EPL1104 : solution de l'examen de juin 2021

1 La règle campinoise de Jurgen

Pour intégrer une fonction $u(x)$, Jurgen définit une règle de quadrature campinoise :

$$\int_{-1}^{1} u(x)\ dx = I \approx I^h = \sum_{i=1}^{4} w_i\ u(X_i)$$

où les quatres abscisses limbourgeoises sont $X_1 = -1$, $X_2 = -\alpha$, $X_3 = \alpha$ et $X_4 = 1$ avec $\alpha \in [0, 1]$.

1. Obtenir les expressions de quatre poids w_i en fonction de α
   afin que la quadrature de Jurgen intégre exactement tout polynôme de degré trois.

   Il faut aussi observer que les abscisses de Jurgen sont symétriques et donc les poids le seront aussi.
   Il y a donc deux uniques expressions à trouver dans cette question !
   On peut obtenir les poids en intégrant les polynômes de Lagrange correspondants.

$$\begin{aligned} w_1 \;=\; w_4 \;&=\; \int_{-1}^{1} \frac{(x+1)(x+\alpha)(x-\alpha)}{(1+1)(1+\alpha)(1-\alpha)}\ dx \;=\; \int_{-1}^{1} \frac{(x+1)(x^2-\alpha^2)}{2(1-\alpha^2)}\ dx \\ &= \frac{1}{2(1-\alpha^2)} \underbrace{\int_{-1}^{1} x^3 + x^2 - \alpha^2 x - \alpha^2\ dx}_{\frac{2}{3} - 2\alpha^2} \\ &\downarrow \\ &= \frac{1-3\alpha^2}{3(1-\alpha^2)} \end{aligned}$$

$$\begin{aligned} w_2 \;=\; w_3 \;&=\; \int_{-1}^{1} \frac{(x+1)(x+\alpha)(x-1)}{(\alpha+1)(\alpha+\alpha)(\alpha-1)}\ dx \;=\; \int_{-1}^{1} \frac{(x^2-1)(x+\alpha)}{-2\alpha(1-\alpha^2)}\ dx \\ &= \frac{1}{-2\alpha(1-\alpha^2)} \underbrace{\int_{-1}^{1} x^3 - x + \alpha x^2 - \alpha\ dx}_{\alpha\frac{2}{3} - 2\alpha} \\ &\downarrow \\ &= \frac{2}{3(1-\alpha^2)} \end{aligned}$$

On conclut que :

$$\boxed{\begin{aligned} w_1 \;=\; w_4 \;&=\; \frac{1-3\alpha^2}{3(1-\alpha^2)} \\ w_2 \;=\; w_3 \;&=\; \frac{2}{3(1-\alpha^2)} \end{aligned}}$$

Une autre option nettement plus efficace et astucieuse consiste à imposer que n'importe quel polyôme de degré trois soit parfaitement intégré par la quadrature campinoise. Et pour une quadrature symétrique, il suffit uniquement de vérifier que l'on intégre parfaitement $u(x) = 1$ et $u(x) = x^2$. On écrit donc :

$$\begin{aligned} 2w_1 + 2w_2 &= \int_{-1}^{1} dx = 2 \\ 2w_1 + 2w_2\alpha^2 &= \int_{-1}^{1} x^2\, dx = \frac{2}{3} \end{aligned}$$

$\downarrow$ *En soustrayant la seconde équation de la première :-)*

$$\begin{aligned} 2(1-\alpha^2)w_2 &= \frac{4}{3} \\ w_2 &= \frac{2}{3(1-\alpha^2)} \end{aligned}$$

On obtient évidemment les mêmes poids qu'en intégrant les polynômes de Lagrange.
Les deux approches sont évidemment admises par le correcteur.
Finalement, on pouvait aussi noter immédiatement que la somme des deux poids vaut l'unité : il n'y avait qu'une seule expression à trouver en réalité :-)
Une question simple qui a posé pourtant pas mal de difficultés pour beaucoup d'étudiants !

