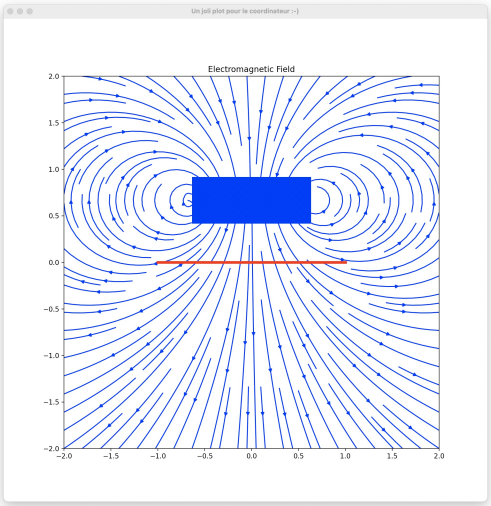
Python 21-22 for dummies : problème 7
Un petit aimant qui danse près d'une bobine !
Une étrange histoire magnétique !



Nous allons créer un modèle numérique simple pour la force exercée par une bobine dans laquelle on fait circuler un courant I sur un aimant cylindrique placé dans l'axe d'une bobine.

L'aimant est un cylindre de masse $m = 0.1\ kg$, de rayon $R = 0.625$, d'épaisseur $e = 0.5$ et se situe au dessus ou en dessous d'une bobine de rayon $R_{coil} = 1$ qui est placée à une hauteur $h = 2$ par rapport au sol. Toutes les données géométriques sont en centimètres. Les axes de révolution de l'aimant et de la bobine coincïdent. On définit l'origine du système d'axe au centre de la bobine. La bobine et l'aimant sont placés horizontalement avec un vecteur normal $[0, 0, 1]$. Il y a 200 spires dans la bobine.

Le moment magnétique de l'aimant est défini par :

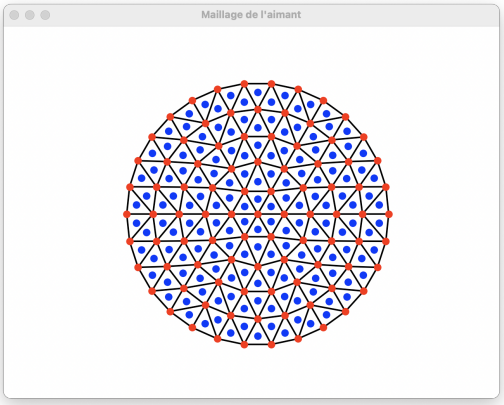
$$\boldsymbol{\mu} = [0, 0, -\mu]$$

La valeur de μ est déduite de la magnétisation résiduelle de Néodyme de bore, de la perméabilité du vide et du volume de l'aimant, comme mon ami Claude vous l'a expliqué.

Et comme votre enseigant favori de physique vous l'a également appris, le champ magnétique $\mathbf{B}$ généré par un dipole à l'origine à un endroit $\mathbf{r}$ de l'espace est défini par la relation suivante.

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi r^3}\left[3(\boldsymbol{\mu}\cdot\widehat{\mathbf{r}})\widehat{\mathbf{r}} - \boldsymbol{\mu}\right]$$

Pour obtenir le champ magnétique généré par l'aimant, nous allons décomposer celui en certains nombres de triangles ne se recouvrant pas. Il s'agit de créer un maillage de n triangles qui formera une partition du disque de l'aimant en une série de petits triangles Ω_e : ce sont des éléments finis :-) Les points rouges sont les sommets et les points bleus sont les centres de gravité de chaque triangle.



Le champ magnétique de l'aimant sera obtenu en superposant tous les champs magnétiques de dipoles élémentaires μ_e que l'on va placer au centre de gravité de chaque triangle Ω_e.

Comment obtenir le moment magnétique de ces dipoles ? Nous allons procéder de manière très simple en distribuant le moment magnétique de l'aimant proportionnellement à la surface de chaque triangle.

