In [1]:

```
"""
Author Nicola Lombardi, SW HW Engineer Computer and Telecommunications Engineer
& Senior Python Dev.


=========================================================
Telecomunicazioni / Elettronica / Informatica
Responsabile Cost&Qualità
Ex Ingegnere Hardware Baseband
Web: Curriculum pubblico
contributi scientifici: Ricerca
=========================================================

---------------------------------------
-> Collab: Andrea Casula, Paolo Pintus
---------------------------------------


Professor Paolo Pintus (Assistant Professor dipartimento di Fisica)
Web - https://web.unica.it/unica/page/it/paolo_pintus
Relazione con il professore: Relatore di tesi del 2022
Ruolo   -  Ricercatore a tempo determinato
Area scientifico disciplinare  - Scienze fisiche
Settore scientifico disciplinare   FIS/03 FISICA DELLA MATERIA
Email   paolo.pintus@unica.it


Professor Andrea Casula (Associate Professor in Sistemi Wireless, Laurea Magistrale
“Internet Engineering”)
Web  -   https://web.unica.it/unica/it/ateneo_s07_ss01.page?contentId=SHD30253
Relazione con il professore: Relatore di tesi del 2022
Ruolo -  Professore associato
Area scientifico disciplinare  -  Ingegneria industriale e dell'informazione
Settore scientifico disciplinare   -    ING-INF/02 CAMPI ELETTROMAGNETICI


DATE: February 2025 NAME OF THE PROJECT: REAL TIME BLE TRACKER SYSTEM

"""
```

Out[1]:

```
"\nAuthor Nicola Lombardi, SW HW Engineer Computer and Telecommunications Engineer \n& Senior Python Dev. \n\n\n=========================================================\nTelecomunicazioni / Elettronica / Informatica\nResponsabile Cost&Qualità  \nEx Ingegnere Hardware Baseband \nWeb: Curriculum pubblico\ncontributi scientifici: Ricerca\n=========================================================\n\n---------------------------------------\n-> Collab: Andrea Casula, Paolo Pintus\n---------------------------------------\n\n\nProfessor Paolo Pintus (Assistant Professor dipartimento di Fisica) \nWeb - https://web.unica.it/unica/page/it/paolo_pintus \nRelazione con il professore: Relatore di tesi del 2022 \nRuolo   -  Ricercatore a tempo determinato \nArea scientifico disciplinare  - Scienze fisiche \nSettore scientifico disciplinare   FIS/03 FISICA DELLA MATERIA \nEmail   paolo.pintus@unica.it\n\n\nProfessor Andrea Casula (Associate Professor in Sistemi Wireless, Laurea Magistrale \n“Internet Engineering”)  \nWeb  -   https://web.unica.it/unica/it/ateneo_s07_ss01.page?contentId=SHD30253 \nRelazione con il professore: Relatore di tesi del 2022 \nRuolo -  Professore associato \nArea scientifico disciplinare  -  Ingegneria industriale e dell'informazione \nSettore scientifico disciplinare   -    ING-INF/02 CAMPI ELETTROMAGNETICI \n\n\nDATE: February 2025 NAME OF THE PROJECT: REAL TIME BLE TRACKER SYSTEM\n\n"
```

In [2]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from SALib.analyze import rbd_fast
```

```python
# this is where you define your own model, procedure, experiment...
from SALib.test_functions import Ishigami
import numpy as np
```

In [3]:

```python
# this is where you define your own model, procedure, experiment...
from SALib.test_functions import Ishigami
import numpy as np

def run_model(x1, x2):
    """
    COPY HERE YOUR OWN CODE

    the function takes as input 1 sample
    returns 1 or more output
    """
    lmbd_0 = 1.55 * (np.math.pow(10,-6)) # central wavelength is 1550 nanometers

    n1 = 2.848 # effective index of the grating teeth
    n2 = 2.534 # effective index of the grating slots
    # Delete from HERE =======================================

    # As an example, we'll look at the famous Ishigami function
    # (A and B are specific parameters for the Ishigami function)
    A = 7
    B = 0.1
    y = np.arcsin(((n1*x1+n2-x1*n2)*x2- lmbd_0)/x2  )
    y_deg = 180/np.pi * y
    # ========= TO HERE and replace with your own piece of code

    return y_deg





def conv_study(n, Y, X):
    # take n samples among the num_samples, without replacement
    subset = np.random.choice(num_samples, size=n, replace=False)
    return rbd_fast.analyze(problem=problem,
                            Y=Y[subset],
                            X=X[subset])['S1']






def bootstrap(problem, Y, X):
    """
    Calculate confidence intervals of rbd-fast indices
    1000 draws
    returns 95% confidence intervals of the 1000 indices

    problem : dictionnary as SALib uses it
    X : SA input(s)
    Y : SA output(s)
    """
    all_indices = []

    for i in range(1000):
        X_new = np.zeros(X.shape)
        Y_new = np.zeros(Y.shape).flatten()
```

```
        # draw with replacement
        tirage_indices = np.random.randint(0, high=Y.shape[0], size = Y.shape[0])

        for j, index in enumerate(tirage_indices):
            X_new[j,:] = X[index, :]
            Y_new[j] = Y[index]

        all_indices.append(rbd_fast.analyze(problem=problem, Y=Y_new, X=X_new)['S1']

    means = np.array([i.mean() for i in np.array(all_indices).T])
    stds = np.array([i.std() for i in np.array(all_indices).T])
    return np.array([means - 2 * stds, means + 2 * stds])
```

In [4]:

