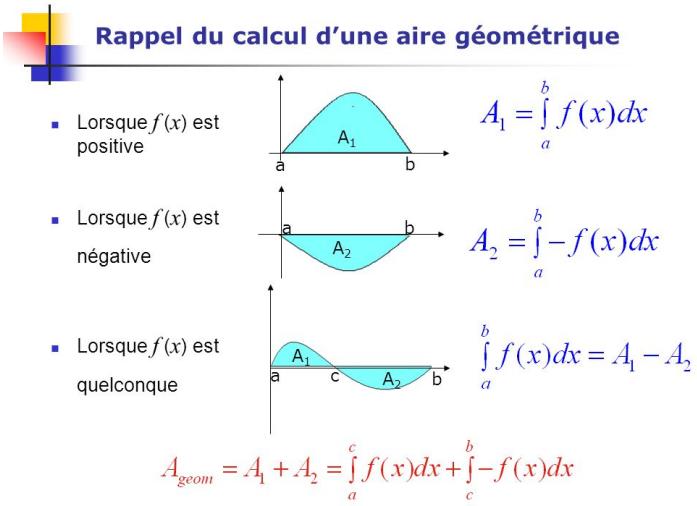
TP2 : INTÉGRATION NUMERIQUE



Noms 1 : kentsa meli habib edgar
Noms 2 : hamdane amine
Nom du prof : Cardinal Jean-Paul

Année : 2017/2018

1 Introduction

Dès la fin du TP1 il était question pour nous de commencer le TP2 qui est centre sur l'étude de l'intégrale de la fonction la fonction $f(x) = \sqrt{1-x^2}$. Le gros travail résidait sur la présentation de quelques méthodes numériques pour le calcul et l'approximation de l'intégrale de d'une fonction sur un intervalle borne et ferme. Les méthodes numériques que nous étudierons seront :

-La méthode du point milieu
-La méthode du trapèze
-La méthode de simpson
-La méthode de Monte-Carlo

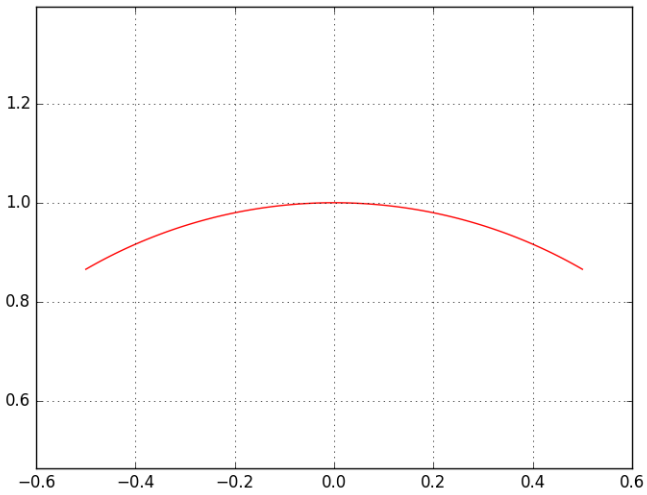
2 Fonctions, tableaux et commentaires

2.1 Définition et courbe de f

Après des recherches fastidieuses, nous avons trouve dans le module "math" la fonction "sqrt" qui nous a permis de mettre dans la racine la fonction $g(x) = 1 - x^2$.
Nous avons pu définir la fonction f comme suit :

```
1 def f(x):
2   y = sqrt(1 - x**2)
3   return y
```

Grâce au module "matplotlib", nous avons pu tracer la courbe de la fonction f sur l'intervalle [-0.5 ;0.5] et garce a "savefig" nous avons pu l'enregistrer. La représentation graphique de la fonction f est la suivante :



2.2 Méthode point milieu

Les résultats obtenus classés dans le tableau ci-dessous nous permettent de constater que le temps croit avec "n" qui est le nombre de subdivisions de l'intervalle sur lequel a été tracé la courbe et que ceci n'est pas pareil avec l'erreur qui est inversement proportionnelle a "n". Par contre,nous pouvons aussi dire que l'erreur est divisée par environ 10 lorsque le nombre de substitutions est multiplié par 10

```
Tableau avec la methode du point_milieu:
n         | erreur       |  temps (sec.)
-------------------------------------------------------------
       10|  0.0888225    |   4.1e-05
      100| 0.00868412    |   0.00015
     1000| 0.000866266   |   0.001302
    10000| 8.66049e-05   |   0.013559
   100000| 8.66028e-06   |   0.130921
    1e+06| 8.66026e-07   |    1.34687
```

Grâce à la documentation trouvée sur "https ://docs.python.org/3/tutorial/inputoutput.html", vous pourrez voir la reproduction de ce tableau sur l'écran en fichier texte et sur lequel l'utilisateur a le droit d'écriture.
Dans la question 2, les questions de a) a c) sont des questions qui ont été traitées a la main et qui vous seront rendues. l'implémentation de la fonction "point milieu" est :

```
def point_milieu(f ,a ,b , n):
  i = 0
  m = (b - a)/n
  x0 = a
  while i <= n:
    if (i == 0):
      xi = a
    else:
                xi = xi + m
                ci = (xi + m)/2
    i = i + 1
  S = 0
      i = 1
      while i <= n:
          si  = m*f(ci)
          S = S + si
          i = i + 1
  return S
```

