```
In [1] : %matplotlib inline
        import matplotlib as mpl
        import matplotlib.pyplot as plt
```

```
In [2] : import matplotlib.colors as colors
        import matplotlib.animation as animation
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('png', 'pdf')
        plt.rcParams['animation.ffmpeg_path'] = r'C :\ffmpeg\bin\ffmpeg.exe'
```

```
In [3] : import numpy as np
        import scipy.special as spec
        import scipy.integrate as inte
        import sympy as sp
        sp.init_printing()
```

1 Position du problème

Étant donné un fil électrique très long parcouru par un courant électrique i, on cherche à déterminer le champ magnétique créé par phénomène d'induction dans l'espace autour. On peut le détecter en approchant une boussole, de la limaille de fer ou d'autres aimants permanents du fil, voire approcher un autre fil électrique lui aussi parcouru par un courant.

2 Construction du champ magnétique

Les cellules suivantes servent à construire la fonction B correspondant à l'intensité du champ magnétique $\mathbf{B}$ à une distance r du fil, à l'instant t.

```
In [4] : r = sp.symbols('r', positive = True)
        t, omega, phi = sp.symbols('t omega phi')

        c = 3e8       # Célérité de la lumière dans le vide
        k = omega/c   # Nombre d'onde

        # Composante du champ magnétique associée à la pulsation omega
        B_component = (sp.besselj(0,k*r)-sp.bessely(0,k*r) ) \
            *sp.cos(omega*t-phi)

        # Création du champ (symbolique et fonction) en sommant les composantes
        def create_Bfield(puls,phas):
            spectr = zip(puls,phas)

            B_field = sum([B_component.subs({omega:om, phi:ph}) for (om,ph) in spectr])
            B_function = sp.lambdify((r, t), B_field,
                modules=['numpy',{"besselj":spec.jn, "bessely":spec.yn}])

            return B_field, B_function
```

In [20] :
```python
def graphe_B(times, ani = False):
    '''
    Construit les graphes du champ magnétique B aux temps donnés dans la liste
    "times"
    Si le drapeau 'ani' est True, alors entrer en mode "animation"
    '''
    radii = np.linspace(rmin, rmax, 10*rmax)

    fig = plt.figure(1)
    ax = plt.axes()

    def legende(ti):
        out = r'$t= {:.3e}$'.format(ti)
        out = out + r"$\ \mathrm{s}$"
        return out

    if not(ani):
        if hasattr(times, '__iter__'):
            for ti in times:
                champ = B_function(radii, ti)
                ax.plot(radii, champ, label=legende(ti))
        else:
            champ = B_function(radii, ti)
            ax.plot(radii, champ, label=legende(ti))
        ax.legend()
    else:
        line, = ax.plot([], [], lw=2)
        time_text = ax.text(0.02, 0.95, '', transform=ax.transAxes)


        t0, t1= times
        interval = t1 - t0
        animtime = 10
        dt = interval/animtime # secondes vidéo par seconde réelle
        framenum = int(np.ceil(30*animtime))

        ymax = B_function(radii,t0).max()

        ax.set_xlim(0,rmax)
        ax.set_ylim((-ymax/2,ymax))

        def init():
            line.set_data([],[])
            time_text.set_text('')
            return line, time_text

        def animate(i):
            ti = dt*i+t0
```

```
            legende_temps = legende(ti)
            champ = B_function(radii, ti)
            line.set_data(radii, champ)
            time_text.set_text(legende_temps)
            return line, time_text

    ax.grid(True)
    ax.set_xlabel("Distance $r$ (m)")
    ax.set_ylabel("Valeur du champ (T)")
    ax.set_title(r'Champ magnétique ' + r'$\mathbf{B}$' \
                 + ' créé par un courant variable')

    fig.tight_layout()

    if ani:
        anima = animation.FuncAnimation(fig, animate, init_func=init,
                    frames = framenum, interval = interval, blit = True)
        mywriter = animation.FFMpegWriter(fps=30)
        anima.save('champ_mag_anim.mp4', writer=mywriter)
    else:
        fig.savefig('profil_champmag.pdf')
        return fig, ax
```

```
In [6] : def build_field(t, colmap='gist_heat'):
             """
             Portrait du champ magnétique à l'instant t
             """
             wind = rmax

             def field_func(x,y):
                 r = np.sqrt(x*x+y*y)
                 Btheta = B_function(r, t)
                 direct = np.array([-y/r, x/r])
                 return Btheta*direct

             Y, X = np.ogrid[-wind:wind:wind*10j, -wind:wind:wind*10j]
             BX, BY = field_func(X, Y)
             intensity = np.sqrt(BX**2+BY**2)


             fig, ax = plt.subplots(1, 1, figsize=(8,8))

             heat = ax.imshow(intensity,
                              cmap=colmap,
                              norm=colors.LogNorm(),
                              extent=[-wind, wind, -wind, wind],
                              alpha=0.6)
             cbar = fig.colorbar(heat,
```

```
                    label='Intensité du champ (T)')

    strm = ax.streamplot(X,Y, BX, BY,
        arrowstyle='->',
        color='w',
        linewidth=0.8,
        arrowsize=2,
        density=1.4,
        )

    ax.grid(False)
    ax.set_aspect('equal')
    ax.set_xlim((-wind,wind))
    title_text = r'Champ magnétique $\mathbf{B}$ à '
    title_text += r"$t={:g}$".format(t)
    title_text += r" $\mathrm{s}$"
    ax.set_title(title_text)
    fig.tight_layout()

    return fig
```

