exercice

April 19, 2020

1 Préambule

Dès 1958, Charles David Keeling a débuté la mesure précise du taux de CO2 dans l'atmosphère à l'observatoire de Mauna Loa, Hawaii, États-Unis.
Ces mesures, qui continuent aujourd'hui, ont permis de montrer une évolution périodique de CO2 dans l'hémisphère Nord. Celle-ci provenant du cycle de vie des plantes.
De même, ces données ont montré une évolution continue du taux de CO2 dans l'atmosphère depuis 1958.

2 Travail à faire

Le but de l'exercice est de réaliser un document computationnel pour : * Réaliser un graphique qui montrera une oscillation périodique superposée à une évolution systématique plus lente. * Séparer ces deux phénomènes. Caractériser l'oscillation périodique et proposer un modèle simple de la contribution lente * Estimer ses paramètres et tenter une extrapolation jusqu'à 2025 (dans le but de pouvoir valider le modèle par des observations futures). * Déposer dans FUN le résultat.

3 Base de données

Les données sont disponibles sur le site Web de l'institut Scripps à l'adresse suivante:
https://scrippsco2.ucsd.edu/data/atmospheric_co2/primary_mlo_co2_record.html

Pour notre étude, nous prendrons les relevés hebdomadaires : *weekly_in_situ_co2_mlo.csv* que l'on peut télécharger à l'adresse suiavnte : https://scrippsco2.ucsd.edu/data/atmospheric_co2/mlo.html

Nous travaillerons sur une base locale (copiée sur le serveur *jupyter* de l'INRIA) téléchargée le 13 avril 2020. La totalité des documents nécessaires à cette étude seront committés sur le serveur *GitLab* de l'INRIA.

Les parties du code nécessaires à l'affichage des courbes ne sera pas affichées dans le rapport final. Elles seront tout de même accessibles dans le fichier *jupyter*.

4 Vérification de la base de données

4.1 Exploration des données

Nous commençons par analyser le contenu du fichier de données (fichier structuré *CSV*) pour ensuite faire un premier tracé de l'ensemble de la base de données.
Nous utiliserons les libraries *pandas* et *matplotlib* pour *python 3.6*.

[1]:
```
# Import des librairies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
# Chargement de la base de données (CSV)
# Les lignes de commentaires sont ignorées
# Le séparateur de champs utilisé dans la base de données est la virgule (,)
# La colonne des dates est parsé et mise en index
date_parser = lambda dates: [pd.datetime.strptime(d, '%Y-%m-%d') for d in dates]

data = pd.read_csv('weekly_in_situ_co2_mlo.csv', sep=',', comment='"',␣
↪header=None,
                   names=["date", "CO2"])
data['date'] = pd.to_datetime(data['date'], format='%Y-%m-%d')

# Les dates sont passées à l'index du dataframe
data.set_index('date', inplace=True)
data.shape
```

[2]:
```
(3156, 1)
```

Les 44 premières lignes de commentaires du fichier brut n'ont pas été prises en compte (commande `comment='"'`).
La base de données est composée de 3156 mesures et de 2 colonnes : * date : date de relevé. Avec un relevé par semaine. * CO2 : concentration de CO2 en *ppm*.

[3]:
```
# Affichage partiel de la base de données mise en forme
data.head(5)
```

[3]:

date	CO2
1958-03-29	316.19
1958-04-05	317.31
1958-04-12	317.69
1958-04-19	317.58
1958-04-26	316.48

L'inspection visuelle de la base de données complète montre que le fichier est bien formé et ne semble pas comporter de valeurs manquantes ou aberrantes.

4.2 Vérification des données

Avant d'aller plus loin dans l'analyse, vérifions avec un code de validation (une fonction *builtin* de *pandas*) la robustesse de la base de données. Nous vérifions la présence de valeurs manquantes et le type des différentes variables (date, float).
Par la suite, l'affichage sous forme graphique nous permettra d'avoir une vue globale de la base de données.

[4]:
```
# La méthode .info() permet d'avoir une vision consice de la base de données
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3156 entries, 1958-03-29 to 2020-02-01
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   CO2     3156 non-null   float64
dtypes: float64(1)
memory usage: 49.3 KB
```

La base de données comporte 3156 lignes de valeurs numériques (float64) sans valeur manquante. L'index est bien au format `datetime` allant du 29/03/1958 au 01/02/2020.

Nous pouvons considérer que la base de données est cohérente. L'analyse est possible.

5 Analyse des données

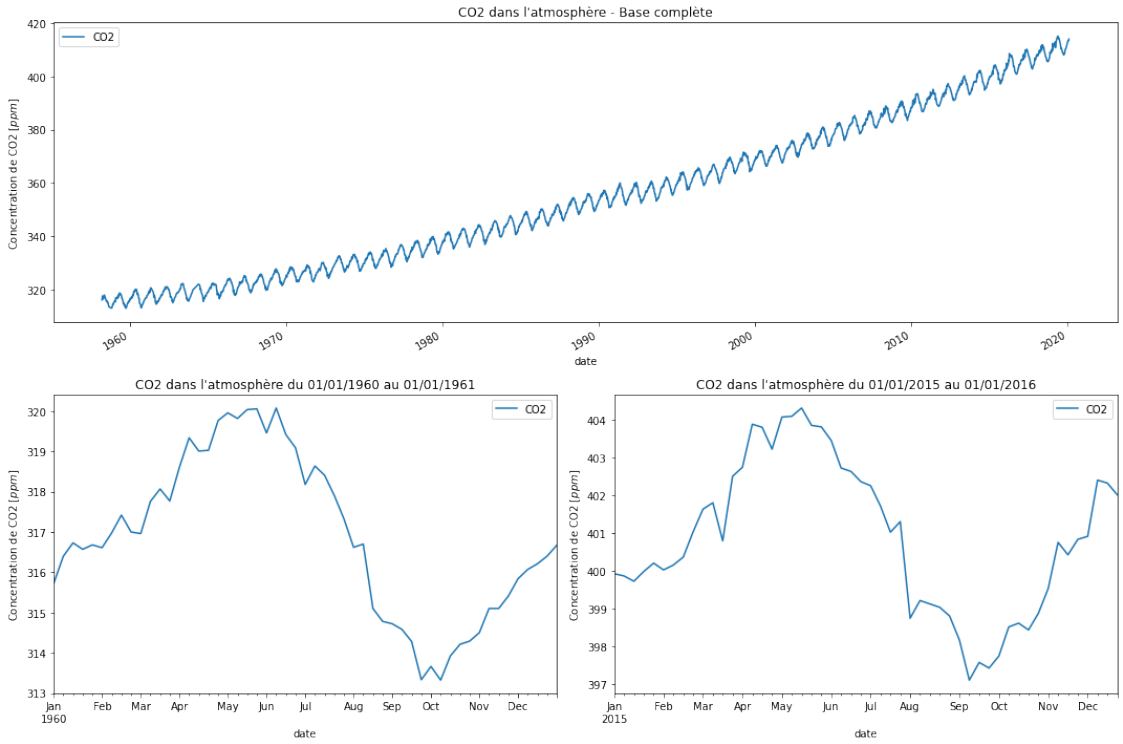
5.1 Mise en évidence des composantes de l'évolution de la concentration de CO2 dans l'atmosphère

Un tracé sous forme graphique nous permettra de constater les phénomènes.
Après l'affichage de la base complète, nous tracerons 2 sous-graphiques correspondant à l'évolution du taux de CO2 sur 2 années différentes. Une en début de base (du 01/01/1960 au 01/01/1961) et une autre en fin (du 01/01/2015 au 01/01/2016).

[5]:
```
# Définition des périodes à afficher
mask1_1y = (data.index > '1960-01-01') & (data.index <= '1961-01-01')
mask2_1y = (data.index > '2015-01-01') & (data.index <= '2016-01-01')
```

[6]:
```
# Initialisation des graphs
# Graphs de 2 périodes différentes de 1 an
fig = plt.figure()

# Définition des axes
ax1 = plt.subplot(223)
ax2 = plt.subplot(224)
ax3 = plt.subplot(211)
```

```
# Titres des graphs
ax1.set_title("CO2 dans l'atmosphère du 01/01/1960 au 01/01/1961")
ax2.set_title("CO2 dans l'atmosphère du 01/01/2015 au 01/01/2016")
ax3.set_title("CO2 dans l'atmosphère - Base complète")

# Labels des axes
ax1.set_ylabel("Concentration de CO2 [$ppm$]")
ax2.set_ylabel("Concentration de CO2 [$ppm$]")
ax3.set_ylabel("Concentration de CO2 [$ppm$]")

# Graph sur la base complète
data[mask1_1y].plot(ax=ax1, figsize=(15,10))
data[mask2_1y].plot(ax=ax2, figsize=(15,10))
data.plot(ax=ax3, figsize=(15,10))

# Optimisation de l'espacement entre les graphs
fig.tight_layout()
```



Nous constatons que l'évolution continue du CO2 dans l'atmosphère ne suit pas une courbe linaire. Le tendance est plutôt quadratique.
Sur année, l'évolution est périodique est semble grossièrement coller à une sinusoïde.

5.2 Caractérisation de la tendance continue

Pour cela, nous allons ajuster une courbe quadratique à la base complète en utilisant la librairie *lmfit*.
Cette courbe suivra la tendance générale des données initiales en "annulant" les oscillations périodiques et donnera donc la tendance à long terme.

Nous utiliserons le coefficient de détermination (r^2) pour estimer la qualité de la régression.

[7]:
```
# Installation de la librairie lmfit
!pip install lmfit
```

[8]:
```
# Import des librairies
from lmfit.models import QuadraticModel, Model
import lmfit
from sklearn.metrics import r2_score
```

Les index au format `dateindex` sont mal pris en compte par la librairie *lmfit*. Il est nécessaire de se recréer une échelle de temps permettant les calculs.
Cette même fonction servira pour tous les calculs à venir.

[9]:
```
# Fonction permettant de définir une échelle de temps en lieu et place de␣
 ↪l'index temporel de pandas
# Cela est nécessaire pour contourner un bug de la librairie lmfit
def dates_to_idx(timelist):
    reference_time = pd.to_datetime('1958-03-29')
    time = (timelist - reference_time) / pd.Timedelta(1*365.25, "D")
    return np.asarray(time)

time = dates_to_idx(data.index)
```

[10]:
```
# Instanciation du modèle
model_long_trend = QuadraticModel()
params_long_trend = model_long_trend.guess(data['CO2'], x=time)

result_long_trend = model_long_trend.fit(data['CO2'], params_long_trend, x=time)
```

[11]:
```
# Mise en dictionnaire des meilleures paramètres calculés
coeffs_long_trend = result_long_trend.params.valuesdict()
```

[12]:
```
#Affichage des meilleures paramètres calculés avec les intervalles de confiances
print(lmfit.fit_report(result_long_trend.params))
```

```
[[Variables]]
    a:  0.01297799 +/- 1.4176e-04 (1.09%) (init = 0.01297799)
    b:  0.76780276 +/- 0.00913486 (1.19%) (init = 0.7678028)
    c:  314.570495 +/- 0.12419930 (0.04%) (init = 314.5705)
[[Correlations]] (unreported correlations are < 0.100)
```

```
C(a, b) = -0.969
C(b, c) = -0.873
C(a, c) =  0.756
```

Ci-dessus, les différents paramètres de la courbe d'ajustement à long terme (fonction quadratique).

[13]:
```
# Fonction de prédiction à long terme
def long_trend_fitted_curve(time):
    long_trend_CO2 = (coeffs_long_trend.get('a')*time*time +
                      coeffs_long_trend.get('b')*time +
                      coeffs_long_trend.get('c'))
    return np.round(long_trend_CO2, 2)
```

[14]:
```
# Ajout d'une colonne dans le dataframe data
data['long_trend_fit'] = pd.Series(long_trend_fitted_curve(time), index=data.
 ↪index)
```

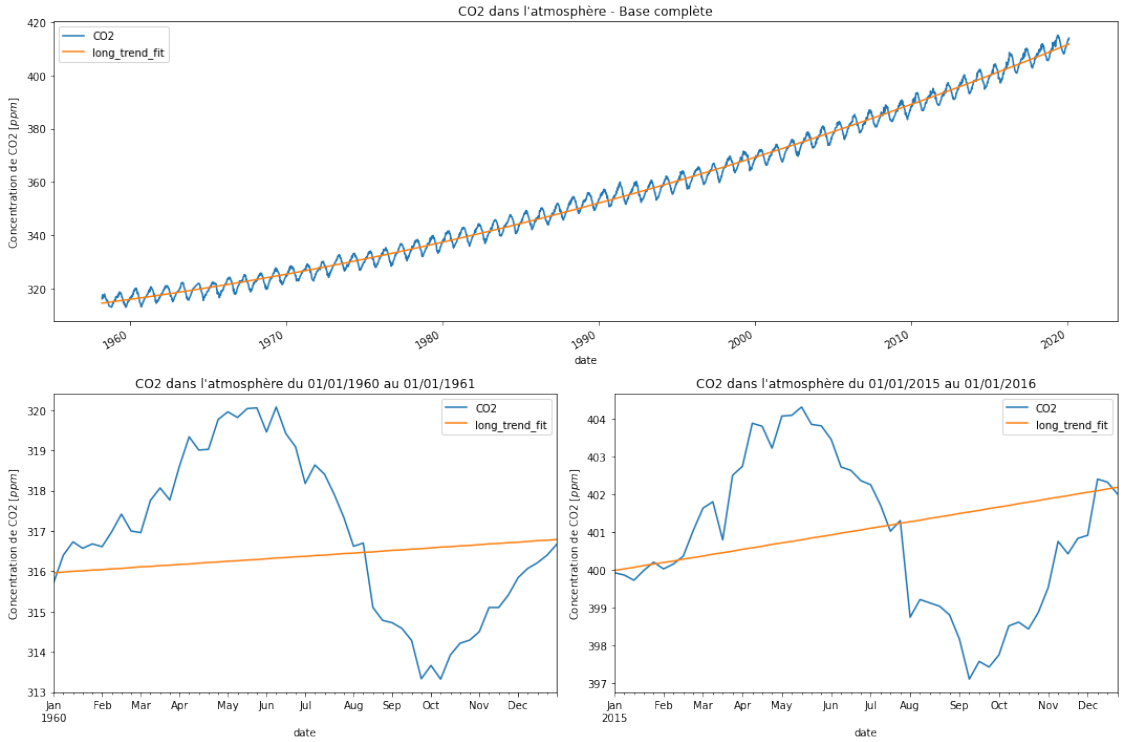
[15]:
```
# Estimation du coefficient de détermination
r2_score(data['long_trend_fit'], data['CO2'])
```

[15]:
```
0.9936252499490063
```

Le coefficient de détermination est très bon.
Ce modèle de prédiction simple (càd sans prendre en compte les oscillations saisonnières) suffirait à calculer de bonnes prédictions de CO2 pour les années futures. Nous allons tout de même caractériser les oscillations saisonnières.

[16]:
```
# Graphs de 2 périodes distinctes de 1 an
# Avec la courbe de tendance
fig = plt.figure()

# Définition des axes
ax1 = plt.subplot(223)
ax2 = plt.subplot(224)
ax3 = plt.subplot(211)

# Titres des graphs
ax1.set_title("CO2 dans l'atmosphère du 01/01/1960 au 01/01/1961")
ax2.set_title("CO2 dans l'atmosphère du 01/01/2015 au 01/01/2016")
ax3.set_title("CO2 dans l'atmosphère - Base complète")

# Labels des axes
ax1.set_ylabel("Concentration de CO2 [$ppm$]")
ax2.set_ylabel("Concentration de CO2 [$ppm$]")
ax3.set_ylabel("Concentration de CO2 [$ppm$]")

# Graph sur la base complète
data[mask1_1y].plot(ax=ax1, figsize=(15,10))
```

```
data[mask2_1y].plot(ax=ax2, figsize=(15,10))
data.plot(ax=ax3, figsize=(15,10))

# Optimisation de l'espacement entre les graphs
fig.tight_layout()
```



L'intervalle de confiance de la courbe ajustée n'est pas tracée sur les graphes précédents car il est très etroit.
Celui-ci dégraderait la lisibilité des graphiques.

5.3 Caractérisation de l'oscillation périodique

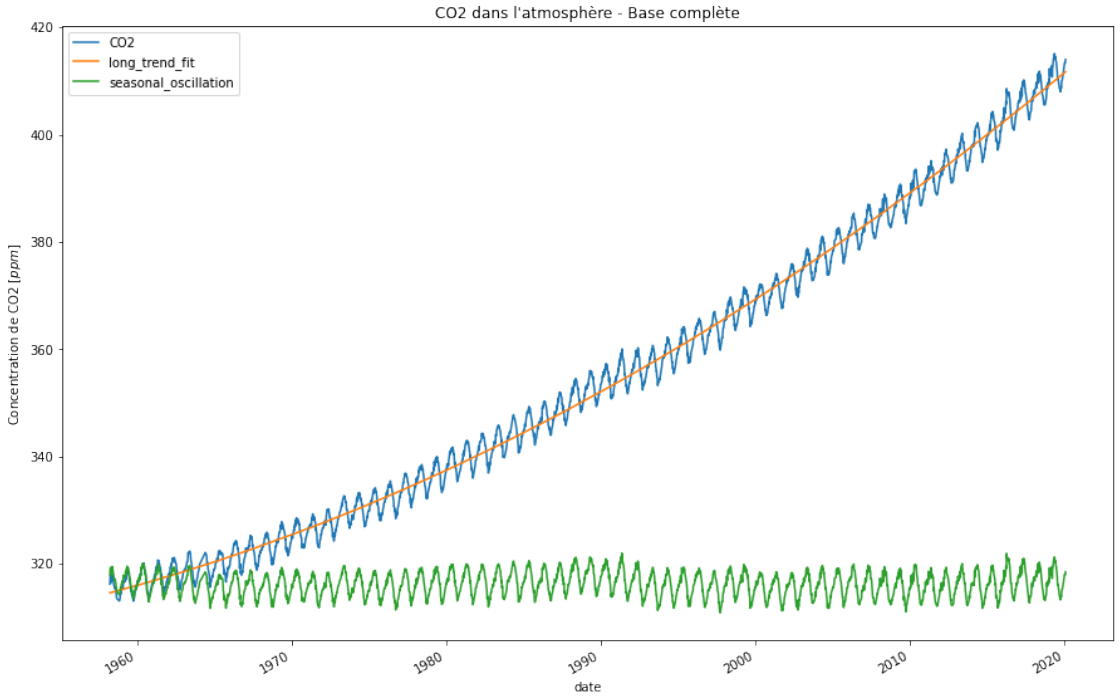
Pour avoir un aperçu de l'oscillation saisonnière, commençons par retraiter les données pour soustraire l'effet de la tendance continue.

[17]:
```
# Calculs des données en soustrayant la tendance continue
data['seasonal_oscillation'] = data['CO2'] - data['long_trend_fit'] +␣
↪data['CO2'][0]
```

[18]:
```
# Initialisation du graph
fig = plt.figure()

# Définition des axes
```

```
ax = plt.subplot(111)

# Titre du graph
ax.set_title("CO2 dans l'atmosphère - Base complète")

# Label des axes
ax.set_ylabel("Concentration de CO2 [$ppm$]")

# Graph sur la base complète
data.plot(ax=ax, figsize=(15,10))
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce32bca518>



La courbe verte représente les données de la variation de CO2 dans l'atmosphère sans l'effet de l'évolution continue que nous avons constaté au début de l'analyse.
Nous constatons qu'en plus des oscillations saisonnières (période de 1 an), il semble avoir une oscillation, à faible amplitude, avec une période de l'ordre de 30 ans. Dans la suite de l'analyse, nous ne prendrons pas en compte cette variation. En effet, avec une simple régression quadratique sur la tendance continue, le résultat est déjà très bon.
Nous nous contenterons donc de caractériser les variations saisonnières avec une fonction sinus ajustée à toute la base de données. Cela aura pour effet d'avoir un modèle "ne fittant" pas parfaitement avec les données mais qui aura probablement plus de capacité à "généraliser" pour de bonnes prédictions futures. C'est le but premier de l'exercice.

[19]:
```
# Définition de la fonction sinus
def sine(x, amp, freq, shift):
    return amp * np.sin(2*np.pi*x*freq) + shift
```

[20]:
```
# Instanciation du modèle
model_seasonal_oscillation = Model(sine)
params_seasonal_oscillation = model_seasonal_oscillation.make_params(amp=7,␣
 ↪freq=1, shift = 0)

result_seasonal_oscillation = model_seasonal_oscillation.
 ↪fit(data['seasonal_oscillation'], params_seasonal_oscillation, x=time)
```

[21]:
```
# Affichage des meilleures paramètres calculés
coeffs_seasonal_oscillation = result_seasonal_oscillation.params.valuesdict()
```

[22]:
```
#Affichage des meilleures paramètres calculés avec les intervalles de confiances
print(lmfit.fit_report(result_seasonal_oscillation.params))
```

```
[[Variables]]
    amp:    2.50225387 +/- 0.03484620 (1.39%) (init = 7)
    freq:   1.00450120 +/- 6.1470e-05 (0.01%) (init = 1)
    shift:  316.199477 +/- 0.02463756 (0.01%) (init = 0)
```

Avec ces résulats nous pouvons caractériser l'oscillation saisonnière comme suit : - +/- 2.5 *ppm* sur une année. - Une féquence de 1 année pour chaque oscillation. Ce qui conforme aux observations. - Un décalage de 316.20 *ppm* qui correspond à la première mesure de la base de données.

Les intervalles de confiance sont là aussi très restreints. Nous faisons le choix de ne pas les représenter sur les graphiques à venir.

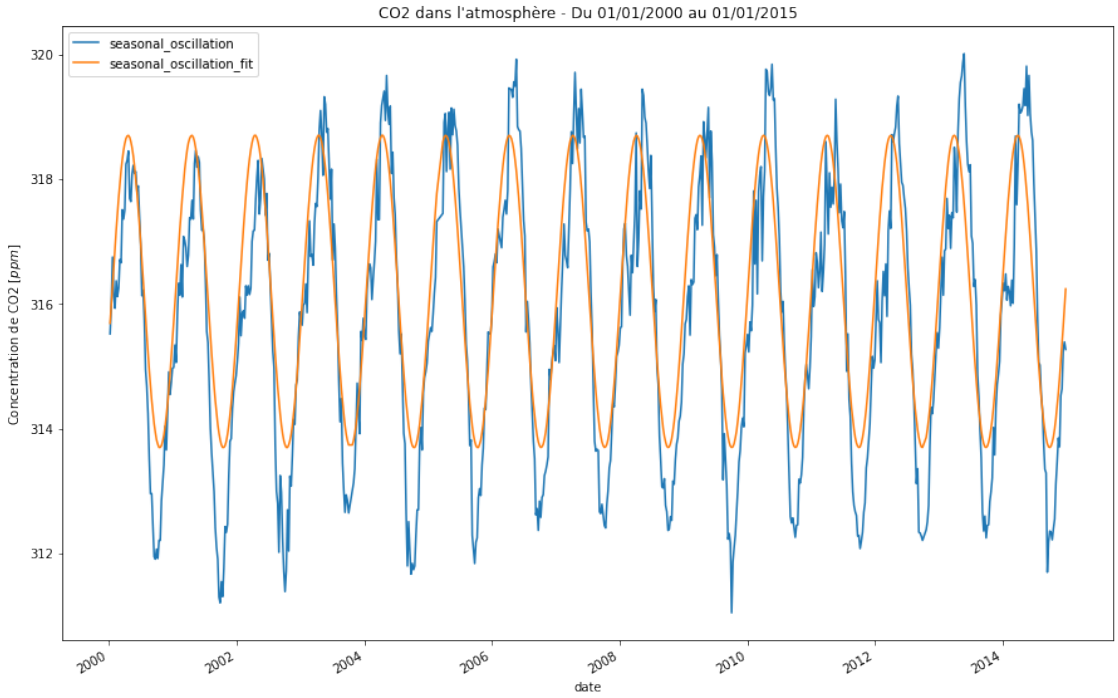
[23]:
```
def seasonal_oscillation_fitted_curve(time):
    seasonal_oscillation_CO2 = (coeffs_seasonal_oscillation.get('amp') *
                                np.sin(2*np.pi*coeffs_seasonal_oscillation.
 ↪get('freq')*time) +
                                coeffs_seasonal_oscillation.get('shift'))
    return np.round(seasonal_oscillation_CO2, 2)
```

[24]:
```
# Ajout d'une colonne dans le dataframe data
data['seasonal_oscillation_fit'] = pd.
 ↪Series(seasonal_oscillation_fitted_curve(time), index=data.index)
```

[25]:
```
# Initialisation du graph
fig = plt.figure()

# Définition des axes
ax = plt.subplot(111)

# Titre du graph
```

```
ax.set_title("CO2 dans l'atmosphère - Du 01/01/2000 au 01/01/2015")

# Label des axes
ax.set_ylabel("Concentration de CO2 [$ppm$]")
data[(data.index > '2000-01-01') & (data.index < '2015-01-01')].drop(['CO2',␣
↪'long_trend_fit'], axis=1).plot(ax=ax, figsize=(15,10))
```

[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fce2136e630>



Comme annoncé précédemment, nous voyons que le modèle basé sur un simple sinus, ne colle pas parfaitement au données.
Cela n'aura probablement pas un fort impact sur le résultat final.

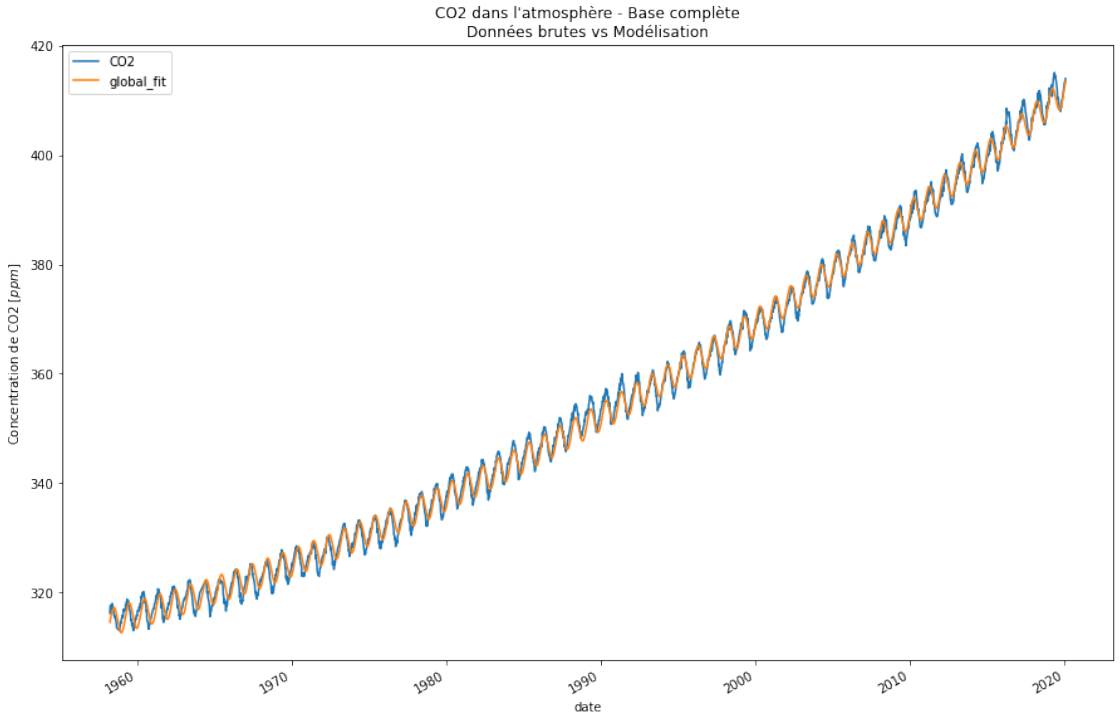
5.4 Définition du modèle global

Dans la suite, nous créeons le modèle prenant en compte l'évolution continue et l'oscillation saisonnière.

[26]:
```
data['global_fit'] = pd.Series(long_trend_fitted_curve(time) +
                               seasonal_oscillation_fitted_curve(time) -
                               data['CO2'][0],
                               index=data.index)
```

[27]:
```
# Initialisation du graph
fig = plt.figure()

# Définition des axes
ax = plt.subplot(111)

# Titre du graph
ax.set_title("CO2 dans l'atmosphère - Base complète\nDonnées brutes vs␣
 ↪Modélisation")

# Label des axes
ax.set_ylabel("Concentration de CO2 [$ppm$]")

# Graph sur la base complète
data.drop(['long_trend_fit', 'seasonal_oscillation',␣
 ↪'seasonal_oscillation_fit'], axis=1).plot(ax=ax, figsize=(15,10))
```

[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce31f67be0>
```



[28]:
```
# Estimation du coefficient de détermination
r2_score(data['CO2'], data['global_fit'])
```

[28]:
```
0.997596481376051
```

L'intégration des oscillations prériodiques permet d'améliorer un peu plus le modèle initial.
Au vu de la très bonne valeur de r^2, nous resterons sur ce modèle pour faire les prédictions pour les années futures.

6 Prédiction pour 2025

Dans les chapitres précedents, nous avons défini un modèle permettant d'intégrer les oscillations saisonnières à la tendance continue. Nous allons donc pouvoir tenter une prédiction de l'évolution du CO2 dans l'atmosphère pour toute l'année 2025.

[29]:
```
# Définition des différentes dates de l'année 2025 à calculer
# Une date par semaine comme pour la base de sonnées initiale
pred_range = pd.date_range(start='2025/01/01', end='2025/12/31', freq='W')
pred_range.shape
```

[29]:
```
(52,)
```

[30]:
```
# Calcul du temps avec la même fonction que pour les calculs des chapitres␣
↪précédents
time = dates_to_idx(pred_range)
```

[31]:
```
# Calcul des préditions sur l'année 2025
# Calcul avec oscillitations périodiques
# Calcul de la tendance continue
pred_2025 = pd.Series(long_trend_fitted_curve(time) +
                              seasonal_oscillation_fitted_curve(time) -
                              data['CO2'][0],
                              index=pred_range)
pred_long_trend_2025 = pd.Series(long_trend_fitted_curve(time),␣
↪index=pred_range)
```

[32]:
```
# Calcul de la valeur moyenne avec le modèle complet
mean_2025 = round(np.mean(pred_2025))
mean_2025
```

[32]:
```
425.0
```

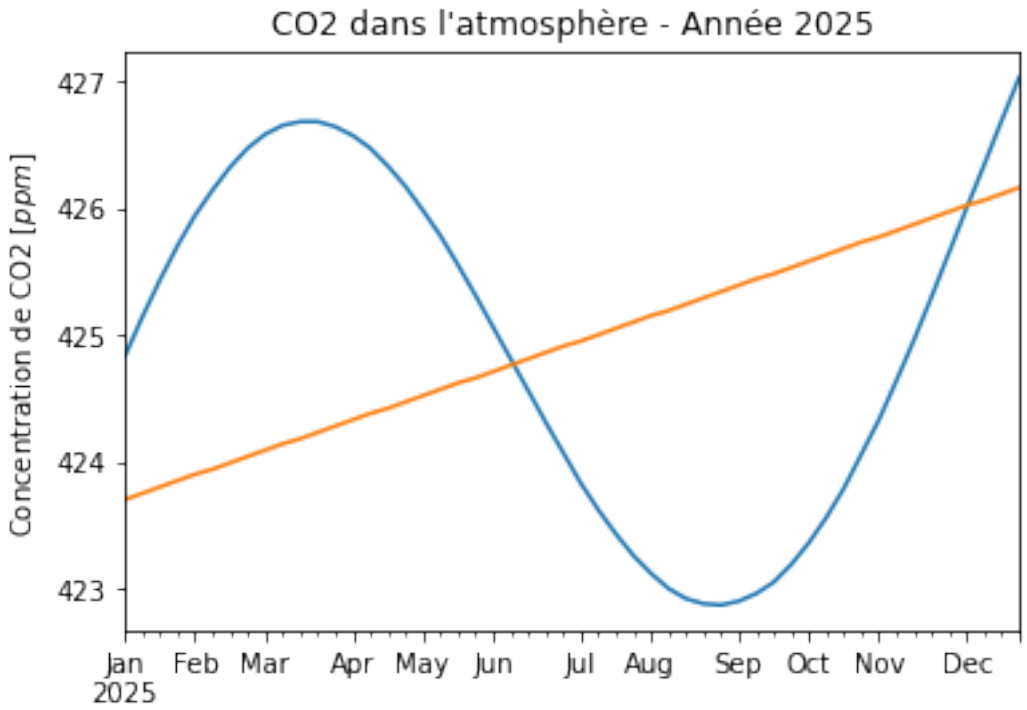
[33]:
```
# Calcul de la valeur moyenne avec le modèle sur la tendance générale
mean_long_trend_2025 = round(np.mean(pred_long_trend_2025))
mean_long_trend_2025
```

[33]:
```
425.0
```

La valeur moyenne attendue de CO2 dans l'atmosphère pour 2025 est de 425 *ppm*.

[34]:
```
# Affichage des résulats
# Initialisation du graph
fig = plt.figure()

# Définition des axes
ax = plt.subplot(111)

# Titre du graph
ax.set_title("CO2 dans l'atmosphère - Année 2025")

# Label des axes
ax.set_ylabel("Concentration de CO2 [$ppm$]")
pred_2025.plot()
pred_long_trend_2025.plot()
```

[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce31e81ac8>
```



L'oscillation saisonnière calculée respecte les caractérisques principales des relevés : - Maximum dans la première moitié de l'année. - Minimum dans la deuxième moitié de l'année. - Valeur au 31/12/2025 supérieure à celle du 01/01/2025.

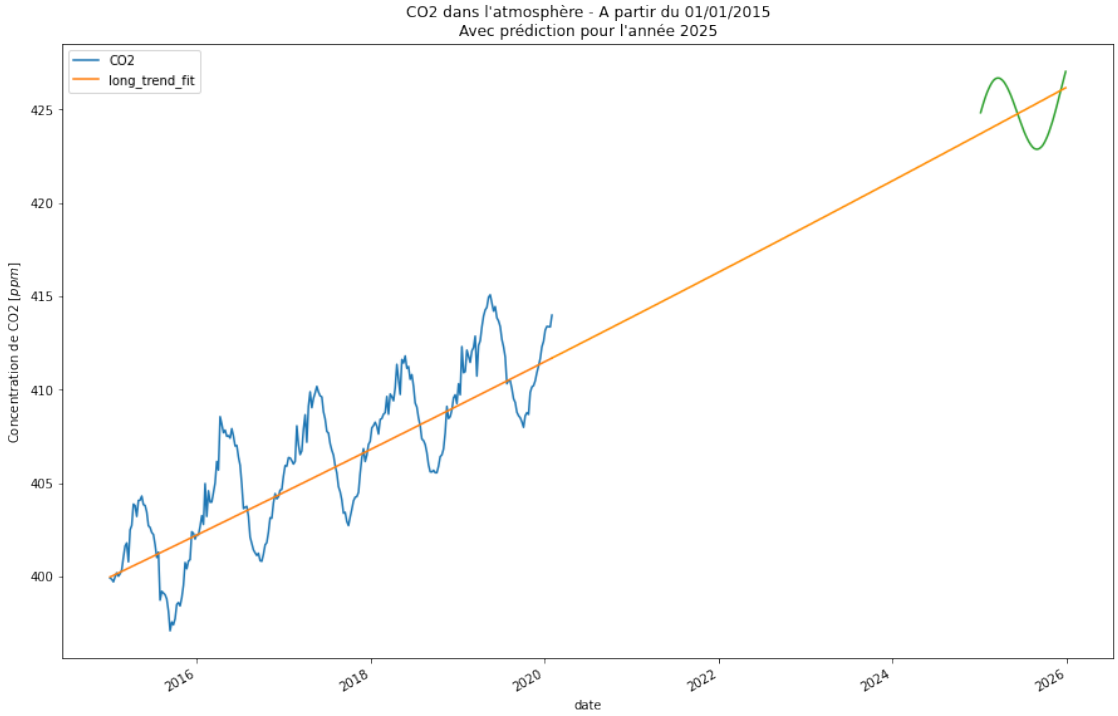
Nous voyons tout de même un déphasage (avance d'environ 1.5 mois) avec les données brutes. Comme nous l'avions pressenti au début de l'analyse, l'intégration des oscillations saisonnières dans le modèle ne permet pas d'améliorer les prédictions. Les 2 modèles (avec et sans intégration

des oscillations saisonnières) donnent le même résulat (425 *ppm* de CO2 dans l'atmosphère).

[35]:
```
# Préparation des échelles de temps pour le tracé des prédictions
hole_range = pd.date_range(start='2020/02/01', end='2025/12/31', freq='W')
time_hole = dates_to_idx(hole_range)

pred_hole_range = pd.Series(long_trend_fitted_curve(time_hole),␣
↪index=hole_range)
```

[36]:
```
# Initialisation du graph
fig = plt.figure(figsize=(15,10))

# Définition des axes
ax = plt.subplot(111)

# Titre du graph
ax.set_title("CO2 dans l'atmosphère - A partir du 01/01/2015\nAvec prédiction␣
↪pour l'année 2025")

# Label des axes
ax.set_ylabel("Concentration de CO2 [$ppm$]")

# Graph sur la base complète
data[data.index > '2015-01-01'].drop(['seasonal_oscillation',␣
↪'seasonal_oscillation_fit', 'global_fit'], axis=1).plot(ax=ax)
pred_2025.plot()
pred_hole_range.plot(color='darkorange')
```

[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce31d6d828>
```



Ci-dessus la prédiction pour l'année 2025 complète (courbe verte) et sa tendance à long terme (courbe orange).
Pour assurer une bonne lisibilité du graphique, les intervalles de confiance (très étroits) ne sont pas représentés.

7 Conclusion

Nous avons vu qu'il est possible de définir un modèle de prédiction crédible et précis avec une simple régression quadratique. L'ajout des oscillations saisonnières permet tout de même d'améliorer très légèrement la qualité des prédictions.
D'un point de vue purement analytique, il serait intéressant de faire une transformée de Fourier sur les données brutes pour caractériser la variation périodique de 30 ans.