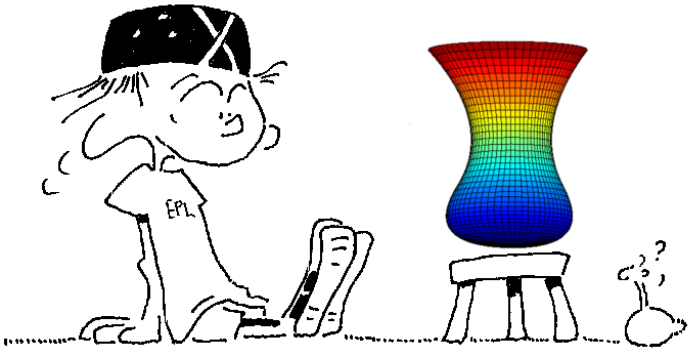
Ecole Polytechnique de Louvain



INFORMATIQUE
ET
METHODES NUMERIQUES
...ou les aspects facétieux du calcul sur un ordinateur

V. Legat

Enoncés des séances d'exercices pour les cours LICAR1104
Année académique 2019-2020 (version 4.7 : 18-09-2019)

python™

Les ordinateurs sont comme les Dieux de l'Ancien testament : beaucoup de règles et aucune pitié.
(Joseph Campbell)

Séance 1

Interpolation

Trouver $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\underbrace{\sum_{j=0}^{n} a_j \ \phi_j(X_i)}_{u^h(X_i)} = U_i \qquad\qquad i = 0, 1, \ldots, n.$$

1 On cherche un polynôme $u^h(x) = a + bx$ qui passe par les points (X_0, U_0) et (X_1, U_1).

1. Exprimer les coefficients a et b en termes des coordonnées des deux points.
2. Utiliser ce résultat pour approcher la fonction $u(x) = \sqrt{x}$ à l'aide de ses valeurs en $X_0 = 0$ et $X_1 = 0.25$.
3. Donner la valeur de l'erreur d'interpolation en $x = 1/9$.

2 On cherche un polynôme $u^h(x) = a + bx + cx^2$ qui passe par les points $(3, 2), (1, 1)$ et $(2, 1)$.

1. Donner le système linéaire que doit satisfaire le vecteur composé des trois coefficients a, b et c.
2. Ce système possède-t-il zéro, une ou plusieurs solutions ?

Latitude	
65	3.10
55	3.22
45	3.30
35	3.32
25	3.17
15	3.07
5	3.02
-5	3.02
-15	3.12
-25	3.20
-35	3.35
-45	3.37
-55	3.25

Exemple : climatologie

Nous disposons de mesures expérimentales récentes (Philosophical Magazine 41,237, 1896) sur la variation de la température annuelle moyenne à différentes latitudes en fonction d'une augmentation de 50% de la concentration d'acide carbonique dans l'atmosphère au niveau du sol.
Nous allons rechercher un polynôme qui interpole les données pour les latitudes $65, 35, 5, -25, -55$ en utilisant les instructions suivantes de `python`

```
>>> from numpy import *
>>> X = [-55,-25,5,35,65]
>>> U = [3.25,3.20,3.02,3.32,3.10]
>>> a = polyfit(X,U,4)
>>> print(a)
[ -8.281893e-08 4.526749e-07 3.468364e-04 -3.775720e-04 3.013212e+00]
```

On peut obtenir le graphe du polynôme de la manière suivante :

```
>>> import matplotlib.pyplot as plt
>>> x = linspace(X[0],X[-1],100)
>>> uh = polyval(a,x)
>>> plt.plot(x,uh,'-b',X,U,'or')
>>> plt.show()
```

Afin d'obtenir une jolie courbe, nous évaluons le polynôme en 100 points équidistants sur l'intervalle $[-55, 65]$. Observons que l'instruction `X(end)` permet d'obtenir la dernière composante d'un vecteur sans en spécifier la longueur. Notons aussi que les plots de `python` sont toujours construits comme une interpolation linéaire par morceaux entre les points donnés comme arguments.
(Exemple tiré de Quarteroni et al.)

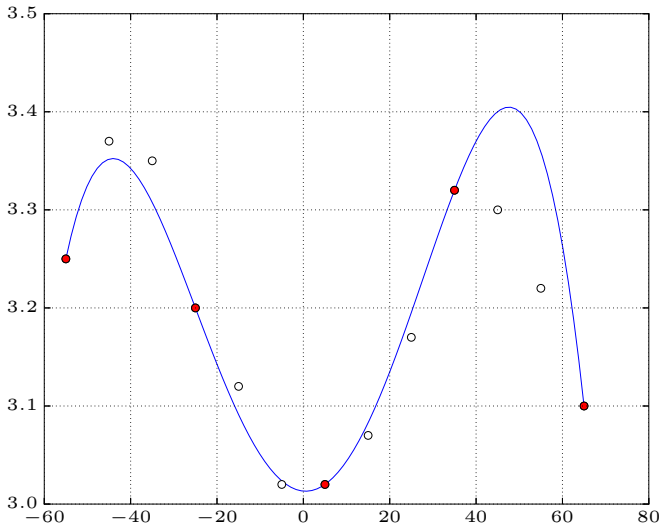


Figure 1.1: Interpolation polynomiale de degré 4 pour les variations de température en termes de latitude : on peut observer l'accord qualitatif entre la courbe et toutes les données expérimentales. Les cercles pleins ont été utilisés pour effectuer l'interpolation, tandis que les cercles vides correspondent aux valeurs dont on n'a pas fait usage.

3 On suppose que l'on connaît dans un vecteur de taille $2n+1$ les valeurs d'une fonction continue $u(x)$ pour les abscisses entières allant de $-n$ à n. En d'autres mots, on suppose connues les valeurs

$$U_k \qquad\qquad k \;=\; 0,\; \pm 1,\; \pm 2, \ldots,\; \pm n$$

Pour une valeur $t \in \mathbb{R}$, on définit $u^h(t)$ comme la valeur du polynôme de degré trois qui interpole u en passant par les quatre points entiers les plus proches. On vous demande d'écrire une fonction `python` qui calcule $u^h(t)$ à partir des arguments t, n et U_k.

4 Il n'existe pas de polynôme de degré n dont la courbe passe par $n+2$ points donnés. Est-ce que cette affirmation est exacte ? Commenter et justifier votre réponse.

5 Considérons une distribution uniforme d'abscisses $X_i = X_{i-1} + h$ pour $i = 1, \ldots, n$ avec un pas $h > 0$ et un point X_0 donnés. Démontrer que pour tout $x \in [X_0, X_n]$, on a l'inégalité suivante :

$$\left| \prod_{i=0}^{n} (x - X_i) \right| \leq n! \frac{h^{n+1}}{4}$$

De ce résultat, on pourra ainsi déduire que l'erreur d'interpolation d'une fonction u par le polynôme de degré n pour les abscisses X_i peut être bornée par la relation suivante.

$$\boxed{\left| e^h(x) \right| \;\leq\; \frac{\max_{x \in [X_0, X_n]} \left| u^{(n+1)}(x) \right|}{4(n+1)} h^{n+1}}$$

Exemple : Runge

Interpolons la fonction $u(x) = 1/(1+x^2)$ à des abscisses équidistantes sur l'intervalle $[-5,5]$. Dans ce cas, la limite du maximum de la valeur absolue de l'erreur d'interpolation tend vers l'infini, lorsque $n \to \infty$. Ceci est dû au fait que la vitesse d'accroissement de l'ordre de grandeur du terme $\max_{x\in[-5,5]} \left|u^{(n+1)}(x)\right|$ dépasse la vitesse avec laquelle le terme $h^{n+1}/4(n+1)$ tend vers zéro, comme l'illustre la Figure ci-dessous.
Nous allons rechercher le polynôme qui interpole les données pour les abscisses de Chebyshev en utilisant les instructions suivantes de `python`

```
>>> from numpy import *
>>> import matplotlib.pyplot as plt
>>> n = 10
>>> f = lambda x : 1/(1 + x*x)
>>> Xcheby = 5*cos(pi*(2.0*arange(0,n+1)+1)/((n+1)*2))
>>> Ucheby = f(Xcheby)
>>> x  = linspace(Xcheby[0],Xcheby[-1],1000)
>>> u  = f(x)
>>> uh = polyval(polyfit(Xcheby,Ucheby,n),x)
>>> plt.plot(x,uh,'-b', x,u,'--b')
>>> plt.show()
```

Il est possible d'observer la convergence de l'erreur en calculant le maximum de l'erreur d'interpolation pour diverses valeurs de n

```
>>> eh = max(abs(u - uh))
>>> print(eh)
0.109151988704
```

n	$\max_{x\in[-5,5]} \lvert e^h(x)\rvert$
5	0.555888991775
10	0.109151988704
20	0.0153305602451
40	0.00038859656587

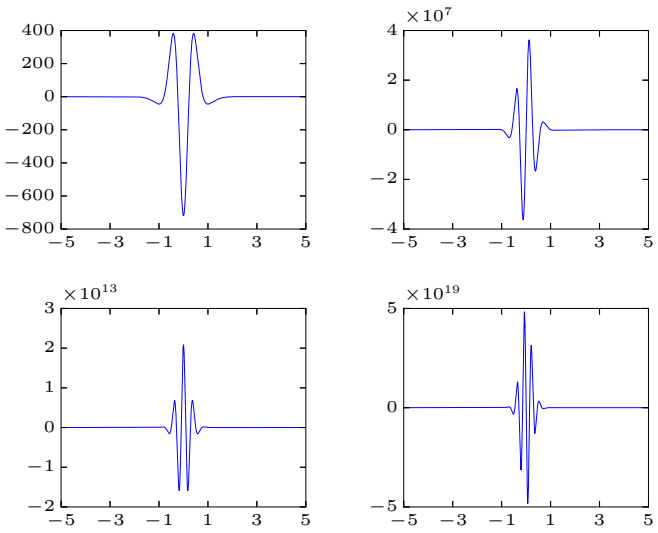


Figure 1.2: Dérivées d'ordre 6, 11, 16 et 21 de la fonction de Runge. On comprend mieux pourquoi, la borne de l'erreur d'interpolation explose lorsque $n \to \infty$. Toutefois, le choix des abscisses de Chebychev permet d'obtenir la convergence même si la borne de l'estimation d'erreur continue à exploser...

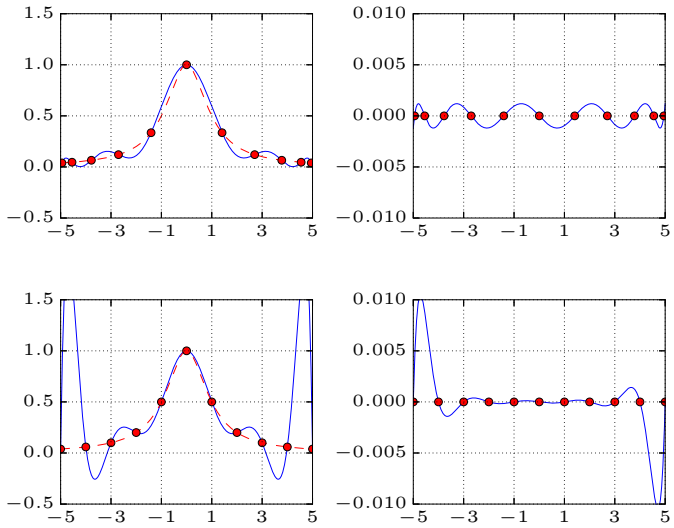


Figure 1.3: Comparaison entre des interpolations polynomiales de degré 10 faites avec des abscisses équidistantes ou avec des abscisses de Chebyshev. On a également représenté le diagramme du terme $(x-X_0)(x-X_1)\dots(x-X_n)/(n+1)!$. Le choix des abscisses de Chebychev permet de réduire l'ampleur de ce facteur du terme d'erreur et d'obtenir la convergence de l'approximation polynomiale.

6 La probabilité qu'une fonction aléatoire X soit inférieure ou égale à x est notée $P(X \leq x)$.

Si nous considérons que la fonction aléatoire suit une loi normale, cette probabilité est donnée par la fonction :

$$P(X \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp \frac{-t^2}{2} dt$$

x	$P(X \leq x)$
1.0	0.8413447
1.1	0.8643339
1.2	0.8849303
1.3	0.9031995
1.4	0.9192433

Comme la fonction $\exp \frac{-t^2}{2}$ ne possède pas de primitive, on calcule, par des méthodes numériques (que l'on verra plus tard :-), cette probabilité pour différentes valeurs de x et l'on conserve les résultats dans une table.

A l'aide de cette table, obtenir $P(X \leq 1.05)$ avec une erreur absolue inférieure à 0.5×10^{-5}.

7 On cherche à modéliser la friction de l'air sur un parachutiste : on suppose que la force de friction exercée est proportionnelle à la vitesse du parachutiste. Le facteur de proportionnalité est appelé coefficient de friction. Avant l'ouverture du parachute, le coefficient de friction peut être considéré comme constant et égal à 0.2. Une fois le parachute complètement ouvert, ce coefficient redevient constant et vaut 2. Le parachute prend environ 3 secondes pour s'ouvrir complètement.

1. Donner les unités du coefficient de friction.
2. Proposer un modèle polynomial cubique qui permette d'obtenir une évolution différentiable du coefficient de friction pendant l'ouverture du parachute, si on suppose que le parachute s'ouvre à l'instant $t = 0$.

Séance 2

Interpolation par morceaux

Trouver $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\sum_{j=0}^{n} \underbrace{a_j \ \phi_j(X_i)}_{u^h(X_i)} = U_i \qquad\qquad i = 0, 1, \ldots, n.$$

Latitude	
65	3.10
55	3.22
45	3.30
35	3.32
25	3.17
15	3.07
5	3.02
-5	3.02
-15	3.12
-25	3.20
-35	3.35
-45	3.37
-55	3.25

Exemple : courbe spline cubique

Reconsidérons les données de notre exemple de climatologie. Nous allons rechercher une courbe spline cubique (une interpolation polynomiale cubique par morceaux) et la comparer à une interpolation polynomiale de Lagrange. On utilise les instructions suivantes de `python`

```
>>> from numpy import *
>>> from scipy.interpolate import CubicSpline as spline
>>> X = arange(-55,75,10)
>>> U = [3.25,3.37,3.35,3.20,3.12,3.02,3.02,
...      3.07,3.17,3.32,3.30,3.22,3.10]
>>> x = linspace(X[0],X[-1],100)
>>> uh = spline(X,U)
```

8 On cherche à approcher la courbe représentant un quart de cercle à partir des points

$$X_k = \sin(\frac{k\pi}{6}), \qquad Y_k = \cos(\frac{k\pi}{6}),$$

avec $k = 0, \ldots, 3$. On utilisera uniquement la fonction `CubicSplines` de `scipy.interpolate`.

1. Dessiner la courbe spline $y^h(x)$ passant par les quatre points.
2. Considérer la représentation paramétrique du cercle et calculer les deux courbes splines $x^h(t)$ et $y^h(t)$ pour les données suivantes

   $$X_k = \sin(\frac{k\pi}{6}), \qquad Y_k = \cos(\frac{k\pi}{6}), \qquad T_k = \frac{k\pi}{6}$$

   avec $k = 0, \ldots, 3$. Comparer les deux interpolations de la courbe dans le plan (x, y).
3. Dessiner un cercle complet en utilisant une représentation paramétrique.

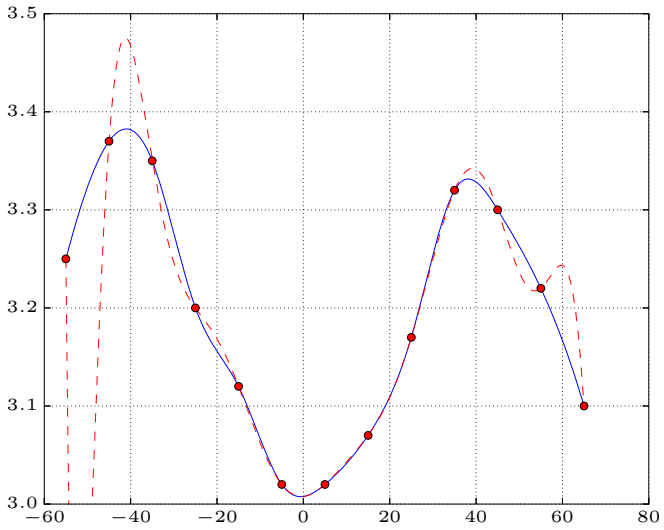


Figure 2.1: Comparaison entre l'interpolation polynomiale de Lagrange et la courbe composée de splines cubiques (ou courbe spline cubique). La courbe spline semble plus plausible que le polynome de Lagrange.

9 Calculer a priori une borne supérieure de la valeur absolue de l'erreur d'interpolation pour les fonctions et les abscisses suivantes :

$$u(x) = \cosh(x), \qquad X_k = -1 + \frac{k}{2}, \qquad k = 0, \ldots, 4,$$

$$u(x) = \sinh(x), \qquad X_k = -1 + \frac{k}{2}, \qquad k = 0, \ldots, 4,$$

$$u(x) = \cos(x) + \sin(x), \qquad X_k = -\frac{\pi}{2} + \frac{\pi k}{4}, \qquad k = 0, \ldots, 4.$$

10 Nous disposons de données statistiques sur la durée de vie moyenne des citoyens de l'Europe de l'Est. Calculer avec `python`, l'interpolation polynomiale de degré trois et la courbe spline cubique.

	Durée de vie
1975	70.2
1980	70.2
1985	70.3
1990	71.2

Utiliser ensuite les interpolations afin d'extrapoler la durée de vie moyenne en 1970 et 1995. Comparez le résultat obtenu en 1970 sachant que la durée de vie observée était 69.6 années en cette année-là. Pouvez-vous sur base de cette nouvelle information, estimer la précision de l'extrapolation effectuée pour 1995.

Commenter vos résultats en comparant ceux obtenus par l'interpolation polynomiale de Lagrange et la courbe spline cubique.

11 Evaluer avec `python` la fonction $u(x) = \sin(2\pi x)$ sur 21 points équidistants sur l'intervalle $[-1, 1]$. Calculer le polynôme d'interpolation de Lagrange et la courbe spline cubique. Effectuer le même calcul en utilisant les données perturbées comme suit :

$$U_k = \sin(2\pi X_k) + (-1)^{k+1}\ 10^{-4}$$

Qu'observez-vous ? Pouvez-vous expliquer, de manière intuitive, ce que vous observez ?

12 On connaît la température T_i en quatre sommets (X_i, Y_i) d'un carré centré à l'origine de côté deux.

X_1	=	-1	Y_1	=	-1
X_2	=	-1	Y_2	=	1
X_3	=	1	Y_3	=	1
X_4	=	1	Y_4	=	-1

1. Proposez une interpolation bidimensionnelle qui soit une généralisation de l'interpolation unidimensionnelle par morceaux. En d'autres mots, on vous demande de fournir une interpolation $t^h(x, y)$ de la température à partir des valeurs aux sommets.
2. Ecrivez un programme `python` qui fournit la valeur de la température pour un point (x, y) à partir des données (T_i) pour les quatre sommets.
3. Que se passe-t-il si (x, y) est hors du carré ? Détectez ce cas dans votre programme en émettant un avertissement à l'utilisateur de votre fonction.

13 Refaites par vous-même la fonction `CubicSpline` de `scipy.interpolate` et vérifiez l'exactitude de votre programme en comparant vos résultats avec ceux obtenus par la fonction originale du logiciel.

En particulier, essayez d'identifier des cas particuliers où votre programme est moins robuste que l'implémentation de `python`...

Séance 3

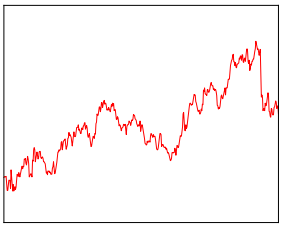
Approximation

Trouver $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\underbrace{\sum_{i=0}^{m} \left(U_i - \sum_{j=0}^{n} \phi_j(X_i)\ a_j \right)^2}_{J(a_0, \ldots, a_n)} \quad \text{soit minimal.}$$

14 On cherche un polynôme $u^h(x) = a + bx$ qui approxime, au sens des moindres carrés, les points (X_0, U_0), (X_1, U_1) et (X_2, U_2).

