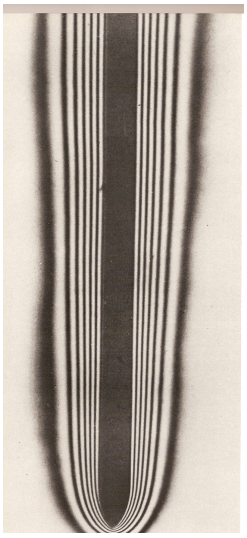
Python 22-23 for dummies : problème 8
Equation de Blasius



La fonction de courant de l'écoulement laminaire d'un fluide le long d'une plaque plane est obtenue en résolvant le problème de Blasius :

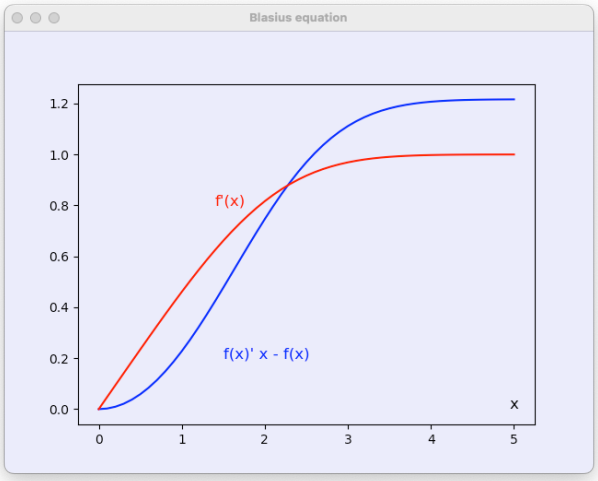
$$\begin{cases} f'''(x) + f(x)f''(x) &= 0 \\ \\ f(0) &= 0 \\ f'(0) &= 0 \\ \lim_{x\to\infty} f'(x) &= 1 \end{cases}$$

La valeur de $f'(x)$ représente la composante tangentielle de la vitesse qui varie de manière monotone de zéro sur la plaque ($x = 0$) à une valeur unitaire à une très grande distance de celle-ci. La valeur de $f'(x)x - f(x)$ est proportionnelle à la composante normale de la vitesse. Il s'agit d'une *solution de similitude* qui fournit les profils de vitesses pour n'importe quelle hauteur si on adapte l'échelle des abscisses de manière adéquate.

Pour résoudre ce problème, nous allons utiliser de manière conjointe la méthode du tir avec une méthode d'intégration numérique d'équations différentielles ordinaires. La méthode du tir consiste à estimer de manière successive une condition initiale $f''(0) = \alpha$ afin de satisfaire approximativement la condition à l'infini :

$$f'(5) \approx \lim_{x\to\infty} f'(x) = 1$$

En effet, on a observé numériquement que la solution tend à devenir constante aux environs de $x = 5$. Pratiquement, on recherchera donc la valeur de α telle que $f'(5) = 1$ lorsqu'on effectue l'intégration numérique de notre système d'équations différentielles ordinaires : on peut -par exemple- utiliser la méthode usuelle de Runge-Kutta d'ordre quatre pour effectuer cette intégration. Les estimations successives α_i des valeurs de α seront obtenues en utilisant la méthode de la bissection en prenant par exemple un intervalle de départ $[0, 1]$. Une telle manière de procéder permet d'obtenir la solution de similitude de Blasius :



Comme programmer l'intégration numérique par la méthode de Runge-Kutta d'ordre quatre n'a plus de secret pour vous, on vous a fourni gracieusement la fonction `python` qui effectue cette opération. Votre mission consistera à implémenter la méthode du tir et d'écrire l'équation de Blasius d'ordre trois sous la forme de 3 équations différentielles ordinaires d'ordre un.

Plus précisément, on vous demande de :

1. Ecrire l'équation différentielle d'ordre trois sous la forme d'un système de trois équations différentielles ordinaires d'ordre un dont les trois fonctions inconnues (u, v, w) sont $f(x)$, $f'(x)$ et $f''(x)$.