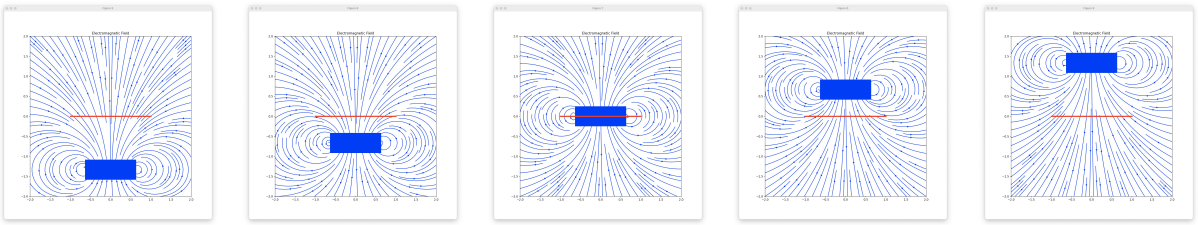
On obtiendra donc évidemment $\mu = \sum_{e=1}^{n} \mu_e$.

C'est de cette manière que nous avons représenté le champs magnétique avec `python`. Enfin, notre implémentation est presque correcte, on s'est contenté d'écrire $\mu_e = \mu$, mais cela n'a quasiment aucun impact sur les lignes de champ magnétique.

Lorsqu'un courant I circule dans la bobine, le champs magnétique de l'aimant va induire une force -la force de Lorentz- qui s'applique sur la bobine et par réaction sur l'aimant. Supposons que la bobine est fixe et que c'est l'aimant qui peut se déplacer verticalement sous l'effet de la force de Lorentz. Si on lâche l'aimant au dessus de la bobine avec le courant choisi adéquatement, la force de Lorentz va s'opposer à la gravité qui fait chuter l'aimant. En raison de la symétrie du problème, les deux forces n'ont qu'une composante verticale et donc l'aimant ne se déplacera donc que dans la direction verticale...

Le champs magnétique et la force de Lorentz dépendent évidemment de la position de l'aimant...

Nous allons illustrer ci-dessous le champs magnétique dans le plan xz, loesque l'aimant se déplace d'une position $X_{start} = -2.0$ vers une position $X_{stop} = 2.0$. On y observe les lignes du champ magnétique $\mathbf{B}$ créé par la présence de l'aimant. A tout instant t, il s'agira de calculer la force de Lorentz pour le champs magnétique qui correspond à la position $z(t)$ de l'aimant !



Le premier objectif de ce devoir sera de calculer la force de Lorentz qui s'applique sur la bobine en présence d'un champs magnétique avec l'expression suivante :

$$\mathbf{F} = nI \int_{\Gamma} d\mathbf{l} \times \mathbf{B}$$

$$\downarrow$$

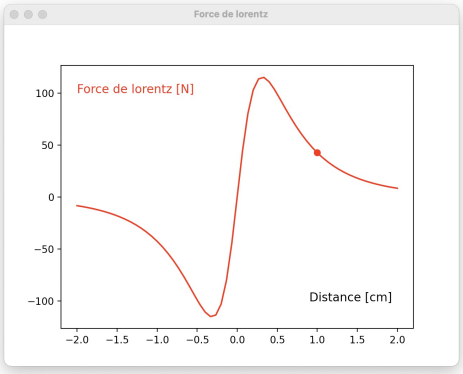
Et comme le problème a une symétrie axiale :-)

$$= nI \; 2\pi R_{coil} \; B_x(R_{coil}, 0, 0)$$

où I est courant qui passe dans le bobine, R_{coil} est le rayon de la bobine, B_x la composante en x (ou plus exactement radiale) du champs magnétique et n le nombre de spires.

Plus précisément, il s'agira de calculer la force de Lorentz pour différentes positions de l'aimant et un courant unitaire $I = 1$.

Le résultat que vous devriez obtenir devrait ressemble à ce qui est illustré à côté. Du moins, si Nicolas et moi ne nous sommes pas trompés:-)



Prédire le mouvement de l'aimant...

Mais, cela n'est pas terminé, nous souhaitons prédire le mouvement de l'aimant en utilisant les équations de Newton. A la force exercée par l'aimant sur la bobine, correspond exactement la force opposée de la bobine sur l'aimant par le célèbre principe de l"action et de la réaction ! On peut donc écrire le système de deux équations différentielles d'ordre un comme suit :

$$\begin{aligned} z'(t) &= v(t) \\ mv'(t) &= -mg + F(z(t)) \end{aligned}$$

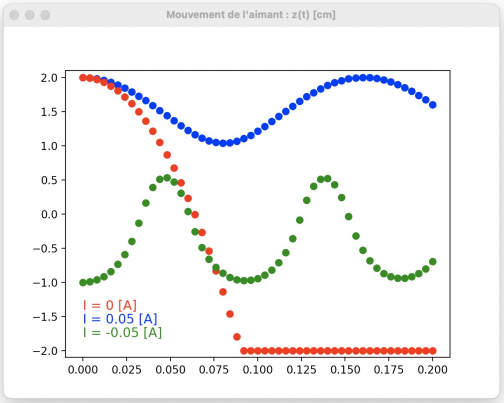
$$\begin{aligned} z(0) &= z_0 \\ v'(0) &= 0 \end{aligned}$$

Ce problème de Canchy pourra être résolu en utilisant la méthode de Runge Kutta d'ordre quatre.