1. Exprimer les coefficients a et b en termes des coordonnées des trois points.
2. Utiliser ce résultat pour approcher la fonction $u(x) = x^2$ à l'aide de ses valeurs en $X_0 = 0$, $X_1 = 1/3$ et $X_2 = 1$.
3. Donner la valeur de l'erreur d'approximation en $x = 1/3$.



Exemple : cours d'une action

Nous disposons de 500 valeurs du cours de l'action Coca-Cola du 30 mai 2003 à 13h50 jusqu'au 18 juin à 12h20 à la bourse de New-York. La courbe est obtenue en joignant avec une droite les valeurs reportées toutes les dix minutes. Comme la bourse est ouverte tous les jours de la semaine de 9h30 à 15h50, on dispose d'un ensemble de 500 valeurs. Cette courbe est une interpolation linéaire par morceaux des données et suppose implicitement que le cours change linéairement pendant les dix minutes qui séparent deux valeurs.
La question que se posent tous les analystes est de savoir s'il est possible de prédire le cours de l'action peu de temps après la dernière valeur disponible. Pour tenter d'y répondre, nous allons calculer les polynômes de degré 1, 4, 6 et 8 qui approximent au sens des moindres carrés les données disponibles. Nous allons rechercher le polynôme de degré quatre qui approxime, au sens des moindres carrés, les données avec les instructions suivantes de `python`

```
>>> from numpy  import *
>>> from pandas import read_csv
>>> csvData = read_csv('stockCocaCola.csv')
>>> X  = list(csvData["time"])
>>> U  = list(csvData["value"])
>>> x  = linspace(X[0],X[-1]+100,100)
>>> uh = polyval(polyfit(X,U,4),x)
>>> from matplotlib import pyplot as plt
>>> plt.plot(x,uh,'b-',X,U,'r-')
>>> plt.show()
```

On observe immédiatement qu'on a utilisé les mêmes instructions que pour l'interpolation polynomiale. En d'autres mots l'interpolation polynomiale par $n+1$ points n'est rien d'autre qu'une approximation polynomiale de degré n qui approxime parfaitement les données...

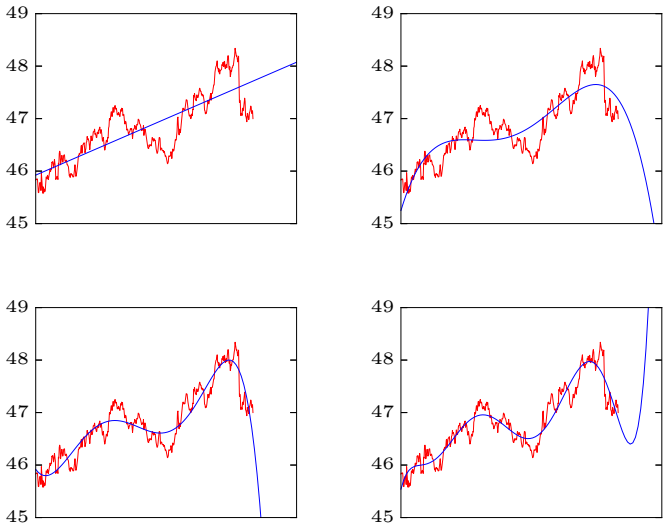


Figure 3.1: Approximations polynomiales de degré 1,4,6 et 8 pour le cours de l'action Coca-Cola. On observe que l'approximation de degré 8 semble reproduire relativement bien l'évolution du cours sur la période considérée et semble aussi suggérer une possible remontée du cours.

15 On cherche une droite $u^h(x) = a + bx$ qui approxime, au sens des moindres carrés, les points $(X_0, U_0), (X_1, U_1)$ et (X_2, U_2). Refaites le raisonnement permettant d'obtenir le système linéaire que doivent satisfaire a et b.

1. Ecrire la fonction $J(a, b)$ qui vaut la somme des carrés des résidus.
2. Ecrire le vecteur gradient de J.
3. Ecrire la matrice hessienne de J.
4. Ecrire les conditions à imposer au vecteur gradient pour avoir un minimum de J en (a, b).
5. Ecrire les conditions à imposer à la matrice hessienne pour avoir un minimum de J en (a, b).
6. Est-ce que ce système possède zéro, une ou plusieurs solutions ?

16 Démontrer que la droite de régression linéaire pour des données $\{(X_i, U_i), i = 0, \ldots, m\}$ passe par un point dont l'abscisse est la moyenne des $\{X_i\}$ et l'ordonnée est la moyenne des $\{U_i\}$.

17 Démontrer que l'approximation polynomiale de degré n par $n + 1$ points distincts au sens des moindres carrés coïncide avec l'interpolation polynomiale pour les mêmes points.

18 Alphonse considère l'intervalle ouvert $]0, 3[$ et la fonction suivante :

$$u(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

Il souhaite trouver l'approximation $u^h(x)$ de cette fonction au sens intégral des moindres carrés. Cette approximation sera une fonction linéaire par morceaux sur des intervalles définis par la suite d'abscisses $0 = X_0 < X_1 < X_2 \ldots X_{n-1} < X_n = 3$. Cette approximation sera ensuite construite en reliant sur chaque intervalle $[X_i, X_{i+1}]$, les points (X_i, a_i) et (X_{i+1}, a_{i+1}). Pour caractériser complètement cette approximation, il faut donc trouver les valeurs a_i.

De manière plus formelle, le problème d'Alphonse peut s'énoncer comme suit :

Trouver $(a_0, a_1, \ldots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\underbrace{\int_0^3 \Big(u(x) - u^h(x) \Big)^2 \, dx}_{J(a_0, a_1, \ldots, a_n)} \quad \text{soit minimal.}$$

où $a_i = u^h(X_i)$. Ecrire un programme `python` résolvant le problème d'Alphonse.

19 Paul trouve le programme d'Alphonse beaucoup trop compliqué. Il a calculé une interpolation continue linéaire par morceaux de la fonction en choisissant simplement $a_i = u(X_i)$ et en est fort satisfait. On définit une erreur globale comme suit

$$E^h = \int_0^3 \Big(u(x) - u^h(x) \Big)^2 \, dx$$

où $u^h(x)$ est respectivement l'approximation d'Alphonse ou l'interpolation de Paul. Sans programmer ni effectuer aucun calcul, répondez aux questions suivantes.

1. Pour le même nombre d'abscisses, est-ce que l'erreur globale de Paul sera toujours supérieure ou inférieure à l'erreur globale d'Alphonse ? Justifiez de manière rigoureuse votre réponse.
2. Comment va évoluer la différence entre l'erreur de Paul et celle d'Alphonse lorsqu'on considère des abscisses uniformément réparties et que leur nombre tend vers l'infini ?

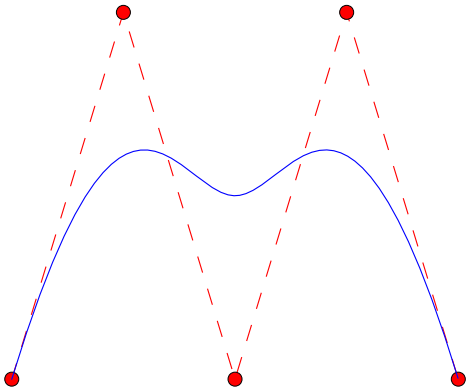
20 Démontrer que la fonctionnelle définie par

$$< f, g >= \int_a^b f(t) g(t) \, dt$$

est un produit scalaire pour l'espace des fonctions continues sur l'intervalle $[a, b]$.

Séance 4

Courbes de Bézier, B-splines et NURBS



Exemple : B-splines

Pur obtenir une courbe, nous implémentons la définition récursive des fonctions B-splines et nous pouvons écrire simplement le programme qui suit :

```
def b(t,T,i,p):
  if p == 0:
    return (T[i] <= t)*(t < T[i+1])
  else:
    if T[i+p ] == T[i] :
      u = 0.0
    else :
      u = (t-T[i])/(T[i+p]-T[i])*b(t,T,i,p-1)
    if T[i+p+1] == T[i+1] :
      u += 0.0
    else :
      u += (T[i+p+1]-t)/(T[i+p+1]-T[i+1])*b(t,T,i+1,p-1)
    return u

T = [0,0,0,0,1,2,2,2,2]
X = [0,1,2,3,4]
Y = [0,3,0,3,0]
p = 3
n = len(T)-1
t = arange(T[p],T[n-p],0.001)
B = zeros((n-p,len(t)))

for i in range(0,n-p):
  B[i,:] = b(t,T,i,p)

x = X @ B
y = Y @ B
plt.plot(X,Y,'--r',X,Y,'or',x,y,'-b')
```

Comme les notations du cours commencent à zéro et les vecteurs de `python` à zéro, la définition des indices dans les boucles se fait de manière très naturelle. Notons ensuite que la présence des points quadruples nécessite de respecter strictement la définition : il faut bien veiller à ne pas calculer la courbe en $t = 2$ et s'arrêter juste un peu avant : ce que fait exactement la fonction `arange` de `python`. De manière tout-à-fait naturelle, on obtient un code élégant et simple. Toutefois, il ne s'agit pas de l'implémentation la plus efficace. Dans les applications graphiques, il est plus indiqué d'utiliser l'algorithme de *de Boor* que nous ne présenterons pas dans cette courte introduction.

21 Démontrer les résultats des théorèmes 1.3 et 1.4 des notes de cours dans le cas particulier pour les fonctions $B_i^p(t)$ où $p = 3$ et les noeuds sont la suite des entiers $\mathbf{T} = \{0, 1, 2, 3, 4, 5...\}$.

22 Obtenir l'expression analytique des quatre polynômes de Bernstein de degré trois sur l'intervalle $[0, 1]$. Démontrer qu'ils satisfont bien le théorème 1.3 des notes de cours.

23 On considère la courbe $\mathcal{C}$ définie par l'équation :

$$\frac{x^2}{16} + \frac{y^2}{4} + \frac{xy}{8} = 1$$

On observe immédiatement que les points $A = (0, 2)$ et $B = (4, 0)$ appartiennent à cette courbe.

1. Démontrer que la courbe $\mathcal{C}$ est une ellipse.
2. Donner l'expression algébrique polynomiale des trois fonctions de base $B_0^2(t)$, $B_1^2(t)$ et $B_2^2(t)$ pour les noeuds définis par $\mathbf{T} = \{0, 0, 0, 1, 1, 1\}$.
3. Choisir trois points de contrôle et trois poids afin que la courbe NURBS ainsi définie avec $t \in [0, 1]$ et $\mathbf{T} = \{0, 0, 0, 1, 1, 1\}$ représente exactement le morceau de la courbe $\mathcal{C}$ entre A et B lorsqu'on parcourt celle-ci dans le sens horlogique.
4. Démontrer proprement et avec rigueur que les deux courbes coïncident parfaitement.

24 On considère $\mathbf{u}^h(t)$ la courbe construite avec les splines de bases cubiques pour les noeuds $\mathbf{T} = \{0, 0, 0, 0, 1, 1, 1, 1\}$ et quatre points de contrôle $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. Il s'agit d'une courbe de Bézier car les splines de base sont les polynômes de Bernstein.

Pour obtenir les coordonnées $\mathbf{u}^h(\xi)$ d'un point d'une courbe de Bézier de degré n définie par $n + 1$ points $\mathbf{P}_j$, Paul de Casteljau a proposé l'algorithme suivant :

> On fournit une suite de $n + 1$ points $(\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n)$,
> une abscisse $\xi \in [0, 1]$, et on calcule successivement
>
> $$\mathbf{P}_{i,j} = (1 - \xi)\ \mathbf{P}_{i-1,j-1} + \xi\ \mathbf{P}_{i,j-1} \qquad \begin{matrix} j = 1, \ldots, n \\ i = j, \ldots, n \end{matrix}$$
>
> avec $\mathbf{P}_{i,0} = \mathbf{P}_i$
>
> Et on obtient finalement : $\mathbf{u}^h(\xi) = \mathbf{P}_{n,n}$

Démontrer la validité de l'algorithme de Paul de Casteljau lorsque $n = 3$.

25 Reprenons la courbe $\mathbf{u}^h(t)$ construite avec les splines de base cubiques pour les noeuds $\mathbf{T} = \{0,0,0,0,1,1,1,1\}$ et les quatre points de contrôle $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$.

On désire diviser cette courbe $\mathbf{u}^h(t)$ où $t \in [0,1]$ en deux autres courbes de Bézier. La première partie de la courbe (lorsque $t \in [0, \frac{1}{2}]$) est définie par :

$$\mathbf{v}^h(\eta)$$

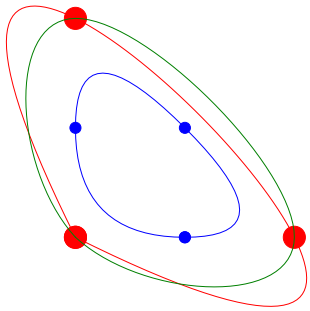
où $\eta \in [0,1]$ et $\eta = 2t$. La seconde partie de la courbe (lorsque $t \in [\frac{1}{2}, 1]$) est donnée par :

$$\mathbf{w}^h(\zeta)$$

où $\zeta \in [0,1]$ et $\zeta = 2t - 1$. Ces deux courbes s'appuient sur 4 points de contrôle chacune, respectivement $\mathbf{V}_0$, $\mathbf{V}_1$, $\mathbf{V}_2$ et $\mathbf{V}_3$ pour la première, et $\mathbf{W}_0$, $\mathbf{W}_1$, $\mathbf{W}_2$ et $\mathbf{W}_3$ pour la seconde, les points $\mathbf{V}_3$ et $\mathbf{W}_0$ étant confondus.

Trouver les points de contrôle $\mathbf{V}_i$ en fonction de $\mathbf{P}_i$ afin que la courbe $\mathbf{v}^h(\eta)$ où $\eta \in [0,1]$ coïncide parfaitement avec la courbe $\mathbf{u}^h(t)$ où $t \in [0, \frac{1}{2}]$.

26 Il y a trois points que la trajectoire inconnue d'une voiture croise en 3 secondes. En $t = 0$, elle se trouvait au premier point, puis au second et au troisième en $t = 1$ et $t = 2$ respectivement. Et finalement, elle repasse au point initial en $t = 3$ et continuera à répéter indéfiniment la même trajectoire...



Nous allons estimer la représentation paramétrique C^0 de cette trajectoire avec une interpolation polynomiale, une approximation B-spline uniforme C^1 de degré deux et une interpolation C^2 avec des splines cubiques usuelles.

	t_i	(X_i, Y_i)
0	0.00	(0.00 , 0.00)
1	1.00	(1.00 , 0.00)
2	2.00	(0.00 , 1.00)
3	3.00	(0.00 , 0.00)

Plus précisément, on vous demande de :

1. Développer l'expression de l'interpolation polynomiale paramétrique : $(x_L(t), y_L(t))$.
   En déduire ensuite la position atteinte en $t = \frac{1}{2}$.
2. Représenter graphiquement les fonctions de base B-splines de degré deux associées aux noeuds $\mathbf{T} = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7\} = \{-2, -1, 0, 1, 2, 3, 4, 5\}$.
3. Développer l'expression de l'approximation B-spline uniforme $(x_B(t), y_B(t))$ lorsque $t \in [0, 1[$.
   A nouveau, en déduire la position atteinte en $t = \frac{1}{2}$.
4. Ecrire un programme `python` pour obtenir une interpolation C^2 avec des splines cubiques usuelles.

Séance 5

Intégration

$$\int_a^b u(x)\,dx \;=\; I \;\approx\; I^h \;=\; \sum_{i=0}^{m} w_i\, u(X_i)$$

27 Soit u une fonction continue sur l'intervalle $[-h, h]$ et u^h le polynôme de degré deux qui interpole u aux points $-h$, 0 et h. Exprimer

$$I^h = \int_{-h}^{h} u^h(t)dt$$

en fonction de $u(-h)$, $u(0)$ et $u(h)$. L'intégrale est ainsi approximée comme une somme pondérée des trois valeurs. Il s'agit donc de calculer les trois poids.

28 Vincent a fait usage de la *méthode dite des rectangles* pour obtenir une approximation de l'intégrale suivante :

$$I = \int_a^b f(x)\ dx$$

où la fonction $f(x) = 1/x$ et les bornes d'intégration sont $a = 2$ et $b = 7$. Cette méthode d'intégration consiste simplement à diviser l'intervalle d'intégration en n intervalles de même taille h et à additionner la superficie des rectangles dont la hauteur est donnée par la valeur de f au côté gauche de chaque intervalle. Vincent a obtenu $I_{100} = 1,2617$ et $I_{200} = 1,2572$ pour $n = 100$ et $n = 200$ respectivement. :

1. Fournir une expression, en fonction de n, d'une bonne borne supérieure B_n de la valeur absolue de l'erreur locale d'intégration commise sur un seul intervalle quelconque de taille h.
2. En déduire l'ordre de précision de la méthode des rectangles.
3. Prédire un nombre n afin que l'erreur $|I - I_n|$ soit inférieure à une tolérance de 10^{-3}. Sans effectuer le calcul de I_n, est-ce que l'utilisation de n intervalles permettra toujours le respect strict ou approximatif de la tolérance ? Justifier votre réponse.
4. Calculer une combinaison linéaire élémentaire de I_{100} et I_{200} qui serait une bien meilleure approximation de I. Justifier votre réponse.

29 Calculer l'intégrale : $\displaystyle\int_0^1 e^x\,dx$

1. par la méthode des rectangles avec 4, 10 et 100 intervalles,
2. par la méthode des trapèzes avec 4, 10 et 100 intervalles,
3. par la méthode de Simpson avec 5 points, 11 points, 101 points,
4. par la méthode de Gauss-Legendre avec 2 points, 5 points,
5. par la méthode de Romberg.

30 On connaît quelques points d'une fonction $u(x)$ dans une table.

x	$u(x)$
0.00	1.570796327
0.25	1.318116072
0.50	1.047197551
0.75	0.722734248
1.00	0.000000000

Calculer, par la méthode de Romberg, l'intégrale de 0 à 1 de cette fonction.

31 La fonction de Bessel d'ordre 1, notée $J_1(x)$, peut être calculée par la relation

$$J_1(x) = \frac{1}{\pi} \int_0^{\pi} \sin(x \sin\theta) \sin\theta \, d\theta$$

Calculer $J_1(2)$ en utilisant la méthode de Romberg.

32 Ecrire le polynôme d'interpolation de Lagrange passant par les deux points (X_0, Y_0) et (X_1, Y_1). Intégrer ensuite ce polynôme sur l'intervalle $[X_0, X_1]$. Quelle règle d'intégration obtient-on ?

33 Extrapoler $u(0)$ en utilisant au mieux les renseignements donnés dans le tableau de mesures.

x	$u(x)$
0.2	10
0.6	12
1.0	15
1.4	18
1.8	22

A priori, estimez-vous que la valeur $u(0)$ est supérieure ou inférieure à la valeur dix ? Pourquoi ?

Que répondre si l'on dispose de `python` ?

Que répondre si l'on ne dispose que d'une calculatrice non programmable permettant de réaliser uniquement les 4 opérations arithmétiques classiques ou si on doit exécuter tous les calculs à la main ?

34 A l'aide d'une méthode numérique, on a calculé $I = \int_0^{\pi/2} \sin x \, dx$ et on a trouvé :

- pour $h = 0.1$, $I = 1.001235$,
- pour $h = 0.2$, $I = 1.009872$,
- pour $h = 0.4$, $I = 1.078979$.

Compte tenu de la valeur exacte de I, quel est l'ordre de la méthode utilisée ?

35 Calculer numériquement l'intégrale $\int_0^1 \frac{1}{\sqrt{x}} \, dx$.

Séance 6

Dérivation

$$u'(0) \;=\; D \;\approx\; D^h \;=\; \sum_{i=0}^{m} w_i\, u(X_i)$$

36 Pour estimer la dérivée seconde à l'origine d'une fonction $u(x)$, Yves souhaite utiliser la formule :

$$u''(0) \;\approx\; \frac{(2U_0 + aU_{-h} + 4U_{-2h} + bU_{-3h})}{ch^2}$$

Malencontreusement, il a oublié de noter les valeurs des paramètres réels a, b et c.