2. Calculer la valeur optimale de α afin que le degré de précision soit le plus élevé possible.

On calcule α afin d'intégrer parfaitement $u(x) = x^4$:

$$2w_1 + 2w_2\alpha^4 = \int_{-1}^{1} x^4\, dx = \frac{2}{5}$$

$\downarrow$ *En y substitutant les expressions des deux poids :-)*

$$\begin{aligned} \frac{2-6\alpha^2}{3(1-\alpha^2)} + \frac{4\alpha^4}{3(1-\alpha^2)} &= \frac{2}{5} \\ 5 - 15\alpha^2 + 10\alpha^4 &= 3 - 3\alpha^2 \\ 1 - 6\alpha^2 + 5\alpha^4 &= 0 \\ \alpha^2 &= \frac{6 \pm \sqrt{36-20}}{10} = \frac{6 \pm 4}{10} \end{aligned}$$

En rejetant $\alpha^2 = 1$ qui n'est pas acceptable, on obtient une unique valeur possible pour $\alpha^2 = \frac{1}{5}$.

On conclut que : $\boxed{\alpha = \frac{1}{\sqrt{5}}}$

3. Quelle est ce degré maximal de précision ?

   Comme la règle est symétrique, on intégrera parfaitement x^5,

   le degré de précision est donc : $\boxed{d = 5}$

2 Marc veut vacciner des B-splines

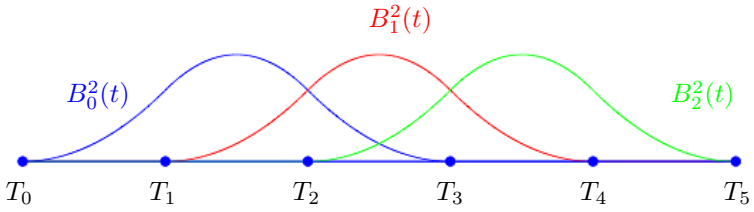
	X_i	U_i
0	0	0
1	1/2	1
2	1/2	2
3	1	4

A partir de quatre mesures indépendantes (X_i, U_i) d'un vaccin inconnu $u(x)$ sur l'intervalle $x \in [0,1]$, Marc recherche une approximation $u^h(x)$ comme la combinaison linéaire :

$$u(x) \approx \underbrace{\sum_{i=0}^{2} a_i \, B_i^2(x)}_{u^h(x)}$$

où $B_i^2(t)$ sont les fonctions B-splines pour les noeuds $\mathbf{T} = [T_0, T_1, T_2, T_3, T_4, T_5] = [-2, -1, 0, 1, 2, 3]$.
Il s'agit de trouver a_0, a_1 et a_2 afin que $u^h(x)$ approxime $u(x)$ au sens des moindres carrés.

1. Esquisser les fonctions $B_0^2(t)$, $B_1^2(t)$ et $B_2^2(t)$ sur l'intervalle $t \in [T_0, T_5] = [-2, 3]$.



   Il ne faut pas un dessin précis, mais il est requis que les 3 fonctions soient plus ou moins semblables et de classe C_1 *(pas de point anguleux !). En outre, la fonction* $B_0^2(t)$ *est définie sur l'intervalle* $[T_0, T_3]$ *(ce qui n'est pas l'intervalle* $[0,3]$ *: idem pour les deux autres fonctions !)*

   Pas mal d'étudiants échouent dans cette question vraiment très facile.
   Et pourtant, c'est vraiment pas une question nouvelle et c'est vraiment facile !

2. Donner l'expression analytique de ces trois fonctions sur l'intervalle $t \in [T_2, T_3] = [0, 1]$.
   Marc vous informe gratuitement que $B_2^2(t) = t^2/2$, afin de vous encourager :-)

   Comme sur l'intervalle $[0,1]$, *seules* $B_1^1(t)$ *et* $B_2^1(t)$ *sont non-nulles, on doit juste évaluer :*

$$\begin{array}{lcrcrcl}
2B_0^2(t) & = & (1-t)B_1^1(t) & & & = & (1-t)(1-t) & = & 1-2t+t^2 \\
2B_1^2(t) & = & (t+1)B_1^1(t) & + & (2-t)B_2^1(t) & = & (t+1)(1-t)+(2-t)t & = & 1+2t-2t^2 \\
2B_2^2(t) & = & & & tB_2^1(t) & = & t^2 & = & t^2
\end{array}$$

