

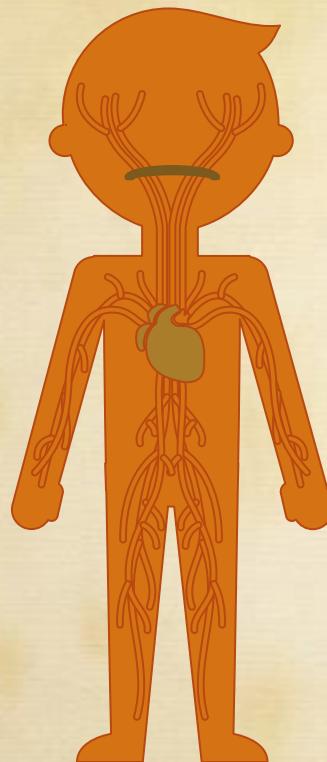
- *Transmission de la puissance électrique sans fil:  
implantations biomédicales*

TIPE 2021-2022

THEME: SANTE ET PREVENTION  
ADHAR Emna



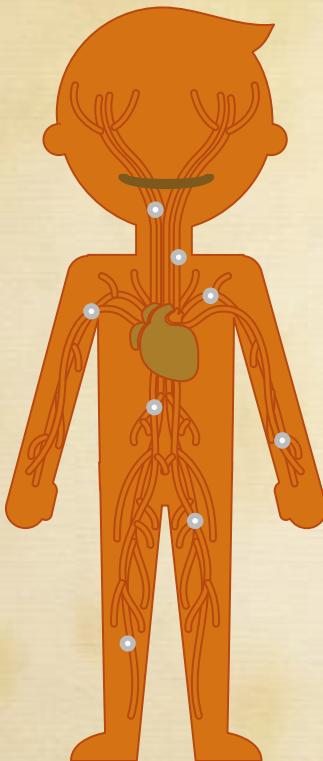
# MOTIVATION



La situation est réelle: vous voulez recharger l'implantation cardiaque de votre patient et vous voulez lui épargner la opération chirurgicale



# OBJECTIF



Charger la batterie par  
INDUCTION



# SOMMAIRE



01

COUPLAGE NON  
RESONNANT

02

COUPLAGE  
RESONNANT

03

ELABORER MES  
PROPRIES MODELES  
POUR AMELIORER LE  
RENDEMENT

Modèle numérique



Modèle expérimental

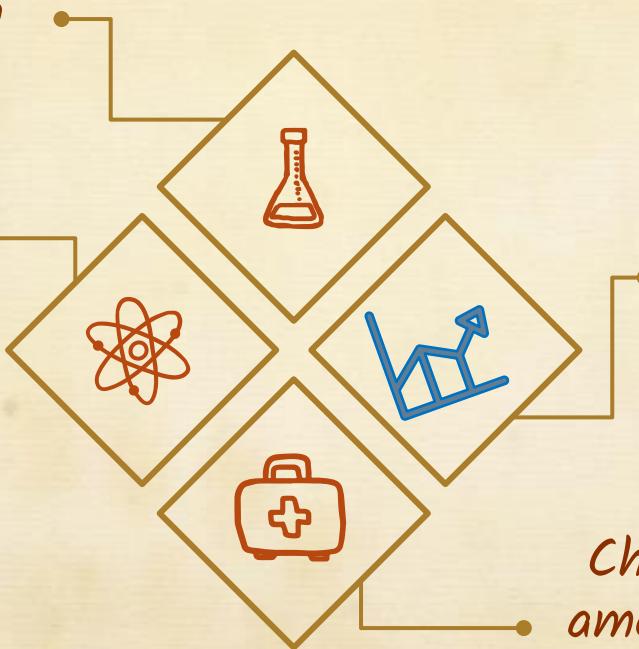
# METHODE DE TRAVAIL

expérimentation

Approche théorique

Résultats

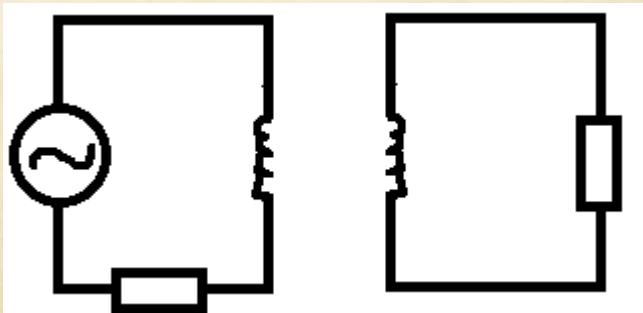
Chercher à améliorer les résultats





# 01

## COUPLAGE NON RESONNANT



# APPROCHE THEORIQUE

- Loi de BIOT-SAVART
- Loi de Faraday
- Calcul du rendement: formule de Yates est-elle valide?



# Loi de Biot-Savart

$$\overrightarrow{B(M)} = \int_{P \in (C)} \frac{\mu_0}{4\pi} i \overrightarrow{dl} \wedge \frac{\overrightarrow{PM}}{PM^3}$$

$$\vec{B}(n) = ?$$

A diagram showing a circular loop of radius  $r$  with current  $i$ . The total current  $I$  is indicated as flowing clockwise. A point  $P$  is shown at a radial distance  $r$  from the center, making an angle  $\theta$  with the horizontal axis.

$$\overrightarrow{B(M)} = \frac{\mu_0 * i}{4\pi * ((r - a)^2 + z^2)^{3/2}} * (az \overrightarrow{er} + (a - r) * a \overrightarrow{ez})$$



# Facteurs à considérer lors de l'expérience

01 Forme de la bobine émettrice

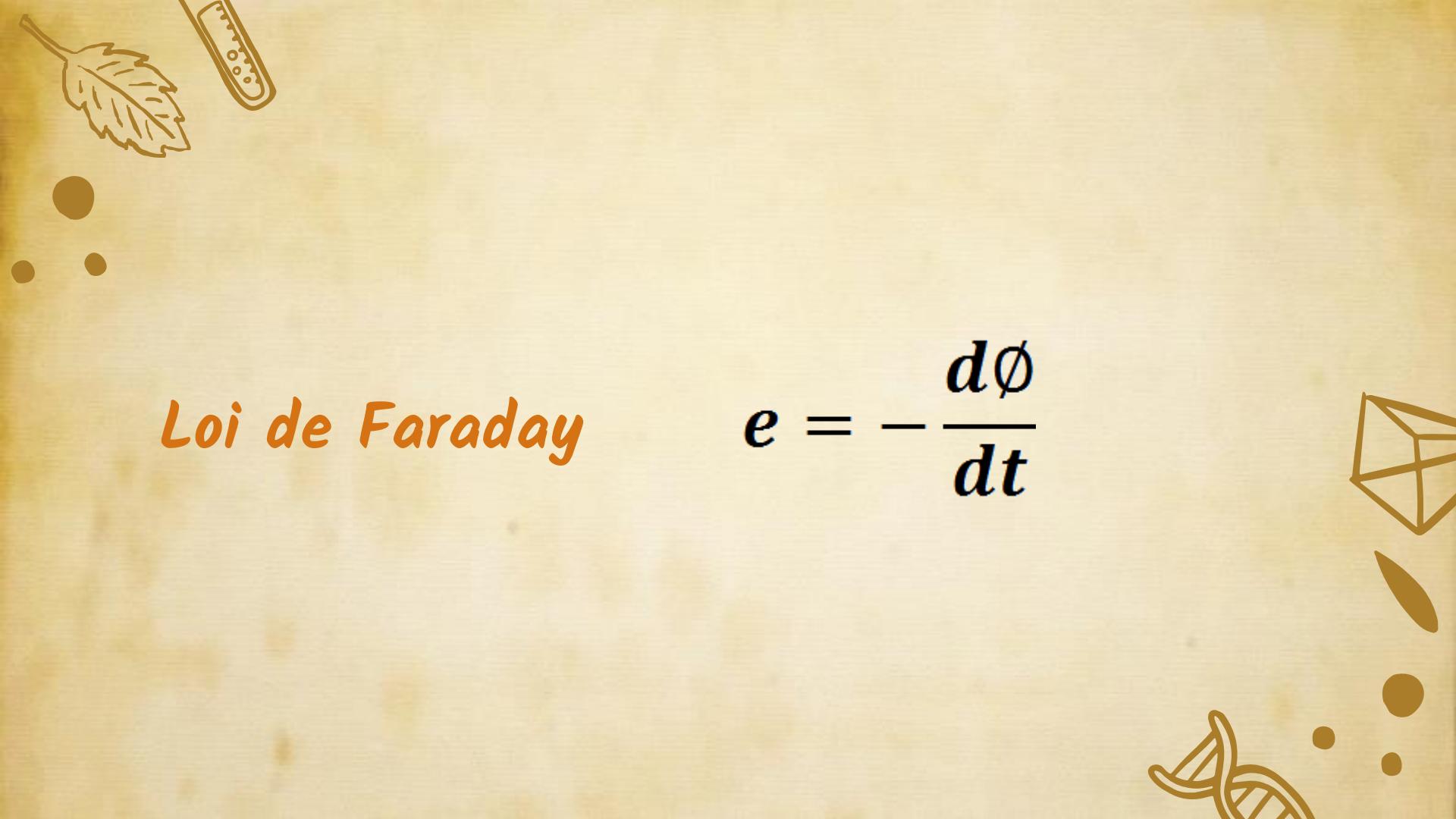
02 Courant qui circule dans le circuit primaire

03 ...

04 ...

05 ...

06 ...



*Loi de Faraday*

$$e = -\frac{d\phi}{dt}$$

# Facteurs à considérer lors de l'expérience

01 Forme de la bobine émettrice

02 Courant qui circule dans le circuit primaire

03 Position de la bobine réceptrice

04 Forme de la bobine réceptrice

05 ...

06 ...

## Calcul du rendement

$$\eta = \frac{1}{R} * \frac{u_{Rm}^2}{u_{Gm}^2} * \sqrt{1 + \left( \frac{L_1 + M}{R} \omega \right)^2}$$

Comment calculer  $M$ ?

$$M = \frac{1}{2\pi f} \sqrt{\frac{u_{Gm}^2}{2 * I_{eff}^2} * L_1}$$

# Formule de Yates (à vérifier)

$$\eta = k \frac{\mu_0^2 N_1^2 N_2^2 a^4 b^4 \omega^2}{R_1 R_2 (d^2 + a^2)^3}$$

- $N_1$**  nombre de spires de la bobine émettrice
- $N_2$**  Nombre de spires de la bobine réceptrice
- $a$**  Rayon de la bobine émettrice
- $b$**  Rayon de la bobine réceptrice
- $d$**  Distance entre les deux bobines

# Facteurs à considérer lors de l'expérience

01 Forme de la bobine émettrice

ET TAILLE!

