TP09 Intégration et tirage aléatoire

As usual, Il suffit de faire un « Pull » depuis GitHubDesktop pour récupérer le dossier associé.

Partie I

Applications directes du cours

I.1 Intégration

Implémentez les fonctions `integration_rectangle(f,a,b,n)` et `integration_trapeze(f,a,b,n)` qui intègrent la fonction `f` entre `a` et `b` sur `n` points d'échantillonnage par, respectivement, la méthode des rectangles et celle des trapèzes. Elle doit renvoyer la valeur de cette intégrale.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

I.2 Recherche d'un zéro d'une fonction par dichotomie

Implémentez la fonction `zero_dichotomie(f,a,b)` qui cherche[1] un zéro de la fonction `f` entre `a` et `b`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie II

Un chouïa de théorie

II.1 Introduction

Le problème que l'on se pose est le suivant : on dispose d'un échantillon (de masses d'étoiles, de valeurs de résistances, de tailles de personnes, de diamètres de bitoniau en plastique sur un lacet...) sur lequel on veut déterminer la probabilité qu'a une mesure donnée de tomber dans tel ou tel intervalle de valeurs.

II.2 Un exemple de distribution numérique

Par exemple, si l'on fait la somme de 10 nombres aléatoires pris entre 0 et 1, cette somme vaudra entre 0 et 10, avec une plus forte probabilité de valoir 5 (car, en moyenne, la moitié des chiffres vaut moins de 0,5, l'autre moitié plus de 0,5, les disparités se « compensant » à peu près). Mais stoppons les longs discours et regardons l'effet « en vrai ». Tapez la suite d'instructions suivantes dans un fichier annexe.

```
import random                          # Pour les tirages aléatoires
import matplotlib.pyplot as plt        # Pour les représentations graphiques
nb_tirages = 10000                     # Le nombre de tirages
sommes = [0]*nb_tirages                # Initialisation du stockage des sommes
for i in range(nb_tirages):            # Boucle sur les tirages
    for j in range(10):                # On choisit 10 nombres au hasard sur [0,1[
        sommes[i] += random.random()   # en rajoutant à la somme
plt.hist(sommes,bins=16,range=(1,9)) # Dessin de l'histogramme
```

1. par dichotomie...

```
plt.hist(sommes,bins=32,range=(1,9)) # Le même en 32 bins
plt.xlabel("Valeur de la somme")     # Description de l'axe x
plt.ylabel("Nombre d'evenements")    # Description de l'axe y
plt.title("Somme de 10 nombres aleatoirement choisis entre 0 et 1")
plt.savefig('TP09_somme_10.png')     # Enregistrement de la figure
plt.clf()                            # Nettoyage
```

On remarque que la distribution des valeurs n'est pas aléatoire mais suit une courbe en cloche et que la probabilité d'avoir une valeur comprise entre 4,5 et 5,5 est de l'ordre de $2 \times 2\,100/10\,000 = 42\%$. Plus la largeur des intervalles de comptage (« bins ») diminue, plus la probabilité de tomber sur un intervalle donné diminue (il y a toujours autant d'évènements à distribuer dans de plus en plus de boîtes), mais les évènements dont la somme est autour de 5 sont toujours les plus nombreux et l'allure de la courbe résiduelle est la même (même si la normalisation, *ie* la valeur du maximum, a changé)

II.3 Densité de probabilité

On voit que l'on peut donc généraliser le procédé et définir, pour une valeur x mesurée, une densité de probabilité $f(x)$ telle que la probabilité $\mathrm{d}p$ d'obtenir une valeur de la mesure située entre x et $x + \mathrm{d}x$ vaille $\mathrm{d}p = f(x)\,\mathrm{d}x$. Comme la somme de toutes les probabilités sur l'ensemble $[\,x_{\min}\,;x_{\max}\,]$ des valeurs de x possibles doit valoir 1, on doit imposer la normalisation

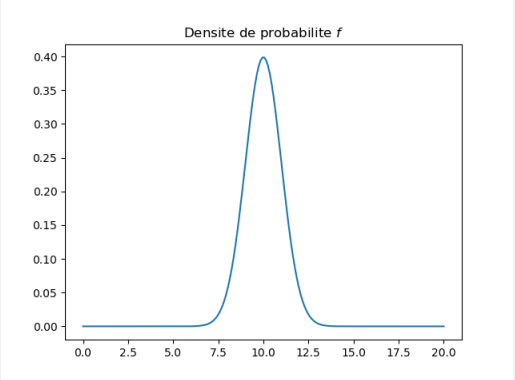
$$\int_{x_{\min}}^{x_{\max}} f(x)\,\mathrm{d}x = 1$$