   On conclut que : $\boxed{B_0^2(t) = \dfrac{1-2t+t^2}{2} \qquad B_1^2(t) = \dfrac{1+2t-2t^2}{2} \qquad B_2^2(t) = \dfrac{t^2}{2}}$

   On peut vérifier le résultat en observant que la somme des trois fonctions vaut un.
   On peut aussi noter qu'une bonne partie de la réponse se trouvait dans l'énoncé, puisque :

$$u(t) = \frac{3\alpha B_1^2(t)}{B_0^2(t) + \alpha B_1^2(t) + B_2^2(t)} = \frac{3\alpha(1+2t-2t^2)}{(\alpha+1)+2(\alpha-1)(t-t^2)},$$

Il est aussi possible d'écrire $B_0^2() = B_2^2(1-t)$ *et d'en déduire* $B_1^2(t) = 1 - B_0^2(t) - B_2^2(t)$.
Si, si , c'était vraiment élémentaire !

3. Ecrire la fonction $J(a_0, a_1, a_2)$ qu'il faut minimiser pour obtenir $u^h(x)$.
   En déduire l'expression du système linéaire qu'il faudrait résoudre.

Tout d'abord, il suffit d'écrire :

$$\boxed{J(a_0,a_1,a_2) \quad = \quad \sum_{i=0}^{3} \left(U_i - \sum_{j=0}^{2} a_j \; B_j^2(X_i) \right)^2}$$

Ensuite, il suffit d'annuler les dérivées partielles de J par rapport aux trois paramètre a_0, a_1 et a_2 afin d'obtenir les équations normales :

$$\begin{bmatrix} \sum_{i=0}^{3} B_0^2(X_i)B_0^2(X_i) & \sum_{i=0}^{3} B_0^2(X_i)B_1^2(X_i) & \sum_{i=0}^{3} B_0^2(X_i)B_2^2(X_i) \\ \sum_{i=0}^{3} B_1^2(X_i)B_0^2(X_i) & \sum_{i=0}^{3} B_1^2(X_i)B_1^2(X_i) & \sum_{i=0}^{3} B_1^2(X_i)B_2^2(X_i) \\ \sum_{i=0}^{3} B_2^2(X_i)B_0^2(X_i) & \sum_{i=0}^{3} B_2^2(X_i)B_1^2(X_i) & \sum_{i=0}^{3} B_2^2(X_i)B_2^2(X_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{3} B_0^2(X_i)U_i \\ \sum_{i=0}^{3} B_1^2(X_i)U_i \\ \sum_{i=0}^{3} B_2^2(X_i)U_i \end{bmatrix}$$

Il n'était pas demandé de calculer les valeurs numériques de ce système : c'était d'ailleurs inutile !

4. Observer qu'on peut obtenir exactement la même courbe en calculant l'interpolation par trois données choisies astucieusement. Quelles seraient ces trois données ?

Géométriquement, on peut immédiatement observer que l'approximation passera exactement par les deux points $(0, U_0)$, $(1, U_3)$, tout en passant à une distance égale de $(1/2, U_1)$ et $(1/2, U_2)$.

Il faut donc interpoler les données suivantes :

	X_i	U_i
0	0	0
1	1/2	3/2
2	1	4

Avec un tout peu de bon sens, on pouvait trouver ceci en n'ayant fait strictement aucun calcul !

5. Finalement, obtenir l'expression des trois paramètres (a_0, a_1, a_2).

   On écrit le système de l'interpolation qui donne :

$$\begin{bmatrix} B_0^2(0) & B_1^2(0) & B_2^2(0) \\ B_0^2(\frac{1}{2}) & B_1^2(\frac{1}{2}) & B_2^2(\frac{1}{2}) \\ B_0^2(1) & B_1^2(1) & B_2^2(1) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{3}{2} \\ 4 \end{bmatrix}.$$

Comme $B_0^2(0) = 1/2$, ...

