— Votre compte-rendu devra être déposé sur Moodle (cours "Physique numérique") au plus tard le 8 février 2021.
https://moodlesciences.univ-brest.fr/moodle/course/view.php?id=367
— Regrouper l'ensemble de vos documents (compte-rendu, graphes, scripts python (fichiers `.py`) dans une archive (format `.zip` par exemple) pour ne déposer **qu'un seul** fichier.
Votre compte rendu devra être un fichier au format `.pdf` (les formats `doc` ou `docx` ne sont pas autorisés) dans lequel vous répondrez aux différentes questions en y incluant vos résultats (données numériques, graphes, ...) et commentaires. N'oubliez pas de commenter largement vos programme en précisant aussi les n° complets des exercices auxquels ils se rapportent.
— Avant de quitter la salle, n'oubliez pas de sauvegarder votre travail sur une clé USB ou sur l'espace de stockage de votre ENT.

— On rappelle que pour commenter une ligne il suffit que celle-ci commence par `#`.
Pour commenter un bloc de lignes, on peut utiliser 3 guillemets `"` et écrire :
```
"""
bloc de commentaire
"""
```
En réalité, il ne s'agit pas d'un commentaire mais d'un texte du programme s'étendant sur plusieurs lignes. En conséquence, il faut respecter l'indentation de ce bloc de code.
— Dans votre fichier script, commencer par importer `numpy` et `matplotlib`. On définit également `eps` qui correspond au « zéro machine ».
```
import numpy as np
import matplotlib.pyplot as plt
eps=np.spacing(1)
```
— Pour modifier la fonte par défaut et obtenir un meilleur rendu des commandes LaTeX, utiliser :
```
plt.rc('text', usetex=True)
plt.rc('font', family='serif')
```
— En plaçant le caractère `r` devant une chaine de caractères, le backslash,`\`, est interprété tel quel et non pas comme caractère d'échappement.
Exemple : `plt.title(r'Le cosinus de $\frac{\pi}{2}$ est nul')`

1 Tracé de fonctions "à problème"

1.1 Tracé de $x \longmapsto \sin(x)/x$

Cette fonction ($\mathrm{sinc}(x) = \sin(x)/x$), appelée « sinus cardinal » est définie $\forall x \in \mathbb{R}$. En effet, en zéro cette fonction est « prolongée par continuité » puisque $\sin(x) \approx x$ si $x \approx 0$. Donc $\mathrm{sinc}(0) = 1$

1-1. Prendre `x=np.linspace(-4,4,101)*np.pi`, tracez et sauvegardez en `pdf` la courbe $\sin(x)/x$, que remarquez-vous ?

1-2. Pour éviter le problème en $x = 0$, substituez la valeur 0 par `eps` dans `x`. Tracez et sauvegardez.

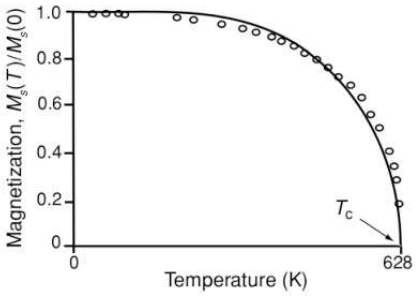
1.2 Tracé de $x \longmapsto \tan(x)$

1-1. En utilisant `np.arange` définir $x \in [-\frac{3\pi}{2}, \frac{3\pi}{2}]$ avec un pas de $\pi/100$, tracer $\tan(x)$. Sauvegardez la figure dans un fichier `pdf` Remarque ?

1-2. Pour avoir un graphe plus représentatif, on va couper les valeurs de $|(\tan(x)|$ supérieure à une valeur maximale qu'on peut prendre égale à 10.
Une solution est de remplacer les valeurs non désirées par `np.NaN`
Tracez et sauvegardez le graphe ainsi obtenu.

2 Résolution d'une équation non linéaire

On étudie la transition de phase entre la phase paramagnétique (phase d'aimantation spontanée nulle) et la phase ferromagnétique (phase d'aimantation spontanée non nulle). La figure suivante tirée de (i) présente l'évolution de l'aimantation du nickel avec la température.



Dans le modèle d'Ising, on obtient une équation qui régit le comportement de l'aimantation normalisée (i) m en fonction de la température normalisée $t = \frac{T}{T_C}$ (T_C : température critique ou température de Curie, en dessous de laquelle la phase ferromagnétique apparaît) : $m = \tanh\left(\frac{m}{t}\right)$ (ii)

2-1. **Résolution graphique**

On se propose de résoudre graphiquement cette équation en traçant, pour $x \geq 0$:

— la courbe $m = \tanh(x)$

— la droite $m = t \times x$

2-1-a) Pour pour t variant de 0.25 à 1.5 par pas de 0.25 tracez $\tanh(x)$ et $t \times x$. Utilisez une boucle `for`. Limitez l'affichage en x et y à la zone utile (utilisez `plt.xlim(...)`. Affichez les légendes des courbes tracées. Utilisez la commande `str` qui permet de transformer un `float` en un `string` Sauvegardez dans un fichier `pdf`.

