Auteurs: Thomas Vaudescal : 11237578

Mathieu Verville :11320263

Benjamin Viau : 11241571

Partie 1. Analyse préliminaire (20 points)

Définissons la série $y_t = \log(100 \times USDCAD_t)$ pour le reste de ce travail.

1.a) Chargez les données et ne gardez que les observations jusqu'à décembre 2019. Tracez la série brute avec des étiquettes sur chaque axe et un titre 'log(100 USDCAD=x)'. La dimension temporelle doit apparaître sur l'axe des abscisses.

In [ ]:
```
#Import the packages necessary and set figure size for the project
from arch import arch_model
import matplotlib.pyplot as plt
import matlab.engine
import numpy as np
import pandas as pd
import platform
import scipy as sp
import statsmodels.api as sm
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.arima.model import ARIMAResultsWrapper
import sys
# Deactivate warnings
import warnings
warnings.filterwarnings('ignore')
# Set figure size for all plots
plt.rcParams["figure.figsize"] = (15, 6)
```

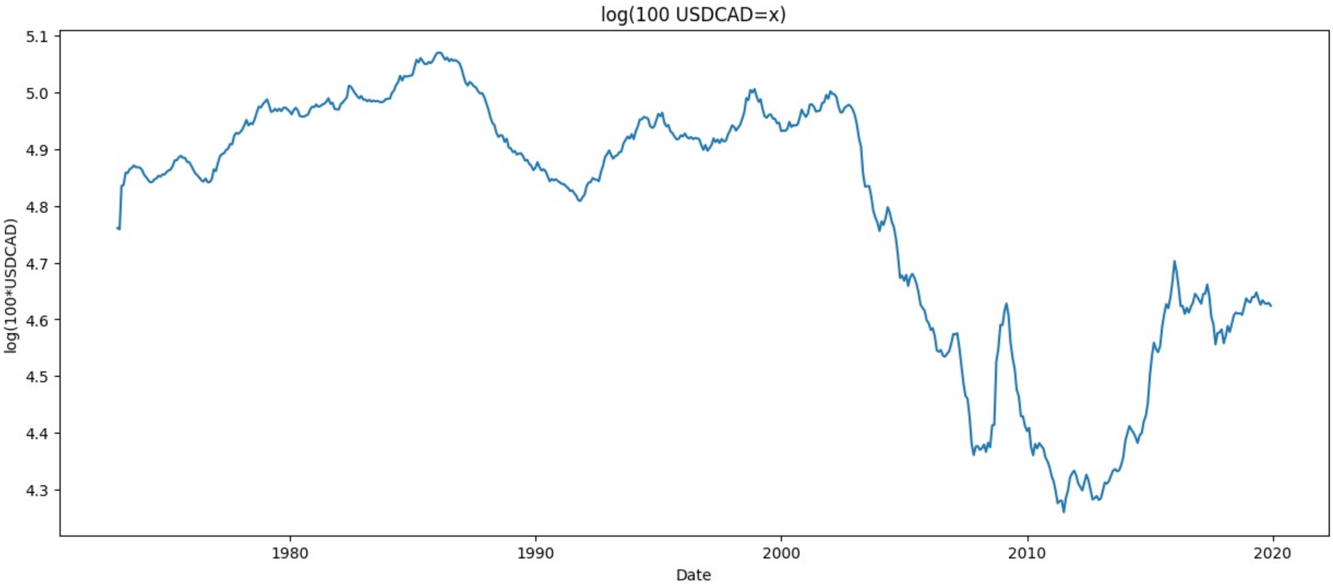
In [ ]:
```
if sys.version_info[0] != 3 or sys.version_info[1] < 7 or sys.version_info[1] > 11:
    print(f"Your version of Python may not be compatible with the matlab engine that is used in question 4.")
    print(f"Your version is: {platform.python_version()}")
    print(f"You may consult",'{https://www.mathworks.com/support/requirements/python-compatibility.html}',"to s
    print(f"Note: We ran it on 3.10.10")
else:
    print(f"Your python version is suitable to run matlabengine")
```

```
Your python version is suitable to run matlabengine
```

In [ ]:
```
#Import the data
df = pd.read_csv("data_W2023.csv")

#Clean data and format the dataframe
df.columns = ["date", "USDCAD"]
df = df[df["date"] < "2020-01-01"]
df["date"] = pd.to_datetime(df["date"])

#Transform the raw data into our base variable
df["log_100_USDCAD"] = np.log(100 * df["USDCAD"])
```

In [ ]:
```
#Generate the plot
plt.plot(df["date"], df["log_100_USDCAD"])
plt.xlabel("Date")
plt.ylabel(f"log(100*USDCAD)")
plt.title("log(100 USDCAD=x)")
```

Out[ ]:
```
Text(0.5, 1.0, 'log(100 USDCAD=x)')
```



Interpretation

Le graphique démontre l'évolution du taux de change USDCAD transformé (log(100*USDCAD)).

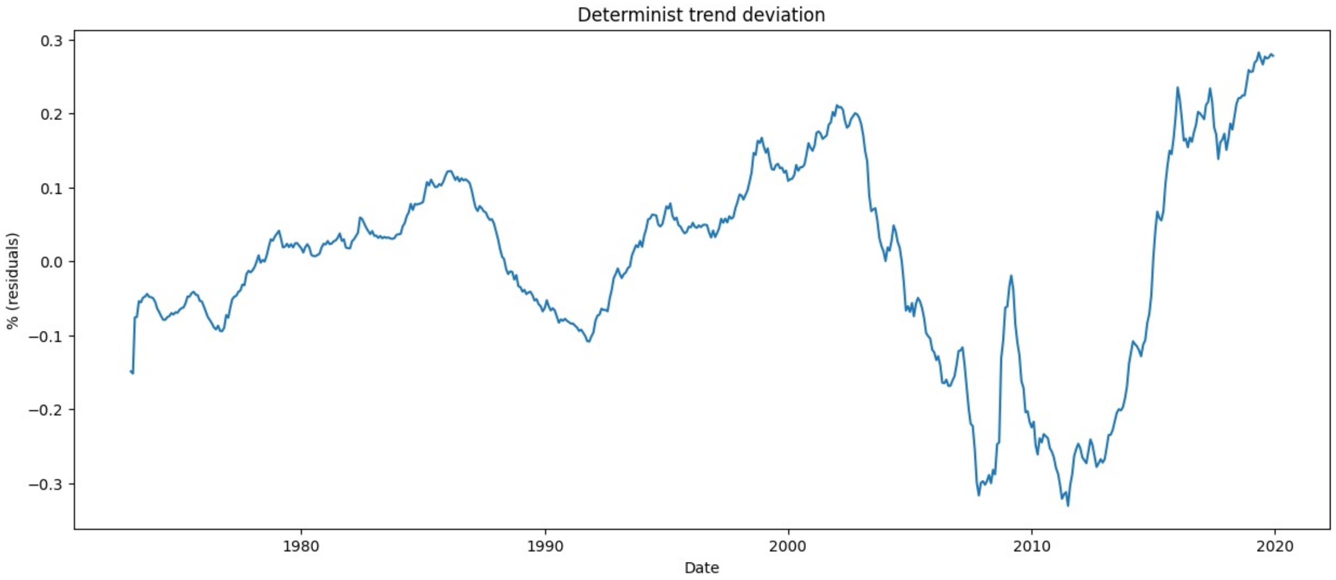
1.b) Supposons qu'il existe une tendance temporelle déterministe (polynôme d'ordre 2) : $\tau_t = \alpha + \beta t + \gamma t^2$.

Appliquez la transformation requise pour stationnariser la série logarithmique (le logarithme du taux de change) et tracez la série résultante.

In [ ]:
```python
Y = df["log_100_USDCAD"].values

T = len(Y)
trend = np.arange(1, T + 1)
ones = np.ones(T)
X = np.column_stack((ones, trend, trend ** 2))

B = np.linalg.inv(X.T @ X) @ X.T @ Y

df["res"] = Y - X @ B
```

In [ ]:
```python
plt.plot(df["date"], df["res"])
plt.xlabel("Date")
plt.ylabel(f"% (residuals)")
plt.title("Determinist trend deviation")
```

Out[ ]:
```
Text(0.5, 1.0, 'Determinist trend deviation')
```



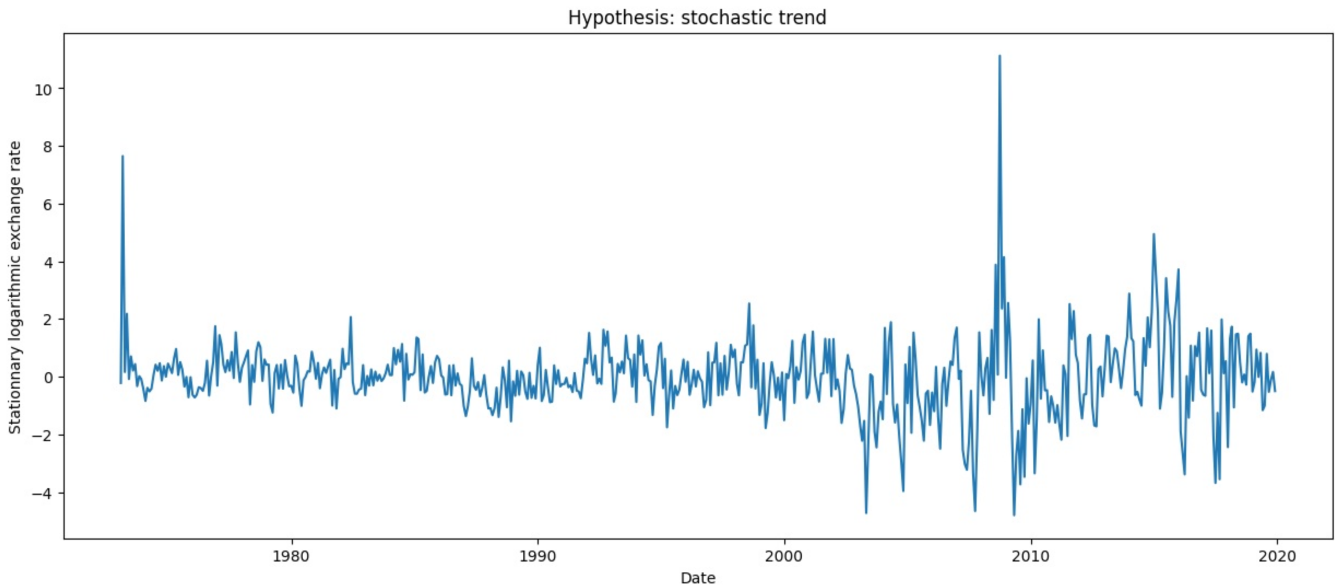
Interpretation

Ce graphique démontre la serie logarithmique apres avoir retirer une supposee tendance temporelle deterministe. A premiere vue, le graphique semble toujours exhiber des tendances et n'apparait pas comme ce a quoi on s'attendrait d'une serie stationnaire.

1.c) Supposons qu'il existe une tendance temporelle stochastique, c'est-à-dire une marche aléatoire.
Appliquez la transformation requise pour stationnariser la série logarithmique (le logarithme du taux d'echange) et tracez la série résultante.

In [ ]:
```
df_stoch = df.copy() # Copy to make first part robust if we run the project out of order
df_stoch["stoch_USDCAD"] = df_stoch["log_100_USDCAD"].diff() * 100
df_stoch.dropna(inplace=True) # Drop first row

plt.plot(df_stoch["date"], df_stoch["stoch_USDCAD"])
plt.xlabel("Date")
plt.ylabel("Stationnary logarithmic exchange rate")
plt.title("Hypothesis: stochastic trend")
```

