == -1]["
ax.legend(["Scaled Interest Rate Differential", "Kalman CI"])

# this code can let us zoom in on certain time periods
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
ax.grid()
```



In [237]:
```
drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
exdf = pd.DataFrame(index = drange)

exdf["EURUSD"] = interest["Euro"]
exdf["EURUSD"] = exdf["EURUSD"].ffill()

daily["EURUSD"] = exdf["EURUSD"]
daily["Returns"] = np.log(daily["EURUSD"]).diff()
```

In [239]:
```
i = 171
daily[str(i)+"_returns"] = np.exp((daily[str(i)+"_signal"].shift()*
s = 171
daily[str(s)+"_success"] = ((daily[daily.test!= 0][str(s)+"_returns
```

In [240]:
```
plt.figure(figsize = (15, 5))

plt.title("Exchange Rates and Possible Interest Rate Surprise"+ " k

plt.plot(daily["EURUSD"])

longsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test"
longfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"] =
shortsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test
shortfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"]

plt.scatter(longsuccess.index, longsuccess["EURUSD"], color = 'blue
plt.scatter(longfail.index, longfail["EURUSD"], color = 'blue', s =

plt.scatter(shortsuccess.index, shortsuccess["EURUSD"], color = 're
plt.scatter(shortfail.index, shortfail["EURUSD"], color = 'red', s

plt.legend(["EURUSD Exchange Rate", "Long Success", "Long Fail", "S
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
plt.grid()
```



Question 2

a)

In [241]:
```
#currency = "EURUSD=X"
#exrate = yf.download(currency)[["Adj Close"]].copy()
#exrate = exrate.set_index(exrate.index.date)
```

In [242]:
```
interest
```

Out[242]:

	I_US	I_EU	Inf_US	Inf_EU	Euro	ir_diff	Filter	Filter Error	std
2003-12-01	1.11	2.1590	0.019284	0.021933	1.196501	-0.292402	-0.292402	0.000000	NaN
2004-01-01	1.10	2.1463	0.020352	0.020207	1.258194	-0.292024	-0.292383	0.000359	NaN
2004-02-01	1.06	2.0895	0.020263	0.018343	1.246805	-0.289458	-0.292237	0.002779	NaN
2004-03-01	1.05	2.0706	0.016885	0.016654	1.244803	-0.287991	-0.292025	0.004034	NaN
2004-04-01	1.05	2.0288	0.017401	0.017175	1.236507	-0.280651	-0.291456	0.010805	NaN
...	...	...	...	...	...	...	...	...	...
2022-08-01	2.50	0.0366	0.084821	0.088662	1.020825	0.337677	0.112756	0.224920	0.113755
2022-09-01	2.76	0.3947	0.082492	0.091406	1.003905	0.322946	0.123266	0.199681	0.118293
2022-10-01	3.21	1.0109	0.082224	0.099272	0.982956	0.298081	0.132006	0.166075	0.112975
2022-11-01	3.85	1.4277	0.077631	0.106206	0.988631	0.343414	0.142577	0.200837	0.106033
2022-12-01	4.46	1.8252	0.071179	0.100546	1.042535	0.386580	0.154777	0.231803	0.095010

229 rows × 12 columns

In [243]:
```
drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
exdf = pd.DataFrame(index = drange)

exdf["EURUSD"] = interest["Euro"]
exdf["EURUSD"] = exdf["EURUSD"].ffill()

daily["EURUSD"] = exdf["EURUSD"]
daily["Returns"] = np.log(daily["EURUSD"]).diff()
```

In [244]:
```
drange
```

Out[244]:
```
DatetimeIndex(['2003-12-01', '2003-12-02', '2003-12-03', '2003-12-
04',
               '2003-12-05', '2003-12-06', '2003-12-07', '2003-12-
08',
               '2003-12-09', '2003-12-10',
               ...
               '2022-12-23', '2022-12-24', '2022-12-25', '2022-12-
26',
               '2022-12-27', '2022-12-28', '2022-12-29', '2022-12-
30',
               '2022-12-31', '2023-01-01'],
              dtype='datetime64[ns]', length=6972, freq='D')
```

In [245]:
```
exdf
```

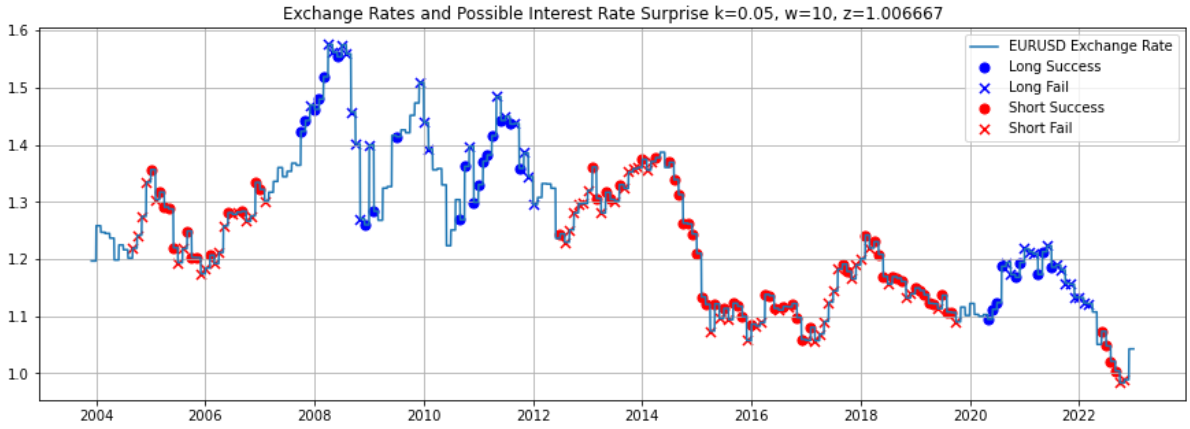
Out[245]:

	EURUSD
2003-12-01	1.196501
2003-12-02	1.196501
2003-12-03	1.196501
2003-12-04	1.196501
2003-12-05	1.196501
...	...
2022-12-28	1.042535
2022-12-29	1.042535
2022-12-30	1.042535
2022-12-31	1.042535
2023-01-01	1.042535

6972 rows × 1 columns

In [246]:
```
i = 171
daily[str(i)+"_returns"] = np.exp((daily[str(i)+"_signal"].shift()*
s = 171
daily[str(s)+"_success"] = ((daily[daily.test!= 0][str(s)+"_returns
```

In [247]:
```
plt.figure(figsize = (15, 5))

plt.title("Exchange Rates and Possible Interest Rate Surprise"+ " k

plt.plot(daily["EURUSD"])

longsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test"
longfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"] =
shortsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test
shortfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"]

plt.scatter(longsuccess.index, longsuccess["EURUSD"], color = 'blue
plt.scatter(longfail.index, longfail["EURUSD"], color = 'blue', s =

plt.scatter(shortsuccess.index, shortsuccess["EURUSD"], color = 're
plt.scatter(shortfail.index, shortfail["EURUSD"], color = 'red', s

plt.legend(["EURUSD Exchange Rate", "Long Success", "Long Fail", "S
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
plt.grid()
```



The graph here shows us how our strategy performs with respect to the actual changes in interest rate differential. As the graph depicts, it shows where all we took a long and short position and whether it was right or not. The success of a strategy can be better shown using the Equity Curve.

b) Equity Curve Visualisation

In [248]:
```
plt.figure(figsize = (15, 5))
d2 = daily[daily.index.year >= 2020]
(daily["171_returns"].dropna()).plot()

plt.title("Interest Differential Strategy:"+ " k=" + str(k)+", w="+
plt.legend(["Equity Curve", "Long Success", "Long Fail", "Short Suc

plt.grid()
#plt.xlim(["1/1/2019", "1/1/2022"])
```

The equity curve shows us how our strategy performed over the years. As we can see, our strategy performed poorly in 2005 but improved in 2006, falling again in 2008 and rising again.

It fell down dramatically from 2012 onwards up until 2014 and then it rose again through to 2018. Similar trends can be observed throughout the graph.

Overall, we can say it is very volatile and fluctuates regularly.

c) Binomial and the Weighted Binomial test

In [249]:
```
df = daily[daily.test != 0][["171_signal", "EURUSD"]].copy()
df['D'] = df["171_signal"]
```

In [250]:
```
df = df[:-6].copy()
df['s_current'] = daily[daily.index.isin(df.index)]["EURUSD"].value
df['s_future'] = daily[daily.index.isin(df.index+datetime.timedelta


# Get the realized exchange rate
df['R'] = np.where(df['s_future'] >= df['s_current'], 1, -1)
```

In [251]:
```
df
```

Out[251]:

	171_signal	EURUSD	D	s_current	s_future	R
2004-09-01	-1	1.219007	-1	1.219007	1.303900	1
2004-10-01	-1	1.241496	-1	1.241496	1.317193	1
2004-11-01	-1	1.274600	-1	1.274600	1.290906	1
2004-12-01	-1	1.334597	-1	1.334597	1.287598	-1
2005-01-01	-1	1.356502	-1	1.356502	1.219602	-1
...	...	...	...	...	...	...
2021-12-01	1	1.133029	1	1.133029	1.050420	-1
2022-01-01	1	1.132503	1	1.132503	1.073411	-1
2022-02-01	1	1.122965	1	1.122965	1.047768	-1
2022-03-01	1	1.121592	1	1.121592	1.020825	-1
2022-06-01	-1	1.073411	-1	1.073411	0.988631	-1

178 rows × 6 columns

In [252]:
```
df['W'] = (df['D']-np.mean(df['D']))*(df['R']-np.mean(df['R']))
T_B = np.mean(df['W'])
```

In [253]:
```
T_B
```

Out[253]:
```
-0.010604721626057245
```

In [254]:
```
dy = df['W'] - np.mean(df['W'])
gamma_0 = sum((dy)**2)/len(df)
gamma_1 = np.mean((dy*dy.shift(-1))[:len(df)-1])
LRV = gamma_0 + 2*(1-1/2)*gamma_1
```

In [255]:
```
from scipy.stats import norm
```

In [256]:
```
statistic = T_B/np.sqrt(LRV/df.shape[0])
print('Test statistic : ', statistic, ', 5 % critical value : ', ro
```

```
Test statistic :  -0.11621741365575423 , 5 % critical value :  1.6
4
```

In [257]:
```
df['W_2'] = df['D']*(df['s_future']-df['s_current'])
T_WB = np.mean(df['W_2'])
```

In [258]:
```
T_WB
```

Out[258]:
```
-0.0065787956955727514
```

In [259]:
```
dy_2 = df['W_2'] - T_WB
gamma_0 = sum((dy_2)**2)/len(df)
gamma_1 = np.mean((dy_2*dy_2.shift(-1))[:len(df)-1])
LRV_2 = gamma_0 + 2*(1-1/2)*gamma_1
```

In [260]:
```
statistic_2 = T_WB/np.sqrt(LRV_2/len(df))
print('Test statistic :' , statistic_2 , ', 5 % critical value : ',
```

```
Test statistic : -0.8403040190641669 , 5 % critical value :  1.64
```

We reject the null. If we reject the null hypothesis, it suggests that our directional forecasts successfully captured the realized appreciation or depreciation of exchange rates.

d)

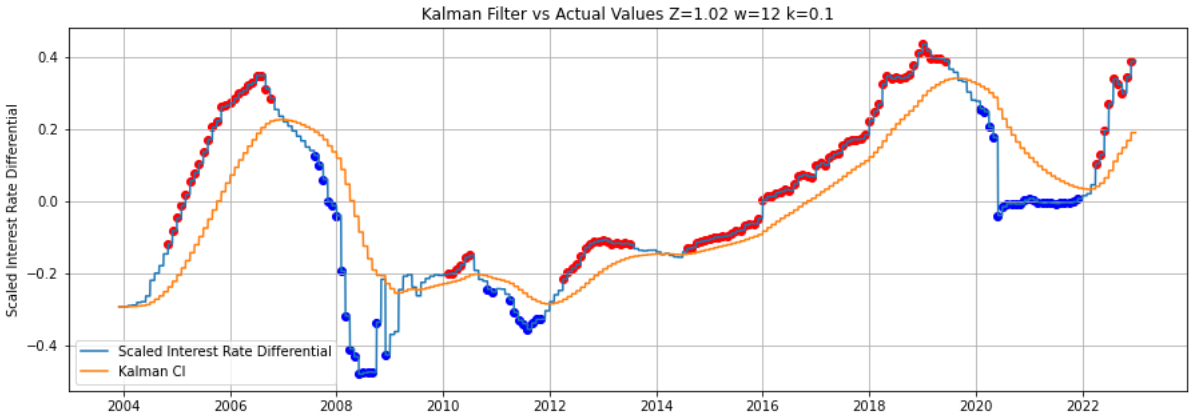
Different Value for k,w,z,t

1. k,w,z,t = (0.01,5,1.02,171)

In [262]:
```
# Version with CI
k = 0.1
w = 12
z = 1.02
```

In [263]:
```
# This implements the kalman filter in python
# It is simple otherwise to create using a for loop
interest["Filter"] = interest.ir_diff.ewm(alpha = k, adjust = False