3 Tracés

3.1 Données initiales

Entrez dans la variable `freqs` les fréquences du courant voulu, et dans `phas` les phases associées, et exécutez la cellule (`Ctrl + Entrée` sur le clavier) pour définir la fonction de champ :

```
In [7] : freqs = [n*1e7 for n in range(5,7)] + \
            [n*1e4 for n in range(2,5)]
        puls = 2*np.pi*np.asarray(freqs) # Pulsations associées aux fréquences

        phases = [0,0,0.7,0.8,0.3]

        B_field, B_function = create_Bfield(puls, phases)
        B_function
```

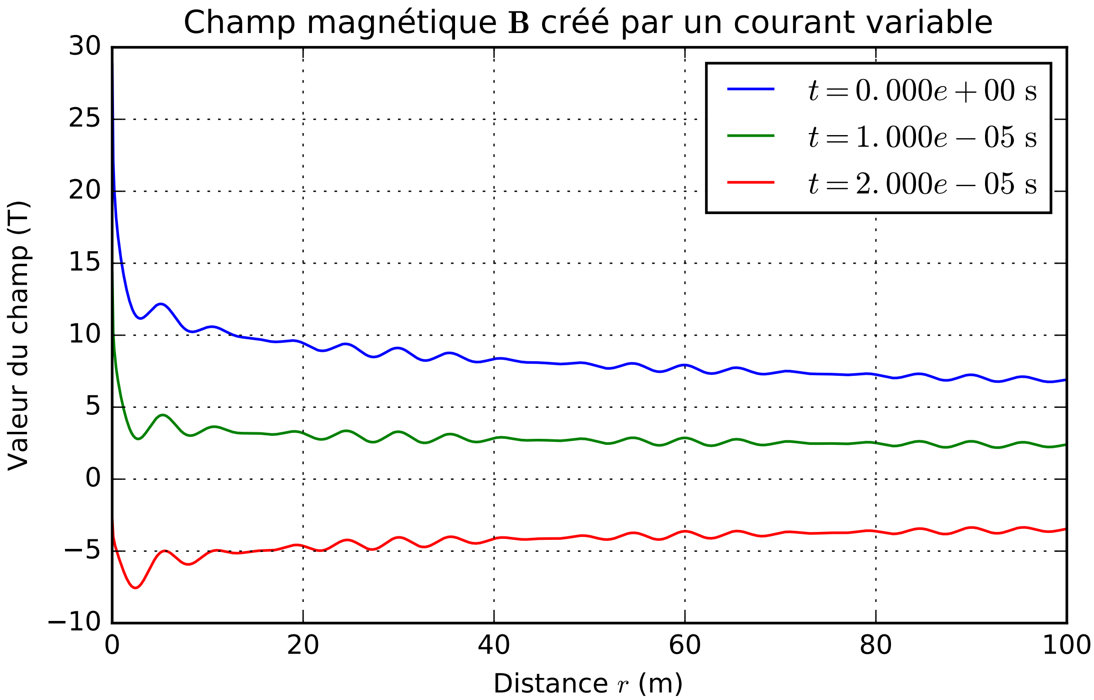
```
Out[7] : <function numpy.<lambda>>
```

La cellule suivante définit les distances minimale et maximale pour lesquels tracer le profil du champ magnétique :