1. Retrouver les valeurs des trois paramètres. Justifier votre réponse.
2. Donner l'ordre de précision de la formule obtenue et obtenir l'expression du terme d'erreur.
3. Calculer la valeur optimale de h afin de minimiser l'erreur totale, c'est-à-dire, les contributions conjointes de l'erreur de discrétisation et des erreurs d'arrondi dues au calcul en virgule flottante. Les valeurs de la fonction u sont calculées en double précision ($\epsilon = 10^{-16}$). En outre, nous supposons qu'en tout point, la valeur absolue de toutes les dérivées de u est toujours inférieure à 4.

37 La probabilité qu'une fonction aléatoire X soit inférieure ou égale à x est notée $P(X \leq x)$.

x	$P(X \leq x)$
1.0	0.8413447
1.1	0.8643339
1.2	0.8849303
1.3	0.9031995
1.4	0.9192433

Si nous considérons que la fonction aléatoire suit une loi normale, cette probabilité est donnée par la fonction :

$$P(X \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp \frac{-t^2}{2} dt$$

Comme la fonction $\exp \frac{-t^2}{2}$ ne possède pas de primitive, on calcule, par des méthodes numériques (que vous êtes invités à obtenir dans le problème :-), cette probabilité pour différentes valeurs de x et l'on conserve les résultats dans une table.

1. A l'aide de cette table, calculer la dérivée $P'(X \leq 1.2)$ avec une précision d'ordre quatre. Comparer avec la valeur exacte de cette dérivée.
2. Toujours en vous servant de la table fournie, calculer la dérivée $P''(X \leq 1.2)$ avec une précision d'ordre deux.

38 Démontrez la relation suivante.

$$u''''(x) = \frac{u(x-2h) - 4u(x-h) + 6u(x) - 4u(x+h) + u(x+2h)}{h^4} + \mathcal{O}(h^2)$$

39 On considère la formule de différentiation numérique suivante

$$u'(x) \approx \frac{u(x-2h) - 8u(x-h) + 8u(x+h) - u(x+2h)}{12h}$$

1. Montrer que cette formule peut être obtenue à partir de la formule centrée d'ordre deux et de l'extrapolation de Richardson.
2. Prédire, grâce à la théorie, l'ordre de cette formule.
3. Déterminer numériquement l'ordre de cette formule de différences en considérant la fonction $u(x) = e^x$ au point $x = 0$ et en utilisant successivement des pas distincts.

40 Considérons $u(x) = \exp(x)$.

1. Calculer une approximation numérique de $u'(2)$ au moyen d'une différence centrée
$$u'(x) \approx \frac{u(x+h) - u(x-h)}{2h}$$
avec un pas $h = 0.1$, $h = 0.01$ et $h = 0.001$.
2. Comparer avec la valeur exacte.
3. Calculer la borne d'erreur de troncature : est-ce une estimation pertinente ?

41 Considérons une fonction u trois fois continûment dérivable. Choisissons trois points X_0, $X_1 = X_0 + h$ et $X_2 = X_0 + 2h$. Finalement, définissons une seconde fonction :

$$\begin{aligned} u^h(x) &= u(X_0) + \left(\frac{u(X_0+h) - u(X_0)}{h}\right)(x - X_0) \\ &\quad + \left(\frac{u(X_0+2h) - 2u(X_0+h) + u(X_0)}{2h^2}\right)(x - X_0)(x - X_1) \end{aligned}$$

1. Vérifier que $u^h(X_j) = u(X_j)$ pour $j = 0, 1, 2$.
2. En déduire tout d'abord qu'il existe $\xi_1 \in [X_0, X_1]$ et $\xi_2 \in [X_1, X_2]$ tels que
$$u'(\xi_1) = (u^h)'(\xi_1) \text{ et } u'(\xi_2) = (u^h)'(\xi_2).$$
3. En déduire ensuite qu'il existe $\xi_3 \in [\xi_1, \xi_2]$ tel que
$$(u - u^h)''(\xi_3) = 0.$$
4. En conclure finalement que pour tout $x \in [X_0, X_2]$, on a :
$$|u(x) - u^h(x)| \leq 2h^3 \max_{\xi \in [X_0, X_2]} |u'''(\xi)|.$$
5. Comparer u^h avec le développement de Taylor de u.

42 Calculer la dérivée troisième de la fonction $u(x) = \cos x$ en $x = 0.8$ en effectuant des extrapolations à la limite sur les valeurs trouvées à l'aide d'une formule centrée d'ordre 2 pour 4 pas différents.

Séance 7

Calcul en virgule flottante

Propagation des erreurs d'arrondi

$x = \tilde{x} \pm \Delta x$

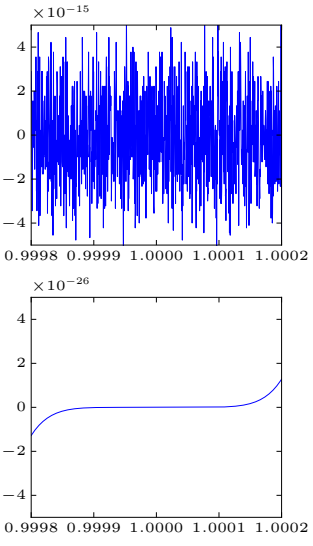
$$f(x) = f(\tilde{x}) \pm \underbrace{|f'(\tilde{x})|\ \Delta x}_{\Delta f} + \mathcal{O}(\Delta x^2)$$

43 Calculer les erreurs propagées relatives et absolues pour les quatre opérations élémentaires à partir d'opérandes $x \pm \Delta x$ et $y \pm \Delta y$.

44 En utilisant n bits a_i et un codage binaire en complément à deux (c'est la manière dont Java procède), on représente un entier e par l'expression :

$$e = -a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + a_{n-3}2^{n-3} \cdots + a_2\ 2^2 + a_1\ 2 + a_0$$

1. Vérifier que tous les entiers positifs ont $a_{n-1} = 0$.
2. Donner, avec 8 bits, la représentation binaire en complément à deux de 125, -125, 0, -175 et -100.



Exemple : $(x-1)^7$

Evaluons le polynôme

$$u(x) = \underbrace{x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1}_{(x-1)^7}$$

sur 1000 points entre les abscisses $1 - 2 \times 10^{-4}$ et $1 + 2 \times 10^{-4}$.

```
>>> from numpy import *
>>> from matplotlib import pyplot as plt
>>> p = [1, -7, 21,-35, 35,-21, 7, -1]
>>> x = linspace(1-2e-4,1+2e-4,1000)
>>> plt.figure(); plt.plot(x,polyval(p,x))
>>> plt.figure(); plt.plot(x,power((x-1.0),7))
>>> print(roots(p))
[ 1.00674577+0.00330183j  1.00674577-0.00330183j
  1.00154726+0.00727405j  1.00154726-0.00727405j
  0.99535345+0.00568063j  0.99535345-0.00568063j
  0.99270705+0.j        ]
```

Les coefficients alternés du polynôme génèrent une accumulation catastrophique des erreurs d'arrondi...

45 Si l'espace des nombres réels $\mathbb{R}$ est bien connu de tous, la manière dont ces nombres sont représentés par un ordinateur est nettement moins connue. Comme les ordinateurs n'ont qu'une mémoire limitée, ils ne peuvent représenter qu'un sous ensemble $\mathbb{F} \subset \mathbb{R}$ des nombres réels : il s'agit des *nombres en virgule flottante*. En général, un ordinateur stocke un nombre sous la forme suivante

$$x = (-1)^s \; 0.\underbrace{a_1 a_2 \ldots a_n}_{m} \; \beta^e, \qquad a_1 \neq 0,$$

L'espace $\mathbb{F}$ est caractérisé par la base β, le nombre n de chiffres (ou *digits*) de la mantisse m et les valeurs minimale L et maximale U de l'index entier e. Nous caractériserons un espace de nombre de virgules flottantes par l'expression $\mathbb{F}(\beta, n, L, U)$.

1. Quel est l'espace des nombres en virgules flottantes utilisé par `python` ? Quel est l'epsilon machine de cet espace ?
2. Combien d'éléments contient $\mathbb{F}(2, 2, -2, 2)$? Quel est l'epsilon machine de cet espace ?
3. Démontrer que $\mathbb{F}(\beta, n, L, U)$ contient exactement $2(\beta - 1)\beta^{n-1}(U - L + 1)$ éléments.
4. Quel est le lien entre l'erreur relative commise par la représentation d'un nombre en virgule flottante et l'epsilon machine ?

46 Considérons un ordinateur fictif utilisant l'ensemble $\mathbb{F}(2, 5, -2, 3)$.

1. Proposer un codage en huit bits de tous les éléments de cet ensemble, en suivant les mêmes principes que la convention IEEE-754.
2. Donner le plus petit nombre positif que l'on pourrait représenter.
3. Donner l'epsilon machine de cette représentation.
4. Exprimer le nombre 3.25 dans cette représentation.

47 On souhaite calculer la dérivée seconde de la fonction $f(x) = x \;\tan(x)\;$ en $x = 0.9$, en utilisant uniquement les valeurs (arrondies à quatre chiffres après la virgule) de la table ci-dessous.

x	0.8000	0.8500	0.9000	0.9500	1.0000
$x\tan(x)$	0.8237	0.9676	1.1341	1.3285	1.5574

1. Donnez le polynôme d'interpolation passant par les points de $f(x)$ correspondant aux abscisses 0.8, 0.9, 1.0 et calculez la dérivée *seconde* de ce polynôme en $x = 0.9$.
2. Calculez une estimation numérique de la dérivée *seconde* de $f(x)$ en $x = 0.9$ en calculant une extrapolation de Richardson à partir des valeurs trouvées à l'aide d'une formule de différence centrée d'ordre deux avec deux pas distincts.
3. Donnez l'ordre théorique de l'erreur pour les deux approximations obtenues de la valeur de la dérivée seconde. En tenant compte des erreurs d'arrondis des données, peut-on utiliser ces ordres de précision pour en déduire une estimation fiable de l'erreur des approximations ?

48 La série de puissances pour obtenir la fonction sinus est :

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \dots$$

Considérons une fonction `python` qui calcule cette série.

```
def powersin(x):
  s=0; t=x; n=1
  while ((s + t) != s):
    s = s+t
    t = - x*x*t/((n+1)*(n+2))
    n = n+2
  return s
```

Ensuite, on se propose d'utiliser cette fonction afin d'obtenir le sinus pour les valeurs suivantes $x = \pi/2,\ 13\pi/2,\ 23\pi/2$ et $33\pi/2$. Qu'observez-vous ?

1. Pourquoi est-ce que l'instruction de boucle se termine,
   alors que le test ne peut jamais être satisfait théoriquement ?
2. Quelle est la précision des résultats ?
3. Combien de termes sont requis ?
4. Quel est le plus grand terme dans ces séries ?
5. Est-ce que vous recommandez l'usage de séries de puissance pour évaluer la fonction sinus ?

49 Ecrire un programme `python` qui effectue le calcul des coefficients binomiaux

$$C_k^n = \frac{n!}{k!(n-k)!},$$

où n et k sont des entiers positifs avec $k \leq n$.

50 Ecrire un programme `python` qui implémente l'algorithme suivant pour évaluer π. On génère n couples de nombres aléatoires sur l'intervalle $[0, 1]$ et on calcule la portion m d'entre eux qui se trouvent sur le premier quadrant du cercle unité. On observe immédiatement que π peut être vu comme la limite de la suite $pi_n = 4m/n$. Fournir également une estimation de l'erreur de l'approximation pour un n donné.

1. Est-ce que le résultat produit par le programme sera totalement aléatoire ?
2. Que fournit la commande `random` de `python` ?

Séance 8

Problèmes aux valeurs initiales

Trouver $u(x)$ tel que

$$\begin{cases} u'(x) &= f(x, u(x)), & x \in [a, b] \\ u(a) &= \bar{u} \end{cases}$$

Exemple : ordre des méthodes d'Euler

Considérons le problème de Cauchy

$$\begin{cases} u'(x) &= cos(2u(x)), & x \in [0, 1] \\ u(0) &= 0 \end{cases}$$

dont la solution est $u(x) = \frac{1}{2}\arcsin((e^{4x}-1)/(e^{4x}+1))$. Nous souhaitons estimer l'ordre des méthodes d'Euler explicite et implicite avec différentes valeurs de pas h : $1/2, 1/4, 1/8 \ldots$ en comparant la valeur exacte et la valeur approchée de $u(1)$.

```
from numpy import *
from scipy.optimize import fsolve

Xstart = 0; Xend = 1
Ustart = 0; Uend = 0.5*arcsin((exp(4*Xend)-1)/(exp(4*Xend)+1))
Eexpl = zeros(8); Eimpl = zeros(8)

for j in range(8):
  n = pow(2,j+1)
  h = (Xend-Xstart)/n
  Uexpl = zeros(n+1); Uexpl[0] = Ustart
  Uimpl = zeros(n+1); Uimpl[0] = Ustart
  for i in range(n):
    Uexpl[i+1] = Uexpl[i] + h*cos(2*Uexpl[i])
    fimpl = lambda x : (x - Uimpl[i])/h - cos(2*x)
    Uimpl[i+1] = fsolve(fimpl,Uimpl[i])
  Eexpl[j] = abs(Uexpl[-1] - Uend)
  Eimpl[j] = abs(Uimpl[-1] - Uend)
```

Pour l'implémentation de la méthode d'Euler implicite, nous avons utilisé la fonction `fsolve` pour trouver la solution du problème non-linéaire à chaque pas de temps. Notons également l'utilisation d'une fonction anonyme `lambda` pour redéfinir, à chaque pas de temps, la fonction du problème non-linéaire. Il est alors possible d'estimer l'ordre de convergence de la manière suivante.

```
>>> Oexpl = log(abs(Eexpl[:-1]/Eexpl[1:]))/log(2)
>>> Oimpl = log(abs(Eimpl[:-1]/Eimpl[1:]))/log(2)
>>> print("orderExpl ", *['%.4f ' % val for val in Oexpl] )
>>> print("orderImpl ", *['%.4f ' % val for val in Oimpl] )
orderExpl  1.2898  1.0810  1.0349  1.0164  1.0080  1.0039  1.0019
orderImpl  0.9070  0.9484  0.9720  0.9853  0.9925  0.9962  0.9981
```

On observe bien que les deux méthodes convergent avec un ordre de précision linéaire !

51 On considère le problème de Cauchy

$$u'(x) = \sin(x) + u(x), \qquad x \in [0,1], \qquad \text{avec } u(0) = 0$$

Calculer la solution analytique.
Appliquer les méthodes d'Euler explicite et d'Euler implicite pour obtenir $u(1)$.
Vérifier que les deux méthodes convergent avec un ordre linéaire.

52 On considère l'équation différentielle ordinaire

$$u'(x) = 5\Big(\, x - u(x) \,\Big) + 1$$

On souhaite trouver u sur l'intervalle $[0,4]$ avec la condition initiale $u(0) = 1$.

1. Est-ce une équation différentielle non linéaire ?
2. Est-ce une équation différentielle non homogène ?
3. Calculer la solution analytique du problème de Cauchy.
4. Analyser la stabilité numérique des méthodes d'Euler explicite, d'Euler implicite et de la méthode de Runge-Kutta d'ordre quatre. Quel est le plus grand pas h possible que l'on pourrait utiliser sans risquer d'observer des instabilités numériques ?
5. Ecrire un programme `python` qui calcule $u(4)$ par les méthodes d'Euler explicite, d'Euler implicite et de Runge-Kutta d'ordre quatre. Comparer les résultats obtenus avec 4 pas, 8 pas, 16 pas et 32 pas respectivement. Obtient-on bien les ordres de convergence prédits par la théorie ?

53 On considère le problème de Cauchy

$$u'(x) = -xe^{-u(x)}, \qquad x \in [0,1], \qquad \text{avec } u(0) = 0$$

Estimer le nombre de chiffres significatifs de la solution approchée en $x = 1$ obtenue par une méthode d'Euler explicite avec $h = 1/100$.

54 Démontrer que la méthode de Crank-Nicolson

$$U_{i+1} = U_i + \frac{h}{2}\left(F_i + F_{i+1}\right)$$

peut être obtenue à partir de la forme intégrale du problème de Cauchy

$$u(x) - u(0) = \int_0^x f(t, u(t))dt$$

où l'intégrale est approchée par une quadrature composite des trapèzes.

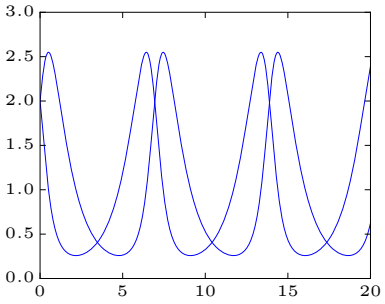
55 On considère le problème modèle $u'(x) = \lambda\, u(x)$ avec $\lambda = -1 + i$.
Trouver les valeurs de pas h pour lesquelles la méthode d'Euler explicite est stable.

Séance 9

Problèmes aux valeurs initiales

Trouver $u(x)$ tel que

$$\begin{cases} u'(x) &= f(x, u(x)), & x \in [a, b] \\ u(a) &= \bar{u} \end{cases}$$



Exemple : équations de Lotka-Volterra

Les équations de Lotka-Volterra modélisent l'évolution démographique d'une proie et d'un prédateur confinés dans un espace clos.

$$\begin{cases} u'(x) &= u(x)(1 - v(x)), \\ v'(x) &= v(x)(u(x) - 1), \end{cases}$$

Pour obtenir une solution numérique de ce système avec `scipy`, il suffit de définir une fonction f vectorielle comme suit.

```
def f(x,u):
  dudx = u[0]*(1 - u[1])
  dvdx = u[1]*(u[0] - 1)
  return [dudx, dvdx]
```

Et une manière classique d'intégrer ce système est d'utiliser une méthode adaptative de Runge-Kutta-Dormand-Prince implémentée dans la fonction `ode45`. On considère ici une condition initiale $(u(0), v(0)) = (2, 2)$ et on calcule l'évolution des deux variables jusqu'à $x = 20$.

```
from matplotlib import pyplot as plt
from scipy.integrate import solve_ivp

sol = solve_ivp(f,[0,20],[2,2],method='RK45',rtol=1e-9,atol=1e-9)
x = sol.t; u = sol.y[0]; v = sol.y[1]
plt.plot(x,u,'-b',x,v,'-r')
```

La résolution des systèmes d'équations différentielles ordinaires est donc particulièrement simple avec `python` et `scipy 1.1.0`

56 On note $x(t)$ le déplacement par rapport à la position d'équilibre d'un système qui vibre. Typiquement, nous considérons une bille de masse $M = 1$ reliée au plafond par un ressort linéaire de raideur $k = 6$ et par un amortisseur assimilé à un frottement visqueux de coefficient $C = 5$. Le ressort et l'amortisseur sont montés en parallèle.

1. Montrer que l'écart $x(t)$ est régi par $Mx''(t) = -kx(t) - Cx'(t)$.
2. Donner la solution analytique.
3. Obtenir l'évolution du déplacement et de la vitesse pour $t \in [0, 5]$ en utilisant la méthode de Heun en supposant que $x(0) = 1$ et $x'(0) = 0$.

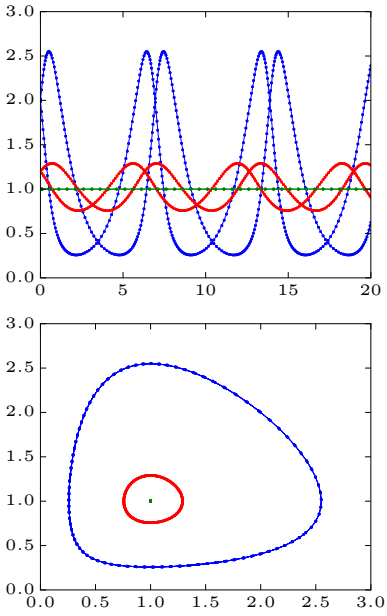
Exemple : points d'équilibre

Reprenons les équations de Lotka-Volterra.

$$\begin{cases} u'(x) &= u(x)(1 - v(x)), \\ v'(x) &= v(x)(u(x) - 1), \end{cases}$$

On a calculé l'évolution temporelle des solutions obtenues en partant respectivement de $(2, 2)$, $(1.2, 1.2)$ et $(1, 1)$. On observe que toutes les solutions sont périodiques (la période vaut $2/\pi$) et que l'amplitude des oscillations est fonction de la condition initiale.