02 Courant qui circule dans le circuit primaire

03 Position de la bobine réceptrice

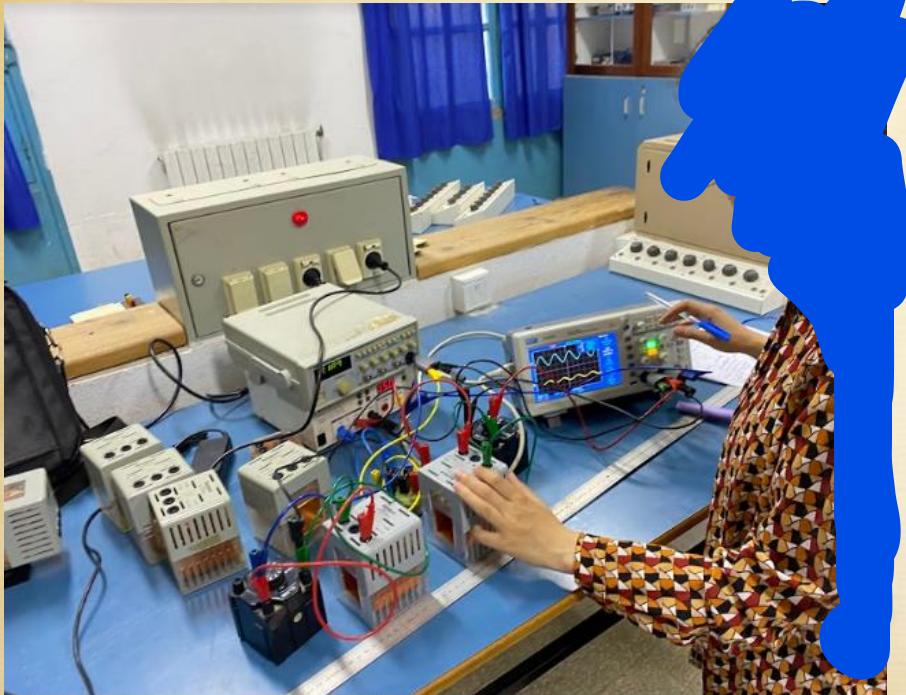
04 Forme de la bobine réceptrice

ET TAILLE!

05 NE PAS OUBLIER DE BRANCHER UN AMPEREMETRE DANS LE CIRCUIT EMETTEUR !

06 Pour une meilleure inductance mutuelle, des bobines différentes ou similaires?

# Réalisation de la première série d'expériences



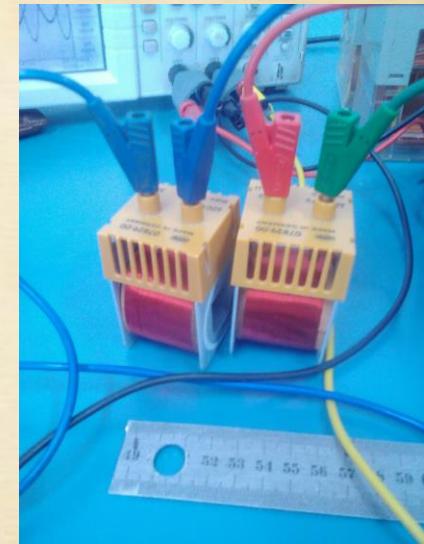
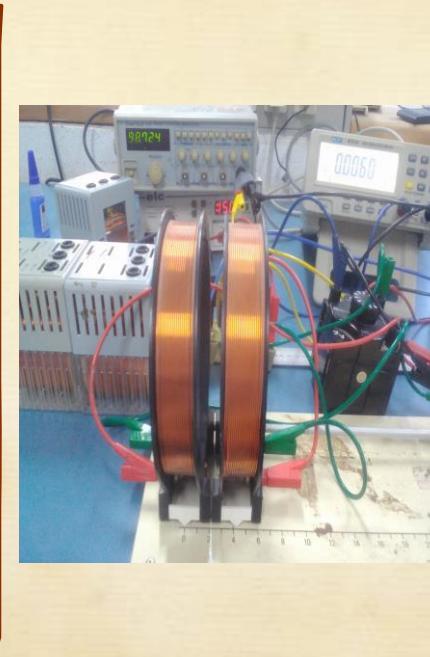
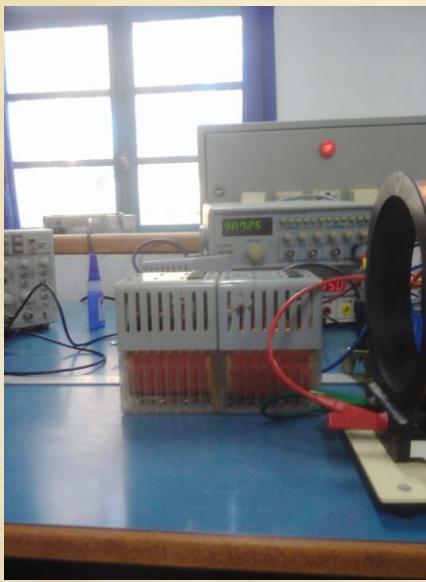
The screenshot shows a Microsoft Excel spreadsheet with a large table of data. The table has columns labeled from A to Z and rows labeled from 1 to 100. Several filters are applied to the data:

- Colonne A filtre :** Filtre sur les colonnes A à Z.
- Colonne B filtre :** Filtre sur les colonnes A à Z.
- Colonne C filtre :** Filtre sur les colonnes A à Z.
- Colonne D filtre :** Filtre sur les colonnes A à Z.
- Colonne E filtre :** Filtre sur les colonnes A à Z.
- Colonne F filtre :** Filtre sur les colonnes A à Z.
- Colonne G filtre :** Filtre sur les colonnes A à Z.
- Colonne H filtre :** Filtre sur les colonnes A à Z.
- Colonne I filtre :** Filtre sur les colonnes A à Z.
- Colonne J filtre :** Filtre sur les colonnes A à Z.
- Colonne K filtre :** Filtre sur les colonnes A à Z.
- Colonne L filtre :** Filtre sur les colonnes A à Z.
- Colonne M filtre :** Filtre sur les colonnes A à Z.
- Colonne N filtre :** Filtre sur les colonnes A à Z.
- Colonne O filtre :** Filtre sur les colonnes A à Z.
- Colonne P filtre :** Filtre sur les colonnes A à Z.
- Colonne Q filtre :** Filtre sur les colonnes A à Z.
- Colonne R filtre :** Filtre sur les colonnes A à Z.
- Colonne S filtre :** Filtre sur les colonnes A à Z.
- Colonne T filtre :** Filtre sur les colonnes A à Z.
- Colonne U filtre :** Filtre sur les colonnes A à Z.
- Colonne V filtre :** Filtre sur les colonnes A à Z.
- Colonne W filtre :** Filtre sur les colonnes A à Z.
- Colonne X filtre :** Filtre sur les colonnes A à Z.
- Colonne Y filtre :** Filtre sur les colonnes A à Z.
- Colonne Z filtre :** Filtre sur les colonnes A à Z.

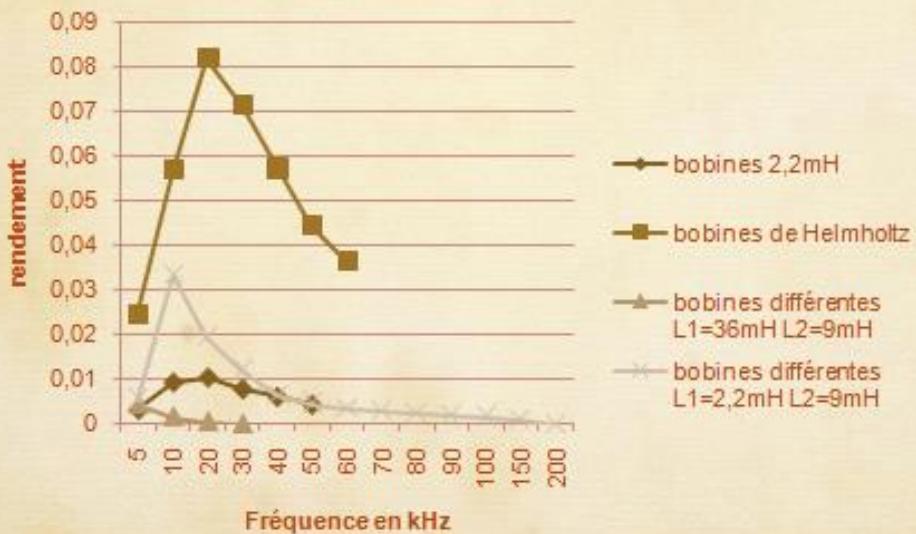
The data table contains numerous rows of numerical values, mostly in the range of 1000 to 2000. The filters are applied across all columns, resulting in many rows being hidden or displayed based on the filter criteria.

01

# Forme de la bobine émettrice



Plusieurs mesures pour différentes bobines à différentes fréquences



Formule de Yates n'est pas valide en Hautes fréquences:  
l'effet de peau

# Facteurs à considérer lors de l'expérience

01 Forme de la bobine émettrice ✓

Meilleur rendement pour les bobines de Helmholtz

02 Courant qui circule dans le circuit primaire ✓

Pushpull pour amplification+ choisir une fréquence adéquate

03 Position de la bobine réceptrice

04 Forme de la bobine réceptrice ✓

Après avoir essayé différentes combinaisons possibles, les meilleures sont celles de Helmholtz

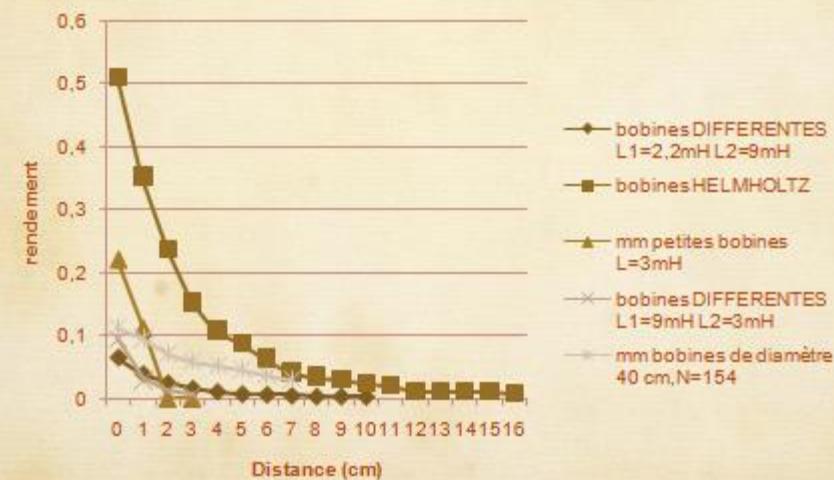
05 NE PAS OUBLIER DE BRANCHER UN AMPEREMETRE DANS LE CIRCUIT EMETTEUR!