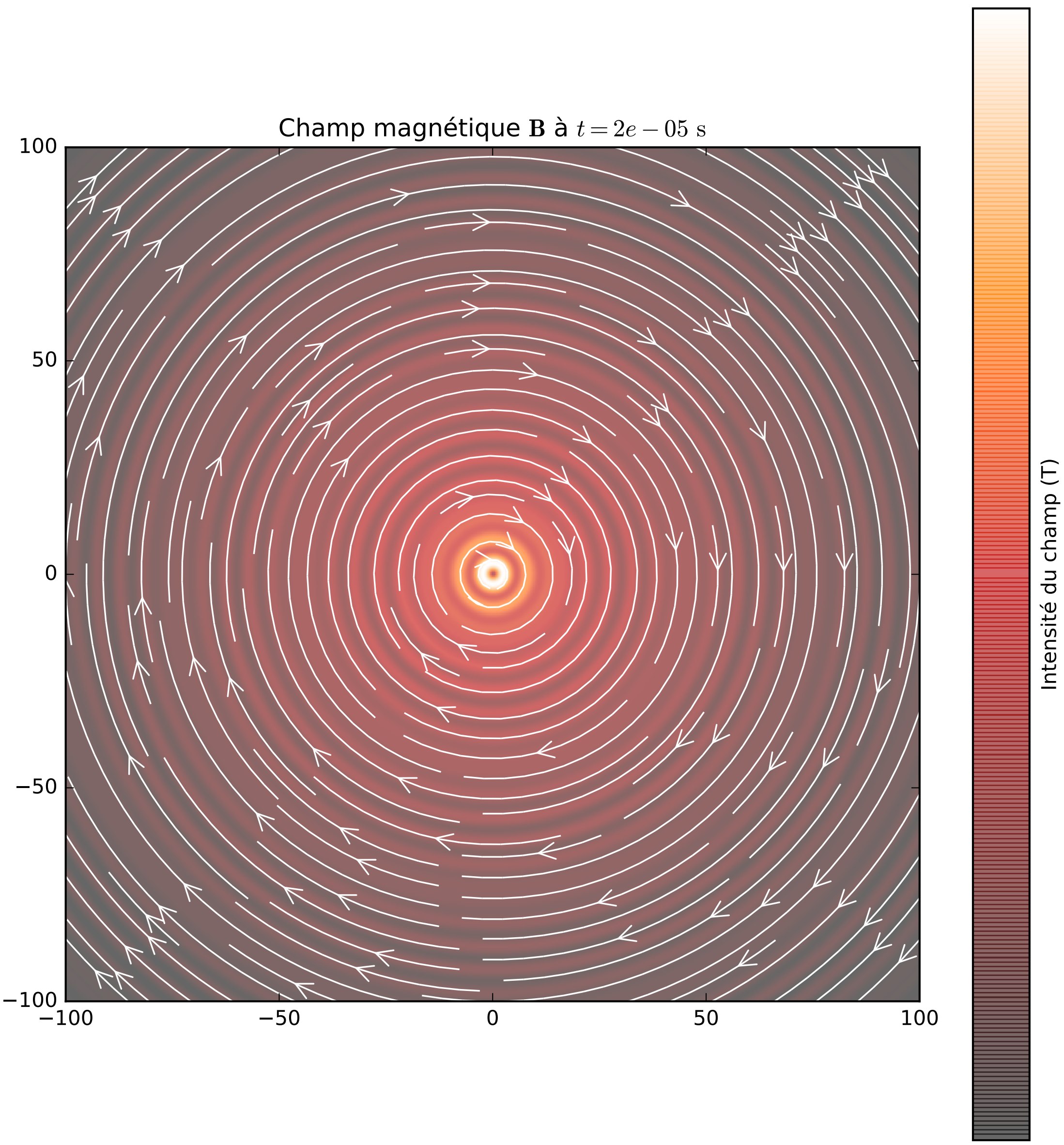
```
In [8] : rmin = 0.01
        rmax = 100
```

```
In [15] : times = [1e-6*k for k in [0, 10, 20]]

         graphe_B(times)
```

```
Out[15] : (<matplotlib.figure.Figure at 0x281af5340b8>,
 <matplotlib.axes._subplots.AxesSubplot at 0x281b20d7ef0>)
```

