COMPLEX_SYSTEM_NOTEBOOK

April 29, 2022

1 Physiques des Systèmes Complexes

```
[ ]: np, pylab as
     plt import scipy.sparse
     as sp from scipy.spatial
     import distance as dist
     import networkx as nx
     import scipy.integrate
     as integrate from sympy.solvers
```

Avis/Reflexion personnelle : Pourquoi, "moi", je trouve les problemes physiques et leur modélisation mathématiques "Belle" ?

1.1 Principe généraux

Le concept de mesure est une généralisation et une formalisation des mesures géométriques (distance/longueur, aire, volume) et d'autres notions courantes, telles que la masse et la probabilité des événements. Les mesures sont fondamentales dans la théorie des probabilités, la théorie de l'intégration et peuvent être généralisées pour prendre des valeurs négatives, comme pour la charge électrique. Les généralisations de mesure de grande envergure sont largement utilisées en physique quantique et en physique en général. On défini la mesure comme une application tel que l'union des famille dénombrable est égale à la somme des mesure disjointes :

$$\mu\left(\bigcup_{k=1}^{\infty} E_k\right) = \sum_{k=1}^{\infty} \mu(E_k)$$

Le théorème de Vaschy-Buckingham ou théorème Pi, est un des théorèmes de base de l'analyse dimensionnelle. Ce théorème établit que si une équation physique met en jeu n variables physiques, celles-ci dépendant de k unités fondamentales, alors il existe une équation équivalente mettant en jeu $n - k$ variables sans dimension construites à partir des variables originelles. Dans ce cas, on cherche à resoudre l'equation $f = 0$, tel que les variables sans dimensions π soit le produit des variables physiques q.

$$f(q_1, q_2, \ldots, q_n) = 0$$
$$F(\pi_1, \pi_2, \ldots, \pi_p) = 0$$
$$\pi_i = q_1^{a_1}\, q_2^{a_2} \cdots q_n^{a_n}$$

```
[ ]: # find polynomial (sympy method) pi
= solve(Symbol('x')**2
- 1, Symbol('x'))
# find polynomial (sympy method) g
= lambda x : (x^a).(x^b).(x^c).(x^d)
pi = fsolve(g,
[1, 1])
```

1.2 Mouvements individuels et collectifs

Un choc élastique est un choc entre deux corps qui n'entraîne pas de modification de leur état interne, notamment de leur masse. Dans un tel choc, l'énergie cinétique est conservée. Dans notre cas, nous considérons un ensemble de N corps (des billes) en collision dans un systeme isolé (pas d'echange avec l'exterieur), ce qui implique que les resultantes des collisions entre les paroies sont symétriques. Sans collision, le mouvement des billes suit une trajectoire rectiligne uniforme (1er loi de Newton), mais lorsqu'il y a un impact, on a une variation des vitesses qui apparait (2eme et 3eme loi de Newton). Pour calculer les changements de vitesse à l'impact, on utilise en coordonnée polaire $ (r,ϑ)$, les 3 loi de conservations fondamentales, tel que, la conservation du moment cinétique (ou angulaire), de la quantité de mouvement et de l'energie cinétique :

$$\begin{cases} \vec{\mathbf{r}_1} \wedge m_1\vec{\mathbf{v}_1} + \vec{\mathbf{r}_2} \wedge m_2\vec{\mathbf{v}_2} = \vec{\mathbf{r}_1'} \wedge m_1\vec{\mathbf{v}_1'} + \vec{\mathbf{r}_2'} \wedge m_2\vec{\mathbf{v}_2'} \\ m_1\vec{\mathbf{v}_1} + m_2\vec{\mathbf{v}_2} = m_1\vec{\mathbf{v}_1'} + m_2\vec{\mathbf{v}_2'} \\ m_1\mathbf{v}_1^2 + m_2\mathbf{v}_2^2 = m_1\mathbf{v}_1'^2 + m_2\mathbf{v}_2'^2 \end{cases}$$

On peut remarquer que si l'on avait un choc inélastique, une partie de l'energie cinétique serait converti en chaleurs/rayonnement, ce qui explique que tout systeme tend à se refroidir (à l'echelle moléculaire). Le moment angulaire correspond à l'unique vecteur, tel que le volume du parallelepipede engendré par les deux vecteur vitesse et de rayon donne le produit mixte $[u, v, w] = det(u, v, w) = (u \wedge v) \cdot w$, on peut aussi le voir comme le vecteur engendrée par une rotation (quaternion). On obtient les solutions vectorielles :

$$\mathbf{v}_1' = \mathbf{v}_1 - \frac{2m_2}{m_1 + m_2} \frac{\langle \mathbf{v}_1 - \mathbf{v}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle}{\|\mathbf{x}_1 - \mathbf{x}_2\|^2} (\mathbf{x}_1 - \mathbf{x}_2),$$
$$\mathbf{v}_2' = \mathbf{v}_2 - \frac{2m_1}{m_1 + m_2} \frac{\langle \mathbf{v}_2 - \mathbf{v}_1, \mathbf{x}_2 - \mathbf{x}_1 \rangle}{\|\mathbf{x}_2 - \mathbf{x}_1\|^2} (\mathbf{x}_2 - \mathbf{x}_1)$$

Il est possible de faire tourner les bords, à l'image d'une centifugeuse. L'ensemble du probleme est symétrique par rotation, si l'on veut appliquer une rotation, il suffit de faire le produit :

$$R(\theta).\vec{\mathbf{x}} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} .\vec{\mathbf{x}}$$

Cette derniere opération permet de visualiser les force fictives, tel que la force centrifuge, ou la force de coriolis.