Out[ ]:
```
Text(0.5, 1.0, 'Hypothesis: stochastic trend')
```

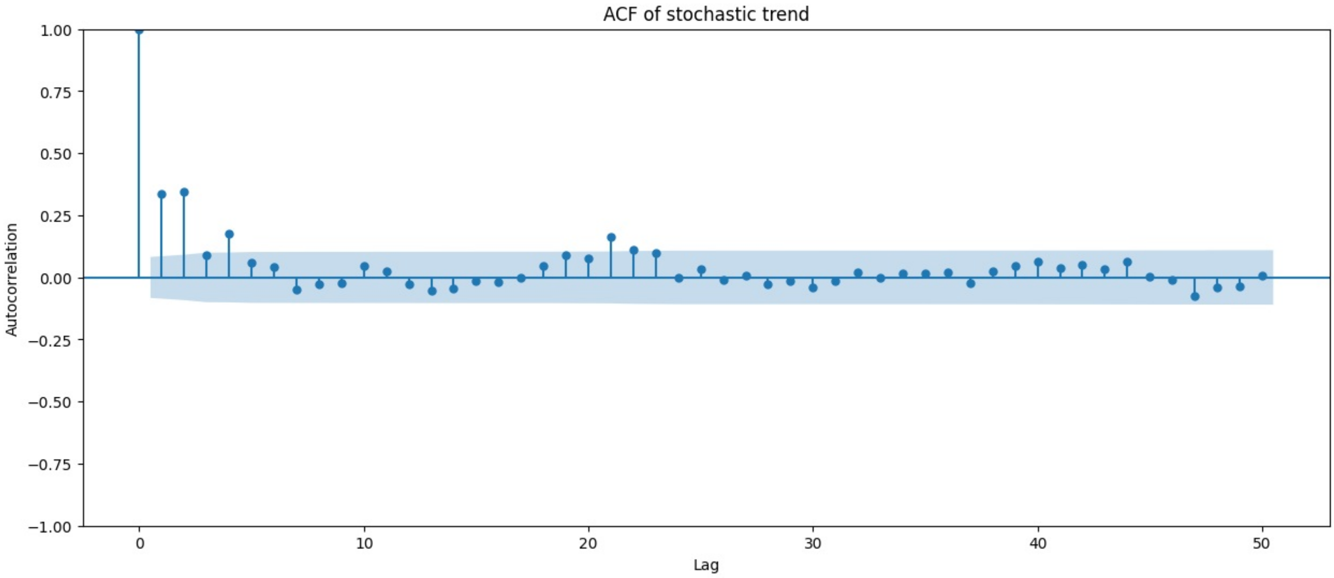


Interpretation

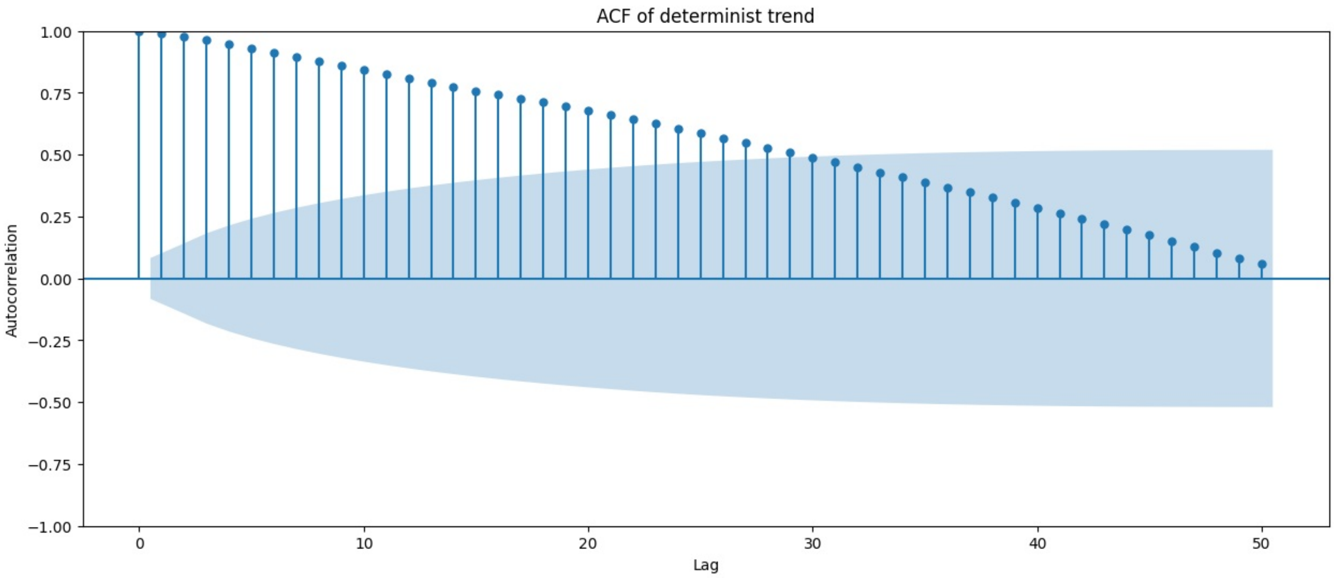
Cette serie demontre la serie logarithmique lorsque un "detrending" stochastique a ete applique. Immediatement, cette serie parait plus comme ce a quoi on pourrait s'attendre d'une serie stationnaire, puisqu'elle varie autour de la moyenne et ne semble pas dependre aussi fortement de l'etat precedent.

1.d) Analyser les fonctions d'autocorrélation de l'échantillon pour évaluer si les séries sont stationnaires. Quelle série choisiriez-vous pour estimer un modèle de série chronologique?

In [ ]:
```
fig, axe_label = plt.subplots()
sm.graphics.tsa.plot_acf(df_stoch["stoch_USDCAD"], lags=50, title="ACF of stochastic trend", ax = axe_label);
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



In [ ]:
```
fig, axe_label = plt.subplots()
sm.graphics.tsa.plot_acf(df_stoch["res"], lags=50, title="ACF of determinist trend", ax = axe_label)
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



Interpretation

Pour commencer, on sait que l'autocorrélation non conditionnelle est une mesure de la dépendance temporelle, ce qui nous permet à partir des graphiques d'autocorrélations, de conclure sur la nature stationnaire de la série. On attend d'une série stationnaire des mouvements autour d'une moyenne et de bornes qui sont fixes dans le temps.

- Concernant le graphique des autocorrélations de la tendance déterministe polynomiale, on observe que les autocorrélations tendent lentement vers 0. La moyenne des autocorrélations n'est donc pas nulle. On peut donc conclure que la série `log_100_USDCAD` n'est pas stationnaire.
- D'autre part, on observe que les autocorrélations de la série à tendance stochastique tendent rapidement vers 0 pour les 50 lags testés, avec une moyenne qui à l'air nulle en fonction du nombre de lags.

On sait que tout processus stationnaire est caractérisé par une courte mémoire: $\mathrm{Corr}(y_t, y_{t-i}) \rightarrow 0$ lorsque i augmente. On peut donc conclure que la série `stoch_USDCAD` est stationnaire.

1.e) A partir de maintenant, utilisez exclusivement la série qui est stationnaire. Effectuez les tests de Ljung-Box avec 1 à 18 lags pour vérifier si la série est un bruit blanc et concluez.

In [ ]:
```
ljung_box_test = sm.stats.acorr_ljungbox(df_stoch["stoch_USDCAD"], lags=18)

#Test if the p-value is inferior to a given thresold for every lag between 1 and 18
p_values = [0.10, 0.05, 0.01]

for p in p_values:
    ljung_box_test[f"p < {p}"] = ljung_box_test["lb_pvalue"] < p

ljung_box_test
```

Out[ ]:

	lb_stat	lb_pvalue	p < 0.1	p < 0.05	p < 0.01
1	63.837877	1.350911e-15	True	True	True
2	130.932915	3.700656e-29	True	True	True
3	135.330291	3.838951e-29	True	True	True
4	153.014709	4.598552e-32	True	True	True
5	155.108804	1.090679e-31	True	True	True
6	156.108470	3.947901e-31	True	True	True
7	157.514338	1.069480e-30	True	True	True
8	157.903502	4.387614e-30	True	True	True
9	158.157909	1.792324e-29	True	True	True
10	159.368871	4.373649e-29	True	True	True
11	159.694552	1.545041e-28	True	True	True
12	160.176521	4.834991e-28	True	True	True
13	161.715202	8.859721e-28	True	True	True
14	162.793787	1.944919e-27	True	True	True
15	162.883982	6.519424e-27	True	True	True
16	163.072280	2.017328e-26	True	True	True
17	163.074212	6.579370e-26	True	True	True
18	164.367578	1.160558e-25	True	True	True

Interpretation

On rappelle que le test de Ljung-Box permet de tester l'hypothèse nulle d'absence d'autocorrélation pour les termes bruits blancs : $H_0 : Corr(y_t, y_{t-i}) = 0, \forall i \in N^+$.

On remarque que les `lb_pvalues` (p-values pour le test de Ljung-Box) sont extrêmement faibles pour les lags 1 à 18. Ceux-ci sont en dessous des seuil critique 1%, 5% et 10% pour tous les lags de 1 à 18. On peut donc rejeter l'hypothèse nulle d'absence de corrélation pour ces lags. On peut donc conclure que les résidus ne sont pas indépendants et que la série est autocorrélée, ce qui implique que le terme d'erreur de la série `stoch_USDCAD` n'est pas bruit blanc.

Partie 2. Choix du modèle (30 points)

Considérons les 8 modèles suivants :

- AR(1): $y_t = \delta + \phi_1 y_{t-1} + \epsilon_t$
- AR(2): $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t$
- AR(3): $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \epsilon_t$
- AR(4): $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \phi_4 y_{t-4} + \epsilon_t$
- ARMA(1,1): $y_t = \delta + \phi_1 y_{t-1} - \theta_1 \epsilon_{t-1} + \epsilon_t$
- ARMA(2,2): $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} + \epsilon_t$
- ARMA(3,3): $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \theta_3 \epsilon_{t-3} + \epsilon_t$
- ARMA(4,4): $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \phi_4 y_{t-4} - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \theta_3 \epsilon_{t-3} - \theta_4 \epsilon_{t-4} + \epsilon_t$

2.a) Estimez les 8 modèles par maximum de vraisemblance, présentez les résultats de l'estimation et vérifiez si les conditions de stationnarité sont satisfaites.

Soit $Y_t = \Delta + \Phi Y_{t-1} + U_t + \Theta U_{t-1}$, l'équation matricielle de l'ARMA(p,q). Afin de vérifier si les conditions de stationnarité sont satisfaites, on peut calculer les valeurs propres de la matrice Φ. On rappelle qu'une condition de stationnarité est satisfaite si toutes les valeurs propres de Φ sont dans le cercle unité.

```
In [ ]: def compute_eigenvalues(arma_model: ARIMAResultsWrapper) -> np.ndarray:
            """Compute the eigenvalues of the ARMA model."""

            # Find length of arparams
            arparams = arma_model.arparams # Store the AR parameters
            n_ar_params_1 = len(arparams) - 1

            # Make the params into a row array
            param_array = arparams.reshape((1, arparams.shape[0]))

            # Create sub array
            ones = np.ones(n_ar_params_1)
            diag_array = np.diag(ones).reshape((n_ar_params_1, n_ar_params_1))

            # Create array of 0s
            zero_col = np.zeros((n_ar_params_1, 1))

            # Stack array
            sub_array = np.hstack((diag_array, zero_col))
            eigmat = np.vstack((param_array, sub_array))

            return np.linalg.eigvals(eigmat) # Compute the eigenvalues of the matrix
```

```
In [ ]: def check_stationarity(eigenvalues: np.ndarray) -> bool:
            """Check if the eigenvalues of the ARMA model are inferior to 1."""
            if eigenvalues.dtype == np.complex128: # If eigenvalues are complex, calculate the norm
                a = np.real(eigenvalues)
                b = np.imag(eigenvalues)
                return all(np.sqrt(a**2 + b**2) < 1)
            else: # If eigenvalues are real, check if they are inferior to 1 in absolute value
                return all(np.abs(eigenvalues) < 1)
```

```
In [ ]: # Fit all ARIMA models and store it in a dictionary.
        # The key is the order of the model and the value is the ARIMA model.
        # Example : arma_models["ARMA(1,1)"] to access the ARMA(1,1) model. etc.
        arma_models = {}

        order_list = [([1], 0, [0]),
                      ([1, 2], 0, [0]),
                      ([1, 2, 3], 0, [0]),
                      ([1, 2, 3, 4], 0, [0]),
                      ([1], 0, [1]),
                      ([1, 2], 0, [1, 2]),
                      ([1, 2, 3], 0, [1, 2, 3]),
                      ([1, 2, 3, 4], 0, [1, 2, 3, 4])]

        for order in order_list:
            print(f"Autoregressive lags: {order[0]}")
            print(f"Moving average lags: {order[2]}")

            model = ARIMA(df_stoch["stoch_USDCAD"], order=order)
            results = model.fit()
            print(results.summary())
            arma_models[f"ARMA({order[0][-1]},{order[2][-1]})"] = results
```

```
Autoregressive lags: [1]
Moving average lags: [0]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(1, 0, 0)   Log Likelihood                -919.472
Date:                Thu, 02 Mar 2023   AIC                           1844.945
Time:                        19:58:16   BIC                           1857.945
Sample:                             0   HQIC                          1850.020
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0249      0.086     -0.291      0.771      -0.193       0.143
ar.L1          0.3353      0.031     10.804      0.000       0.274       0.396
sigma2         1.5345      0.035     43.732      0.000       1.466       1.603
===================================================================================
Ljung-Box (L1) (Q):                   4.25   Jarque-Bera (JB):              4966.20
Prob(Q):                              0.04   Prob(JB):                         0.00
Heteroskedasticity (H):               4.30   Skew:                             1.58
Prob(H) (two-sided):                  0.00   Kurtosis:                        17.20
===================================================================================

Warnings:
```

```
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2]
Moving average lags: [0]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(2, 0, 0)   Log Likelihood                -898.321
Date:                Thu, 02 Mar 2023   AIC                           1804.642
Time:                        19:58:16   BIC                           1821.976
Sample:                             0   HQIC                          1811.409
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0176      0.113     -0.156      0.876      -0.239       0.203
ar.L1          0.2412      0.033      7.256      0.000       0.176       0.306
ar.L2          0.2784      0.030      9.158      0.000       0.219       0.338
sigma2         1.4231      0.041     35.077      0.000       1.344       1.503
===================================================================================
Ljung-Box (L1) (Q):                   0.76   Jarque-Bera (JB):              3465.59
Prob(Q):                              0.38   Prob(JB):                         0.00
Heteroskedasticity (H):               4.63   Skew:                             1.27
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.88
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2, 3]
Moving average lags: [0]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(3, 0, 0)   Log Likelihood                -894.764
Date:                Thu, 02 Mar 2023   AIC                           1799.528
Time:                        19:58:16   BIC                           1821.194
Sample:                             0   HQIC                          1807.986
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0204      0.101     -0.202      0.840      -0.219       0.178
ar.L1          0.2708      0.035      7.722      0.000       0.202       0.340
ar.L2          0.3094      0.032      9.566      0.000       0.246       0.373
ar.L3         -0.1161      0.037     -3.154      0.002      -0.188      -0.044
sigma2         1.4051      0.043     32.415      0.000       1.320       1.490
===================================================================================
Ljung-Box (L1) (Q):                   0.23   Jarque-Bera (JB):              3171.79
Prob(Q):                              0.63   Prob(JB):                         0.00
Heteroskedasticity (H):               4.55   Skew:                             1.22
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.37
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2, 3, 4]
Moving average lags: [0]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(4, 0, 0)   Log Likelihood                -891.030
Date:                Thu, 02 Mar 2023   AIC                           1794.060
Time:                        19:58:16   BIC                           1820.060
Sample:                             0   HQIC                          1804.210
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0178      0.114     -0.157      0.876      -0.240       0.205
ar.L1          0.2838      0.035      8.106      0.000       0.215       0.352
ar.L2          0.2759      0.034      8.088      0.000       0.209       0.343
ar.L3         -0.1514      0.037     -4.063      0.000      -0.224      -0.078
ar.L4          0.1188      0.038      3.164      0.002       0.045       0.192
sigma2         1.3864      0.043     31.884      0.000       1.301       1.472
===================================================================================
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):              3109.47
Prob(Q):                              0.85   Prob(JB):                         0.00
Heteroskedasticity (H):               4.50   Skew:                             1.21
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.26
===================================================================================
```