# Compute the filter error
interest["Filter Error"] = interest.ir_diff - interest["Filter"]

# compute the rolling standard deviation
interest["std"] = interest["Filter Error"].rolling(w).std()

# create our confidence intervals or "boundaries of inaction"
# these are scaled by teh number of standard deviations "z"
interest["Upper"] = interest["Filter"] + z*interest["std"]
interest["Lower"] = interest["Filter"] - z*interest["std"]

# Create signal that evaluates whether we are outside the threshold
# then multiply by the direction of the mistake
# (we use economic theory to decide which direction is long or shor
interest["test"] = np.where(interest["Filter Error"].abs()>z*intere

drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
daily  = pd.DataFrame(index = drange)

# Integrate the monthly dta into the daily data
daily["test"] = interest["test"]

daily["Upper"] = interest["Upper"]
daily["Lower"] = interest["Lower"]
daily["Filter"] = interest["Filter"]
daily["ir_diff"] = interest["ir_diff"]

# Fill NA values with the last available value
daily["Upper"] = daily["Upper"].ffill()
daily["Lower"] = daily["Lower"].ffill()
daily["Filter"] = daily["Filter"].ffill()
daily["ir_diff"] = daily["ir_diff"].ffill()

# fill the remaining NA values with 0's
# also populates the test column
daily = daily.fillna(0)

i = 171
daily.loc[:, str(i)+"_signal"] = 0
for j in daily.index:
    if daily.loc[j, "test"] != 0:
        daily.loc[j:j+datetime.timedelta(i), str(i)+"_signal"] = da
```

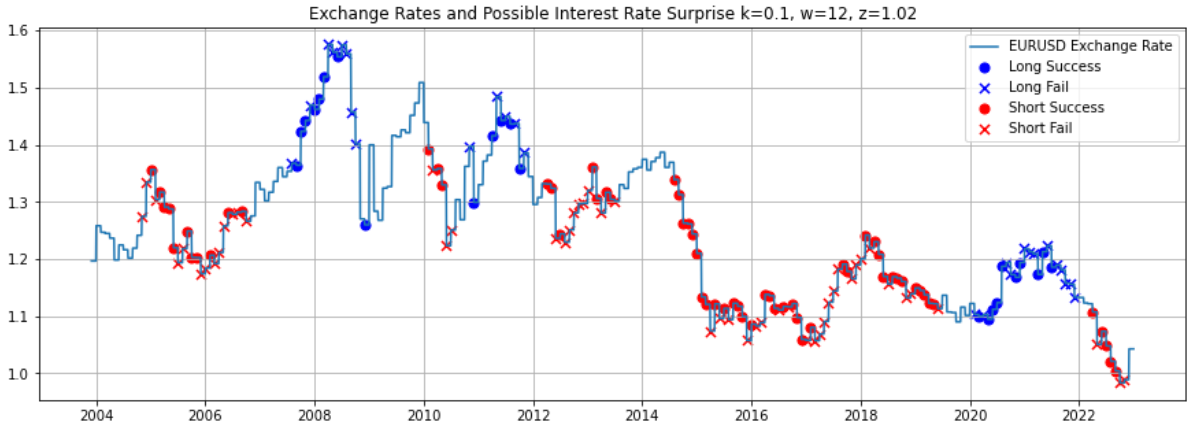
In [264]:
```
data2 = daily.dropna()
fig, ax = plt.subplots(figsize = (15, 5))
ax.set_title("Kalman Filter vs Actual Values " + "Z="+str(z) + " w=

ax.set_ylabel("Scaled Interest Rate Differential")

# Plot the actual series and the filter
ax.plot(data2["ir_diff"])
ax.plot(data2["Filter"])

# This code block is used to add confidence intervals when z > 0
#ax.fill_between(data2.index, data2.Lower, data2.Upper, color='b',

# add scatterplots using boolean indexing
# We change the colors and shapes based on the conditions
ax.scatter(data2[data2.test == 1].index, data2[data2.test == 1]["ir
ax.scatter(data2[data2.test == -1].index, data2[data2.test == -1]["
ax.legend(["Scaled Interest Rate Differential", "Kalman CI"])

# this code can let us zoom in on certain time periods
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
ax.grid()
```



In [265]:
```
drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
exdf = pd.DataFrame(index = drange)

exdf["EURUSD"] = interest["Euro"]
exdf["EURUSD"] = exdf["EURUSD"].ffill()

daily["EURUSD"] = exdf["EURUSD"]
daily["Returns"] = np.log(daily["EURUSD"]).diff()
```

In [266]:
```
i = 171
daily[str(i)+"_returns"] = np.exp((daily[str(i)+"_signal"].shift()*
s = 171
daily[str(s)+"_success"] = ((daily[daily.test!= 0][str(s)+"_returns
```

In [267]:
```
plt.figure(figsize = (15, 5))

plt.title("Exchange Rates and Possible Interest Rate Surprise"+ " k

plt.plot(daily["EURUSD"])

longsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test"
longfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"] =
shortsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test
shortfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"]

plt.scatter(longsuccess.index, longsuccess["EURUSD"], color = 'blue
plt.scatter(longfail.index, longfail["EURUSD"], color = 'blue', s =

plt.scatter(shortsuccess.index, shortsuccess["EURUSD"], color = 're
plt.scatter(shortfail.index, shortfail["EURUSD"], color = 'red', s

plt.legend(["EURUSD Exchange Rate", "Long Success", "Long Fail", "S
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
plt.grid()
```



In [268]:
```
df = daily[daily.test != 0][["171_signal", "EURUSD"]].copy()
df['D'] = df["171_signal"]
```

In [269]:
```
df = df[:-6].copy()
df['s_current'] = daily[daily.index.isin(df.index)]["EURUSD"].value
df['s_future'] = daily[daily.index.isin(df.index+datetime.timedelta


# Get the realized exchange rate
df['R'] = np.where(df['s_future'] >= df['s_current'], 1, -1)
```

In [270]:
```
df['W'] = (df['D']-np.mean(df['D']))*(df['R']-np.mean(df['R']))
T_B = np.mean(df['W'])
```

In [271]:
```
T_B
```

Out[271]:
```
0.06442452026451377
```

In [272]:
```
dy = df['W'] - np.mean(df['W'])
gamma_0 = sum((dy)**2)/len(df)
gamma_1 = np.mean((dy*dy.shift(-1))[:len(df)-1])
LRV = gamma_0 + 2*(1-1/2)*gamma_1
```

In [273]:
```
from scipy.stats import norm
```

In [274]:
```
statistic = T_B/np.sqrt(LRV/df.shape[0])
print('Test statistic : ', statistic, ', 5 % critical value : ', ro
```