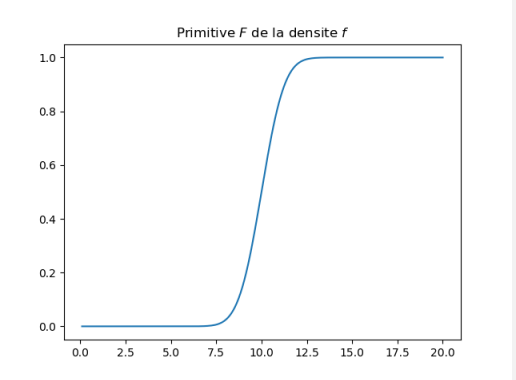
II.4 Tirer une valeur d'une distribution donnée

Se pose alors le problème inverse. Supposons que des séries de mesures à grande échelle aient pu déterminer la densité de probabilité $f(x)$. Vous voulez simuler un plus faible échantillon (comme par exemple préparer un amas stellaire en tirant « au hasard » les valeurs des masses de chacune des 10 000 étoiles de votre simulation). Comment passer de $f(x)$ au choix effectif des masses pour que celui-ci soit « vraisemblable » ? Définissons la densité cumulée F telle que

$$F : \begin{cases} [\,x_{\min}\,;x_{\max}\,] \longrightarrow [\,0\,;1\,] \\ x \longmapsto \displaystyle\int_{x_{\min}}^{x} f(x')\,\mathrm{d}x' \end{cases}$$

Cette fonction a une pente d'autant plus forte au point x que la valeur de f en x est grande. Ainsi, autour du maximum de f, l'intervalle image par F est d'autant plus important que le maximum est marqué. Il suffit alors de tirer un nombre aléatoire entre 0 et 1 et regarder son image par la fonction réciproque F^{-1} pour obtenir une valeur de x dont la densité de probabilité associée est bien f.





Partie III

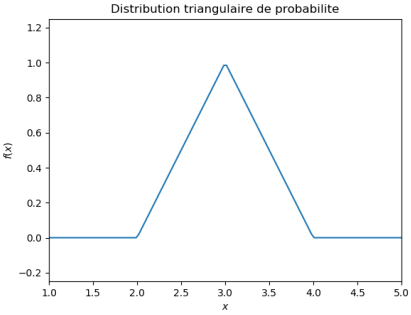
Applications

III.1 Ce qu'il va falloir faire

Avoir bien compris ce qui précède est certes utile pour faire la suite, mais pas forcément nécessaire [2]. Voici en détaillé ce que vous allez devoir faire :

1. Dans chacun des exemples suivants, il va falloir définir la fonction f correspondant à la densité de probabilité décrite de sorte que l'intégrale de cette fonction sur l'ensemble de définition vaille 1.
2. Une fois cette fonction définie, il faudra utiliser l'une de vos procédures d'intégration pour calculer la primitive $F(x) = \int_{x_{\min}}^{x} f(x')\,\mathrm{d}x'$.
3. Faire une boucle sur le nombre d'évènements [3] à tirer de manière aléatoire et pour chacun
   (a) En utilisant la fonction `random.random()` [4], tirer aléatoirement un nombre compris entre 0 et 1.
   (b) Adapter (ou utiliser avec ruse) la procédure de recherche de zéro par dichotomie pour trouver l'antécédent du nombre aléatoire par la fonction F : c'est la valeur de l'évènement cherché [5].
4. Une fois les tirages terminés, faire un histogramme de la répartition des évènements tirés (en s'inspirant de l'exemple donné en début de TP) pour vérifier que la répartition correspond bien à ce que vous vouliez simuler.