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1]
Moving average lags: [1]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(1, 0, 1)   Log Likelihood                -906.890
Date:                Thu, 02 Mar 2023   AIC                           1821.779
Time:                        19:58:16   BIC                           1839.113
Sample:                             0   HQIC                          1828.546
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0188      0.114     -0.164      0.869      -0.242       0.205
ar.L1          0.7285      0.049     14.773      0.000       0.632       0.825
ma.L1         -0.4378      0.068     -6.415      0.000      -0.572      -0.304
sigma2         1.4672      0.036     41.174      0.000       1.397       1.537
===================================================================================
Ljung-Box (L1) (Q):                   0.88   Jarque-Bera (JB):              4419.83
Prob(Q):                              0.35   Prob(JB):                         0.00
Heteroskedasticity (H):               4.44   Skew:                             1.46
Prob(H) (two-sided):                  0.00   Kurtosis:                        16.41
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2]
Moving average lags: [1, 2]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(2, 0, 2)   Log Likelihood                -892.400
Date:                Thu, 02 Mar 2023   AIC                           1796.799
Time:                        19:58:17   BIC                           1822.799
Sample:                             0   HQIC                          1806.949
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0200      0.104     -0.192      0.848      -0.224       0.184
ar.L1         -0.1971      0.075     -2.621      0.009      -0.344      -0.050
ar.L2          0.4295      0.076      5.631      0.000       0.280       0.579
ma.L1          0.4819      0.079      6.135      0.000       0.328       0.636
ma.L2         -0.0028      0.089     -0.032      0.974      -0.177       0.171
sigma2         1.3932      0.044     31.725      0.000       1.307       1.479
===================================================================================
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):              3056.67
Prob(Q):                              0.86   Prob(JB):                         0.00
Heteroskedasticity (H):               4.53   Skew:                             1.19
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.16
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2, 3]
Moving average lags: [1, 2, 3]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  563
Model:                 ARIMA(3, 0, 3)   Log Likelihood                -891.399
Date:                Thu, 02 Mar 2023   AIC                           1798.798
Time:                        19:58:17   BIC                           1833.464
Sample:                             0   HQIC                          1812.331
                                - 563
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0185      0.111     -0.167      0.867      -0.236       0.199
ar.L1         -0.0460      0.542     -0.085      0.932      -1.108       1.016
ar.L2          0.4222      0.115      3.669      0.000       0.197       0.648
ar.L3          0.0650      0.231      0.282      0.778      -0.387       0.517
ma.L1          0.3307      0.546      0.606      0.545      -0.740       1.401
ma.L2         -0.0424      0.252     -0.168      0.866      -0.536       0.451
ma.L3         -0.1384      0.083     -1.677      0.094      -0.300       0.023
sigma2         1.3883      0.045     30.575      0.000       1.299       1.477
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):              3076.26
Prob(Q):                              0.88   Prob(JB):                         0.00
```

```
Heteroskedasticity (H):                4.48   Skew:                             1.20
Prob(H) (two-sided):                   0.00   Kurtosis:                        14.20
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2, 3, 4]
Moving average lags: [1, 2, 3, 4]
                               SARIMAX Results
==============================================================================
Dep. Variable:            stoch_USDCAD   No. Observations:                  563
Model:                  ARIMA(4, 0, 4)   Log Likelihood                -890.153
Date:                 Thu, 02 Mar 2023   AIC                           1800.305
Time:                         19:58:18   BIC                           1843.638
Sample:                              0   HQIC                          1817.221
                                 - 563
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0037      0.106     -0.035      0.972      -0.211       0.203
ar.L1          0.8918      0.083     10.689      0.000       0.728       1.055
ar.L2         -0.2750      0.104     -2.657      0.008      -0.478      -0.072
ar.L3         -0.6861      0.112     -6.111      0.000      -0.906      -0.466
ar.L4          0.3929      0.081      4.826      0.000       0.233       0.552
ma.L1         -0.6049      0.090     -6.731      0.000      -0.781      -0.429
ma.L2          0.3796      0.100      3.800      0.000       0.184       0.575
ma.L3          0.4881      0.111      4.391      0.000       0.270       0.706
ma.L4          0.0397      0.091      0.438      0.662      -0.138       0.218
sigma2         1.3803      0.046     29.695      0.000       1.289       1.471
===================================================================================
Ljung-Box (L1) (Q):                    0.00   Jarque-Bera (JB):              3189.22
Prob(Q):                               0.99   Prob(JB):                         0.00
Heteroskedasticity (H):                4.51   Skew:                             1.25
Prob(H) (two-sided):                   0.00   Kurtosis:                        14.39
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [ ]:

```python
df_stationarity = pd.DataFrame(index=arma_models.keys(), columns=["is_stationary"])

for model_name, model in arma_models.items():
    eigenvalues = compute_eigenvalues(model)
    df_stationarity.loc[model_name, "is_stationary"] = check_stationarity(eigenvalues)

df_stationarity
```

Out[ ]:

	is_stationary
ARMA(1,0)	True
ARMA(2,0)	True
ARMA(3,0)	True
ARMA(4,0)	True
ARMA(1,1)	True
ARMA(2,2)	True
ARMA(3,3)	True
ARMA(4,4)	True

On constate que les conditions de stationnarité sont satisfaites pour tous les modèles i.e les valeurs propres pour chacun des modèles sont dans le cercle unité.

2.b) Effectuez des tests de rapport de vraisemblance pour justifier la sélection de deux modèles. Sélectionnez d'abord le meilleur parmi AR(1), AR(2), AR(3) et AR(4). Sélectionnez ensuite le meilleur modèle parmi ARMA(1,1), ARMA(2,2), ARMA(3,3) et ARMA(4,4). Utilisez le BIC pour justifier la sélection du meilleur de ces deux modèles.

In [ ]:

```python
def compute_lr_statistic(reduced_ll: float, full_ll: float) -> float:
    """Compute the likelihood ratio statistic.

    Args:
        reduced_ll (float): Log likehood of the reduced model.
        full_ll (float): Log likehood of the full model.

    Returns:
        float: The Likehood-Ratio Test Statistics
    """
    return -2 * (reduced_ll - full_ll)
```

In [ ]:
```
def compute_p_value_llr(reduced_ll: float, full_ll: float, df: int = 1) -> float:
    """Compute the p-value of the likelihood ratio test.

    Args:
        reduced_ll (float): Log likehood of the reduced model.
        full_ll (float): Log likehood of the full model.
        df (int): Degree of freedom of the Chi-Square test.

    Returns:
        float: Returns the p-value of the test. That is the probability that the full model is better than the
    """
    lr_statistic = compute_lr_statistic(reduced_ll=reduced_ll, full_ll=full_ll)
    p_val = sp.stats.chi2.sf(lr_statistic, df=df)
    return p_val
```

On commence par sélectionner le meilleur modèle parmi AR(1), AR(2), AR(3) et AR(4).

- H0: Le modèle complet ARMA(2,0) fit les données aussi bien que le modèle réduit ARMA(1,0). Par conséquent, on devrait choisir le modèle ARMA(1,0) car il est plus simple.
- H1: Le modèle complet ARMA(2,0) fit mieux les données que le modèle réduit ARMA(1,0). Par conséquent, on devrait choisir le modèle ARMA(2,0) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma_models["ARMA(1,0)"].llf,
                    arma_models["ARMA(2,0)"].llf,
                    df=1)
```

Out[ ]:
```
7.820378576429521e-11
```

Comme la p-value est inférieur au seuil significatif de 5%, on rejette l'hypothèse nulle et on peut donc conclure que le modèle ARMA(2,0) fit mieux les données que le modèle ARMA(1,0).

- H0: Le modèle complet ARMA(3,0) fit les données aussi bien que le modèle réduit ARMA(2,0). Par conséquent, on devrait choisir le modèle ARMA(2,0) car il est plus simple.
- H1: Le modèle complet ARMA(3,0) fit mieux les données que le modèle réduit ARMA(2,0). Par conséquent, on devrait choisir le modèle ARMA(3,0) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma_models["ARMA(2,0)"].llf,
                    arma_models["ARMA(3,0)"].llf,
                    df=1)
```

Out[ ]:
```
0.007646861433316317
```

Comme la p-value est inférieur au seuil significatif de 5%, on rejette l'hypothèse nulle et on peut donc conclure que le modèle ARMA(3,0) fit mieux les données que le modèle ARMA(2,0).

- H0: Le modèle complet ARMA(4,0) fit les données aussi bien que le modèle réduit ARMA(3,0). Par conséquent, on devrait choisir le modèle ARMA(3,0) car il est plus simple.
- H1: Le modèle complet ARMA(4,0) fit mieux les données que le modèle réduit ARMA(3,0). Par conséquent, on devrait choisir le modèle ARMA(4,0) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma_models["ARMA(3,0)"].llf,
                    arma_models["ARMA(4,0)"].llf,
                    df=1)
```

Out[ ]:
```
0.006280983257625679
```

Comme la p-value est inférieur au seuil significatif de 5%, on rejette l'hypothèse nulle et on peut donc conclure que le modèle ARMA(4,0) fit mieux les données que le modèle ARMA(3,0).

On peut donc conclure que d'après le test du ratio de vraisemblance, le modèle AR(4) est le meilleur modèle pour modéliser les données parmis les modèles AR.

On utilise le même raisonnement pour les modèles ARMA(p, q) afin de sélectionner le meilleur modèle parmi ARMA(1,1), ARMA(2,2), ARMA(3,3) et ARMA(4,4).

- H0: Le modèle complet ARMA(2,2) fit les données aussi bien que le modèle réduit ARMA(1,1). Par conséquent, on devrait choisir le modèle ARMA(1,1) car il est plus simple.
- H1: Le modèle complet ARMA(2,2) fit mieux les données que le modèle réduit ARMA(1,1). Par conséquent, on devrait choisir le modèle ARMA(2,2) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma_models["ARMA(1,1)"].llf,
```

```
                    arma_models["ARMA(2,2)"].llf,
                    df=2)
```

Out[ ]: 5.093383915436467e-07

Comme la p-value est inférieur au seuil significatif de 5%, on rejette l'hypothèse nulle et on peut donc conclure que le modèle ARMA(2,2) fit mieux les données que le modèle ARMA(1,1).

- H0: Le modèle complet ARMA(3,3) fit les données aussi bien que le modèle réduit ARMA(2,2). Par conséquent, on devrait choisir le modèle ARMA(2,2) car il est plus simple.
- H1: Le modèle complet ARMA(3,3) fit mieux les données que le modèle réduit ARMA(2,2). Par conséquent, on devrait choisir le modèle ARMA(3,3) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma_models["ARMA(2,2)"].llf,
                    arma_models["ARMA(3,3)"].llf,
                    df=2)
```

Out[ ]: 0.3676921427807781

Comme la p-value est supérieur au seuil significatif de 5%, on ne peut pas rejeter l'hypothèse nulle et on peut donc conclure que le modèle ARMA(2,2) fit aussi bien les données que le modèle ARMA(3,3). Par conséquent, on devrait choisir le modèle ARMA(2,2) car il est plus simple.

- H0: Le modèle complet ARMA(4,4) fit les données aussi bien que le modèle réduit ARMA(2,2). Par conséquent, on devrait choisir le modèle ARMA(2,2) car il est plus simple.
- H1: Le modèle complet ARMA(4,4) fit mieux les données que le modèle réduit ARMA(2,2). Par conséquent, on devrait choisir le modèle ARMA(4,4) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma_models["ARMA(2,2)"].llf,
                    arma_models["ARMA(4,4)"].llf,
                    df=4)
```

Out[ ]: 0.3432525503882037

Comme la p-value est supérieur au seuil significatif de 5%, on ne peut pas rejeter l'hypothèse nulle et on peut donc conclure que le modèle ARMA(2,2) fit aussi bien les données que le modèle ARMA(4,4). Par conséquent, on devrait choisir le modèle ARMA(2,2) car il est plus simple.

On peut donc conclure que d'après le test du ratio de vraisemblance, le modèle ARMA(2,2) est le meilleur modèle pour modéliser les données parmis les modèles ARMA.

In [ ]:
```
df_bic = pd.DataFrame({"BIC": [arma_models["ARMA(2,2)"].bic, arma_models["ARMA(4,0)"].bic]},
                      index=["ARMA(2,2)", "AR(4)"])
df_bic
```

Out[ ]:

	BIC
ARMA(2,2)	1822.798787
AR(4)	1820.059890

On remarque que le BIC du modèle AR(4) est inférieur à celui du modèle ARMA(2,2). Par conséquent, et d'après le critère d'information de Bayes, le modèle AR(4) est le meilleur modèle pour modéliser les données.

2.c) Évaluez l'hypothèse du bruit blanc pour chacun des deux modèles, et pour le modèle AR(1). Que pouvons-nous conclure ?

Soit $H_0 : \mathrm{Corr}(y_t, y_{t-i}) = 0, \forall i \in N^+$, l'hypothèse nulle du test de Ljung-Box (les termes d'erreurs ne présentent pas d'autocorrélation).

$H_1 : \mathrm{Corr}(y_t, y_{t-i}) \neq 0$ pour au moins un $i \in N^+$ Dans notre cas, on test 18 lags.