On a aussi dessiné les trois trajectoires dans le plan de phase dont les axes de coordonnées sont respectivement u et v. On voit que toutes les trajectoires sont confinées dans une région bornée de ce plan. Si on part de $(1.2, 1.2)$, la trajectoire restera dans une plus petite région entourant le point $(1, 1)$. Ce comportement est lié au fait que le système admet deux points d'équilibre caractérisés par $u' = v' = 0$ et l'un de ces points est justement $(1, 1)$ (l'autre étant $(0, 0)$). Si les données initiales coïncident avec ces points d'équilibre, la solution reste constante dans le temps. Toutefois, il y a une différence entre ces deux points d'équilibre... on dira que $(0, 0)$ est un point d'équilibre instable tandis que le point $(1, 1)$ sera un point d'équilibre stable car toutes les trajectoires issues d'un point proche de $(1, 1)$ resteront proches de ce point...



57 On considère l'équation différentielle ordinaire

$$u'(x) = -10\,(x - 1)\,u(x)$$

On souhaite trouver u sur l'intervalle $[0, 2]$ avec la condition initiale $u(0) = 0.05$.

1. Est-ce une équation différentielle non linéaire ?
2. Est-ce une équation différentielle non homogène ?
3. Calculer la solution analytique du problème de Cauchy.
4. Analyser la stabilité numérique des méthodes d'Euler explicite, d'Euler implicite, de la méthode de Runge-Kutta d'ordre quatre et de la méthode d'Adams-Bashforth d'ordre quatre. Quel est le plus grand pas h possible que l'on pourrait utiliser sans risquer d'observer des instabilités numériques ?
5. Ecrire un programme `python` qui calcule $u(2)$ par les méthodes d'Euler explicite et implicite. Comparer les résultats obtenus avec 10 pas, 100 pas , 1000 pas et 10000 pas respectivement. Refaire le calcul avec une méthode de Runge-Kutta d'ordre quatre avec 4, 8, 16 et 32 pas. Obtient-on bien les ordres de convergence prédits par la théorie ?

58 Ecrire un programme `python` qui dessine la région de stabilité d'une méthode de Heun.

59 Le mouvement sans friction d'un pendule de Foucault est décrit par les deux équations

$$\begin{aligned} x''(t) - 2\omega \sin(\phi)\ y'(t) + k^2\ x(t) &= 0, \\ y''(t) + 2\omega \sin(\phi)\ x'(t) + k^2\ y(t) &= 0, \end{aligned}$$

où ϕ est la latitude de l'endroit où le pendule est localisé, $\omega = 7.29\ 10^{-5}\ \mathrm{s}^{-1}$ est la vitesse angulaire de la rotation de la Terre, $k = \sqrt{g/l}$ avec $g = 9.81\ \mathrm{m/s}^2$ et $l = 20$ m est la longueur du pendule.

Prédire le mouvement du pendule lorsqu'on part d'une position initiale $(x, y) = (1, 0)$ et d'une vitesse initiale nulle. Utiliser les méthodes d'Euler explicite et de Runge-Kutta d'ordre trois (`ode23`) pour un temps entre 0 et 300 secondes et une latitude de $\pi/2$.

60 On souhaite analyser la zone de stabilité d'une méthode prédicteur-correcteur dont la prédiction consiste à appliquer une formule d'Adams-Bashforth d'ordre deux et dont la correction réside en une unique application d'une formule d'Adams-Moulton du même ordre.

$$\left\{ \begin{array}{rcl} P_{i+1} &=& U_i + \dfrac{h}{2}\ \Big(-f(X_{i-1}, U_{i-1}) + 3f(X_i, U_i) \Big) \\ U_{i+1} &=& U_i + \dfrac{h}{2}\ \Big(f(X_i, U_i) + f(X_{i+1}, P_{i+1}) \Big) \end{array} \right.$$

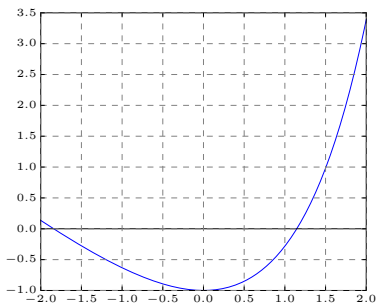
Ecrire un programme `python` qui dessine la région de stabilité d'une telle méthode dans le plan complexe $h\lambda$.

Séance 10

Equations non linéaires ... et linéaires

Trouver x tel que

$$f(x) = 0$$



Exemple : zéros d'une fonction

Considérons la fonction :

$$f(x) = e^x - 2 - x$$

et essayons d'en trouver les zéros avec `python`.

```
>>> from math import exp
>>> from scipy.optimize import fsolve
>>>
>>> f = lambda x : exp(x) - 2.0 - x
>>> fsolve(f,+1.0)
array([1.14619322])
>>> fsolve(f,-1.0)
array([-1.84140566])
```

C'est évidemment très facile !

61 En n'utilisant pas les commandes `plot` et `fsolve`, trouver la ou les solutions des équations :

1. $\ln(x) - 5 + x = 0$
2. $(x-2)^2 = \ln(x)$
3. $e^{-x} = x$
4. $x^3 + 4x^2 - 10 = 0$

62 Ecrire un programme `python` pour résoudre l'équation

$$x\,e^x \;=\; 0$$

par la méthode de Newton-Raphson en prenant successivement comme points de départ $x_0 = 0.2$ et $x_0 = 20$.

63 Résoudre l'équation $x^2 = 4(x-1)$ en utilisant l'itération

$$x_{i+1} = 2\sqrt{x_i - 1}$$

en prenant successivement comme points de départ $x_0 = 1.5$ et $x_0 = 2.5$. Est-il possible de prédire si la convergence sera rapide ?

64 Résoudre l'équation $x^2 - 6x + 8 = 0$, en utilisant l'itération

$$x_{i+1} = 4x_i - \frac{x_i^2}{2} - 4$$

en prenant successivement comme points de départ $x_0 = 1.9$ et $x_0 = 3.8$. Essayer finalement de partir du point $x_0 = 6.0$, qu'observez-vous ?

65 Quel est le point de la parabole $y = x^2$ le plus proche du point P de coordonnées $(3, 1)$?

66 On dispose d'un rectangle de carton de 16 x 10 cm. On forme une boîte parallélépipédique ouverte en coupant aux quatre coins du rectangle des carrés identiques de côtés x et en relevant les 4 rectangles latéraux obtenus. Que doit valoir x pour que le volume de cette boîte soit de 100 cm^3 ?

67 Calculer $x = \sqrt[3]{3 + \sqrt[3]{3 + \sqrt[3]{3 + \dots}}}$.

68 Pour obtenir une solution du système non linéaire

$$\begin{cases} x^2 - y = 0 \\ 4x^2 + 9y^2 - 8x - 32 = 0 \end{cases}$$

on souhaite partir du point $(1.4, 2.0)$ et utiliser l'itération

$$\begin{cases} x_{i+1} = (2x_i - x_i^2 + y_i)/2 \\ y_{i+1} = (2x_i - x_i^2 + 8)/9 + (4y_i - y_i^2)/4 \end{cases}$$

Démontrer que si l'itération converge, on obtiendra une solution du système.
Confirmer ce résultat avec la méthode de Newton-Raphson.

69 Résoudre le système : $\begin{cases} 3\ x^2 - 2\ y^2 = 1 \\ x^2 - 2\ x + y^2 + 2\ y = 8 \end{cases}$

70 Résoudre le système : $\begin{cases} x + y = 1 \\ y + z = -2 \\ x + 2\ y + z = 1 \end{cases}$

71 Résoudre le système : $\begin{cases} x + 5\ y = 6 \\ 1.0001\ x + 5\ y = 6.0005 \end{cases}$

1. Quelle est sa solution $\mathbf{x}$?
2. On trouve par erreur la solution $\mathbf{s}_1 = (5.1\ 0.3)$ ou la solution $\mathbf{s}_2 = (1\ 1)$.
   Que valent les résidus $\mathbf{r}_1$ et $\mathbf{r}_2$?
   Que valent leurs normes ?
   Que peut-on en conclure ?
3. Que devient la solution exacte si les deux termes constants sont égaux à 6 ?
   Que peut-on en conclure ?

72 Résoudre le système suivant par la méthode de Gauss-Seidel :

$$\begin{bmatrix} 5 & 3 \\ 4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

73 **Juin 2003**

On dispose de n mesures (X_i, Y_i) d'une fonction inconnue $y = u(x)$ et on souhaite approximer $u(x)$ par une expression

$$u^h(x) = \frac{a}{x+b}$$

en ajustant les deux paramètres réels a et b afin de minimiser la somme des carrés des n écarts $Y_i - u^h(X_i)$. On vous demande :

1. de donner deux équations que doivent satisfaire a et b en termes des n données X_i et Y_i.
2. de considérer la méthode de Newton-Raphson pour la résolution de ces deux équations à partir d'une estimation initiale (a_0, b_0). Plus précisément, on vous demande de décrire comment on obtient une nouvelle estimation (a_{k+1}, b_{k+1}), en termes des n données X_i et Y_i et d'une estimation précédente (a_k, b_k). [1]
3. de calculer a sachant que $b = 1$ pour les trois mesures (X_i, Y_i) ci-dessous.
4. de décrire ce que produira la méthode de Newton-Raphson en partant de $(a_0, b_0) = (0, 0)$ pour les trois mesures (X_i, Y_i) ci-dessous.

X_i	0	1	2
Y_i	$\frac{49}{8}$	2	$\frac{25}{8}$

Il n'est pas nécessaire de disposer d'une calculatrice pour résoudre cette question ! Observez bien que les deux premières sous-questions demandent une réponse générale totalement indépendante des trois mesures de la table !

74 **Septembre 2003**

On dispose toujours de n mesures (X_i, Y_i) d'une fonction inconnue $y = u(x)$ et on souhaite approximer $u(x)$ par une expression

$$u^h(x) = \frac{6}{x+b}$$

en ajustant le paramètre réel b afin de minimiser la somme des carrés des n écarts $Y_i - u^h(X_i)$. On vous demande :

1. de donner une équation que doit satisfaire b en termes des n données X_i et Y_i.
2. de considérer la méthode de Newton-Raphson pour la résolution de cette équation à partir d'une estimation initiale b_0. Plus précisément, on vous demande de décrire avec *concision et rigueur* comment on obtient une nouvelle estimation b_{k+1}, en termes des n données X_i et Y_i et d'une estimation précédente b_k.

[1] Recopier la définition du schéma de Newton-Raphson fournie dans le formulaire NE répond PAS à la question et est donc TOTALEMENT inutile!

3. de décrire ce que produira la méthode de Newton-Raphson en partant de $b_0 = 1$ pour les deux mesures (X_i, Y_i) du tableau.

X_i	0	1
Y_i	$\frac{25}{4}$	2

Réaliser un dessin schématique du problème (en y incluant les données numériques du tableau ci-dessus) est souvent utile avant de répondre aux trois questions !

Séance 11

Transfert de chaleur et équation de la diffusion

Trouver $u(\mathbf{x}, t)$ tel que

$$\rho c \frac{\partial u}{\partial t} = k \boldsymbol{\nabla}^2 u + f$$

75 On souhaite obtenir $u(x, y)$ tel que :

$$\begin{aligned} \boldsymbol{\nabla}^2 u(x,y) + 1 &= 0, \quad (x,y) \in \Omega, \\ u(x,y) &= 0, \quad (x,y) \in \partial\Omega, \end{aligned}$$

où le domaine Ω est le carré dont le côté vaut deux et dont le centre est l'origine du plan. La frontière de ce domaine est désignée par $\partial\Omega$.

1. Calculer la solution analytique de ce problème.
2. Ecrire un programme `python` qui calcule la solution numérique par différences finies.

   ```
   U = poissonSolve(nx,ny)
   ```

   où `nx` et `ny` sont les nombres de noeuds dans les directions x et y. La fonction renvoie un tableau de taille $n_x \times n_y$ contenant l'ensemble des valeurs nodales.
3. En utilisant plusieurs maillages, déduire le taux de convergence de la méthode.
4. Comparer la vitesse d'exécution du programme lorsqu'on utilise une matrice pleine et une matrice creuse.

76 On refroidit un système par un convecteur muni d'ailettes en aluminium. Comme ces ailettes sont très nombreuses et très longues, on peut se contenter de calculer la température dans une section d'une ailette centrale. A la base de l'ailette, une densité de flux de chaleur q_{in} entre, tandis qu'aux autres côtés, la densité de flux de chaleur sortant vaut $h(T - T_f)$ où T_f est la température de l'air ambiant et h est un coefficient de convection donné[1].

On souhaite calculer l'évolution de la température dans le cas où l'ailette est initialement à la température de l'air et est soumise en $t = 0$ au flux de chaleur entrant. En conclusion, il faut résoudre le problème sur une demi-ailette (symétrie par rapport à l'axe y);

[1] En réalité, il faut le déterminer en fonction de la vitesse de l'air et du régime de l'écoulement (laminaire ou turbulent) : ce sera l'objet des cours de mécanique des fluides et transferts...

$$
\begin{cases}
\rho c \dfrac{\partial T}{\partial t}(x,y,t) &=& k\Big(\dfrac{\partial^2 T}{\partial x^2}(x,y,t) + \dfrac{\partial^2 T}{\partial y^2}(x,y,t)\Big), & \\
-k\dfrac{\partial T}{\partial y}(x,L_y,t) &=& q_{in} & -k\dfrac{\partial T}{\partial y}(x,0,t) = h_y\Big(T(x,0,t) - T_f\Big), \\
-k\dfrac{\partial T}{\partial x}(0,y,t) &=& 0 & -k\dfrac{\partial T}{\partial x}(L_x,y,t) = h_x\Big(T(L_x,y,t) - T_f\Big), \\
T(x,y,0) &=& T_f. &
\end{cases}
$$

Pour résoudre numériquement ce problème, nous allons utiliser des différences finies avec un maillage où $\Delta x = \Delta y$ et une méthode d'Euler explicite.

$$
\begin{aligned}
\rho c\Big(\frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t}\Big) &= k\,\Big(\frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{(\Delta x)^2} + \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{(\Delta y)^2}\Big) \\
&\Big\downarrow \quad \text{En définissant } \beta = \frac{k\Delta t}{\rho c(\Delta x)^2} = \frac{k\Delta t}{\rho c(\Delta y)^2}, \\
T_{i,j}^{n+1} &= T_{i,j}^n + \beta\Big(T_{i+1,j}^n + T_{i-1,j}^n + T_{i,j+1}^n + T_{i,j-1}^n - 4T_{i,j}^n\Big)
\end{aligned}
$$

où $i = 1, \ldots, n_x, j = 1, \ldots, n_y$ et n sont respectivement les indices spatiaux et temporels.

Comme toutes les conditions aux limites font intervenir le flux, il est astucieux d'utiliser un tableau de taille $(n_x + 2) \times (n_y + 2)$ afin d'ajouter une couche en plus au domaine physique et d'appliquer l'opérateur différentiel sur tous les noeuds de l'ailette. Par contre, nous utiliserons les conditions aux limites, pour fixer les valeurs de noeuds supplémentaires. Ainsi, avec la condition à la paroi supérieure,

$$
\begin{aligned}
-k\,\Big(\frac{T_{i,2}^n - T_{i,0}^n}{2\Delta y}\Big) &= h_y\,\Big(T_{i,1}^n - T_f\Big) \\
&\Big\downarrow \\
T_{i,0}^n &= T_{i,2}^n - \frac{2h_y\Delta y}{k}\Big(T_{i,1}^n - T_f\Big)
\end{aligned}
$$

on peut déterminer la valeur de $T_{i,0}^n$ à utiliser avant chaque pas de temps.
On peut procéder exactement de la même manière pour les trois autres côtés.

Valeurs numériques

k	204.0	$W\ m^{-1}\ K^{-1}$
c	896.0	$J\ kg^{-1}\ K^{-1}$
ρ	2707.0	$kg\ m^{-3}$
T_f	300.0	K
h_x	60.0	$W\ m^{-2}\ K^{-1}$
h_y	100.0	$W\ m^{-2}\ K^{-1}$
q_{in}	30000.0	$W\ m^{-2}$

L_x	2.5	mm
L_y	15.0	mm
$\Delta x = \Delta y$	0.5	mm

1. Effectuer l'analyse de stabilité et montrer qu'il faut choisir un pas de temps tel que $\beta \leq 1/4$.
2. Ecrire un programme `python` qui calcule la solution numérique par différences finies après un laps de temps de 60 secondes.

```
U = diffusionSolve(betaRequested,n)
```

   où `n` est le nombre de maille dans la directions x et `betaRequested` est la valeur de β souhaitée. La fonction renvoie un tableau de taille $(n_x + 2) \times (n_y + 2)$ contenant l'ensemble des valeurs nodales avec `nx = n` 1+, tandis que `ny` sera choisie pour obtenir des mailles carrées telles que $\Delta x = \Delta y$.
3. Comment peut-on adapter le programme pour extraire l'évolution temporelle en un point précis du maillage, sans conserver toutes les valeurs en mémoire...
4. Comparer une implémentation vectorielle avec une implémentation plus naïve avec des boucles. On pourra ainsi observer que l'utilisation explicite de boucles `for` dans un code `python` est extrêmement pénalisant en termes d'efficacité !
5. Le coefficient de convection pour la paroi supérieure est quasiment deux fois supérieur à celui des parois latérales : comment expliquer ce choix ?

77 Recherchons maintenant la solution de régime du problème précédent :

$$\left\{\begin{array}{rcll} 0 & = & \Big(\dfrac{\partial^2 T}{\partial x^2}(x,y,t) + \dfrac{\partial^2 T}{\partial y^2}(x,y,t)\Big), & \\ -k\dfrac{\partial T}{\partial y}(x,L;y,t) & = & q_{in} & -k\dfrac{\partial T}{\partial y}(x,0,t) \;=\; h_y\Big(T(x,0,t) - T_f\Big), \\ -k\dfrac{\partial T}{\partial x}(0,y,t) & = & 0 & -k\dfrac{\partial T}{\partial x}(L_x,y,t) \;=\; h_x\Big(T(L_x,y,t) - T_f\Big). \end{array}\right.$$

On procède exactement comme pour le problème transitoire en ajoutant il est astucieux d'utiliser un tableau de taille $(n_x + 2) \times (n_y + 2)$ afin d'ajouter une couche en plus au domaine physique et d'appliquer l'opérateur différentiel sur tous les noeuds de l'ailette.

Pour obtenir une solution de régime, nous allons nous inspirer du problème transitoire pour construire une méthode itérative... Comme seule la solution de régime nous intéresse, tous les coups sont permis et nous allons donc également appliquer l'idée de Gauss et Seidel en utilisant les valeurs les plus récentes au fur et à mesure que l'on itère. Ensuite, nous tenterons encore d'accélérer la convergence en introduisant un facteur de sur-relaxation $\omega = 1.5 \ldots 1.9$. Théoriquement, on peut monter qu'il existe un facteur optimum tel que les itérations convergeront le plus rapidement possible vers la solution stationnaire.

Jacobi sur-relaxée	$T_{i,j}^{n+1} = T_{i,j}^n + \dfrac{\omega}{4}\Big(T_{i+1,j}^n + T_{i-1,j}^n + T_{i,j+1}^n + T_{i,j-1}^n - 4T_{i,j}^n\Big)$
Gauss-Seidel sur-relaxée	$T_{i,j}^{n+1} = T_{i,j}^n + \dfrac{\omega}{4}\Big(T_{i+1,j}^n + T_{i-1,j}^{n+1} + T_{i,j+1}^n + T_{i,j-1}^{n+1} - 4T_{i,j}^n\Big)$

1. Expliquer comment on peut calculer le bilan global de flux chaleur entre le flux entrant $Q_{in} = q_{in}\ L_x$ et le flux sortant Q_{out} entre l'aluminium et le fluide ambiant
2. Ecrire un programme `python` pour obtenir la solution de régime avec soit la méthode de Jacobi ou la méthode de Gauss-Seidel avec un facteur de sur-relaxation. Afin de mesurer globalement le taux de convergence, le programme estimera à chaque itération, l'erreur relative sur le bilan de flux de chaleur $e^n = |Q_{in}^n - Q_{out}^n|/Q_{in}^n$.

3. Quelle est la méthode itérative la plus efficace pour obtenir une solution de régime ?
4. Trouver la valeur optimale de ω en faisant appel aux informations du web :-)
   Est-ce que vous obtenez la même chose avec votre programme ?