2-1-b) Pour quelles valeurs de t les 2 courbes ne se coupent-elles qu'en un seul point ?

2-1-c) Pour quelles valeurs de t, y-a-t-il 2 solutions à l'équation $m = \tanh(m/t)$

2-1-d) Pour quelle valeurs de t, m tend-il vers sa valeur limite 1 ?

2-2. **Résolution numérique**

Préambule :

L'instruction `fsolve` du module `scipy.optimize` à importer avec :

```
from scipy.optimize import fsolve
```

permet de résoudre de façon approchée des systèmes du type $f(x) = 0$.

Par exemple, pour résoudre numériquement $\cos(x) = x, (x > 0)$, on commence par définir la fonction $f(x) = \cos(x) - x$:

```
def f(x):
    y=np.cos(x)-x
    return y
```

Puis la solution x_0 est obtenue avec `x0=fsolve(f,x_ini)`

`x_ini` est la première valeur pour initier la recherche (prendre ici $x_{ini} = 1$). On obtient alors $x_0 = 0.7390851$

On peut ensuite vérifier que la solution est correcte en calculant $f(x_0)$

On veut donc résoudre numériquement l'équation $m = \tanh(m/t)$ pour $m > 0$.

(i). J. M. D. Coey, Magnetism and Magnetic Materials, CAMBRIDGE UNIVERSITY PRESS

(ii). L'aimantation M est le moment magnétique par unité de volume. L'aimantation à saturation M_S correspond à l'alignement de tous les moments magnétiques. L'aimantation normalisée est $m = \frac{M}{M_S}$.

(iii). Rappel : $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ est une fonction définie dans numpy : `np.tanh(x)`.

2-2-a) Commencer par définir la fonction $y = g(m,t) = \tanh(m/t) - m$

2-2-b) Déterminer la solution non nulle (m_1) pour $t = 0.5$ puis celle (m_2) pour $t = 0.9$.
Attention : $g(m,t)$ dépendant de m et de t, l'appel à `fsolve` doit comporter l'argument supplémentaire `args=(t)` : `fsolve(g,m_ini,args=(t))`. Consultez l'aide de `fsolve` pour les détails.

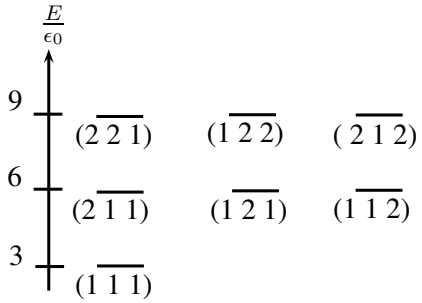
2-2-c) Écrire une boucle sur t_2 variant de 0.1 à 1.1 par pas de 0.1 pour déterminer les solutions m_0 et les tracer en fonction de t_2.

2-2-d) En utilisant l'instruction `subplot` (voir l'aide) qui permet d'afficher plusieurs graphes dans une seule fenêtre graphique, compléter le graphe courant précédent en traçant dans un 2ème graphe les courbes $\tanh(x)$, $t \times x$ et également leur point d'intersection x_0.
Là encore, limiter l'affichage en x et y à la zone utile.

2-2-e) Mettre des titres généraux aux figures, et des titres aux axes des 2 graphes.

3 Dénombrement d'états

On rappelle que les états (quantiques) d'une particule de masse m dans une boite cubique d'arête L (puits de potentiel infini à 3 dimensions) sont déterminés par (n_x, n_y, n_z) trois entiers strictement positifs. L'énergie d'un état est donnée par :
$E = (n_x^2 + n_y^2 + n_z^2)\epsilon_0$ avec $\epsilon_0 = \hbar^2\pi^2/(2mL^2)$.
Le but de cet exercice est de calculer $\Omega(E)$, le nombre d'états dont l'énergie reste inférieure à une valeur donnée E.
La figure suivante présente les 3 premiers niveaux et les états correspondants notés $(n_x\ n_y\ n_z)$.
On a donc $\Omega(6\epsilon_0) = 1 + 3 = 4$ états et $\Omega(9\epsilon_0) = 1 + 3 + 3 = 7$ états.



Comme $n_x^2 + n_y^2 + n_z^2 = E/\epsilon_0$ est l'équation de la sphère de rayon $\sqrt{E/\epsilon_0}$, une valeur approchée de $\Omega(E)$ est $\Omega_a(E) = \frac{1}{8} \times \frac{4\pi}{3} \left(\sqrt{E/\epsilon_0}\right)^3$ (le terme $\frac{1}{8}$ vient du fait que n_x, n_y, n_z sont positifs).
La valeur exacte, $\Omega_e(E)$, est la somme des dégénérescences des niveaux d'énergie inférieure ou égale à E : $\Omega_e(E) = \sum_{E_i \leq E} g_i$ où g_i est la dégénérescence de l'état d'énergie E_i.

