Pont de Wheatstone

- Equations de base du pont.
- Relation entre la variation de résistance et la tension
- Influence de la plage de variation
- Linéarisation
    - Identification
    - Dérivée

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn
from sklearn.linear_model import LinearRegression

def plotIt(x,y, title="sortie", xlab="X", ylab="Y"):
    fig, ax = plt.subplots()
    ax.plot(x, y)
    ax.set_xlabel(xlab)
    ax.set_ylabel(ylab)
    ax.set_title(title)
    ax.grid(True, which='both')

    seaborn.despine(ax=ax, offset=0)

def plotIt2(x1,y1,x2,y2,title="sortie", xlab="X", ylab="Y", legends=""):
    fig, ax = plt.subplots()
    ax.plot(x1, y1, x2,y2)
    ax.set_xlabel(xlab)
    ax.set_ylabel(ylab)
    ax.set_title(title)
    ax.grid(True, which='both')
    ax.legend(legends)

    seaborn.despine(ax=ax, offset=0)
```

```
vR0=1000
Vp=1
```

```
def vUDiff(DR):
    Ug=(vR0+DR)/(2*vR0+DR)
    Ud=vR0/(2*vR0)
    return Ug-Ud
```

Influence de la plage de mesure

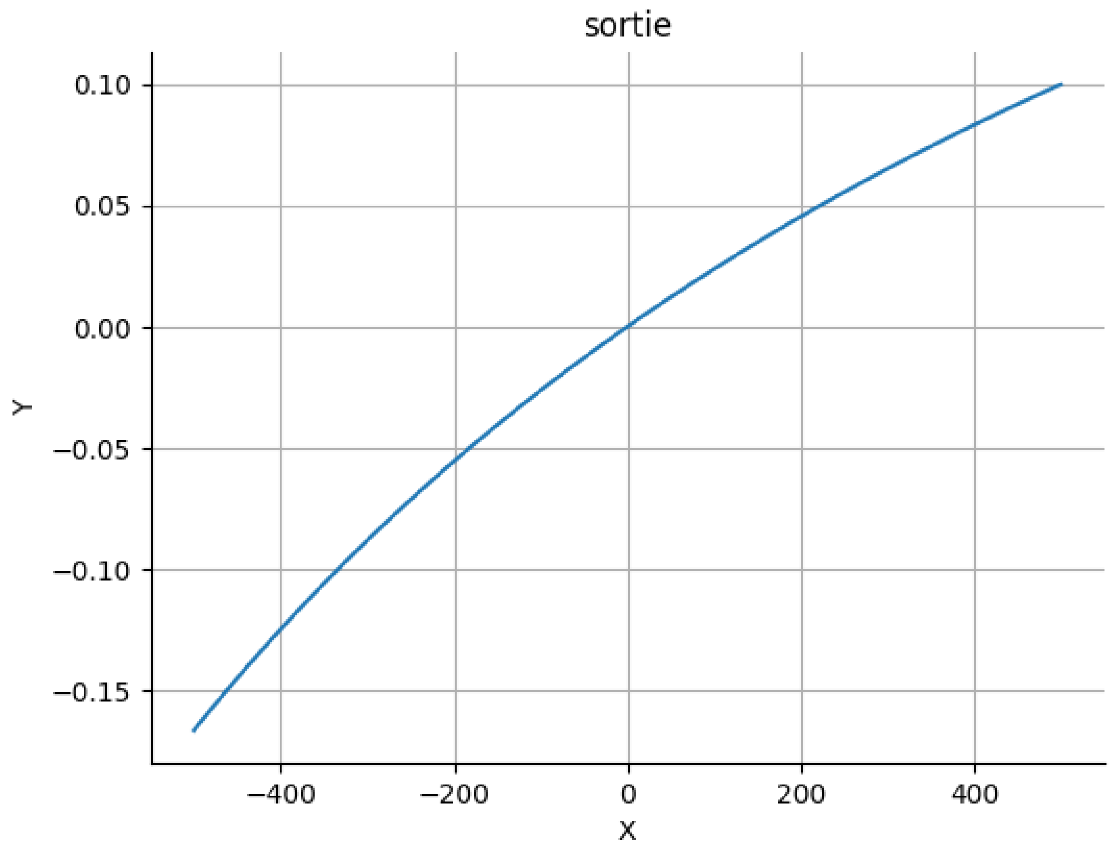
Varier la plage de mesure. Commencer avec 500 ohms pour voir l'effet, puis 50 ohms pour suggérer que c'est négligeable pour une petite variation.