[2]:
```
r2, v1, v2,
m1, m2):
# input variable M =
m1 + m2 #
conservation law d = np.linalg.norm(r1
- r2)**2
u1 = v1 -
2*m2 / M
* np.dot(v1-v2,
r1-r2)
/ d * (r1
- r2) u2
= v2 - 2*m1
/ M * np.dot(v2-v1,
r2-r1)
/ d * (r2
- r1) ##
return new velocity return np.array([u1,u2])

def border_collision(X,V):
x,y = X
### update position V[0,x
< 0] =
-V[0,x
< 0] V[0,x
> 1] =
-V[0,x
> 1] V[1,y
< 0] =
-V[1,y
< 0] V[1,y
> 1] =
-V[1,y
> 1] return
```

Ici nous avons considéré un systeme en l'absence de champs, c'est à dire de "force" externe pouvant perturber la trajectoire de l'objet lorsqu'il est en trajectoire rectiligne uniforme. Un champs peut etre la gravité, l'electromagnétisme et les autres interactions fondamentale. En mécanique classique, cela revient à ajouter une force dans l'expression du principe fondamental de la dynamique. Pour résoudre ce probleme, il suffit d'integrer la solution en temps en reecrivant la définition de la différentielle (méthodes d'Euler). En l'absence de force de frottement (ce qui est absurde à cette echelle), on obtient pour un champs quelquonque variant dans l'espace :

$$m\frac{\mathrm{d}\vec{\mathbf{v}}}{\mathrm{d}t} = m\vec{\mathbf{\Gamma}}(\vec{\mathbf{x}}) \Leftrightarrow \mathbf{v}_{\vec{t+1}} = \vec{\mathbf{v}_t} + dt.\vec{\mathbf{\Gamma}}(\vec{\mathbf{x}})$$

```
[ ]: x,y = X
     # spatial (exemple) g =
     lambda x,y :
     (x*y,
     9.81 + x+y)
     G = g(x,y)
     # euler method v[0]
     += dt*G[0]
     v[1] +=
     dt*G[1]
```

Il est aussi possible que chaque corps génère un champs local, lorsque l'on fait cela, on peut faire apparaitre des mouvement collectif, on parle alors de matière active. Le modele le plus simple est celui de Vicsek, où il y a des particules ponctuelles autopropulsées qui évoluent à vitesse constante et alignent leur vitesse avec celle de leurs voisines en présence de bruit.

$$v(\mathbf{r},\Theta) = \mathbf{r}.e^{i\Theta} = \Re(v) + i\Im(v)$$
$$\Theta_i(t+\Delta t) = \langle\Theta_j\rangle_{|r_i - r_j|<R} + \eta_i(t)$$
$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + v\Delta t \begin{pmatrix}\cos\Theta_i(t)\\ \sin\Theta_i(t)\end{pmatrix}$$

Ce modele décrit une forme d'universalité, car il permet d'etudier des phénomènes physique (cristaux liquide), biologique (essaimage, bactérie) et sociologiques (suiveur).

```
[1]: # complex U = v[0]+1j*v[1]
     # construct t+1 r =
     [] for u in
     U : d = np.absolute(u
     - U) index
     = np.where((d<Radius))[0]
     # circular mean polar_sum
     = np.sum(np.exp(1j*np.angle(U[index])))
     angle_mean = np.angle(polar_sum)
     r += [[np.cos(angle_mean),
     np.sin(angle_mean)]]
     # return new pertubate position return
     np.array(r)
```

La visualisation de particule en collision sous forme de bille est un probleme qui permet d'avoir une vue d'ensemble, une intuition de la plupart des problemes physiques classique. Elle permet de comprendre les principes de conservation, l'emergence de la notion de frottement, la pression d'un gaz, la notion de température, etc. Néamoins, ce type d'approche intuitive ne permet pas de resoudre tout les problemes, en particulier à la complexité calculatoire, il est necessaire pour cela faire appel à d'autre niveau d'abstraction. On verra dans les 2 prochaines partie, 2 cas, l'un où l'on a transport de matière avec des réactions chimiques et l'autre où les particules formes des aggrégats.

1.3 Milieu continu et Transports

La diffusion de la matière désigne la tendance naturelle d'un système à rendre uniforme le potentiel chimique de chacune des espèces chimiques qu'il comporte. La diffusion chimique est un phénomène de transport irréversible qui tend à homogénéiser la composition du milieu. Dans le cas d'un mélange binaire et en l'absence des gradients de température et de pression, la diffusion se fait des régions de plus forte concentration vers les régions de concentration moindre. Le potentiel chimique, en tant que dérivée partielle d'un potentiel thermodynamique, peut être défini l'energie libre ou par :

$$\mu_i = \left(\frac{\partial U}{\partial n_i}\right)_{V,S,n_{j\neq i}} = \left(\frac{\partial G}{\partial n_i}\right)_{P,T,n_{j\neq i}} \mathrm{d}U = -P\,\mathrm{d}V + T\,\mathrm{d}S + \sum_{i=1}^{N} \mu_i\,\mathrm{d}n_i G = V\,\mathrm{d}P - S\,\mathrm{d}T + \sum_{i=1}^{N} \mu_i\,\mathrm{d}n_i$$

Dans notre cas, nous considérons que la diffusion D est le resultat des collision de sphères dur et où l'ensemble des mouvement donne l'homogénéisation du systeme global. Si l'on considère que l'on est dans un milieu sans mouvement (n'est pas un écoulement) et incompressible (n'est pas un gaz), on peut considérer un decoupage du systeme où chaque petite boite va avoir une interaction de transport de matiere de proche en proche, on parle d'equation locale de diffusion avec un rayon d'action unitaire. Dans le cas 1D, on retrouve une forme approché de la loi de Fick :

$$c(t+1, x_i) = h + D \sum_{r<R} c(t, x_j) = h + D(c(t, x_{i+1}) + c(t, x_{i-1}))$$