In [ ]:
```
resid_arma_2_2 = arma_models["ARMA(2,2)"].resid
resid_arma_4_0 = arma_models["ARMA(4,0)"].resid
resid_arma_1_0 = arma_models["ARMA(1,0)"].resid
```

In [ ]:
```
ljung_box_test = sm.stats.acorr_ljungbox(resid_arma_2_2, lags=18, return_df=True)

#Test if the p-value is inferior to a given thresold for every lag between 1 and 18
p_values = [0.10, 0.05, 0.01]

for p in p_values:
    ljung_box_test[f"p < {p}"] = ljung_box_test["lb_pvalue"] < p

ljung_box_test
```

Out[ ]:

	lb_stat	lb_pvalue	p < 0.1	p < 0.05	p < 0.01
1	0.019365	0.889326	False	False	False
2	0.687187	0.709217	False	False	False
3	0.707994	0.871322	False	False	False
4	1.595710	0.809563	False	False	False
5	4.884819	0.430098	False	False	False
6	5.217740	0.516205	False	False	False
7	6.400846	0.493800	False	False	False
8	7.974911	0.435925	False	False	False
9	8.162355	0.517871	False	False	False
10	10.680986	0.382906	False	False	False
11	12.215130	0.347692	False	False	False
12	13.144515	0.358625	False	False	False
13	14.564130	0.335347	False	False	False
14	15.017013	0.376993	False	False	False
15	15.194254	0.437517	False	False	False
16	15.331456	0.500509	False	False	False
17	16.381676	0.496957	False	False	False
18	17.481005	0.490301	False	False	False

In [ ]:

```python
# Check that all lb_pvalues are > 0.05
all(sm.stats.acorr_ljungbox(resid_arma_2_2, lags=18, return_df=True).lb_pvalue > 0.05)
```

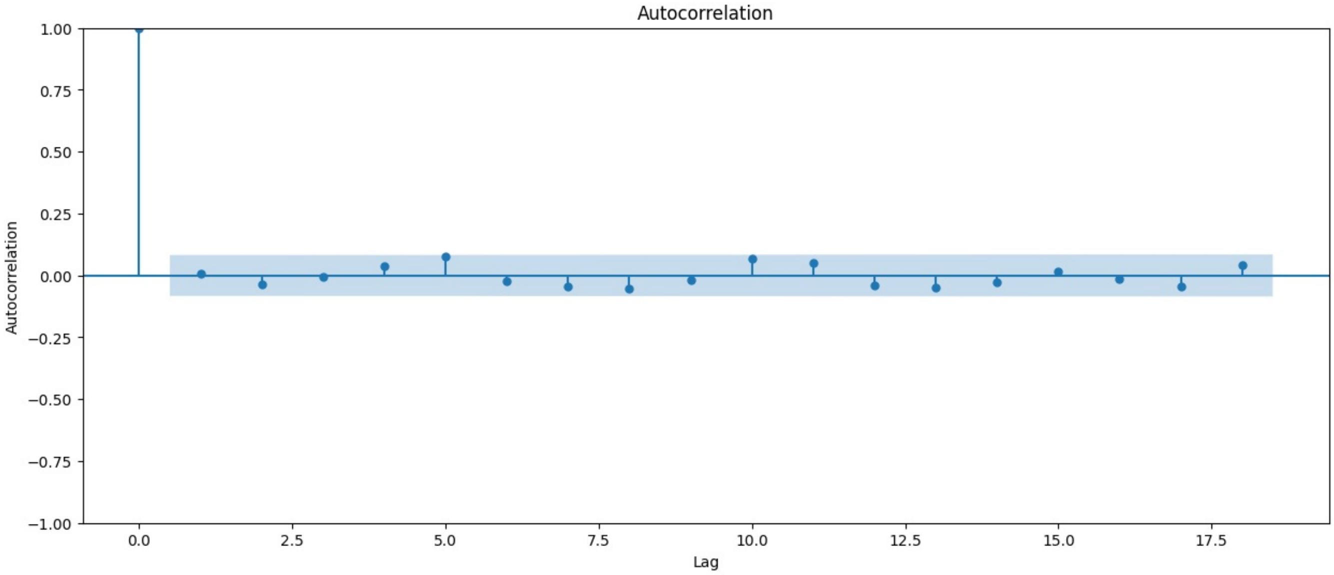
Out[ ]:

```
True
```

Après avoir effectué le test de Ljung-Box sur le modèle ARMA(2,2), on remarque que la p-value est supérieur au seuil significatif de 5% pour tous les 18 lags. Par conséquent, on ne peut pas rejeter l'hypothèse nulle que le terme d'erreur possède des autocorrélations de 0. Par conséquent, on peut conclure que le terme d'erreur du modèle ARMA(2,2) est un bruit blanc.

In [ ]:

```python
# Autocorrelation plot to confirm our results
fig, axe_label = plt.subplots()
sm.graphics.tsa.plot_acf(resid_arma_2_2, lags=18, ax=axe_label)
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



In [ ]:

```python
ljung_box_test = sm.stats.acorr_ljungbox(resid_arma_4_0, lags=18, return_df=True)

#Test if the p-value is inferior to a given thresold for every lag between 1 and 18
p_values = [0.10, 0.05, 0.01]

for p in p_values:
    ljung_box_test[f"p < {p}"] = ljung_box_test["lb_pvalue"] < p

ljung_box_test
```

Out[ ]:

	lb_stat	lb_pvalue	p < 0.1	p < 0.05	p < 0.01
1	0.022608	0.880481	False	False	False
2	0.160005	0.923114	False	False	False
3	0.284247	0.962964	False	False	False
4	0.286832	0.990648	False	False	False
5	0.827707	0.975233	False	False	False
6	1.626631	0.950644	False	False	False
7	3.507857	0.834393	False	False	False
8	4.859857	0.772444	False	False	False
9	5.277459	0.809482	False	False	False
10	7.568857	0.670869	False	False	False
11	8.909719	0.630225	False	False	False
12	9.628721	0.648496	False	False	False
13	10.908685	0.618467	False	False	False
14	11.634383	0.635638	False	False	False
15	11.674433	0.703486	False	False	False
16	11.946228	0.747673	False	False	False
17	13.270993	0.717864	False	False	False
18	14.420867	0.701272	False	False	False

In [ ]:

```
# Check that all lb_pvalues are > 0.05
all(sm.stats.acorr_ljungbox(resid_arma_4_0, lags=18, return_df=True).lb_pvalue > 0.05)
```

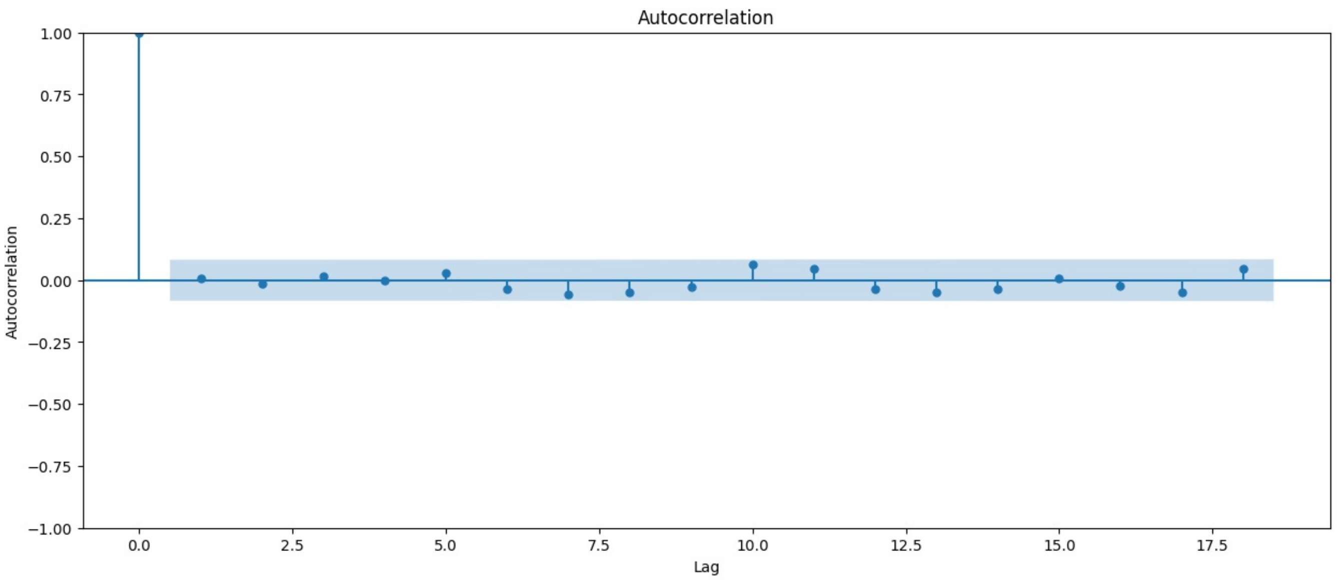
Out[ ]:

```
True
```

Après avoir effectué le test de Ljung-Box sur le modèle AR(4), on remarque que la p-value est supérieur au seuil significatif de 5% pour tous les 18 lags. Par conséquent, on ne peut pas rejeter l'hypothèse nulle que le terme d'erreur possède des autocorrélations de 0. Par conséquent, on peut conclure que le terme d'erreur du modèle AR(4) est un bruit blanc.

In [ ]:

```
# Autocorrelation plot to confirm our results
fig, axe_label = plt.subplots(figsize=(15,6))
sm.graphics.tsa.plot_acf(resid_arma_4_0, lags=18, ax=axe_label)
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



In [ ]:

```
ljung_box_test = sm.stats.acorr_ljungbox(resid_arma_1_0, lags=18, return_df=True)

#Test if the p-value is inferior to a given thresold for every lag between 1 and 18
p_values = [0.10, 0.05, 0.01]

for p in p_values:
    ljung_box_test[f"p < {p}"] = ljung_box_test["lb_pvalue"] < p

ljung_box_test
```

Out[ ]:

	lb_stat	lb_pvalue	p < 0.1	p < 0.05	p < 0.01
1	4.261575	3.898377e-02	True	True	False
2	45.933833	1.060706e-10	True	True	True
3	50.167877	7.357811e-11	True	True	True
4	65.664649	1.863951e-13	True	True	True
5	65.689926	8.060455e-13	True	True	True
6	67.027592	1.661358e-12	True	True	True
7	69.662315	1.728765e-12	True	True	True
8	69.683916	5.680488e-12	True	True	True
9	70.341134	1.305289e-11	True	True	True
10	72.165245	1.690236e-11	True	True	True
11	72.480374	4.114880e-11	True	True	True
12	72.856889	9.320066e-11	True	True	True
13	73.676291	1.677939e-10	True	True	True
14	74.203834	3.311269e-10	True	True	True
15	74.236735	7.775153e-10	True	True	True
16	74.408672	1.665443e-09	True	True	True
17	74.510374	3.558077e-09	True	True	True
18	74.904251	6.572236e-09	True	True	True

In [ ]:

```
# Check that all lb_pvalues are < 0.05
all(sm.stats.acorr_ljungbox(resid_arma_1_0, lags=18, return_df=True).lb_pvalue < 0.05)
```

Out[ ]:

```
True
```

Après avoir effectué le test de Ljung-Box sur le modèle AR(1), on remarque que la p-value est inférieur au seuil significatif de 1%, 5% et 10% pour tous les 18 lags. Par conséquent, on peut rejeter l'hypothèse nulle que le terme d'erreur possède des autocorrélations de 0. Par conséquent, on peut conclure que le terme d'erreur du modèle AR(1) n'est pas un bruit blanc.

Partie 3. Réponse dynamique et prévision (20 points)

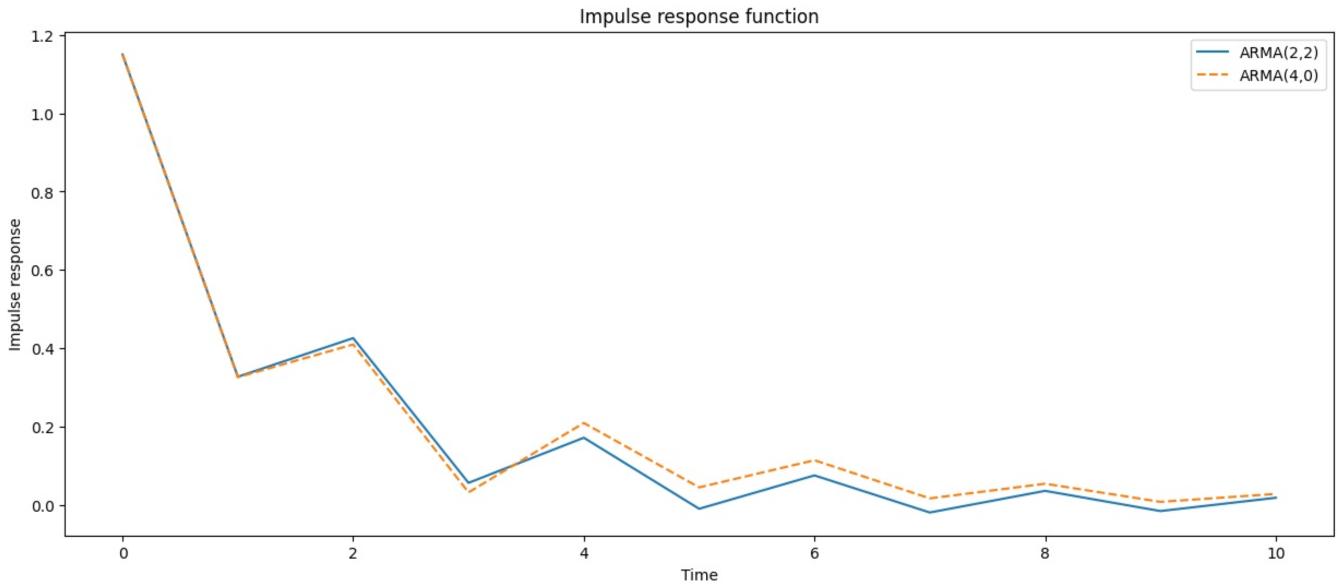
3.a) Pour les deux modèles sélectionnés, évaluez la réponse dynamique pour un horizon de 10 périodes suite à un choc positif de taille σ = 1.15 survenant à la première période de l'horizon. Tracez les deux fonctions de réponse impulsionnelle sur la même figure et commentez.