DONE! Ainsi on a pu calculer l'inductance mutuelle et le rendement

06 Pour une meilleure inductance mutuelle, des bobines différentes ou similaires? ✓

Les inductances ne sont pas nécessairement similaires

On fait varier la distance entre les bobines=> tester différentes positions de la bobine réceptrice  
(on travaille à la fréquence adéquate pour chaque montage)



# Facteurs à considérer lors de l'expérience

01 Forme de la bobine émettrice ✓

Meilleur rendement pour les bobines de Helmholtz

02 Courant qui circule dans le circuit primaire ✓

Pushpull pour amplification+ choisir une fréquence adéquate

03 Position de la bobine réceptrice ✓

A une grande influence sur le rendement, les meilleures bobines qui y résistent sont de loin celles de Helmholtz

04 Forme de la bobine réceptrice ✓

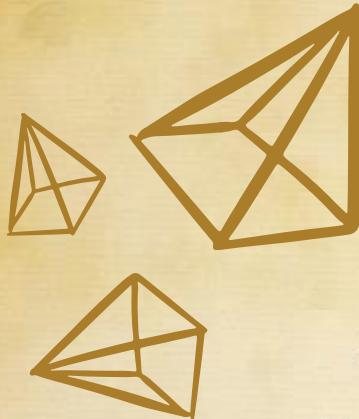
Après avoir essayé différentes combinaisons possibles, les meilleures sont celles de Helmholtz

05 NE PAS OUBLIER DE BRANCHER UN AMPEROMETRE DANS LE CIRCUIT EMETTEUR!

DONE! Ainsi on a pu calculer l'inductance mutuelle et le rendement

06 Pour une meilleure inductance mutuelle, des bobines différentes ou similaires? ✓

Les inductances ne sont pas nécessairement similaires



## *bobines de Helmholtz*



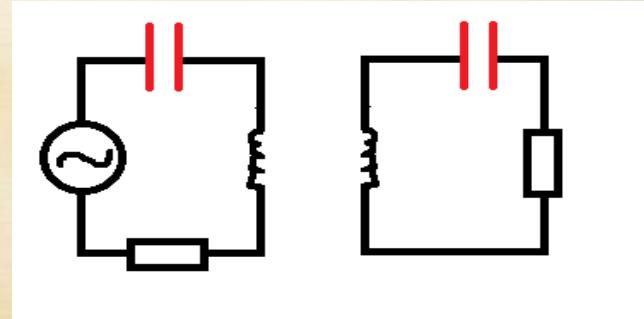
bobines de  
Helmholtz





# 02

## COUPLAGE RESONNANT



# APPROCHE THEORIQUE

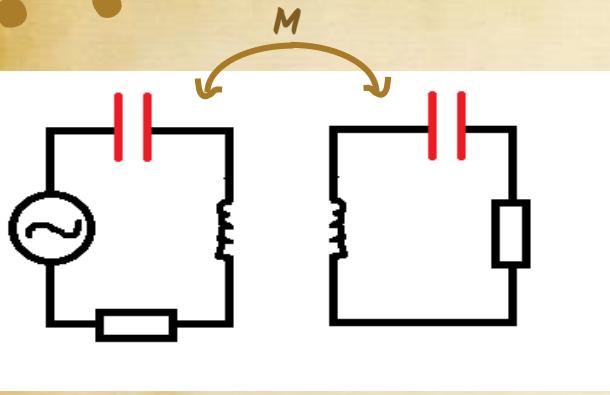
- Pourquoi les deux circuits doivent-être identiques ?
- Oscillations forcées en régime harmonique, notion de résonnance et d'antirésonnance,
- Calcul du rendement



# Doit-on choisir des circuits identiques ?

Démonstration annexe 1

# Résonance/ antirésonance



on applique la loi des mailles en négligeant les résistances

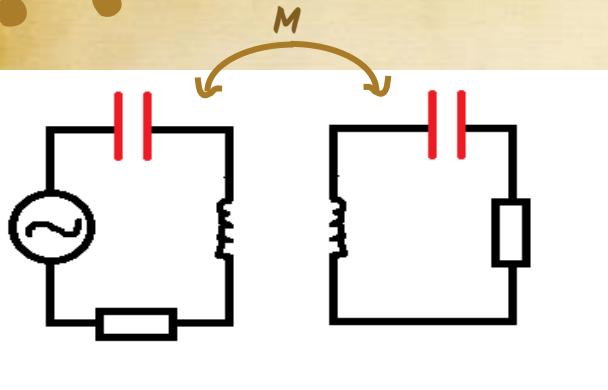
$$\underline{i_1} = j\omega * \frac{k}{M} * \frac{\omega^2 - \omega_0^2}{\omega^4 k^2 - (\omega^2 - \omega_0^2)^2} * \underline{E}$$

$$\underline{i_2} = j\omega * \frac{k^2 * \omega^2}{M} * \frac{1}{(\omega^2 - \omega_0^2)^2 - \omega^4 * k^2} \underline{E}$$



- Deux pulsations annulent le dénominateur: les intensités tendent alors vers l'infini et l'on observe le phénomène de résonance
- Pour  $\omega = \omega_0$ ,  $i_1$  s'annule, c'est la pulsation d'antirésonance

# Résonance/ antirésonance



On note les deux pulsations de résonances  $\omega_d$  et  $\omega_s$   
Comme elles sont les zéros positifs du polynôme:

$$P(X) = k^2 X^4 - (X^2 - \omega_0^2)^2$$

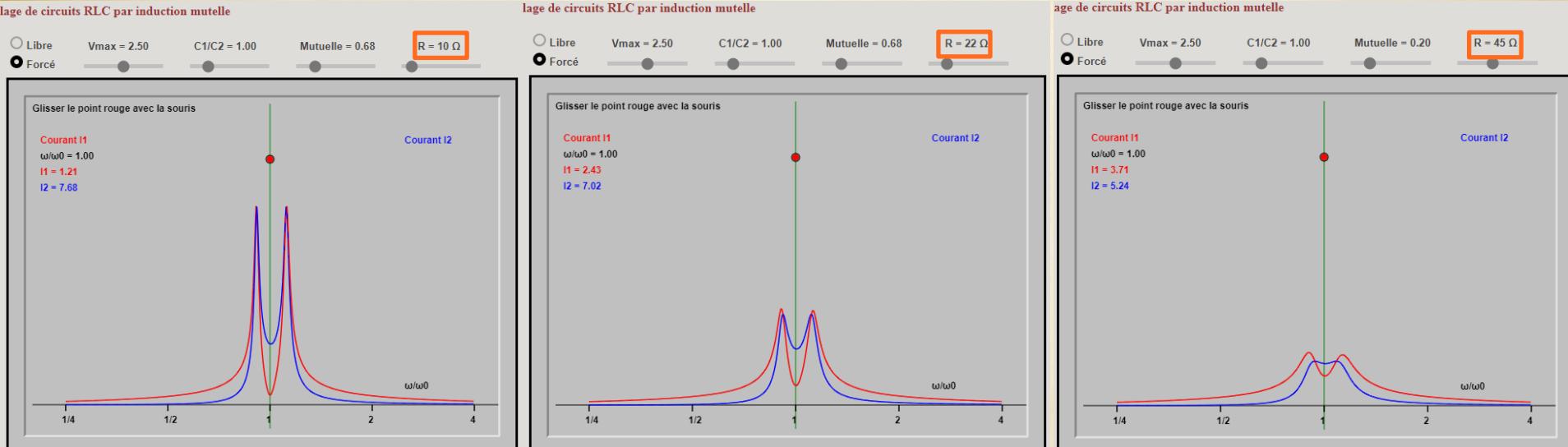
Alors on a:

$$\omega_s \omega_d = \frac{\omega_0^2}{\sqrt{1 - k^2}}$$

En découle une façon de calculer  $k$  !

# Résonance/ antirésonance

Résistance  
=> Atténuation de l'acuité de la  
résonance



Simulation réalisée par le site: [Couplage de circuits RLC par induction mutuelle \(univ-lemans.fr\)](http://www.univ-lemans.fr)

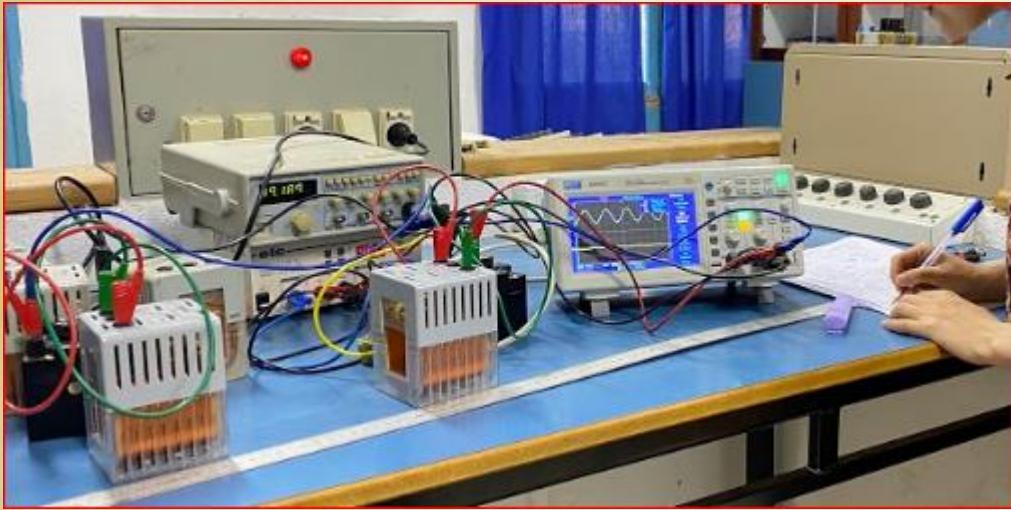


# Calcul du rendement

On tient compte de la résistance

$$\eta = \frac{1}{R^2} * \frac{u_{Rmax}^2}{E_{max}^2} * \frac{\left( R^2 - \left( \omega L - \frac{1}{C\omega} \right)^2 + (\omega M)^2 \right) + 4R^2(\omega L - \frac{1}{C\omega})^2}{R^2 + (\omega M)^2 + (\omega L - \frac{1}{C\omega})^2}$$

# Deuxième série d'expériences



The screenshot shows a Microsoft Excel spreadsheet with a large data table. The table has approximately 20 columns and over 100 rows. The columns are labeled with dates (Year, Month, Day, Hour, Minute, Second) and various numerical values. A red box highlights a specific row in the 2012 column, specifically the row for January 1st, 2012, at 12:00:00. The rest of the table is filled with similar data for other years and months.

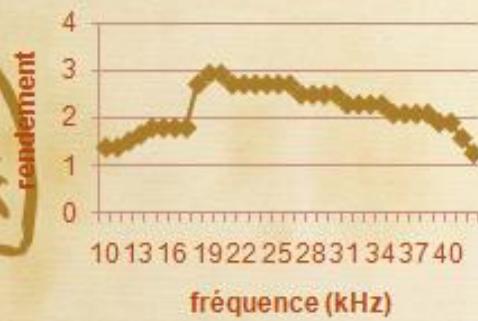


# En ajoutant deux condensateurs...

Avant ...

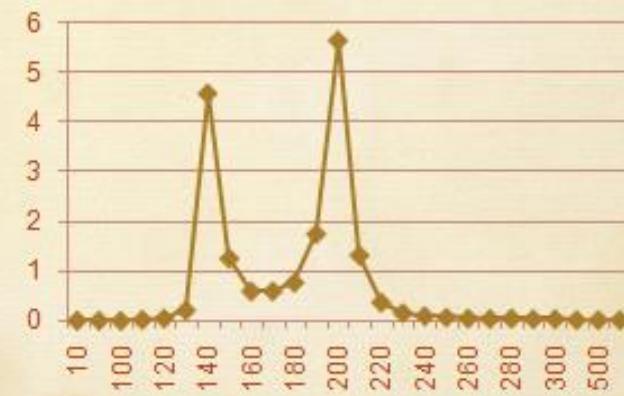
bobines de Helmholtz: couplage

non résonnant



Après ...

bobines de Helmholtz couplage résonnant



Quelle fréquence précisément ?

bobines de  
Helmholtz  
couplage  
résonnant



# Quelles fréquences précisément ?

Mesures précédentes...