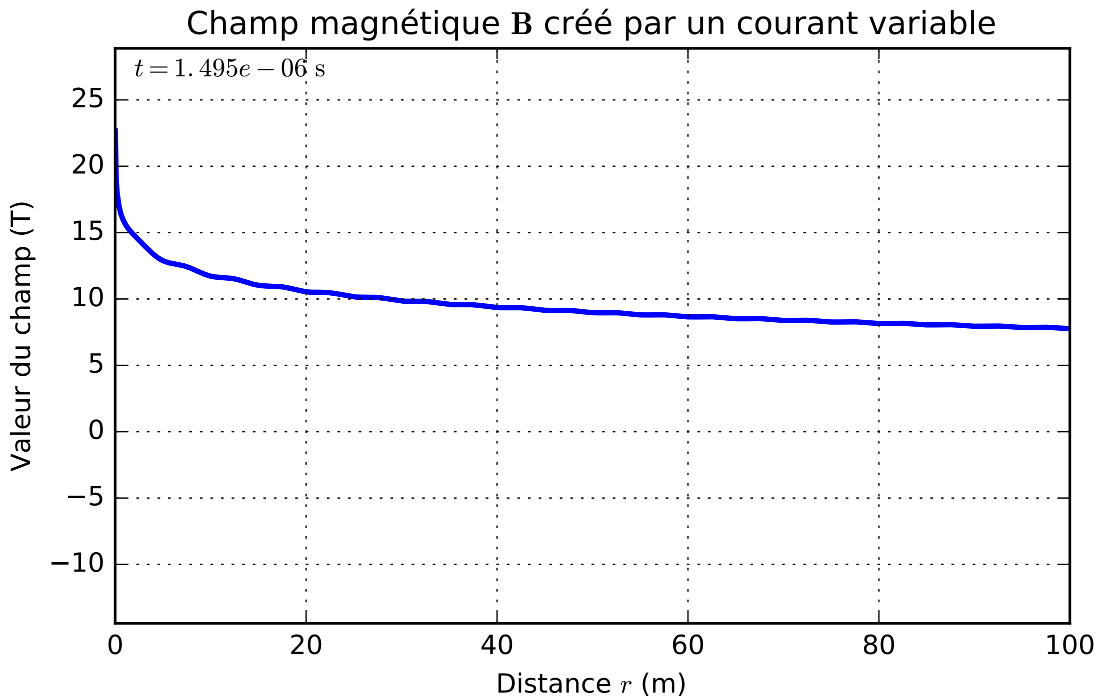



```
In [10] : t = times[2]     # Temps auquel calculer le portrait du champ (pas de liste)

          output_field = build_field(t)
          output_field.savefig('portrait_champmag.pdf')
```



3.2 Animations

```
In [23] : times = [0, 5e-8]

          graphe_B(times, True)
```



4 Théorie (Bac + 1,5)

Le champ magnétique $\mathbf{B}$ dérive d'un champ $\mathbf{A}$ appelé *potentiel vecteur* : $\mathbf{B} = \nabla \wedge \mathbf{A}$. Par symétrie cylindrique, on a $\mathbf{B}(\mathbf{r}, t) = B(r, t)\mathbf{e}_\theta$. Par suite $\mathbf{A}(\mathbf{r}, t) = A(r, t)\mathbf{e}_z$.

Le potentiel vecteur $\mathbf{A} = A(r, t)\mathbf{e}_z$ est solution de l'équation d'onde

$$\Delta \mathbf{A} - \frac{1}{c^2}\frac{\partial^2 \mathbf{A}}{\partial t^2} = -\mu_0 \mathbf{J}(r, t), \tag{1}$$

avec $\mathbf{J}(r, t) = \dfrac{i(t)\delta(r)}{2\pi r}\mathbf{e}_\theta$ la densité volumique de courant.

Pour un courant sinusoïdal $i(t) = I\exp(i\omega t)$, le potentiel s'écrit $A(r, t) = f(r)\exp(i\omega t)$ et l'équation aux dérivées partielles se réduit à

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}f}{\mathrm{d}r}\right) + k^2 f(r) = -\frac{\mu_0 I \delta(r)}{2\pi r}, \tag{2}$$

avec $k = \dfrac{\omega}{c}$.

La solution générale prend la forme

$$f(r) = CJ_0(kr) + DY_0(kr)$$

où C et D dépendent de la pulsation ω du courant, et J_0, Y_0 sont les 0-ièmes fonctions de Bessel de la première et seconde espèce, solutions de

$$xy''(x) + y'(x) + xy(x) = 0.$$