Lorsqu'on a des molécules différentes avec des interactions (réactif), il est necessaire de construire un système d'équation pour chaque composante moléculaire. Néamoins, on peut aussi réduire se probleme à une seule equation, dans le cas où l'on considère des boites "binaire", où soit elle ont une concentration maximale d'un réactif, soit l'autre, on parle alors d'automate cellulaire. On obtient l'equation actif/inactif suivante :

$$c(t+1, x_i) = sign\left[h + \sum_k D_k \sum_{r<R_k} c(t, x_j)\right]$$

[3]:
```
new_C = np.zeros(len(C))
for i in range(len(C)):
c_i = 0 for
r,d in zip(R,D)
: d = np.linalg.norm(X
- X[i],
axis=1) c_i
+= d*np.sum(C[d<r])
# add not included new_C[c]
= np.sign(h+c_i)
```

Ici la répartition des points dans l'espace peuvent etre quelconque, on pourrait tres bien avoir une ligne, une répartition aléatoire, ou encore des maillages. Pour avoir un probleme symetrique par translation et par rotation, on utilise des mailles conventionnelles connu aussi sous le nom de réseau

de Bravais (famille cristalline + mode de réseau). Elle est décrit dans un espace bidimensionnel par la relation de translation :

$$\mathbf{R} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2$$

On distingues 3 familles : orthorhombique, similaire à un rectangle, quadratique, similaire à un carré et haxagonale, où l'angle est de 120°. Dans notre cas, on construit maille hexagonale d'un vecteur linéarisé contenant les coordonnée des points.

[ ]:
```
p, hexagonal=False):
# hexagonal case if hexagonal
: a,b,theta
= 1,1,120
else : a,b,theta
= p # Al Kashi Theorem
H = a**2
+ b**2
- 2*a*b*np.cos(theta*np.pi/180)
# find (d,c) grid translation c
= (H**2
- b**2
- a**2)/(2*b)
d = np.sqrt(np.abs(a**2
- c**2))
# grid print(b,d,c)
grid = np.mgrid[0:N,
0:N].astype(float)
# grid transform grid[1]*=
b grid[0]*=
d grid[1,::2]
+= c/N
# return coordinate return grid.reshape(2,-1).T
```

Pour chaque point, on calcule le resultat de l'équation suivant un rayon d'action, mais dans le cas où le nombre de reactif devient trop grand, on ne peut pas utiliser l'astuce precedante et il devient compliqué de resoudre un tel système. De plus, si la taille du rayon d'action est tres superieur à la distance entre les centroides, alors les points de l'espace suivent un continuum où leur valeur ne varie pas beaucoup entre les points. Pour simplifier un tel systeme, on peut appliquer la méthodes d'approximation des milieux continues (serie de Taylor) :

$$f(x_0 + a) = f(x_0) + a\frac{\mathrm{d}f(x_0)}{\mathrm{d}x} + \frac{a^2}{2}\frac{\mathrm{d}^2 f(x_0)}{\mathrm{d}x^2} + O(n)$$

Lorqu'on remplace cette expression dans l'equation du transport discrete $c(t+1, x_i)$, les termes d'ordre 1 se simplifient, ce qui permet de retrouver l'equation de la diffusion. On a dans le cas d'un seul element et suivant une seule dimension :

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x}$$

Pour résoudre cette equation différentiel, il existe 2 grandes méthodes, la premiere par différence fini (Euler), la seconde par les éléments finis (integration directe). Dans notre cas, nous utilisons uniquement le schéma d'Euler, où la dérivée seconde peut se reecrire dans le cas 2D (Laplacien) de la sorte :

$$u''(x,y) = \frac{1}{h^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} u(x,y)$$

Concrètement, cela revient à calculer la moyenne glissante en chaque points par un noyaux local (convolution). Néamoins, il est aussi possible de resoudre directement l'equation differentiel par le passage dans l'espace propre du systeme. Dans le cas de la dérivée seconde à une dimension, l'equation se reecrit :

$$\frac{U_{n+1} - U_n}{dt} - \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & & & & \\ & & & & -1 & 2 \end{bmatrix} U_n = 0 \Leftrightarrow U_{n+1} = M \cdot U_n$$

Contrairement à l'expression precedante, nous trouvons la solution en $t+1$ par la resolution de l'équation linéaire. Cela revient à trouver les solution du polynome de matrice tel que $p(X) = \det(XI_n - A) = X^n + p_{n-1}X^{n-1} + \ldots + p_1 X + p_0 = 0$ (Cayley-Hamilton). La construction de la matrice M en 2D est simplement remplacé par le schéma d'Euler 2D (diagonale = 4) et est creuse, par contre, il est necessaire de prendre en compte le cas où le probleme est asymétrique.

```
[ ]: kernel=False):
if kernel : #
convolution kernel kernel = np.array([[0,1,0],[1,-4,1],[0,1,0]],np.float32)*dt/
 ↪(h**2)
# resolution LAP =
signal.convolve2d(U,
kernel, boundary='symm',
mode='same')
else : # slices method
LAP = (U[0:-2,1:-1]
+ U[1:-1,0:-2]
- 4*U[1:-1,1:-1]
+ U[1:-1,2:]
+ U[2:
,1:-1])*dt/(h**2)
return LAP

def laplacian_global(U,h,dt):
N,M = U.shape
U_ = np.reshape(U.T,N*M)
# dimension choice l =
h*(N/M)
# diagonal construct (with border correction)
lower = (-1/l**2)*np.ones(N*M);
lower[N::N]
= 0 main =
(1/dt+2/(l**2)+2/(h**2))*np.ones(N*M)
upper = (1/l**2)*np.ones(N*M);
upper[N-1::N]
= 0 # asymetric
(if l fixed) asym = (-1/h**2)*ones(N*M)
# matrix construction M =
spdiags(main,[0],N*M,N*M)
+ spdiags(upper,[1],N*M,N*M)+spdiags(lower,[-1],N*M,N*M)
M += spdiags(asym,[N],N*M,N*M)+spdiags(asym,[-N],N*M,N*M)
# resolution new_U
= spsolve(M,
U) LAP = (new_U-U_).reshape(LAP)
```