*bobines de Helmholtz couplage*

*résonnant*

bobines de Helmholtz

bobines de Helmholtz  
couplage résonnant

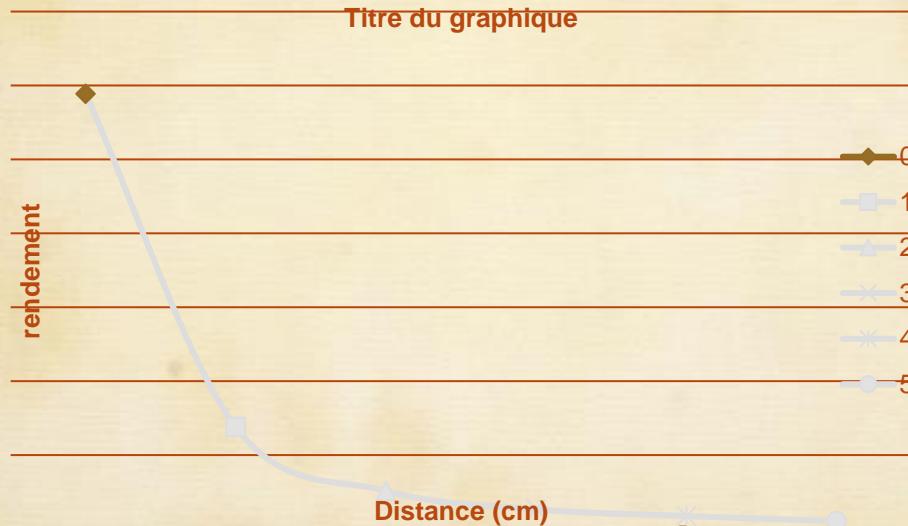
En plus fin...

*Titre du graphique*



142kHz et 200kHz  
=>  $\sqrt{wd*ws} \neq w_0$  => le couplage n'est pas faible !

*on se fixe dans les conditions optimales qu'on a trouvées, et on fait varier la distance entre les deux bobines*



*Et si on rapprochait des plaques métalliques des circuits couplés ?*

*sans plaques*

2V

AVEC UNE SEULE  
PLAQUE COTE  
RECPTEUR

1,52V

AVEC UNE SEULE  
PLAQUE COTE  
EMETTEUR

1,92V

*Avec deux  
plaques*

2V

*Explication? Naissance des courants de Foucault*

*Et maintenant que  
nous avons une idée  
des conditions  
optimales du  
couplage résonnant,  
concevons notre  
propre modèle de  
recharge de  
l'implantation  
cardiaque !*





03

ELABORER MES  
PROPRIES  
MODELES POUR  
AMELIORER LE  
RENDEMENT



# Conception du modèle

Facteurs à prendre en considération

Résultats des expériences précédentes

Les meilleures des bobines pour transfert de puissance électrique ont été les bobines de helmholtz, fonctionnant en couplage résonnant, en absence de tout autre métal

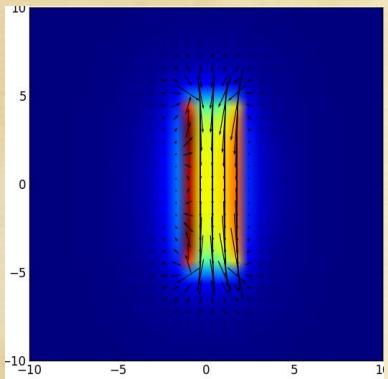


*Commençons par une  
simulation numérique  
pour mieux choisir!*

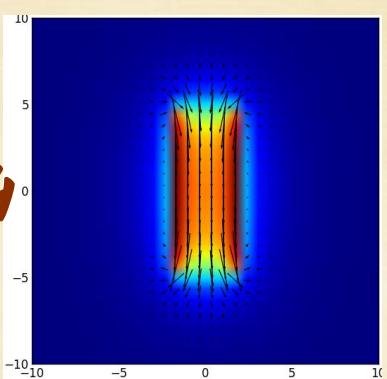
*La loi de BIOT-SAVART m'a permis de  
rédiger un code Python qui a donné  
naissance à ..*



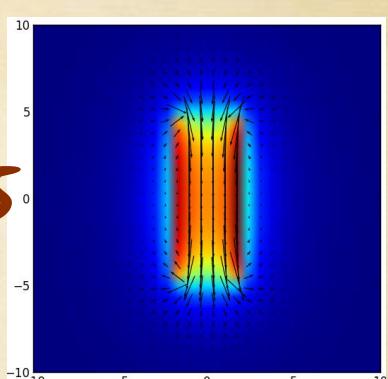
$N=3$



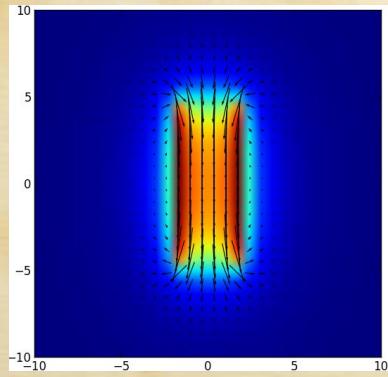
$N=4$



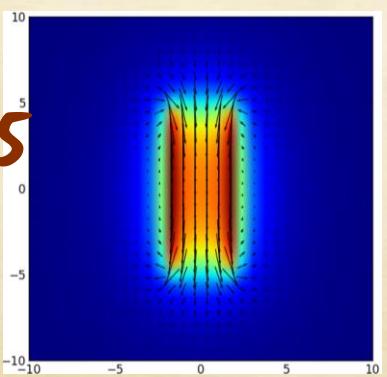
$N=5$



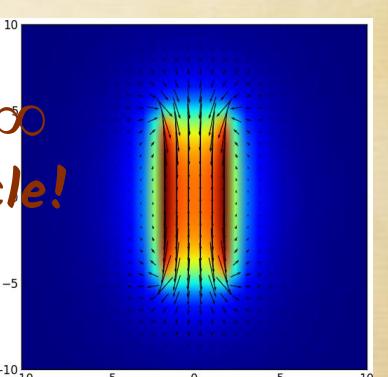
$N=6$



$N=15$



$N=+\infty$   
Un cercle!



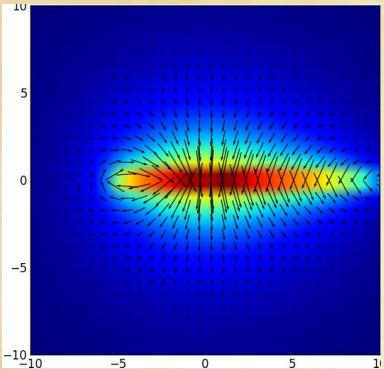
Des polygones à  $N$  côtés dont j'ai  
choisi le rayon=2cm et le nbre de  
spires=100

*Plus on fait augmenter  $N$ ,  
plus le solénoïde rayonne  
=> le solénoïde à spires  
circulaires serait alors le  
meilleur de tous !*

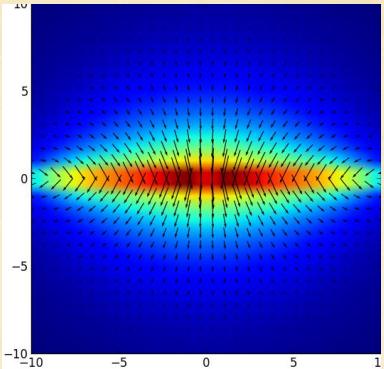
# ET EN 2D?



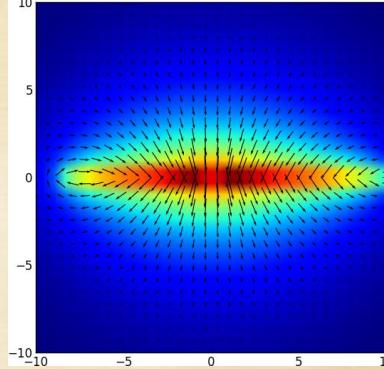
$N=3$



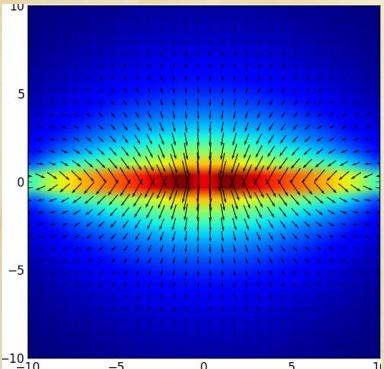
$N=4$



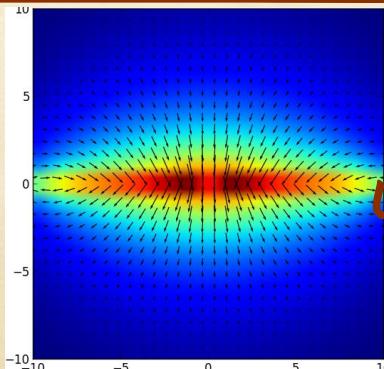
$N=5$



$N=6$

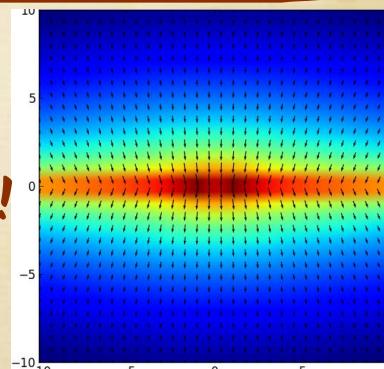


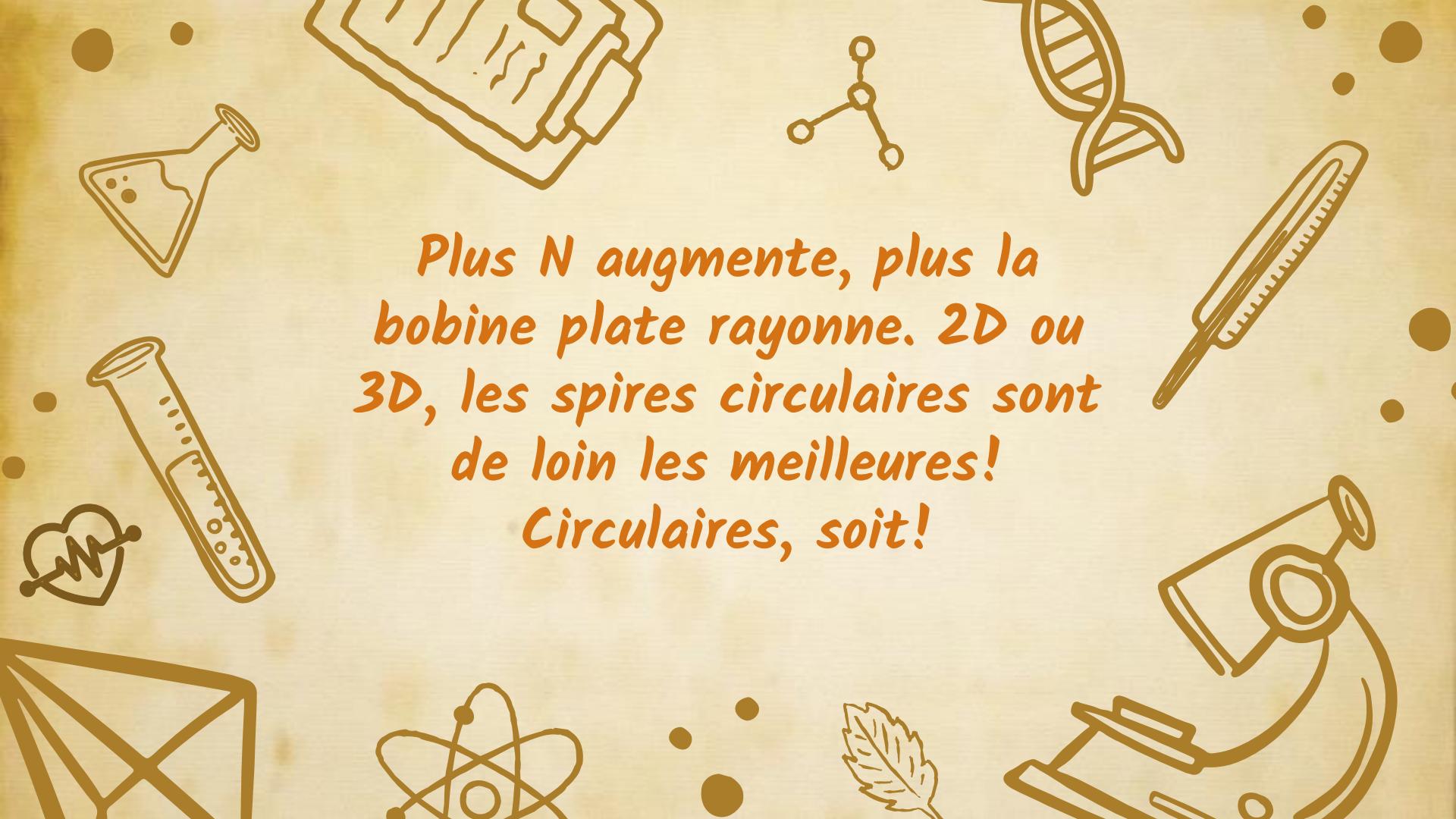
$N=15$



Des polygones à  $N$  côtés dont j'ai  
choisi le nbre de spires=100

$N=+\infty$   
Un cercle!