2.3 Méthode du trapèze

Dans cette troisième question, nous avons pu donner une approximation de la fonction f sur $[x_{k-1}; x_k]$ grâce à la méthode du trapèze qui consiste à approximer f par la fonction affine qui prend les memes valeurs que f en x_{k-1} et x_k. L'implémentation de la fonction trapèze n'a été la plus facile néanmoins, nous avons pu l'implémenter comme suit :

```
def point_trapeze (f ,a, b, n):
  m = (b - a)/n
  xi = a
  xj = a
  i = 1
  S = 0
  while i < n:
            xi = a + m*i
    xj = a + m*(i - 1)
    si  = m*(f(xj) + f(xi))/2
         S = S + si
         i = i + 1
  return S
```

et un tableau récapitulatif des erreurs commises et du temps mis pour l'exécution est le suivant :

```
Tableau avec la methode du trapeze:
n         | erreur       |  temps (sec.)
-------------------------------------------------------------
       10|  0.0900884   |    5.6e-05
      100| 0.00869836   |   0.000268
     1000| 0.00086641   |   0.003059
    10000| 8.66064e-05 |   0.025136
   100000| 8.66029e-06 |   0.259175
    1e+06| 8.66026e-07 |    2.49519
```

Nous remarquons que bien que l'erreur soit la même à partie d'un certain rang, il n'en est pas de même pour le temps d'exécution qui est deux fois plus grand dans la méthode du trapèze que dans la méthode du point milieu.

2.4 Méthode de simpson

Pour la méthode de simpson, le principe est le même que celui de la méthode du trapèze sauf que au lieu d'utiliser une fonction affine pour approximer f, on utilise plutôt une fonction polynôme de degré 2.
Son implémentation est la suivante :

```
def point_simpson (f , a, b, n):
  m = (b - a)/n
  xi = a
  xj = a
  i = 1
  S = 0
  while i <= n:
            xi = a + m*i
    xj = a + m*(i - 1)
    si = (f(xi) + 4*f((xi + xj)/2) +f(xj))/6*m
    S = S + si
         i = i + 1
  return S
```

Le tableau récapitulatif des erreurs et du temps mis pour l'exécution en fonction des puissances de 10 est le suivant :

```
Tableau avec la methode simpson:
n         | erreur       |  temps (sec.)
-------------------------------------------------------------
       10 | 2.11626e-07  |    5.1e-05
      100 | 2.13808e-11  |    0.00039
     1000 | 2.44249e-15  |    0.003744
    10000 | 1.33227e-15  |    0.038553
   100000 | 5.9952e-15   |    0.36542
    1e+06 | 3.88578e-15  |    3.70452
```

Avec la méthode de simpson, on constate que les erreurs commises sont mi-

nimes par rapport au deux méthodes vues jusqu'ici, contrairement au temps d'exécution qui est relativement pareil voire même supérieur à parti d'un certain rang.

2.5 Comparaison

2.5.1 Erreurs

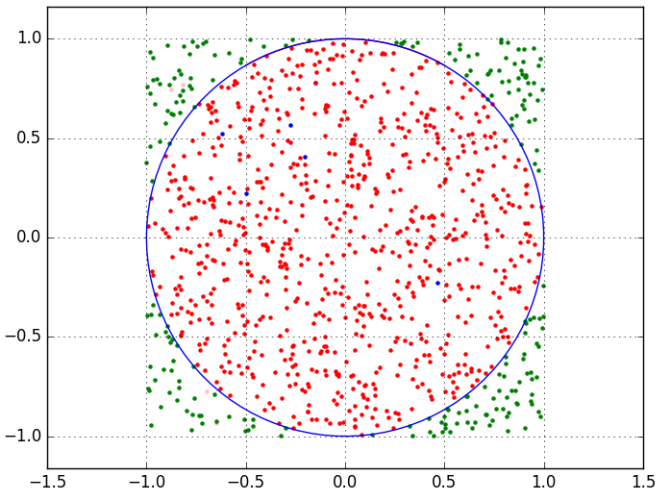
Comme nous le voyons depuis le haut, nous les erreurs qu'on obtient en utilisant la fonction point milieu sont inférieures aux erreurs avec la fonction trapèze jusqu'à parti d'un certain rang, rang au delà duquel les erreurs sont égales. Mais avec la méthode de simpson, les erreurs sont inférieures de l'ordre 10^{-7} en moyenne.

2.5.2 temps d'exécution

Ces différentes méthodes ont des erreurs différentes et aussi, notons que les temps d'exécution sont aussi différents.
la méthode du point milieu est la méthode la plus rapide et d'environ deux fois plus rapide que la méthode du trapèze qui elle est plus rapide que la méthode de simpson qui est environ trois fois plus rapide que la méthode du point milieu.

2.6 Méthode de Monté-Carlo

Le graphique ci-dessous, tracé à l'aide du module "matplotlib.pyplot", est le cercle de Monte-Carlo.



Le tableau regroupant les erreurs et le temps d'exécution en fonction de N est le suivant :

```
Tableau avec la methode monte_carlo:
N         | erreur              |  temps (sec.)
-----------------------------------------------------
        10|           0.0584073|   3.8e-05
       100|           0.0984073|  0.000137
      1000|           0.0384073|  0.001264
     10000|          0.00520735|  0.012643
    100000|         0.000872654|  0.083617
     1e+06|          0.00311935|  0.762843
```

Nous constatons dès le premier regard que la méthode de monte-Carlo est la méthode qui met le moins de temps à s'exécuter contrairement aux erreurs qui sont assez considérables.

3 Conclusion

Dans ce TP2, était question pour nous de donner une approximation de l'intégrale $\int \sqrt{1-x^2}dx$. Il en ressort qu'il existe plusieurs méthodes que nous avons énumérées à l'introduction. la méthode la plus rapide est la méthode de monte-Carlo avec un rapport de 10^{-2} entre les autres et elle. Pour ce qui est de de la méthode la plus précise, nous constatons que c'est la méthode de Simpson.