2. Ecrire la fonction `[dudt,dvdt,dwdt]= f(u)` décrivant ce système d'équations différentielles ordinaires. Le tableau `u` contient les 3 composantes de $\mathbf{u}$ à une abscisses x et la fonction renvoie un tableau avec les 3 composantes de la dérivée à cette abscisse :

$$\mathbf{u}' = \mathbf{f}(\mathbf{u})$$

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_u(u,v,w) \\ f_v(u,v,w) \\ f_w(u,v,w) \end{bmatrix}$$

3. Ecrire une fonction `alpha,message = blasius(delta,nmax,tol,h,integrator)` qui fournit la valeur `alpha` qu'il faut imposer comme condition initiale $f''(0) = \alpha$ afin d'obtenir $f'(5) = 1$. La fonction permettant d'effectuer l'intégration numérique est fournie par l'argument `integrator` et la pas à utiliser pour cette fonction est donnée par `h`. Les autres arguments sont liés à la méthode de bissection : il s'agit de la précision absolue requise `tol` pour α, du nombre maximal d'itérations `nmax` que l'on pourra exécuter et de l'intervalle initial de recherche `delta = [a,b]`.

   Ensuite, il s'agit aussi de fournir un message de diagnostic `message` à l'issue de l'exécution du programme. Trois cas doivent être envisagés :

   - Tout se passe correctement et on obtient une valeur de α comme espéré.
   - L'intervalle de départ n'est pas adéquat pour démarrer la méthode de bissection : en d'autres mots, les valeurs obtenues pour $f'_a(5)$ et $f'_b(5)$ n'encadrent pas la valeur unitaire requise. Dans ce cas, la fonction renverra une valeur nulle pour α.
   - Il n'a pas été possible d'obtenir la convergence avec un nombre d'itérations inférieur à `nmax`. Dans ce cas, la fonction fournira la dernière estimation obtenue pour α.

   Attention, il faut exactement fournir les messages fournis dans le canevas du code car c'est cela qui sera testé lors de l'évaluation automatique de votre programme !

   Finalement, il faut bien noter que votre fonction doit pouvoir fonctionner avec n'importe intégrateur numérique par l'utilisateur et fournir la réponse correspondant à cette intégrateur ! Il ne faut donc pas ré-implémenter ou modifier ou adapter ou tenter d'améliorer la méthode de Runge-Kutta dans votre solution pour tenter d'être plus rapide... Attention, on effectuera la correction en utilisant d'autres intégrateurs et il est assez facile d'imaginer que l'on obtiendra pas la même valeur de α en utilisant par exemple une méthode moins précise telle qu'Euler explicite : faites vous-même le test si vous n'êtes pas convaincu :-)

4. Comme d'habitude, pour tester votre programme, on vous a fourni un tout petit programme simple `blasiusTest.py`.

```
from numpy import *

def integrator(Xstart,Ustart,Xend,h,f):
  imax = int((Xend-Xstart)/h)
  X = Xstart + arange(imax+1)*h
  U = zeros((imax+1,3)); U[0,:] = Ustart
  for i in range(imax):
    K1 = f(U[i,:]       )
    K2 = f(U[i,:]+K1*h/2)
    K3 = f(U[i,:]+K2*h/2)
    K4 = f(U[i,:]+K3*h  )
    U[i+1,:] = U[i,:] + h*(K1+2*K2+2*K3+K4)/6
  return X,U

h   = 0.1; tol = 1e-7; nmax = 40
a,message = blasius([0,1],nmax,tol,h,integrator)
X,U = integrator(0,[0,0,a],5,h,f)
print(" === Resquested f''(0) = %.4f === %s" % (a,message))
print(" === Obtained final value for f'(-1) = %.4f " % U[-1,1])

from matplotlib import pyplot as plt

fig = plt.figure("Blasius equation")
plt.plot(X,U[:,1]*X - U[:,0],'-b',X,U[:,1],'-r')
plt.text(1.4,0.8,"f'(x)",color='red',fontsize=12)
plt.text(1.5,0.2,"f(x)' x - f(x)",color='blue',fontsize=12)
plt.text(4.95,0.0,"x",color='black',fontsize=12)
plt.show()
```

5. Si votre fonction est bien correcte, vous devriez obtenir la figure de l'énoncé :-)

6. Lers deux fonctions (avec les éventuelles sous-fonctions que vous auriez créées) seront soumises sur `zouLab` sans y adjoindre le programme de test fourni ! Ce travail est individuel et sera évalué.

   Pour rappel, toutes vos soumissions seront systématiquement analysées par un logiciel anti-plagiat. Faites vraiment votre programme seul...