Dans la suite, les énergie seront normalisées à ϵ_0 ce qui revient à prendre $\epsilon_0 = 1$.

Tracez sur un même graphe Ω_e et Ω_a pour $E_{max} = 30\epsilon_0$ puis pour $E_{max} = 10000\epsilon_0$.
N'oubliez pas « d'habiller » les figures (avec un titre général, les titres des axes, la légende des courbes, une grille si nécessaire, ...

Voici ce qu'on obtient pour $E_{max} = 12$

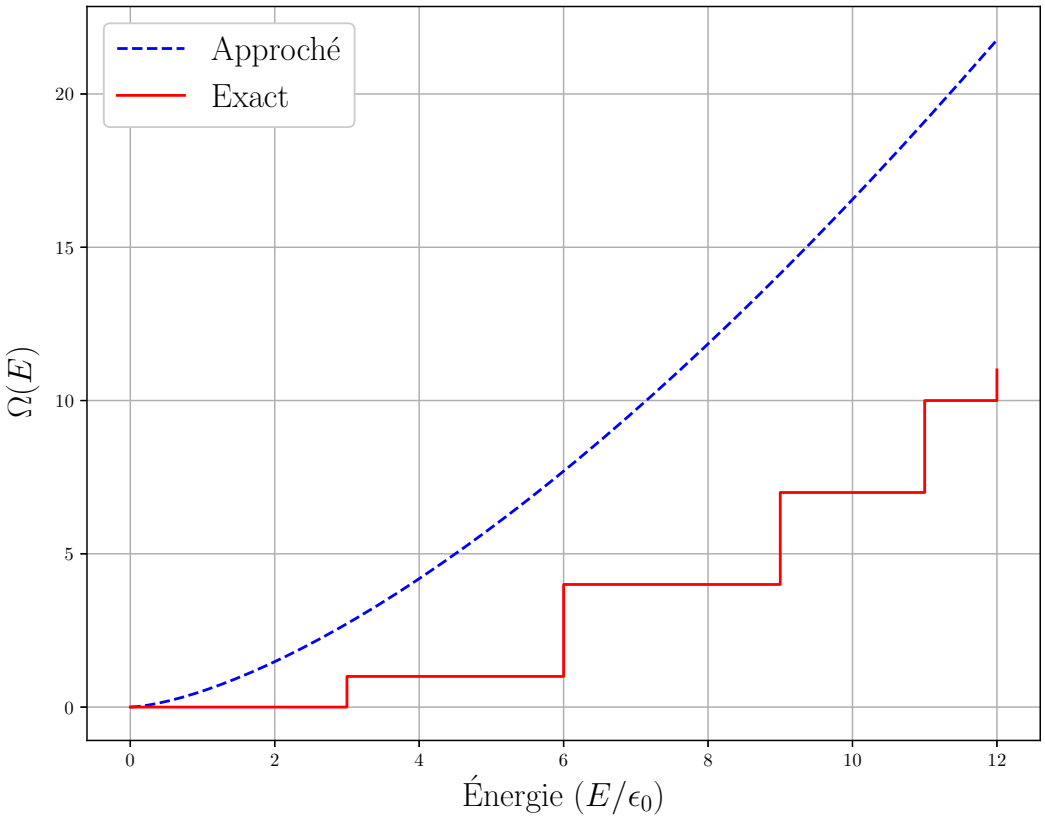


FIGURE 1 – Nombre d'états dont l'énergie ne dépasse pas une valeur donnée

Quelques pistes pour calculer Ω_e :

- Demandez la rentrée au clavier de E_{max} qui sera la valeur maximale jusqu'où Ω sera calculée. En déduire la valeur maximale des 3 entiers n_i.
- Créer la fonction `fE(x,y,z)` qui calcule la somme des carrés de x, y et z.
- à partir des vecteurs n_x, n_y, n_z créez une grille (avec `np.meshgrid`) sur laquelle sera évaluée la somme des carrés : $Somme$. Cette matrice représente tous les états d'énergie E possibles.
- Utiliser `np.unique(Somme,return_counts=True)` qui donne 2 arguments en sortie : le premier (qu'on peut appeler `TabE`) est un vecteur ne contenant que les valeurs différentes de $SommeZ$ et le deuxième(qu'on peut appeler `g`) le nombre de fois où ces valeurs apparaissent dans $Somme$ (dégénérescence). On ne gardera que les valeurs de `TabE` inférieures ou égales à E_{max}.
- Comme $\Omega_e(E) = \sum_{E_i \leq E} g_i$ où g_i est le nombre d'états différents de l'état d'énergie E_i, utilisez `np.cumsum` pour déterminer $\Omega_e(E)$.
- Pour le tracé de $\Omega_e(E)$, on pourra utiliser `plt.step(...)`.
- Attention à correctement placer les « marches » (les discontinuités) de $\Omega_e(E)$: cf figure 1.