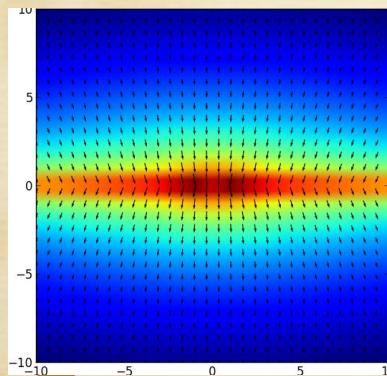




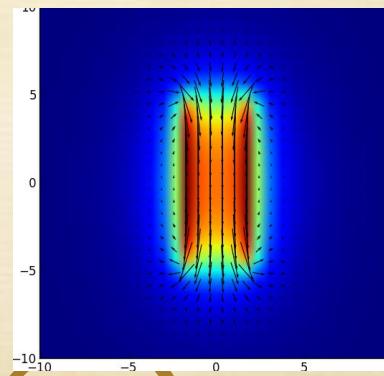
Plus  $N$  augmente, plus la bobine plate rayonne. 2D ou 3D, les spires circulaires sont de loin les meilleures!  
Circulaires, soit!

# 2D ou 3D? Plate ou solénoïde?

2D

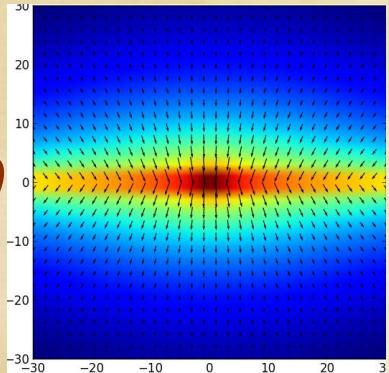


3D

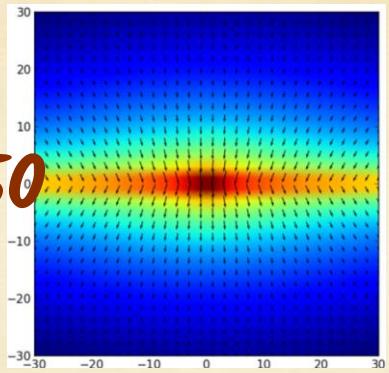


*On choisit les bobines plates à spires circulaires, bien sûr. Mais combien de spires?*

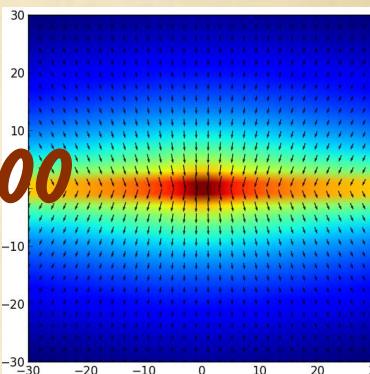
$N=100$



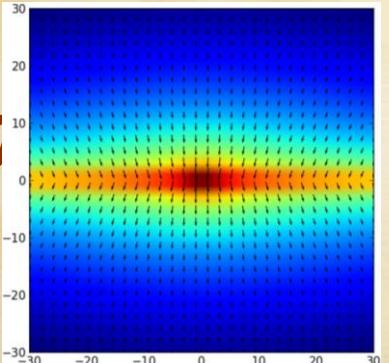
$N=150$



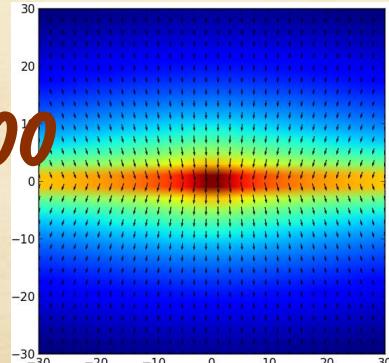
$N=200$



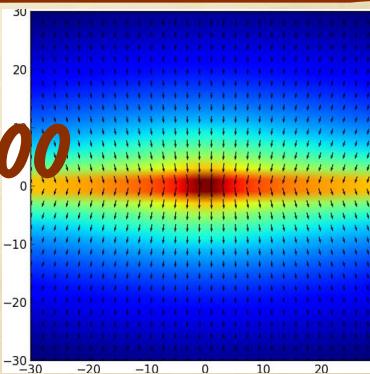
$N=300$



$N=400$



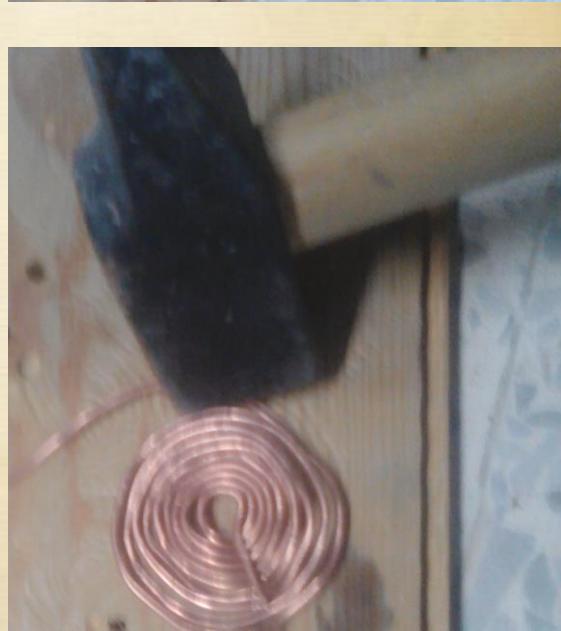
$N=500$



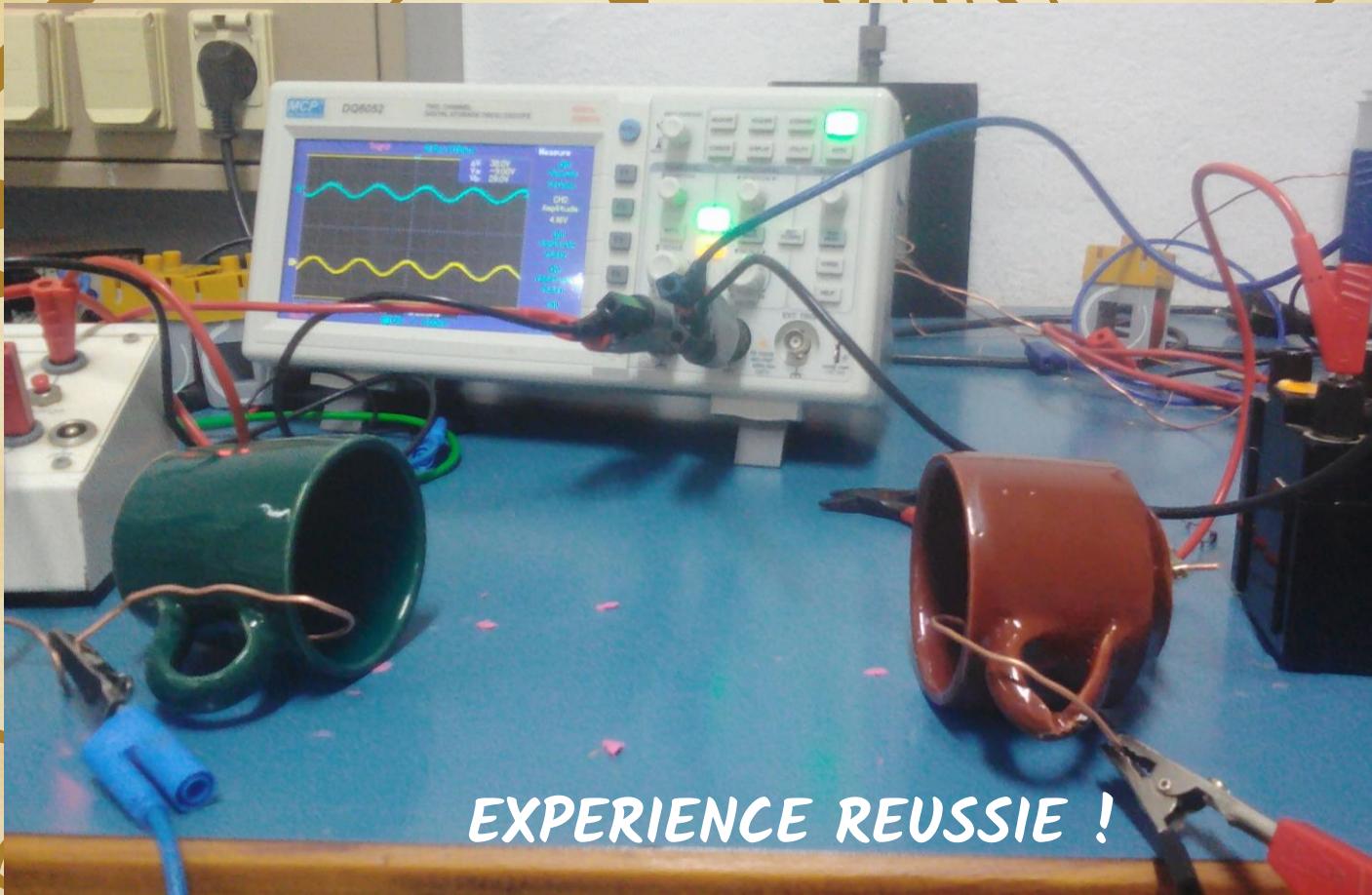
On n'a pas à faire augmenter  
le nombre de spires de notre  
bobine

+ et on trouve dans la  
bibliographie: une cavité  
semi-résonante en céramique  
éliminera les pertes par  
rayonnement.