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{8} & \frac{6}{8} & \frac{1}{8} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{3}{2} \\ 4 \end{bmatrix}.$$

On conclut en résolvant ce système :

$$\boxed{\begin{array}{lcl} a_0 & = & -1 \\ a_1 & = & 1 \\ a_2 & = & 7 \end{array}}$$

On peut finalement observer que cette solution est également la solution du système de l'approximation : ce qui confirme le choix judicieux des trois données à interpoler.

3 Marc et Jurgen travaillent de manière conjointe

Avec la condition initiale $u(0) = 1$, ils considèrent le problème de Cauchy :

$$u'(x) = \underbrace{-\Big[u(x)\Big]^3 + \cos(x)}_{f(x,u)}$$

Ils souhaitent obtenir les valeurs $U_i \approx u(X_i)$ pour $X_i = ih$ avec $h > 0$ et $i = 0 \dots n$.
Marc est un fan de Newton et de Raphson tandis que Jurgen rêve implicitement d'Euler.

1. Ecrire le schéma d'Euler implicite pour ce problème afin d'obtenir U_{i+1} à partir de U_i.

   Oui : cette question était d'une simplicité consternante :-)

Une itération d'Euler implicite s'écrit simplement :

$$\boxed{U_{i+1} = U_i - hU_{i+1}^3 + h\cos(X_{i+1})}$$

2. Décrire l'application de Newton-Raphson pour résoudre l'équation non-linéaire ainsi obtenue.

   Oui : on vient bien d'écrire une équation non-linéaire

$$\underbrace{t - U_i + ht^3 - h\cos(X_{i+1}}_{g(t)} = 0$$

   dont l'inconnue $t = U_{i+1}$.
   L'unique difficulté est de bien identifier l'inconnue :-)
   Considérer x comme l'inconnue est évidemment l'erreur fatale !

Le schéma de Newton-Raphson s'écrit :

$$\boxed{\begin{array}{rcl} t_0 & = & U_i \\ t_{n+1} & = & t_n - \underbrace{\dfrac{t_n - U_i + ht_n^3 - h\cos(X_{i+1})}{1 + 3ht_n^2}}_{\dfrac{g(t)}{g'(t)}} \end{array}}$$

   Il faut bien avoir l'expression correcte de la dérivée $g'(t)$ pour valider cette question !
   Toute tentative de dériver le cosinus est considérée comme une erreur fatale :-)

3. Ecrire une fonction `python` : `solveEulerImplicit(n,h,tol)`
   qui effectue `n` itérations temporelles en utilisant, de manière conjointe, les méthodes d'Euler implicite et de Newton-Raphson pour ce problème de Cauchy avec un pas `h`.
   On arrêtera[1] les itérations de Newton-Raphson lorsque l'incrément est inférieur à `tol`.
   La fonction renverra un tableau `U` contenant les `n+1` valeurs nodales en y incluant $U_0 = 1$.

   Une implémentation possible est :

```
from numpy import *

def solveEulerImpl(m,h,tol) :
  X = h*arange(m+1)
  U = zeros(m+1); U[0] = 1
  for i in range(m) :
    U[i+1] = U[i]
    dx = tol + 1
    while (abs(dx) >= tol) :
      g = U[i+1] - U[i] + h * U[i+1]**3 -  h * cos(X[i+1])
      dgdx = 1 + 3 * h * U[i+1]**2
      dx = -g/dgdx
      U[i+1] += dx
  return U
```

   Ce code n'est vraiment pas bien compliqué à écrire et correspond à une version fortement simplifiée du programme `eulerNewtonRaphson.py` *disponible sur le site web du cours :-)*

   C'était donc vraiment impardonnable de ne pas arriver à l'écrire sachant que vous pouviez disposer un formulaire où il est permis d'inclure des codes `python`. *Quelques étudiants ont utilisé une calculatrice pour cet examen, alors que c'était spécifiquement interdit : non seulement, c'est inutile, stupide et peu respecteux, mais toute expression numérique obtenue avec une calculatrice a été automatiquement considérée comme erronée : not a good idea !*

[1] Pour la simplicité, on supposera que la convergence de la méthode de Newton-Raphson est toujours garantie. Il est donc inutile de prévoir le cas où la méthode itérative ne convergerait pas...