```
Test statistic :  0.6818611496537675 , 5 % critical value :  1.64
```

In [275]:
```
df['W_2'] = df['D']*(df['s_future']-df['s_current'])
T_WB = np.mean(df['W_2'])
```

In [276]:
```
T_WB
```

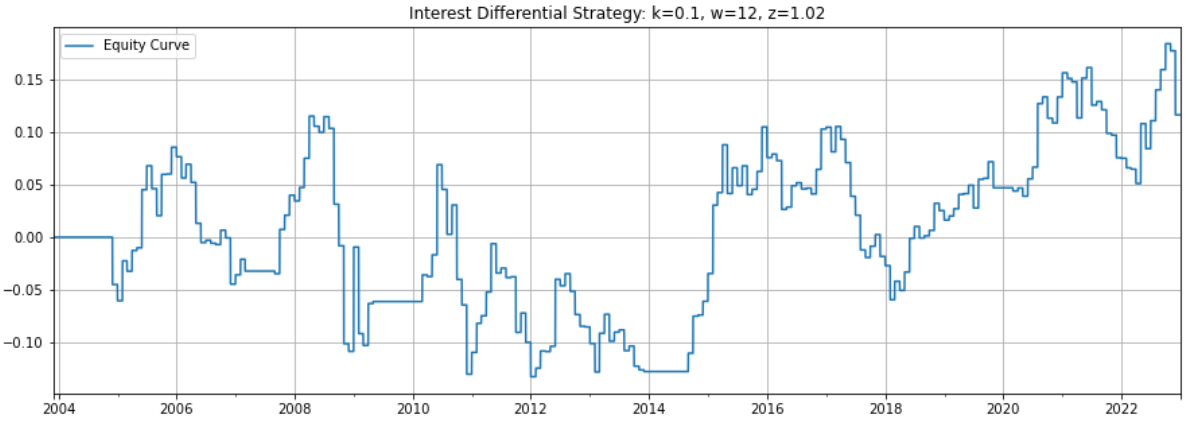
Out[276]:
```
-0.000542351394701913
```

In [277]:
```
dy_2 = df['W_2'] - T_WB
gamma_0 = sum((dy_2)**2)/len(df)
gamma_1 = np.mean((dy_2*dy_2.shift(-1))[:len(df)-1])
LRV_2 = gamma_0 + 2*(1-1/2)*gamma_1
```

In [278]:
```
statistic_2 = T_WB/np.sqrt(LRV_2/len(df))
print('Test statistic :' , statistic_2 , ', 5 % critical value : ',
```

```
Test statistic : -0.06393012218768479 , 5 % critical value :  1.64
```

In [279]:
```
plt.figure(figsize = (15, 5))
d2 = daily[daily.index.year >= 2020]
(daily["171_returns"].dropna()).plot()

plt.title("Interest Differential Strategy:"+ " k=" + str(k)+", w="+
plt.legend(["Equity Curve", "Long Success", "Long Fail", "Short Suc

plt.grid()
#plt.xlim(["1/1/2019", "1/1/2022"])
```



In this strategy, we see that the it performs well in the year 2008 and then falls by a lot right after.

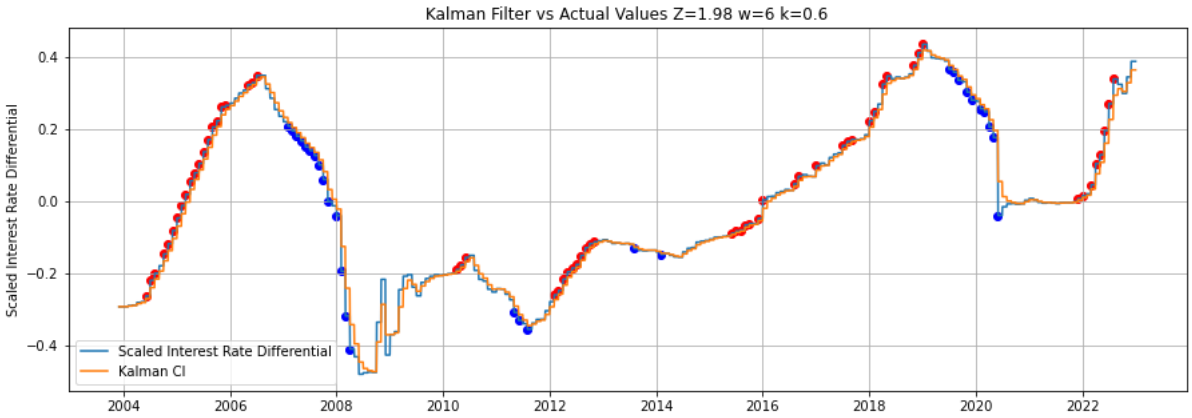
It follows an increasing trend from the year 2014 onwards with occasional downward trends around 2018 and then follown an increasing general trend from then on up till 2022.

2. (k,w,z,t) = (0.6,6,1.08,171)

In [280]:
```
k = 0.6
w = 6
z = 1.98
```

In [281]:
```
interest["Filter"] = interest.ir_diff.ewm(alpha = k, adjust = False

# Compute the filter error
interest["Filter Error"] = interest.ir_diff - interest["Filter"]

# compute the rolling standard deviation
interest["std"] = interest["Filter Error"].rolling(w).std()

# create our confidence intervals or "boundaries of inaction"
# these are scaled by teh number of standard deviations "z"
interest["Upper"] = interest["Filter"] + z*interest["std"]
interest["Lower"] = interest["Filter"] - z*interest["std"]

# Create signal that evaluates whether we are outside the threshold
# then multiply by the direction of the mistake
# (we use economic theory to decide which direction is long or shor
interest["test"] = np.where(interest["Filter Error"].abs()>z*intere

drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
daily  = pd.DataFrame(index = drange)

# Integrate the monthly dta into the daily data
daily["test"] = interest["test"]

daily["Upper"] = interest["Upper"]
daily["Lower"] = interest["Lower"]
daily["Filter"] = interest["Filter"]
daily["ir_diff"] = interest["ir_diff"]

# Fill NA values with the last available value
daily["Upper"] = daily["Upper"].ffill()
daily["Lower"] = daily["Lower"].ffill()
daily["Filter"] = daily["Filter"].ffill()
daily["ir_diff"] = daily["ir_diff"].ffill()