Séance 12

Résolution numérique de l'équation d'onde

Trouver $u(x,t)$ tel que

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

78 Considérons une corde tendue vibrante dont la longueur à l'équilibre est L et dont les deux extrémités sont fixées : $u(0,t) = 0$ et $u(L,t) = 0$. La masse de la corde par unité de longueur est $\rho = 0.01 kg/m$ et la tension à l'équilibre est donnée par $T_0 = 0.1\ N$. Le déplacement transversal de la corde par rapport à sa position d'équilibre est donné par $u(x,t)$ et est régi par :

$$\rho \frac{\partial^2 u}{\partial t^2} = T_0 \frac{\partial^2 u}{\partial x^2} \qquad\qquad x \in]0, L[$$

On souhaite montrer que l'équation d'onde conserve l'énergie totale définie comme la somme de l'énergie cinétique et de l'énergie potentielle : $E = E_c(t) + E_p(t)$.

1. Donner l'expression analytique $E_p(t)$ de l'énergie potentielle de déformation à un instant donné en fonction de $u(x,t)$ et de ses dérivées partielles.
2. Donner l'expression analytique $E_c(t)$ de l'énergie cinétique à un instant donné en fonction de $u(x,t)$ et de ses dérivées partielles.
3. Démontrer que la dérivée de l'énergie totale peut être écrite sous la forme suivante :

$$\frac{dE}{dt} = \rho c^2 \left[\frac{\partial u}{\partial t} \frac{\partial u}{\partial x} \right]_0^L$$

On en déduit immédiatement que dans le cas considéré, l'énergie totale reste constante.

79 La solution analytique du problème aux conditions aux limites

$$\begin{cases} \rho \dfrac{\partial^2 u}{\partial t^2}(x,t) &= T_0 \dfrac{\partial^2 u}{\partial x^2}(x,t) \\ u(0,t) = u(L,t) &= 0 \\ u(x,0) &= u_0 \left(1 - \dfrac{x}{L}\right) \left(\dfrac{x}{L}\right)^2 \\ \dfrac{\partial u}{\partial t}(x,0) &= 0 \end{cases}$$

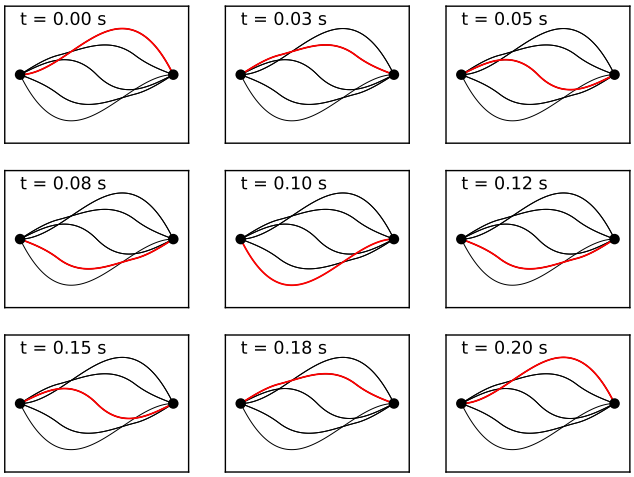


Figure 12.1: Solution analytique avec un incrément temporel de $T/8$ où T est la période : la position initiale est $u_0\left(1-\frac{x}{L}\right)\left(\frac{x}{L}\right)^2$ avec une vitesse initiale nulle..

peut être obtenue par la technique de séparation de variables

$$u(x,t) = \frac{4u_0}{\pi^3}\sum_{n=1}^{\infty}\frac{1}{n^3}\left(2(-1)^{n+1}-1\right)\sin\left(\frac{n\pi x}{L}\right)\cos\left(\frac{n\pi ct}{L}\right)$$

.

avec $c=\sqrt{\frac{T_0}{\rho}}$.

1. Obtenir cette solution exacte. Pour vous aider, on vous fournit les primitives suivantes autrefois disponibles dans la plupart des bonnes tables d'intégrales et que vous pouvez facilement retrouver sur `wikipedia`.

$$\int x^2\sin(ax)dx = \frac{2x}{a^2}\sin(ax)+\left(\frac{2}{a^3}-\frac{x^2}{a}\right)\cos(ax)$$

$$\int x^3\sin(ax)dx = \left(\frac{3x^2}{a^2}-\frac{6}{a^4}\right)\sin(ax)+\left(\frac{6x}{a^3}-\frac{x^3}{a}\right)\cos(ax)$$

   Il est aussi possible de faire appel au calcul symbolique de `simpy` au passage !

2. Ecrire un programme `python` évaluant la solution exacte en point x à l'instant t avec une précision absolue de de 10^{-8} :

```
u = waveAnalytic(x,t,L,c,u0)
```

3. Calculer la valeur de l'énergie totale de cette solution exacte.

Figure 12.2: Description de la fonction $scipy.solve_ivp$ qui peut réaliser intégration adaptative avec un calcul en parallèle d'une méthode de Runge-Kutta d'ordre quatre et cinq. L'erreur est contrôlée sur base de l'estimation de l'erreur de la méthode d'ordre quatre, tandis que l'intégration est faite avec la méthode d'ordre cinq. Ne pas hésiter à consulter la documentation en ligne avant de tenter de faire l'implémentation demandée :-).

80 Restons toujours avec ce même problème aux conditions aux limites :

$$\rho \frac{\partial^2 u}{\partial t^2}(x,t) = T_0 \frac{\partial^2 u}{\partial x^2}(x,t) \qquad u(0,t) = u(L,t) = 0 \qquad u(x,0) = u_0(x) \qquad \frac{\partial u}{\partial t}(x,0) = 0$$

Nous allons d'abord obtenir une solution approchée avec une méthode de Runge-Kutta adaptative.

1. Ecrire un programme `python` qui intègre ce problème avec `scipy.solve_ivp`

   ```
   sol = waveRungeKutta(nx,t,c,L,Uo)
   ```

   où les arguments sont respectivement le nombre d'intervalles n_x et le temps final à atteindre dans l'intégration. Les paramètres matériels du problème sont la vitesse de l'onde c et la longueur du domaine L. La condition initiale est fournie dans le tableau `Uo` de taille $n_x + 1$. La fonction renvoie simplement `sol` directement fourni par la fonction `scipy.solve_ivp`.
2. Comment sélectionner `t = T` pour intégrer l'équation sur une période temporelle ?
3. A partir des résultats obtenus, expliquer comment on peut dessiner aisément la solution numérique aux temps intermédiaires avec un incrément $T/8$ si T correspond à la période du problème.
4. A partir des résultats obtenus, effectuer un graphe de l'énergie cinétique, l'énergie potentielle et l'énergie totale en fonction du temps.
5. Estimer a priori l'ordre de grandeur du plus grand pas de temps utilisé ?

81 Pour encore approfondir la question, restons encore et toujours avec ce même problème aux conditions aux limites :

$$\rho\frac{\partial^2 u}{\partial t^2}(x,t) = T_0\frac{\partial^2 u}{\partial x^2}(x,t) \qquad u(0,t) = u(L,t) = 0 \qquad u(x,0) = u_0(x) \qquad \frac{\partial u}{\partial t}(x,0) = 0$$

Nous allons maintenant obtenir une solution approchée avec la méthode des différences finies centrées :

$$\begin{aligned} \frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{(\Delta t)^2} &= c^2\frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{(\Delta x)^2} \\ &\downarrow \\ U_i^{n+1} &= 2U_i^n + \beta^2\Big(U_{i+1}^n - 2U_i^n + U_{i-1}^n\Big) - U_i^{n-1} \end{aligned}$$

où i et n sont respectivement l'indice spatial et l'indice temporel. Le schéma est caractérisé par le coefficient :

$$\beta = \frac{c\Delta t}{\Delta x}$$

où Δx et Δt sont les pas spatial et temporel.

1. Effectuer l'analyse de stabilité et montrer qu'il faut choisir un pas de temps tel que $\beta \leq 1$.
2. Ecrire un programme `python` qui calcule la solution numérique.

   ```
   U = waveSolve(beta,nx,nt,c,L,Uo)
   ```

   où les arguments sont respectivement β, le nombre d'intervalles n_x , nombre d'itérations temporelles à effectuer n_t. Les paramètres matériels du problème sont la vitesse de l'onde c et la longueur du domaine L. La condition initiale est fournie dans le tableau `Uo` de taille $n_x + 1$. La fonction renvoie simplement le vecteur `U` qui contient les valeurs de U_i^n obtenues au dernier pas de temps. Il s'agit donc aussi d'un tableau de taille $n_x + 1$.
3. On effectue deux simulations avec $\beta = 0.5, n_t = 80$ et avec $\beta = 1.0, n_t = 40$ respectivement. Les deux simulations fournissent un vecteur qui correspond à même un instant.
   Quelle simulation sera la plus précise ?
4. Finalement, on exécute votre programme avec $\beta = 1.1$.
   Qu'observez-vous ?

Séance 13

Exemples de questions d'examen

Remarque générale : le contenu et le cahier des charges du cours ayant été largement modifiés par les réformes pédagogiques successives, il faut parfois regarder avec un oeil critique certaines questions d'examen des années précédentes : elles sont nettement inférieures ou supérieures aux exigences de compétences de cette année-ci. Avant septembre 2018, le langage utilisé était MATLAB et non pas `python`*. Même si la syntaxe des deux langages (en particulier avec* `numpy`*) est assez proche, il faudrait évidemment effectuer une traduction des programmes demandés, mais cela reste assez semblable en termes d'exigences. Certaines questions d'examens ont été intégrés dans les séances d'exercices et ne sont donc pas reprises ici.*

82 Juin 2001

1. Définir l'erreur de discrétisation d'une méthode numérique et donner deux moyens de la limiter (cinq lignes maximum).
2. En utilisant des développements en série de Taylor, démontrez la relation suivante.

$$u''''(x) = \frac{u(x-2h) - 4u(x-h) + 6u(x) - 4u(x+h) + u(x+2h)}{h^4} + \mathcal{O}(h^2)$$

83 Septembre 2001

Une approximation de la dérivée première d'une fonction u en un point X_i peut être effectuée au moyen d'une différence centrée écrite sous la forme

$$u'(X_i) \approx \frac{U_{i+1} - U_{i-1}}{2h}$$

où h est l'écart entre deux abscisses voisines et U_i est la valeur de la fonction u en un point X_i. Une telle approximation permet de construire une méthode connue sous le nom de *leapfrog method*, définie par la relation :

$$U_{i+1} = U_{i-1} + 2hf(X_i, U_i)$$

et utilisée pour la résolution numérique du problème différentiel à la valeur initiale

$$\left\{ \begin{array}{rcl} u' & = & f(x,u) \\ u(0) & = & U_0 \end{array} \right.$$

On vous demande

1. de dire si cette méthode est une méthode à pas simple ou à pas liés,
2. de dire si on peut directement démarrer la résolution numérique du problème différentiel à la valeur initiale en utilisant la formule de cette méthode pour obtenir U_1,
3. de déterminer l'ordre de précision de la méthode et de justifier votre réponse,
4. de considérer le problème modèle $f(x,u) = \lambda u$ (où λ est une valeur **réelle** négative quelconque) et de déterminer les valeurs **réelles** de $h\lambda$ pour lesquelles la méthode proposée est stable numériquement[1].

84 Septembre 2002

On souhaite calculer la dérivée seconde de la fonction $f(x) = x\ \tan(x)$ en $x = 0.9$, en utilisant uniquement les valeurs (arrondies à quatre chiffres après la virgule) de la table ci-dessous.

x	0.8000	0.8500	0.9000	0.9500	1.0000
$x\tan(x)$	0.8237	0.9676	1.1341	1.3285	1.5574

1. Donnez le polynôme d'interpolation passant par les points de $f(x)$ correspondant aux abscisses 0.8, 0.9, 1.0 et calculez la dérivée *seconde* de ce polynôme en $x = 0.9$.
2. Calculez une estimation numérique de la dérivée *seconde* de $f(x)$ en $x = 0.9$ en calculant une extrapolation de Richardson à partir des valeurs trouvées à l'aide d'une formule de différence centrée d'ordre deux avec deux pas distincts.
3. Donnez l'ordre théorique de l'erreur pour les deux approximations obtenues de la valeur de dérivée seconde. En tenant compte des erreurs d'arrondis des données, peut-on utiliser ces ordres de précision pour en déduire une estimation fiable de l'erreur des approximations ?

Il n'est pas nécessaire de disposer d'une calculatrice pour répondre à cette question! L'expression analytique de la dérivée seconde vaut évidemment $f''(x) = (2 + 2x\tan(x))(1 + \tan(x)^2)$. *Mais, comme vous ne disposez pas de calculatrice, cela ne va pas beaucoup vous aider...*

85 Juin 2004 : Modèle de Maxwell

Le modèle de Maxwell permet de décrire le comportement des matériaux viscoélastiques comme les plastiques, le ketchup ou la pâte dentifrice... Ici, nous souhaitons prédire l'évolution de la contrainte de cisaillement u d'un matériau polymérique qui se trouve au repos et qu'on soumet brutalement à un taux de cisaillement $\dot{\gamma} > 0$ à l'instant $t = 0$. Plus concrètement, on place le matériau entre deux plans parallèles et un des plans est mis soudainement en mouvement à une vitesse constante. Le modèle se réduit au problème de Cauchy

$$\lambda u'(t) + u(t) = \mu\dot{\gamma} \qquad u(0) = 0$$

où $\lambda > 0$ et $\mu > 0$ représentent respectivement un temps de relaxation et une viscosité du matériau.

[1] Indication : il est conseillé d'utiliser le changement de variable $U_i = a^i U_0$ et de déterminer la valeur de a en termes de $h\lambda$ pour effectuer cette analyse de stabilité.

1. Calculer la solution analytique $u(t)$.
2. David souhaite utiliser la méthode d'Euler explicite d'ordre un.

$$U_{i+1} = U_i + hF_i$$

   Pour quelles valeurs de h cette méthode est-elle stable ?
3. Nicolas souhaite utiliser la méthode de Taylor d'ordre trois.

$$U_{i+1} = U_i + h \left[f + \frac{h}{2!} \left. \frac{df}{dt} \right|_{u'=f} + \frac{h^2}{3!} \left. \frac{d^2 f}{dt^2} \right|_{u'=f} \right]_{(T_i, U_i)}$$

   Evaluer les dérivées du membre de droite afin de faire apparaître une relation algébrique ne faisant intervenir que U_i, h, λ, μ et $\dot{\gamma}$.
4. Roman souhaite utiliser la méthode de Petrov-Smacholowski-Birnov :

$$U_{i+1} = -4\ U_i + 5U_{i-1} + h \Big(\ 4F_i + 2F_{i-1} \ \Big)$$

   Quel est l'ordre de précision de l'erreur locale commise à chaque pas de temps ?
   Pour quelles valeurs de h ce schéma est-il stable ?

86 Septembre 2004 : Quadrature de l'été

Vincent souhaite développer une nouvelle méthode d'intégration numérique sur l'intervalle $[0, h]$:

$$\underbrace{\int_0^h u(x)\ dx}_{I} \quad \approx \quad \underbrace{\frac{h}{4} \left(u(0) + \alpha\ u(\frac{2h}{3}) \right)}_{I^h}$$

1. Déterminer α afin que cette quadrature soit exacte pour tout polynôme de degré un.
2. Donner le développement de Taylor d'ordre cinq autour de l'origine pour estimer $u(a)$ à partir des valeurs de la fonction et de ses dérivées à l'origine.
3. En utilisant ce développement avec $a = \frac{2h}{3}$ et $a = x$, déduire l'expression du premier terme d'erreur de la formule de quadrature.
4. Donner l'ordre de précision (l'exposant en h du premier terme d'erreur) de la quadrature obtenue.
5. Donner le degré de précision (le degré des polynômes qui sont intégrés exactement) de la quadrature obtenue.

87 Janvier 2006 : Dérivation numérique

On considère la formule de dérivation numérique suivante

$$u'(x) \quad \approx \quad \frac{u(x-2h) - 8u(x-h) + 8u(x+h) - u(x+2h)}{12h}$$

1. Montrer que cette formule peut être obtenue à partir de la formule centrée d'ordre deux et de l'extrapolation de Richardson.
2. Donner l'ordre de cette formule. Justifier votre réponse.

88 Janvier 2006 : Equation de transport

Pour résoudre numériquement une équation de transport

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x},$$

nous considérons les schémas de Lajos et de Lax :

Lajos (1913)	$\frac{U_j^{n+1} - U_j^n}{\Delta t} = -c\,\frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}$
Lax (1954)	$\frac{2U_j^{n+1} - U_{j-1}^n - U_{j+1}^n}{2\Delta t} = -c\,\frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}$

où j et n sont respectivement les indices spatiaux et temporels.

On souhaite effectuer l'analyse de stabilité sur un domaine infini en considérant l'évolution d'une perturbation quelconque de la forme $U_j^n = U^n \exp(ikX_j)$ et en calculant le module du facteur d'amplification :

$$G = \frac{U^{n+1}}{U^n}.$$

1. Ces schémas sont-ils explicites ou implicites ?
2. Donner l'expression du nombre complexe G pour les deux schémas.
3. Donner la condition de stabilité en termes de Δx et Δt pour les deux schémas[2].

89 Janvier 2007 : Dérivation numérique

x	$-2h$	$-h$	0	h	$2h$
$u(x)$	0.418	0.423	0.426	0.426	0.424

Paul-Emile a calculé une estimation $u''(0)$ à l'origine, en utilisant uniquement les valeurs (arrondies à trois chiffres après la virgule) de la table.

1. Retrouver le résultat de Paul-Emile sachant qu'il a obtenu une valeur approchée de $u''(0)$ avec une extrapolation de Richardson appliquée à une différence centrée d'ordre deux avec deux pas distincts.
2. Donner une expression de la borne de l'erreur (en tenant compte des erreurs d'arrondis) du calcul de Paul-Emile sachant que l'erreur d'une différence centrée d'ordre deux satisfait :

$$|\tilde{E}^h| \leq \frac{4\epsilon}{h^2} + \frac{C_4 h^2}{12} + \frac{C_6 h^4}{360},$$

3. Calculer la valeur optimale de h que notre ami Paul-Emile a sélectionnée en supposant que $C_6 \approx 1$.

[2] Il peut être utile d'utiliser les relations suivantes : $2\cos\alpha = e^{i\alpha} + e^{-i\alpha}$, $2i\sin\alpha = e^{i\alpha} - e^{-i\alpha}$

90 Janvier 2007 : Méthode de Gear

Pour résoudre numériquement le problème $u'(x) = f(x,u)$, nous souhaitons utiliser la méthode bien connue de Gear d'ordre trois définie par l'expression :

$$U_{i+1} = \frac{1}{11}\left(2U_{i-2} - 9U_{i-1} + 18U_i\right) + \frac{6h}{11}\,F_{i+1}$$

1. Quelle est la valeur de p dans l'expression $\mathcal{O}(h^p)$ de l'erreur locale commise à chaque pas de temps ?
2. Démontrer rigoureusement que la formule de Gear est une méthode d'ordre trois.

91 Janvier 2007 : Problème 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 :

$$\left\{\begin{array}{rcl} u'''(x) + u(x)u''(x) & = & 0 \\ \\ u(0) & = & 0 \\ u'(0) & = & 0 \\ \lim_{x\to\infty} u'(x) & = & 1 \end{array}\right.$$

La valeur de u' représente 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. 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.
2. Ecrire une fonction MATLAB

   `function u = blasius(h)`

   qui calcule, par l'usage conjoint de la technique du tir[3] et de la méthode de Heun[4], la solution numérique du problème de Blasius avec le pas constant `h`. Les valeurs $[u(0), u(h), u(2h), u(3h)\ldots]$ seront fournies dans le vecteur `u`. Il n'est évidemment pas possible d'intégrer jusqu'à une distance infinie : on arrêtera donc l'intégration lorsque la variation relative de la valeur de u' sur un pas est inférieure à 1%.

92 Janvier 2008 : Différences finies

Pour estimer la dérivée à l'origine d'une fonction $u(x)$, Jonathan souhaite utiliser la formule :

$$u'(0) \quad \approx \quad \frac{u(-2h) + au(-h) - au(h) - u(2h)}{b\ h}$$

Malencontreusement, il a oublié de noter les valeurs des paramètres réels a et b afin d'obtenir un ordre de précision le plus élevé possible à partir de ces quatre valeurs nodales.

