Programme n°7

ELECTROCINETIQUE

EL3 Les circuits linéaires du premier ordre (Cours et exercices)

EL4 Régime transitoire du second ordre (Cours uniquement)

- Observation - Circuit électrique
  - Conclusion
- Mise en équation - Cas général
  - Cas particulier où R = 0 Ω
  - Forme canonique (introduction du facteur de qualité)

CINETIQUE CHIMIQUE

CX1. Généralité sur la cinétique chimique (Cours et exercices)

CX2 Cinétique formelle, réaction et ordre (Cours et exercices)

- Ordre d'une réaction - Ordre au cours du temps
  - Exemples
  - Aspect expérimental → Ordre initial
    → Ordre global, ordre partiel
- Les réactions d'ordre simple - L'ordre 0
  - L'ordre 1
  - L'ordre 2
  - L'ordre 2
- Etude expérimentale de l'ordre d'une réaction - Aspect expérimental
  - La méthode intégrale
  - La méthode différentielle
  - La méthode du temps de demi-réaction
  - Méthode d'Oswald
- Influence de la température

TP

Mesure de résistances : montage courte ou longue dérivation. Mesure d'incertitudes.
Capacité numérique : simuler à l'aide d'un langage de programmation, un processus aléatoire permettant de caractériser la variabilité d'une grandeur composée. (le programme de base doit être fourni, l'élève doit savoir l'adapter)
Prise en main de l'oscilloscope : approche de la synchronisation, problème de masse…

```python
# distance entre 2 points.py

import numpy as np
import matplotlib.pyplot as plt

# Positions de des points

A = 11.1 # cm
B = 19.5 # cm

# Entrez les précisions

DeltaA = 0.5 # cm
DeltaB = 0.5 # cm

# Entrez la fonction de composition

def ecart(a,b):
    return b-a

# Entrez le nombre de simulation que vous voulez effectuer

N = 100000

# Calculs avec une distribution de probabilité uniforme
Difference = []

for i in range(0,N):
    a = np.random.uniform(A-DeltaA,A+DeltaA)
    b = np.random.uniform(B-DeltaB,B+DeltaB)
    Difference.append(ecart(a,b))

plt.figure(1)
plt.hist(Difference,bins = 'rice')
plt.title('Résultat du tirage aléatoire de la différence après simulation')
plt.xlabel("b-a (cm)")
plt.show()

# Calcul et affichage moyenne et écart type

moy = np.mean(Difference)
std = np.std(Difference,ddof=1)

print("Moyenne = {:.2f} cm".format(moy))
print("Ecart type = {:.2f} cm".format(std))
```

```python
# Montecarlo produit.py

import numpy as np
import matplotlib.pyplot as plt

# Mesure

f = 40983 # Hz
lambd = 0.00840 # m

# Entrez les incertitudes types et précisions

uf = 94 # Hz
ulambd = 0.00041 # m

Deltaf=uf*np.sqrt(3)
Deltalambd=ulambd*np.sqrt(3)

# Entrez la fonction de composition

def produit(a,b):
    return a*b

# Entrez le nombre de simulation que vous voulez effectuer

N = 100000

# Calculs avec une distribution de probabilité uniforme
Celerite = []

for i in range(0,N):
    a = np.random.uniform(f-Deltaf,f+Deltaf)
    b = np.random.uniform(lambd-Deltalambd,lambd+Deltalambd)
    Celerite.append(produit(a,b))

plt.figure(1)
plt.hist(Celerite,bins = 'rice')
plt.title('Résultat du tirage aléatoire du produit après simulation')
plt.xlabel("c (m/s)")
plt.show()

# Calcul et affichage moyenne et écart type

moy = np.mean(Celerite)
std = np.std(Celerite,ddof=1)

print("Moyenne = {:.2f} m/s".format(moy))
print("Ecart type = {:.2f} m/s".format(std))
```