```
# STATE THE PROBLEM DICTIONNARY
# what will be varying (=inputs) ? in what bounds ?
FF_min = 0.2
FF_max = 0.7
g_period_power = (np.math.pow(10,-9)) # structure grating period is 660 nanometers
g_period_min = 620 * g_period_power # 620 nm
g_period_max = 690 * g_period_power # 690 nm
problem = {
    'num_vars': 2,
    'names': ['FillFactor', 'GratingPeriod'],
    'bounds': [[FF_min,FF_max ],

                [g_period_min, g_period_max]]
}
# say we want to draw 150 samples

num_samples = 150
# we draw a Latin Hypercube Sampling (LHS) that is fitted for an RBD FAST analysis
# (other sampling metods available in the library though)

from SALib.sample.latin import sample

all_samples = sample(problem, num_samples)

# run your model, procedure, experiment for each sample of your sampling

# unpack all_samples into 3 vectors x1, x2, x3
x1, x2 = all_samples.T
```

In [5]:

```
# run the model, all samples at the same time
ishigami_results = run_model(x1, x2)

# Let us look at the analyze method

rbd_fast.analyze(problem=problem, Y=ishigami_results, X=all_samples)
# storing the first order indices of the analyze method
si1 = rbd_fast.analyze(problem=problem, Y=ishigami_results, X=all_samples)['S1']


# make nice plots with the indices (looks good on your presentations)
# do not use the plotting tools of SALib, they are made for the method of Morris ...
```

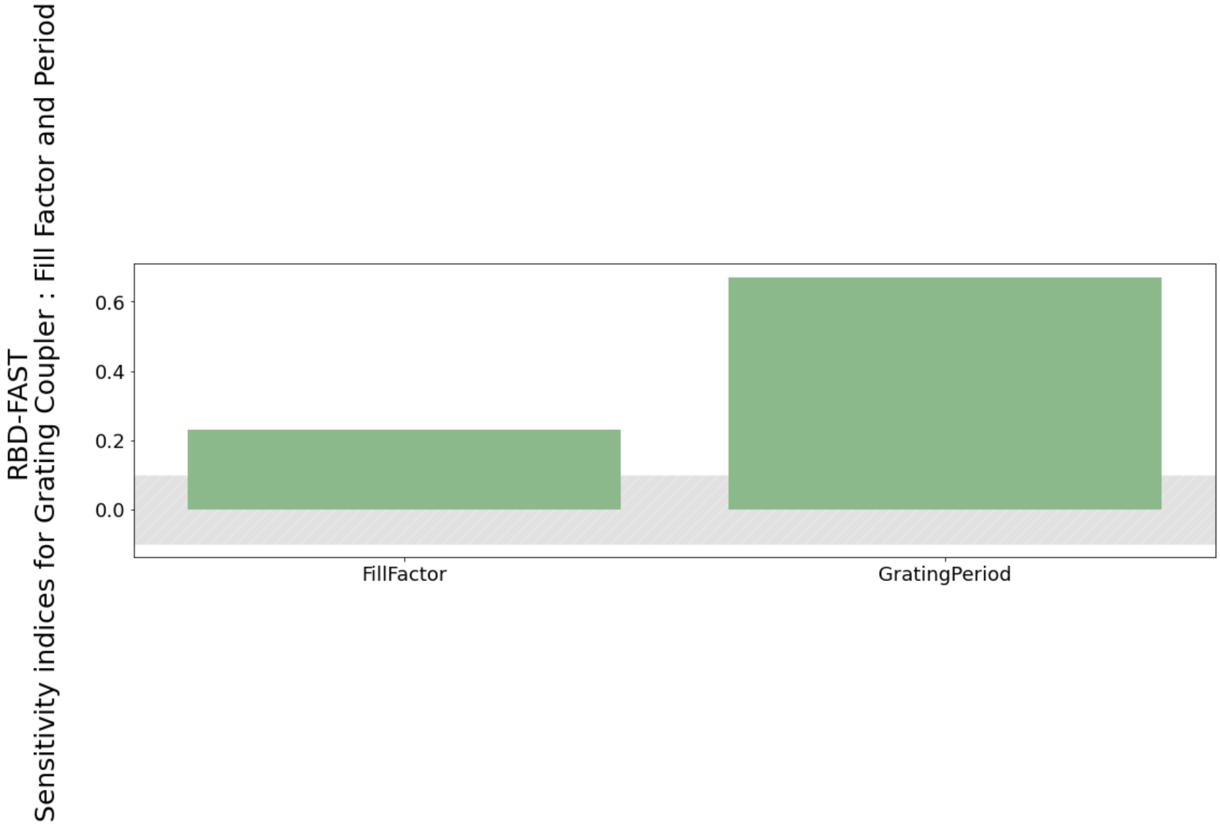
In [6]:

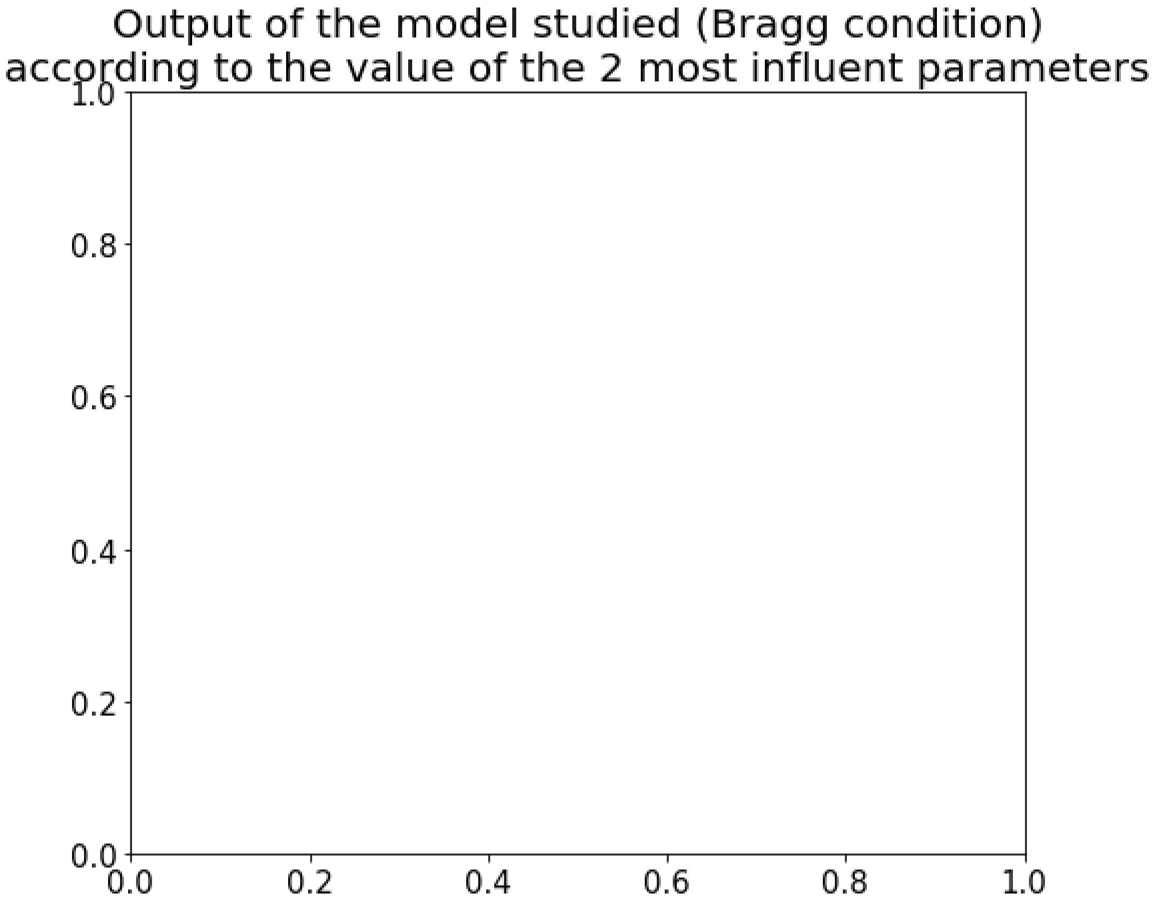
```
fig, ax = plt.subplots()
fig.set_size_inches(18, 5)

