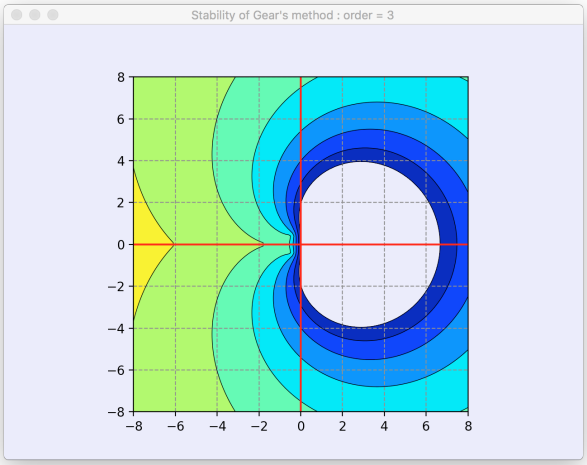
Python for dummies : problème 7
Zone de stabilité des méthodes BDF
(ou méthodes de Gear :-)



On peut montrer que la région de stabilité de ces méthodes *contient l'entièreté de l'axe réel négatif* pour un ordre $n \leq 6$. Ces méthodes semblent donc convenir particulièrement bien à la résolution de problèmes scalaires raides. Une méthode de Gear d'ordre n s'écrit sous la forme :

$$U_{i+1} = \sum_{j=1}^{n} \alpha_j U_{i-j} + \alpha_0 \; hF_{i+1}$$

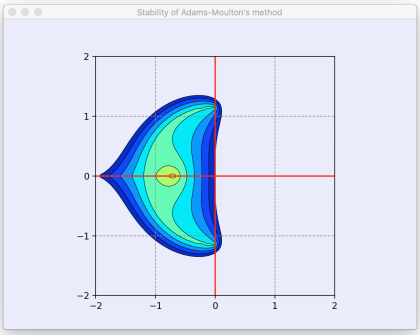
Il s'agit de méthodes implicites et donc le calcul des zones de stabilité va nécessiter la résolution numérique de polynômes de stabilité. On se propose -ici- de calculer ces fameuses zones de stabilité.

Plus précisément, les méthodes de Gear d'ordre un à quatre sont définies par :

$$\begin{aligned}
U_{i+1} &= U_i + h \; F_{i+1} \\
U_{i+1} &= \frac{1}{3}\left(-U_{i-1} + 4U_i\right) + \frac{2h}{3} \; F_{i+1} \\
U_{i+1} &= \frac{1}{11}\left(2U_{i-2} - 9U_{i-1} + 18U_i\right) + \frac{6h}{11} \; F_{i+1} \\
U_{i+1} &= \frac{1}{25}\left(-3U_{i-3} + 16U_{i-2} - 36U_{i-1} + 48U_i\right) + \frac{12h}{25} \; F_{i+1}
\end{aligned}$$

On observe -au passage- que la méthode de Gear d'ordre un n'est rien d'autre que la méthode d'Euler implicite. A titre d'exemple, nous avions obtenu la zone de stabilité d'une méthode implicite d'Adams-Moulton à pas multiples avec ce petit programme dans l'exercice 60.

$$\begin{aligned}
P_{i+1} &= U_i + \frac{h}{2}\left(-f(X_{i-1}, U_{i-1}) + 3f(X_i, U_i)\right) \\
U_{i+1} &= U_i + \frac{h}{2}\left(f(X_i, U_i) + f(X_{i+1}, P_{i+1})\right)
\end{aligned}$$




```
s = linspace(-2,2,100)
x,y = meshgrid(s,s)
z = x + 1j*y
b = 0.5 + z/2 + 3*z*z/8
c = z*z/4.0
f1 = abs(b - sqrt(b*b - c))
f2 = abs(b + sqrt(b*b - c))
gain = maximum(f1,f2)
plt.contourf(x,y,gain,arange(0,1.1,0.1))
```

Il faudra juste généraliser cette démarche pour les méthodes de Gear. Obtenir les racines d'un polynômes est très simple en faisant appel à la fonction `roots`[1], il ne faut donc pas reprogrammer cette fonction évidemment !

Plus précisément, on vous demande de :

1. Calculer les coefficients des méthodes de Gear d'ordre cinq et six : cela peut à nouveau se faire avec un crayon ou un bic, avant d'aller vous précipiter sur votre ordinateur !

2. Ecrire une fonction `gain,coeff = stabilityGear(x,y,order)` qui fournit le module du facteur d'amplification pour tout point du plan complexe $h\lambda$ dont on fournit les valeurs réelles et imaginaires dans les tableaux `x` et `y` de même taille. L'entier `n` est l'ordre de la méthode de Gear considérée. On se limitera aux cas $0 < n < 7$.

   Le tableau `gain` aura les dimensions des tableaux `x` et `y`. Le tableau `coeff` contiendra les coefficients α_i de la méthode. A titre d'exemple, pour la méthode de Gear d'ordre trois, on devrait obtenir le vecteur suivant :

   ```
   [6/11 18/11 -9/11 2/11]
   ```

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

   ```
   import numpy as np

   order = 3
   n = 100
   x,y = np.meshgrid(np.linspace(-8,8,n),np.linspace(-8,8,n))
   gain,coeff = stabilityGear(x,y,order)

   import fractions
   np.set_printoptions(formatter={'all':lambda x: str(fractions.Fraction(x).limit_denominator())})
   print("==== Coefficients of Gear's method for order = %d ========== " % order)
   print("     ",end='')
   print(coeff)

   import matplotlib.pyplot as plt
   plt.figure("Stability of Gear's method : order = %d" % order)
   plt.contourf(x,y,gain,np.arange(0,1.1,0.1),cmap=plt.cm.jet_r)
   plt.contour(x,y,gain,np.arange(0,1.1,0.1),colors='black',linewidths=0.5)
   ax = plt.gca()
   ax.axhline(y=0,color='r')
   ax.axvline(x=0,color='r')
   ax.yaxis.grid(color='gray',linestyle='dashed')
   ax.xaxis.grid(color='gray',linestyle='dashed')
   ax.set_aspect('equal', 'box')
   plt.show()
   ```

   Pour les étudiants curieux, bien observer l'utilisation du module `fractions` qui permet d'imprimer des variables en virgule flottante sous la forme approchée d'une fraction :-) L'obtention de la figure avec la librairie `matplotlib` est un peu technique et fastidieuse, si on souhaite paramétrer finement l'apparence du résultat final. Oui, j'aime pas trop les paramètres par défaut de la libraire :-(

4. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera soumise via le site web du cours.

5. Bonus pour les fous : pour ceux qui ne savent vraiment pas à quoi passer leurs longues soirées ensoleillées de printemps, on peut généraliser la fonction afin qu'elle fournisse le résultat pour n'importe quel n positif (sans exagérer toutefois, car le temps de calcul risque de devenir un peu prohibitif à un certain moment !)

[1] Voir https://docs.scipy.org/doc/numpy/reference/generated/numpy.roots.html