# fill the remaining NA values with 0's
# also populates the test column
daily = daily.fillna(0)

i = 171
daily.loc[:, str(i)+"_signal"] = 0
for j in daily.index:
    if daily.loc[j, "test"] != 0:
        daily.loc[j:j+datetime.timedelta(i), str(i)+"_signal"] = da
```

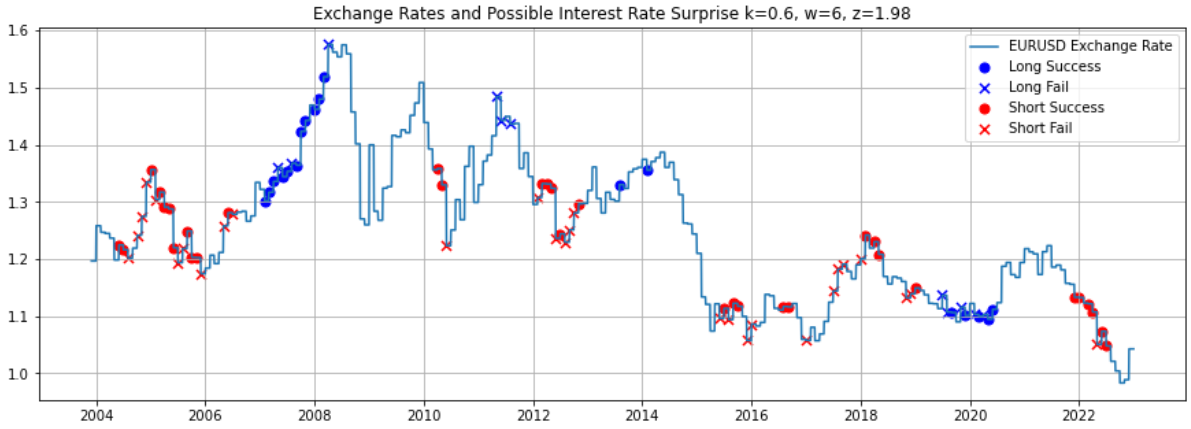
In [282]:
```
data2 = daily.dropna()
fig, ax = plt.subplots(figsize = (15, 5))
ax.set_title("Kalman Filter vs Actual Values " + "Z="+str(z) + " w=

ax.set_ylabel("Scaled Interest Rate Differential")

# Plot the actual series and the filter
ax.plot(data2["ir_diff"])
ax.plot(data2["Filter"])

# This code block is used to add confidence intervals when z > 0
#ax.fill_between(data2.index, data2.Lower, data2.Upper, color='b',

# add scatterplots using boolean indexing
# We change the colors and shapes based on the conditions
ax.scatter(data2[data2.test == 1].index, data2[data2.test == 1]["ir
ax.scatter(data2[data2.test == -1].index, data2[data2.test == -1]["
ax.legend(["Scaled Interest Rate Differential", "Kalman CI"])

# this code can let us zoom in on certain time periods
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
ax.grid()
```



In [283]:
```
drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
exdf = pd.DataFrame(index = drange)

exdf["EURUSD"] = interest["Euro"]
exdf["EURUSD"] = exdf["EURUSD"].ffill()

daily["EURUSD"] = exdf["EURUSD"]
daily["Returns"] = np.log(daily["EURUSD"]).diff()
```

In [284]:
```
i = 171
daily[str(i)+"_returns"] = np.exp((daily[str(i)+"_signal"].shift()*
s = 171
daily[str(s)+"_success"] = ((daily[daily.test!= 0][str(s)+"_returns
```

In [285]:
```
plt.figure(figsize = (15, 5))

plt.title("Exchange Rates and Possible Interest Rate Surprise"+ " k

plt.plot(daily["EURUSD"])

longsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test"
longfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"] =
shortsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test
shortfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"]

plt.scatter(longsuccess.index, longsuccess["EURUSD"], color = 'blue
plt.scatter(longfail.index, longfail["EURUSD"], color = 'blue', s =

plt.scatter(shortsuccess.index, shortsuccess["EURUSD"], color = 're
plt.scatter(shortfail.index, shortfail["EURUSD"], color = 'red', s

plt.legend(["EURUSD Exchange Rate", "Long Success", "Long Fail", "S
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
plt.grid()
```



In [286]:
```
df = daily[daily.test != 0][["171_signal", "EURUSD"]].copy()
df['D'] = df["171_signal"]
```

In [287]:
```
df = df[:-6].copy()
df['s_current'] = daily[daily.index.isin(df.index)]["EURUSD"].value
df['s_future'] = daily[daily.index.isin(df.index+datetime.timedelta


# Get the realized exchange rate
df['R'] = np.where(df['s_future'] >= df['s_current'], 1, -1)
```

In [288]:
```
df['W'] = (df['D']-np.mean(df['D']))*(df['R']-np.mean(df['R']))
T_B = np.mean(df['W'])
```

In [289]:
```
T_B
```

Out[289]:
```
0.28179930795847746
```

In [290]:
```
dy = df['W'] - np.mean(df['W'])
gamma_0 = sum((dy)**2)/len(df)
gamma_1 = np.mean((dy*dy.shift(-1))[:len(df)-1])
LRV = gamma_0 + 2*(1-1/2)*gamma_1
```

In [291]:
```
statistic = T_B/np.sqrt(LRV/df.shape[0])
print('Test statistic : ', statistic, ', 5 % critical value : ', ro
```

```
Test statistic :  2.3825350615086167 , 5 % critical value :  1.64
```

In [292]:
```
df['W_2'] = df['D']*(df['s_future']-df['s_current'])
T_WB = np.mean(df['W_2'])
```

In [293]:
```
T_WB
```

Out[293]:
```
0.015022094109479129
```

In [294]:
```
dy_2 = df['W_2'] - T_WB
gamma_0 = sum((dy_2)**2)/len(df)
gamma_1 = np.mean((dy_2*dy_2.shift(-1))[:len(df)-1])
LRV_2 = gamma_0 + 2*(1-1/2)*gamma_1
```

In [295]:
```
statistic_2 = T_WB/np.sqrt(LRV_2/len(df))
print('Test statistic :' , statistic_2 , ', 5 % critical value : ',
```