ax.tick_params(labelsize=18)
```

```python
# ===== X-AXIS =====
ax.set_xticks(np.arange(problem['num_vars']))
ax.set_xticklabels(problem['names'])
ax.set_xlim(xmin=-0.5, xmax=problem['num_vars'] - 0.5)

# ===== Y-AXIS =====
ax.set_ylabel('RBD-FAST\nSensitivity indices for Grating Coupler : Fill Factor and P

# ===== BARS REPRESENTING THE SENSITIVITY INDICES =====
ax.bar(np.arange(problem['num_vars']), si1,
       color='DarkSeaGreen');

#in striped grey : not significant indices
ax.fill_between(x=[-0.5, 5.5], y1=-0.1, y2=0.1, color='grey', alpha=0.2, hatch='//',



# take a closer look to undestand interactions (looks even better on your presentati
# this part can be done without analyzing with SALib, just with the ouput from the s

fig, ax = plt.subplots()
fig.set_size_inches(8, 7)

ax.tick_params(labelsize=15)
ax.set_title('Output of the model studied (Bragg condition)\n'
             'according to the value of the 2 most influent parameters',
             fontsize=20)
```

Out[6]:

```
Text(0.5, 1.0, 'Output of the model studied (Bragg condition)\naccording to the value
of the 2 most influent parameters')
```





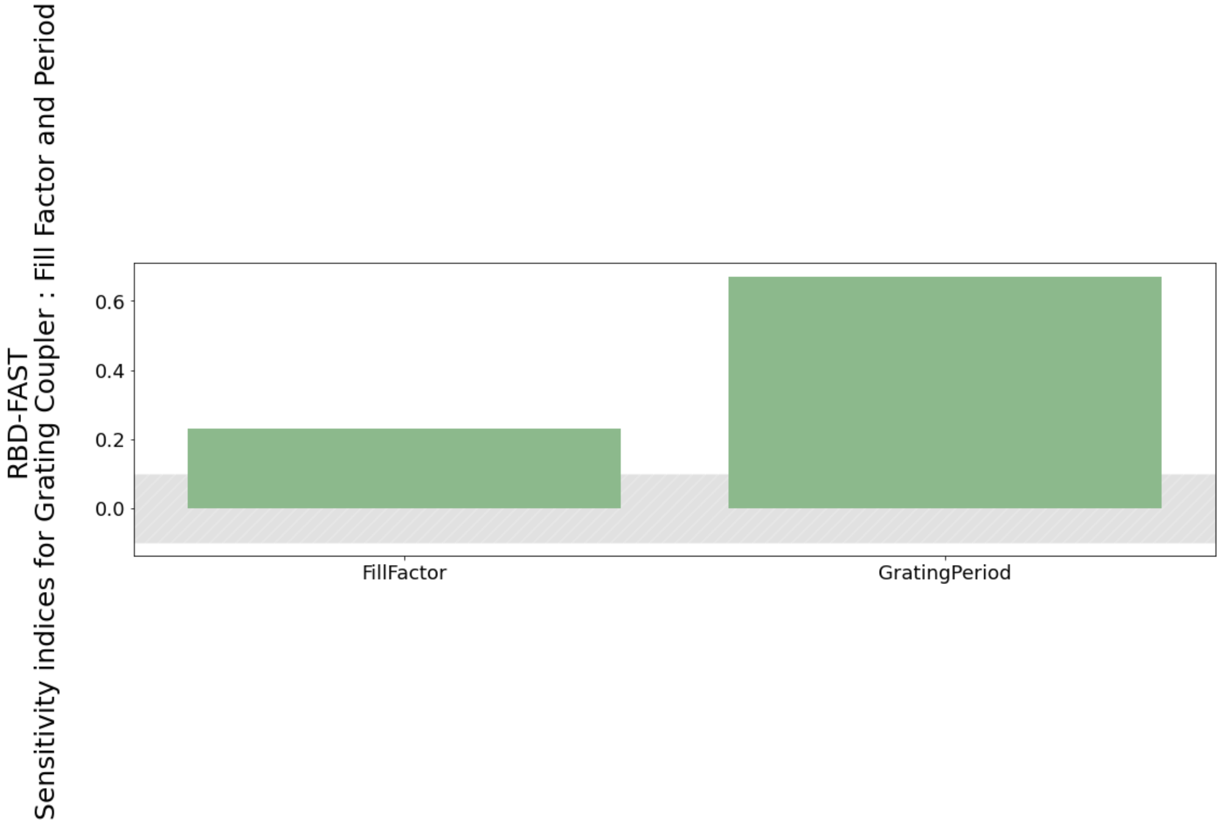
In [7]:

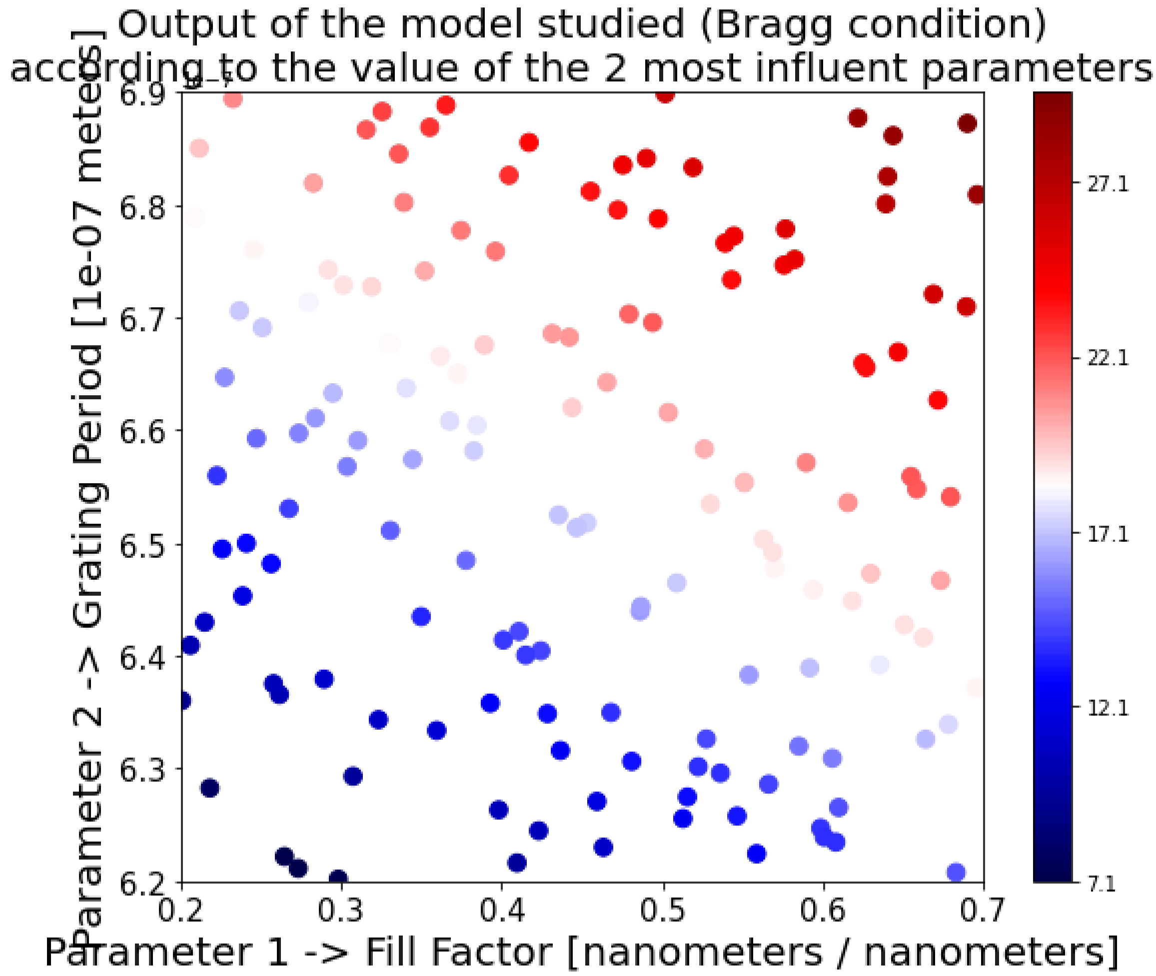
```
fig, ax = plt.subplots()
fig.set_size_inches(18, 5)

ax.tick_params(labelsize=18)