En considérant deux positions différentes et diverses valeurs de courant, nous avons réussi à faire osciller notre aimant pour la plus grande joie de notre enseignant de physique. Eh oui, ce sont deux forces conservatives, et donc un comportement oscillant n'est pas surprenant !

Attention, si vous utilisez un pas de temps trop grossier, le comportement prédit ne sera plus fiable !

Vous devriez ainsi obtenir les trois courbes de la position en fonction du temps $z(t)$ représentées sur la figure. A nouveau si Nicolas et moi ne nous sommes pas trompés:-)



La courbe rouge représente une chute libre jusqu'au sol situé en $z = -2$ où on supposera un choc parfaitement inélastique (sans aucun rebond !). La courbe bleue correspond à un aimant laché au dessus de la bobine où la force de Lorentz s'oppose à la gravité et permet de le maintenir en lévitation au dessus de la bobine. Le dernière courbe est obtenue en inversant le courant dans la bobine et en faisant monter la bobine avec la force de Lorentz : dans cette seconde configuration, l'aimant va osciller en traversant la bobine !

Plus précisément, on vous demande de :

1. Tout d'abord, écrire une fonction

   ```
   F = lorentzComputeForce(Xmagnet,Ymagnet,Zmagnet,Rcoil,triangles,Z,mu0,mu,nSpires)
   ```

   qui calcule la force de Lorentz induite sur la bobine par l'aimant pour une série de positions de l'aimant défini par le vecteur `Z` de taille `m`. Les autres arguments sont les coordonnées en x, et y des sommets des maillages de l'aimant : `Xmagnet`, `Ymagnet`. Il s'agit de tableaux qui ont une taille `nNode` égale au nombre de sommets du maillage. On fournit également le tableau d'appartenance commun du maillage contenant les indices des 3 sommets de chaque triangle du maillage : `triangles`. C'est un tableau de taille `nElem`×`3`. Enfin, il y a les deux paramètres matériels `mun` `mu0`, la hauteur de l'aimant par rapport à la bobine `Zmagnet` et le nombre de spires `nSpires`.

   Il faut ensuite renvoyer un tableau de même taille que `Z`.

2. Ensuit, founir une fonction

   ```
   f = lorentzInterpolate(z,Z,F))
   ```

qui effectue une interpolation linéaire par morceau afin d'obtenir, pour une hauteur de l'aimant `z`, une valeur intermédaire `f` de la force de Lorentz à partir de la table `Z,F` obtenue précédemment. Attention à bien traiter les deux cas limites si la valeur de `z` correspond à une extrapolation. L'idée est d'éviter de recalculer systématiquement la force de Lorentz pour de très nombreuses itérations de Runge-Kutta et d'utiliser une table calculée dans une première étape avec un nombre limité mais raisonnable de valeurs de notre force.

3. Finalement, reproduire une fonction qui a été vue en séance d'exercices...

   ```
   T,U = lorentzRungeKutta(Tstart,Tend,Ustart,n,f)
   ```

   qui effectue l'intégration par la méthode de Runge-Kutta d'ordre quatre des équations de Newton fournie dans l'énoncé et dans le script de test. Il faudra prévoir que si la position calculée est égale ou inférieure à l'altitude du sol $z = -2$, alors le mouvement se fige avec une position qui reste à cette valeur et une vitesse nulle. Le temps initial et final sont donnés par `Tstart` et `Tend`, tandis que `n` est le nombre de pas et `f` est le pointeur de la fonction pour le calcul de la dérivée des inconnues. Les conditions initiales sont dans le vecteur `Ustart`. La fonction ne doit qu'intégrer que des systèmes à deux équations et doit renvoyer un tableau avec les temps intermédiaires `T` de taille $(n+1)$ et les valeurs de la position et de la vitesse de l'aimant `U` de taile $(n+1) \times 2$.