```
Test statistic : 1.5548769680714378 , 5 % critical value :  1.64
```

In [296]:
```
plt.figure(figsize = (15, 5))
d2 = daily[daily.index.year >= 2020]
(daily["171_returns"].dropna()).plot()

plt.title("Interest Differential Strategy:"+ " k=" + str(k)+", w="+
plt.legend(["Equity Curve", "Long Success", "Long Fail", "Short Suc

plt.grid()
#plt.xlim(["1/1/2019", "1/1/2022"])
```

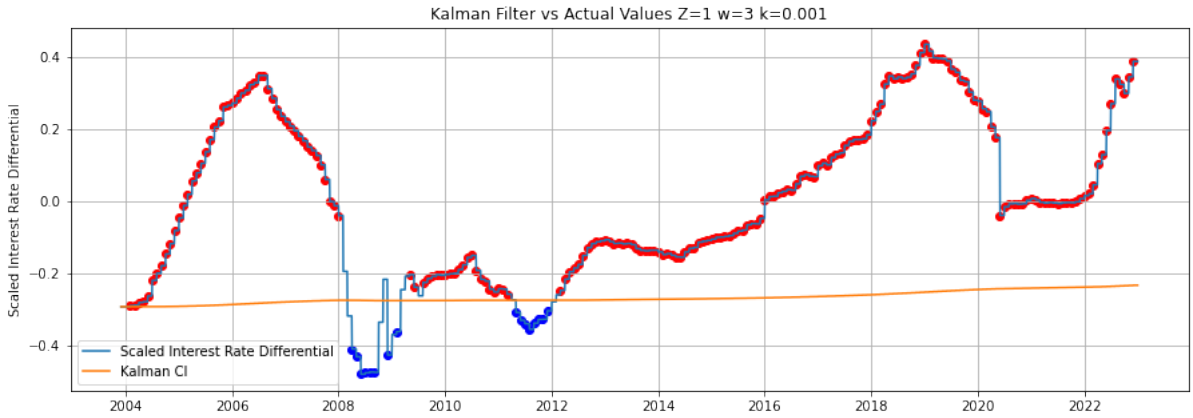
In this strategy, our strategy performs extremely well around 2011 but the suffers a strong fall after that going up till 2012 and then continues to be low until 2022.

3. (k,w,z,t) = (0.001,3,1,171)

In [297]:
```
k = 0.001
w = 3
z = 1
```

In [298]:
```
interest["Filter"] = interest.ir_diff.ewm(alpha = k, adjust = False

# Compute the filter error
interest["Filter Error"] = interest.ir_diff - interest["Filter"]

# compute the rolling standard deviation
interest["std"] = interest["Filter Error"].rolling(w).std()

# create our confidence intervals or "boundaries of inaction"
# these are scaled by teh number of standard deviations "z"
interest["Upper"] = interest["Filter"] + z*interest["std"]
interest["Lower"] = interest["Filter"] - z*interest["std"]

# Create signal that evaluates whether we are outside the threshold
# then multiply by the direction of the mistake
# (we use economic theory to decide which direction is long or shor
interest["test"] = np.where(interest["Filter Error"].abs()>z*intere

drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
daily  = pd.DataFrame(index = drange)

# Integrate the monthly dta into the daily data
daily["test"] = interest["test"]

daily["Upper"] = interest["Upper"]
daily["Lower"] = interest["Lower"]
daily["Filter"] = interest["Filter"]
daily["ir_diff"] = interest["ir_diff"]

# Fill NA values with the last available value
daily["Upper"] = daily["Upper"].ffill()
daily["Lower"] = daily["Lower"].ffill()
daily["Filter"] = daily["Filter"].ffill()
daily["ir_diff"] = daily["ir_diff"].ffill()

# fill the remaining NA values with 0's
# also populates the test column
daily = daily.fillna(0)

i = 171
daily.loc[:, str(i)+"_signal"] = 0
for j in daily.index:
    if daily.loc[j, "test"] != 0:
        daily.loc[j:j+datetime.timedelta(i), str(i)+"_signal"] = da
```

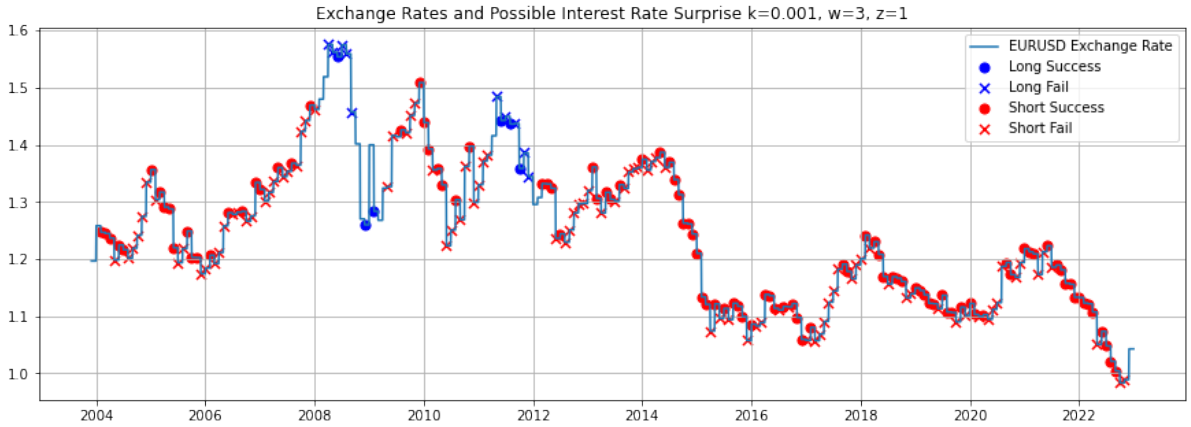
In [299]:
```
data2 = daily.dropna()
fig, ax = plt.subplots(figsize = (15, 5))
ax.set_title("Kalman Filter vs Actual Values " + "Z="+str(z) + " w=

ax.set_ylabel("Scaled Interest Rate Differential")

# Plot the actual series and the filter
ax.plot(data2["ir_diff"])
ax.plot(data2["Filter"])

# This code block is used to add confidence intervals when z > 0
#ax.fill_between(data2.index, data2.Lower, data2.Upper, color='b',

# add scatterplots using boolean indexing
# We change the colors and shapes based on the conditions
ax.scatter(data2[data2.test == 1].index, data2[data2.test == 1]["ir
ax.scatter(data2[data2.test == -1].index, data2[data2.test == -1]["
ax.legend(["Scaled Interest Rate Differential", "Kalman CI"])