Précedement on a pris uniquement en compte la diffusion de deux potentiels chimiques sans interactions. Néamoins, il est frequent que les especes chimiques interagissent entre elle, on parle alors de réactifs. On cherche ainsi l'evolution de la concentration des especes chimiques au cours du temps. L'evolution de la reaction est défini par la probabilité de collision entre 2 réactifs et où les valeurs sont obtenu empiriquement par spectroscopie. Pour le cas d'une reaction à vitesse k, soit \$ 2A + B $\xrightarrow{k}$ 2C\$($exemple$: $2dihydrogneavecdudioxygenedonnant2molculesd'eau), l'volutiondelaprobabilitdecollisions'crit$:

$$\frac{\mathrm{d}[A]}{\mathrm{d}x} = \frac{\mathrm{d}[B]}{\mathrm{d}x} = -\frac{\mathrm{d}[C]}{\mathrm{d}x} = -k[A]^2[B]^1$$

Dans le cas où l'on a un système de reaction chimique, il y a juste à sommer des termes croissant et décroissant des réaction. Une reaction connu est celle du modèle de Gray-Scott, où l'on a le système de reaction, avec ses equations differentielle :

$$\begin{cases} U + 2V \xrightarrow{1} 3V \\ P \xrightarrow{\mathrm{f}} U \\ V \xrightarrow{\mathrm{f+k}} P \\ U \xrightarrow{\mathrm{f}} P \end{cases} \Leftrightarrow \begin{aligned} \frac{\partial u}{\partial t} &= -uv^2 + f(1-u) \\ \frac{\partial v}{\partial t} &= +uv^2 - (f+k)v \end{aligned}$$

Cette expression s'ajoute à l'equation de diffusion tel que $\partial_t q = \underline{D}\,\Delta q + R(q)$, avec R, les reactions locales.

[ ]:
```
# grayscott uvv =
u*v*v
# reaction Ru =
- uvv + F*(1-u)
Rv = + uvv -
(F+k)*v
```

Pour etudier les effets des différents reactifs sur leurs propres evolution, on fixe le ratio des coefficients de diffusion et on fait varier les vitesses des reactions. On obtients ainsi un diagramme des différents motifs possibles. On remarque ainsi qu'une pertubation locale ou un bruit global va former la propagation d'une instabilité. Cette instabilité va se stabiliser est former des motifs périodiques. Ainsi, du hasard on obtient une organisation spatiale structuré, néamoins parfois, une simple perturbation peut entrainer des cycles d'instabilité à l'infini alors meme que notre systeme à été en partie grandement linéarisé. Cette notion, d'instabilité cyclique qui n'atteind jamais un equilibre est l'une des définitions de ce que l'on appele le chaos et les fractales, ce qui est vu plus en détail dans la seconde branche.

1.4 Equilibre, Chaos et Fractales

Une autre facon de voir la diffusion est de le voir comme un mouvement brownien. Le mouvement brownien, est une description du mouvement d'une Â« grosse Â» particule immergée dans un fluide et qui n'est soumise à aucune autre interaction que des chocs avec les Â« petites Â» molécules du fluide environnant. Ce mouvement est décrit comme aléatoire est peut etre modélisé par une marche aléatoire telque :

$$\mathbb{P}\Big(X_{n+1} = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1}, X_n = i\Big) = \mathbb{P}\left(X_{n+1} = j \mid X_n = i\right).$$

Lorsqu'on mesure la moyenne quadratique de ce deplacement quadratique, on retrouve aussi l'expression du coefficient de diffusion $ $\langle, X\Theta 2(t)\,\rangle = 2, d, D, t\$etcorrespondl'energied'unsignal.Danscecasparticulier, onpeutconsidrerquecesgrossepa$

$$\vec{F} = q\vec{E} + q\vec{v} \wedge \vec{B} \Leftrightarrow X < R \Rightarrow X_{t+1} = X_t$$

```
[ ]: N_WALKER = (state
     == 0).size
     N_AGGREG = (state
     == 1).size
     # calculate new position of walker WAY
     = 2*np.pi*(np.random.randint(0,4,N_WALKER)/4.)
     VAR = delta*(np.random.random(WALKER_NUMBER)
     - 0.5) Z
     = np.exp(1j*(WAY
     + VAR)) X[state
     == 0] +=
     np.stack((Z.real,
     Z.imag)).T
     # distance calculation tree Dist
     = dist.cdist(X[state
     == 0],X[state
     == 1], 'euclidean')
     # network (source,target,time) tree[state==0][(Dist
     < dmin).any(1)][1]
     = np.where()
     tree[state==0][(Dist
     < dmin).any(1)][2]
     = max(tree[2])+1
     # change state state[state==0][(Dist
     < dmin).any(1)]
     = 1 return X,
```