1. Retrouver les valeurs des deux paramètres et l'ordre de la formule. Justifier votre réponse.

[3] La méthode du tir est nécessaire pour imposer la condition à l'infini...

[4] Pour mémoire, la méthode de Heun est définie par $\left\{\begin{array}{rcl} U_{i+1} & = & U_i + \frac{h}{2}(K_1 + K_2) \\ \\ K_1 & = & f(X_i, U_i) \\ K_2 & = & f(X_i + h, U_i + hK_1) \end{array}\right.$

2. Calculer la valeur optimale de h afin de minimiser l'erreur totale, c'est-à-dire, les contributions conjointes de l'erreur de discrétisation et des erreurs d'arrondi dues au calcul en virgule flottante. Plus précisément, nous supposerons que les valeurs de la fonction u sont fournies avec une double précision ($\epsilon = 10^{-16}$) et la valeur absolue de toutes les dérivées de u est toujours inférieure à l'unité[5] pour toute valeur de x. Justifier votre réponse.

93 Janvier 2008 : Courbes de Bézier

On considère les données suivantes pour écrire les équations paramétriques $(x(t), y(t))$ des trajectoires des voitures de Patrick et de Quentin dans le plan comme des courbes de Bézier :

$$\begin{aligned} [T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7] &= [0,0,0,0,1,1,1,1] \\ [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] &= [(3,3),(2,2),(\alpha,\alpha),(0,0)] \\ [\mathbf{Q}_0, \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3] &= [(0,0),(-1,0),(0,2),(1,0)] \end{aligned}$$

où α est un paramètre réel positif[6].

1. Donner l'expression des quatre fonctions de base $B_i^3(t)$ définies pour les huit noeuds ci-dessus. Il s'agit -ici- des polynômes de Bernstein. Esquisser graphiquement l'allure de ces quatre fonctions.
2. Donner l'expression paramétrique des deux trajectoires définies par les expressions.

$$\underbrace{(x_p(t), y_p(t))}_{\mathbf{p}(t)} = \sum_{i=0}^{3} B_i^3(t)\ \mathbf{P}_i \quad \text{et} \quad \underbrace{(x_q(t), y_q(t))}_{\mathbf{q}(t)} = \sum_{i=0}^{3} B_i^3(t)\ \mathbf{Q}_i \qquad 0 \le t \le 1$$

3. Calculer l'endroit où se trouvent Patrick et Quentin au temps $t = 0.5$.
   La position de Patrick dépendra évidemment de la valeur du paramètre α.
4. Calculer le(s) point(s) du plan (x, y) où les deux courbes se croisent.
5. Pour quelle valeur du paramètre α, observera-t-on une collision entre les deux voitures ?

94 Janvier 2008 : Equation de la chaleur

On souhaite résoudre le problème aux conditions aux limites suivant :

$$\frac{\partial u}{\partial t}(x,t) = \alpha \frac{\partial^2 u}{\partial x^2}(x,t)$$

avec $x \in [0,1]$ et $t \in [0,10]$ exprimés en centimètres et secondes. La valeur numérique de la diffusivité thermique α est unitaire. La valeur de la condition initiale est donnée par la relation $u(x,0) = x$. Aux instants $t > 0$, on impose simultanément une inversion des températures aux deux extrémités du domaine : $u(0,t) = 1$ et $u(1,t) = 0$. Afin d'obtenir une solution numérique, nous introduisons une valeur nodale U_j^n pour chaque instant $T_n = n\ \Delta t$ et position $X_j = j\ \Delta x$, afin d'obtenir une expression du type :

$$U_j^{n+1} = U_j^n + \beta\Big(U_{j+1}^n + U_{j-1}^n - 2U_j^n\Big) \tag{13.1}$$

qui permet d'obtenir n'importe quelle valeur nodale au temps $n+1$ à partir des valeurs du temps n.

[5] Oui, il existe des fonctions qui satisfont une telle condition, par exemple cos ! Maintenant, on est bien d'accord que vous connaissez la dérivée analytique du cosinus et que cela n'a strictement aucun intérêt de la calculer numériquement. Mais il faut bien simplifier un peu...

[6] Vraisemblablement, il n'est pas totalement inutile de représenter graphiquement les points de contrôle et d'esquisser les deux trajectoires (par exemple, pour $\alpha = 0$) avant de se plonger aveuglément dans une algèbre calculatoire et fastidieuse...

1. Donner les unités de la diffusivité thermique et exprimer β en termes de α, Δx et Δt.
2. Déduire la condition de stabilité que doit satisfaire β en obtenant une estimation du facteur d'amplification d'une perturbation quelconque de la forme

$$U_j^n = U^n e^{ikX_j}$$

   La relation $2\cos(\theta) = e^{i\theta} + e^{-i\theta}$ pourrait être utile. Le caractère i représente le nombre imaginaire !
3. Ecrire une fonction MATLAB `edpEuler(m)` qui utilise la relation (13.1) pour afficher la courbe d'une solution discrète de `m` valeurs nodales spatiales à tous les centièmes de seconde entre 0 et 0.1 seconde.
4. Ecrire une fonction MATLAB `edpOde45(m)` qui résout le même problème en remplaçant la méthode d'Euler explicite par une utilisation adéquate de la fonction `ode45`.

```
ODE45  Solve non-stiff differential equations, medium order method.
   [TOUT,YOUT] = ODE45(ODEFUN,TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates  the system of differential
   equations y' = f(t,y) from time T0 to TFINAL with initial conditions Y0. ODEFUN is a function handle.
   For a scalar T and a vector Y, ODEFUN(T,Y) must return a column vector corresponding to f(t,y).
   Each row in the solution array YOUT corresponds to a time returned in the column vector TOUT.
   To obtain solutions at specific times T0,T1,...,TFINAL (all increasing or all decreasing),
   use TSPAN = [T0 T1 ... TFINAL].

   Example [t,y]=ode45(@vdp1,[0 20],[2 0]);  plot(t,y(:,1));
           solves the system y' = vdp1(t,y), and plots the first component of the solution.
```

95 Janvier 2009 : Herman ou l'intégration numérique

Herman vient d'introduire une nouvelle méthode composite de crise pour effectuer l'intégration numérique d'une fonction sur un intervalle $[a, b]$. On partage cet intervalle en n sous-intervalles égaux dont la longueur vaudra $2h$. Sur chaque sous-intervalle, on utilise la règle d'Herman définie par :

$$\int_{-h}^{h} u(x)\, dx \;\approx\; \frac{h}{2}\Big(3U_{-h/3} + U_h\Big) \tag{13.2}$$

En utilisant 10 et 20 sous-intervalles pour intégrer une fonction $u(x)$ sur l'intervalle $[a, b]$, Herman a obtenu H_{2h} et H_h respectivement. Herman ignore quasiment tout de cette mystérieuse fonction $u(x)$. Toutefois, Didier, Elio et Joëlle lui ont dit qu'il existe des constantes C_i qui bornent les valeurs absolues de la dérivée i-ème sur l'intervalle considéré.

1. En intégrant l'erreur d'interpolation[7] du polynôme pour les abscisses $x = -h/3$ et $x = h$, obtenir l'ordre de précision de la méthode composite d'Herman et l'expression de l'erreur.
2. Définir le degré de précision d'une méthode numérique d'intégration : quel est le degré de précision de la méthode composite d'Herman ?
3. Calculer α et β afin que la combinaison linéaire $H_{extr} = \alpha H_{2h} + \beta H_h$ fournisse la meilleure estimation possible de l'intégrale exacte.

[7] L'erreur du polynôme d'interpolation pour les abscisses $X_0 < X_1 < \ldots < X_n$ est donnée par l'expression :

$$e^h(x) = \frac{C_{(n+1)}}{(n+1)!}\,(x - X_0)(x - X_1)(x - X_2)\cdots(x - X_n).$$

96 Janvier 2009 : Equation aux dérivées partielles

Il s'agit de résoudre l'équation aux dérivées partielles :

$$\frac{\partial u}{\partial t}(x,y,t) = \gamma \frac{\partial^2 u}{\partial x \partial y}(x,y,t),$$

avec $(x,y) \in [0,1] \times [0,1]$ et $t \in [0,10]$ exprimés en mètres et secondes. La valeur numérique du paramètre γ est unitaire et la fonction $u(x)$ représente une énergie. A l'instant initial $t = 0$, nous avons

$$u(x,y,0) = x^2 + y^2.$$

Aux instants ultérieurs $t > 0$, on maintient cette valeur uniquement sur les côtés du domaine carré. Afin d'obtenir une solution numérique approchée, nous introduisons une valeur nodale $U^n_{j,k}$ pour chaque instant $T_n = n\ \Delta t$ et position $(X_j, Y_k) = (j\ \Delta x, k\ \Delta x)$, et nous écrivons la relation de récurrence :

$$U^{n+1}_{j,k} \quad = \quad U^n_{j,k} + \beta\Big(U^n_{j+1,k+1} + U^n_{j-1,k-1} - U^n_{j+1,k-1} - U^n_{j-1,k+1}\Big) \tag{13.3}$$

qui permet d'obtenir n'importe quelle valeur nodale au temps $n+1$ à partir des valeurs du temps n.

1. Donner les unités du paramètre γ.
2. Donner l'expression de β en termes de γ, Δx et Δt.
3. Ecrire une fonction MATLAB

   `edpEuler(m,n,beta)`

   qui utilise la relation (13.3) et retourne une matrice contenant les m^2 valeurs nodales obtenues après avoir effectué n itérations en utilisant la valeur de β fournie en argument. Soyez bien attentifs à définir correctement les tailles des matrices et les valeurs limites dans les éventuelles boucles de votre petit programme.
4. Quelles valeurs de β peut-on choisir afin d'obtenir un comportement stable? Répondre à cette question en obtenant une estimation du facteur d'amplification d'une perturbation quelconque de la forme

   $$U^n_{j,k} = U^n e^{ik_x X_j} e^{ik_y Y_k}.$$

   La relation $2i\sin(\theta) = e^{i\theta} - e^{-i\theta}$ pourrait être utile. Le caractère i représente le nombre imaginaire!

97 Janvier 2010 : Approximation de Fourier

Nous disposons d'un ensemble de huit données d'une fonction inconnue $u(x)$ dont nous souhaitons calculer une approximation au sens des moindres carrés $u^h(x)$ à partir de trois fonctions cosinus

$$u^h(x) = \sum_{i=1}^{3} a_i \cos(ix)$$

	X_i	U_i
0	0	2
1	$\pi/4$	1
2	$\pi/2$	0
3	$3\pi/4$	2
4	π	1
5	$5\pi/4$	1
6	$3\pi/2$	0
7	$7\pi/4$	0

1. Ecrire la fonction $J(a_1, a_2, a_3)$ qu'il faut minimiser pour résoudre un tel problème.
2. En déduire le système à résoudre pour obtenir les trois coefficients a_i.
3. Calculer les valeurs des coefficients a_i.

4. Compléter le programme MATLAB ci-dessous afin d'obtenir la courbe de l'approximation.

```
X = [0 1 2 3 4 5 6 7] * pi / 4;
U = [2 1 0 2 1 1 0 0];
x = linspace(0,2*pi,100);

... == à compléter == ...

plot(x,uh);
```

98 Janvier 2010 : Méthode de Steffensen

Afin de trouver la solution d'un problème non linéaire, nous allons comparer les performances des méthodes de Newton-Raphson, de la sécante et de Steffensen.

		Taux de convergence
méthode de Newton-Raphson	$x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$	2
méthode de la sécante	$x_{i+1} = x_i - \dfrac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$	1.618
méthode de Steffensen	$x_{i+1} = x_i - \dfrac{f(x_i)f(x_i)}{f(x_i + f(x_i)) - f(x_i)}$	r

1. Définir le taux de convergence d'une méthode itérative.
2. Calculer le taux de convergence de la méthode de Steffensen, en justifiant rigoureusement votre réponse avec une petite démonstration[8].
3. Comparer les *taux de convergence des trois méthodes par nombre d'évaluation* de f (ou f'). Quelle est la méthode dont le taux de convergence par évaluation de fonction est le plus élevé?
4. Ecrire le code MATLAB de `function x = steffensen(x,tol,nmax,f)` implémentant la méthode de Steffensen pour trouver une racine d'une fonction `f` avec une tolérance `tol` et un nombre maximal d'itérations `nmax`. Le candidat initial est `x`.

99 Janvier 2010 : Méthodes BDF

Pour intégrer une équation différentielle ordinaire

$$u'(x) = f(x, u(x)),$$

nous souhaitons utiliser des formules de différentiation rétrograde (notées BDF) définies par :

$$U_{i+1} = \sum_{j=0}^{p} a_j U_{i-j} + hb \underbrace{f(X_{i+1}, U_{i+1})}_{F_{i+1}}.$$

[8] Vous pouvez toutefois considérer comme résultat acquis les taux de convergence de la méthode de la sécante et de celle de Newton-Raphson fournis dans l'énoncé. Il ne faut donc PAS redémontrer que le taux de convergence de la méthode de Newton-Raphson est quadratique.

Ce sont des méthodes multi-pas implicites dont les coefficients a_j et b sont calculés en approchant directement la valeur de la dérivée première de u au noeud X_{i+1} par la dérivée première du polynôme interpolant u aux $p+2$ noeuds $X_{i+1}, X_i, \ldots, X_{i-p}$ avec $p \geq 0$. Le pas entre deux abscisses est noté h. Les méthodes BDF les plus simples ($p = 0$ et $p = 1$) sont définies par le tableau.

	a_0	a_1	a_2	b
$p=0$	1	0	0	1
$p=1$	$\frac{4}{3}$	$-\frac{1}{3}$	0	$\frac{2}{3}$

1. Donner l'expression du polynôme d'interpolation de u aux 4 noeuds $X_{i+1}, X_i, X_{i-1}, X_{i-2}$.
2. Calculer la dérivée de ce polynôme en $x = X_{i+1}$ en termes de $U_{i+1}, U_i, U_{i-1}, U_{i-2}$ et h. En déduire les coefficients a_0, a_1, a_2 et b pour la méthode BDF2 ($p = 2$).
3. Quel est l'ordre de précision de la méthode BDF2 ?
4. Compléter le programme MATLAB ci-dessous afin d'obtenir la zone de stabilité[9] du plan complexe de $h\lambda$ de la méthode BDF2.

```
[x,y]=meshgrid([-8:0.05:8],[-8:0.05:8]);
z = x+i*y;

... == à compléter == ...

contourf(x,y,-alpha,[-1:0.1:0]); grid;
```

Noter que la commande MATLAB `roots` calcule toutes les racines d'un polynôme quelconque.

```
ROOTS  Find polynomial roots.
  ROOTS(C) computes the roots of the polynomial whose coefficients
  are the elements of the vector C. If C has N+1 components,
  the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).
```

100 Janvier 2011 : Approximation de Bézier

	T_i	U_i
0	0	0
1	1/3	1
2	2/3	0
3	1	2

Nous disposons d'un ensemble de quatre données d'une fonction inconnue $u(t)$ sur l'intervalle $[0, 1]$. Nous souhaitons calculer une approximation au sens des moindres carrés $u^h(t)$ à partir de trois fonctions de Bézier

$$u^h(t) = \sum_{i=0}^{2} a_i B_i^2(t)$$

1. Donner l'expression des trois fonctions de Bézier $B_i^2(t)$ définies[10]

[9] En effectuant l'analyse du problème modèle $u' = \lambda u$.

[10] On définit les B-splines à partir de $n+1$ noeuds $T_0 \leq T_1 \leq T_2 \leq \ldots \leq T_n$.
Une fonction $B_i^p(t)$ est nulle sauf dans l'intervalle $[\, T_i, T_{i+1+p} \,[$ où elle est définie par la relation de récurrence :

$$B_i^p(t) = \frac{(t - T_i)}{(T_{i+p} - T_i)} B_i^{p-1}(t) + \frac{(T_{i+1+p} - t)}{(T_{i+1+p} - T_{i+1})} B_{i+1}^{p-1}(t)$$

avec $i = 0, \ldots, n-p-1$ et en partant de :

$$B_i^0(t) = \begin{cases} 1, & \text{si } \; t \in [T_i T_{i+1}[\\ 0, & \text{ailleurs} \end{cases}$$

On observe immédiatement que pour des noeuds de multiplicité supérieure à un, la formule de récurrence peut faire apparaître une valeur nulle à l'un ou l'autre dénominateur. On complète donc la définition en spécifiant qu'il ne faut tenir compte que des termes dont les dénominateurs ne s'annulent pas. Les fonctions de Bézier sont juste des B-splines avec un choix particulier de noeuds.

pour les six noeuds $[T_0, T_1, T_2, T_3, T_4, T_5] = [0, 0, 0, 1, 1, 1]$.
Esquisser graphiquement l'allure des trois fonctions.

2. Ecrire la fonction $J(a_0, a_1, a_2)$ qu'il faut minimiser pour résoudre un tel problème.
3. Pour obtenir l'approximation polynomiale d'ordre deux, est-il mieux d'utiliser les fonctions de Bézier ou de choisir les trois monômes 1, x et x^2 comme fonctions de base ?
Justifier brièvement votre réponse !
4. Compléter le programme MATLAB ci-dessous afin d'obtenir la courbe de l'approximation.

```
T = [0 1 2 3] /3;
U = [0 1 0 2];
t = linspace(0,1,100);

... == à compléter == ...
plot(t,uh);
```

101 Janvier 2011 : Méthode d'ordre quatre de Johan

On se propose de déterminer une subtile solution numérique du délicat problème de Johan :

$$\begin{cases} u''(x) &= f(x), \\ \\ u(0) &= 1, \\ u(1) &= 1. \end{cases}$$

On recherche les valeurs nodales inconnues $U_i = u^h(X_i)$ aux abscisses $X_i = ih$ avec $h = 1/n$.
Les deux valeurs frontières sont évidemment connues : $U_0 = U_n = 1$.
En définissant $F_i = f(X_i)$, Johan suggère d'utiliser les équations suivantes :

$$\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} = \frac{F_{i-1} + \alpha F_i + F_{i+1}}{\gamma} + \mathcal{O}(h^4)$$

1. En observant que $F_i = U_i''$, trouver α et γ afin que la méthode numérique soit d'ordre quatre[11].
Justifier votre réponse !
2. Pour une fonction f quelconque, écrire le programme MATLAB :

```
function u = johan(n,f)
```

qui calcule les `n+1` valeurs nodales par la méthode numérique de Johan.

102 Janvier 2011 : Schéma symplectique de Bart

Bart souhaite résoudre le système d'équations différentielles ordinaires :

$$\underbrace{\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix}}_{\mathbf{u}'(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}}_{\mathbf{A}\mathbf{u}(t)},$$

[11] Il s'agit bien de l'ordre quatre et *pas de l'ordre deux !* La méthode de Johan n'est donc pas :

$$\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2} = F_i + \mathcal{O}(h^2).$$

Cela serait bien trop simple et cela ne satisferait ni Johan, ni Albert, ni Elio, ni Bart et encore moins le correcteur !

avec $x(0) = 1$ et $y(0) = 0$. Bart se propose d'utiliser le schéma suivant :

$$\mathbf{U}_{i+1} = \mathbf{U}_i + h\ (\beta \mathbf{A}\mathbf{U}_{i+1} + (1-\beta)\mathbf{A}\mathbf{U}_i).$$

1. Calculer les deux valeurs propres λ_i de la matrice $\mathbf{A}$.
2. Donner la solution analytique et observer que celle-ci reste toujours sur le cercle de rayon unitaire.
3. Pour quelles valeurs de β, la méthode de Bart est-elle explicite ?
4. Esquisser les zones de stabilité numérique pour $\beta = 0$, $\beta = 1$ et $\beta = \frac{1}{2}$ dans le plan complexe $h\lambda$.
5. Que va-t-on observer si on résout ce système avec la méthode d'Euler explicite ? La solution numérique va-t-elle rester sur le cercle, osciller autour du cercle, s'éloigner vers l'infini ou tendre vers zéro lorsque le temps tend vers l'infini ?
   Et avec la méthode d'Euler implicite, que va-t-il se passer ?
6. Identifier β afin que la solution discrète reste sur le cercle unitaire à tout instant.
   On dira alors que le schéma est un intégrateur symplectique pour le système.
7. Quel est l'ordre de cette méthode symplectique de Bart ?