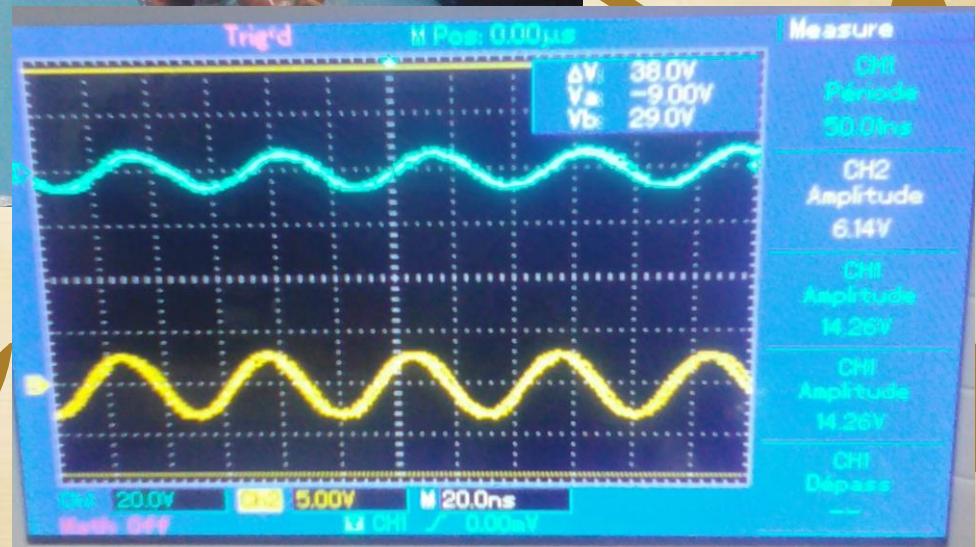
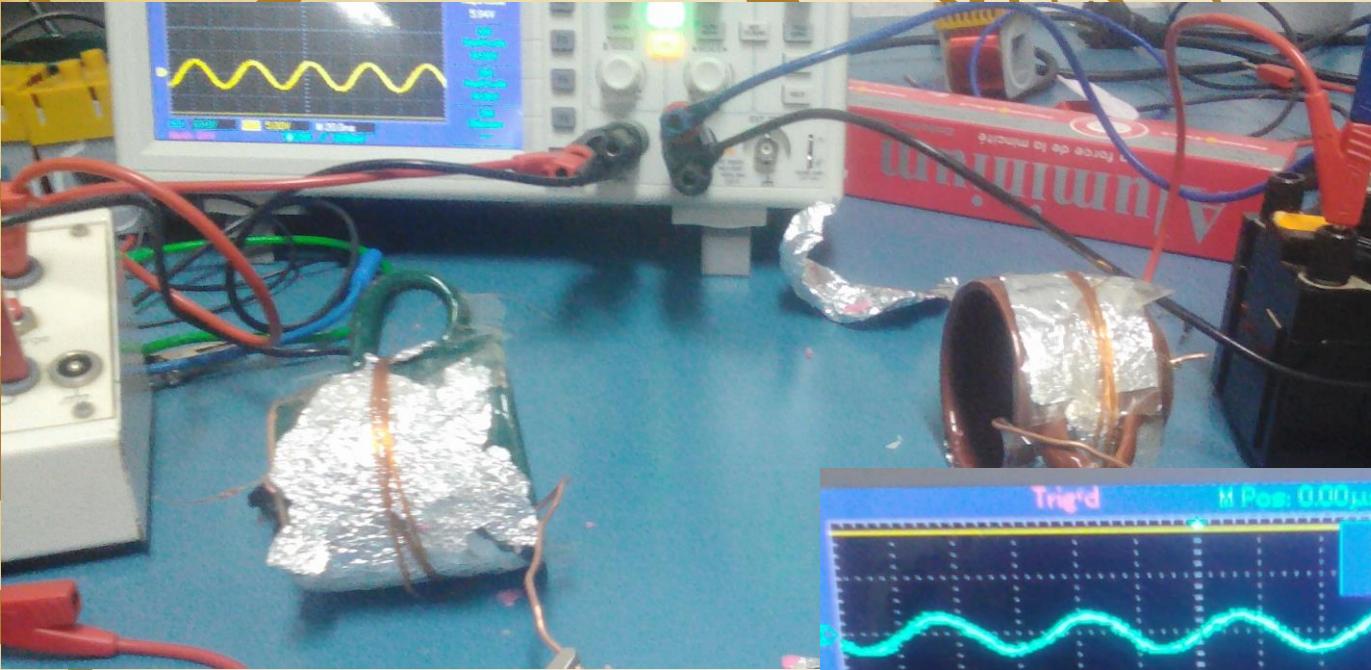
Maintenant, à nos outils!







EXPERIENCE REUSSIE !

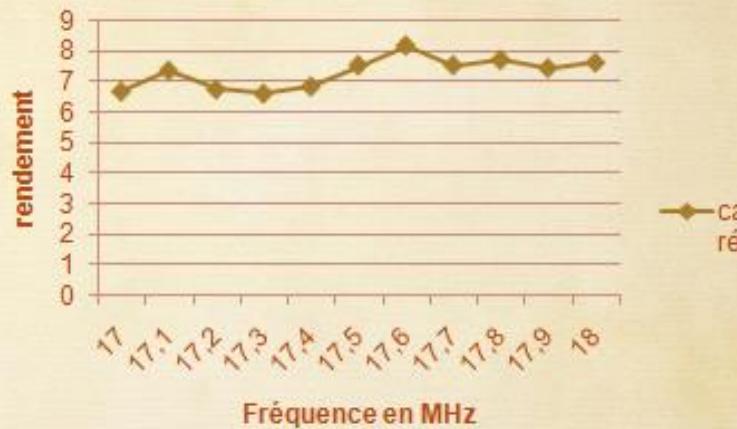


## CAVITE SEMI-RESONANTE



À la recherche des valeurs exactes  
des fréquences de résonance

## cavités semi-résonantes

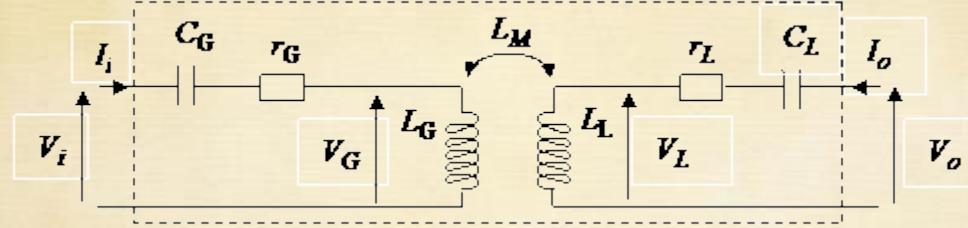


# *conclusion*

- La cavité semi-résonante, assurant un couplage résonant, permet un transfert de puissance électrique très fiable, et résiste au désalignement comme à la distance qui sépare émetteur et récepteur
- On peut l'améliorer encore plus en argentant les fils, et en ajoutant deux résonateurs en proximité
- Ceci est adaptable pour une implantation cardiaque, dans la mesure où on peut concevoir des cavités portables par le patient

# ANNEXE I

Le schéma électrique le plus général concernant deux circuits résonants couplés par le champ magnétique extérieur associé aux inductances est le suivant :



**Schéma équivalent le plus général pour l'étude du couplage magnétique entre circuits résonants**

On peut dans un premier temps considérer le système sous la forme d'un quadripôle sans tenir compte de la charge finale. Pour simplifier les expressions on utilise les notations suivantes :

$$\begin{cases} x = \frac{1}{L_L C_L \omega^2} \\ y = \frac{1}{L_G C_G \omega^2} \end{cases}$$

et

$$\begin{cases} Q_G = \frac{L_G \omega}{r_G} \\ Q_L = \frac{L_L \omega}{r_L} \end{cases}$$

Notons que l'on suppose implicitement que les pertes sont associées seulement aux inductances. Un calcul élémentaire conduit à la forme matricielle suivante :

# ANNEXE I

$$\begin{pmatrix} V_i \\ V_o \end{pmatrix} = j\omega \begin{pmatrix} L_g(1-x+j/Q_g) & L_m \\ L_m & L_t(1-y+j/Q_t) \end{pmatrix} \begin{pmatrix} I_i \\ I_o \end{pmatrix}$$

Le déterminant de cette matrice est :

$$Det = j\omega L_g L_t [1-x+j/Q_g)(1-y+j/Q_t) - k^2]$$

Il est assez facile de montrer que ce déterminant ne s'annule jamais et ainsi que la matrice peut toujours être inversée. Par ailleurs cette matrice est très similaire à celle du couplage magnétique, elle n'en diffère que par les termes diagonaux entre parenthèses. Il est ainsi tentant de définir un coefficient de couplage généralisé par la formule :

$$K^2 = \frac{L_m^2}{L_g(1-x+j/Q_t)L_t(1-y+j/Q_g)}$$

En utilisant la définition du coefficient de couplage magnétique on obtient :

$$K^2 = \frac{k^2 Q_t Q_g}{[j+Q_t(1-x)][j+Q_g(1-y)]}$$

Si on pose :

$$\begin{cases} A_t = \frac{Q_t}{j+Q_t(1-x)} \\ A_g = \frac{Q_g}{j+Q_g(1-y)} \end{cases}$$

On obtient :  $K^2 = k^2 A_t A_g$

# ANNEXE I

Le couplage originel est donc multiplié par un gain de chaque côté de la liaison. Les valeurs optimales sont obtenues pour :  
 $x = y = 1$

C'est à dire lorsque les deux circuits ont la même fréquence de résonance et sont accordés sur cette dernière, on a alors :

$$|A_L|_{\max} = Q_L \quad |A_G|_{\max} = Q_G$$

Et finalement :

$$|K^2|_{\max} = k^2 Q_L Q_G$$

L'énergie transférée étant proportionnelle au carré du coefficient de couplage, tout se passe donc comme si chaque résonance, celle de la charge et celle du générateur, amplifiait le transfert d'énergie dans un facteur égal au facteur de qualité du circuit correspondant.

Source:

[www.tmms.co.jp/Nearfield/approche\\_pedagogique/Circuits\\_resonants\\_couples.htm](http://www.tmms.co.jp/Nearfield/approche_pedagogique/Circuits_resonants_couples.htm)

# ANNEXE 2: code PYTHON N spires circulaires 2D

76 N spires circulaires 2D.py - C:\Doc Emna\type implantation cardiaque\N spires circulaires 2D.py

```
File Edit Format Run Options Windows Help
import numpy as np

# mu0 = 12.566370614e-7
mu0 = 4 * np.pi
I = 1.
N=int(input("donner le nombre de spires désirées"))

def circular_loop(R, C = [0., 0., 0.], N = 200):
    XP = R * np.cos(np.linspace(0, 2*np.pi, N)) - C[0]
    ZP = R * np.sin(np.linspace(0, 2*np.pi, N)) - C[1]
    YP = np.zeros(N) + C[2]
    return XP, YP, ZP

def biot_savart(xP, yP, zP, XM, YM, ZM):
    """
    xP, yP et zP sont des ndarrays à une dimension contenant
    les NP coordonnées des points P représentant la distribution
    de courant.

    XM, YM sont des ndarrays à deux dimensions obtenus typiquement
    par
    XM, YM = np.meshgrid(np.linspace(xmin, xmax, NX), np.linspace(ymin, ymax, NY))

    ZM est un ndarray à deux dimensions représentant la cote des points de la
    grille XM, YM (typiquement rempli de 0 : observation dans le plan z = 0).

    La fonction renvoie deux ndarrays de dimension (NX, NY) qui sont les
    composantes (utiles) du champ dans le plan d'observation.
    """

    P = np.array([xP, yP, zP])
    # P.shape = (3, NP)

    # Vecteur déplacement élémentaire le long de la distribution
    vec_dlp = P[:,1:] - P[:,:-1]
    # vec_dlp.shape = (3, NP-1)

    # Localisation du point courant de la distribution
    mid_dlp = (P[:,1:] + P[:,:-1]) / float(2)
    # mid_dlp.shape = (3, NP-1)
```