```
spanDR=500
```

```
vDR=np.arange(-spanDR,spanDR,1)
vUD=np.zeros(len(vDR))
for i in range(len(vDR)):
    vUD[i]=vUDiff(vDR[i])
```

::: {.cell 0='l' 1='a' 2='b' 3='e' 4='l' 5=' ' 6='f' 7='i' 8='g' 9='-' 10='s' 11='o' 12='r' 13='t' 14='i' 15='e' 16='-' 17='l' 18='a' 19='r' 20='g' 21='e' 22='-' 23='v' 24='a' 25='r' 26='i' 27='a' 28='t' 29='i' 30='o' 31='n' execution_count=29}

```
plotIt(vDR, vUD)
```

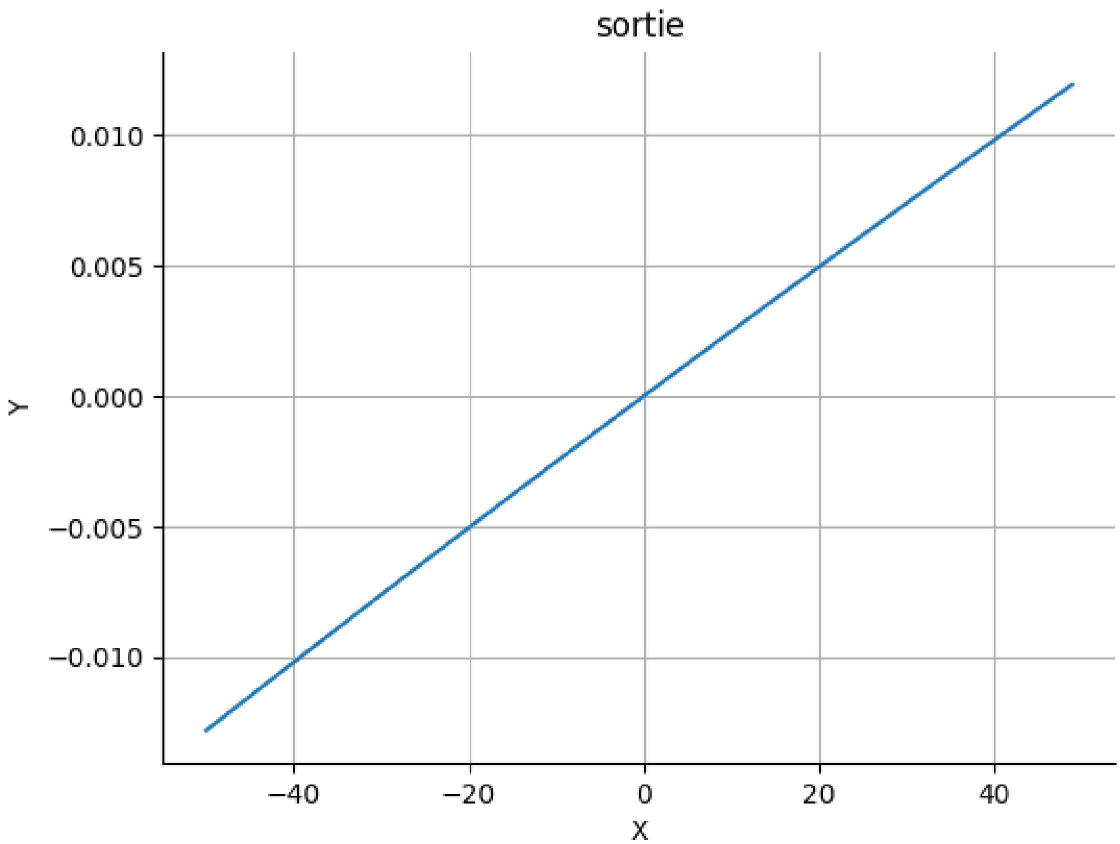


:::