103 Janvier 2012 : Une bête équation aux dérivées partielles

On souhaite résoudre l'équation aux dérivées partielles :

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 u}{\partial y^2} = 1$$

avec $(x, y) \in [0, 1] \times [0, 1]$. On impose que $u(x, y) = 0$ sur les côtés du domaine.
Afin d'obtenir une solution numérique, définissons :

$$U_{i,j} \approx u(X_i, Y_j)$$

pour $X_i = ih$ et $Y_j = jh$ avec $h = 1/n$.

1. Ecrire les équations que doivent satisfaire $U_{i,j}$ pour des différences finies centrées d'ordre deux.
2. Ecrire une fonction MATLAB

   ```
   function u =  edp(n)
   ```

   qui retourne une matrice contenant les valeurs nodales $U_{i,j}$.
   Soyez bien attentifs à définir et initialiser les matrices correctement.

104 Janvier 2012 : Newton, Raphson et Euler tous ensemble !

On considère le problème non linéaire de Cauchy :

$$\begin{cases} u'(t) &= -\Big(u(t)\Big)^3 + cos(t), \\ u(0) &= 0. \end{cases}$$

On souhaite utiliser la méthode d'Euler implicite pour les instants $T_n = nh$ avec $n = 0, 1, 2, \ldots, m$. Comme le schéma d'Euler est implicite, il faut utiliser une méthode itérative pour résoudre le problème non linéaire à chaque pas. Ici, on se propose de faire appel à la méthode de Newton-Raphson.

1. Donner la relation (non linéaire) liant U^{n+1} et U^n pour le schéma d'Euler implicite.
2. Ecrire l'itération de Newton-Raphson donnant U_{i+1}^{n+1} à partir de U_i^{n+1}.
   Le candidat initial est $U_0^{n+1} = U^n$. L'indice i réfère aux itérations du schéma de Newton-Raphson.
3. Calculer les trois premières approximations U_0^1, U_1^1 et U_2^1 pour le premier pas d'Euler implicite U^1.
4. Proposer un critère d'arrêt du schéma de Newton-Raphson.
   Justifier votre choix[12].
5. Ecrire une fonction MATLAB
   `function u = eulerImplicite(h,m)`
   qui calcule, par l'usage conjoint de la technique de la méthode de Newton-Raphson et de celle d'Euler implicite, la solution numérique du problème de Cauchy pour `m` pas de taille `h`.
   Les valeurs $[u(0), u(h), u(2h), u(3h) \dots u(mh)]$ seront fournies dans le vecteur `u`.

105 Janvier 2012 : Trajectoire d'un monstre maléfique

Afin d'annihiler un monstre maléfique, il s'agit de calculer le paramètre α. Sur l'intervalle de temps $t \in [T_2, T_4] = [-1, 1]$, la trajectoire du monstre est donnée par :

	X_i	Y_i	Z_i
0	4	0	0
1	4	0	α
2	-8	16	α
3	0	4	0

$$\begin{cases} x(t) &= \sum_{i=0}^{3} B_i^2(t) X_i \\ y(t) &= \sum_{i=0}^{3} B_i^2(t) Y_i \\ z(t) &= \sum_{i=0}^{3} B_i^2(t) Z_i \end{cases}$$

où $B_i^2(t)$ sont les fonctions B-splines[13] pour les noeuds $\mathbf{T} = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6\} = \{-1, -1, -1, 0, 1, 2, 3\}$.

1. Donner la position du monstre en $t = 0$.
2. Donner les expressions de $B_0^2(t)$, $B_1^2(t)$, $B_2^2(t)$ et $B_3^2(t)$ pour $t \in [T_2, T_3] = [-1, 0]$. Afin d'éviter de vous égarer dans une algèbre purement calculatoire, nous vous donnons les expressions de toutes les autres fonctions sur les deux intervalles utiles avec $\mathbf{T} = \{-1, -1, -1, 0, 1, 2, 3\}$:

	$t \in [-1, 0[$	$t \in [0, 1[$		$t \in [-1, 0[$	$t \in [0, 1[$
$B_0^1(t) =$	0	0	$B_0^2(t) =$	...	0
$B_1^1(t) =$	$-t$	0	$B_1^2(t) =$	...	$(1 - 2t + t^2)/2$
$B_2^1(t) =$	$(t + 1)$	$(1 - t)$	$B_2^2(t) =$	...	$(1 + 2t - 2t^2)/2$
$B_3^1(t) =$	0	t	$B_3^2(t) =$	...	$t^2/2$

[12] Il n'y a pas de réponse unique à cette question : il suffit donc de justifier votre choix avec un minimum de bon sens !

[13] La définition de fonctions B-splines est fournie à toute fin utile :-)
Soit les $n + 1$ noeuds $T_0 \leq T_1 \leq T_2 \leq \dots \leq T_n$.
Les B-splines de degré p sont les $n - p$ fonctions $B_i^p(t)$. Une fonction B_i^p est nulle sauf dans l'intervalle $[\, T_i, T_{i+1+p} \,[$ où elle est définie par la relation de récurrence :

$$B_i^p(t) = \frac{(t - T_i)}{(T_{i+p} - T_i)} B_i^{p-1}(t) + \frac{(T_{i+1+p} - t)}{(T_{i+1+p} - T_{i+1})} B_{i+1}^{p-1}(t)$$

avec $i = 0, \dots, n - p - 1$ et en partant de : $B_i^0(t) = \begin{cases} 1, & \text{si } t \in [T_i T_{i+1}[\\ 0, & \text{ailleurs} \end{cases}$

3. Esquisser les fonctions $B_0^2(t)$, $B_1^2(t)$, $B_2^2(t)$ et $B_3^2(t)$ pour $t \in [T_2, T_4] = [-1, 1]$.
4. Calculer la valeur de α afin que $z(1/2) = 8$.

106

Janvier 2013 : Hervé s'emberlificote avec Heun

En faisant appel à la méthode de Heun, Hervé souhaite résoudre numériquement le problème :

$$\begin{cases} x'(t) &= -10\ x(t) - 9\ y(t), \\ y'(t) &= -9\ x(t) - 10\ y(t), \\ \\ x(0) &= 2, \\ y(0) &= 0. \end{cases}$$

Pour rappel, la méthode de Heun est définie comme suit :

$$\begin{aligned} U_{i+1} &= U_i + \tfrac{h}{2}(K_1 + K_2) \\ \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + h, U_i + hK_1) \end{aligned}$$

1. Est-ce que le problème d'Hervé est stable ?
   Justifier votre réponse.
2. Donner l'expression analytique de la région de stabilité de la méthode de Heun.
   Esquisser cette région dans le plan complexe en définissant précisément les axes de la figure.
   Y indiquer aussi la zone de stabilité de la méthode d'Euler explicite.
   Ici, on considère évidemment le problème modèle habituel : $u'(t) = \lambda\ u(t)$.
3. Obtenir la contrainte de stabilité à imposer sur le pas de temps de Heun pour le problème d'Hervé.
4. Ecrire une fonction MATLAB

   ```
   function [x y] =  HeunHerve(n,dt)
   ```

   qui retourne deux vecteurs de taille `n+1` qui contiennent les différentes valeurs de $x(t)$ et $y(t)$ obtenues en effectuant n pas avec la méthode de Heun. On commence le calcul à partir de la condition initiale en $t = 0$. Le pas de temps est donné par le second argument `dt`.

107

Janvier 2013 : Stable ou pas stable ?

Pour résoudre numériquement une équation de transport

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x},$$

où c est une constante strictement positive, Francis considère le schéma numérique suivant :

$$\boxed{\frac{U_j^{n+1} - U_j^n}{\Delta t} = -c\ \frac{U_j^n - U_{j-1}^n}{\Delta x}}$$

où j et n sont respectivement les indices spatiaux et temporels.

On va effectuer l'analyse de stabilité en considérant l'évolution d'une pertubation quelconque de la forme $U_j^n = U^n \exp(ikX_j)$ et en calculant le carré du module $|G|^2$ du facteur d'amplification :

$$G = \frac{U^{n+1}}{U^n}.$$

1. Que représente k dans l'expression $U_j^n = U^n \exp(ikX_j)$?
2. Ce schéma temporel est-il explicite ou implicite ?
3. Donner l'expression du nombre complexe G pour ce schéma.
4. En déduire la condition de stabilité en termes de c, Δx et Δt pour ce schéma.

108 Janvier 2013 : Différences finies compactes

Christian a découvert le concept de différences finies compactes :

$$\alpha U'_{i-1} + U'_i + \alpha U'_{i+1} \quad = \quad \beta \frac{U_{i+1} - U_{i-1}}{2h} + \gamma \frac{U_{i+2} - U_{i-2}}{4h} + \mathcal{O}(h^p)$$

Cela lui permet d'obtenir une estimation très précise des valeurs $U'_i \approx u'(X_i)$ de la dérivée d'une fonction périodique à partir des ordonnées $U_i = u(X_i)$ aux abscisses $X_i = ih$ avec $i = 1 \dots n$.

La périodicité implique évidemment $U_1 = U_n$ et $U'_1 = U'_n$.

Malencontreusement, un informaticien facétieux a chapardé les valeurs des 3 coefficients α, β et γ.

1. Quel est l'ordre de précision le plus élevé que l'on aura avec les α, β et γ les plus adéquats ? Justifier brièvement.
2. Quelles relations doivent satisfaire α, β et γ pour obtenir la méthode la plus précise ?
3. Calculer les valeurs optimales de α, β sachant que $\gamma = \frac{1}{9}$.
4. Ecrire une fonction MATLAB

   `function [dU] = compactDerivative(U,alpha,beta,gamma,h)`

   qui calcule le vecteur U'_i à partir des données U_i.
   Le résultat sera fourni dans `dU` dont la taille sera celle de `U`.
   Les coefficients α, β et γ ainsi que h sont donnés en argument d'entrée.

109 Janvier 2014 : It is a piece of integration cake :-)

Pour estimer l'intégrale $I = \int_{-h}^{h} u(x)dx$, comparons les méthodes des trapèzes et du point milieu :

$$I^h_{trapeze} = h\big(U_{-h} + U_h\big) \quad \approx \quad I$$

$$I^h_{midpoint} = 2hU_0 \quad \approx \quad I$$

1. Ecrire les sept premiers termes du développement de Taylor de la fonction $u(x)$ autour de l'origine.
2. En intégrant ce développement, obtenir l'ordre de précision de la méthode composite du point milieu et l'expression de l'erreur en termes des dérivées successives de u à l'origine..
3. En procédant de la même manière, obtenir l'ordre de précision et l'erreur pour les trapèzes.
4. Calculer α et β afin que la combinaison linéaire

   $$I_{extr} = \alpha I^h_{trapeze} + \beta I^h_{midpoint}$$

   fournisse la meilleur estimation possible de l'intégrale exacte.
5. Donner l'expression de l'erreur de I_{extr}.

110 Janvier 2014 : Méditation rectorale

Nous connaissons trois points de la trajectoire d'un candidat recteur qui médite en parcourant toutes les 4 secondes la même boucle dans la salle du Sénat académique des Halles universitaires.

En $t = 0$, il se trouve au premier point, puis au second et au troisième en $t = 1$ et $t = 3$ respectivement. Et finalement, il repasse au point initial en $t = 4$ et continuera à répéter indéfiniment la même trajectoire...

	t_i	(X_i, Y_i)
0	0.00	(0.00 , 0.00)
1	1.00	(2.00 , 1.00)
2	3.00	(0.00 , 1.00)
3	4.00	(0.00 , 0.00)

Nous allons estimer la représentation paramétrique C^0 de cette trajectoire avec une interpolation polynomiale et une interpolation C^2 avec des splines cubiques usuelles.

1. Esquisser les quatre polynômes de Lagrange $\phi_i(t)$ associés à $T_0 = 0$, $T_1 = 1$, $T_2 = 3$ et $T_3 = 4$.
2. Développer l'expression de l'interpolation polynomiale paramétrique : $(x(t), y(t))$.
   En déduire ensuite la position atteinte en $t = \frac{1}{2}$.
3. Compléter ce programme MATLAB pour tracer la trajectoire rectorale avec une interpolation C^2 avec des splines cubiques.

```
function theRectorMeditationCurve()

% 4 ou 5 lignes à ajouter par vos bons soins !

t = linspace(0,4,100);
plot(spline(T,X,t),spline(T,Y,t),'-r'); end
```

111 Janvier 2014 : Une convergence en or !

Considérons la méthode de la sécante pour la recherche d'une racine x simple : nous avons donc $f'(x) \neq 0$.

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

L'erreur commise à chaque itération i est définie par $e_i = x_i - x$.

1. Ecrire le code MATLAB de `function x = secante(x0,x1,tol,nmax,f)` implémentant la méthode de la sécante pour trouver une racine d'une fonction `f` avec une tolérance `tol` et un nombre maximal d'itérations `nmax`. Deux premières valeurs pour lancer la méthode itérative sont données dans `x0` et `x1` .
2. Définir le taux de convergence α d'une méthode itérative.
3. Calculer ce taux de convergence sachant qu'il existe une constante réelle D telle que

   $$e_{i+1} = D\ e_i e_{i-1}$$

   lorsque la méthode de la sécante converge et que $i \to \infty$.
4. Ensuite, calculer
   l'expression de la constante D en termes de $f'(x)$ et $f''(x)$.

112 Janvier 2015 : Un petit logarithme :-)

Pour obtenir une estimation de $\ln(2)$, nous allons calculer

$$I = \int_a^b \frac{1}{x}\, dx$$

en utilisant la méthode composite des trapèzes avec $[a, b] = [1, 2]$.

1. Démontrer que la règle d'intégration du trapèze sur un intervalle $[a, b]$ correspond à l'intégration d'une interpolation polynomiale $u^h(x)$ de la fonction à intégrer $u(x)$.
2. Obtenir une jolie fraction proche de $\ln(2)$ en utilisant cette quadrature avec deux intervalles.
3. Sachant que l'erreur d'intégration de la méthode composite des trapèzes satisfait la relation

$$|E^h| \leq \frac{C_2(b-a)}{12}\, h^2,$$

calculer[14] le nombre d'intervalles requis à utiliser pour obtenir une erreur absolue inférieure ou égale à 10^{-8} pour le calcul d'une estimation de $\ln(2)$. Justifier votre réponse !

113 Janvier 2015 : Charles veut faire de l'ordre trois !

Charles veut résoudre une équation différentielle de degré trois :

$$u'''(t) - 2u''(t) + u'(t) = \cos(t)$$

avec les conditions initiales $u(0) = u'(0) = u''(0) = 1$.

1. Ecrire ce problème comme un système de trois équations différentielles du premier ordre.
   Les trois fonctions inconnues seront notées $u(x)$, $v(x)$ et $w(x)$ respectivement.
   Préciser les trois conditions initiales.
2. Ecrire les relations de récurrence permettant d'obtenir U_{i+1}, V_{i+1} et W_{i+1} à partir de U_i, V_i et W_i pour une méthode d'Euler explicite avec un pas de temps h. Les données initiales sont notées U_0, V_0 et W_0. Le temps correspondant à l'itération i est $T_i = ih$.
3. Ajouter les termes adéquats à ces trois relations pour avoir une méthode de Taylor d'ordre deux.
4. Ecrire le programme MATLAB qui effectue `n` itérations de la méthode de Taylor d'ordre deux. Le vecteur `T` de taille $n + 1$ contiendra l'ensemble de temps discrets et la matrice `U` de taille $3 \times (n + 1)$ contiendra l'ensemble des valeurs discrètes des fonctions u, v et w.

   ```
   function [T,U] = taylorSystem(n,h)
   ```

   Même si vous n'avez pas tous les termes requis dans la question précédente, il est possible d'écrire la plus grande partie réellement utile de ce programme !
5. (**) Ecrire les relations de récurrence de la méthode de Taylor d'ordre trois fera le bonheur complet de Charles !

[14] A toutes fins utiles, il peut être judicieux de savoir que $1/\sqrt{(2)} = 0.7071$ ou $1/\sqrt{(3)} = 0.5774$ ou encore $1/\sqrt{(6)} = 0.4082$... Enfin, peut-être que c'est aussi totalement inutile :-)

114 Janvier 2015 : Une méthode spéciale pour des intégrales pondérées

Considérons l'intégrale pondérée d'une fonction u quelconque multipliée par le cosinus. En d'autres mots, on peut dire que le cosinus est *la fonction de pondération*

Il s'agit maintenant de trouver des méthodes numériques particulières pour ce type d'intégrale en écrivant une formule de quadrature sous la forme :

$$\int_{-\pi/2}^{\pi/2} u(x)\ \cos(x)\ dx\ =\ I\ \approx\ I^h\ =\ a_1\ u(X_1) + a_2\ u(X_2)$$

où les deux abscisses X_1 et X_2, ainsi que les deux poids a_1 et a_2 doivent être choisis judicieusement. Les bornes d'intégration sont toujours $-\pi/2$ et $\pi/2$.

Le degré de précision pondérée de ces méthodes numériques est défini comme le nombre entier positif d tel que l'erreur d'intégration pondérée d'un polynôme u soit nulle pour tous les polynômes de de degré inférieur ou égal à d mais soit non nulle pour au moins un polynôme de degré $d+1$.

1. Montrer qu'il est impossible que le degré de précision pondérée de la méthode proposée puisse être supérieur ou égal à quatre.
2. Déterminer les valeurs des poids et des points pour que le degré de précision pondérée soit trois.
3. Ecrire un programme MATLAB qui implémente la méthode composite correspondante pour une fonction `u` et un nombre de sous-intervalles `n`. Les autres arguments `a` et `X` sont des vecteurs de taille deux contenant les poids et les abscisses de la méthode. Il est donc possible de réussir cette sous-question en ayant échoué à la précédente.

```
function weightedIntegral(u,n,a,X)
```

115 Janvier 2016 : Orbite d'une planète mystérieuse

Une planète se déplace suivant une orbite décrite par un paramètre inconnu a. On dispose toutefois de $n = 3$ mesures approchées du rayon de l'orbite R_i pour diverses valeurs de l'angle Θ_i et on connaît la relation exacte liant le rayon et l'angle de l'orbite :

$$r(\theta) = \frac{10}{6 - a\cos(\theta) + \cos(\theta)}$$

Θ_i	R_i
0	5
$\frac{\pi}{2}$	$\frac{5}{3}$
π	0

On veut obtenir la meilleure estimation de a afin de minimiser la somme du carrés des n écarts $R_i - r(\Theta_i)$.

1. Ecrire l'équation $K(a) = 0$ que l'on doit annuler pour obtenir a en termes des n données.
2. Mettre en oeuvre la méthode de Newton-Raphson pour la résolution de cette équation à partir d'une estimation initiale a_0.
   Plus précisément, on vous demande de décrire comment on obtient une nouvelle estimation a_{k+1}, en termes des n données R_i et Θ_i et d'une estimation précédente a_k.
3. Calculer la valeur a_1 en partant de $a_0 = 5$.

116 Janvier 2016 : Le train arrivera-t-il ?

Avec la condition initiale $u(0) = 2$, considérons le problème de Cauchy :

$$u'(t) = \underbrace{u(t) + \exp(2t)}_{f(t,u)}$$

1. Donner la solution analytique du problème de Cauchy.
2. Démontrer rigoureusement que la formule du cheminot est une méthode d'ordre deux.

$$U_{i+1} = \frac{1}{3}\left(-U_{i-1} + 4U_i\right) + \frac{2h}{3}\ F_{i+1}$$

3. Calculer U_1 en effectuant une itération d'Euler explicite avec un pas $h = 1$.
4. Calculer ensuite U_2 en effectuant une itération du cheminot avec un pas $h = 1$.
5. Ecrire un programme qui effectue $n - 1$ itérations du cheminot de pas h après l'itération d'Euler.

   ```
   function [T,U] = cheminot(n,h)
   ```

   Les vecteurs `T` et `U` de taille $n+1$ contiendront $T_i = ih$ et $U_i \approx u(T_i)$ avec les indices $i = 0 \dots n$. Le pas des itérations est `h`.

117 Janvier 2016 : Polynômes d'Hermite

Considérons une fonction u définie dans l'intervalle $[0, 1]$, on connaît les valeurs U_0, U_1 et les dérivées premières U'_0, U'_1 de cette fonction aux deux extrémités de l'intervalle. On définit ensuite une approximation u^h de u comme l'unique polynôme de degré trois

$$u^h(x) = U_0\ \phi_0(x) + U_1\ \phi_1(x) + U'_0\ \phi_2(x) + U'_1\ \phi_3(x)$$

tel que les valeurs et les dérivées premières de u et de u^h coïncident aux extrémités $x = 0$ et $x = 1$. Les quatre fonctions de base $\phi_i(x)$ sont les polynômes d'Hermite de degré trois.