In [ ]:

```
# Choc sigma is constant
steps = 10
sigma = 1.15
#Compute the impulse response
impulse_res_arma22 = arma_models["ARMA(2,2)"].impulse_responses(steps=steps, impulse=[sigma])
impulse_res_arma40 = arma_models["ARMA(4,0)"].impulse_responses(steps=10, impulse=[sigma])
```

In [ ]:

```
plt.plot(impulse_res_arma22, label="ARMA(2,2)")
plt.plot(impulse_res_arma40, label="ARMA(4,0)", linestyle="--")
plt.legend()
plt.title("Impulse response function")
plt.xlabel("Time")
plt.ylabel("Impulse response")
plt.show()
```



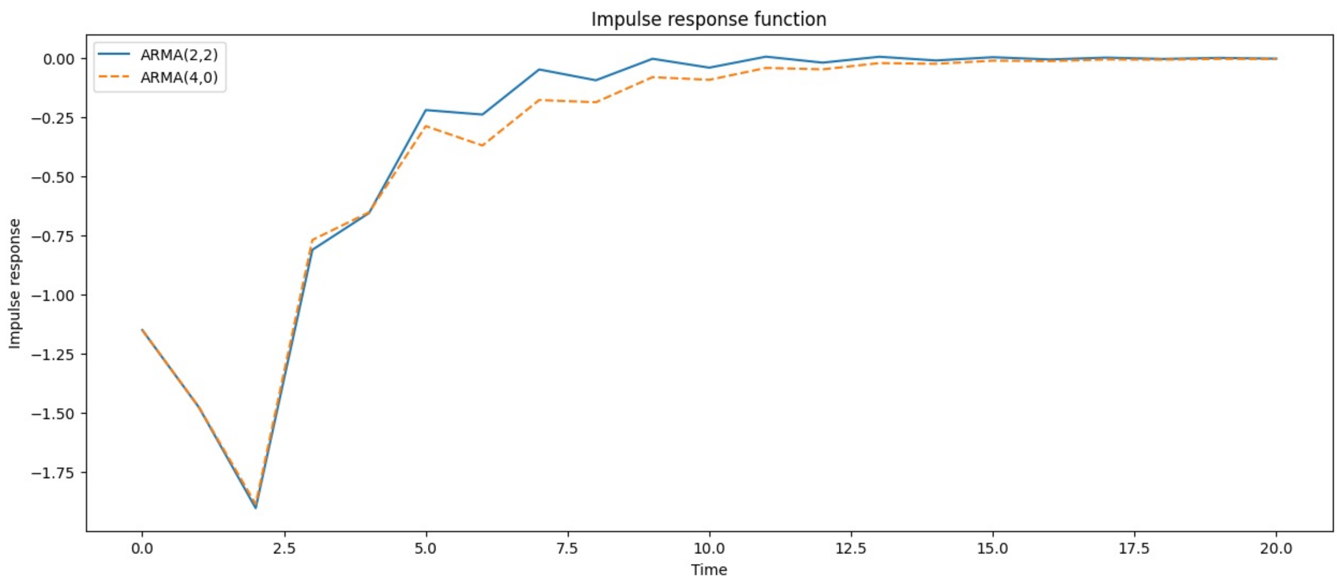
Puisque les conditions de stationnarité sont satisfaites pour les deux modèles, nous observons que la réponse dynamique tend vers zéro asymptotiquement. Par conséquent, le choc positif de taille σ a un effet transitoire dans chaque cas.

3.b) Un choc négatif de taille σ = 1.15 se produit pendant 3 périodes consécutives de l'horizon (t, t+1, t+2). Tracez les deux fonctions de réponse impulsionnelle pour les deux modèles sélectionnés, montrant la réponse dynamique pour un horizon de 20 périodes après ces chocs, et expliquez ce qui se passe.

In [ ]:
```python
def compute_three_impulse(arma_model, sigma, steps):
    """Compute the impulse response of three subsequent shocks in the earliest periods"""
    first_impulse = arma_model.impulse_responses(steps=int(steps), impulse=[sigma])
    second_impulse = first_impulse
    second_impulse = pd.concat([pd.Series([0]), second_impulse], ignore_index=True)
    second_impulse = second_impulse[0:-1]
    third_impulse = first_impulse
    third_impulse = pd.concat([pd.Series([0, 0]), third_impulse], ignore_index=True)
    third_impulse = third_impulse[0:-2]
    total_impulse = first_impulse + second_impulse + third_impulse
    return total_impulse
```

In [ ]:
```python
steps = 20
sigma = -1.15
#Compute the impulse response
arma22_total_impulse = compute_three_impulse(arma_models["ARMA(2,2)"], sigma, steps)
arma40_total_impulse = compute_three_impulse(arma_models["ARMA(4,0)"], sigma, steps)
```

In [ ]:
```python
# Draw the Impulse plot
plt.plot(arma22_total_impulse, label="ARMA(2,2)")
plt.plot(arma40_total_impulse, label="ARMA(4,0)", linestyle="--")
plt.legend()
plt.title("Impulse response function")
plt.xlabel("Time")
plt.ylabel("Impulse response")
plt.show()
```



Comme à la question précédente, il devient encore plus évident que la réponse dynamique $\psi_j = \partial y_{t+j}/\partial u_t$ devient nulle lorsque $j \to \infty$. Les

Comme à la question précédente, il devient encore plus évident que la réponse dynamique ψ_j = ... devient nulle lorsque $j \to \infty$. Les trois chocs négatifs consécutifs de taille σ ont un effet plus grand sur la fonction de réponse impulsionnelle initialement, mais l'effet est clairement transitoire à long terme.

3.c) Prévision : Divisez l'échantillon en un échantillon d'entraînement (training sample) et un échantillon de validation (holdout sample). L'échantillon de validation devrait être composé des 34 dernières observations.

- **Ré-estimez les deux modèles sélectionnés en utilisant uniquement l'échantillon d'entraînement.**
- **Pour chacun des deux modèles, calculez les prévisions à un pas en avant : $E_t(USDCAD_{t+1})$ pour les modèles AR(4) et ARMA(4,4).**
- **Tracez ces prévisions avec la série réelle (différence première) sur le même graphique pour la période couverte par l'échantillon de validation.**
- **Tracez la série réelle (données brutes en niveaux) contre les prévisions à un pas d'avance et la prévision naïve : $E_t(USDCAD_{t+1}) = USDCAD_t$.**

In [ ]:
```python
# Estimate both models on training sample
treshold = 34
train_sample = df_stoch["stoch_USDCAD"].iloc[:-treshold]
val_sample = df_stoch["stoch_USDCAD"].iloc[-treshold:]

arma_models2 = {}
order_list = [([1, 2, 3, 4], 0, [0]),
              ([1, 2], 0, [1, 2]),
              ([1, 2, 3, 4], 0, [1, 2, 3, 4])]

for order in order_list:
    print(f"Autoregressive lags: {order[0]}")
    print(f"Moving average lags: {order[2]}")

    model = ARIMA(train_sample, order=order)
    results = model.fit()
    print(results.summary())
    arma_models2[f"ARMA({order[0][-1]},{order[2][-1]})"] = results
```

```
Autoregressive lags: [1, 2, 3, 4]
Moving average lags: [0]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  529
Model:                 ARIMA(4, 0, 0)   Log Likelihood                -826.065
Date:                Thu, 02 Mar 2023   AIC                           1664.130
Time:                        19:58:20   BIC                           1689.756
Sample:                             0   HQIC                          1674.161
                                - 529
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0176      0.122     -0.144      0.886      -0.257       0.222
ar.L1          0.3024      0.036      8.376      0.000       0.232       0.373
ar.L2          0.2910      0.035      8.286      0.000       0.222       0.360
ar.L3         -0.1726      0.038     -4.547      0.000      -0.247      -0.098
ar.L4          0.1327      0.040      3.357      0.001       0.055       0.210
sigma2         1.3289      0.042     31.343      0.000       1.246       1.412
===================================================================================
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):              3631.46
Prob(Q):                              0.79   Prob(JB):                         0.00
Heteroskedasticity (H):               4.33   Skew:                             1.38
Prob(H) (two-sided):                  0.00   Kurtosis:                        15.53
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2]
Moving average lags: [1, 2]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  529
Model:                 ARIMA(2, 0, 2)   Log Likelihood                -827.330
Date:                Thu, 02 Mar 2023   AIC                           1666.659
Time:                        19:58:20   BIC                           1692.285
Sample:                             0   HQIC                          1676.690
                                - 529
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0205      0.111     -0.185      0.853      -0.238       0.197
ar.L1         -0.2105      0.069     -3.049      0.002      -0.346      -0.075
ar.L2          0.4625      0.072      6.414      0.000       0.321       0.604
```

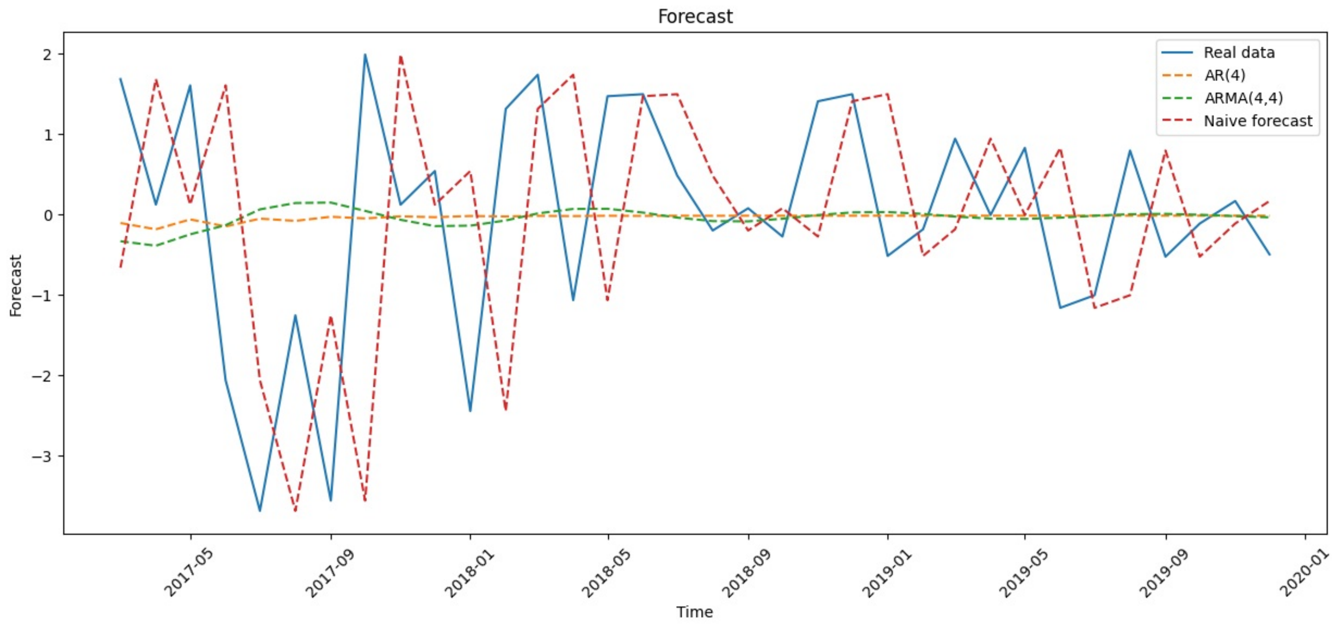
```
ma.L1          0.5154      0.073      7.053      0.000       0.372       0.659
ma.L2         -0.0018      0.086     -0.021      0.983      -0.170       0.166
sigma2         1.3354      0.043     31.166      0.000       1.251       1.419
===================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):              3553.93
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               4.38   Skew:                             1.36
Prob(H) (two-sided):                  0.00   Kurtosis:                        15.41
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Autoregressive lags: [1, 2, 3, 4]
Moving average lags: [1, 2, 3, 4]
                               SARIMAX Results
==============================================================================
Dep. Variable:           stoch_USDCAD   No. Observations:                  529
Model:                 ARIMA(4, 0, 4)   Log Likelihood                -819.322
Date:                Thu, 02 Mar 2023   AIC                           1658.644
Time:                        19:58:21   BIC                           1701.354
Sample:                             0   HQIC                          1675.363
                                - 529
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0216      0.106     -0.203      0.839      -0.230       0.187
ar.L1          1.0108      0.114      8.835      0.000       0.787       1.235
ar.L2         -0.1887      0.149     -1.264      0.206      -0.481       0.104
ar.L3         -0.6044      0.135     -4.476      0.000      -0.869      -0.340
ar.L4          0.2184      0.090      2.430      0.015       0.042       0.395
ma.L1         -0.7202      0.107     -6.700      0.000      -0.931      -0.510
ma.L2          0.2863      0.132      2.162      0.031       0.027       0.546
ma.L3          0.3213      0.125      2.573      0.010       0.077       0.566
ma.L4          0.2077      0.091      2.275      0.023       0.029       0.387
sigma2         1.2913      0.045     28.435      0.000       1.202       1.380
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):              4137.11
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               4.07   Skew:                             1.51
Prob(H) (two-sided):                  0.00   Kurtosis:                        16.36
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [ ]:

```python
# Compute predictions for one period ahead
forecast_arma40 = arma_models2["ARMA(4,0)"].forecast(steps=treshold).reset_index(drop=True)
forecast_arma44 = arma_models2["ARMA(4,4)"].forecast(steps=treshold).reset_index(drop=True)

val_sample.reset_index(drop=True)
forecast_naive = np.zeros(treshold)
forecast_naive[0] = train_sample.iloc[-1]
forecast_naive[1:] = val_sample.iloc[:-1]

# Change index to date for plotting
forecast_arma40.index = df_stoch["date"].iloc[-treshold:]
forecast_arma44.index = df_stoch["date"].iloc[-treshold:]
val_sample.index = df_stoch["date"].iloc[-treshold:]
forecast_naive = pd.Series(forecast_naive, index=df_stoch["date"].iloc[-treshold:])
```

In [ ]:

```python
#Draw the plot of the forecast
plt.plot(val_sample, label="Real data")
plt.plot(forecast_arma40, label="AR(4)", linestyle="--")
plt.plot(forecast_arma44, label="ARMA(4,4)", linestyle="--")
plt.plot(forecast_naive, label="Naive forecast", linestyle="--")
plt.legend()
plt.title("Forecast")
plt.xlabel("Time")
# Rotate xticks by 45 degrees
plt.xticks(rotation=45)
plt.ylabel("Forecast")
plt.show()
```



3.d) Indiquez l'erreur quadratique moyenne pour chaque modèle en utilisant uniquement l'échantillon de 34 observations pour évaluer la performance de la prévision. Comparez ces statistiques à l'estimateur naïf suivant : $E_t(y_{t+1}) = y_t$ pour h=1 seulement. Quel modèle devrions-nous utiliser dans chaque cas ?

In [ ]:
```
def compute_mse(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

mse_arma40 = compute_mse(val_sample, forecast_arma40)
mse_arma44 = compute_mse(val_sample, forecast_arma44)
mse_naive = compute_mse(val_sample, forecast_naive)

mse_df = pd.DataFrame({"AR(4)": mse_arma40,
                       "ARMA(4,4)": mse_arma44,
                       "Naive forecast": mse_naive},
                      index=["MSE"])
mse_df
```

Out[ ]:

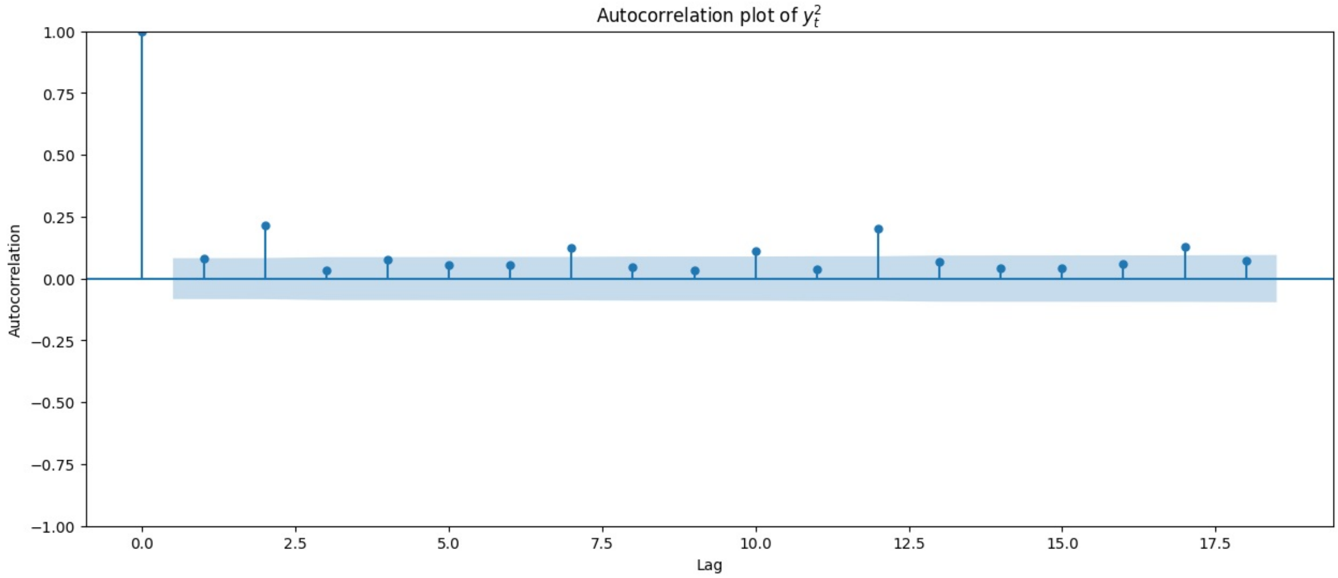
	AR(4)	ARMA(4,4)	Naive forecast
MSE	1.986487	2.087125	3.832131

On remarque que dans le cas des deux modèles ARMA, l'erreur quadratique moyenne est plus faible que celle du modèle naïf. De plus, le modèle AR(4) est plus performant que le modèle ARMA(4,4) dans le cas de la prédiction à un pas en avant, d'après la métrique du MSE. On peut donc conclure que le modèle AR(4) est le plus approprié pour la prédiction à un pas en avant.

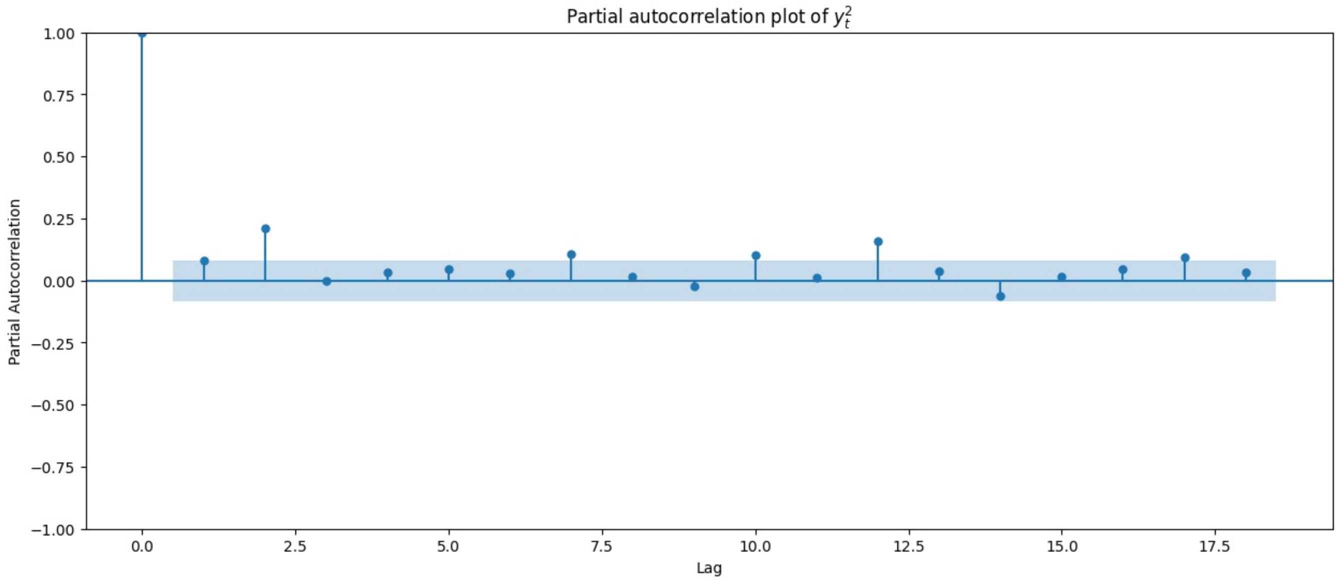
Partie 4. Améliorations supplémentaires par la modélisation de la variance conditionnelle (30 points)

4.a) Tracez les fonctions d'autocorrélation et d'autocorrélation partielle de y_{2t}. Que pouvez-vous conclure ?

In [ ]:
```
df_stoch["stoch_USDCAD**2"] = df_stoch["stoch_USDCAD"] ** 2

# Plot autocorrelation
fig, axe_label = plt.subplots(figsize=(15,6))
sm.graphics.tsa.plot_acf(df_stoch["stoch_USDCAD**2"], lags=18, title=f"Autocorrelation plot of $y_t^2$", ax=axe
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



```
In [ ]: # Plot partial autocorrelation
        fig, axe_label = plt.subplots(figsize=(15,6))
        sm.graphics.tsa.plot_pacf(df_stoch["stoch_USDCAD**2"], lags=18, title=f"Partial autocorrelation plot of $y_t^2$
        axe_label.set_xlabel("Lag")
        axe_label.set_ylabel("Partial Autocorrelation")
        plt.show()
```



```
In [ ]: sm.stats.acorr_ljungbox(df_stoch["stoch_USDCAD**2"], lags=18)
```

Out[ ]:

	lb_stat	lb_pvalue
1	3.760424	5.247928e-02
2	29.896207	3.221968e-07
3	30.458884	1.104966e-06
4	33.891385	7.844033e-07
5	35.740680	1.070269e-06
6	37.483063	1.417428e-06
7	46.434587	7.194495e-08
8	47.746010	1.104781e-07
9	48.313950	2.231168e-07
10	55.549973	2.492367e-08
11	56.391805	4.305277e-08
12	79.601998	4.915342e-12
13	82.404824	3.884540e-12
14	83.359975	6.705421e-12
15	84.392255	1.089350e-11
16	86.467592	1.113452e-11
17	96.115079	4.642082e-13
18	99.324257	2.944998e-13

Dans le cas du graphique d'autocorrélation de la série y_{2t}, on remarque que quelques autocorrélations sortent de la borne de variance, comme pour le retard 2, 7, 10, 12 et 17. On observe le même phénomène pour l'autocorrélation partielle avec les mêmes retards. Par conséquent, on pourrait conclure que la série y_{2t} n'est pas généré par un processus bruit blanc de moyenne nulle et de variance constante au cours du temps.

4.b) Estimez les versions ARCH(1) et GARCH(1,1) des deux modèles sélectionnés par maximum de vraisemblance en utilisant seulement l'échantillon d'entraînement, et rapportez les estimations.

In [ ]:

```python
def compute_arima_garch_models(train_sample: pd.Series,
                               p_arima: int,
                               q_arima: int,
                               p_garch: int,
                               q_garch: int,
                               vol: str = "GARCH"):
    """Compute ARIMA and GARCH models."""
    # Estimate ARIMA model
    arima = ARIMA(train_sample, order=(p_arima, 0, q_arima))
    arima_results = arima.fit()
    arima_resid = arima_results.resid

    # Estimate GARCH model
    garch = arch_model(arima_resid, p=p_garch, q=q_garch, vol=vol)
    garch_results = garch.fit(disp="off")

    return arima_results, garch_results
```

In [ ]:

```python
# Estimate a AR(4)ARCH(1) model
ar4, ar4arch1 = compute_arima_garch_models(train_sample, p_arima=4, q_arima=0, p_garch=1, q_garch=0, vol="ARCH"
ar4arch1.summary()
```

Out[ ]:

Constant Mean - ARCH Model Results

Dep. Variable:	None	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-820.306
Distribution:	Normal	**AIC:**	1646.61
Method:	Maximum Likelihood	**BIC:**	1659.42
		No. Observations:	529
Date:	Thu, Mar 02 2023	**Df Residuals:**	528
Time:	19:58:23	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.0315	4.987e-02	0.632	0.527	[-6.623e-02, 0.129]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.1086	0.306	3.624	2.896e-04	[0.509, 1.708]
alpha[1]	0.2122	0.123	1.719	8.553e-02	[-2.968e-02, 0.454]

Covariance estimator: robust

In [ ]:

```python
# Estimate a AR(4)GARCH(1,1) model
ar4, ar4garch11 = compute_arima_garch_models(train_sample, p_arima=4, q_arima=0, p_garch=1, q_garch=1, vol="GAR
ar4garch11.summary()
```

Out[ ]:

Constant Mean - GARCH Model Results

Dep. Variable:	None	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	-724.595
Distribution:	Normal	**AIC:**	1457.19
Method:	Maximum Likelihood	**BIC:**	1474.27
		No. Observations:	529
Date:	Thu, Mar 02 2023	**Df Residuals:**	528
Time:	19:58:23	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	7.2359e-03	3.032e-02	0.239	0.811	[-5.219e-02,6.666e-02]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0426	1.743e-02	2.446	1.446e-02	[8.465e-03,7.677e-02]
alpha[1]	0.2892	9.709e-02	2.978	2.899e-03	[9.887e-02, 0.479]
beta[1]	0.7065	6.366e-02	11.098	1.280e-28	[0.582, 0.831]

Covariance estimator: robust

In [ ]:

```python
# Estimate a ARMA(4,4)ARCH(1) model
arma44, arma44arch1 = compute_arima_garch_models(train_sample, p_arima=4, q_arima=4, p_garch=1, q_garch=0, vol=
arma44arch1.summary()
```

Out[ ]:

Constant Mean - ARCH Model Results

Dep. Variable:	None	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-818.345
Distribution:	Normal	**AIC:**	1642.69
Method:	Maximum Likelihood	**BIC:**	1655.50
		No. Observations:	529
Date:	Thu, Mar 02 2023	**Df Residuals:**	528
Time:	19:58:24	**Df Model:**	1

Mean Model

	coef	**std err**	**t**	**P>|t|**	**95.0% Conf. Int.**
mu	0.0206	4.734e-02	0.436	0.663	[-7.214e-02, 0.113]

Volatility Model

	coef	**std err**	**t**	**P>|t|**	**95.0% Conf. Int.**
omega	1.1587	0.307	3.772	1.620e-04	[0.557, 1.761]
alpha[1]	0.1388	0.109	1.271	0.204	[-7.527e-02, 0.353]

Covariance estimator: robust

In [ ]:

```
# Estimate a ARMA(4,4)GARCH(1,1) model
arma44, arma44garch11 = compute_arima_garch_models(train_sample, p_arima=4, q_arima=4, p_garch=1, q_garch=1, vo
arma44garch11.summary()
```

Out[ ]:

Constant Mean - GARCH Model Results

Dep. Variable:	None	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	-731.481
Distribution:	Normal	**AIC:**	1470.96
Method:	Maximum Likelihood	**BIC:**	1488.05
		No. Observations:	529
Date:	Thu, Mar 02 2023	**Df Residuals:**	528
Time:	19:58:25	**Df Model:**	1

Mean Model

	coef	**std err**	**t**	**P>|t|**	**95.0% Conf. Int.**
mu	8.5877e-03	3.325e-02	0.258	0.796	[-5.657e-02,7.375e-02]

Volatility Model

	coef	**std err**	**t**	**P>|t|**	**95.0% Conf. Int.**
omega	0.0428	1.862e-02	2.301	2.141e-02	[6.345e-03,7.934e-02]
alpha[1]	0.2276	8.663e-02	2.627	8.603e-03	[5.782e-02, 0.397]
beta[1]	0.7538	6.148e-02	12.262	1.452e-34	[0.633, 0.874]

Covariance estimator: robust

4.c) Effectuez des tests de rapport de vraisemblance pour sélectionner un modèle entre AR(4), AR(4) ARCH(1), et AR(4) GARCH(1,1). Répétez cette procédure pour sélectionner un modèle entre ARMA(4,4), ARMA(4,4) ARCH(1), et ARMA(4,4) GARCH(1,1).