Lorsqu'on réalise cette expérience, on obtient des formes en dendrite. Ces formes se repete par homothétie, c'est à dire que la forme finale est une transformation en un point de la forme initiale. Pour mesurer la maniere dont le motifs rempli l'espace, on ne peut ni utiliser la notion de distance, ni la notion de surface car l'une peut tendre vers l'infini et l'autre vers 0. Dans ce cas, on utilise une dimension intermédiaire, et où le degrée exprime le niveau d'autosimilarité de la structure, c'est la dimension fractale. Dans le cas d'une simple homothétie avec un rapport de meme taille, on cherche à calculer :

$$a^{D_h} \; ; \; D_h = \frac{\ln(N)}{\ln(\frac{1}{r})}$$

Pour la mesurer, on calcule la distance qui sépare le centre d'agregat du nouveau point aggrégé à chaque instant t. Ensuite on va calculer entre chaque temps, l'evolution du rapport de dimension. On peut démontrer que la relation est linéaire, et où la pente caractérise la dimension fractale de notre probleme, tel que :

$$C(r) \equiv N^{-1} \sum_{r'} \rho(r')\rho(r'+r)$$

Dans notre cas, nous mesurons les differentes distances qu'il y a dans l'arbre généalogique généré par les collisions d'aggrégation. Par facilité, nous mesurons le plus courts chemin entre chaque point (Dijkstra) et les noeuds qui augmente la communication entre les noeuds (indicateur de proximité).

[ ]:
```
# graph G = nx.from_pandas_dataframe(tree,
'min_dist_index',
'initial_index')
# position node

# calculate topological distance SHORT_PATH
= nx.shortest_path(G)

# closeness
```

On obtient ainsi une distance fractale entre 1 et 2. Ici on considère que le mouvement est aléatoire est non déterministe, néamoins, nous savons que le mouvement aléatoire est la resultante des collisions deterministe entre particule, ce qui implique qu'hormis les phénomènes quantiques de fluctuation, les problemes sont pseudo-deterministe tendant toujours vers des etats d'instabilité par augmentation de l'entropie (theorie de l'information).

$$H_b(X) = -\mathbb{E}[\log_b P(X)] = \sum_{i=1}^{n} P_i \log_b \left(\frac{1}{P_i}\right) = -\sum_{i=1}^{n} P_i \log_b P_i.$$

On peut imaginer un cas d'une molécule tri-atomique, ou la premieres suit une trajectoire rectiligne uniforme, la seconde à un degrée de liberté suivant les bords d'un steradian du premier atome et le derniers fait la meme chose avec la seconde molécule. Suivant la position initiale avec un choc, les mouvements de cette molécule peut s'apparenter à celui d'un double pendule. Ce probleme peut se resoudre par un point de vue de minimisation de l'action d'un systeme : Le lagrangien d'un système dynamique est une fonction des variables dynamiques qui permet d'écrire de manière concise les équations du mouvement du système, et cela, par le principe de symetrie de Noether.

$$L = E_c - E_p = T - V \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} = \frac{\partial \mathcal{L}}{\partial q_k}$$

Ce qui donne dans le cas de coordonnée polaire $ q_k = (ϑ_1, ϑ_2)$, les coordonnées de symétrie, et $ v = = = l $, la vitesse suivant la longeurs d'arc (regles de la chaine). Ce qui donne dans le cas d'un double pendule :

$$\dot{p}_{\theta_1} = \frac{\partial L}{\partial \theta_1} = -\tfrac{1}{2} ml^2 \left(\dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + 3\frac{g}{l}\sin\theta_1\right)$$
$$\dot{p}_{\theta_2} = \frac{\partial L}{\partial \theta_2} = -\tfrac{1}{2} ml^2 \left(-\dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + \frac{g}{l}\sin\theta_2\right).$$

On peut simplifier les fonction trigonométrique en passant par la théorie des perturbations, c'est à dire les petites pertubation (developpement limité), mais dans notre cas, nous utilisons une résolution numérique. Ce système possède des solutions périodiques décomposables en deux modes, mais il est chaotique, il ne possède des solutions ni périodiques ni pseudo-périodiques. Pour visualiser l'effet de la sensibilité au condition initiale, on peut prendre des valeurs de θ tel que la position initale du bout du pendule soit au dessus de la fixation et on mesure le temps qu'il faut pour que le pendule bascule vers la position du dessus. On observes que le temps caracteristique est de puissance de 10 :

$$10^{\alpha}\sqrt{\frac{l}{g}}$$

Nous pouvons aussi observer le portrait de phase du systeme, càd, la trajectoire du système suivant les coordonnées vitesse/position $\dot{\theta}(\theta)$ (espace des phases). Cela permet de caractériser la présence d'un attracteur, d'un répulseur (voir nature des extremas) ou d'un cycle limite pour les valeurs de paramètres et des conditions initiales choisies.

```
[ ]: t): #
     [th1, w1, th2, w2] res = np.zeros_like(state)
     res[0] =
     state[1] #
     L1 del_ = state[2]
     - state[0]
     den1 = (M1 +
     M2)*L1 -
     M2*L1*cos(del_)*cos(del_)
     res[1] =
     (M2*L1*state[1]*state[1]*sin(del_)*cos(del_)
     + M2*G*sin(state[2])*cos(del_)
     + M2*L2*state[3]*state[3]*sin(del_)
     - (M1 +
     M2)*G*sin(state[0]))/den1
     # L2 res[2]
     = state[3]
     den2 = (L2/L1)*den1
     res[3] =
     (-M2*L2*state[3]*state[3]*sin(del_)*cos(del_)
     +(M1 + M2)*G*sin(state[0])*cos(del_)
     - (M1 +
     M2)*L1*state[1]*state[1]*sin(del_)
     - (M1 +
     M2)*G*sin(state[2]))/den2
     return res

     def integrate(state,
     t) res = integrate.odeint(derivs,
     state, t) x1,
     y1 = L1*sin(res[:,
     0]), -L1*cos(res[:,
     0]) x2,
     y2 = L2*sin(res[:,
     2]) + x1,
     -L2*cos(res[:,
     2]) + y1
```

On a vue dans les 2 dernieres partie que les mouvements chaotiques sont tres présents dans les systemes à multi-echelle, on parle d'ailleurs pour cela d'effet Papillons. Dans les prochaines partie, on restera sur des problemes d'echelle macroscopique, toujours avec le double pendule et aussi les problemes de chauffages. On verra, comment décrire et controler un probleme (1) d'abords sous forme de signal entrée/sortie, et (2), comment généraliser un problèmes sous formes de chaine de decision markovienne d'esperance.