# ===== X-AXIS =====
ax.set_xticks(np.arange(problem['num_vars']))
ax.set_xticklabels(problem['names'])
ax.set_xlim(xmin=-0.5, xmax=problem['num_vars'] - 0.5)

# ===== Y-AXIS =====
ax.set_ylabel('RBD-FAST\nSensitivity indices for Grating Coupler : Fill Factor and P

# ===== BARS REPRESENTING THE SENSITIVITY INDICES =====
ax.bar(np.arange(problem['num_vars']), si1,
       color='DarkSeaGreen');

#in striped grey : not significant indices
ax.fill_between(x=[-0.5, 5.5], y1=-0.1, y2=0.1, color='grey', alpha=0.2, hatch='//',



# take a closer look to undestand interactions (looks even better on your presentati
# this part can be done without analyzing with SALib, just with the ouput from the s

fig, ax = plt.subplots()
fig.set_size_inches(8, 7)

ax.tick_params(labelsize=15)
ax.set_title('Output of the model studied (Bragg condition)\n'
             'according to the value of the 2 most influent parameters',
             fontsize=20)
```

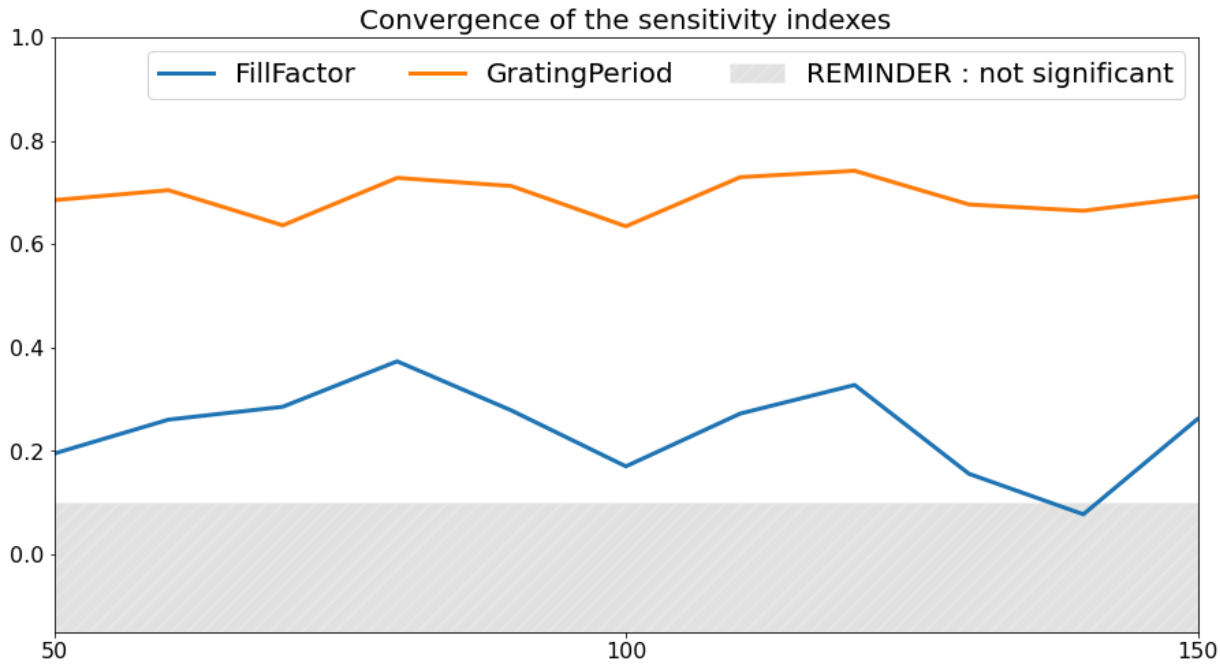
```
# ===== SCATTER =====
size = np.ones(num_samples) * 75

sc = ax.scatter(x1, x2,
                c=ishigami_results,
                s=size,
                vmin=ishigami_results.min(),
                vmax=ishigami_results.max(),
                cmap='seismic',
                edgecolor=None)


# ===== X-AXIS =====
ax.set_xlim(xmin=problem['bounds'][0][0], xmax=problem['bounds'][0][1])
ax.set_xlabel('Parameter 1 -> Fill Factor [nanometers / nanometers]', fontsize=20)


# ===== Y-AXIS =====
ax.set_ylim(ymin=problem['bounds'][1][0], ymax=problem['bounds'][1][1])
ax.set_ylabel('Parameter 2 -> Grating Period [1e-07 meters]', fontsize=20)

# ===== COLORBAR =====
ticks = np.arange(ishigami_results.min(), ishigami_results.max(), 5)
cb = plt.colorbar(sc, ticks=ticks);
cb.ax.set_yticklabels([str(round(i,1)) for i in ticks])