- H0: Le modèle complet AR(4)-ARCH(1) fit les données aussi bien que le modèle réduit AR(4). Par conséquent, on devrait choisir le modèle AR(4) car il est plus simple.
- H1: Le modèle complet AR(4)-ARCH(1) fit mieux les données que le modèle réduit AR(4). Par conséquent, on devrait choisir le modèle AR(4)-ARCH(1) car il est plus précis.

In [ ]:

```
compute_p_value_llr(ar4.llf,
                    ar4arch1.loglikelihood,
                    df=1)
```

Out[ ]:

```
0.0006890672977983893
```

Comme la p-value est inférieure à 0.05, on rejette l'hypothèse nulle et on choisit le modèle AR(4)-ARCH(1).

- H0: Le modèle complet AR(4)-GARCH(1,1) fit les données aussi bien que le modèle réduit AR(4)-ARCH(1). Par conséquent, on devrait choisir le modèle ARMA(4,4) car il est plus simple.
- H1: Le modèle complet AR(4)-GARCH(1,1) fit mieux les données que le modèle réduit AR(4)-ARCH(1). Par conséquent, on devrait choisir le modèle AR(4)-GARCH(1,1) car il est plus précis.

In [ ]:
```
compute_p_value_llr(ar4arch1.loglikelihood,
                    ar4garch11.loglikelihood,
                    df=1)
```

Out[ ]:
```
1.5560401305834263e-43
```

Comme la p-value est inférieure à 0.05, on rejette l'hypothèse nulle et on choisit le modèle AR(4)-GARCH(1,1).

- H0: Le modèle complet ARMA(4,4)-ARCH(1) fit les données aussi bien que le modèle réduit ARMA(4,4). Par conséquent, on devrait choisir le modèle ARMA(4,4) car il est plus simple.
- H1: Le modèle complet ARMA(4,4)-ARCH(1) fit mieux les données que le modèle réduit ARMA(4,4). Par conséquent, on devrait choisir le modèle ARMA(4,4)-ARCH(1) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma44.llf,
                    arma44arch1.loglikelihood,
                    df=1)
```

Out[ ]:
```
0.16216082087895325
```

Comme la p-value est supérieur à 0.05, on ne peut pas rejeter l'hypothèse nulle et on choisit le modèle ARMA(4,4).

- H0: Le modèle complet ARMA(4,4)-GARCH(1,1) fit les données aussi bien que le modèle réduit ARMA(4,4)-ARCH(1). Par conséquent, on devrait choisir le modèle ARMA(4,4)-ARCH(1) car il est plus simple.
- H1: Le modèle complet ARMA(4,4)-GARCH(1,1) fit mieux les données que le modèle réduit ARMA(4,4)-ARCH(1). Par conséquent, on devrait choisir le modèle ARMA(4,4)-GARCH(1,1) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma44arch1.loglikelihood,
                    arma44garch11.loglikelihood,
                    df=1)
```

Out[ ]:
```
1.1346354300030327e-39
```

Comme la p-value est inférieure à 0.05, on rejette l'hypothèse nulle et on choisit le modèle ARMA(4,4)-GARCH(1,1).

- H0: Le modèle complet ARMA(4,4)-GARCH(1,1) fit les données aussi bien que le modèle réduit ARMA(4,4). Par conséquent, on devrait choisir le modèle ARMA(4,4) car il est plus simple.
- H1: Le modèle complet ARMA(4,4)-GARCH(1,1) fit mieux les données que le modèle réduit ARMA(4,4). Par conséquent, on devrait choisir le modèle ARMA(4,4)-GARCH(1,1) car il est plus précis.

In [ ]:
```
compute_p_value_llr(arma44.llf,
                    arma44garch11.loglikelihood,
                    df=2)
```

Out[ ]:
```
7.096102374833897e-39
```

Comme la p-value est inférieure à 0.05, on rejette l'hypothèse nulle et on choisit le modèle ARMA(4,4)-GARCH(1,1).

On devrait donc choisir le modèle ARMA(4,4)-GARCH(1,1) et AR(4)-GARCH(1,1) d'après le critère du ratio de vraisemblance.

4.d) Effectuez un test de spécification sur les résidus pour AR(4) ARCH(1) et AR(4) GARCH(1,1). Quel est le meilleur modèle pour la variance conditionnelle ?

In [ ]:
```
def compute_specification_test(resid_square: pd.Series, h_t: pd.Series) -> pd.Series:
    """
    Compute the specification test for the conditional variance
    :param resid_square: the squared residuals
    :param h_t: the conditional variance
    :return: the specification test
    """
    return resid_square / h_t
```

In [ ]:
```
ar4arch1_resid_sq = ar4arch1.resid.reset_index(drop=True) ** 2
ar4garch11_resid_sq = ar4garch11.resid.reset_index(drop=True) ** 2

# Get ar4arch1 parameters
```

```
alpha_0 = ar4arch1.params.iloc[1]
alpha_1 = ar4arch1.params.iloc[2]

# Compute variance of AR(4) ARCH(1) model
ar4arch1_h = np.zeros(len(ar4arch1_resid_sq))
# Initial guess for conditionnal variance
ar4arch1_h[0] = alpha_0 / (1 - alpha_1)

for i in range(1, len(ar4arch1_resid_sq)):
    ar4arch1_h[i] = alpha_0 + alpha_1 * ar4arch1_resid_sq[i-1]
```

In [ ]:

```
# Get ar4garch11 parameters
alpha_0 = ar4garch11.params.iloc[1]
alpha_1 = ar4garch11.params.iloc[2]
beta_1 = ar4garch11.params.iloc[3]

# Compute variance of AR(4) GARCH(1,1) model
ar4garch11_h = np.zeros(len(ar4garch11_resid_sq))
# Initial guess
ar4garch11_h[0] = alpha_0 / (1 - alpha_1 - beta_1)

for i in range(1, len(ar4garch11_resid_sq)):
    ar4garch11_h[i] = alpha_0 + alpha_1 * ar4garch11_resid_sq[i-1] + beta_1 * ar4garch11_h[i-1]
```

In [ ]:

```
# Compute specification test
ar4arch1_spec_test = compute_specification_test(ar4arch1_resid_sq, ar4arch1_h)
ar4garch11_spec_test = compute_specification_test(ar4garch11_resid_sq, ar4garch11_h)
```

In [ ]:

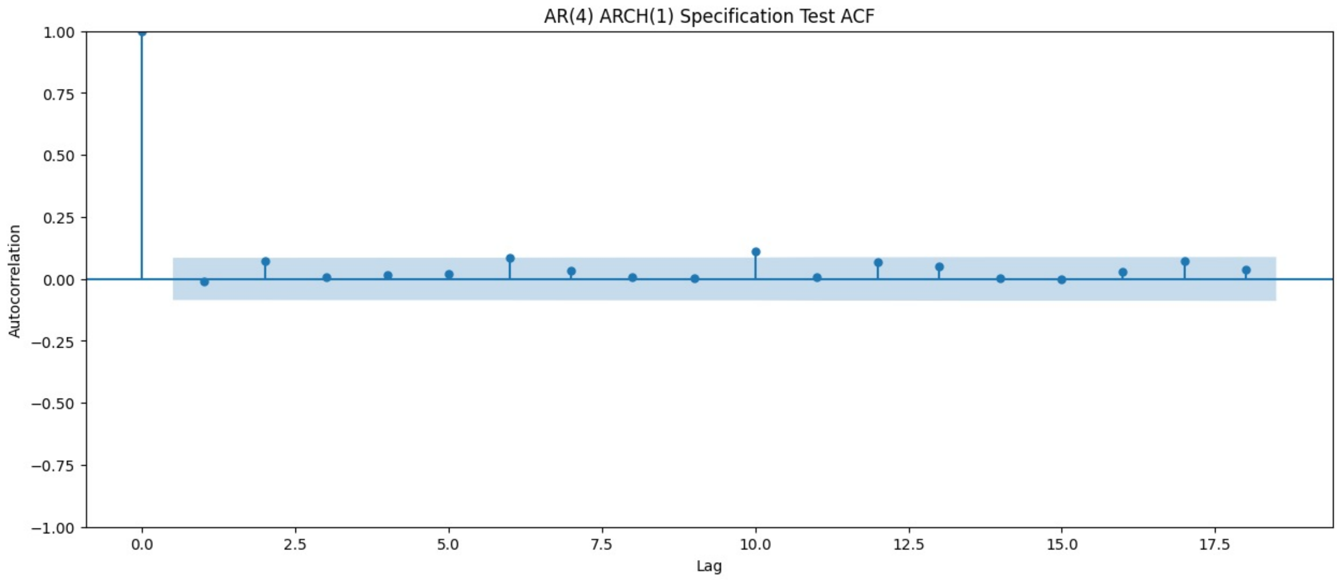
```
# Use ljung box test on the specification test
ar4arch1_ljung_box = sm.stats.acorr_ljungbox(ar4arch1_spec_test, lags=18)

for p in p_values:
    ar4arch1_ljung_box[f"p < {p}"] = ar4arch1_ljung_box["lb_pvalue"] < p

ar4arch1_ljung_box
```

Out[ ]:

	lb_stat	lb_pvalue	p < 0.1	p < 0.05	p < 0.01
1	0.065457	0.798069	False	False	False
2	2.704905	0.258605	False	False	False
3	2.723726	0.436210	False	False	False
4	2.879816	0.578135	False	False	False
5	3.110696	0.682924	False	False	False
6	7.111502	0.310659	False	False	False
7	7.689660	0.360753	False	False	False
8	7.731774	0.460100	False	False	False
9	7.736727	0.560880	False	False	False
10	14.642018	0.145668	False	False	False
11	14.669034	0.198156	False	False	False
12	17.144327	0.144248	False	False	False
13	18.542049	0.138001	False	False	False
14	18.548354	0.182935	False	False	False
15	18.550136	0.234848	False	False	False
16	18.948364	0.271350	False	False	False
17	21.670851	0.197711	False	False	False
18	22.493765	0.210799	False	False	False

In [ ]:

```
# Plot the acf of the specification test
fig, axe_label = plt.subplots(figsize=(15,6))
sm.graphics.tsa.plot_acf(ar4arch1_spec_test, lags=18, title="AR(4) ARCH(1) Specification Test ACF", ax= axe_lab
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



In [ ]:

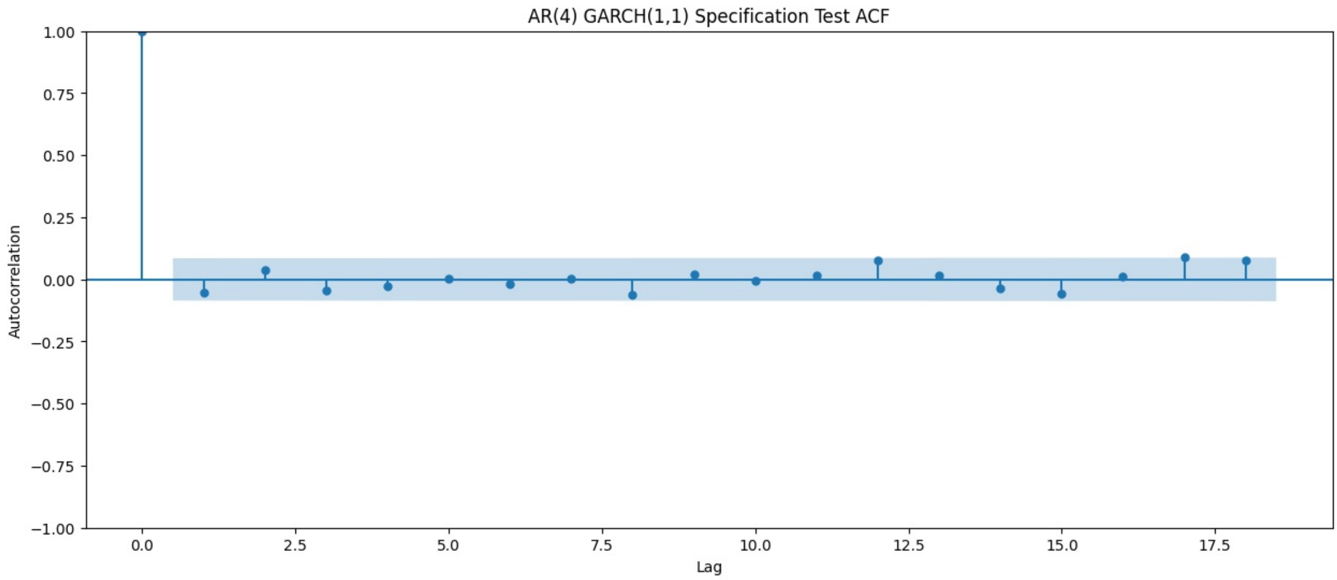
```
ar4garch11_ljung_box = sm.stats.acorr_ljungbox(ar4garch11_spec_test, lags=18)

for p in p_values:
    ar4garch11_ljung_box[f"p < {p}"] = ar4garch11_ljung_box["lb_pvalue"] < p

ar4garch11_ljung_box
```

Out[ ]:

	lb_stat	lb_pvalue	p < 0.1	p < 0.05	p < 0.01
1	1.540613	0.214527	False	False	False
2	2.225624	0.328634	False	False	False
3	3.388078	0.335571	False	False	False
4	3.791518	0.434955	False	False	False
5	3.794509	0.579365	False	False	False
6	3.985054	0.678699	False	False	False
7	3.986422	0.781340	False	False	False
8	6.101311	0.635885	False	False	False
9	6.364119	0.702995	False	False	False
10	6.370918	0.783197	False	False	False
11	6.546744	0.834513	False	False	False
12	9.903270	0.624446	False	False	False
13	10.049280	0.689900	False	False	False
14	10.743098	0.706089	False	False	False
15	12.467224	0.643375	False	False	False
16	12.531517	0.706653	False	False	False
17	17.113403	0.446709	False	False	False
18	20.291238	0.316664	False	False	False

In [ ]:

```
# Plot the acf of the specification test
fig, axe_label = plt.subplots(figsize=(15,6))
sm.graphics.tsa.plot_acf(ar4garch11_spec_test, lags=18, title="AR(4) GARCH(1,1) Specification Test ACF", ax= ax
axe_label.set_xlabel("Lag")
axe_label.set_ylabel("Autocorrelation")
plt.show()
```



Pour commencer, on sait que si les modèles GARCH(1,1) et ARCH(1) sont des spécifications adéquates pour la variance conditionnelle, on devrait s'attendre à ce que le terme $u_{2t}h_t$ présentent de l'autocorrélation. À partir des graphiques d'autocorrélation et du test de Ljung-Box, on remarque que l'on ne peut pas rejetter l'hypothèse nulle au seuil de 5% pour les deux modèles AR(4) ARCH(1) et AR(4) GARCH(1,1). On peut donc conclure que les deux modèles sont des spécifications adéquates pour la variance conditionnelle. On remarque tout de même à partir des graphiques d'autocorrélations que le modèle AR(4) ARCH(1) présente une autocorrélation plus importante que le modèle AR(4) GARCH(1,1)pour certains lags (lag 6 et 10 par exemple). On pourrait donc conclure que le modèle AR(4) GARCH(1,1) est une meilleure spécification que le modèle AR(4) ARCH(1) pour la variance conditionnelle.

4.e) Pour ces deux nouveaux modèles et les deux modèles étudiés dans la partie 3, tracez les fonctions de réponse impulsionnelle du second cas décrit dans la partie 3. Dans quelle mesure sont-elles différentes ?

In [ ]:
```
def compute_three_impulse_mixed(unit_impulse, sigma = 1):
    """Compute the impulse response for 3 periods

    Args:
        unit_impulse: A (1,n) array showing the impulse response on n periods
        sigma: The size of the input impulse

    Output:
        total_impulse: The Impulse on n periods given a choc in the first three periods
    """
    first_impulse = pd.Series(unit_impulse.transpose().reshape(-1)) * sigma
    second_impulse = first_impulse
    second_impulse = pd.concat([pd.Series([0]), second_impulse], ignore_index=True)
    second_impulse = second_impulse[0:-1]
    third_impulse = first_impulse
    third_impulse = pd.concat([pd.Series([0, 0]), third_impulse], ignore_index=True)
    third_impulse = third_impulse[0:-2]
    total_impulse = first_impulse + second_impulse + third_impulse
    return total_impulse
```

In [ ]:
```
#Start the matlab engine
eng = matlab.engine.start_matlab()
```

In [ ]:
```
#Transform the variable into a suitable matlab object
mat_USDCAD = matlab.double(list(df_stoch["stoch_USDCAD"]))
vec_USDCAD = eng.transpose(mat_USDCAD)
```

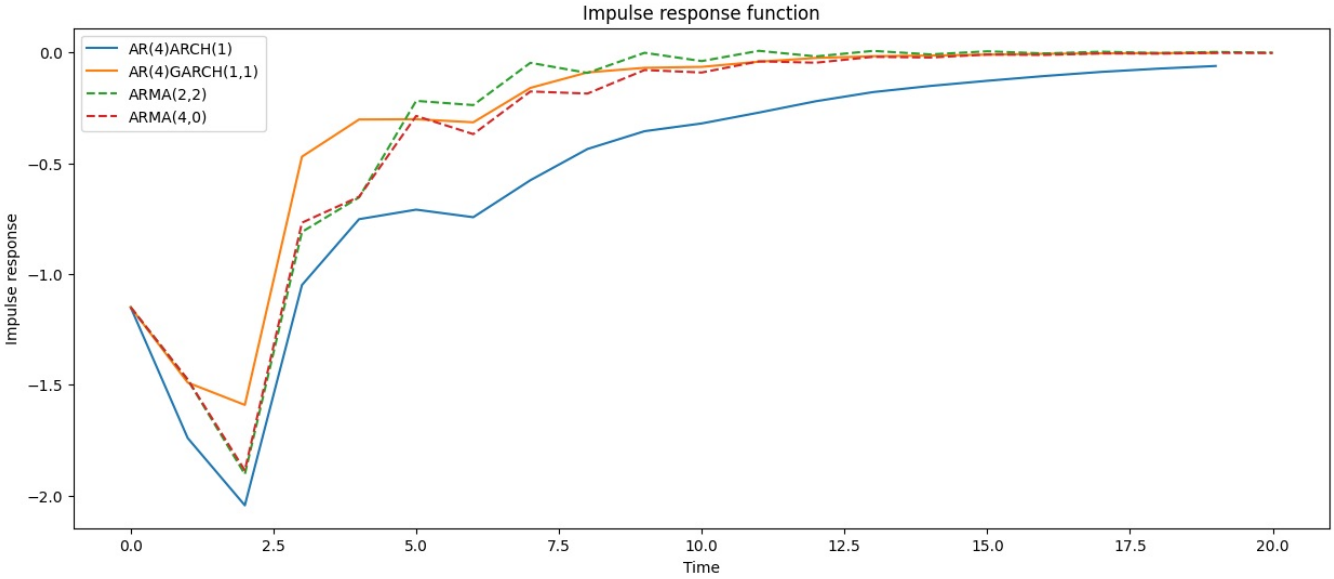
In [ ]:
```
#Estimate the models
ar4arch1_m = eng.arima('ARLags',matlab.double([1,4]),'Variance',eng.garch(0.0,1.0));
ar4arch1_matlab = eng.estimate(ar4arch1_m,vec_USDCAD)

ar4garch11_m = eng.arima('ARLags',matlab.double([1,4]),'Variance',eng.garch(1.0,1.0));
ar4garch11_matlab = eng.estimate(ar4garch11_m,vec_USDCAD)
```

In [ ]:
```
#Calculate the impulse response for 1 shock in the initial period
steps = 20.0
impulse_ar4arch1_matlab = np.asarray(eng.impulse(ar4arch1_matlab, steps)).transpose()
impulse_ar4garch11_matlab = np.asarray(eng.impulse(ar4garch11_matlab, steps)).transpose()
```

In [ ]:
```
#Calculate the impulse response for 3 shocks in the earliest periods
impulse_res_ar4arch1 = compute_three_impulse_mixed(impulse_ar4arch1_matlab, sigma = -1.15)
impulse_res_ar4garch11 = compute_three_impulse_mixed(impulse_ar4garch11_matlab, sigma = -1.15)
```

In [ ]:
```
#Plot the impulse response
```

```
plt.plot(impulse_res_ar4arch1, label="AR(4)ARCH(1)")
plt.plot(impulse_res_ar4garch11, label="AR(4)GARCH(1,1)")
plt.plot(arma22_total_impulse, label="ARMA(2,2)", linestyle="--")
plt.plot(arma40_total_impulse, label="ARMA(4,0)", linestyle="--")
plt.legend()
plt.title("Impulse response function")
plt.xlabel("Time")
plt.ylabel("Impulse response")
plt.show()
```



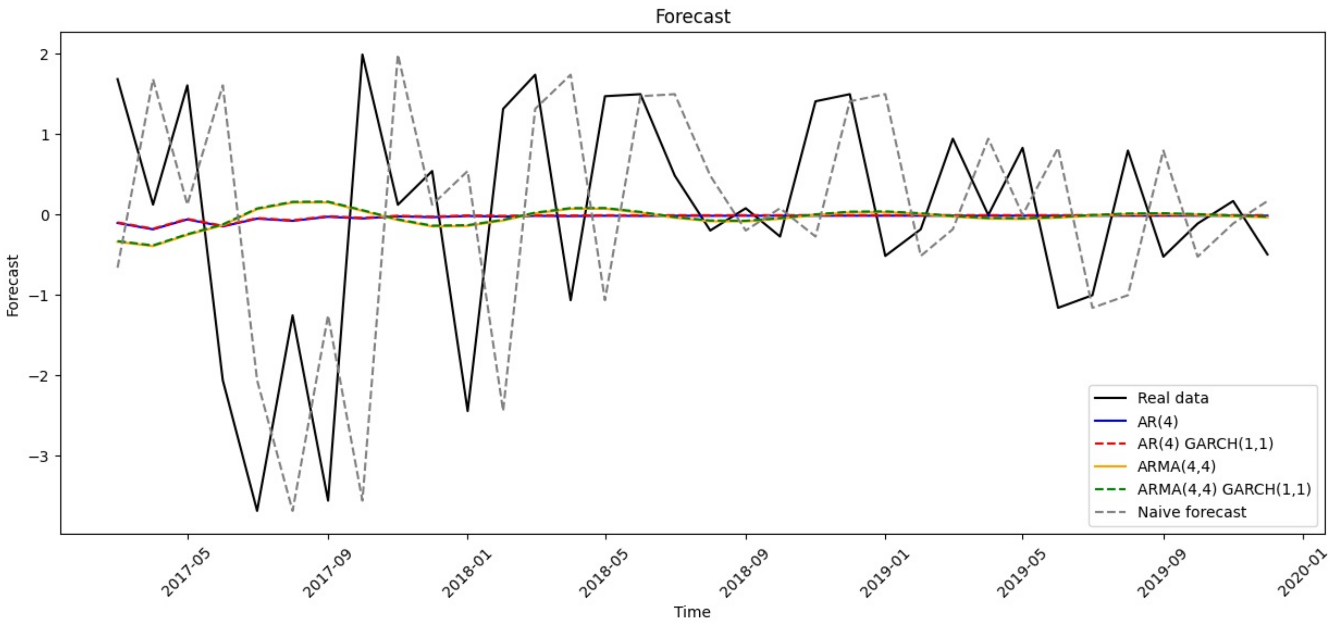
Interpretation

Comparativement aux models ARMA(2,2) et AR(4), le AR(4)GARCH(1,1) subit un impact moins severe des chocs que les deux modeles precedents. Le AR(4)ARCH(1) est cependant plus sensible aux chocs introduit dans le modele. Alors que le AR(4)GARCH(1,1) converge rapidement vers ARMA(2,2) et AR(4), le AR(4)ARCH(1) conserve un sensibilite plus grande au chocs sur un grand nombre de periodes. Cette difference entre la version qui combine le ARCH et celle qui combine le GARCH s'explique par l'introduction de la moyenne mobile sur le terme d'erreur dans le GARCH. Cela rend l'effet du choc plus parcimonieux que pour le ARCH.

4.f) Comme dans la partie 3, tracez les prévisions à un pas d'avance $E_t(y_{t+1})$ pour la période couverte par l'échantillon de validation (holdout sample). Comparez-les avec celles obtenues dans la partie 3.

In [ ]:
```
# Get AR(4) GARCH(1,1) forecast
forecast_ar4 = ar4.forecast(steps=treshold).reset_index(drop=True)
forecast_garch11 = ar4garch11.forecast(horizon=treshold).mean['h.01'].iloc[-1]
forecast_ar4garch11 = forecast_ar4 + forecast_garch11

# Get ARMA(4,4) GARCH(1,1) forecast
forecast_arma_44 = arma44.forecast(steps=treshold).reset_index(drop=True)
forecast_garch11 = arma44garch11.forecast(horizon=treshold).mean['h.01'].iloc[-1]
forecast_arma44garch11 = forecast_arma_44 + forecast_garch11

# Change index to date for plotting
forecast_arma_44.index = df["date"].iloc[-treshold:]
forecast_ar4garch11.index = df["date"].iloc[-treshold:]
forecast_arma44garch11.index = df["date"].iloc[-treshold:]
```

In [ ]:
```
plt.plot(val_sample, label="Real data", color="black")
plt.plot(forecast_arma40, label="AR(4)", color="blue")
plt.plot(forecast_ar4garch11, label="AR(4) GARCH(1,1)", color="red", linestyle="--")
plt.plot(forecast_arma_44, label="ARMA(4,4)", color="orange")
plt.plot(forecast_arma44garch11, label="ARMA(4,4) GARCH(1,1)", color="green", linestyle="--")
plt.plot(forecast_naive, label="Naive forecast", color="grey", linestyle="--")
plt.legend()
plt.title("Forecast")
plt.xlabel("Time")
# rotate xticks by 45 degrees
plt.xticks(rotation=45)
plt.ylabel("Forecast")
plt.show()
```



4.g) Comparez les erreurs quadratiques moyennes de la prévision à un pas (pour la série transformée !) pour les deux modèles sélectionnés dans la partie 3, les deux nouveaux modèles, la prévision naïve $E_t(y_{t+1}) = y_t$ et la prévision paresseuse $E_t(y_{t+1}) = 0$ pour tous les t. Cette dernière suppose que la série en niveaux ne changera pas. Que pouvez-vous conclure ? La modélisation de la covariance conditionnelle était-elle utile ?

In [ ]:

```
mse_ar4garch11 = compute_mse(val_sample, forecast_ar4garch11)
mse_arma44garch11 = compute_mse(val_sample, forecast_arma44garch11)
mse_lazy = compute_mse(val_sample, [0]*34)

mse_df.insert(1, "AR(4) GARCH(1,1)", mse_ar4garch11, True)
mse_df.insert(3, "ARMA(4,4) GARCH(1,1)", mse_arma44garch11, True)
mse_df["Lazy forecast"] = mse_lazy
mse_df
```

Out[ ]:

	AR(4)	AR(4) GARCH(1,1)	ARMA(4,4)	ARMA(4,4) GARCH(1,1)	Naive forecast	Lazy forecast
MSE	1.986487	1.986162	2.090525	2.089992	3.832131	1.996051

On observe que tous nos modèles parviennent à battre en terme de MSE le modèle naïf. Cependant, seul le modèle AR(4) GARCH(1,1) et AR(4) parviennent à battre le modèle paresseux. Cela signifie que la modélisation de la covariance conditionnelle est utile pour prédire la volatilité de la série, mais seulement dans le cas ou le modèle plus simple AR(4)-GARCH(1,1) a été utilisé. De plus, la modélisation de la variance conditionnelle dans le cas du modèle AR(4)-GARCH(1,1) présente un très léger avantage sur le modèle AR(4) en terme de MSE (amélioration du MSE de 0.000325)

Processing math: 100%