III.2 Distribution triangulaire



On considère une distribution triangulaire du type de celle représentée ci-contre (avec $f(2) = f(4) = 0$ et $f(3) = 1$). Tirer 10 000 évènements correspondant à cette distribution et représenter l'histogramme correspondant. Enregistrez le fichier sous le nom `TP09_tirage_triangle_VotreNom.png` (en remplaçant bien sûr la chaîne `VotreNom` par votre vrai nom) et pilotez la génération du fichier par une procédure nommée `tirage_triangle()`

Remarquez que le calcul peut être assez long (selon comment vous l'avez codé, cela peut représenter pas mal d'intégrales à calculer). Il pourra être utile d'utiliser la fonction `scipy.interpolate.interp1d(x,y)` qui permet d'interpoler n'importe quelle fonction y à partir d'un échantillonnage (`x` et `y` sont des tableaux). En outre, il est très facile d'établir la bijection réciproque d'une fonction : il suffit d'échanger les rôles de `x` et `y` (qui sont alors tous deux des tableaux triées). Voici un exemple d'utilisation :

```
>>> import scipy as sp
>>> import scipy.interpolate
>>> X = sp.linspace(0,20,100)
>>> fX= [Xi**2 - Xi + 2 for Xi in X]
>>> # interp1d renvoie une fonction qui renvoie un array: il faut donc ruser
>>> f = lambda x: float(scipy.interpolate.interp1d(X,fX)(x))
>>> f(12.321),12.321**2 - 12.321 + 2
(141.48648699112331, 141.486041)
```

Stop GitHub

Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

2. Cela fait partie des joies de la programmation informatique !
3. Notes, tailles, masses d'étoiles, ce que vous voulez...
4. On n'oubliera donc pas de commencer le programme par un `import random`...
5. Stockez-le dans un coin !

III.3 Distribution gaussienne

La distribution des notes dans une classe suit généralement une gaussienne, c'est-à-dire que l'on a $f(x) = A \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$ où μ est la moyenne de la distribution et σ son écart-type (σ^2 est la variance). Faire un tirage pour une classe de 44 élèves en ajustant la constante A de sorte que $x_{\min} = 0$ et $x_{\max} = 20$ et en imposant une moyenne à 10 et un écart-type de 3. Comparer l'histogramme obtenu à celui généré pour un tirage de $44 \times 3 = 132$ notes correspondant aux trois PCSI rassemblées en un DS commun. Enregistrez le fichier sous le nom `TP09_tirage_gauss_VotreNom.png` et pilotez la génération du fichier par une procédure nommée `tirage_gauss()`

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Vérifiez, en implémentant la fonction `moyenne_et_variance(liste)` que l'on retrouve bien aux environs de μ pour moyenne et aux environs de σ^2 pour variance. Vérifiez ce que cela donne pour la distribution précédente.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

III.4 IMF : Initial Mass Function

Lors de la naissance d'un amas stellaire, la distribution des masses m des étoiles (exprimées en masses solaires $M_\odot$) suit ce que l'on appelle l'IMF (pour « Initial Mass Function ») dont les astronomes pensent qu'elle est universelle, c'est-à-dire que quel que soit le coin de l'Univers observé, c'est cette distribution qui semble prévaloir. Celle-ci peut se modéliser [6] de manière simple comme un raccordement par continuité de trois lois de puissance

$$f(m) \propto \begin{cases} m^{-\alpha} & \text{si } m < 1\ M_\odot \\ m^{-\beta} & \text{si } 1\ M_\odot < m < 10\ M_\odot \\ m^{-\gamma} & \text{si } m > 10\ M_\odot \end{cases}$$

où l'on prend généralement $\alpha = 1{,}30$, $\beta = 2{,}35$ et $\gamma = 4{,}0$. Vous devez préparer des amas stellaires « nouveaux-nés » contenant chacun 10 000 étoiles pour lancer vos simulations de dynamique stellaire. On vous impose de plus les masses minimale et maximale admissibles pour vos étoiles qui sont respectivement $m_{\min} = 0{,}1\ \mathrm{M}_\odot$ et $m_{\max} = 100\ \mathrm{M}_\odot$.

1. Combien avez-vous, en moyenne, d'étoiles de plus de 1 $\mathrm{M}_\odot$ dans vos simulations ? de plus de 10 $\mathrm{M}_\odot$?
2. L'essentiel (90%) de la lumière de l'amas est fourni, durant ses premiers millions d'années d'existence, par les étoiles de plus de 5 $\mathrm{M}_\odot$: quelle fraction de la masse totale de l'amas représentent-elles ? En quoi cela peut-il être intéressant pour l'astrophysicien ?

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

6. Kroupa, P. 2002, Science, 295, 82