1.5 Système et Traitement du signal

Un signal est une fonction qui véhicule une information. Toute quantité pouvant varier dans l'espace ou dans le temps peut être utilisée comme signal pour partager des messages entre observateurs. Un signal peut également être défini comme tout changement observable d'une quantité dans l'espace ou dans le temps, même s'il ne contient pas d'information. De cette facon, n'importe quel systemes peut etre appréhendé sous l'angle du traitement du signal. Lorsque le systeme est linéaire, on peut décrire un systeme par la fonction de transfert suivante :

$$Y(s) = H(s)\ X(s)$$

L'idée est de trouver l'expression de $H(s)$. Pour cela, on utilise ce que l'on appele une réponse impulsionnel, en effet, lorsqu'on applique une impulsion au systeme linéaire et invariant, la sortie est caracterisé entiereemnt par la fonction de transfert. Une maniere de le voir est d'utiliser le produit de convolution $y = u\hat{a}h$ et la prorpriété $u(\cdot)\delta(\cdot) = u(0)\delta(\cdot)$. Ainsi, la réponse est equivalente à la somme des contribution décalé en temps, tel que :

$$y(t) = \int_{-\infty}^{\infty} h(t-\tau)u(\tau)d\tau = \int_{-\infty}^{\infty} h(\tau)u(t-\tau)d\tau.$$

Dans le cas ou le veut mesurer la diffusion décrit comme la mesure de la temperature d'un signal discret $X[n]$, on a la sortie qui est egales à l'expression de la dérivée temporelle $Y[n] = (X[n] - X[n-1])/Te$ (doit respecter le critere d'echantillonnage de Shannon). Pour résoudre ce probleme, on utilise ce que l'on appele les Transformations en Z. Elle transforme un signal réel du domaine temporel en un signal représenté par une série complexe et appelé transformée en Z. En utilisans les formules usuelle, on a de cette facons :

$$H[z] = \frac{Y[z]}{X[z]} = \frac{1 - z^{-1}}{T_e}$$

La description d'un filtre se fait par l'analyse des pÃ´les, càd, les valeurs pour laquelle la fraction diverge. Trouver la transformation inverse de la réponse impulsionnel nous permet de decrire ainsi completement notre systeme.

[ ]:
```
# H = scipy.signal.firwin()
# freq = scipy.signal.freqz(b,a)
# conv = scipy.signal.convolve()
return H,freq,
```

La description precedante permet de determiner les frequences avec laquele la fonction de transfert "filtre" le signal d'entrée. Pour comprendre cela, nous allons resoudre le probleme de diffusion et montrer que l'on peut décomposer n'importe quel signal en serie de fonction trigonométrique, c'est les series de Fourier. Pour le cas d'un problème de diffusion, on a $\partial_t T = \alpha \partial^2_{xx} T$ et où l'on peut appliquer la séparation de variables $T(x,t) = X(x)Y(t)$. Ainsi, lorsqu'on integre la forme différentielle, la solution s'ecrit comme une somme infini de fonction sinusoidale :

$$\frac{Y'(t)}{\alpha Y(t)} = \frac{X''(x)}{X(x)} T(x,t) = \sum_{n=1}^{+\infty} D_n \sin\left(\frac{n\pi x}{L}\right) e^{-\frac{n^2\pi^2\alpha t}{L^2}}$$

Ici D_n correspond aux différentes composante de diffusion en fonction des fréquences, soit l'amplitude des fréquence qui caractérise le système. Comme toute serie infinie, D_n peut avoir plusieur valeurs, car on peut lui meme le décomposer en somme de fonction (Théorème convergence analytique), mais pour cela, il faut qu'il y est une équivalence entre la somme et la fonction. Pour cela, il faut que la sommes quadratique des coefficient de Fourier soit de carré sommable avec la fonction initiale, tel que :

$$\sum_{n=-\infty}^{+\infty} |c_n(f)|^2 = \frac{1}{T} \int_T |f(t)|^2 \, \mathrm{d}t = \|f\|^2$$

La solution à ce probleme est de construire les coefficients de fourier comme l'integrale complexe maximisant l'inertie de la fonction à un frequence donnée par rapport au centre de masse. On parle alors d'harmonique d'un signal et s'ecrit sous la forme :

$$c_n(f) = \frac{1}{T} \int_T f(t) \mathrm{e}^{-\mathrm{i}2\pi \frac{n}{T} t} \mathrm{d}t$$

Cela montre que n'importe quel fonction peut etre approché par une serie complexe, mais cela à partir d'un moment où elle est linéaire. Dans l'espace frequenciel, le produit de convolution devient une simple multiplication. Néamoins, un probleme peut etre linéarisé par certaine astuce, il y a la méthodes du noyaux, mais une autre consiste à decouper l'espace en plusieurs morceaux avec des distances euclidiennes constante (cas 2D-3D). Ce cas s'applique particulierment pour les courbe parametré 2D quelquonque (exemple : Courbe de Bezier) formant des motifs ou les points n'ont pas le meme espacement.

```
[ ]: control1, control2,
end): inputs =
np.array([start,
control1, control2,
end]) cubic_bezier_matrix
= np.array([
[-1,
3, -3,
1], [ 3,
-6, 3,
0], [-3,
3, 0,
0], [ 1,
0, 0,
0] ]) partial
= cubic_bezier_matrix.dot(inputs)
return (lambda t:
np.array([t**3,
t**2,
t, 1]).dot(partial))

def linearize_2Dshape(X,
nb=1000):
# rolling X_ =
np.roll(X,1)
# euclidean dist sum dist =
np.linalg.norm(X_-X)
d = np.cumsum(dist)
# decomposition (x,y) to (r,x,y) r
= np.linspace(0,
d.max(),nb)
x = np.interp(r,
d, X[0])
y = np.interp(r,
d, X[1])
# new X return np.concatenate([r[None],x[None],y[None]])

def fourier_coeff(X,
N_series = 100):
# manual version (else fft) T
= X[0].max()
cf = [] #
construct for (x,y) for n in
range(N_series):
cf += [[simps(X[1]*np.exp(-1j*2*n*np.pi*X[0]/T)/T,
X[0]),
simps(X[2]*np.exp(-1j*2*n*np.pi*X[0]/T)/T,
X[0])]]
return np.array(cf)

def fourier_series(X):
```

Ces outils ouvre les portes de plein de problématique, on a d'abords les problématiques de filtrage et régulation/commandes. Ces 2 problématiques sont tres différentes, mais fonctionne par leurs utilisation de fonction de Transfert. On va voir brievement 4 techniques qui utilise ce concepts : Les filtres passe-bande, les filtres de Kalman, les régulateurs PID et la commande LQ. Un filtre passe-bande est un filtre ne laissant passer qu'une bande ou intervalle de fréquences compris entre une fréquence de coupure basse et une fréquence de coupure haute du filtre. Pour appliquer un filtre, il suffit de faire le produit de convolution avec la sortie $h' = h_b * h_s$, avec h_s la fonction de transfert du système. La fonction de transfert d'un filtre passe-bande du second ordre s'écrit sous la forme :

$$h(jw) = \frac{A_0}{1 + j \cdot Q \cdot (x - \frac{1}{x})}$$

Le filtre de Kalman est un filtre récursif à réponse impulsionnelle infinie qui estime les états d'un système dynamique à partir d'une série de mesures incomplètes ou bruitées. L'algorithme utilise deux étapes, une étapes de prédiction pour prédire le dernier état et une étape de correction qui utilise la mesure pour corriger l'etat predit. Il est necessaire pour cela d'avoir une modélisation du système, pour l'equation differentiel de vitesse, on aurait $X_t = X_{t-1} + v_{(x,t-1)}.dt$. On se retrouve avec plusieurs matrice à utiliser :

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

Le régulateur PID, appelé aussi correcteur PID (proportionnel, intégral, dérivé) est un système de contrÃ´le permettant d'améliorer les performances d'un asservissement, c'est-à-dire un système ou procédé en boucle fermée. Le correcteurs doit etre parametré à la main et à pour but d'atteindre une valeurs de consigne $E(s)$ en minimisant l'erreur $\epsilon(s)$. Concretement, c'est un algorithme qui délivre un signal de commande à partir de la différence entre la consigne et la mesure (l'erreur) et dont le but est de résoudre :

$$u(t) = K_\mathrm{p}\epsilon(t) + K_\mathrm{i}\int_0^t \epsilon(\tau)\,d\tau + K_\mathrm{d}\frac{d\epsilon(t)}{dt}$$

En automatique, la Commande linéaire quadratique, dite Commande LQ, est une méthode qui permet de calculer la matrice de gains d'une commande par retour d'état. L'idée consiste à minimiser un critère de performance J, quadratique en l'état x et la commande u, et qui est une somme pondérée de l'énergie de x et de celle de u. Le but de la commande consiste, à la suite d'une perturbation, à ramener, de préférence aussi rapidement que possible, l'état à sa valeur d'équilibre 0. Un tel outils permet, si l'on connait l'equation differentiel du systeme, controler sa position. Dans le cas d'un pendule double en equilibre, on aurait la solution vu precedement, où l'on met en amont :

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) + \mathbf{v}(t)\mathbf{y}(t) = C(t)\mathbf{x}(t) + \mathbf{w}(t)$$

[ ]:
```
pass

def Kalman(X):
pass

def PID(X):
pass

def LQ_commande(X):
```

La notion de fonction de transfert nous permet d'une part, de trouver les equations qui caractérise le systeme s'ils sont linéaire et invariant, et d'autre part, nous permet de controler precisement la sortie de notre systeme si l'on connait les equations differentielle du systeme, qu'elle soit linéaire ou non. Le formalisme de fourier permet de valider une l'approche des fonctions de transfert par sa transformation inverse.

1.6 Décision et Potentiel

Lorsque les problèmes physiques et mathématiques sont difficile par l'utilisation des approches précedente, il est possible de passer par une méthodes probabiliste. Ces methodes probabiliste se base sur le theorement de transfert qui dit que l'on peut estimer l'integrale de n'importe quel fonction à partir de l'esperance d'un fonction à variable aléaoire de loi X. Sa forme générale est la suivante :

$$\mathbb{E}\left[\varphi(X)\right] \overset{\text{dÃ©f.}}{=} \int_{\Omega} \varphi\big(X(\omega)\big)\mathbb{P}(\mathrm{d}\omega) = \int_{\mathbb{R}} \varphi(x)\mathbb{P}_X(\mathrm{d}x)$$

Cette approche probabiliste est tres utilisé en Physique statistique, en particulier avec statistique de Maxwell-Boltzmann (loi de probabilité) pour déterminer la répartition des particules entre différents niveaux d'énergie, c'est une des base de la théorie cinétique des gaz qu'on a vue en premiere partie, mais aussi pour le modèle d'Ising sur un réseau de moments magnétiques (ferromagnétisme). Numériquement, cette methodes de resolution se nomme l'algorithme de Monte-Carlo.