```
spanDR=50
vDR=np.arange(-spanDR,spanDR,1)
vUD=np.zeros(len(vDR))
for i in range(len(vDR)):
    vUD[i]=vUDiff(vDR[i])
```

::: {.cell 0='l' 1='a' 2='b' 3='e' 4='l' 5=' ' 6='f' 7='i' 8='g' 9='-' 10='s' 11='o' 12='r' 13='t' 14='i' 15='e' 16='-' 17='p' 18='e' 19='t' 20='i' 21='t' 22='e' 23='-' 24='v' 25='a' 26='r' 27='i' 28='a' 29='t' 30='i' 31='o' 32='n' execution_count=31}

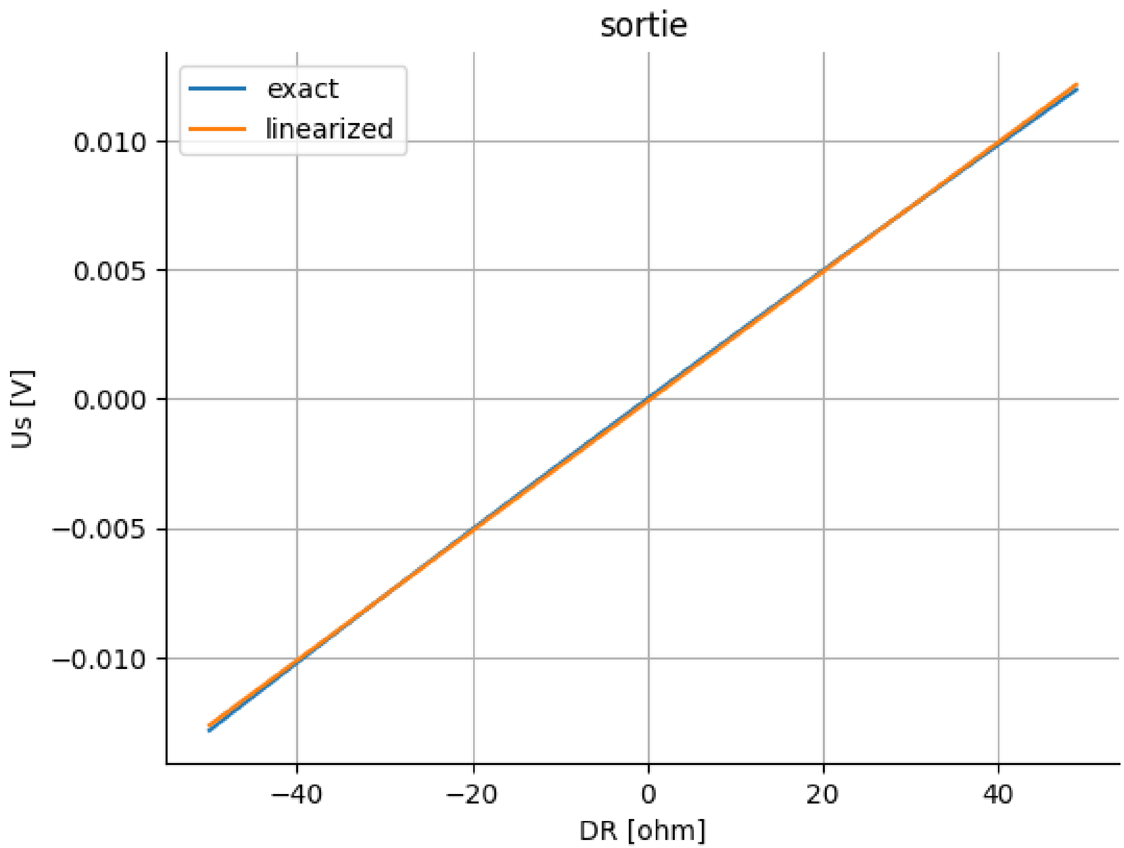
```
plotIt(vDR, vUD)
```



:::

```
model = LinearRegression()
vDRt=vDR.reshape((-1, 1))
tmp=model.fit(vDRt, vUD)
vPred= model.predict(vDRt)
```

```
plotIt2(vDR, vUD, vDR, vPred, xlab="DR [ohm]", ylab="Us [V]", legends=["exact","linearized
```



Calculs symboliques

On calcule la valeur algébriquement pour déterminer la pente de la courbe au point d'intérêt.

```
from sympy import symbols
R0, DR = symbols('R0 DR')
def Udiff(R0,DR):
    Ug=(R0+DR)/(2*R0+DR)
    Ud=R0/(2*R0)
    return Ug-Ud
```

```
Udiff(R0, DR)
```

$$\frac{DR + R_0}{DR + 2R_0} - \frac{1}{2}$$

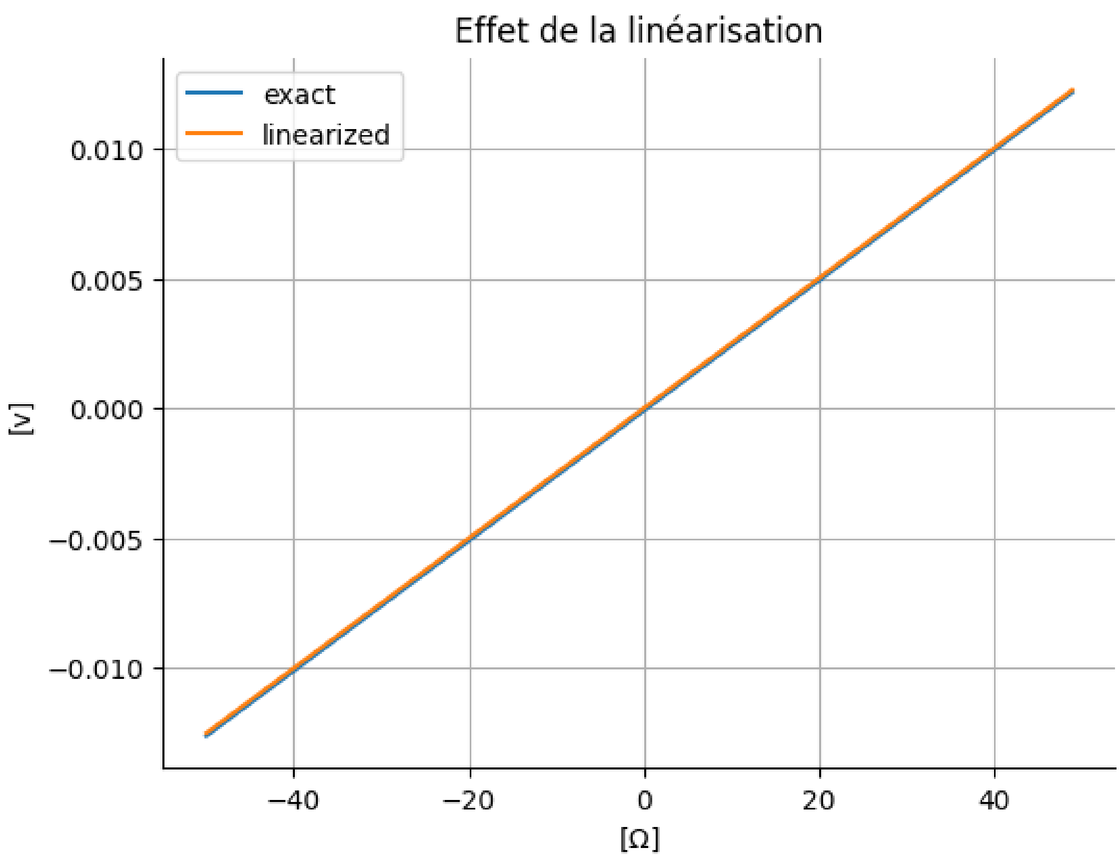
```
dDUdiff = Udiff(R0,DR).diff(DR)
dDUdiff
```

$$-\frac{DR + R_0}{(DR + 2R_0)^2} + \frac{1}{DR + 2R_0}$$

```
# Pente autour du point de fonctionnement
dDUdiff.subs(DR,0)
```

$$\frac{1}{4R_0}$$

```
vDiff = vDR/(4*vR0)
plotIt2(vDR, vPred, vDR, vDiff , xlab="[$\Omega$]", ylab="[v]", title="Effet de la linéari
```



```
plotIt(vDR, vPred-vUD, xlab="[$\Omega$]", ylab="[v]", title="Erreur due à la linéarisation
```