fig.tight_layout()
```

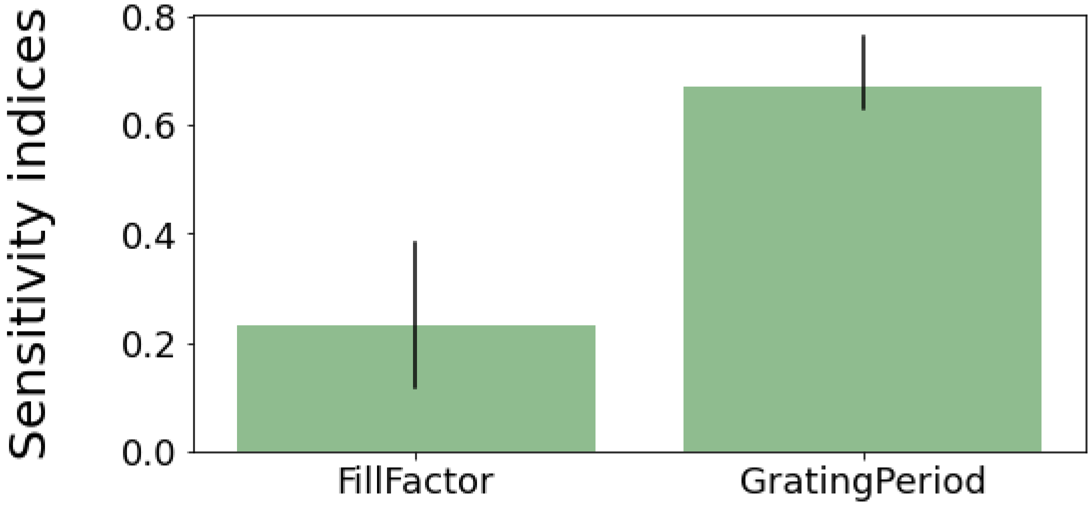
```
all_indices = np.array([conv_study(n=n, Y=ishigami_results, X=all_samples)
                        for n in np.arange(50, num_samples + 1, 5)])


# convergence check
fig, ax = plt.subplots()
fig.set_size_inches(15,8)

ax.set_title('Convergence of the sensitivity indexes', fontsize=20)

ax.tick_params(labelsize=16)

ax.set_xlim(xmin=0, xmax=(num_samples - 50)//10)
ax.set_xticks(np.arange(0,(num_samples - 50)//10 +1,5))
ax.set_xticklabels([str(i) for i in range(50,num_samples+1,50)])

ax.set_ylim(ymin=-0.15, ymax=1)


for p,param in enumerate(problem['names']):
    ax.plot(all_indices[:,p], linewidth=3, label=param)

ax.fill_between(x=[0,(num_samples - 50)//10], y1=-0.15, y2=0.1, color='grey', alpha=
              label='REMINDER : not significant')

ax.legend(fontsize=20, ncol=4)

# Get bootstrap confidence intervals for each index

bootstrap_conf = bootstrap(problem=problem, Y=ishigami_results, X=all_samples)
```

```python
# make nice plots with the indices (looks good on your presentations)
# do not use the plotting tools of SALib, they are made for the method of Morris ...

fig, ax = plt.subplots()
fig.set_size_inches(8,4)

ax.tick_params(labelsize=18)

# ===== X-AXIS =====
ax.set_xticks(np.arange(problem['num_vars']))
ax.set_xticklabels(problem['names'])
ax.set_xlim(xmin=-0.5, xmax=problem['num_vars'] - 0.5)

# ===== Y-AXIS =====
ax.set_ylabel('Sensitivity indices\n', fontsize=25)

# ===== BARS REPRESENTING THE SENSITIVITY INDICES =====
ax.bar(np.arange(problem['num_vars']), si1,
         color='DarkSeaGreen');

# ===== LINES REPRESENTING THE BOOTSTRAP "CONFIDENCE INTERVALS" =====
for j in range(problem['num_vars']):
    ax.plot([j, j], [bootstrap_conf[0, j], bootstrap_conf[1, j]],
            'k')

#ax.fill_between(x=[-0.5, 5.5], y1=-0.1, y2=0.1, color='grey', alpha=0.2, hatch='//'
```




Output of the model studied (Bragg condition)
according to the value of the 2 most influent parameters
1e-7
Parameter 2 -> Grating Period [1e-07 meters]
Parameter 1 -> Fill Factor [nanometers / nanometers]
6.9
6.8
6.7
6.6
6.5
6.4
6.3
6.2
0.2
0.3
0.4
0.5
0.6
0.7
27.1
22.1
17.1
12.1
7.1


Convergence of the sensitivity indexes
FillFactor
GratingPeriod
REMINDER : not significant
1.0
0.8
0.6
0.4
0.2
0.0
50
100
150



In [8]:

```
#################################################################
```

In [9]:

```
# this is where you define your own model, procedure, experiment...
from SALib.test_functions import Ishigami
import numpy as np

def run_model(x1, x2):
    """
    COPY HERE YOUR OWN CODE

    the function takes as input 1 sample
    returns 1 or more output
    """
    #lmbd_0 = 1.55 * (np.math.pow(10,-6)) # central wavelength is 1550 nanometers
    # lmbd_0 is the new x1
    FF=0.6
    n1 = 2.848 # effective index of the grating teeth
    n2 = 2.534 # effective index of the grating slots
    # Delete from HERE =======================================

    # As an example, we'll look at the famous Ishigami function
    # (A and B are specific parameters for the Ishigami function)
    A = 7
    B = 0.1
    y = np.arcsin(((n1*FF+n2-FF*n2)*x2- x1)/x2  )
    y_deg = 180/np.pi * y
    # ========= TO HERE and replace with your own piece of code

    return y_deg




def conv_study(n, Y, X):
    # take n samples among the num_samples, without replacement
    subset = np.random.choice(num_samples, size=n, replace=False)
    return rbd_fast.analyze(problem=problem,
                            Y=Y[subset],
                            X=X[subset])['S1']
```