# this code can let us zoom in on certain time periods
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
ax.grid()
```



In [300]:
```
drange = pd.date_range(start =interest.index[0], end = "01/01/2023"
exdf = pd.DataFrame(index = drange)

exdf["EURUSD"] = interest["Euro"]
exdf["EURUSD"] = exdf["EURUSD"].ffill()

daily["EURUSD"] = exdf["EURUSD"]
daily["Returns"] = np.log(daily["EURUSD"]).diff()
```

In [301]:
```
i = 171
daily[str(i)+"_returns"] = np.exp((daily[str(i)+"_signal"].shift()*
s = 171
daily[str(s)+"_success"] = ((daily[daily.test!= 0][str(s)+"_returns
```

In [302]:
```
plt.figure(figsize = (15, 5))

plt.title("Exchange Rates and Possible Interest Rate Surprise"+ " k

plt.plot(daily["EURUSD"])

longsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test"
longfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"] =
shortsuccess = daily[(daily[str(s)+"_success"] == 1) & (daily["test
shortfail = daily[(daily[str(s)+"_success"] == 0) & (daily["test"]

plt.scatter(longsuccess.index, longsuccess["EURUSD"], color = 'blue
plt.scatter(longfail.index, longfail["EURUSD"], color = 'blue', s =

plt.scatter(shortsuccess.index, shortsuccess["EURUSD"], color = 're
plt.scatter(shortfail.index, shortfail["EURUSD"], color = 'red', s

plt.legend(["EURUSD Exchange Rate", "Long Success", "Long Fail", "S
#plt.xlim([datetime.date(2022, 1, 1), datetime.date(2023, 1, 1)])
plt.grid()
```



In [303]:
```
df = daily[daily.test != 0][["171_signal", "EURUSD"]].copy()
df['D'] = df["171_signal"]
```

In [304]:
```
df = df[:-6].copy()
df['s_current'] = daily[daily.index.isin(df.index)]["EURUSD"].value
df['s_future'] = daily[daily.index.isin(df.index+datetime.timedelta


# Get the realized exchange rate
df['R'] = np.where(df['s_future'] >= df['s_current'], 1, -1)
```

In [305]:
```
df['W'] = (df['D']-np.mean(df['D']))*(df['R']-np.mean(df['R']))
T_B = np.mean(df['W'])
```

In [306]:
```
T_B
```

Out[306]: -0.109931972789115

In [307]:
```
dy = df['W'] - np.mean(df['W'])
gamma_0 = sum((dy)**2)/len(df)
gamma_1 = np.mean((dy*dy.shift(-1))[:len(df)-1])
LRV = gamma_0 + 2*(1-1/2)*gamma_1
```

In [308]:
```
statistic = T_B/np.sqrt(LRV/df.shape[0])
print('Test statistic : ', statistic, ', 5 % critical value : ', ro
```

```
Test statistic :  -2.385209315081952 , 5 % critical value :  1.64
```

In [309]:
```
df['W_2'] = df['D']*(df['s_future']-df['s_current'])
T_WB = np.mean(df['W_2'])
```

In [310]:
```
T_WB
```

Out[310]:
```
-0.009143783364977155
```

In [311]:
```
dy_2 = df['W_2'] - T_WB
gamma_0 = sum((dy_2)**2)/len(df)
gamma_1 = np.mean((dy_2*dy_2.shift(-1))[:len(df)-1])
LRV_2 = gamma_0 + 2*(1-1/2)*gamma_1
```

In [312]:
```
statistic_2 = T_WB/np.sqrt(LRV_2/len(df))
print('Test statistic :' , statistic_2 , ', 5 % critical value : ',
```

```
Test statistic : -1.244212134981302 , 5 % critical value :  1.64
```

In [313]:
```
plt.figure(figsize = (15, 5))
d2 = daily[daily.index.year >= 2020]
(daily["171_returns"].dropna()).plot()

plt.title("Interest Differential Strategy:"+ " k=" + str(k)+", w="+
plt.legend(["Equity Curve", "Long Success", "Long Fail", "Short Suc

plt.grid()
#plt.xlim(["1/1/2019", "1/1/2022"])
```

Here, we can see that it follows a conitnous falling trend from the year 2004 till 2012. Although it started at a relativelty higher value in 2022, it falls till 2012. After that, it increases but only at a marginal rate.

Question 3

a)

We first type the formula for Price of the Bond:

Price of Bond (P):

$$\sum_{t=1}^{T} \left(\frac{C}{(1+y)^t} + \frac{F}{(1+y)^t} \right)$$

Here: C: Coupon Payment F: Face Value T: Time period y: Yield to Maturity

In [314]:
```
C = 0.05
F = 1000
P = 984.96
n = 10
```

Calculating the YTM using this formula gives us 0.05037884894405933 or 5.03%

b)

We now calculate if there was a 20% chance that it defaults and 80% chance that it pays the full face value using the following formula

$$0 \cdot 0.2 + 0.8 \cdot \frac{1000}{1+r}$$

Here: r = risk-free rate i.e. 1%

We get the price today as $792.0792

Question 4

a)

Calculating for the 7-year bond

In [316]:
```
n2 = 7
cr2 = 0.03
r2 = 0.03
face_value = 1000
```

In [317]:
```
C2 = cr2*face_value
```

In [318]:
```
num_summation2 = 0
for i in range (1,n2):
    num_summation2 += (i*C2/((1+r2)**i))
```

In [319]:
```
num2 = num_summation2 + (n2*(face_value + C2)/((1+r2)**n2))
```

In [320]:
```
denom_summation2 = 0
for i in range (1,n2+1):
    denom_summation2 += (C2/((1+r2)**i))
```

In [321]:
```
denom2 = denom_summation2 + face_value/((1+r2)**n2)
```

In [322]:
```
dur_2 = num2/denom2
dur_2
```

Out[322]:
```
6.417191443878187
```

Calculating for the 10-year bond

In [323]:
```
n = 10
cr = 0.15
r = 0.1
face_value = 1000
```

In [324]:
```
C = cr*face_value
C
```

Out[324]:
```
150.0
```

In [325]:
```
num_summation = 0
for i in range (1,n):
    num_summation += (i*C/((1+r)**i))
num = num_summation + (n*(face_value + C)/((1+r)**n))

denom_summation = 0
for i in range (1,n+1):
    denom_summation += (C/((1+r)**i))