76 N spires circulaires 2D.py - C:\Doc Emna\type implantation cardiaque\N spires circulaires 2D.py

```
File Edit Format Run Options Windows Help
XPM = XM[:, :, np.newaxis] - mid_dlp[0, :]
YPM = YM[:, :, np.newaxis] - mid_dlp[1, :]
ZPM = ZM[:, :, np.newaxis] - mid_dlp[2, :]
# XPM.shape = YPM.shape = ZPM.shape = (NX, NY, NP)

PMcubed = (XPM**2 + YPM**2 + ZPM**2)**(3/2.)
# PMcubed.shape = (NX, NY, NP)

# Éviter la division par zéro (trop proche des sources)
PMcubed[PMcubed < 1e-6] = np.nan

Xdl_cross_PM = - YPM * vec_dlp[2, :] + ZPM * vec_dlp[1, :]
Ydl_cross_PM = - ZPM * vec_dlp[0, :] + XPM * vec_dlp[2, :]
Zdl_cross_PM = - XPM * vec_dlp[1, :] + YPM * vec_dlp[0, :]
# Xdl_cross_PM.shape = Ydl_cross_PM.shape = Zdl_cross_PM.shape =
# (NX, NY, NP)

Xdl_cross_PM_over_PMcubed = Xdl_cross_PM / PMcubed
Ydl_cross_PM_over_PMcubed = Ydl_cross_PM / PMcubed
Zdl_cross_PM_over_PMcubed = Zdl_cross_PM / PMcubed

BX = np.sum(Xdl_cross_PM_over_PMcubed, axis = 2)
BY = np.sum(Ydl_cross_PM_over_PMcubed, axis = 2)
BZ = np.sum(Zdl_cross_PM_over_PMcubed, axis = 2)
# BX.shape = BY.shape = BZ.shape = (NX, NY)

BNorme = (BX**2 + BY**2 + BZ**2)**(.5)
# BNorme.shape = (NX, NY)

return mu0 * I * BX/(4 * np.pi), mu0 * I * BY/(4 * np.pi),
       mu0 * I * BNorme/(4 * np.pi)

def Bzloop(z):
    return mu0 * I * R**2/(2 * (R**2 + z**2)**(3/2.))

import matplotlib.pyplot as plt
import matplotlib.cm as cm

R = 1

xP, yP, zP = circular_loop(R, [0., 0., 0.])
```

```
# Grille de NX*NY points
NX = 30
NY = 30

# Coordonnées min et max des points de la grille
xmax = 30
xmin = -xmax
ymax = 30
ymin = -ymax

# Les coordonnées des points de la grille sont répartis uniformément
# sur [xmin, xmax]x[ymin, ymax]
xM = np.linspace(xmin, xmax, NX)
yM = np.linspace(ymin, ymax, NY)

# Création de la grille
XM, YM = np.meshgrid(xM, yM)

# On impose la cote de la grille : z = 0
ZM = np.zeros((yM.size, xM.size))

# Calcul du champ magnétique
# La norme de B (BNorme) permet de normer le champ
BX, BY, BNorme = biot_savart(xP, yP, zP, XM, YM, ZM)

for i in range(1,N-1):
    xP, yP, zP = circular_loop(1+0.7*i, [0., 0., 0.])
    BX2, BY2, BNorme2 = biot_savart(xP, yP, zP, XM, YM, ZM)

    BX += BX2
    BY += BY2
    BNorme += BNorme2

plt.quiver(XM, YM, BX, BY, pivot = 'middle', units = 'width')

plt.imshow(BNorme, interpolation = 'bilinear', origin = 'lower',
           cmap = cm.jet, extent = (xmin, xmax, ymin, ymax))
# plt.contour(XM, YM, BNorme, 10)
plt.show()
```

# ANNEXE 3: code PYTHON N spires polygonales 2D

76 N spires polygone régulier 2D.py - C:\Doc Emna\tipe implantation cardiaque\N spires polygone

```
File Edit Format Run Options Windows Help  
import numpy as np  
  
# mu0 = 12.566370614e-7  
mu0 = 4 * np.pi  
PII=np.pi  
I = 1.  
  
M=int(input('donnez le nombre de cotes désiré pour votre spire'))  
  
C=[0.,0.,0.]  
N=200  
def polygone_loop (RAY, C):  
    XP = []  
    ZP = []  
    YP = []  
    Xs = []  
    Zs = []  
    Ys = []  
    for k in range (M):  
        Xs.append(np.cos(2*k*PII/M)*RAY+C[0])  
        Zs.append(np.sin(2*k*PII/M)*RAY+C[1])  
        Ys.append(0+C[2])  
    for i in range(M-1):  
        X=np.linspace(Xs[i],Xs[i+1],N)  
        XL=list(X)  
        XP=XP+XL  
        Y=np.zeros(N)  
        YL=list(Y)  
        YP=YP+YL  
        Z=np.linspace(Zs[i],Zs[i+1],N)  
        ZL=list(Z)  
        ZP=ZP+ZL  
    X=np.linspace(Xs[M-1],Xs[0],N)  
    XL=list(X)  
    XP=XP+XL  
    Y=np.zeros(N)  
    YL=list(Y)  
    YP=YP+YL  
    Z=np.linspace(Zs[M-1],Zs[0],N)  
    ZL=list(Z)  
    ZP=ZP+ZL
```

76 N spires polygone régulier 2D.py - C:\Doc Emna\tipe implantation cardiaque\N spires polygone régulier 2D.py

```
File Edit Format Run Options Windows Help  
ZP=ZP+ZL  
return XP, YP, ZP  
  
def biot_savart(xP, yP, zP, XM, YM, ZM):  
    """  
    xP, yP et zP sont des ndarrays à une dimension contenant  
    les NP coordonnées des points P représentant la distribution  
    de courant.  
  
    XM, YM sont des ndarrays à deux dimensions obtenus typiquement  
    par  
    XM, YM = np.meshgrid(np.linspace(xmin, xmax, NX), np.linspace(ymax, NY))  
  
    ZM est un ndarray à deux dimensions représentant la cote des points de la  
    grille XM, YM (typiquement rempli de 0 : observation dans le plan z = 0).  
  
    La fonction renvoie deux ndarrays de dimension (NX, NY) qui sont les  
    composantes (utiles) du champ dans le plan d'observation.  
    """  
  
    P = np.array([xP, yP, zP])  
    # P.shape = (3, NP)  
  
    # Vecteur déplacement élémentaire le long de la distribution  
    vec_d1P = P[:,1:] - P[:,:-1]  
    # vec_d1P.shape = (3, NP-1)  
  
    # Localisation du point courant de la distribution  
    mid_d1P = (P[:,1:] + P[:,:-1]) / float(2)  
    # mid_d1P.shape = (3, NP-1)  
  
    XPM = XM[:, :, np.newaxis] - mid_d1P[0, :]  
    YPM = YM[:, :, np.newaxis] - mid_d1P[1, :]  
    ZPM = ZM[:, :, np.newaxis] - mid_d1P[2, :]  
    # XPM.shape = YPM.shape = ZPM.shape = (NX, NY, NP)  
  
    PMcubed = (XPM**2 + YPM**2 + ZPM**2)**(3/2.)  
    # PMcubed.shape = (NX, NY, NP)  
  
    # Éviter la division par zéro (trop proche des sources)  
    PMcubed[PMcubed < 1e-6] = np.nan
```

## 76 N spires polygone régulier 2D.py - C:\Doc Emna\type implantation cardiaque\N spires polygone régulier

```
File Edit Format Run Options Windows Help

Xdl_cross_PM = - YPM * vec_d1P[2,:] + ZPM * vec_d1P[1,:]
Ydl_cross_PM = - ZPM * vec_d1P[0,:] + XPM * vec_d1P[2,:]
Zdl_cross_PM = - XPM * vec_d1P[1,:] + YPM * vec_d1P[0,:]
# Xdl_cross_PM.shape = Ydl_cross_PM.shape = Zdl_cross_PM.shape =
# (NX, NY, NP)

Xdl_cross_PM_over_PMcubed = Xdl_cross_PM / PMcubed
Ydl_cross_PM_over_PMcubed = Ydl_cross_PM / PMcubed
Zdl_cross_PM_over_PMcubed = Zdl_cross_PM / PMcubed

BX = np.sum(Xdl_cross_PM_over_PMcubed, axis = 2)
BY = np.sum(Ydl_cross_PM_over_PMcubed, axis = 2)
BZ = np.sum(Zdl_cross_PM_over_PMcubed, axis = 2)
# BX.shape = BY.shape = BZ.shape = (NX, NY)

BNorme = (BX**2 + BY**2 + BZ**2)**(.5)
# BNorme.shape = (NX, NY)

return mu0 * I * BX/(4 * np.pi), mu0 * I * BY/(4 * np.pi),\
       mu0 * I * BNorme/(4 * np.pi)

def Bzloop(z):
    return mu0 * I * R**2/(2 * (R**2 + z**2)**(3/2.))

import matplotlib.pyplot as plt
import matplotlib.cm as cm

R=1

xP, yP, zP = polygone_loop(R, [0., 0., 0.])

# Grille de NX*NY points
NX = 30
NY = 30

L=int(input("donner le nombre de spires désirées"))

# Coordonnées min et max des points de la grille
xmax = 10
xmin = -xmax
```

```
ymax = 10
ymin = -ymax

# Les coordonnées des points de la grille sont répartis uniformément
# sur [xmin, xmax]x[ymin, ymax]
xM = np.linspace(xmin, xmax, NX)
yM = np.linspace(ymin, ymax, NY)

# Création de la grille
XM, YM = np.meshgrid(xM, yM)

# On impose la cote de la grille : z = 0
ZM = np.zeros((yM.size, xM.size))
|
BX, BY, BNorme = biot_savart(xP, yP, zP, XM, YM, ZM)

for i in range(1,L):
    xP, yP, zP = polygone_loop(R+i*0.1, [0., 0., 0.])
    BX2, BY2, BNorme2 = biot_savart(xP, yP, zP, XM, YM, ZM)

    BX += BX2
    BY += BY2
    BNorme += BNorme2
plt.quiver(XM, YM, BX, BY, pivot = 'middle', units = 'width')
plt.imshow(BNorme, interpolation = 'bilinear', origin = 'lower',
           cmap = cm.jet, extent = (xmin, xmax, ymin, ymax))
plt.show()
```

# ANNEXE 4: code PYTHON N spires circulaires 3D

76 N spires circulaires.py.py - C:\Doc Emna\tipe implantation cardiaque\N spires circulaires.py.py

File Edit Format Run Options Windows Help

```
import numpy as np

# mu0 = 12.566370614e-7
mu0 = 4 * np.pi
I = 1.

def circular_loop(R = 1e-2, C = [0., 0., 0.], N = 200):
    XP = R * np.cos(np.linspace(0, 2*np.pi, N)) - C[0]
    ZP = R * np.sin(np.linspace(0, 2*np.pi, N)) - C[1]
    YP = np.zeros(N) + C[2]
    return XP, YP, ZP

def biot_savart(xP, yP, zP, XM, YM, ZM):
    """
    xP, yP et zP sont des ndarrays à une dimension contenant
    les NP coordonnées des points P représentant la distribution
    de courant.

    XM, YM sont des ndarrays à deux dimensions obtenus typiquement
    par
    XM, YM = np.meshgrid(np.linspace(xmin, xmax, NX), np.linspace(ymin, ymax, NY))

    ZM est un ndarray à deux dimensions représentant la cote des points de la
    grille XM, YM (typiquement rempli de 0 : observation dans le plan z = 0).

    La fonction renvoie deux ndarrays de dimension (NX, NY) qui sont les
    composantes (utiles) du champ dans le plan d'observation.
    """

    P = np.array([xP, yP, zP])
    # P.shape = (3, NP)

    # Vecteur déplacement élémentaire le long de la distribution
    vec_d1P = P[:,1:] - P[:,:-1]
    # vec_d1P.shape = (3, NP-1)

    # Localisation du point courant de la distribution
    mid_d1P = (P[:,1:] + P[:,:-1]) / float(2)
    # mid_d1P.shape = (3, NP-1)
```

76 N spires circulaires.py.py - C:\Doc Emna\tipe implantation cardiaque\N spires circulaires.py.py

File Edit Format Run Options Windows Help

```
XPM = XM[:, :, np.newaxis] - mid_d1P[0, :]
YPM = YM[:, :, np.newaxis] - mid_d1P[1, :]
ZPM = ZM[:, :, np.newaxis] - mid_d1P[2, :]
# XPM.shape = YPM.shape = ZPM.shape = (NX, NY, NP)

PMcubed = (XPM**2 + YPM**2 + ZPM**2)**(3/2.)
# PMcubed.shape = (NX, NY, NP)

# Éviter la division par zéro (trop proche des sources)
PMcubed[PMcubed < 1e-6] = np.nan

Xdl_cross_PM = - YPM * vec_d1P[2, :] + ZPM * vec_d1P[1, :]
Ydl_cross_PM = - ZPM * vec_d1P[0, :] + XPM * vec_d1P[2, :]
Zdl_cross_PM = - XPM * vec_d1P[1, :] + YPM * vec_d1P[0, :]
# Xdl_cross_PM.shape = Ydl_cross_PM.shape = Zdl_cross_PM.shape =
# (NX, NY, NP)

Xdl_cross_PM_over_PMcubed = Xdl_cross_PM / PMcubed
Ydl_cross_PM_over_PMcubed = Ydl_cross_PM / PMcubed
Zdl_cross_PM_over_PMcubed = Zdl_cross_PM / PMcubed

BX = np.sum(Xdl_cross_PM_over_PMcubed, axis = 2)
BY = np.sum(Ydl_cross_PM_over_PMcubed, axis = 2)
BZ = np.sum(Zdl_cross_PM_over_PMcubed, axis = 2)
# BX.shape = BY.shape = BZ.shape = (NX, NY)

BNorme = (BX**2 + BY**2 + BZ**2)**(.5)
# BNorme.shape = (NX, NY)

return mu0 * I * BX/(4 * np.pi), mu0 * I * BY/(4 * np.pi),
       mu0 * I * BNorme/(4 * np.pi)

def Bzloop(z):
    return mu0 * I * R**2/(2 * (R**2 + z**2)**(3/2.))

import matplotlib.pyplot as plt
import matplotlib.cm as cm

R = [float(input('donner le rayon de la spire désiré'))]
xP, yP, zP = circular_loop(R, [0., 0., 0.])
```

76 \*N spires circulaires.py.py - C:\Doc Emna\tipe implantation cardiaque\N spires circulaires.py.py\*

```
File Edit Format Run Options Windows Help

# Grille de NX*NY points
NX = 30
NY = 30

# Coordonnées min et max des points de la grille
xmax = 10
xmin = -xmax
ymax = 10
ymin = -ymax

# Les coordonnées des points de la grille sont répartis uniformément
# sur [xmin, xmax]x[ymin, ymax]
xM = np.linspace(xmin, xmax, NX)
yM = np.linspace(ymin, ymax, NY)

# Création de la grille
XM, YM = np.meshgrid(xM, yM)

# On impose la cote de la grille : z = 0
ZM = np.zeros((yM.size, xM.size))

# Calcul du champ magnétique
# La norme de B (BNorme) permet de normer le champ
BX, BY, BNorme = biot_savart(xP, yP, zP, XM, YM, ZM)

L=int(input("donner le nombre de spires désirées"))
ZP=[i*0.1 for i in range(1, L//2)]
ZN=[i*(-0.1) for i in range (1,L//2)]
zspires=ZP+ZN

for z in zspires:
    xP, yP, zP = circular_loop(R, [0., 0., z])
    BX2, BY2, BNorme2 = biot_savart(xP, yP, zP, XM, YM, ZM)

    BX += BX2
    BY += BY2
    BNorme += BNorme2

plt.quiver(XM, YM, BX, BY, pivot = 'middle', units = 'width')
plt.imshow(BNorme, interpolation = 'bilinear', origin = 'lower',
           cmap = cm.jet, extent = (xmin, xmax, ymin, ymax))
plt.show()
```

# ANNEXE 4: code PYTHON N spires polygonales 3D

76 \*N spires polygone régulier.py - C:\Doc Emna\tipe implantation cardiaque\N spires polygone r

File Edit Format Run Options Windows Help

```
import numpy as np

# mu0 = 12.566370614e-7
mu0 = 4 * np.pi
PII=np.pi
I = 1.

M=int(input('donnez le nombre de cotes désiré pour votre spire'))

C=[0.,0.,0.]
N=200
def polygone_loop (RAY, C):
    XP = []
    ZP = []
    YP = []
    Xs = []
    Zs = []
    Ys = []
    for k in range (M):
        Xs.append(np.cos(2*k*PII/M)*RAY-C[0])
        Zs.append(np.sin(2*k*PII/M)*RAY-C[1])
    for i in range(M-1):
        X=np.linspace(Xs[i],Xs[i+1],N)
        XL=list(X)
        XP=XP+XL
        Z=np.linspace(Zs[i],Zs[i+1],N)
        ZL=list(Z)
        ZP=ZP+ZL

    X=np.linspace(Xs[M-1],Xs[0],N)
    XL=list(X)
    XP=XP+XL
    YP=[C[2]]*(N*M)
    Z=np.linspace(Zs[M-1],Zs[0],N)
    ZL=list(Z)
    ZP=ZP+ZL
    return XP, YP, ZP
```

76 \*N spires polygone régulier.py - C:\Doc Emna\tipe implantation cardiaque\N spires polygone régulier.py\*

File Edit Format Run Options Windows Help

```
def biot_savart(xP, yP, zP, XM, YM, ZM):
    """
    xP, yP et zP sont des ndarrays à une dimension contenant
    les NP coordonnées des points P représentant la distribution
    de courant.

    XM, YM sont des ndarrays à deux dimensions obtenus typiquement
    par
    XM, YM = np.meshgrid(np.linspace(xmin, xmax, NX), np.linspace(ymin, ymax, NY))

    ZM est un ndarray à deux dimensions représentant la cote des points de la
    grille XM, YM (typiquement rempli de 0 : observation dans le plan z = 0).

    La fonction renvoie deux ndarrays de dimension (NX, NY) qui sont les
    composantes (utiles) du champ dans le plan d'observation.
    """

    P = np.array([xP, yP, zP])
    # P.shape = (3, NP)

    # Vecteur déplacement élémentaire le long de la distribution
    vec_d1P = P[:,1:] - P[:,:-1]
    # vec_d1P.shape = (3, NP-1)

    # Localisation du point courant de la distribution
    mid_d1P = (P[:,1:] + P[:,:-1]) / float(2)
    # mid_d1P.shape = (3, NP-1)

    XPM = XM[:, :, np.newaxis] - mid_d1P[0, :]
    YPM = YM[:, :, np.newaxis] - mid_d1P[1, :]
    ZPM = ZM[:, :, np.newaxis] - mid_d1P[2, :]
    # XPM.shape = YPM.shape = ZPM.shape = (NX, NY, NP)

    PMcubed = (XPM**2 + YPM**2 + ZPM**2)**(3/2.)
    # PMcubed.shape = (NX, NY, NP)

    # Éviter la division par zéro (trop proche des sources)
    PMcubed[PMcubed < 1e-6] = np.nan

    Xdl_cross_PM = - YPM * vec_d1P[2, :] + ZPM * vec_d1P[1, :]
    Ydl_cross_PM = - ZPM * vec_d1P[0, :] + XPM * vec_d1P[2, :]
```

```

76 *N spires polygone régulier.py - C:\Doc Emna\tipe implantation cardiaque\N spires polygone régulier
File Edit Format Run Options Windows Help
Xdl_cross_PM = - YPM * vec_dlp[2,:] + ZPM * vec_dlp[1,:]
Ydl_cross_PM = - ZPM * vec_dlp[0,:] + XPM * vec_dlp[2,:]
Zdl_cross_PM = - XPM * vec_dlp[1,:] + YPM * vec_dlp[0,:]
# Xdl_cross_PM.shape = Ydl_cross_PM.shape = Zdl_cross_PM.shape =
# (NX, NY, NP)

Xdl_cross_PM_over_PMcubed = Xdl_cross_PM / PMcubed
Ydl_cross_PM_over_PMcubed = Ydl_cross_PM / PMcubed
Zdl_cross_PM_over_PMcubed = Zdl_cross_PM / PMcubed

BX = np.sum(Xdl_cross_PM_over_PMcubed, axis = 2)
BY = np.sum(Ydl_cross_PM_over_PMcubed, axis = 2)
BZ = np.sum(Zdl_cross_PM_over_PMcubed, axis = 2)
# BX.shape = BY.shape = BZ.shape = (NX, NY)

BNorme = (BX**2 + BY**2 + BZ**2)**(.5)
# BNorme.shape = (NX, NY)

return mu0 * I * BX/(4 * np.pi), mu0 * I * BY/(4 * np.pi), \
mu0 * I * BNorme/(4 * np.pi)

def Bzloop(z):
    return mu0 * I * R**2/(2 * (R**2 + z**2)**(3/2.))

import matplotlib.pyplot as plt
import matplotlib.cm as cm

R=float(input('donner le rayon désiré du polygone='))

xP, yP, zP = polygone_loop(R, [0., 0., 0.])

# Grille de NX*NY points
NX = 30
NY = 30

L=int(input("donner le nombre de spires désirées"))

# Coordonnées min et max des points de la grille
# l'induction doit être sur un minimum de 7cm
xmax = 10
xmin = -xmax

```

```

ymax = 10
ymin = -ymax

# Les coordonnées des points de la grille sont répartis uniformément
# sur [xmin, xmax]x[ymin, ymax]
xM = np.linspace(xmin, xmax, NX)
yM = np.linspace(ymin, ymax, NY)

# Création de la grille
XM, YM = np.meshgrid(xM, yM)

# On impose la cote de la grille : z = 0
ZM = np.zeros((yM.size, xM.size))

BX, BY, BNorme = biot_savart(xP, yP, zP, XM, YM, ZM)

#on suppose que le fil est de diamètre 1mm

ZP=[i*0.1 for i in range(1, L//2)]
ZN=[i*(-0.1) for i in range (1,L//2)]
zspires=ZP+ZN

for z in zspires:
    xP, yP, zP = polygone_loop(R, [0., 0., z])
    BX2, BY2, BNorme2 = biot_savart(xP, yP, zP, XM, YM, ZM)

    BX += BX2
    BY += BY2
    BNorme += BNorme2

plt.quiver(XM, YM, BX, BY, pivot = 'middle', units = 'width')
plt.imshow(BNorme, interpolation = 'bilinear', origin = 'lower',
           cmap = cm.jet, extent = (xmin, xmax, ymin, ymax))
plt.show()

```