```python
def bootstrap(problem, Y, X):
    """
    Calculate confidence intervals of rbd-fast indices
    1000 draws
    returns 95% confidence intervals of the 1000 indices

    problem : dictionnary as SALib uses it
    X : SA input(s)
    Y : SA output(s)
    """
    all_indices = []

    for i in range(1000):
        X_new = np.zeros(X.shape)
        Y_new = np.zeros(Y.shape).flatten()
        # draw with replacement
        tirage_indices = np.random.randint(0, high=Y.shape[0], size = Y.shape[0])

        for j, index in enumerate(tirage_indices):
            X_new[j,:] = X[index, :]
            Y_new[j] = Y[index]

        all_indices.append(rbd_fast.analyze(problem=problem, Y=Y_new, X=X_new)['S1']

    means = np.array([i.mean() for i in np.array(all_indices).T])
    stds = np.array([i.std() for i in np.array(all_indices).T])
    return np.array([means - 2 * stds, means + 2 * stds])
```

In [10]:

```python
# STATE THE PROBLEM DICTIONNARY
# what will be varying (=inputs) ? in what bounds ?

g_period_power = (np.math.pow(10,-9)) # structure grating period is 660 nanometers
g_period_min = 620 * g_period_power # 620 nm
g_period_max = 690 * g_period_power # 690 nm
lmbd_BAND_C_min = 1530*g_period_power
lmbd_BAND_C_max = 1565*g_period_power

problem = {
    'num_vars': 2,
    'names': ['Wavelength', 'GratingPeriod'],
    'bounds': [[lmbd_BAND_C_min,lmbd_BAND_C_max ],

               [g_period_min, g_period_max]]
}
# say we want to draw 150 samples

num_samples = 150
# we draw a Latin Hypercube Sampling (LHS) that is fitted for an RBD FAST analysis
# (other sampling metods available in the library though)

from SALib.sample.latin import sample

all_samples = sample(problem, num_samples)

# run your model, procedure, experiment for each sample of your sampling

# unpack all_samples into 3 vectors x1, x2, x3
x1, x2 = all_samples.T
```

In [11]:

```python
# run the model, all samples at the same time
ishigami_results = run_model(x1, x2)
```

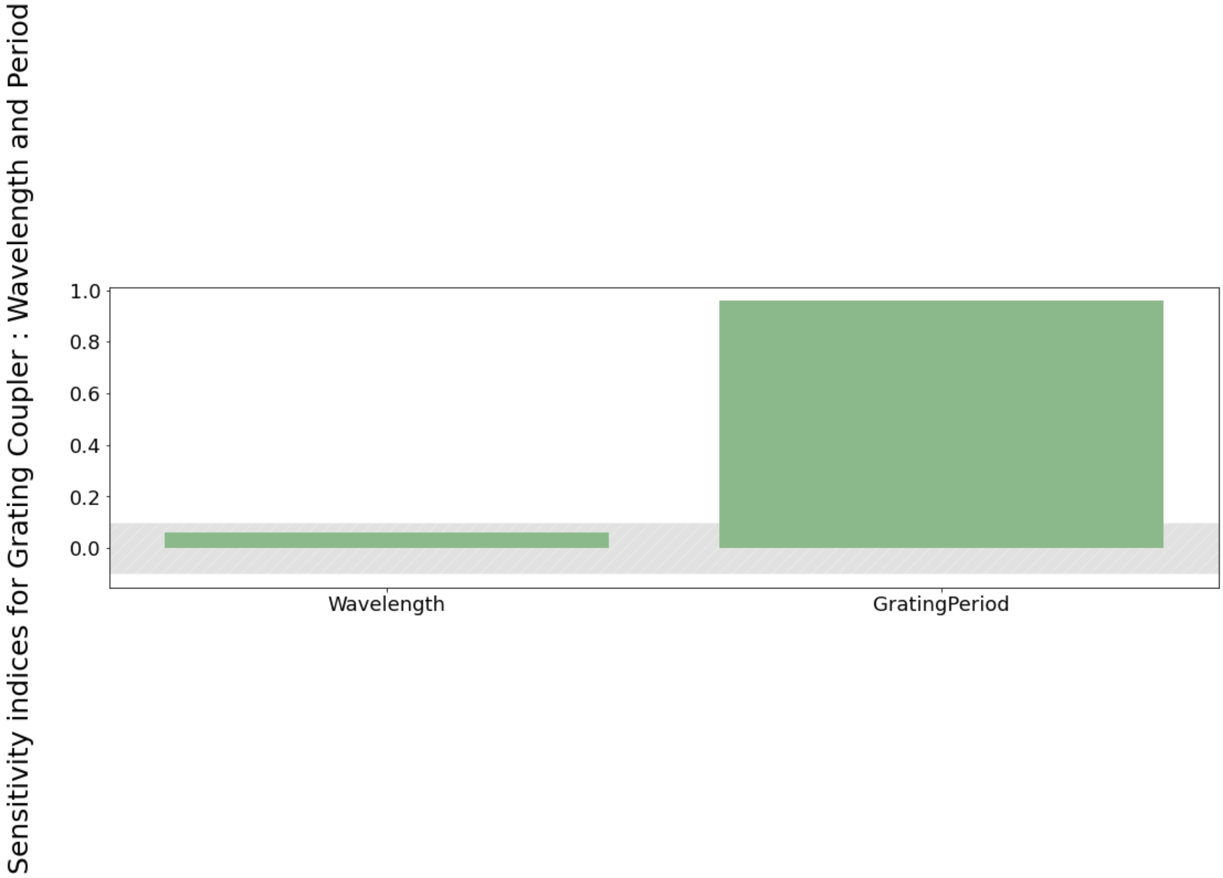
```
# Let us look at the analyze method

rbd_fast.analyze(problem=problem, Y=ishigami_results, X=all_samples)
# storing the first order indices of the analyze method
si1 = rbd_fast.analyze(problem=problem, Y=ishigami_results, X=all_samples)['S1']