denom = denom_summation + face_value/((1+r)**n)
```

In [326]:
```
dur = num/denom
dur
```

Out[326]:
```
6.281090250274969
```

b)

Duration can measure how long it takes, in years, for an investor to be repaid a bond's price by the bond's total cash flows.

If the interest rate increases by 1%, the asset with the higher duration will be more volatile to changes. In our case, we see that the 7-year bond has a slightly higher duration, hence we can expect it to undergo a higher change in value with a rise in interest rates.

c)

Since we use 30% of the 7-year bond and 70% of the 10-year bond, we can simply calculate the duration of the portfolio as:

0.3 x 6.41 + 0.7 x 6.28

Which we get as 6.319 or 6.31% approximately

Question 5

a)

FOMC Report

Difference between the FOMC statement for December 2022 and Febaruary 2023:

The first release is the statement released by the Federal Open Market Committee (FOMC) outlining the monetary policy decisions and outlook for the economy. It highlights the reasons for the Committee's decision to raise the target range for the federal funds rate, their goals for maximum employment and inflation, and their plans for reducing the size of the Federal Reserve's balance sheet.

The second release is a decision regarding the implementation of the monetary policy announced in the first release. It provides information on specific actions taken by the Federal Reserve to carry out the monetary policy, such as adjusting the interest rate paid on reserve balances, conducting open market operations, and reinvesting principal payments from their holdings of Treasury securities and agency debt and mortgage-backed securities. The second release also details the adjustments made to the primary credit rate.

Further, accoring to the notice, the Fed observes that inflation is higher in December which suggests that there are supply and demand imbalances with higher prices observed in Food and Energy. All in all, there are broader price pressures. The inflation, although inflated, is still lower in February 2023 as compared to December 2022.

In short, the first release outlines the decisions made by the FOMC and the outlook for the economy, while the second release details the steps taken by the Federal Reserve to implement the monetary policy decisions.

European Central Bank: Monetary Report

Dec 2022: As for the release in December, they do plan to raise the interest rate by 50 basis points, but there is no notice of raising interest rate at a particular time in the future. However, they do say that they will cotninue to raise the interest rate in light of rising inflation.

As for to the asset purchase programme (APP) portpolio, the notice said that it will decline at a measured and predictable pace, as the Eurosystem will not reinvest all of the principal payments from maturing securities. This is a difference between the two reports as here, the central bank says that it WILL NOT re-invest all of the principay payments in order to squeeze liquidity and help curb inflation.

The Governing Council decided on the modalities for reducing the Eurosystem's holdings of securities under the APP The APP portfolio will decline by €15 billion per month until the end of June 2023 and the subsequent pace will be determined over time Partial reinvestments will be conducted broadly in line with current practice and will be allocated proportionally The remaining reinvestments for corporate bonds will be tilted towards issuers with a better climate performance, supporting the gradual decarbonisation of the Eurosystem's holdings.

To summarise:

1) The Governing Council decided to raise the three key ECB interest rates by 50 basis points 2) Expects to raise the rates further due to the upward revision of the inflation outlook 3) Interest rates will have to rise significantly to reach restrictive levels to ensure a timely return of inflation to the 2% target 4) Eurosystem will NOT REINVEST all of the principal payments from maturing securities.

Feb 2023: European Central Bank has raised the interest rates as of February 2023 but it also expects the aise the interest rate in the monetary policy meeting in March 2023 and will then evaluate further plan of action.

As compared to December, The European Central Bank will conitnue to practise what it decided in December 2022. In particular, the remaining reinvestment amounts WILL BE allocated proportionally to the share of redemptions across each constituent programme of the APP and, under the public sector purchase programme (PSPP), to the share of redemptions of each jurisdiction and across national and supranational issuers. Here, a major difference as compared to December is that the central has decided to REINVESTED amount in certain sectors. The difference however, will be for the Eurosystem’s corporate bond purchases, the remaining reinvestments of which will be tilted more strongly towards issuers with a better climate performance.

To summarise:

1. The Governing Council decided on the modalities for reducing the Eurosystem's holdings of securities under the APP
2. The APP portfolio will decline by €15 billion per month until the end of June 2023 and the subsequent pace will be determined over time
3. Partial reinvestments will be conducted broadly in line with current practice and will be allocated proportionally
4. The remaining reinvestments for corporate bonds will be tilted towards issuers with a better climate performance, supporting the gradual decarbonisation of the Eurosystem's holdings.

b)

Interest rate differentials can be impacted by a variety of factors, both expected and unexpected. Some of the shocks or surprises that could affect interest rate differentials include:

1. Unexpected changes in monetary policy: Central banks may adjust interest rates in response to changes in the economy, and these changes can have a significant impact on interest rate differentials.
2. Political events: Political events, such as elections or policy changes, can impact the direction of interest rates and interest rate differentials.
3. Natural disasters: Natural disasters can disrupt economic activity, leading to changes in interest rates and interest rate differentials.
4. Market volatility: Unexpected market volatility can lead to changes in interest rates and interest rate differentials.
5. Changes in inflation expectations: Changes in inflation expectations can impact interest rate differentials, as investors adjust their expectations for future inflation and interest rate changes.
6. Global economic developments: Developments in the global economy, such as changes in growth rates or economic shocks in other countries, can impact interest rate differentials.

The shock or surpise to the interest rate differential was the the war between Russia and Ukrain, as is captured by bullet point number 6. This has caused inflationary pressures on many countries and especially in the European Union. Further, in the report, the European Central bank although increases the interest rate by 50 basis points in February, it says that it will further increase it in March, therefore signifying that it believs the inflation to remain high. This can affect the expectations of the market and hence can come as a shock or surpirse. This has also been captured by the FOMC's febraury statement which talks about its attentiveness to inflation due to the highly uncertain situations in Europe.

c)

Interest rate increases can affect a currency's exchange rate in a variety of ways. The increase in interest rates in Europe could lead to a strengthening of the euro, as higher interest rates tend to attract more foreign investment and increase demand for a currency. However, in this case, the situation is complicated by the fact that there is a war going on in Europe. Wars can lead to increased uncertainty and instability, which can result in a decrease in foreign investment and a depreciation of a currency.

Further, I also feel that since the European Central Bank will increase the interest rate in March as well as in February, it might affect market expectations and hence can result in the depreciation of the Euro. This has also been writtena about in the Wall Street Journal article which talks about how the investors and the market believes the inflation to not hold as high as the central bank believes it to be. Hence, as the expectations of the market are that the inflation will continue to remain high and the market will be volatile. This might also have an impact on the exchange rate.

In the case of the United States, the increase in interest rates could also have an impact on the value of the euro, as higher interest rates in the US tend to attract more foreign investment away from Europe and towards the US, which can lead to a strengthening of the US dollar and a depreciation of the euro.

Ultimately, the impact of the interest rate changes on the euro's exchange rate will depend on a variety of factors, including the state of the global economy, the political climate, and investor sentiment.

Question 6

Some suggestions other than increasing interest rates that i would give to the Fed are as follows:

1. Expansionary monetary policy: The Fed could pursue an expansionary monetary policy by increasing the money supply and lowering interest rates. This could help to stimulate spending and boost economic growth, which could in turn help to ease inflationary pressures.
2. Increased transparency and strong commincation to the public: The Fed could improve its communication with the public and financial markets by being more transparent about its policy intentions and decisions. This could help to build confidence and reduce uncertainty, which could be beneficial for the economy and inflation. Clear and transparent communication can help to anchor inflation expectations and build confidence in the central bank's ability to manage inflation. The Fed could provide regular updates on its assessment of inflation and its policy intentions to help reduce uncertainty.
3. If interest rates are not sufficient to control inflation, the Fed could consider using alternative monetary policy tools, such as large-scale asset purchases or direct lending to specific sectors, to help stimulate spending and promote economic growth.

In [ ]: