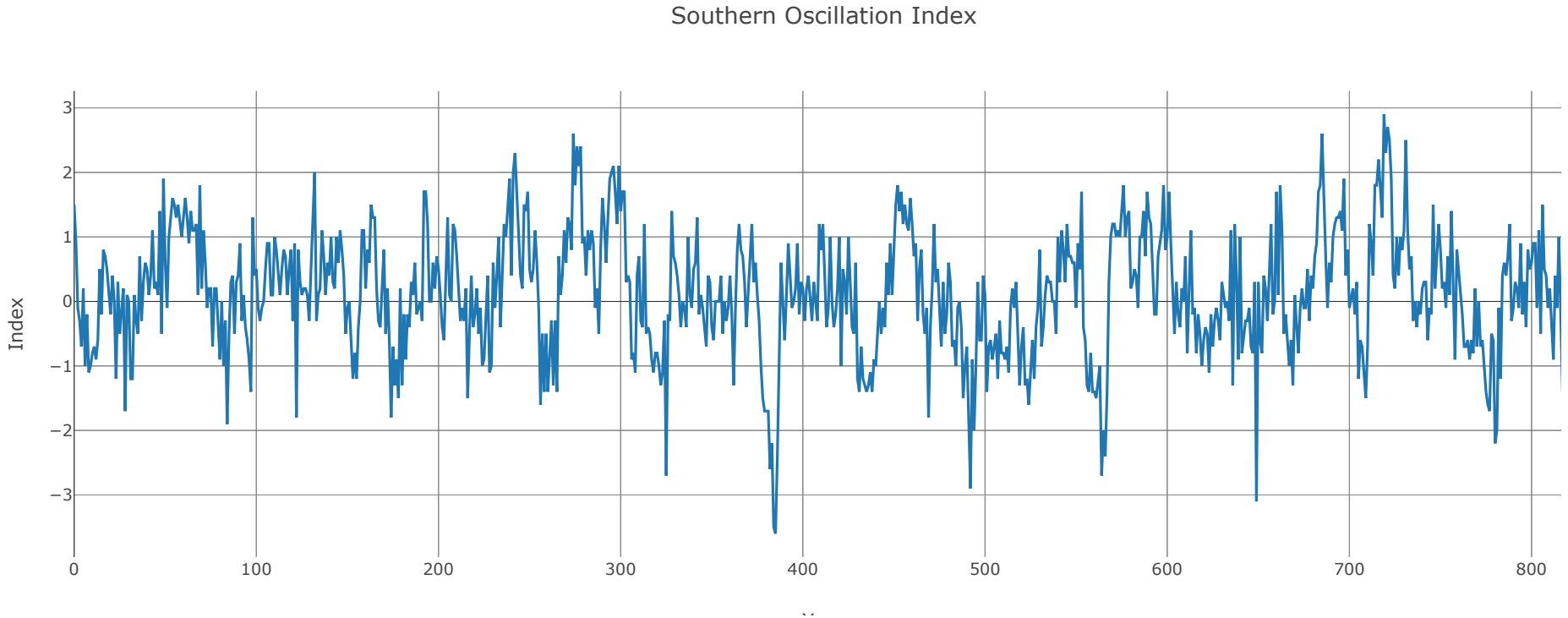
In [1]:

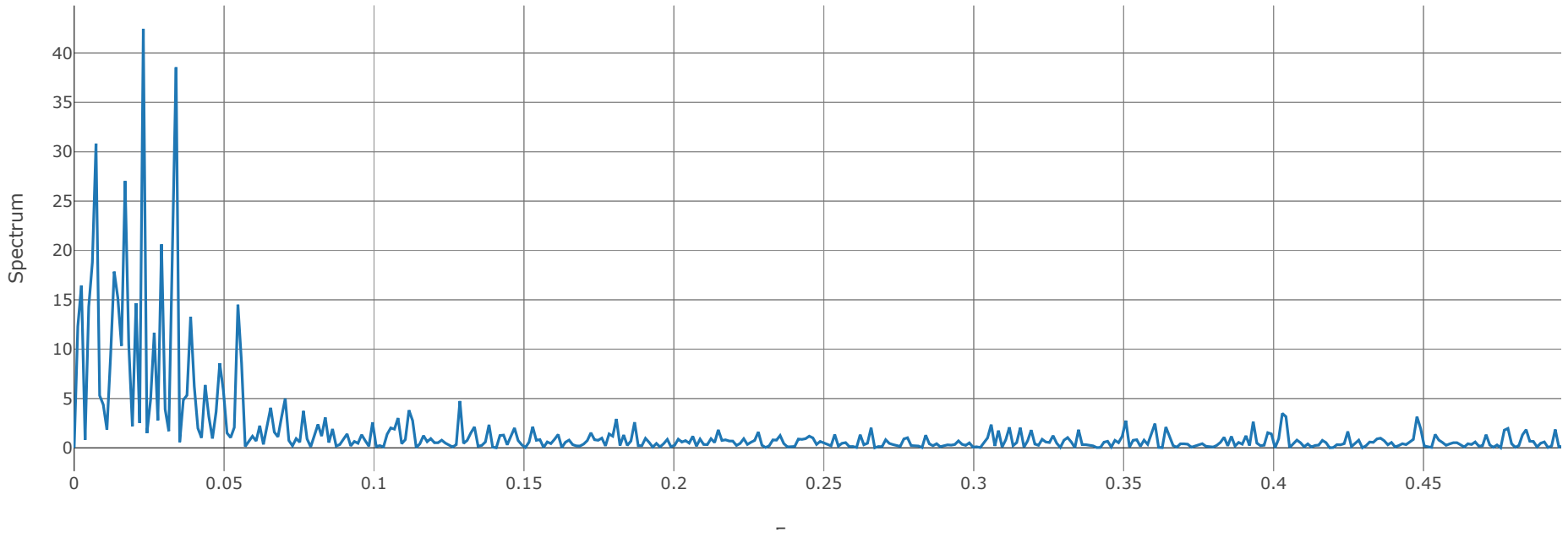
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
from pandas import DataFrame, read_csv
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
import datetime
import matplotlib.pyplot as plt
# import plotly.graph_objects as go
import plotly.offline as py
import plotly
from plotly.offline import init_notebook_mode
from plotly import graph_objs as go
plotly.offline.init_notebook_mode(connected=True)

#serie de mares de ubatuba em 1981
link = "https://www.ime.usp.br/~pam/soi.txt"
link = "https://www.ncdc.noaa.gov/teleconnections/enso/indicators/soi/data.csv"
f = requests.get(link)

Fs = 150.0;  # sampling rate
Ts = 1.0/Fs; # sampling interval

a = f.text.split("\n")
a = a[2:-1]

x = [i.strip().split(",")[0] for i in a]
y = [i.strip().split(",")[1] for i in a]
x = list(map(str, x))
y = list(map(float, y))

n = len(y) # length of the signal

# df.set_index('year',inplace=True)
soi = np.array(y)
# print(soi)
t = np.array(x)
t = np.arange(len(y))
# print(t)
```

In [2]:

```
from scipy import signal

fs = 1 # sampling rate
x = soi
freq, espectro = signal.periodogram(x, fs)

data=[go.Scatter(x=t, y=soi, mode='lines')]
layout = go.Layout(
    title='Southern Oscillation Index',
    xaxis={'title':'Year'},
    yaxis={'title':'Index'}
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)

data=[go.Scatter(x=freq, y=espectro, mode='lines')]
layout = go.Layout(
    title='Fourier Periodogram of Southern Oscillation Index',
    xaxis={'title':'Frequency'},
    yaxis={'title':'Spectrum'}
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)

index_greatest = np.where(espectro == max(espectro))[0]
print(f"PICOS DO PERIODOGRAMA")
print(f"Maior Pico: {max(espectro)} em frequência: {freq[index_greatest]}")
tmp_pico = max(espectro)
tmp_index_greatest = index_greatest
espectro[index_greatest] = 0
index_greatest = np.where(espectro == max(espectro))[0]
print(f"Segundo Maior Pico: {max(espectro)} em frequência: {freq[index_greatest]}\n\n")
espectro[tmp_index_greatest] = tmp_pico

import matplotlib.pyplot as plt
import numpy as np
from scipy import interpolate
from scipy import ndimage

print("**************************************************************************")
print(f"Estimador Suavizado de periodograma usando filtro Gaussiano")
print("**************************************************************************")

print(f"\nEstimação não paramétrica\n")

print(f"\nPara construir um estimador suavizado de periodograma,\n"+
      f"devemos aplicar o método de suavização no domínio do frequência\n"+
      f"depois de calcular o periodograma cru")

y_sm = espectro
x_sm = freq

y = y_sm.tolist()
x = x_sm.tolist()

y_sm = np.array(y)
```

```
x_sm = np.array(x)

# resample to lots more points - needed for the smoothed curves
x_smooth = np.linspace(x_sm.min(), x_sm.max(), 200)

sigma = 5 #tamanho da janela do filtro gaussiano
x_g1d2 = ndimage.gaussian_filter1d(x_sm, sigma)
y_g1d2 = ndimage.gaussian_filter1d(y_sm, sigma)

sigma = 2 #tamanho da janela do filtro gaussiano
x_g1d3 = ndimage.gaussian_filter1d(x_sm, sigma)
y_g1d3 = ndimage.gaussian_filter1d(y_sm, sigma)

sigma = 1 #tamanho da janela do filtro gaussiano
x_g1d4 = ndimage.gaussian_filter1d(x_sm, sigma)
y_g1d4 = ndimage.gaussian_filter1d(y_sm, sigma)

data = [go.Scatter(x=x_sm, y=y_sm, mode='lines', name="Original Periodogram")]
data += [go.Scatter(x=x_g1d2, y=y_g1d2, mode='lines', name="Gaussian Kernel width 5")]
data += [go.Scatter(x=x_g1d3, y=y_g1d3, mode='lines', name="Gaussian Kernel width 2")]
data += [go.Scatter(x=x_g1d4, y=y_g1d4, mode='lines', name="Gaussian Kernel width 1")]

layout = go.Layout(
    title='Smoothed Periodogram of Southern Oscillation Index',
    xaxis={'title':'Smoothed Spectrum'},
    yaxis={'title':'Frequency'}
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)

print(f"\nObs2.: O tamanho de janela 2 parece mais adequado\n\n")
```


Southern Oscillation Index
Index
3
2
1
0
−1
−2
−3
0
100
200
300
400
500
600
700
800

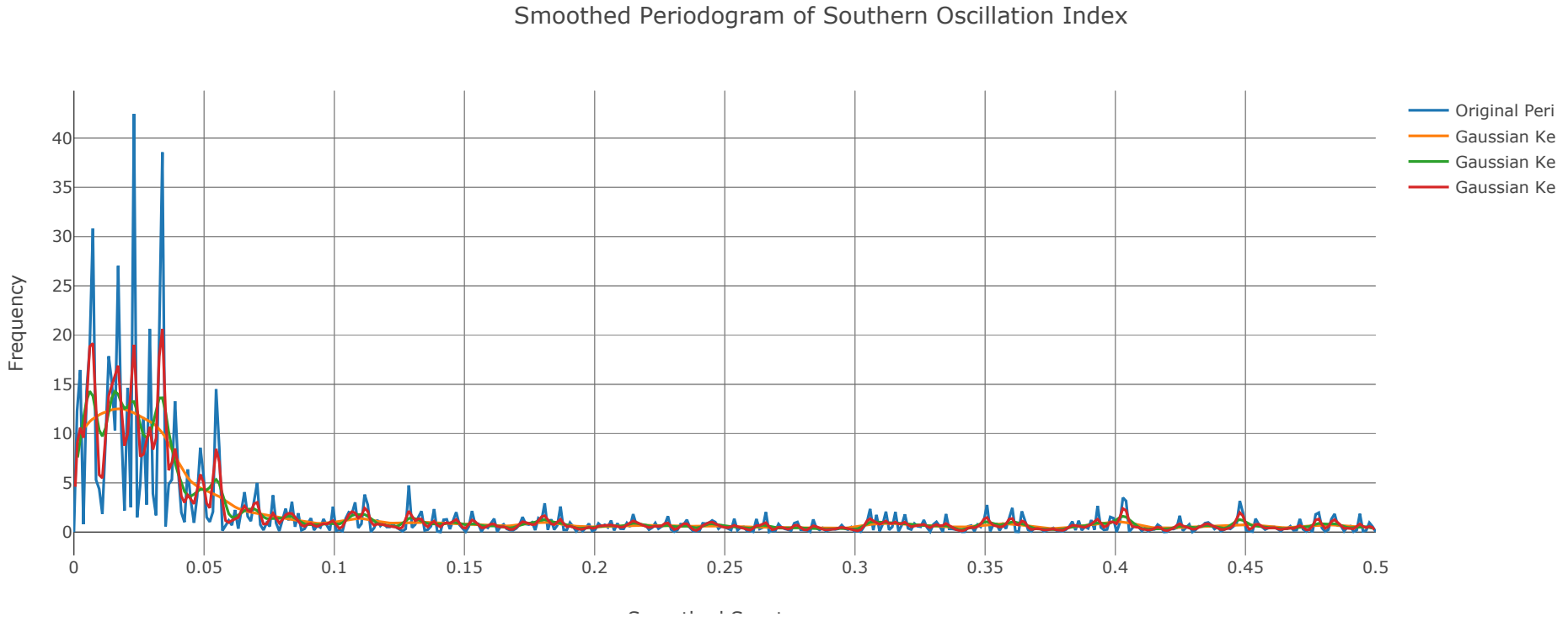


PICOS DO PERIODOGRAMA
Maior Pico: 42.44722345502437 em frequência: [0.02305825]
Segundo Maior Pico: 38.560556822731606 em frequência: [0.03398058]

****
Estimador Suavizado de periodograma usando filtro Gaussiano
****

Estimação não paramétrica

Para construir um estimador suavizado de periodograma,
devemos aplicar o método de suavização no domínio do frequência
depois de calcular o periodograma cru

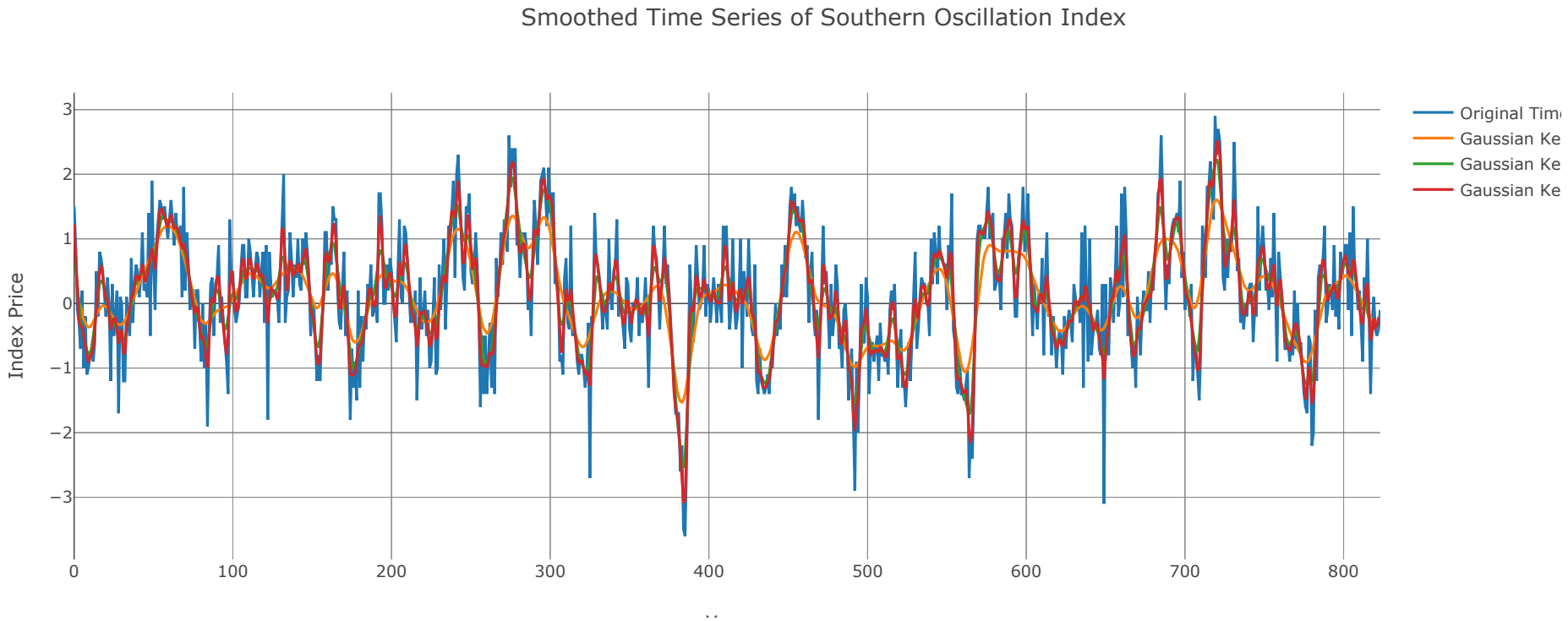


Obs2.: O tamanho de janela 2 parece mais adequado

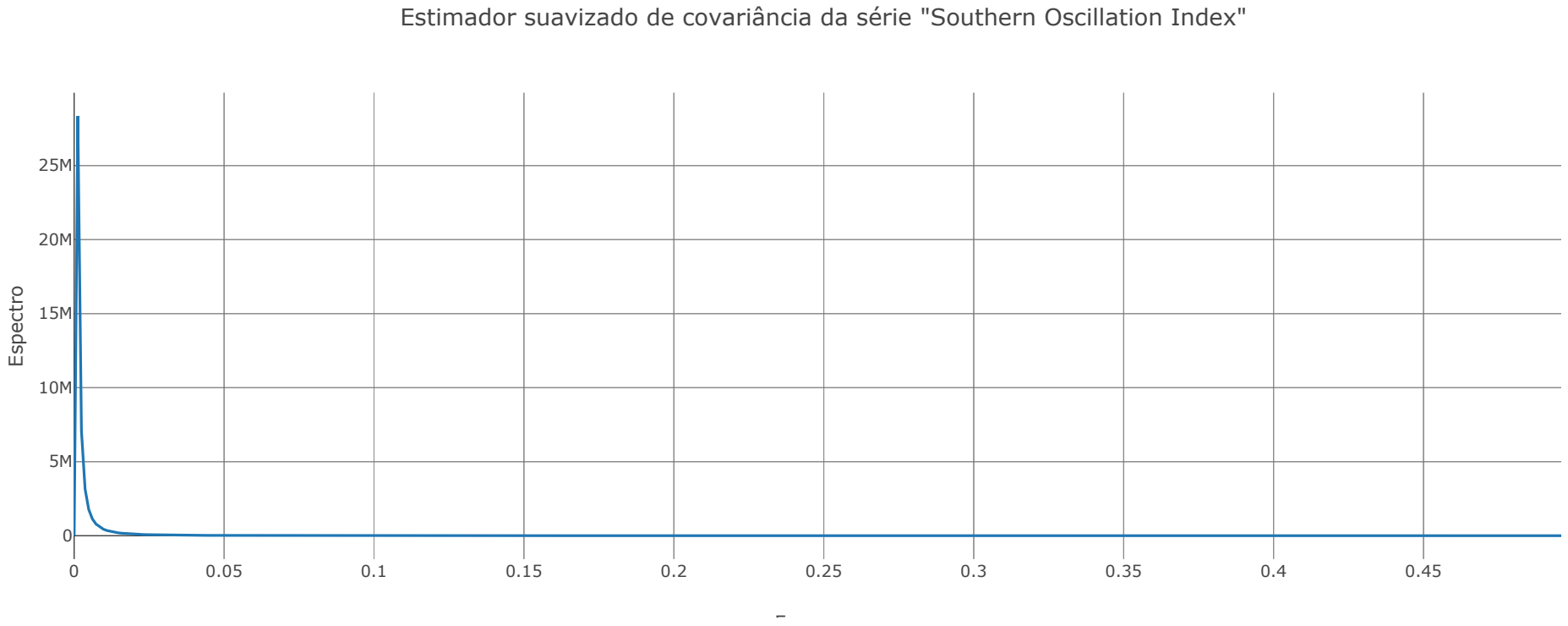
In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy import interpolate
from scipy import ndimage

print("\n\n*************************************************************************")
print(f"Estimador Suavizado de covariância usando filtro Gaussiano")
print("*************************************************************************")
print(f"Estimação não paramétrica")

y_sm = soi
x_sm = t

y = y_sm.tolist()
x = x_sm.tolist()

y_sm = np.array(y)
x_sm = np.array(x)

# resample to lots more points - needed for the smoothed curves
x_smooth = np.linspace(x_sm.min(), x_sm.max(), 200)

sigma = 5 #tamanho da janela do filtro gaussiano
x_g1d2 = ndimage.gaussian_filter1d(x_sm, sigma)
y_g1d2 = ndimage.gaussian_filter1d(y_sm, sigma)

sigma = 2 #tamanho da janela do filtro gaussiano
x_g1d3 = ndimage.gaussian_filter1d(x_sm, sigma)
y_g1d3 = ndimage.gaussian_filter1d(y_sm, sigma)

sigma = 1 #tamanho da janela do filtro gaussiano
x_g1d4 = ndimage.gaussian_filter1d(x_sm, sigma)
y_g1d4 = ndimage.gaussian_filter1d(y_sm, sigma)

data = [go.Scatter(x=x_sm, y=y_sm, mode='lines', name="Original Time series")]
# data += [go.Scatter(x=x_g1d, y=y_g1d, mode='lines', name="Gaussian Kernel width 10")]
data += [go.Scatter(x=x_g1d2, y=y_g1d2, mode='lines', name="Gaussian Kernel width 5")]
data += [go.Scatter(x=x_g1d3, y=y_g1d3, mode='lines', name="Gaussian Kernel width 2")]
data += [go.Scatter(x=x_g1d4, y=y_g1d4, mode='lines', name="Gaussian Kernel width 1")]
# data += [go.Scatter(x=x_g1d5, y=y_g1d5, mode='lines', name="Gaussian Kernel width 0.5")]

layout = go.Layout(
    title='Smoothed Time Series of Southern Oscillation Index',
    xaxis={'title':'Year'},
    yaxis={'title':'Index Price'}
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)

print(f"Para construir um estimador suavizado de covariância,\n"+
      f"devemos aplicar o método de suavização no domínio do tempo\n"+
      f"para então calcular o periodograma")
print(f"Obs.: Para comparação, mostra-se o spline da série")

print(f"\nObs2.: O tamanho de janela 2 parece mais adequado\n\n")
```

```
print("************************************************************************")
print(f"Calculando periodograma a partir da série suavizada no tempo")
print("************************************************************************")

fs = 1
freq, espectro = signal.periodogram(x_g1d3, fs)
data = [go.Scatter(x=freq, y=espectro, mode='lines', name="Smoothed Resulting Periodogram")]
# data += [go.Scatter(x=x_g1d5, y=y_g1d5, mode='lines', name="Gaussian Kernel width 0.5")]

layout = go.Layout(
    title='Estimador suavizado de covariância da série "Southern Oscillation Index"',
    xaxis={'title':'Frequency'},
    yaxis={'title':'Espectro'}
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)

index_greatest_2 = np.where(espectro == max(espectro))[0]
print(f"PICOS DO PERIODOGRAMA")
print(f"Maior Pico: {max(espectro)} em frequência: {freq[index_greatest_2]}")
```

```
**********************************************************************
Estimador Suavizado de covariância usando filtro Gaussiano
**********************************************************************
Estimação não paramétrica
```




```
Para construir um estimador suavizado de covariância,
devemos aplicar o método de suavização no domínio do tempo
para então calcular o periodograma
Obs.: Para comparação, mostra-se o spline da série

Obs2.: O tamanho de janela 2 parece mais adequado


**********************************************************************
Calculando periodograma a partir da série suavizada no tempo
**********************************************************************
```

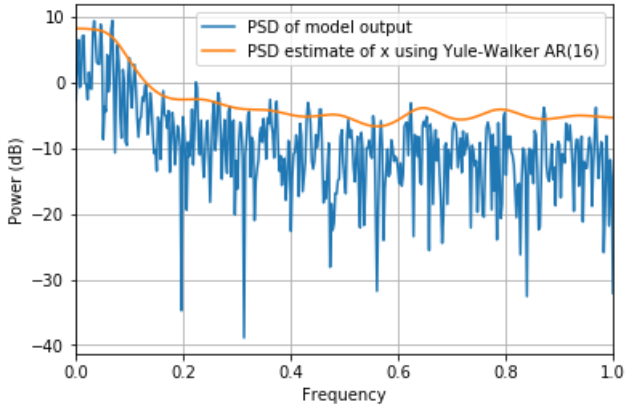


PICOS DO PERIODOGRAMA
Maior Pico: 28343402.14581507 em frequência: [0.00121359]

In [4]:
```
from pylab import legend
import scipy.signal
from spectrum import Periodogram, pyule
print("************************************************************************")
print(f"Estimador espectral autorregressivo baseado em Yule-Walker usando AR(16)")
print("************************************************************************")
print(f"Estimação paramétrica - Hipótese Autorregressiva")
y = soi
p = Periodogram(y, sampling=2)
p.plot()
p = pyule(y, 16, sampling=2, scale_by_freq=False)
p.plot()
legend(['PSD of model output','PSD estimate of x using Yule-Walker AR(16)'])
```

```
************************************************************************
Estimador espectral autorregressivo baseado em Yule-Walker usando AR(16)
************************************************************************
Estimação paramétrica - Hipótese Autorregressiva
```

Out[4]:
```
<matplotlib.legend.Legend at 0x1c1f1214e0>
```



Comentários sobre os estimadores

- Periodograma de Fourier:

Apesar de indicar que em baixas frequências existe uma periodicidade maior, possui vários picos de alturas parecidas. Isso significa que o periodograma não é capaz de segregar bem uma faixa de frequências definidas para indicação de periodicidade.

- Periodograma Suavizado:

Tanto o estimador suavizado de periodograma quanto o estimador suavizado de covariância são mais fáceis de interpretar, pois concentram a significância de determinadas frequências. No entanto, é preferível usar o estimador suavizado de periodograma tendo em mente que ele preserva relevâncias diferentes para frequências baixas e próximas (isso pode ser um diferencial na flexibilidade da análise do fenômeno).

- Estimador espectral autorregressivo

O estimador espectral autorregressivo gerou um resultado similar ao estimador suavizado de periodograma com janela de tamanho 5 para frequências superiores a 0.2. Porém perdeu relevância de algumas frequências abaixo de 0.2, o que dificultaria uma análise mais precisa sobre periodicidades em baixas frequências.

De maneira geral, escolheria o estimador suavizado de periodograma com janela Gaussiana de ordem 2. Acredito ser a melhor opção para extrair uma análise.