

# Séries temporelles

*Alexandra TON, Erivan INAN, Inès CHABANI, Sarah OUAHAB*

17/12/2023

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contextualisation de l'étude . . . . .	1
1.2	Librairies . . . . .	2
<b>2</b>	<b>Prétraitement des données</b>	<b>2</b>
2.1	Série temporelle . . . . .	3
2.2	Décomposition de la série temporelle : Fonction Python . . . . .	3
2.3	Décomposition de la série temporelle : Méthode analytique . . . . .	4
2.4	Test de stationnarité de Dickey-Fuller . . . . .	6
<b>3</b>	<b>Modélisation</b>	<b>7</b>
3.1	Modèle MA . . . . .	7
3.2	Modèle AR . . . . .	8
3.3	Modèle ARMA . . . . .	9
<b>4</b>	<b>Étude des résidus</b>	<b>11</b>
4.1	Outils d'évaluation des résidus . . . . .	11
4.2	Résidus des modèles étudiés . . . . .	13
<b>5</b>	<b>Conclusion</b>	<b>17</b>

## 1 Introduction

### 1.1 Contextualisation de l'étude

Notre étude se base sur le Dataset mis à disposition par ODRE : Open Data Réseaux Energie, un groupement comprenant notamment GRT Gaz et Teréga. Ces groupes agissent dans la chaîne d'approvisionnement en gaz naturel, assurant pour GRTgaz le transport et, dans le cas de Teréga, le stockage du gaz. Leurs clients sont des acteurs du marché gazier qui interviennent à différentes étapes de la production, de la distribution, et de la commercialisation du gaz naturel, ainsi que les industriels.

Voici le **lien des données** utilisées dans ce projet. Ces données de consommation correspondent aux données de consommation relevées sur le réseau de transport.

Nous étudions la consommation brute de gaz totale en France. Celle-ci est mesurée en Mégawatt (MW), avec pour convention, un Pouvoir Calorifique Supérieur (PCS) à 0 degré. Celui-ci correspond à la quantité totale de chaleur dégagée à volume constant par la combustion d'un kg de combustible. Les mesures s'étendent sur la période 2012-2023.

Nous avons choisi d'entraîner notre modèle sur 80% des données mises à notre disposition, ce qui correspond à la période allant des années 2012 à mi-2020. Nous tâcherons de prédire les données de consommation quotidienne brute de gaz totale en France d'octobre 2021 à décembre 2022.

Le secteur résidentiel occupe 40% de la consommation de gaz naturel en France. C'est la part la plus significative des consommateurs, après les industriels (36%) (Voir cette étude de l'INSEE). En effet, les ménages utilisent le gaz pour le chauffage, la cuisson, l'eau chaude, et d'autres besoins domestiques. Nous allons donc supposer dans notre étude le schéma usuel de consommation de gaz chez un ménage.

## 1.2 Librairies

Voici les librairies que nous avons importées sur Python :

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import statsmodels.api as sm
import datetime
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from datetime import timedelta
import seaborn as sns
from scipy.stats import norm
import datetime
import warnings
```

## 2 Prétraitemet des données

La base de données que nous possédons recueille les données sur la consommation brute totale de gaz en France pour la période s'étendant de 2012 à 2023. Celle-ci est mesurée en MW PCS 0°C, toutes les demi-heures, chaque jour.

Nous allons additionner toutes les mesures journalières afin de travailler sur les données de consommation brute totale de gaz par jour entre 2012 et fin 2022. La bibliothèque panda permet de lire et d'afficher ces données sur Python. Ci-dessous un aperçu des données implémentées sur Python :

	Date	Consommation brute gaz totale (MW PCS 0°C)
0	2012-01-01	1446815.0
1	2012-01-02	3131931.0
2	2012-01-03	1820934.0
3	2012-01-04	1151109.0
4	2012-01-05	875789.0
	...	...
3982	2022-12-07	685910.0
3983	2022-12-08	521077.0
3984	2022-12-09	676490.0
3985	2022-12-10	985912.0
3986	2022-12-11	1209577.0

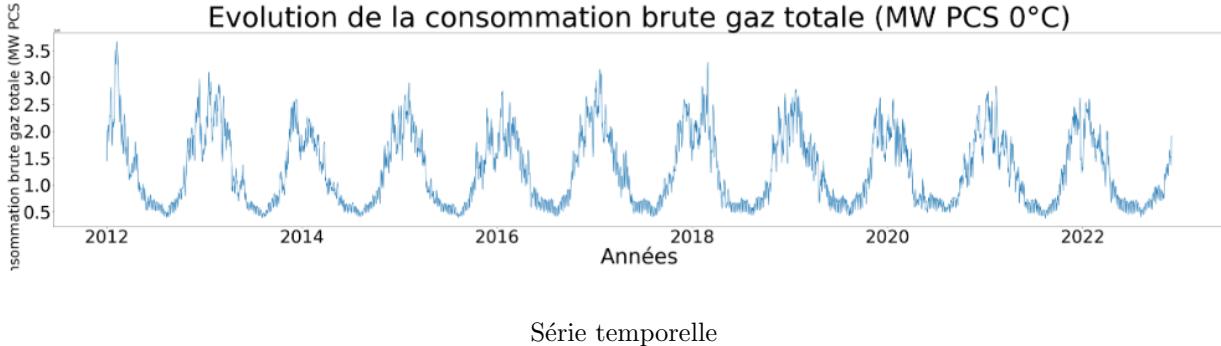
3987 rows × 2 columns

Données de consommation brute de gaz totale journalière

Par la suite, ces données constitueront notre base et seront stockées sous le nom **Dgj**.

## 2.1 Série temporelle

Ce tableau de données Python nous permet d'afficher le graphique de notre série temporelle  $(X_t)_{t \in \mathbb{N}}$ .



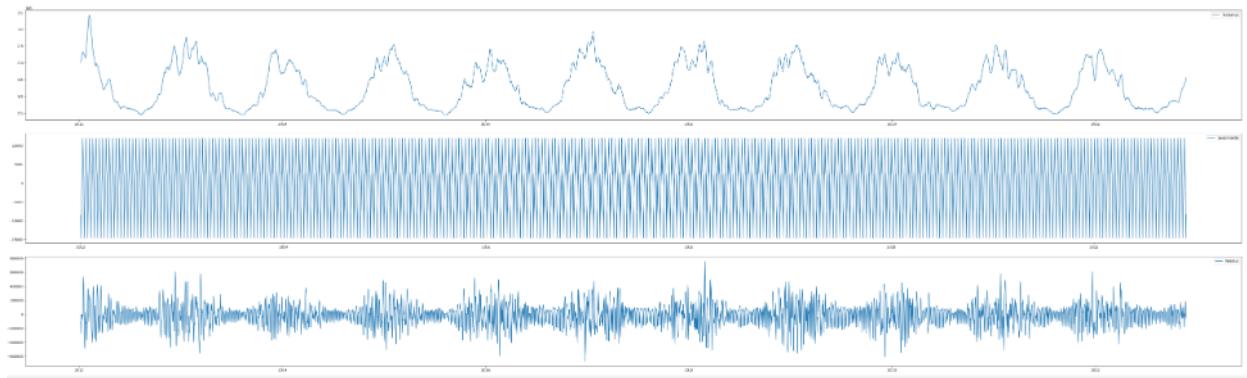
Série temporelle

## 2.2 Décomposition de la série temporelle : Fonction Python

Nous allons utiliser une fonction déjà existante sur Python, `seasonal_decompose()`, qui décompose automatiquement la série temporelle et permet d'isoler de la série temporelle les tendance, saisonnalité et résidus.

```
result = seasonal_decompose(Dgj[ 'Consommation brute gaz totale (MW PCS 0C)' ], model='additive', period=12)
# Extraction des composantes
tendance = result.trend
saisonalite = result.seasonal
residus = result.resid
```

Puis nous pouvons afficher les graphiques des différentes composantes isolées :



Isolation des composantes : tendance, saisonnalité, résidus

Nous avons rencontré à cette étape du projet une difficulté.

En effet, le dernier graphique, correspondant aux résidus de notre série temporelle semble conserver une périodicité. Cela suppose donc que la fonction Python a ses limites, notamment lorsque la série temporelle admet à la fois une saisonnalité additive et multiplicative. Nous allons donc passer par une méthode analytique qui implique une transformation fonctionnelle.

## 2.3 Décomposition de la série temporelle : Méthode analytique

La série temporelle étant une fonction du temps, nous allons supposer deux choses : nous sommes dans le cadre d'un **modèle déterministe** et **additif**. Dans ce cadre, notre série temporelle  $(D_t)_{t \in \mathbb{N}}$  peut s'écrire sous la forme suivante :

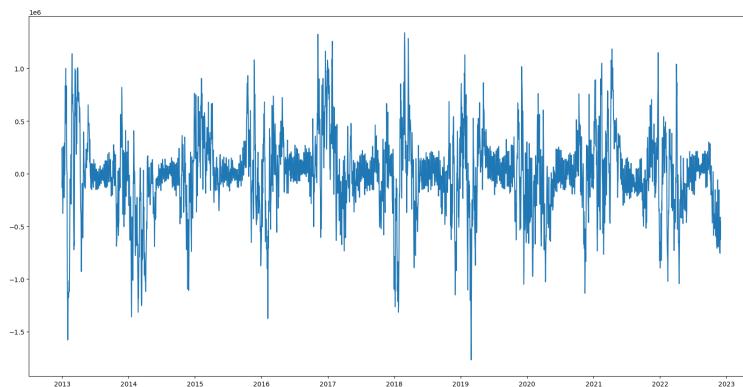
$$D_t = f(t) + S(t) + X_t$$

avec :  $f$  la tendance

$S$  la saisonnalité

$X_t$  la série stationnaire, correspondant aux résidus, représentant les variations irrégulières du bruit.

En réalisant une première différenciation dans le but de se débarrasser de la tendance et saisonnalité additive, on observe qu'il reste une forte saisonnalité, on soupçonne donc que notre série contient une saisonnalité multiplicative :



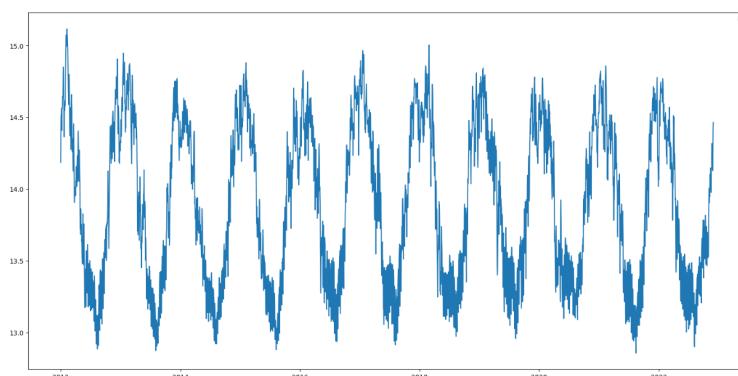
Série différenciée de période 365 (ordre 1)

Nous allons donc réaliser une transformation logarithmique, afin de rendre toute saisonnalité restante additive. Cette transformation sert également à stabiliser la série dans laquelle nous observons graphiquement une évolution de la variance dans le temps.

En effet, en appliquant une fonction logarithmique, nous sommes capables de transformer un produit en somme. Nous obtenons alors l'équation suivante :

$$\log(D_t) = \log(f(t) \times S(t) \times X_t) = \log(f(t)) + \log(S(t)) + \log(X_t)$$

Cette transformation change en partie notre série, mais nous permet de réaliser la stationnarisation de manière plus simple, en utilisant des outils initialement adaptés aux modèles additifs.



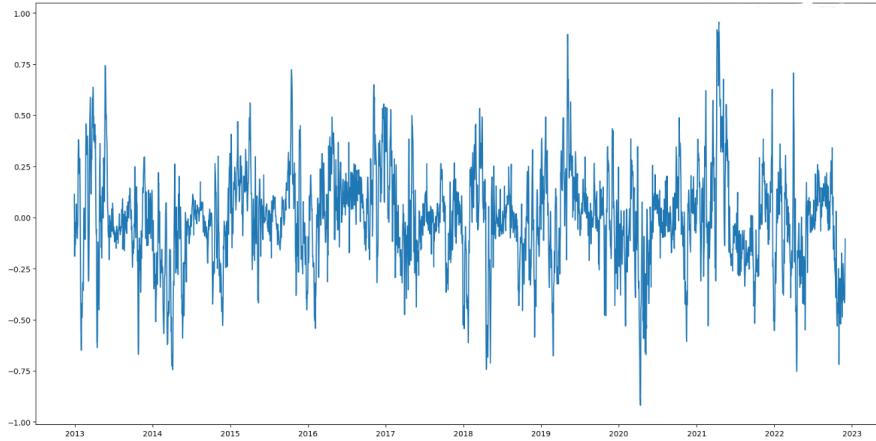
Logarithme de la série temporelle

Afin de décomposer notre série, nous allons d'abord procéder à la différenciation de notre série.

Pour commencer, nous pouvons effectuer une différenciation sur une période d'un an, équivalente à 365 jours, étant donné que cette période est clairement identifiable sur notre graphique. Cette approche est également intuitive, car la consommation de gaz possède une tendance à la baisse en avril/mai, lorsque les températures commencent à augmenter, puis une tendance à la hausse courant septembre/octobre.

Mathématiquement, différencier une série temporelle de période 365 revient à effectuer l'opération suivante :

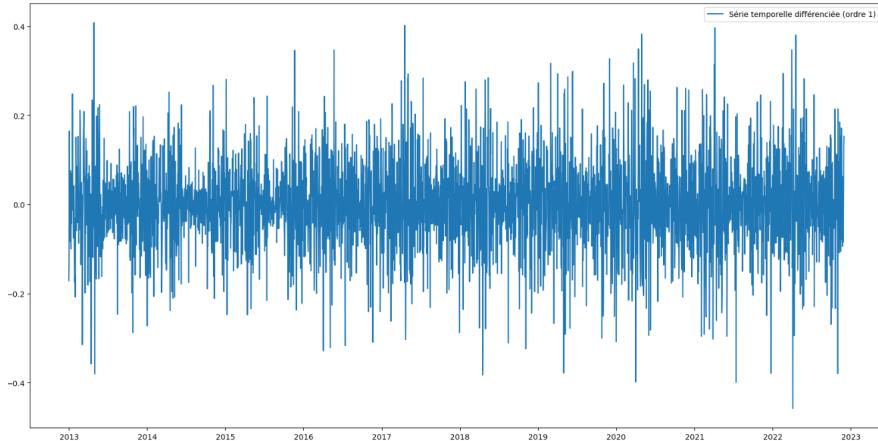
$$\Delta D_t = D_t - D_{t-365}$$



Série différenciée de période 365 (ordre 1), après transformation logarithmique

Une fois cette première différenciation réalisée, on perçoit toujours une légère saisonnalité, la série ne paraît pas parfaitement stationnaire. On peut alors procéder à une seconde différenciation afin d'éliminer une éventuelle variation quotidienne :

$$\Delta^2 D_t = \Delta D_t - \Delta D_{t-1}$$



Série différenciée de période 1 (ordre 2), après transformation logarithmique

La série a bien l'air stationnaire à l'oeil nu, nous n'observons pas de tendance ou de période visible. Nous pouvons maintenant réaliser un test pour confirmer la stationnarité de notre série.

## 2.4 Test de stationnarité de Dickey-Fuller

Les tests de stationnarité permettent de vérifier si une série est stationnaire ou non.

Il existe deux types de test de stationnarité :

1. Les tests partant de l'hypothèse nulle **H0** est que la série est stationnaire. L'hypothèse alternative **Ha** est que la série est non stationnaire.
2. Les tests de racine unitaire, ils partent de l'hypothèse nulle **H0** que la série ne l'est pas. L'hypothèse alternative **Ha** est donc que la série est stationnaire.

Le test de Dickey-Fuller est du deuxième type.

Une série temporelle avec une racine unitaire présente une caractéristique où les valeurs de la série dépendent de manière aléatoire de leurs valeurs passées. Cela signifie qu'il existe une sorte de mémoire dans la série qui affecte son comportement futur. La présence d'une racine unitaire dans une série temporelle est un indicateur de non-stationnarité. Une série non-stationnaire a des propriétés statistiques, comme la moyenne et la variance, qui changent avec le temps. Cela rend difficile la prévision de son comportement futur, car les modèles qui supposent la stationnarité ne seront pas fiables.

Le terme *racine unitaire* provient du fait que l'unité est une racine du polynôme associé à la série temporelle. Mathématiquement, une série temporelle stationnaire  $(X_t)_{t \in Z}$  après différenciation a une représentation autorégressive de la forme :

$$\Phi_p = 1 - \phi_1 L - \phi_2 L^2 - \dots - \phi_p L^p$$

où :  $L$  est l'opérateur de retard

$\phi_1, \dots, \phi_p$  sont les coefficients du modèle AR

$p$  est l'ordre du modèle

Ce qui implique la factorisation suivante du polynôme :

$$\Phi_p(L) = (1 - L)\Phi'_{p-1}(L)$$

où  $\Phi'_{p-1}(L)$  est un polynôme d'ordre  $p - 1$  en  $L$

Ci-dessous un exemple d'implémentation du test de Dickey-Fuller en Python :

```
import pandas as pd
from statsmodels.tsa.stattools import adfuller
y = diff2_Dgj-exp ['Consommation brute gaz totale (MW PCS 0C)'].dropna()

#test ADF
resultats_adf = adfuller(y)

# Affichage des resultats
print('Statistique ADF :', resultats_adf[0])
print('P-valeur :', resultats_adf[1])
print('Nombre de retards utilises :', resultats_adf[2])
print('Nombre d'observations utilisees :', resultats_adf[3])
print('Valeurs critiques :', resultats_adf[4])

# Interpretation des resultats
if resultats_adf[1] <= 0.05:
    print('La serie temporelle semble stationnaire (rejeter l'hypothese nulle)')
else:
    print('La serie temporelle ne semble pas stationnaire (echec de rejeter l'hypothese nulle)')
```

L'exécution du code nous renvoie :

```
Statistique ADF : -19.118421487493663
P-valeur : 0.0
Nombre de retards utilisés : 19
Nombre d'observations utilisées : 3602
Valeurs critiques : {'1%': -3.432166759109125, '5%': -2.862342742516423, '10%': -2.567197312560947}
La série temporelle semble stationnaire (rejeter l'hypothèse nulle)
```

### Interprétation du test :

$H_0$  = La série possède une racine unitaire

$H_a$  = La série ne possède pas de racine unitaire, elle est stationnaire

Étant donné que notre p-value est très proche de 0, elle est inférieure aux niveaux de prédictions fournis par les valeurs critiques  $\alpha = 0.10$ ,  $\alpha = 0.05$  et  $\alpha = 0.01$ .

## 3 Modélisation

Dans cette partie, nous allons décrire et utiliser différents modèles pour comprendre et prédire, éventuellement, les valeurs futures de cette série. Nous déterminerons enfin le modèle le plus adapté à nos données.

### 3.1 Modèle MA

Étant donné une série stationnaire  $(X_t)_{t \in \mathbf{Z}}$ , son bruit blanc  $(Z_t)_{t \in \mathbf{Z}}$ , un processus MA( $q$ ) est une solution (sous réserve d'existence) à l'équation suivante :

$$X_t = \sum_{j \geq 0} \psi_j Z_{t-j}$$

avec  $\psi_j = 0$  pour  $j > q$

On considère ici que le processus est la résultante d'une combinaison linéaire de perturbations décorrélées (un bruit blanc et son passé).

### Caractérisation

Si  $(X_t)_{t \in \mathbf{Z}}$  est un processus MA( $q$ ) alors ses autocorrélations simples s'annulent à partir du rang  $q + 1$  :

$$\begin{cases} \rho(q) \neq 0 \\ \forall h \in N, h \geq q + 1 : \rho(h) = 0 \end{cases}$$

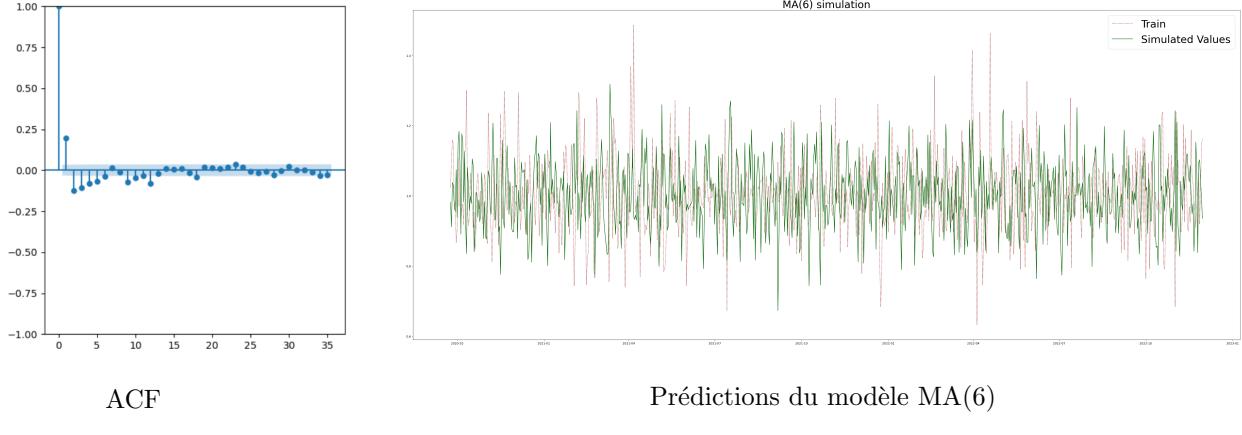
### Identification de l'ordre du modèle

Traçons le graphique de notre fonction d'Autocorrélation (ACF) afin de déterminer la valeur du paramètre  $q$  :

En observant le graphique de la fonction d'autocorrélation, on constate une diminution progressive des valeurs, atteignant des niveaux négligeables après quelques lags. La corrélation s'annule initialement vers environ 6 lags, mais elle redevient ensuite significative avant de diminuer à nouveau et rester négligeable définitivement après 12 lags environ. Nous pouvons donc tenter d'utiliser le modèle de moyenne mobile en utilisant le paramètre 6. En effet, beaucoup de termes sont peu significatifs ou proches de l'être entre 6 et 12.

On en conclut donc que notre série  $X_t$  peut s'écrire sous la forme :

$$X_t = \sum_{j \geq 0} \psi_j Z_{t-j} \quad \text{avec } \psi_j = 0 \text{ pour } j > q = 6$$



Ce modèle nous fournit le sommaire suivant :

	coef	std err	z	P> z	[0.025	0.975]
const	1.0046	0.001	1085.439	0.000	1.003	1.006
ma.L1	0.2029	0.015	13.105	0.000	0.173	0.233
ma.L2	-0.1460	0.018	-8.131	0.000	-0.181	-0.111
ma.L3	-0.1443	0.018	-8.053	0.000	-0.179	-0.109
ma.L4	-0.1724	0.017	-9.914	0.000	-0.206	-0.138
ma.L5	-0.1274	0.018	-7.096	0.000	-0.163	-0.092
ma.L6	-0.0963	0.018	-5.402	0.000	-0.131	-0.061
sigma2	0.0085	0.000	50.420	0.000	0.008	0.009

Notre équation est donc la suivante :

$$X_t = 1.0046 + 0.2029X_{t-1} - 0.1460X_{t-2} - 0.1443X_{t-3} - 0.1724X_{t-4} - 0.1274X_{t-5} - 0.0963X_{t-6} + \epsilon_t$$

avec  $\epsilon_t$  le résidu de notre modèle

### 3.2 Modèle AR

Étant donné une série stationnaire  $(X_t)_{t \in \mathbb{Z}}$ , un processus AR( $p$ ) est une solution à l'équation suivante :

$$X_t = \phi_1 X_{t-1} + \dots + \phi_p X_{t-p} + Z_t$$

avec  $\theta = (\phi_1, \phi_2, \dots, \phi_p) \in R^p$   
 $(Z_t)_{t \in \mathbb{Z}}$  est un bruit blanc

#### Caractérisation

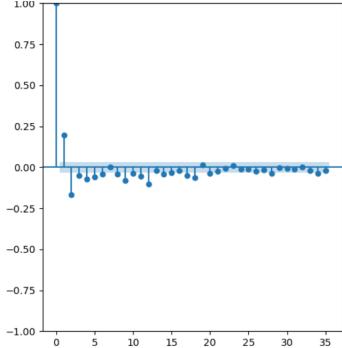
Si  $(X_t)_{t \in \mathbb{Z}}$  est un processus AR( $p$ ) alors ses autocorrélations partielles s'annulent à partir du rang  $p+1$  :

$$\begin{cases} \rho(p) \neq 0 \\ \forall h \in N, h \geq p+1 : \rho(h) = 0 \end{cases}$$

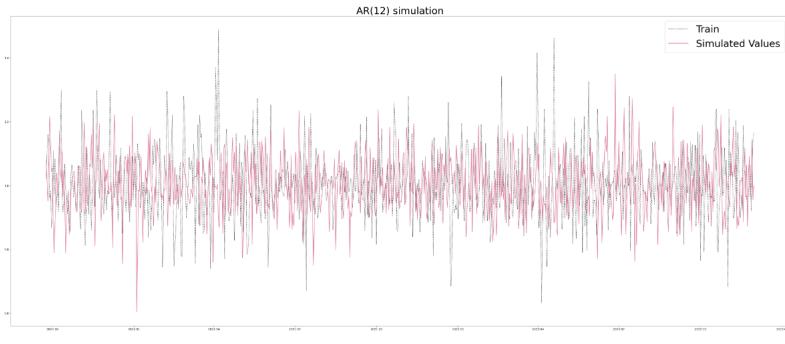
Réciproquement, il s'agit d'une condition nécessaire et suffisante pour qu'un processus  $(X_t)_{t \in \mathbb{Z}}$  soit un AR( $p$ ). Les autocorrélations simples, quant à elles, décroissent rapidement vers 0, de manière exponentielle ou sinusoïdale amortie.

### Identification de l'ordre du modèle

Afin de déterminer l'ordre  $p$  du modèle autorégressif, affichons le graphique de la fonction d'autocorrélation partielle (PACF) de notre série temporelle :



PACF



Prédictions du modèle MA(12)

Le tracé du PACF nous mène à choisir l'ordre  $p = 12$ .

Ce modèle nous fournit le sommaire suivant :

	coef	std err	z	P> z	[0.025	0.975]
const	1.0046	0.001	898.142	0.000	1.002	1.007
ar.L1	0.2051	0.015	13.403	0.000	0.175	0.235
ar.L2	-0.1758	0.018	-9.593	0.000	-0.212	-0.140
ar.L3	-0.0566	0.019	-2.999	0.003	-0.094	-0.020
ar.L4	-0.1193	0.018	-6.746	0.000	-0.154	-0.085
ar.L5	-0.0574	0.019	-3.060	0.002	-0.094	-0.021
ar.L6	-0.0635	0.020	-3.250	0.001	-0.102	-0.025
ar.L7	-0.0061	0.019	-0.329	0.742	-0.043	0.030
ar.L8	-0.0762	0.017	-4.388	0.000	-0.110	-0.042
ar.L9	-0.0712	0.019	-3.754	0.000	-0.108	-0.034
ar.L10	-0.0560	0.019	-3.012	0.003	-0.092	-0.020
ar.L11	-0.0165	0.017	-0.968	0.333	-0.050	0.017
ar.L12	-0.1864	0.016	-6.536	0.000	-0.138	-0.075
sigma2	0.0084	0.000	49.537	0.000	0.008	0.009

L'équation de notre modèle AR devient :

$$X_t = 1.0046 + 0.2051X_{t-1} - 0.1758X_{t-2} - 0.0566X_{t-3} - 0.1193X_{t-4} - 0.0574X_{t-5} - 0.0635X_{t-6} \\ - 0.0061X_{t-7} - 0.0762X_{t-8} - 0.0712X_{t-9} - 0.0560X_{t-10} - 0.0165X_{t-11} - 0.1064X_{t-12} + \epsilon_t$$

### 3.3 Modèle ARMA

Étant donné une série stationnaire  $(X_t)_{t \in \mathbf{Z}}$ , un processus ARMA( $p, q$ ) est une solution (sous réserve d'existence) à l'équation suivante :

$$X_t = \phi_1 X_{t-1} + \dots + \phi_p X_{t-p} + Z_t + \gamma_1 Z_{t-1} + \dots + \gamma_q Z_{t-q}, \quad t \in \mathbf{Z}$$

avec :  $\theta = (\phi_1, \dots, \phi_p, \gamma_1, \dots, \gamma_q)' \in \mathbf{R}^{p+q}$  les paramètres du modèle

$(Z_t)_{t \in \mathbf{Z}}$  est un BB( $\sigma^2$ )

## Identification de l'ordre du modèle

Le modèle est composé de deux parties : une partie Autorégressive (AR) et une partie Moyenne Mobile (MA). Le modèle est généralement noté ARMA( $p, q$ ), où  $p$  est l'ordre de la partie AR et  $q$  l'ordre de la partie MA.

À l'aide d'un programme Python, nous allons tester la combinaison optimale des ordres possibles (1 à 12 pour AR et de 1 à 6 pour MA) utilisant le Critère d'Information d'Akaike (AIC). L'AIC est égal à :

$$AIC = \frac{1}{n} L_n^0(\hat{\theta}_n(p, q)) + \frac{2(p+q)}{n}$$

Ce critère est basé sur la log-vraisemblance pénalisée. Le premier terme est lié à la log-vraisemblance du modèle, qui mesure à quel point le modèle s'ajuste bien aux données. Un meilleur ajustement donne une log-vraisemblance plus élevée.

Le deuxième terme représente la pénalité pour la complexité du modèle. Plus il y a de paramètres dans le modèle, plus cette pénalité sera élevée.

Le principe de l'AIC est donc de trouver un équilibre entre la qualité de l'ajustement du modèle et la complexité de ce modèle. Il pénalise les modèles qui ont plus de paramètres (plus complexes) tout en récompensant un bon ajustement aux données. Ci-dessous le programme Python permettant de déterminer les meilleurs paramètres.

```
#Definir les ordres p et q pour les modeles ARMA
p-values = np.arange(1,12)
q-values = np.arange(1,6)

#Initialiser une liste pour stocker les resultats
results = []

#Boucle a travers les combinaisons d'ordres p et q
for p in p-values:
    for q in q-values:
        try:
            #Ajuster le modele ARMA
            model = sm.tsa.ARIMA(diff2_Dgj_exp[ 'Consommation brute gaz totale (MW PCS 0C)' ], order=(p,0,q))
            result = model.fit()
            #Enregistrer les resultats (AIC et autres metriques si necessaire)
            results.append({ 'Order': (p,0,q), 'AIC': result.aic })
        except Exception as e:
            #Gerer les erreurs eventuelles lors de l'ajustement du modele
            print(f"ARMA({p},{q}) a echoue : {e}")

#Creer un DataFrame a partir des resultats
results_df = pd.DataFrame(results)

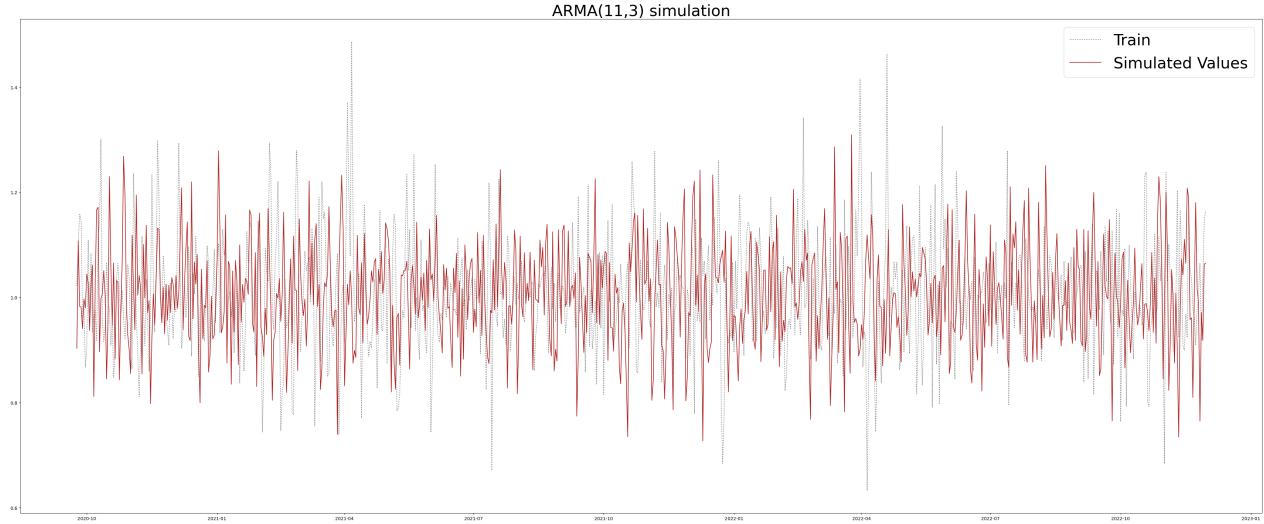
#Trouver le modele avec le plus petit AIC
best_model = results_df.loc[ results_df[ 'AIC' ].idxmin() ]

#Imprimer les resultats
print("Meilleur modele (ordre p, q) : ", best_model[ 'Order' ])
print("AIC correspondant : ", best_model[ 'AIC' ])
```

L'exécution du code nous renvoie :

```
Meilleur modèle (ordre p, q) : (11, 0, 3)
AIC correspondant : -6830.003043355977
```

Nous allons donc tester le modèle ARMA pour les paramètres définis  $p = 11$  et  $q = 3$ .



### Prédictions sur la série stationnaire

Ce modèle nous fournit le sommaire suivant :

	coef	std err	z	P> z	[0.025	0.975]
const	1.0046	0.000	2182.516	0.000	1.004	1.006
ar.L1	-0.0332	0.165	-0.201	0.841	-0.357	0.290
ar.L2	0.3603	0.131	2.754	0.006	0.104	0.617
ar.L3	0.4217	0.145	2.908	0.004	0.137	0.706
ar.L4	-0.1781	0.050	-3.567	0.000	-0.276	-0.080
ar.L5	0.0434	0.025	1.764	0.078	-0.005	0.092
ar.L6	0.0099	0.023	0.435	0.664	-0.035	0.054
ar.L7	0.0792	0.022	3.675	0.000	0.037	0.121
ar.L8	-0.0088	0.020	-0.431	0.666	-0.049	0.031
ar.L9	-0.0594	0.020	-2.999	0.003	-0.098	-0.021
ar.L10	-0.0466	0.020	-2.341	0.019	-0.086	-0.008
ar.L11	0.0623	0.019	3.350	0.001	0.026	0.099
ma.L1	0.2249	0.166	1.359	0.174	-0.100	0.549
ma.L2	-0.5037	0.117	-4.309	0.000	-0.733	-0.275
ma.L3	-0.6322	0.154	-4.103	0.000	-0.934	-0.330
sigma2	0.0083	0.000	50.085	0.000	0.008	0.009

Ainsi, nous pouvons établir l'équation du modèle ARMA(11,3) :

$$\begin{aligned}
 X_t = & 1.0046 - 0.0332X_{t-1} + 0.3603X_{t-2} + 0.4217X_{t-3} - 0.1781X_{t-4} + 0.0434X_{t-5} + 0.0099X_{t-6} \\
 & + 0.0792X_{t-7} - 0.0088X_{t-8} - 0.0594X_{t-9} - 0.0466X_{t-10} + 0.0623X_{t-11} \\
 & + 0.2249Z_{t-1} - 0.5037Z_{t-2} - 0.6322Z_{t-3} + \epsilon_t
 \end{aligned}$$

## 4 Étude des résidus

### 4.1 Outils d'évaluation des résidus

Nous allons maintenant étudier les résidus de nos modèles MA, AR et ARMA afin d'évaluer leur ajustement. Les résidus fournissent des informations sur la qualité de l'ajustement du modèle aux données.

Plusieurs indicateurs peuvent être utilisés pour évaluer les résidus. Parmi eux, le test de Durbin-Watson, le diagramme Quantile-Quantile (Q-Q plot), l'histogramme des résidus, ou encore la fonction d'autocorrélation.

## Le test de Durbin-Watson

Dans le cadre de prédictions avec des séries temporelles, il est important de déceler la présence d'autocorrélation dans les résidus. En effet, une corrélation entre les valeurs de la série temporelle et les valeurs précédentes peut influencer les résultats des analyses statistiques.

Le test de Durbin-Watson est alors utilisé pour détecter la présence d'autocorrélation dans les résidus d'un modèle. L'autocorrélation des résidus signifie que les erreurs du modèle ne sont pas indépendantes les unes des autres à travers le temps.

Dans le contexte de la prédiction de séries temporelles, cela peut entraîner une sous-estimation ou une sur-estimation de la variabilité des prédictions, ce qui compromet la fiabilité des prévisions.

### Fonctionnement du test de Durbin-Watson

- **Calcul de la statistique de test :** La statistique de test de Durbin-Watson est calculée à partir des résidus du modèle. La formule prend en compte la somme des carrés des différences entre les résidus successifs :

$$DW = \sum_{t \in Z} \frac{(\varepsilon_t - \varepsilon_{t-1})^2}{\varepsilon_t^2} \text{ où } \varepsilon_t \text{ correspond au résidu estimé du modèle}$$

- **Plage de valeurs :** La statistique de test varie entre 0 et 4. Une valeur de 2 indique l'absence d'autocorrélation de premier ordre (indépendance des résidus). Les valeurs proches de 0 indiquent une autocorrélation positive, tandis que les valeurs proches de 4 indiquent une autocorrélation négative.

### Interprétation du test

Une valeur proche de 2 suggère que les résidus sont indépendants, ce qui est souhaitable. Des valeurs significativement inférieures à 2 indiquent une autocorrélation positive. Des valeurs significativement supérieures à 2 indiquent une autocorrélation négative.

### Implémentation du test sur Python

```
#Calcul de la statistique de test Durbin-Watson
def calculer_statistique_durbin_watson(residus):
    num = np.sum(np.diff(residus) ** 2)
    denom = np.sum(residus ** 2)
    dw = num / denom
    return dw

#Utilisation de Statsmodels pour effectuer le test de Durbin-Watson
def tester_durbin_watson(residus):
    dw_statistique = calculer_statistique_durbin_watson(residus)
    print(f"Statistique de test Durbin-Watson : {dw_statistique}")

#Affichage d'un message interprétatif
if dw_statistique < 1.8:
    print("Autocorrelation positive possible.")
elif dw_statistique > 2.2:
    print("Autocorrelation négative possible.")
else:
    print("Pas d'autocorrelation détectée.")
```

### **Le diagramme Quantile-Quantile**

Le diagramme Quantile-Quantile, aussi appelé Q-Q plot, évalue la normalité des résidus du modèle. Il compare la distribution des résidus avec la distribution théorique attendue, généralement une distribution normale. Si les points sur le graphique s'écartent de la droite correspondant à la première bissectrice du plan, cela suggère que la distribution des résidus n'est pas parfaitement normale.

### **L'histogramme des résidus**

Les résidus sont regroupés en intervalles de valeurs, et un histogramme est construit pour représenter la fréquence d'occurrence de ces intervalles.

Tous ces indicateurs sont décisifs dans l'évaluation de la fiabilité de nos modèles pour notre série temporelle.

## **4.2 Résidus des modèles étudiés**

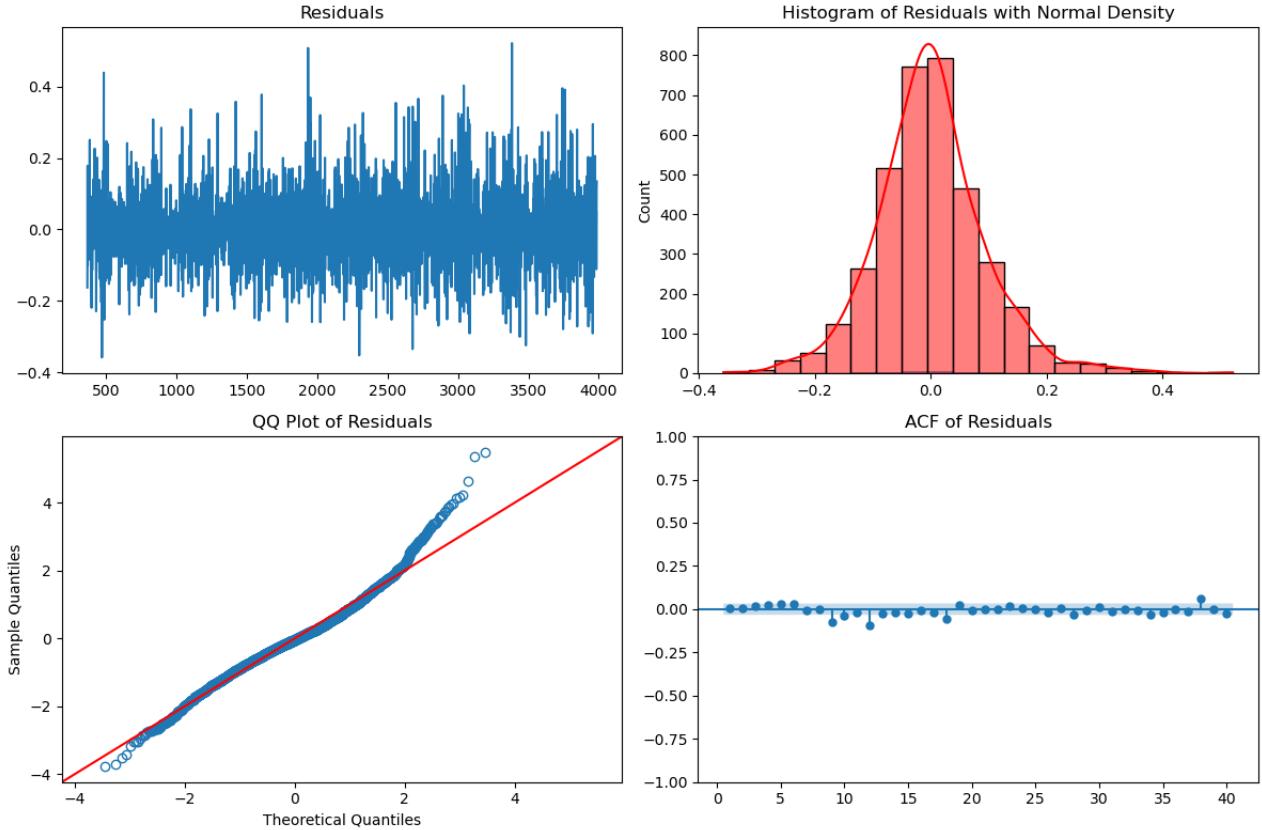
Nous allons étudier les résidus de chaque modèle afin de déterminer s'ils sont réellement adaptés. Pour ce faire, il nous faudra vérifier :

- Pour le modèle MA : la normalité de nos résidus ainsi que l'absence de corrélation afin qu'il corresponde à un bruit blanc.
- Pour le modèle AR, il suffit que nos résidus soient un bruit blanc et donc qu'il n'y ait pas de corrélation.
- Pour le modèle ARMA, nous allons vérifier que nos résidus sont un bruit blanc gaussien.

Nous allons dorénavant afficher les différents indicateurs évoqués précédemment pour chacun des modèles étudiés.

## Résidus du modèle MA :

Affichons les résidus, l'histogramme, le Q-Q plot, la fonction d'autocorrélation, ainsi que le résultat du test de Durbin-Watson de notre modèle MA(6) :



### Étude des résidus du modèle MA(6)

Le diagramme Q-Q et l'histogramme montrent qu'on peut approximer le modèle par une loi gaussienne centrée. On a donc  $E[\varepsilon_t] = 0$ . De plus le graphique de l'ACF nous laisse penser qu'il n'y a pas de corrélation entre nos variables résiduelles.

Nous allons confirmer ces dires par le test de Durbin Watson 4.1 :

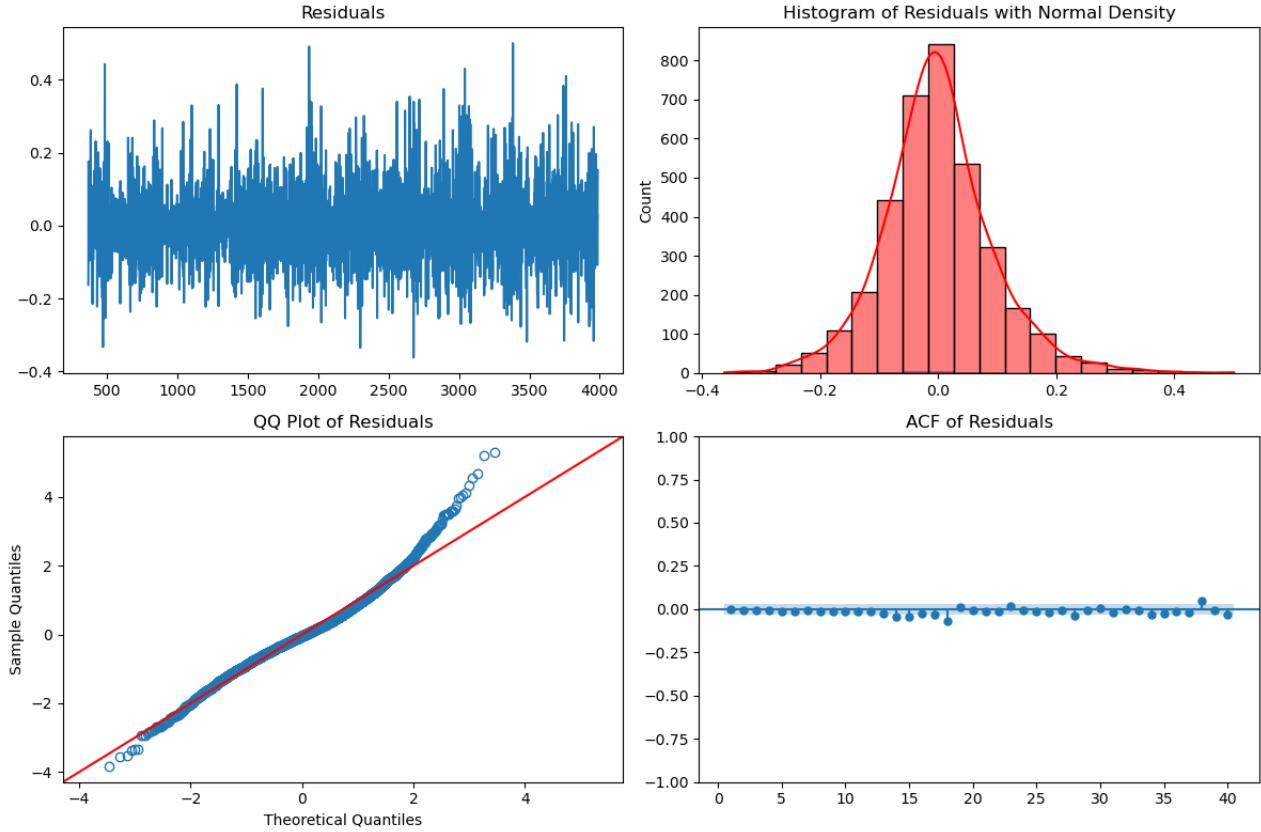
```
tester_durbin_watson(arima_residuals_ma_6.dropna())
Statistique de test Durbin-Watson : 1.9895156240760057
Pas d'autocorrélation détectée.
```

Il n'y a pas d'autocorrélation.

Nous déduisons que notre modèle est un bruit blanc  $BB(\sigma^2 = 0.0085)$ . Le modèle MA(6) semble donc adapté.

## Résidus du modèle AR :

Étudions les résidus de notre modèle AR(12).



### Étude des résidus du modèle AR(12)

Le diagramme Q-Q et l'histogramme montrent qu'on peut approximer le modèle par une loi gaussienne centrée. On a donc  $E[\varepsilon_t] = 0$ . De plus le graphique de l'ACF nous laisse penser qu'il n'y a pas de corrélation entre nos variables résiduelles.

Nous allons confirmer ces dires par le test de Durbin Watson 4.1 :

```
tester_durbin_watson(arima_residuals_ar_12.dropna())
```

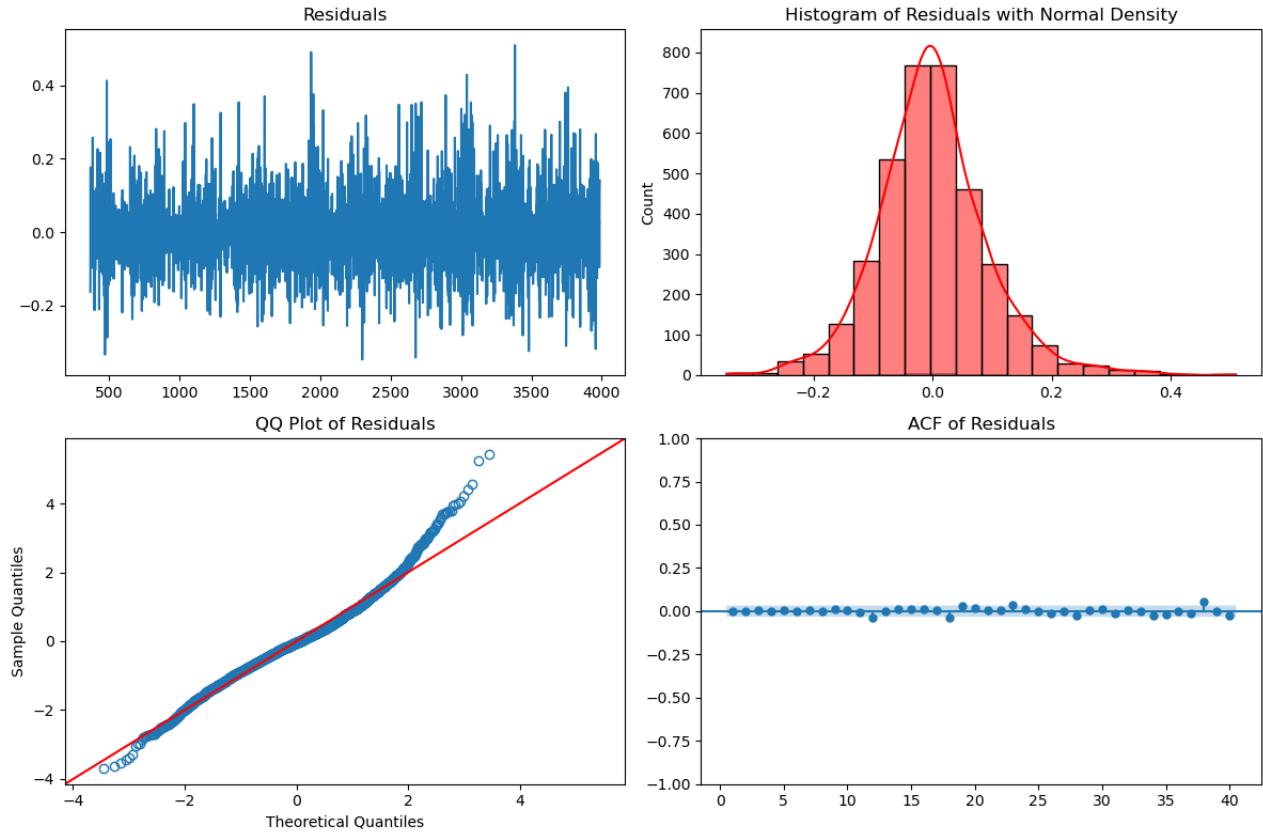
```
Statistique de test Durbin-Watson : 2.0028059801700873
Pas d'autocorrélation détectée.
```

Il n'y a pas d'autocorrélation.

Nous déduisons que notre modèle est un bruit blanc  $BB(\sigma^2 = 0.0084)$ . Le modèle AR(12) semble donc adapté.

## Résidus du modèle ARMA :

Étudions les résidus de notre modèle ARMA(11,3) :



### Étude des résidus du modèle ARMA(11,3)

Le diagramme Q-Q et l'histogramme montrent qu'on peut approximer le modèle par une loi gaussienne centrée. On a donc  $E[\varepsilon_t] = 0$ . De plus le graphique de l'ACF nous laisse penser qu'il n'y a pas de corrélation entre nos variables résiduelles.

Nous allons confirmer ces dires par le test de Durbin Watson 4.1 :

```
tester_durbin_watson(arima_residuals_11_3.dropna())
```

```
Statistique de test Durbin-Watson : 2.0032676152105595
Pas d'autocorrélation détectée.
```

Il n'y a pas d'autocorrélation.

Nous déduisons que notre modèle est un bruit blanc gaussien  $BB(\sigma^2 = 0.0083)$ . Le modèle ARMA(11,3) semble donc adapté.

## 5 Conclusion

Nos graphiques de prédition précédemment affichés sont valables pour la série stationnaire  $(X_t)_{t \in \mathbb{Z}}$ . Or, nous souhaitons obtenir les prédictions dans le cadre de nos données réelles  $(D_t)_{t \in \mathbb{Z}}$ .

Nous allons réaliser les prédictions sur le meilleur modèle parmi ceux que nous avons étudié. Pour ce faire, comparons l'AIC 3.3 de chaque modèle obtenu :

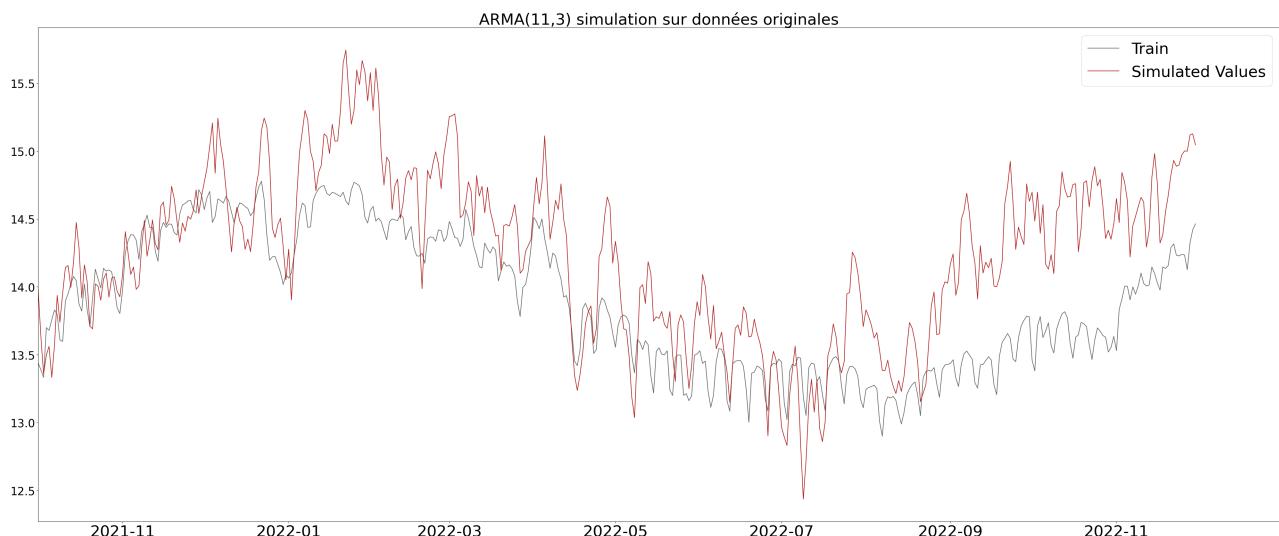
Modèle MA(6) :  $AIC = -5430.220510723172$

Modèle AR(12) :  $AIC = -5443.99965753198$

Modèle ARMA(11, 3) :  $AIC = -6830.003043355977$

Ainsi, le meilleur modèle semble être le modèle ARMA(11, 3).

Affichons les prédictions réalisées sur notre série temporelle  $(D_t)_{t \in \mathbb{Z}}$  :



Prédictions avec le modèle ARMA(11,3) sur la série temporelle

Avec une racine de l'erreur quadratique moyenne (RMSE) valant 0.5259759656802978. Plus la RMSE est basse, meilleure est la précision de notre modèle. Celle-ci est correcte par rapport à nos données de l'ordre de  $10^1$ .

En vertu de ces résultats, la prédition de la consommation brute de gaz totale en France, dans la période allant d'octobre 2021 à décembre 2022, réalisée à l'aide du modèle ARMA(11,3) est assez précise.

Cependant, ajouter des variables explicatives extérieures comme les températures ou encore les crises énergétiques augmenterait davantage la précision de nos prédictions.