4. Comme d'habitude, pour tester votre programme, on vous a fourni un tout petit programme simple `lorentzTest.py` pour tester votre programme. On y construit d'abord le maillage de triangles nécessaire pour notre intégration numérique et pour la superposition des champs magnétiques de tous les dipoles composant notre aimant. Et ensuite, on fait appel à votre fonction !

   Le programme fait aussi plein de jolis dessins et d'animation qui seront très utiles pour faire un super joli rapport pour votre projet :-)

```
  from numpy import *

  mu0     = 4e-7*pi*1e-2     # permeabilité du vide en [H/cm]
  mMagnet = 0.1              # masse de l'aimant [kg]
  Rmagnet = 0.635            # rayon de l'aimant [cm]
  Hmagnet = 0.5              # épaisseur de l'aimant [cm]
  Zmagnet = 0.0              # position verticale de l'aimant en [cm]
  Br      = 1.4              # magnetisation residuelle du Néodyme fer bore (NdFeB) en [T] ou [kg/(A s)]
  mu      = Rmagnet**2*Hmagnet*pi*Br / mu0
                             # moment magnétique de l'aimant [A cm2]
  Rcoil   = 1                # rayon de la bobine [cm]
  nSpires = 200              # nombre de spires

#
# -1- Construction d'un maillage de triangles pour un cercle de rayon unitaire
#

  from scipy.spatial import Delaunay
  nR      = 6
  nTheta  = 5
  nNode   = 1 + sum(arange(1,nR))*nTheta
  R     = zeros(nNode)
  Theta = zeros(nNode)
  index = 1; dR = 1.0/(nR-1)
  for i in range(1,nR):
      dTheta = 2*pi/(i*nTheta)
      for j in range(0,i*nTheta):
          R[index]     = i*dR
          Theta[index] = j*dTheta; index += 1
  X       = R*cos(Theta)
  Y       = R*sin(Theta)
  triangles = Delaunay(stack((X,Y),1)).simplices
  nElem = len(triangles)
  print(" Number of triangles : %d " % nElem)
  print(" Number of nodes     : %d " % nNode)

#
```

```
# -2- Calcul de la force de lorentz pour diverses hauteurs
#

  m       = 61
  Zstart  = -2                        # [cm]
  Zstop   =  2                        # [cm]
  Zshift  = linspace(Zstart,Zstop,m)
  Tstart  = 0                         # [s]
  Tstop   = 0.5                       # [s]
  T,delta = linspace(Tstart,Tstop,m,retstep=True)

  Xmagnet = Rmagnet*R*cos(Theta)
  Ymagnet = Rmagnet*R*sin(Theta)
  Florentz = lorentzComputeForce(Xmagnet,Ymagnet,Zmagnet,Rcoil,triangles,Zshift,mu0,mu,nSpires)

#
# -3- Calcul du mouvement de l'aimant
#

  def f(u):
    Fcoil = lorentzInterpolate(u[0],Zshift,Florentz)
    dzdt = u[1]
    dvdt = (100/mMagnet) * (I*Fcoil - 9.81 * mMagnet)
    return array([dzdt,dvdt])

  Tstart = 0; Tend = 0.2; n = 50
  Ustart = [2.0,0.0]
  I = 0.05; T,UwithMagnet = lorentzRungeKutta(Tstart,Tend,Ustart,n,f)
  I = 0; T,UwithoutMagnet = lorentzRungeKutta(Tstart,Tend,Ustart,n,f)
```

5. Ce devoir n'est pas particulièrement compliqué.
   Mais il faut essayer d'implémenter ce calcul de manière efficace et surtout d'avoir la bonne réponse !

6. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera soumise sur `zouLab` sans y adjoindre le programme de test fourni ! Cette fonction devra être soumise via le web : ce travail est individuel et sera évalué. Pour rappel, toutes vos soumissions seront systématiquement analysées par un logiciel anti-plagiat. Faites vraiment votre programme seul... :-)

7. A la fin du devoir, vous pourrez adapter les paramètre pour votre prototype et introduire la possibilité de contrôle le courant en fonction de la position de l'aimant : mais cela c'est une autre histoire qui fera l'objet d'un autre épisode ! Et oui, la modélisation physique et les méthodes numériques sont les clés qui permettent de comprendre ce qu'on observe dans le laboratoire !