1. Calculer les expressions[15] des quatre fonctions de base $\phi_i(x)$
2. En intégrant ces polynômes, obtenir les quatre coefficients de la quadrature suivante :

$$\underbrace{\int_0^1 u(x)\ dx}_{I} \approx \underbrace{a\ U_0 + b\ U_1 + c\ U'_0 + d\ U'_1}_{I^h}$$

3. Quelle est le degré de précision de la quadrature obtenue ?

118 Janvier 2017 : Hillary perdue dans les différences finies

Pour estimer la dérivée seconde à l'origine d'une fonction $u(x)$, Hillary a trouvé la formule suivante:

$$u''(0) \approx \frac{2U_0 + \alpha U_h + \beta U_{2h} - U_{3h}}{h^2}$$

Malencontreusement, un slave facétieux lui a chapardé les valeurs des paramètres réels α et β qui donnent l'ordre de précision le plus élevé possible pour cette formule de différences finies.

1. Aider Hillary à retrouver les valeurs des deux paramètres.
2. Donner l'ordre de précision et calculer l'expression du terme d'erreur.

[15] A titre d'encouragement, une partie de la réponse vous est fournie : $\phi_3(x) = x^2(x-1)$

3. Calculer la valeur optimale de h afin de minimiser l'erreur totale, c'est-à-dire, les contributions conjointes de l'erreur de discrétisation et des erreurs d'arrondi dues au calcul en virgule flottante.
   Les valeurs U_0, U_h, U_{2h} et U_{3h} sont fournies avec une double précision ($\epsilon = 10^{-16}$).
   La dérivée quatrième de u est constante et vaut l'unité.

119 Janvier 2017 : Donald veut calculer pi !

Pour obtenir une approximation de π, Donald utilise la méthode de Newton-Raphson pour la fonction :

$$f(x) = \cos\left(\frac{x}{2}\right)$$

En partant avec $x_1 = 3$, son programme MATLAB fournit les résultats :

```
x =  3.00000000000000 : Error 1.4159265e-01 at iteration 1
x =  3.14182968860530 : Error 2.3703502e-04 at iteration 2
x =  3.14159265358868 : Error 1.1097789e-12 at iteration 3
x =  3.14159265358979 : Error 0.0000000e+00 at iteration 4
```

1. Ecrire la relation entre x_{n+1} et x_n qui définit la suite générée par la méthode de Newton-Raphson.
2. Définir rigoureusement l'ordre de convergence d'une méthode itérative.
3. Démontrer que la méthode de Newton-Raphson sera au moins d'ordre trois pour $f(x) = \cos\left(\frac{x}{2}\right)$.
4. Ecrire le programme MATLAB de Donald :

   ```
   function [x error] = cosinus(x0,tol,nmax)
   ```

   Les vecteurs `x` et `error` contiendront les itérations successives ainsi que l'erreur correspondante. Les arguments sont le nombre maximal d'itérations `nmax`, l'incrément maximal admis `tol` et le candidat initial `x0`. L'exécution de `[x error] = cosinus(3,1e-13,100)` fournit les résultats ci-dessus.
5. Estimer l'ordre de convergence observé par Donald lors de l'exécution de son programme. Est-ce en accord avec la théorie ?

120 Janvier 2017 : Heun revisité par Barack

Pour obtenir une solution approchée d'une équation différentielle $u'(x) = f(x, u(x))$,
Barack propose d'utiliser une nouvelle méthode de Runge-Kutta définie comme suit :

$$\begin{aligned} U_{i+1} &= U_i + h\left(\frac{3K_1 + K_2}{4}\right) \\ K_1 &= f(X_i, U_i) \\ K_2 &= f(X_i + \alpha, U_i + \beta K_1) \end{aligned}$$

1. Calculer les valeurs de α et β afin que cette méthode soit du même ordre que la méthode de Heun.
2. Donner l'expression analytique de la région de stabilité de la méthode de Barack.
   Esquisser cette région dans le plan complexe en définissant précisément les axes de la figure.
   Y indiquer aussi la zone de stabilité de la méthode d'Euler explicite.
   Ici, on considère évidemment le problème modèle habituel : $u'(x) = \lambda\, u(x)$.

3. Ecrire une fonction MATLAB[16]

```
function [X,U] =  barackIntegrate(n,h,U0,f)
```

qui retourne X_i et U_i obtenues en effectuant n pas `h` avec la méthode de Barack. Ce sont deux vecteurs de taille `n+1`. On commence le calcul avec `U0` en $x = 0$. Le dernier argument `f` est un pointeur vers la fonction `dudx = f(x,u)`.

121 **Janvier 2018 : Faut-il rapatrier Bézier ?**

	T_i	U_i
0	0	0
1	1/2	1
2	1/2	−1
3	1	2

Nous disposons d'un ensemble de quatre données d'une fonction inconnue $u(t)$ sur l'intervalle $[0, 1]$. Nous souhaitons calculer une approximation au sens des moindres carrés $u^h(t)$ à partir de trois fonctions de Bézier

$$u^h(t) = \sum_{i=0}^{2} a_i B_i^2(t)$$

1. Donner l'expression des trois fonctions de Bézier $B_i^2(t)$ définies[17] pour les six noeuds $[T_0, T_1, T_2, T_3, T_4, T_5] = [0, 0, 0, 1, 1, 1]$. Esquisser graphiquement l'allure des trois fonctions.
2. Ecrire la fonction $J(a_0, a_1, a_2)$ qu'il faut minimiser pour résoudre un tel problème.
3. Pour obtenir une approximation polynomiale d'ordre deux, est-il préférable d'utiliser les fonctions de Bézier ou les trois monômes 1, t et t^2 comme fonctions de base ? Justifier brièvement votre réponse !
4. Calculer les valeurs des coefficients a_i.

122 **Janvier 2018 : Théo a expulsé la division !**

Au parc Maximilien, se trouvaient quatre opérateurs permettant d'effectuer l'addition, la soustraction, la multiplication et la division. Hélas, Théo a expulsé la division et il doit remplacer l'opération a/b par la multiplication $a * (1/b)$ où $(1/b)$ sera obtenu en recherchant une racine de la fonction :

$$f(x) = b - \frac{1}{x}$$

Théo doit donc faire appel à un ingénieur soudanais pour implémenter la méthode de Newton-Raphson. Avec $x_0 = 1.5$ et $b = 0.5$, le programme fournit les résultats :

[16] Il est parfaitement possible d'écrire la quasi-totalité du programme et d'obtenir l'ensemble des points qui y sont associé même si vous n'avez pas obtenu les valeurs de α et β à la première sous-question !

[17] On définit les B-splines à partir de $n + 1$ noeuds $T_0 \leq T_1 \leq T_2 \leq \ldots \leq T_n$.
Une fonction $B_i^p(t)$ est nulle sauf dans l'intervalle $[\, T_i, T_{i+1+p} \,[$ où elle est définie par la relation de récurrence :

$$B_i^p(t) = \frac{(t - T_i)}{(T_{i+p} - T_i)} B_i^{p-1}(t) + \frac{(T_{i+1+p} - t)}{(T_{i+1+p} - T_{i+1})} B_{i+1}^{p-1}(t)$$

avec $i = 0, \ldots, n - p - 1$ et en partant de :

$$B_i^0(t) = \begin{cases} 1, & \text{si } \; t \in [T_i T_{i+1}[\\ 0, & \text{ailleurs} \end{cases}$$

On observe immédiatement que pour des noeuds de multiplicité supérieure à un, la formule de récurrence peut faire apparaître une valeur nulle à l'un ou l'autre dénominateur. On complète donc la définition en spécifiant qu'il ne faut tenir compte que des termes dont les dénominateurs ne s'annulent pas. Les fonctions de Bézier sont juste des B-splines avec un choix particulier de noeuds.

```
x = 1.50000000000000 : Estimated error 3.7500000e-01  at 1 iteration
x = 1.87500000000000 : Estimated error 1.1718750e-01  at 2 iteration
x = 1.99218750000000 : Estimated error 7.7819824e-03  at 3 iteration
x = 1.99996948242188 : Estimated error 3.0517112e-05  at 4 iteration
x = 1.99999999953434 : Estimated error 4.6566129e-10  at 5 iteration
x = 2.00000000000000 : Estimated error 0.0000000e+00  at 6 iteration
```

1. Ecrire[18] la relation entre x_{n+1} et x_n qui définit la suite générée par la méthode de Newton-Raphson.
2. Estimer l'ordre de convergence observé par Théo lors de l'exécution du programme.
3. Pour une valeur quelconque $b > 0$, calculer le plus grand encadrement possible pour le candidat x_0 afin que la convergence de la méthode de Newton-Raphson soit toujours assurée.
   Cet encadrement dépend évidemment de la valeur de b !
4. Re-écrire le programme MATLAB pour Théo[19] :

   ```
   function [x] = inverse(b,x0,tol,nmax)
   ```

   Le vecteur `x` contiendra les itérations successives. Les arguments sont le nombre maximal d'itérations `nmax`, l'incrément maximal admis `tol`, le paramètre b et le candidat initial `x0`.

123 Janvier 2018 : Quadratures de Rajoy et Puigdemont

Soit les abscisses $(X_0, X_1, X_2) = (-\alpha, 0, \alpha)$, où α est un nombre réel donné tel que $0 < \alpha < 1$.
Soit trois nombres réels w_0, w_1, w_2. On estime alors l'intégrale de $u(x)$ sur un intervalle avec les quadratures de Rajoy et Puigdemont avec 3 points de la manière suivante :

$$I_h \;=\; \sum_{i=0}^{2} w_i\, u(X_i) \;\approx\; \int_{-1}^{1} u(x)dx \;=\; I$$

Le choix de α sera toutefois différent en fonction de l'option catalane ou castillanne de la méthode.

1. Trouver une expression des trois nombres w_0, w_1, w_2 en fonction de α afin que les quadratures soient exactes pour tout polynôme quelconque de degré 2.
2. Donner l'expression de I_h en fonction de α lorsqu'on applique Rajoy-Puigdemont à 3 points avec un unique intervalle $[-1, 1]$ pour la fonction :

   $$g(x) = \frac{1}{x+2}$$

3. Obtenir α de Rajoy afin que l'erreur $E_h = I - I_h$ soit minimale pour cette fonction g.
4. Obtenir α de Puigdemont afin que la quadrature soit exacte pour tout polynôme de degré 4.
   Est-ce que les valeurs α de nos deux protagonistes peuvent ou doivent être identiques ?
5. Donner l'ordre de précision de la méthode **composite de Puigdemont**[20] à 3 points.
   Justifier votre réponse brièvement.
6. Considérons I_h et $I_{h/2}$ obtenus par Puigdemont composite avec 1 et 2 sous-intervalles.
   Donner I_* la combinaison linéaire de I_h et $I_{h/2}$ qui sera la meilleure extrapolation de Richardson.
   Quelle sera l'ordre théorique de précision de I_* ?

[18] Il faut évidemment tenir compte que vous ne disposez plus de la division que Théo a expulsée !
[19] Eh oui, il a aussi expulsé son ingénieur soudanais et n'a donc plus de programme pour pouvoir diviser...
[20] Il s'agit bien de l'exposant m du terme d'erreur écrit sous la forme $\mathcal{O}(h^m)$ lorsqu'on applique la méthode avec un nombre variable de sous-intervalles pour intégrer une intervalle $[a, b]$ donné.

124 Janvier 2019 : C'est fini, les différences pour Charles...

Pour estimer la dérivée seconde à l'origine d'une fonction, Charles souhaite utiliser la formule :

$$U_0'' \approx \frac{(2U_0 - 5U_{-h} + \alpha U_{-2h} + \beta U_{-3h})}{\gamma h^2}$$

Malencontreusement, il a bêtement oublié de noter les valeurs des paramètres réels α, β et γ.

1. Retrouver les valeurs des trois paramètres.
2. Donner l'ordre de précision de la formule obtenue et obtenir l'expression du terme d'erreur.
3. Calculer la valeur de h afin de minimiser l'erreur totale en tenant compte des erreurs d'arrondis.
   Les valeurs de la fonction u sont calculées en double précision ($\epsilon = 10^{-16}$).
   En tout point, la valeur absolue de toutes les dérivées de u est supposée inférieure à 10^4.

125 Janvier 2019 : Newton et Raphson ont-ils des gilets jaunes ?

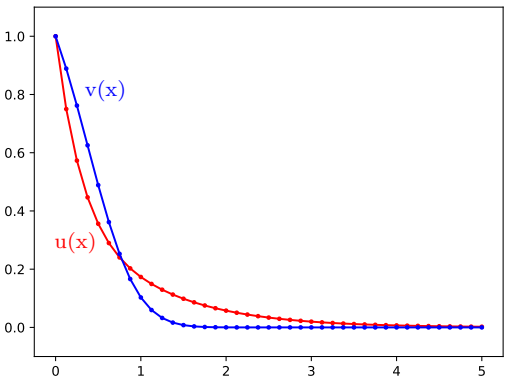
Pour résoudre le système d'équations $\begin{cases} x^2 + y^2 &= 1 \\ y(x+z) &= 0 \\ z^2 + y^2 &= 1 \end{cases}$

Emmanuel souhaite utiliser la méthode de Newton-Raphson.

1. Quel est le nombre de solutions de ce système d'équations ?
2. Ecrire le système linéaire à résoudre lors de chaque itération de Newton-Raphson pour obtenir le vecteur $\Delta\mathbf{x} = [\Delta x, \Delta y, \Delta z]$.
3. Calculer la première itération $\mathbf{x}_1$ en partant de $\mathbf{x}_0 = [4, 0, 4]$.
4. Démontrer que la suite des itérées obtenues à partir de ce $\mathbf{x}_0$ s'écriront sous la forme $\mathbf{x}_i = [\alpha_i, 0, \alpha_i]$.
   Donner la fonction de récurence g telle que $\alpha_{i+1} = g(\alpha_i)$.
5. Pour quelles valeurs de α_0 est-ce que la suite converge vers un point fixe ?
6. Quel est le point fixe vers lequel la méthode converge avec $\alpha_0 = 4$?

126 Janvier 2019 : Une économie de décroissance... pour Nancy et Donald

Pour résoudre le problème de la décroissance $\begin{cases} u'(x) &= -u(x)\ (1 + v(x)) \\ v'(x) &= -\dfrac{v(x)}{u(x)} \\ u(0) &= v(0) = 1 \end{cases}$



dont les solutions $u(x)$ et $v(x)$ convergent vers zéro, Nancy a implémenté une méthode semi-implicite

$$\begin{cases} \dfrac{U_{n+1} - U_n}{h} &= -U_n\ (1 + V_{n+1}) \\ \dfrac{V_{n+1} - V_n}{h} &= -\dfrac{V_{n+1}}{U_n} \end{cases}$$

Donald trouve la méthode de Nancy trop compliquée et préfèrerait utiliser un schéma d'Euler explicite.

1. Ecrire le schéma explicite d'Euler pour les équations de la décroissance.
2. Ecrire une fonction `python` qui calcule U_i et V_i avec la méthode de Nancy :

   ```
   U =  integrateNancy(n,h)
   ```

   La fonction calculera `n` itérations avec un pas de temps `h`.
   Les résultats seront fournis dans un tableau `U` de dimensions $(n+1) \times 2$.
3. Comment choisir le pas de temps h pour que la méthode de Nancy soit stable ?
4. Comment choisir le pas de temps h pour que la méthode de Donald soit stable sur l'intervalle de temps $[0, 5]$, sachant que $u(5) \approx 3.5 \times 10^{-3}$?

127 Juin 2019 : Bart exige d'avoir des B-splines !

Soit les points $[T_0, T_1, T_2, T_3, T_4, T_5] = [-2, -1, 0, 1, 2, 3]$.
Un étudiant facétieux souhaite dessiner une nouvelle courbe :

$$u(t) = \frac{\sum_{i=0}^{2} B_i^2(t) W_i U_i}{\sum_{i=0}^{2} B_i^2(t) W_i} = \frac{3\alpha(1 + 2t - 2t^2)}{(\alpha + 1) + 2(\alpha - 1)(t - t^2)},$$

	U_i	W_i
0	0	1
1	3	α
2	0	1

sur l'intervalle $t \in [T_2, T_3] = [0, 1]$.
Les fonctions $B_i^2(t)$ sont les fonctions B-splines de degré deux.
Le paramètre α est un réel positif à déterminer.

1. Esquisser les fonctions $B_0^2(t)$, $B_1^2(t)$ et $B_2^2(t)$ sur l'intervalle $t \in [T_0, T_5] = [-2, 3]$
2. Donner l'expression analytique de ces trois fonctions sur l'intervalle $t \in [T_2, T_3] = [0, 1]$.
3. Calculer la valeur de α afin que la valeur maximale de la courbe soit égale à $\frac{5}{2}$.

128 Juin 2019 : Tom et Théo découvrent Newton-Raphson

A partir de n données (X_i, U_i), Tom et Théo souhaitent approximer $u(x)$ le transfert du Nord vers le Sud par une expression :

$$u^h(x) = \frac{1}{x + a}$$

en ajustant le paramètre réel a afin de minimiser la somme des carrés des n écarts $U_i - u^h(X_i)$.

	X_i	U_i
0	0	0.625
1	1	2.000

1. Ecrire l'équation que doit satisfaire a en termes des n données X_i et U_i.
2. Utiliser la méthode de Newton-Raphson pour la résolution de cette équation à partir d'une estimation initiale a_0. Plus précisément, il faut fournir l'équation qui permet d'obtenir une nouvelle estimation a_{k+1}, à partir des n données X_i et U_i et d'une estimation précédente a_k.
3. Obtenir[21] a en partant de $a_0 = 1$ pour les deux données du tableau $(n = 2)$.

[21] Réaliser un dessin schématique du problème (en y incluant les données) peut être une bonne idée :-)

129 Juin 2019 : Maggie et les différences finies compactes

Maggie a découvert le concept de différences finies compactes :

$$\alpha U'_{i-1} + U'_i + \alpha U'_{i+1} \quad = \quad \beta \frac{U_{i+1} - U_{i-1}}{2h} + \gamma \frac{U_{i+2} - U_{i-2}}{4h} + \mathcal{O}(h^p)$$

Cela lui permet d'obtenir une estimation très précise des valeurs $U'_i \approx u'(X_i)$ de la dérivée d'une fonction périodique à partir de ordonnées $U_i = u(X_i)$ aux abscisses $X_i = ih$ avec $i = 0 \dots n$.

La périodicité implique évidemment $U_0 = U_n$ et $U'_0 = U'_n$.

Malencontreusement, Théo lui a chapardé les valeurs des 3 coefficients α, β et γ.

1. Quel est l'ordre de précision le plus élevé[22] avec les α, β et γ les plus adéquats ?
   Justifier brièvement.
2. Quelles relations doivent satisfaire α, β et γ pour obtenir la méthode la plus précise ?
3. Calculer les valeurs optimales de β, γ sachant que $\alpha = \frac{1}{3}$.
4. Ecrire une fonction `python`

   ```
   dU = compactDerivative(U,alpha,beta,gamma,h)
   ```

   qui calcule le vecteur U'_i à partir des données U_i.
   Le résultat sera fourni dans un tableau `dU` dont la taille sera celle de `U`.
   Les coefficients α, β et γ ainsi que h sont donnés en argument d'entrée[23].

[22] Il s'agit de la valeur de l'exposant p dans le terme d'erreur $\mathcal{O}(h^p)$
Immédiatement, on peut observer que choisir $\alpha = \gamma = 0$ et $\beta = 1$ donne une différence finie centrée classique d'ordre deux.
Mais, cette réponse ne rendra heureux ni Maggie, ni Théo, ni le correcteur...
C'est l'ordre le plus élevé possible qu'on souhaite avoir !

[23] Il est donc possible d'avoir un programme correct, même si on n'a pas obtenu les bonnes valeurs de α, β et γ.
Les données fournies sont périodiques : il n'est pas nécessaire de le tester.
Le code doit être simple et efficace.
Il n'est pas nécessaire d'écrire de commentaires :-)