PROJET FINAL

Prévision du cours des actions des fiducies de placement immobilier : une comparaison entre divers modèles

Introduction

Nous souhaitons estimer un modèle ARIMA en vue de prédire la valeur mensuelle du cours de l'indice des fiducies de placement immobilier (FPI).

Au final, deux modèles ARIMA ont été estimés en utilisant diverses méthodologies :

- Box-Jenkins,
- automatique.

Ensuite, la technique de "rolling forecast" a été appliquée aux deux modèles.

Les modèles estimés ont été comparés sur leur capacité à faire des prévisions au regard de certains critères communément utilisés dans la littérature. Il s’agit notamment des critères Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE).

Les données sollicitées sont les cours des indices des FPI.

In [1]:
```
# !pip install matplotlib
# !pip install statsmodels
# !pip install numpy scipy patsy
# !pip install statsforecast
# !pip install pmdarima
```

In [2]:
```
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
import urllib
import yfinance as yf
import datetime
import matplotlib.pyplot as plt
import numpy as np
import math


from datetime import date

from pandas import DataFrame
from matplotlib import pyplot
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf, plot_predict
```

```
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.api import VAR

from numpy import log
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.vector_ar.var_model import VAR
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_
from statsmodels.stats.stattools import durbin_watson

from pmdarima.arima import auto_arima
from pmdarima.utils import diff_inv

### Just to remove warnings to prettify the notebook.
import warnings
warnings.filterwarnings("ignore")
```

1. Collecte des données FPI (Fiducie de placements immobiliers)

Le choix de l'année 1993 pour le début des séries n'est pas le fait du hasard car c'est l'année durant laquelle une importante loi (Omnibus Budget Reconciliation Act of 1993) est entrée en vigueur. Comme bien d'autres auteurs, Chan et al. (2003, p. 30) considèrent que cet évènement est à l'origine de la croissance importante que connaîtra le secteur parce que cette loi permettait désormais aux fonds de pension d'investir massivement dans les FPI. Cette période à partir de 1993 est reconnue comme étant l'Ère Moderne des FPI. Elle coïncide avec le fait identifié par certains auteurs selon lequel les actions FPI se comportaient comme les actions ordinaires.

1.1 Données prix des indices pour les FPI (All REITs)

In [3]:
```
dls = "https://www.reit.com/sites/default/files/returns/MonthlyHistoricalReturns.xl
urllib.request.urlretrieve(dls, "MonthlyHistoricalReturns.xls")
```

Out[3]:
```
('MonthlyHistoricalReturns.xls', <http.client.HTTPMessage at 0x1713e8c3ef0>)
```

Entête du fichier Excel contenant les données des FPI

FTSE Nareit U.S. Real Estate Index Series
Monthly Index Values and Returns
December 1971 - September 2024
(Returns and dividend yield in percent)

	All REITs						Composite					
	Total		Price		Income	Dividend	Total		Price		Income	Dividend
Date	Return	Index	Return	Index	Return	Yield	Return	Index	Return	Index	Return	Yield
déc-71		100,00		100,00				100,00		100,00		
janv-72	1,22	101,22	0,33	100,33	0,89	6,51	1,22	101,22	0,33	100,33	0,89	6,51
févr-72	0,95	102,18	0,92	101,25	0,03	6,39	0,95	102,18	0,92	101,25	0,03	6,39
mars-72	0,25	102,44	-0,44	100,81	0,69	6,32	0,25	102,44	-0,44	100,81	0,69	6,32
avr-72	0,25	102,70	-0,39	100,41	0,65	6,52	0,25	102,70	-0,39	100,41	0,65	6,52
mai-72	-1,51	101,15	-1,78	98,63	0,27	7,10	-1,51	101,15	-1,78	98,63	0,27	7,10
juin-72	0,01	101,16	-0,68	97,96	0,68	7,28	0,01	101,16	-0,68	97,96	0,68	7,28
juil-72	2,82	104,02	1,86	99,78	0,97	7,27	2,82	104,02	1,86	99,78	0,97	7,27
août-72	-1,13	102,84	-1,26	98,52	0,13	7,33	-1,13	102,84	-1,26	98,52	0,13	7,33

	Real Estate 50™						All Equity REITs					
	Total		Price		Income	Dividend	Total		Price		Income	Dividend
Date	Return	Index	Return	Index	Return	Yield	Return	Index	Return	Index	Return	Yield
déc-71								100,00		100,00		
janv-72							0,00	100,00	-1,35	98,65	1,35	6,13
févr-72							1,74	101,74	1,74	100,37	0,00	5,74
mars-72							-0,32	101,42	-0,74	99,63	0,42	5,92
avr-72							3,84	105,31	3,24	102,85	0,60	5,72
mai-72							-9,37	95,44	-9,75	92,83	0,38	6,53
juin-72							9,98	104,96	9,61	101,75	0,37	6,20
juil-72							5,80	111,05	4,53	106,36	1,26	6,02
août-72							-1,34	109,56	-1,45	104,81	0,11	6,16

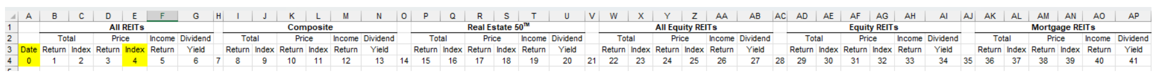
	Equity REITs						Mortgage REITs					
	Total		Price		Income	Dividend	Total		Price		Income	Dividend
Date	Return	Index	Return	Index	Return	Yield	Return	Index	Return	Index	Return	Yield
déc-71		100,00		100,00				100,00		100,00		
janv-72	0,00	100,00	-1,35	98,65	1,35	6,13	1,24	101,24	0,87	100,87	0,38	6,10
févr-72	1,74	101,74	1,74	100,37	0,00	5,74	0,52	101,76	0,52	101,39	0,00	6,07
mars-72	-0,32	101,42	-0,74	99,63	0,42	5,92	-1,59	100,15	-2,74	98,61	1,16	6,02
avr-72	3,84	105,31	3,24	102,85	0,60	5,72	0,37	100,52	-0,10	98,51	0,47	6,45
mai-72	-9,37	95,44	-9,75	92,83	0,38	6,53	1,36	101,88	1,22	99,71	0,14	7,49
juin-72	9,98	104,96	9,61	101,75	0,37	6,20	-3,39	98,43	-4,56	95,16	1,17	8,00
juil-72	5,80	111,05	4,53	106,36	1,26	6,02	1,42	99,83	0,93	96,04	0,49	8,06
août-72	-1,34	109,56	-1,45	104,81	0,11	6,16	-1,74	98,09	-1,93	94,19	0,19	8,08

In [4]:
```
Reits_data = pd.read_excel('MonthlyHistoricalReturns.xls', sheet_name='Index Data',
# print(Reits_data)
Reits_data.head()
```

Out[4]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnai
0	1971-12-31	NaN	100.000000	NaN	100.000000	NaN	NaN	
1	1972-01-31	1.220353	101.220353	0.326895	100.326895	0.893458	6.51	
2	1972-02-29	0.949680	102.181621	0.919890	101.249791	0.029790	6.39	
3	1972-03-31	0.252432	102.439561	-0.435933	100.808410	0.688366	6.32	
4	1972-04-30	0.254852	102.700630	-0.394636	100.410583	0.649488	6.52	

5 rows × 42 columns



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		All REITs							Composite						
2		Total		Price		Income	Dividend		Total		Price		Income	Dividend	
3	Date	Return	Index	Return	Index	Return	Yield		Return	Index	Return	Index	Return	Yield	
4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	Real Estate 50™							All Equity REITs						
2	Total		Price		Income	Dividend		Total		Price		Income	Dividend	
3	Return	Index	Return	Index	Return	Yield		Return	Index	Return	Index	Return	Yield	
4	15	16	17	18	19	20	21	22	23	24	25	26	27	28

	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP
1	Equity REITs							Mortgage REITs					
2	Total		Price		Income	Dividend		Total		Price		Income	Dividend
3	Return	Index	Return	Index	Return	Yield		Return	Index	Return	Index	Return	Yield
4	29	30	31	32	33	34	35	36	37	38	39	40	41

In [5]:
```python
Reits_data = Reits_data.iloc[:, [0, 4]]
```

In [6]:
```python
Reits_data.head()
```

Out[6]:

	Unnamed: 0	Unnamed: 4
0	1971-12-31	100.000000
1	1972-01-31	100.326895
2	1972-02-29	101.249791
3	1972-03-31	100.808410
4	1972-04-30	100.410583

In [7]:
```python
Reits_data.columns = ["Date", "All_REITs"]
```

In [8]:
```python
Reits_data.head()
```

Out[8]:

	Date	All_REITs
0	1971-12-31	100.000000
1	1972-01-31	100.326895
2	1972-02-29	101.249791
3	1972-03-31	100.808410
4	1972-04-30	100.410583

In [9]:
```
# Changer nom colonne
All_REIT = Reits_data.rename(columns={'All_REITs': 'Indice'})

# Ajouter 10 jours
All_REIT['Date'] = All_REIT['Date'] + datetime.timedelta(10)

# Recréer la date
All_REIT['Date'] = pd.to_datetime(dict(year = All_REIT['Date'].dt.year,
                                       month = All_REIT['Date'].dt.month,
                                        day = 1))

All_REIT.head()
```

Out[9]:

	Date	Indice
0	1972-01-01	100.000000
1	1972-02-01	100.326895
2	1972-03-01	101.249791
3	1972-04-01	100.808410
4	1972-05-01	100.410583

In [10]:
```
# On prend les données des FPI à partir de 1993-01-01
All_REIT = All_REIT.loc[All_REIT['Date'] >= '1993-01-01']
```

In [11]:
```
# Transformation de la colonne date en index comme pour la série S&P500
All_REIT.set_index(['Date'], drop=True, append = False, inplace = True)

All_REIT.head()
```

Out[11]:

Date	Indice
1993-01-01	74.775965
1993-02-01	79.113240
1993-03-01	82.367563
1993-04-01	87.581381
1993-05-01	83.255836

In [12]:

```
All_REIT.describe()
```

Out[12]:

	Indice
count	383.000000
mean	148.189372
std	53.083268
min	61.430000
25%	94.871745
50%	148.331115
75%	196.320850
max	284.613800

In [13]:

```
All_REIT.isna().sum()
```

Out[13]:

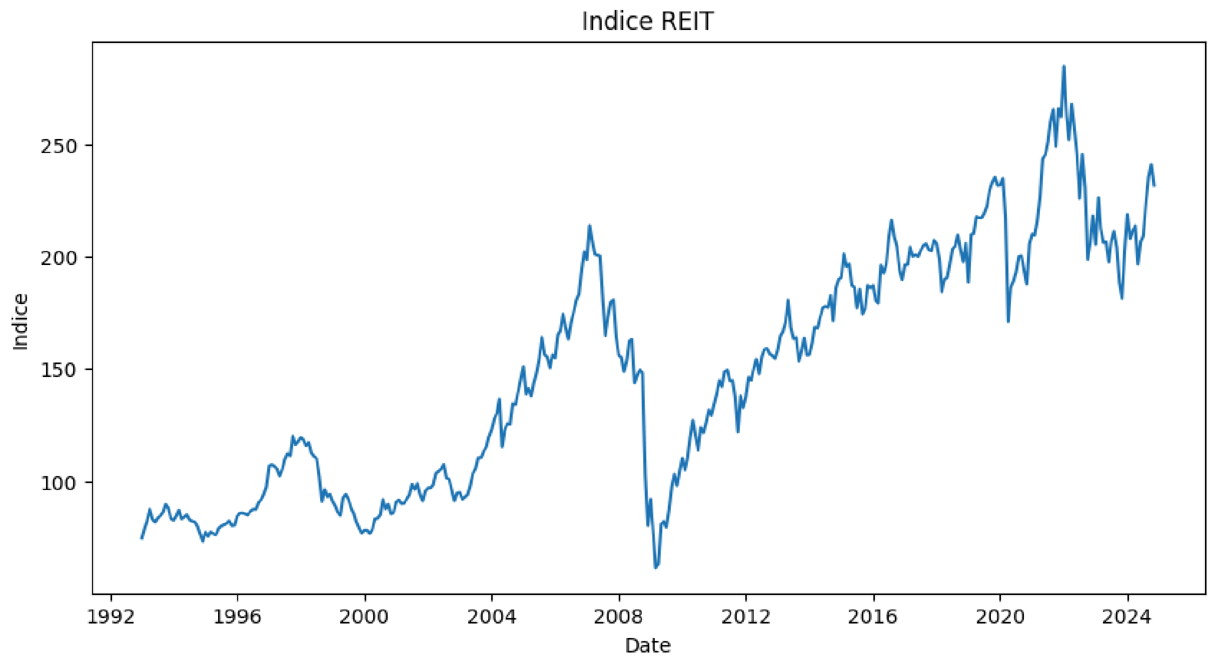
```
Indice    0
dtype: int64
```

In [14]:

```
# Canevas
fig, axs = plt.subplots(figsize=(10, 5))

plt.plot(All_REIT)
plt.title('Indice REIT')
plt.xlabel('Date')
plt.ylabel('Indice')
```

Out[14]:

```
Text(0, 0.5, 'Indice')
```

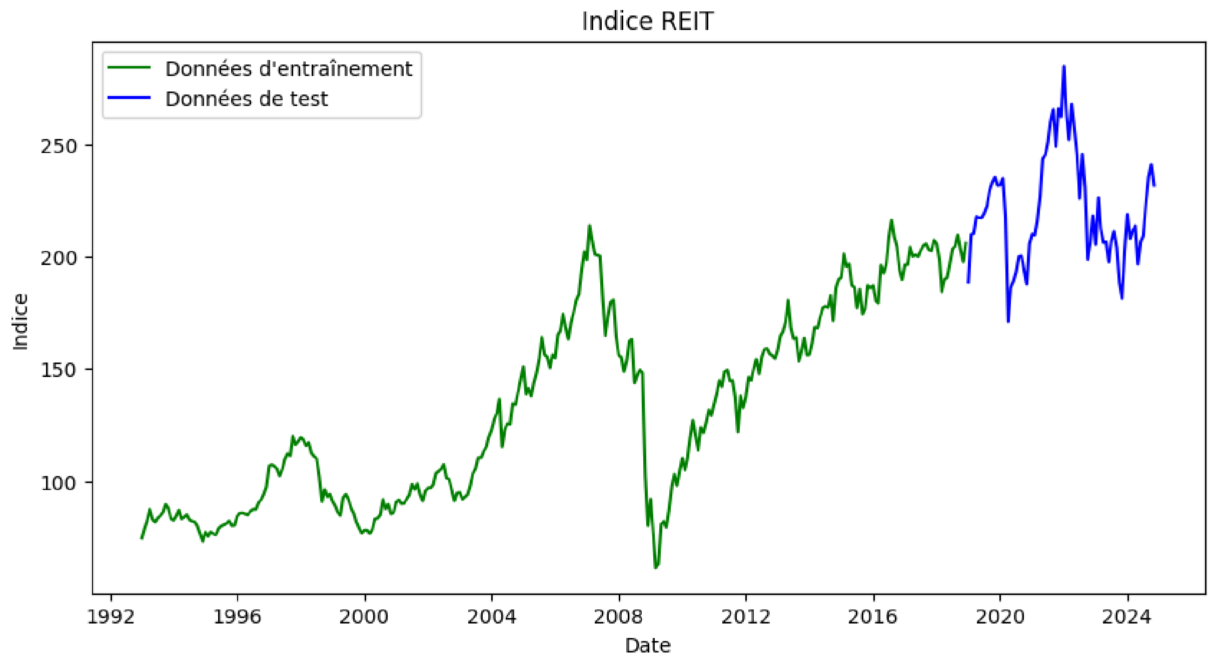


1.2 Séparation en données d'entraînement (de 1993-01-01 à 2018-12-01) et de test (de 2019-01-01 à 2024-10-01)

In [15]:
```
# Séparation en données d'entraînement et de test pour la série des FPI
All_REIT_train = All_REIT[:'2018-12-01']

All_REIT_test = All_REIT['2019-01-01':]
```

In [16]:
```
# Canevas
fig, axs = plt.subplots(figsize=(10, 5))

plt.plot(All_REIT_train, 'green', label='Données d\'entraînement')
plt.plot(All_REIT_test, 'blue', label='Données de test')
plt.title('Indice REIT')
plt.xlabel('Date')
plt.ylabel('Indice')
plt.legend()

plt.legend()
```

Out[16]:
```
<matplotlib.legend.Legend at 0x1713ebd60c0>
```



2. Modélisation ARIMA (données FPI)

2.1 Méthode Box-Jenkins

Elle consiste en trois étape:

1. La première étape consiste à identifier le modèle ARIMA(p,d,q) qui pourrait engendrer la série . Elle consiste, d'abord en transformer la série afin de la rendre stationnaire (le nombre de différenciations détermine l'ordre d'intégration: d), et ensuite d'identifier le modèle ARMA(p,q) de la série transformée avec l'aide du corrélogramme et du corrélogramme partiel. Le graphique des coefficients d'autocorrélation (corrélogramme) et d'autocorrélation partielle (corrélogramme partiel) donnent information sur l'ordre du modèle

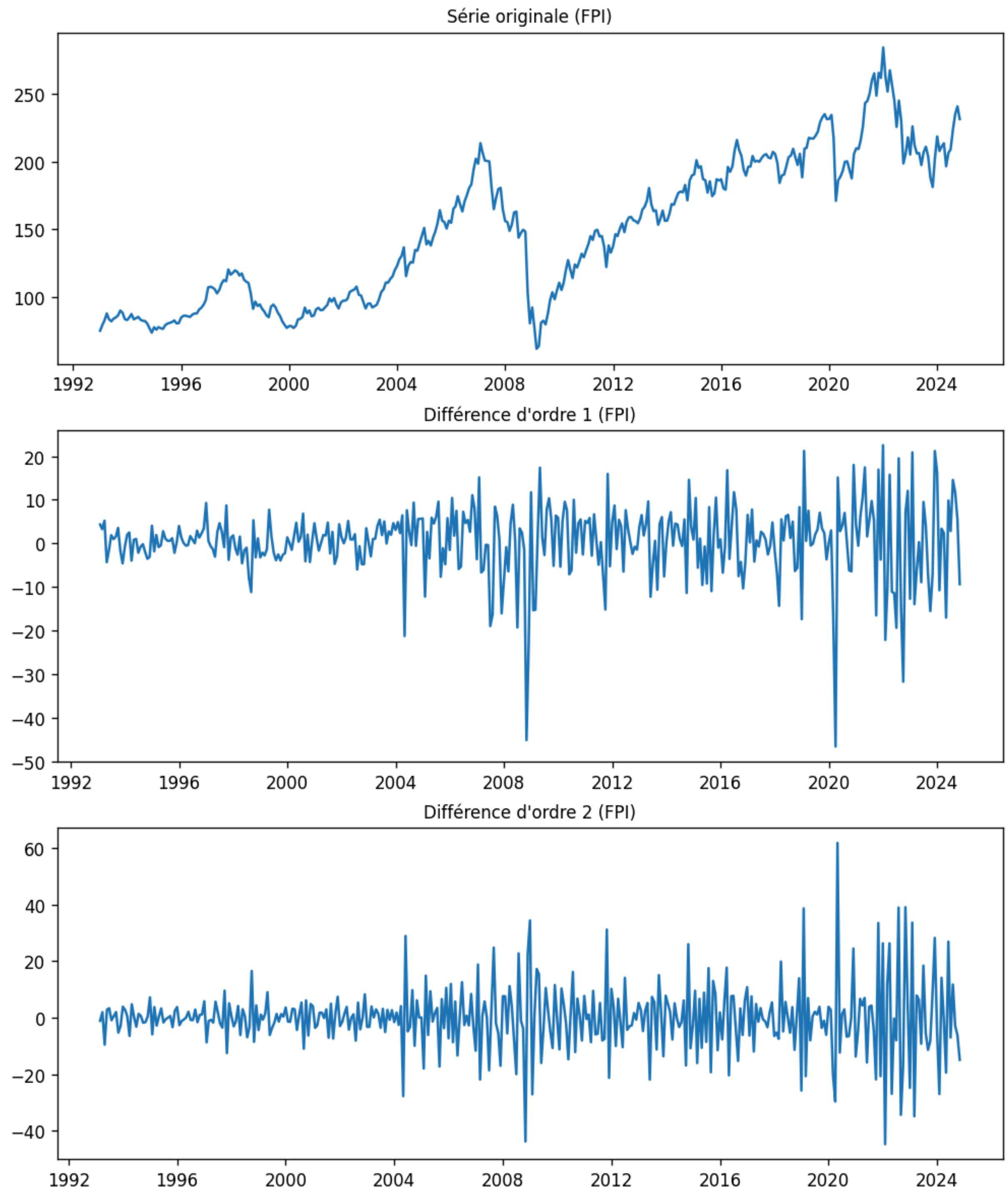
2. La deuxième étape consiste à estimer le modèle ARIMA en utilisant une méthode non linéaire (moindres carrés non-linéaires ou maximum de vraisemblance). Ces méthodes sont appliquées en utilisant les degrés p, d et q trouvés dans l'étape d'identification.

3. La troisième étape consiste à vérifier si le modèle estimé reproduit le modèle qui a engendré les données. Pour cela les résidus obtenus à partir du modèle estimé sont utilisés pour vérifier s'ils se comportent comme des erreurs bruit blanc.

2.1.1 Identification du modèle

a)Détermination de l'ordre de différenciation (d) : Test de Dicker-Fuller

In [17]:
```
#

plt.rcParams.update({'figure.figsize':(10, 12), 'figure.dpi':120})

# Série originale
fig, axes = plt.subplots(3)
axes[0].plot(All_REIT); axes[0].set_title('Série originale (FPI)', fontsize = 10)

# Première différence
axes[1].plot(All_REIT.diff()); axes[1].set_title('Différence d\'ordre 1 (FPI)', fon

# Deuxième différence
axes[2].plot(All_REIT.diff().diff()); axes[2].set_title('Différence d\'ordre 2 (FPI

plt.show()

print('ADF Statistic pour la série originale (FPI)')
result = adfuller(All_REIT.dropna())
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Valeurs critiques:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
print('\n')

print('ADF Statistic pour la différence d\'ordre 1 (FPI)')
result = adfuller(All_REIT.diff().dropna())
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Valeurs critiques:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
print('\n')

print('ADF Statistic pour la différence d\'ordre 2 (FPI)')
result = adfuller(All_REIT.diff().diff().dropna())
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Valeurs critiques:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```


Série originale (FPI)
250
200
150
100
1992
1996
2000
2004
2008
2012
2016
2020
2024
Différence d'ordre 1 (FPI)
20
10
0
−10
−20
−30
−40
−50
1992
1996
2000
2004
2008
2012
2016
2020
2024
Différence d'ordre 2 (FPI)
60
40
20
0
−20
−40
1992
1996
2000
2004
2008
2012
2016
2020
2024

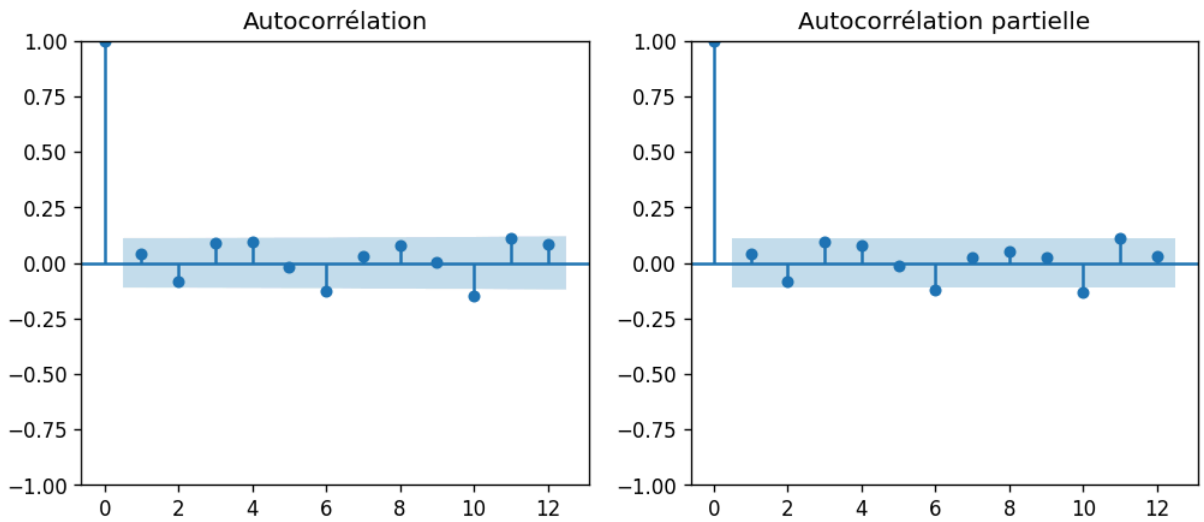
```
ADF Statistic pour la série originale (FPI)
ADF Statistic: -1.049099
p-value: 0.734938
Valeurs critiques:
	1%: -3.448
	5%: -2.869
	10%: -2.571


ADF Statistic pour la différence d'ordre 1 (FPI)
ADF Statistic: -6.181121
p-value: 0.000000
Valeurs critiques:
	1%: -3.448
	5%: -2.869
	10%: -2.571


ADF Statistic pour la différence d'ordre 2 (FPI)
ADF Statistic: -8.360313
p-value: 0.000000
Valeurs critiques:
	1%: -3.448
	5%: -2.869
	10%: -2.571
```

On peut voir avec les figures que la série des FPI semble staionnaire dès la première différence. Cela est confirmé par les tests de Dickey-Fuller. En effet, les valeurs de la statistique de Dickey-Fuller sont inférieures à toutes les valeurs critiques à tous les seuils (1%, 5% et 10%) pour les première et deuxième différences.

On conclut donc que les première et deuxième différence de la série des FPI sont stationnaires.

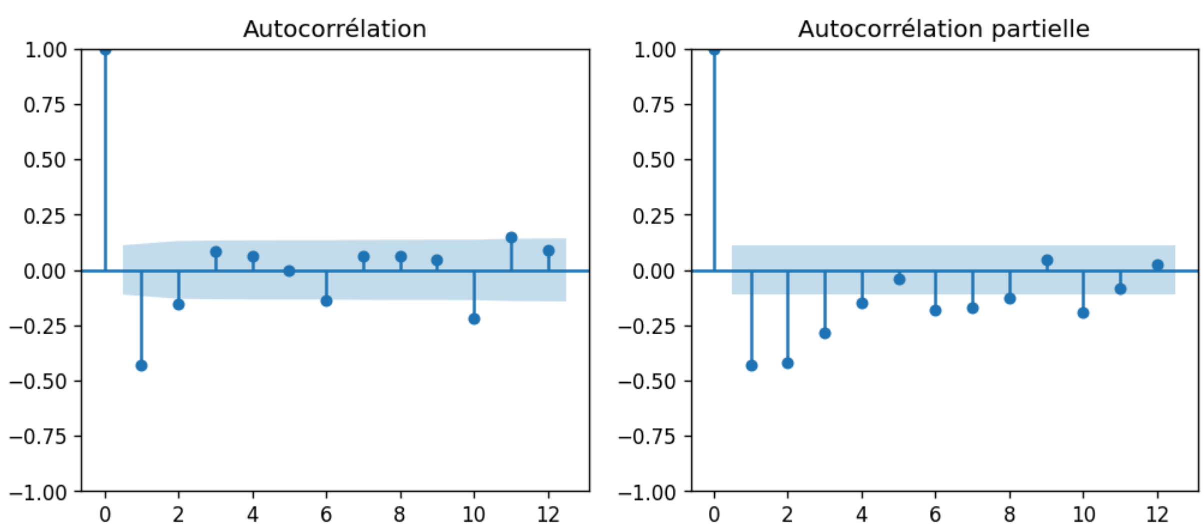
b) Détermination des termes p et q : Corrélogramme des acf et pacf

In [18]:
```
# Corrélogramme acf et pacf première différence

f = plt.figure(figsize=(10,4))

ax1 = f.add_subplot(121)
plot_acf(All_REIT_train.diff().dropna(), ax = ax1, title = 'Autocorrélation', lags=

ax2 = f.add_subplot(122)
plot_pacf(All_REIT_train.diff().dropna(), ax = ax2, title = 'Autocorrélation partie

plt.show()
```



In [19]:

```
# Corrélogramme acf et pacf deuxième différence

f = plt.figure(figsize=(10,4))

ax1 = f.add_subplot(121)
plot_acf(All_REIT_train.Indice.diff().diff().dropna(), ax = ax1, title = 'Autocorré

ax2 = f.add_subplot(122)
plot_pacf(All_REIT_train.Indice.diff().diff().dropna(), ax = ax2, title = 'Autocorr

plt.show()
```



L'observation des corrélogrammes de la première différence de la série des FPI ne permet de déduire de valeurs significatives.

Par contre, les corrélogrammes de la deuxième différence nous permettent d'avoir des valeurs significatives de 2 pour le terme q et de 4 pour le terme p.

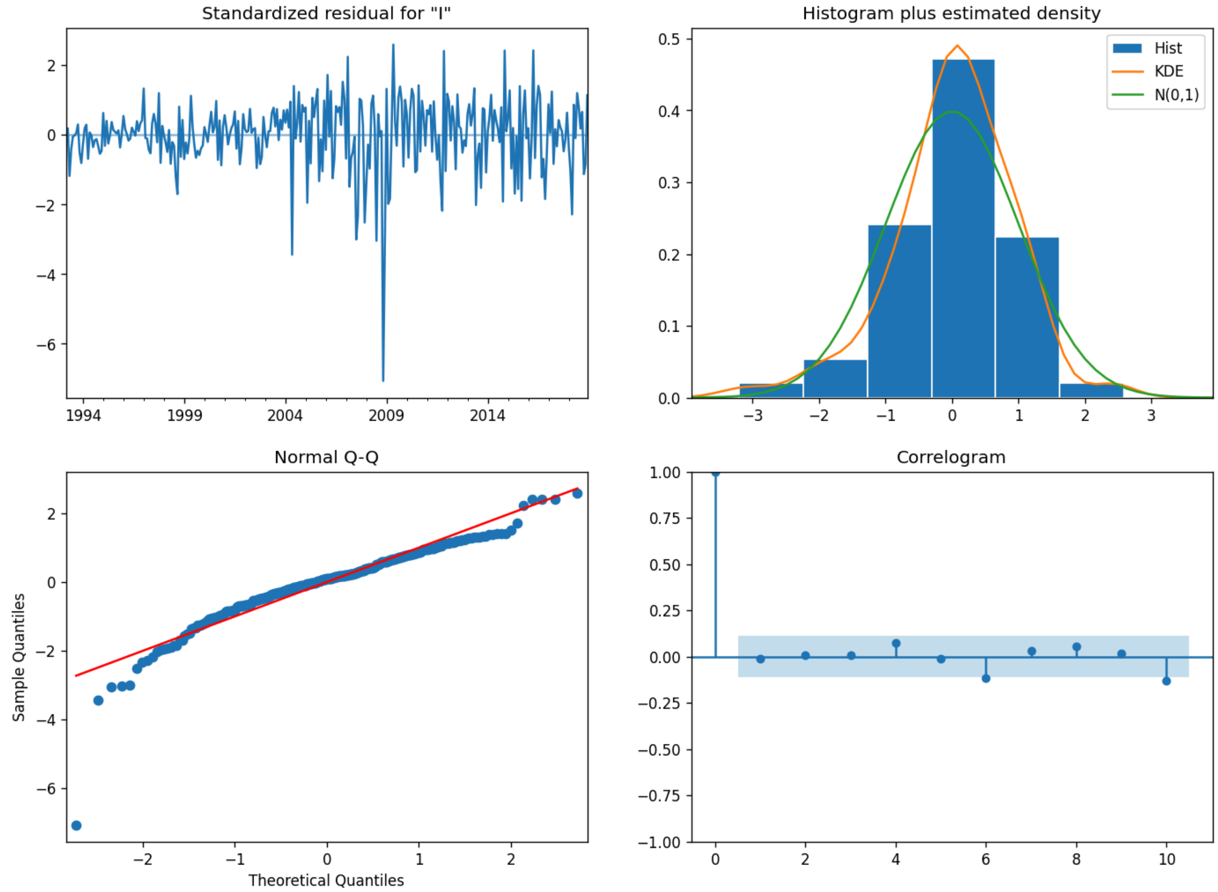
Au final, le modèle retenu sera un modèle ARIMA(4, 2, 2).

2.1.2 Estimation du modèle

In [20]:
```
# ARIMA(p=1, d=1, q=2) / ARIMA(AR, diff, MA)

arima_model = ARIMA(All_REIT_train.Indice, order = (4, 2, 2))
model = arima_model.fit()
print(model.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 Indice   No. Observations:                  312
Model:                 ARIMA(4, 2, 2)   Log Likelihood               -1023.824
Date:                Sat, 16 Nov 2024   AIC                           2061.648
Time:                        11:40:23   BIC                           2087.804
Sample:                    01-01-1993   HQIC                          2072.104
                         - 12-01-2018
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.9227      0.063    -14.674      0.000      -1.046      -0.799
ar.L2         -0.0306      0.070     -0.437      0.662      -0.168       0.107
ar.L3          0.0148      0.068      0.218      0.827      -0.118       0.148
ar.L4          0.0892      0.053      1.690      0.091      -0.014       0.193
ma.L1         -0.0169      0.967     -0.017      0.986      -1.911       1.878
ma.L2         -0.9829      0.952     -1.033      0.302      -2.848       0.882
sigma2        42.4854     40.542      1.048      0.295     -36.975     121.946
===================================================================================
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):               976.07
Prob(Q):                              0.85   Prob(JB):                         0.00
Heteroskedasticity (H):               4.04   Skew:                            -1.54
Prob(H) (two-sided):                  0.00   Kurtosis:                        11.13
===================================================================================

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

In [21]:
```
# https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting

model.plot_diagnostics(figsize=(14,10))
plt.show()
```



2.1.3 Vérification du modèle

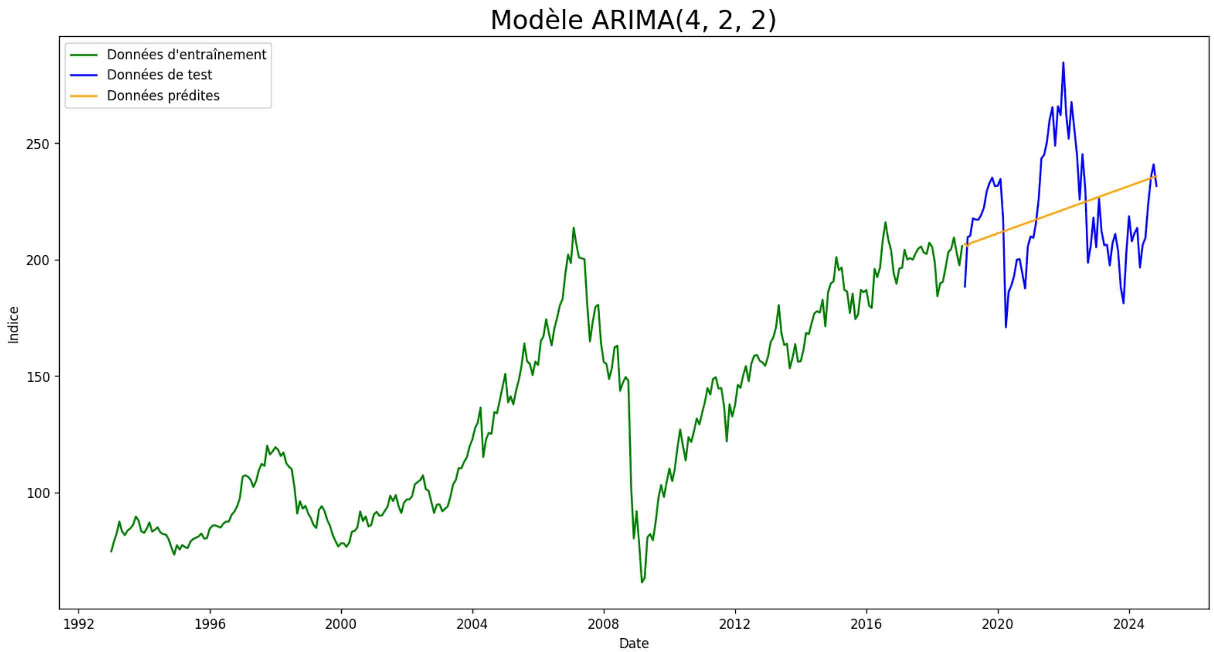
In [22]:
```
# Prévision méthode Box-Jenkins
All_REIT_predit = model.forecast(All_REIT_test.shape[0], alpha=0.05)  # 95% confian
All_REIT_predit = pd.Series(All_REIT_predit, index=All_REIT_test.index)
# lower_series = pd.Series(conf[:, 0], index=All_REIT_test.index)
# upper_series = pd.Series(conf[:, 1], index=All_REIT_test.index)

# Indicateurs d'évaluation du modèle
rmse = np.sqrt(mean_squared_error(All_REIT_test, All_REIT_predit))
mae = mean_absolute_error(All_REIT_test, All_REIT_predit)
mape = mean_absolute_percentage_error(All_REIT_test, All_REIT_predit, multioutput='
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
Test RMSE: 25.246
Test MAE: 21.509
Test MAPE: 0.097
```

In [23]:
```
# Comparaison entre les données prédites et les données de test
plt.figure(figsize=(16,8))

plt.xlabel('Date')
plt.ylabel('Indice')
plt.plot(All_REIT_train, 'green', label='Données d\'entraînement')
plt.plot(All_REIT_test, 'blue', label='Données de test')
```

```python
plt.plot(All_REIT_predit, 'orange', label='Données prédites')
plt.legend()
plt.title('Modèle ARIMA(4, 2, 2)', fontsize = 20)
```

Out[23]: Text(0.5, 1.0, 'Modèle ARIMA(4, 2, 2)')



2.2 Utilisation de la méthode automatique de détermination des termes (p, d, q)

2.2.1 Identification du modèle

In [24]:
```python
# Utilisation de la méthode automatique de détermination
arima_model_auto =  auto_arima(All_REIT_train, start_p=1, start_q=1,
                          test='adf',
                          max_p=6, max_q=6,
                          m=12,
                          d=2,
                          seasonal=False,
                          start_P=0,
                          D=1,
                          trace=True,
                          error_action='ignore',
                          suppress_warnings=True,
                          stepwise=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.19 sec
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=2252.864, Time=0.02 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=2190.572, Time=0.04 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.12 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=2250.865, Time=0.02 sec
 ARIMA(2,2,0)(0,0,0)[0] intercept   : AIC=2131.698, Time=0.05 sec
 ARIMA(3,2,0)(0,0,0)[0] intercept   : AIC=2106.536, Time=0.11 sec
 ARIMA(4,2,0)(0,0,0)[0] intercept   : AIC=2101.270, Time=0.16 sec
 ARIMA(5,2,0)(0,0,0)[0] intercept   : AIC=2102.852, Time=0.14 sec
 ARIMA(4,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.41 sec
 ARIMA(3,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.34 sec
 ARIMA(5,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.40 sec
 ARIMA(4,2,0)(0,0,0)[0]             : AIC=2099.273, Time=0.06 sec
 ARIMA(3,2,0)(0,0,0)[0]             : AIC=2104.539, Time=0.05 sec
 ARIMA(5,2,0)(0,0,0)[0]             : AIC=2100.854, Time=0.06 sec
 ARIMA(4,2,1)(0,0,0)[0]             : AIC=inf, Time=0.33 sec
 ARIMA(3,2,1)(0,0,0)[0]             : AIC=inf, Time=0.22 sec
 ARIMA(5,2,1)(0,0,0)[0]             : AIC=inf, Time=0.36 sec

Best model:  ARIMA(4,2,0)(0,0,0)[0]
Total fit time: 3.093 seconds
```

2.2.2 Estimation du modèle

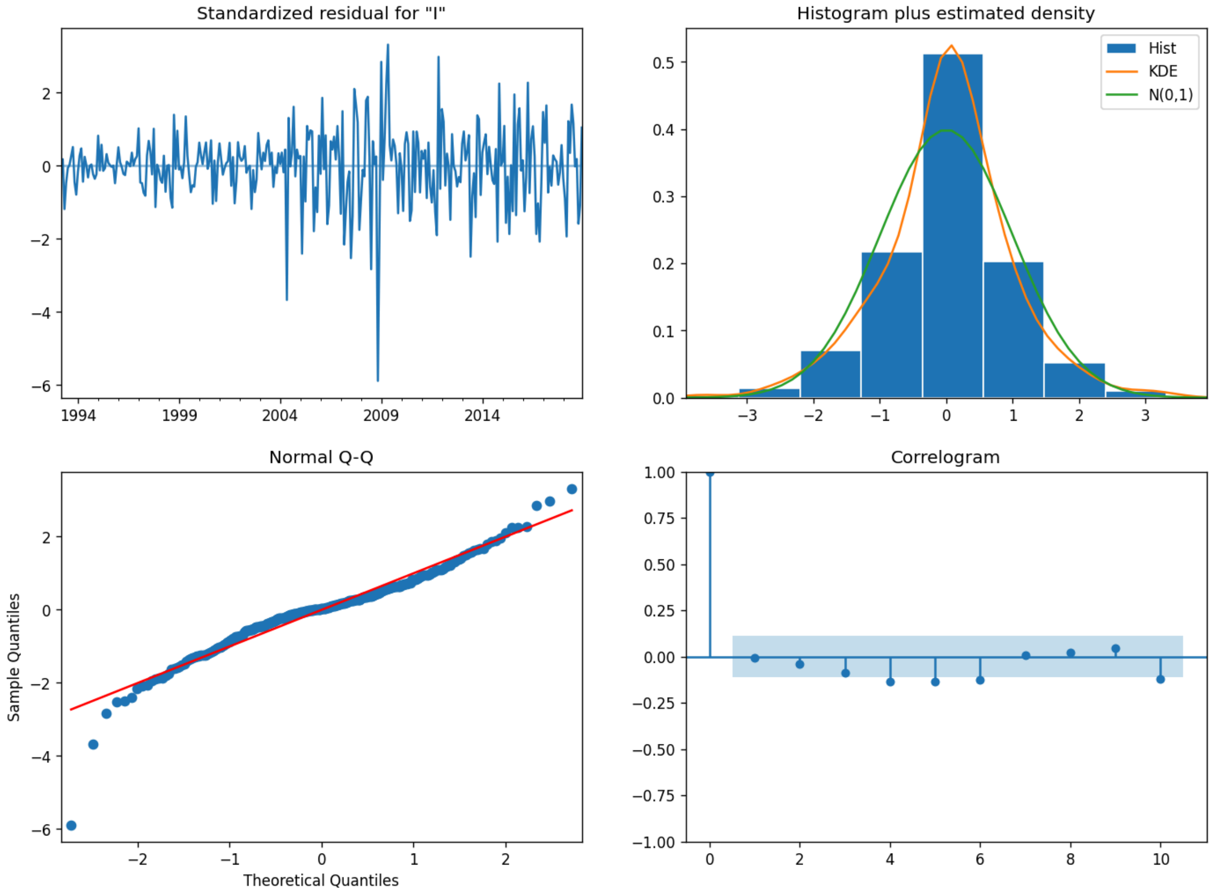
In [25]:

```
arima_model_auto = ARIMA(All_REIT_train.Indice, order = (4, 2, 0))
model_auto = arima_model_auto.fit()
print(model_auto.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 Indice   No. Observations:                  312
Model:                 ARIMA(4, 2, 0)   Log Likelihood               -1044.636
Date:                Sat, 16 Nov 2024   AIC                           2099.273
Time:                        11:40:27   BIC                           2117.956
Sample:                    01-01-1993   HQIC                          2106.741
                         - 12-01-2018
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7851      0.050    -15.846      0.000      -0.882      -0.688
ar.L2         -0.6943      0.050    -13.859      0.000      -0.792      -0.596
ar.L3         -0.4038      0.052     -7.787      0.000      -0.505      -0.302
ar.L4         -0.1525      0.042     -3.594      0.000      -0.236      -0.069
sigma2        49.3336      2.816     17.522      0.000      43.815      54.852
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):               275.33
Prob(Q):                              0.90   Prob(JB):                         0.00
Heteroskedasticity (H):               4.61   Skew:                            -0.71
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.39
===================================================================================

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

In [26]:

```
model_auto.plot_diagnostics(figsize=(14,10))
plt.show()
```



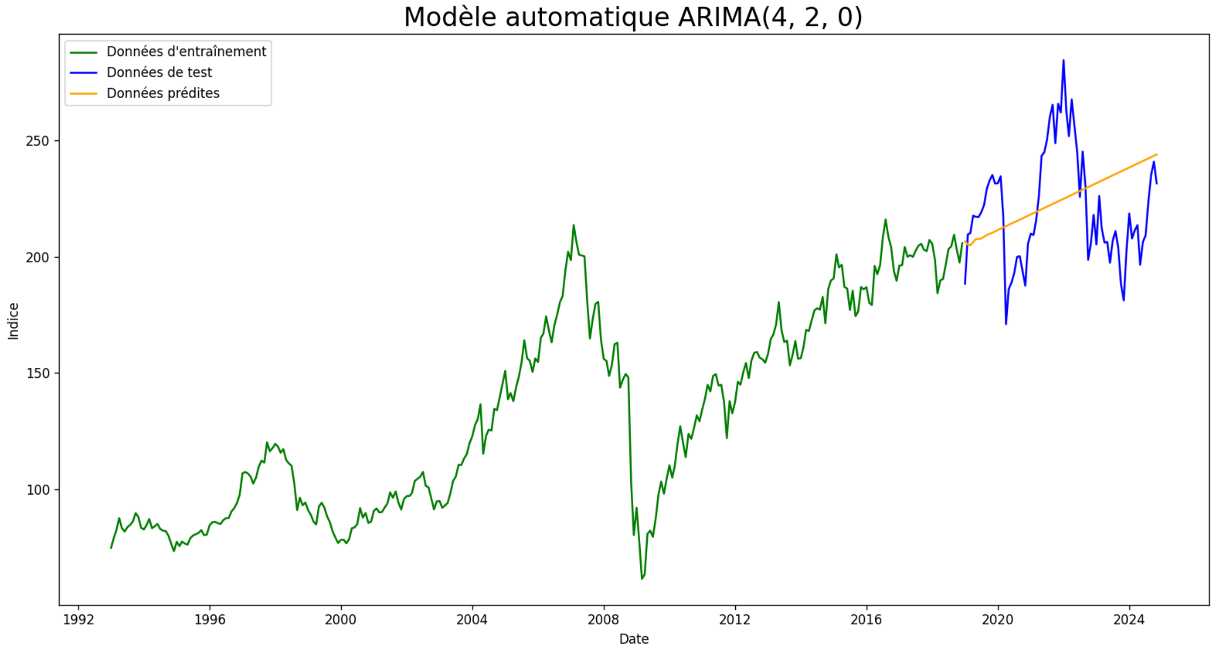
2.2.3 Vérification du modèle

In [27]:
```
# Prévisions modèle automatique
All_REIT_predit_auto = model_auto.forecast(All_REIT_test.shape[0], alpha=0.05)  # 9
All_REIT_predit_auto = pd.Series(All_REIT_predit_auto, index=All_REIT_test.index)

# Indicateurs d'évaluation du modèle
rmse = np.sqrt(mean_squared_error(All_REIT_test, All_REIT_predit_auto))
mae = mean_absolute_error(All_REIT_test, All_REIT_predit_auto)
mape = mean_absolute_percentage_error(All_REIT_test, All_REIT_predit_auto, multiout
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
Test RMSE: 26.579
Test MAE: 23.215
Test MAPE: 0.106
```

In [28]:
```
# Comparaison entre les données prédites et les données de test
plt.figure(figsize=(16,8))
plt.xlabel('Date')
plt.ylabel('Indice')
plt.plot(All_REIT_train, 'green', label='Données d\'entraînement')
plt.plot(All_REIT_test, 'blue', label='Données de test')
plt.plot(All_REIT_predit_auto, 'orange', label='Données prédites')
plt.legend()
plt.title('Modèle automatique ARIMA(4, 2, 0)', fontsize = 20)
```

Out[28]: Text(0.5, 1.0, 'Modèle automatique ARIMA(4, 2, 0)')



3. Application de la technique de "Rolling forecast"

Le principe de la technique de "rolling forecast" consiste à :

- faire la prévision pour une période,
- inclure la prévison dans les données d'entraînement,
- refaire l'estimation du modèle avec la nouvelle série obtenue.

3.1 Rolling forecast pour le modèle ARIMA(4, 2, 2)

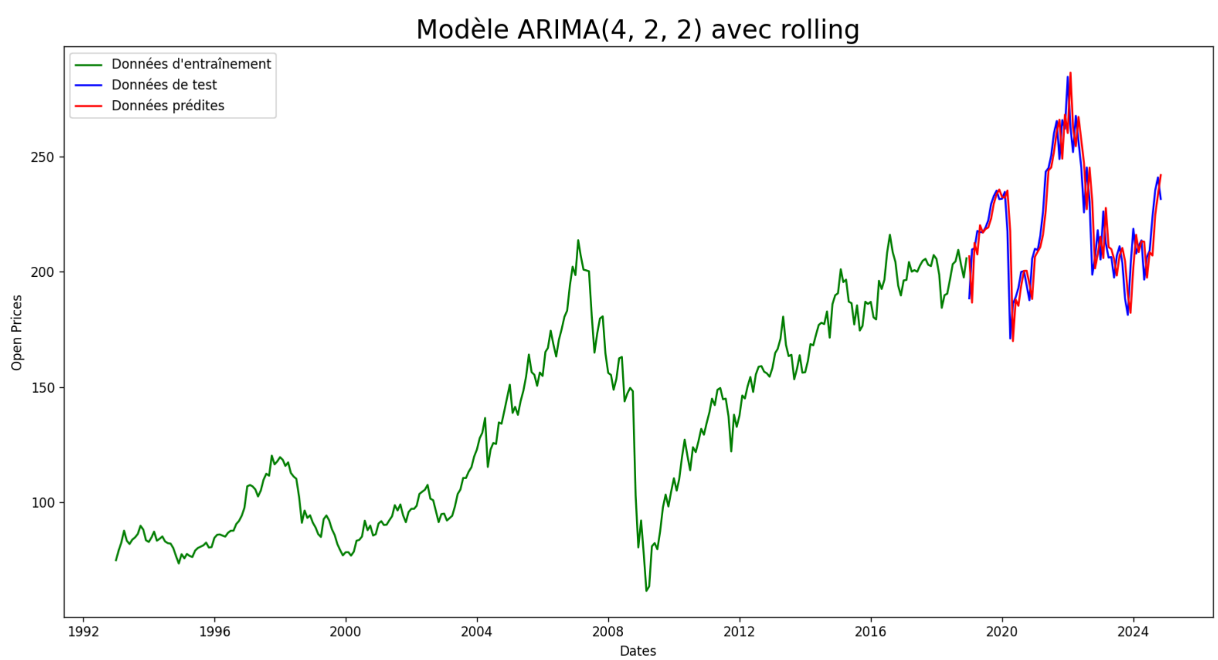
In [29]:
```
# https://mlpills.dev/time-series/forecasting-in-time-series/

# Définir le modèle et l'horizon
order = (4,2,2)
steps = All_REIT_test.shape[0]

# Initialisation liste des prévisions
predictions = []

# Boucle pour chaque mois
for step in range(steps):

    # Ajouter nouvelle donnée prédite
    All_REIT_train_i = pd.concat([All_REIT_train.Indice, All_REIT_test.Indice[:step

    # Entraînement du modèle
    model_i = ARIMA(All_REIT_train_i.values, order=order).fit()
```

```
    # Prédire la valeur pour le prochain mois
    pred = model_i.forecast(steps=1)

    # Ajouter à la liste des prévisions
    predictions.append(pred)

# Convertir la liste en dataframe
Predictions_manuel_rolling = pd.DataFrame(predictions,
                                          columns=['Predict'],
                                          index=All_REIT_test[:steps].index)

# Indicateurs d'évaluation du modèle
rmse = np.sqrt(mean_squared_error(All_REIT_test, Predictions_manuel_rolling))
mae = mean_absolute_error(All_REIT_test, Predictions_manuel_rolling)
mape = mean_absolute_percentage_error(All_REIT_test, Predictions_manuel_rolling, mu
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
Test RMSE: 13.181
Test MAE: 10.178
Test MAPE: 0.047
```

In [30]:
```
plt.figure(figsize=(16,8))
plt.xlabel('Dates')
plt.ylabel('Open Prices')
plt.plot(All_REIT_train, 'green', label='Données d\'entraînement')
plt.plot(All_REIT_test, 'blue', label='Données de test')
plt.plot(Predictions_manuel_rolling, 'red', label='Données prédites')
plt.legend()
plt.title('Modèle ARIMA(4, 2, 2) avec rolling', fontsize = 20)
```

Out[30]:
```
Text(0.5, 1.0, 'Modèle ARIMA(4, 2, 2) avec rolling')
```



3.2 Rolling forecast pour le modèle automatique ARIMA(4, 2, 0)

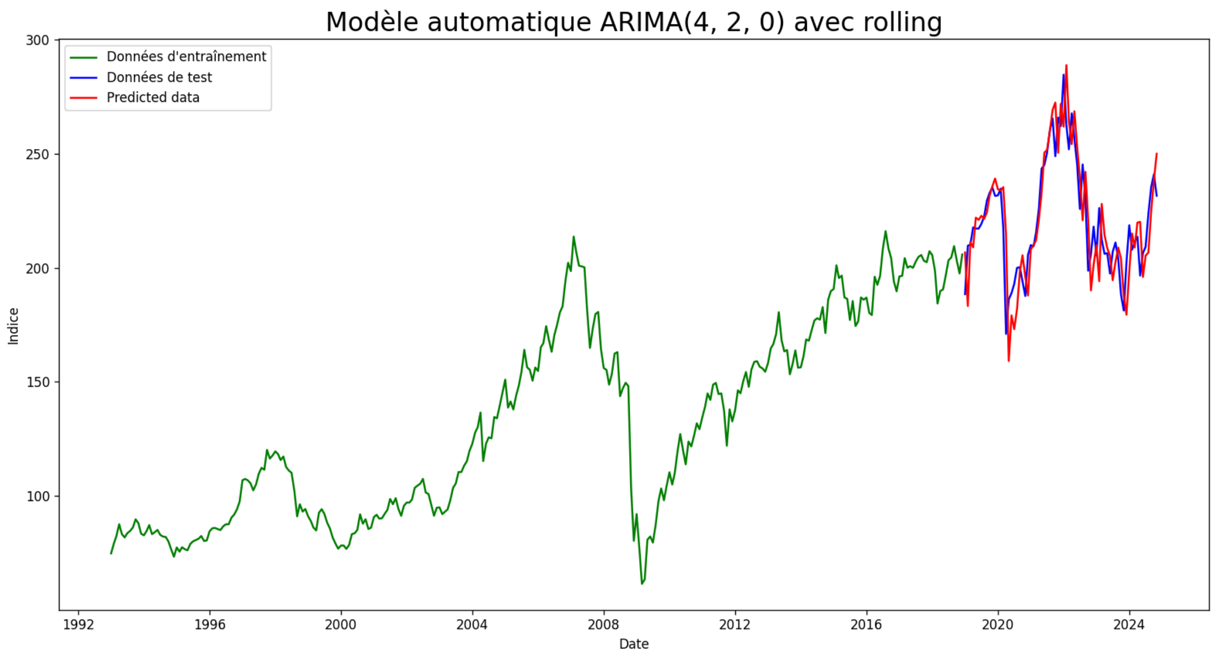
In [31]:
```
# https://mlpills.dev/time-series/pour laecasting-in-time-series/

# Définir le modèle et l'horizon
order = (4,2,0)
steps = All_REIT_test.shape[0]

# Initialisation liste des prévisions
predictions = []

# Boucle pour chaque mois
for step in range(steps):

    # Ajouter nouvelle donnée prédite
    All_REIT_train_i = pd.concat([All_REIT_train, All_REIT_test[:step]], axis=0)

    # Entraînement du modèle
    model_i = ARIMA(All_REIT_train_i.values, order=order).fit()

    # Prédire la valeur pour le prochain mois
    pred = model_i.forecast(steps=1)

    # Ajouter à la liste des prévisions
    predictions.append(pred)

# Convertir la liste en dataframe
Predictions_auto_rolling = pd.DataFrame(predictions,
                                   columns=['Predict'],
                                   index=All_REIT_test[:steps].index)
```

In [32]:
```
# Indicateurs d'évaluation du modèle
rmse = np.sqrt(mean_squared_error(All_REIT_test, Predictions_auto_rolling))
mae = mean_absolute_error(All_REIT_test, Predictions_auto_rolling)
mape = mean_absolute_percentage_error(All_REIT_test, Predictions_auto_rolling, mult
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)
print('Test MAPE: %.3f' % mape)
```

```
Test RMSE: 14.427
Test MAE: 11.361
Test MAPE: 0.053
```

In [33]:
```
plt.figure(figsize=(16,8))
plt.xlabel('Date')
plt.ylabel('Indice')
plt.plot(All_REIT_train, 'green', label='Données d\'entraînement')
plt.plot(All_REIT_test, 'blue', label='Données de test')
plt.plot(Predictions_auto_rolling, 'red', label='Predicted data')
plt.legend()
plt.title('Modèle automatique ARIMA(4, 2, 0) avec rolling', fontsize = 20)
```

```
Out[33]:  Text(0.5, 1.0, 'Modèle automatique ARIMA(4, 2, 0) avec rolling')
```



4. Conclusion

Les figures comparant les données prédites et les données de test montrent clairement que la technique de "rolling forecast" améliore beaucoup la qualité des prévisions. Les modèles de base sans application de cette technique montrent des performances décevantes.

Dans tous les cas, le modèle estimé à partir de la méthodologie de Box-Jenkins performe mieux que le modèle automatique.

Récapitulatif des résultats des tests d'évaluation

		SANS "Rolling forecast"		AVEC "Rolling forecast"	
		ARIMA(4, 2, 2)	ARIMA(4, 2, 0)	ARIMA(4, 2, 2)	ARIMA(4, 2, 0)
Test	RMSE	25.246	26.579	13.181	14.427
	MAE	21.509	23.215	10.178	11.361
	MAPE	0.097	0.106	0.047	0.053

Le récapitulatif des résultats des tests d'évaluation de modèle confirme ce fait. Plus précisément, l'appication de la technique de "rolling forecast" permet de diviser d'environ de moitié les valeurs des différents tests.