# make nice plots with the indices (looks good on your presentations)
# do not use the plotting tools of SALib, they are made for the method of Morris ...
```

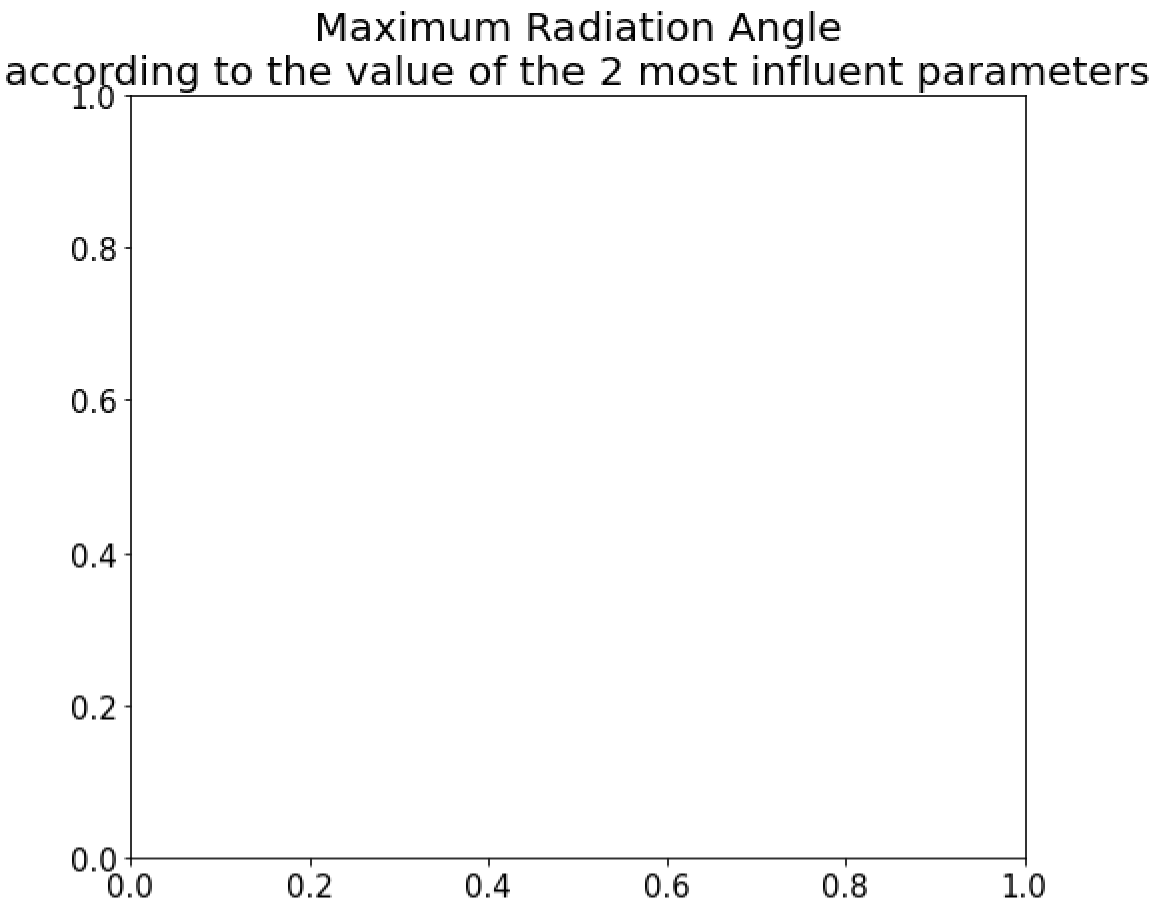
In [12]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from SALib.analyze import rbd_fast

# this is where you define your own model, procedure, experiment...
from SALib.test_functions import Ishigami
import numpy as np
fig, ax = plt.subplots()
fig.set_size_inches(18, 5)

ax.tick_params(labelsize=18)

# ===== X-AXIS =====
ax.set_xticks(np.arange(problem['num_vars']))
ax.set_xticklabels(problem['names'])
ax.set_xlim(xmin=-0.5, xmax=problem['num_vars'] - 0.5)

# ===== Y-AXIS =====
ax.set_ylabel('Sensitivity indices for Grating Coupler : Wavelength and Period\n', f

# ===== BARS REPRESENTING THE SENSITIVITY INDICES =====
ax.bar(np.arange(problem['num_vars']), si1,
       color='DarkSeaGreen');

#in striped grey : not significant indices
ax.fill_between(x=[-0.5, 5.5], y1=-0.1, y2=0.1, color='grey', alpha=0.2, hatch='//',



# take a closer look to undestand interactions (looks even better on your presentati
# this part can be done without analyzing with SALib, just with the ouput from the s

fig, ax = plt.subplots()
fig.set_size_inches(8, 7)

ax.tick_params(labelsize=15)
ax.set_title('Maximum Radiation Angle\n'
             'according to the value of the 2 most influent parameters',
             fontsize=20)
```

Out[12]:

```
Text(0.5, 1.0, 'Maximum Radiation Angle\naccording to the value of the 2 most influen
t parameters')
```


Sensitivity indices for Grating Coupler : Wavelength and Period
1.0
0.8
0.6
0.4
0.2
0.0
Wavelength
GratingPeriod


Maximum Radiation Angle
according to the value of the 2 most influent parameters
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [13]:

```
####################################################################
```

In [ ]:

In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```