[ ]:
```
a,b = border
# uniform  X =
np.random.uniform(a,b,(N,dim))
# algo  Y = []
for x in X
: Y += [f(X)]
Y = np.array(Y)
E = Y.sum()/dim
```

Avec cette méthodes on peut retrouver la dynamique d'un systeme physique, un exemple serait un chariot que l'on laisse tomber dans une vallée à plusieurs hauteurs aléatoire, on se retrouve ainsi avec des valeurs de position, de vitesse et de temps/coÃ»t pour atteindre un points fixe. Avec cela, on caracterise completement le systeme avec 3 parametre et l'on retrouve un portrait de phase par le calcul de l'esperance. La force de ce processus est que l'on peut ajouter des actions aléatoire

de décision discrete, dans le cas du chariot, aller à gauche ou à droite (2 action). Ces actions s'ajouterai à la fonction initiale et permettrai de savoir les meilleurs valeurs V^* pour atteindre un objectif donnée (une hauteur spécifique) en fonction d'un transition d'etats $s \rightarrow s'$. On obtient l'equation du gain suivant :

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} [R(s, \pi(s), s') + \gamma V^*(s')] T(s, a, s')$$

Pour trouver les meilleurs actions qui maximise la valeurs V^*, on a besoin d'une fonction supplémentaire Q^* qui va prendre directement en compte le meilleurs choix des precedante valeurs. Un tel outils permet de decrire n'importe quel probleme discretisable sous forme de chaine de markov et où l'on peut savoir quels sont les actions les plus adapté à l'etat donnée. Cette équation s'ecrit :

$$Q^*(s, a) = \sum_{s' \in S} [R(s, a, s') + \gamma \max_{a' \in A} Q^*(s', a')] T(s, a, s')$$

[6]:
```
min_value = 0,
step = 1):
discrete_state = (state-min_value)/step
return tuple(discrete_state.astype(np.int))

def q_table_construct(Ns,Na):
# Nb state, Nb action q_table
= np.random.uniform(low=-1,
high=1, size=(Ns,Na))
return q_table

def markov_decision(s,a,q_table,f,use=True):
# get values discrete_state
= discretise(s)
if use : action
= np.argmax(q[discrete_state])
else : action =
np.random.randint(Na)
# get action new_state,
reward, done,
info=f.step(action)
new_discrete_state = discretise(new_state)
# update Q(s,a) max_future_q
= np.max(q_table[new_discrete_state])
current_q = q_table[discrete_state][action]
new_q = (1-alpha)*current_q+alpha*(reward+gamma*max_future_q)
q_table[discrete_state][action]=new_q
# q and new s return q_table,
```

Le problème de decision se formule de manière imagée par le probleme du bandit manchot (processus markovien à un seul état). Un utilisateur (un agent), face à des machines à sous, doit décider quelles

machines jouer. Chaque machine donne une récompense moyenne que l'utilisateur ne connait pas a priori. L'objectif est de maximiser le gain cumulé de l'utilisateur. La politique de l'utilisateur oscille entre exploitation (utiliser la machine dont il a appris la récompense) et exploration (tester une autre machine pour espérer gagner plus). Formellement le regret s'exprime :

$$r_n = n\mu^* - \sum_{k=1}^{n} \mathbb{E}(\mu_{I_k})$$

Dans le cas où l'on ne prend en compte que l'exploitation, on a un politique gloutonne, où la fonction Q s'exprime simplement par $ Q(a') = Q(a) + (r_n - Q(a)) / N(a) $. Néamoins, si l'on veut que le probleme converge plus rapidement, il est plus frequent de trouver un dilemme entre l'exploiration/exploitation pour trouver la valeurs de l'esperence coherente avec le theoreme de transfert. Généralement, on commence avec 100% d'exploration, et au bout de 50%-70% de temps prévu de l'experience, on reste stabilisé à 5-10% d'exploration de facons à perturber le systeme et de trouver des cas où le modèle n'avait pas encore estimer certaine vallée de stabilité.

[ ]:
```
T): #
probability if t <
T/2 : p
= np.random.choice([True,False],
p=[t/(T/2),
1-t/(T/2)])
else : p = np.random.choice([True,False],
p=[0.9,0.1])
# for decision "use" return
```

Ce type d'outils est tres frequent aujourd'hui dans les probleme d'apprentissage où l'on a pas de donnée d'entrainement, on parle d'apprentissage par renforcement. On le retrouve particulierement couplé avec l'utilisation des reseau de neurones où ces dernier fonctionne comme estimateur de l'esperance.

1.7 Conclusion

L'ensemble de ses outils vu ici sont pour aller plus en profondeurs sur les illustrations intagram. Elle permettent d'avoir un apercu de la physique des systèmes complexes (emergence, controle, structures, etc.), mais ne permette pas de comprendre la physique plus modernes "quantique" et "relativité". Néamoins elle permette de comprendre certain aspect de la physique du quotidient. Pour chaque partie, vous retrouverez des codes utilisant chacune des fonction vu dans le notebook pour comprendre ce qu'ils font.

Note : c'est encore une feuille de note incomplete !