TP4 : Systèmes linéaires et équations différentielles

Ce TP sera noté : à la fin de TP, envoyez un mail à `tibor.stanko@inria.fr`. N'oubliez pas d'inclure vos noms et 1-2 images commentées pour chacune des exercices 1 et 2.

Exercice 1 (Splines cubiques).

Pour rappel, une spline cubique $x(t)$ est une fonction polynomiale par morceaux qui interpole un ensemble de $p+1$ points (t_i, x_i) tel que $a = t_0 < t_1 < \cdots < t_p = b$. Pour simplicité, on suppose que $a = 0, b = 1$, et $t_{i+1} - t_i = h = \frac{1}{p}$.

Une spline cubique peut être défini comme le résultat de minimisation suivante avec des contraintes d'égalité :

$$\min_x \int_0^1 \|x''(t)\|^2 \, dt, \qquad x(t_i) = x_i \quad \forall i = 0, \ldots, p. \tag{1}$$

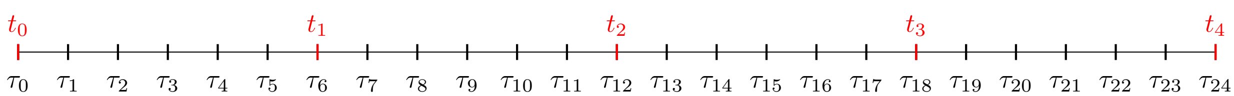
Résoudre le problème (1) est équivalent à résoudre *l'équation biharmonique* (équation différentielle de l'ordre 4) avec des contraintes positionnelles :

$$x^{(4)}(t) = 0, \qquad x(t_i) = x_i \quad \forall i = 0, \ldots, p. \tag{2}$$

On va maintenant discretizer le problème (2) afin de le résoudre. Pour discretizer le paramétre t (variable temporelle), on découpe chaque interval $[t_i, t_{i+1}]$ à n sous-intervals pour avoir une abscisse uniforme

$$\tau_k = k/N, \qquad k = 0, \ldots, N = n \times p$$

avec $n-1$ points intérieurs par morceau de spline. Exemple pour $p = 4, n = 6$:



On note $x_k = x(\tau_k)$. Pour discretizer la dérivée quatrième de x, on considère la différence finie de l'ordre 4 suivante :

$$x^{(4)}(\tau_k) \quad \approx \quad \frac{x_{k-2} - 4x_{k-1} + 6x_k - 4x_{k+1} + x_{k+2}}{h^4}.$$

Utilisez les questions suivantes pour compléter le code dans `TP4_spline.sce` afin de calculer une spline cubique qui interpole les données fournies dans le fichier.

1. Definir les variables `indices` et `coeffs` qui permettent de calculer les différences finies de l'ordre 4.
   (a) `indices` contient les indices des voisins de x_k relative à l'indice k, soit le vecteur $[-2, -1, 0, 1, 2]$.
   (b) `coeffs` contient les coefficients utilisés pour calculer $x^{(4)}$.
2. Résoudre le systeme `A*X=B` en utilisant la factorisation LU de la matrice `A`. Pour calculer cette factorisation dans Scilab, utiliser la fonction `[L,U]=lu(A)`. Tracer la courbe (décommenter la ligne `xpoly(X…)`).
3. Experimenter avec les 4 ensembles fournis et varier le nombre de points par morceau (variable `n`). Visualiser toutes les points qui constituent la courbe (décommenter la ligne `plot2d(X…)`). Quel est le problème avec la discretization utilisé ?

Exercice 2 (Équation de la chaleur).

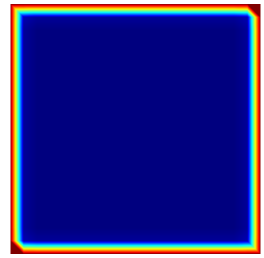
Soit une fonction $u(x, y, t)$ solution de l'équation différentielle suivante (équation de la chaleur)

$$\frac{\partial u}{\partial t} - c\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \;=\; \frac{\partial u}{\partial t} - c\nabla^2 u \;=\; 0$$

— $t \geq 0$, variable temporelle ;
— x, y, variables spatiales comprises entre 0 et N ;
— $c > 0$ coefficient dit de diffusion thermique.
Pour simplification, **on suppose que** $c = 1$.

On a les conditions de bord suivantes :

$$\begin{aligned} u(0, y, t) = u(1, y, t) &= 100, & t \geq 0,\ y \in [0, 1] \\ u(x, 0, t) = u(x, 1, t) &= 100, & t \geq 0,\ x \in [0, 1] \\ u(x, y, 0) &= 0, & x, y \in]0, 1[\end{aligned}$$



Problème initial ($t = 0$) :
rouge = 100°C, bleu = 0°C

On veut résoudre de manière numérique cette équation. Pour cela, on subdivise le temps et l'espace :
— pour l'espace, on a une grille uniforme : $x_i = i,\ y_j = j, \quad i, j = 0, \ldots, N$
— pour le temps, $t_k = k\tau$, avec $\tau > 0$ un pas de temps **choisi**.
— on note $u^k_{i,j} = u(x_i, y_j, y_k)$.

i. Même sans la résolution de l'équation différentielle, on connait déjà un certain nombre de $u^k_{i,j}$. Lesquels ?

On suppose maintenant qu'on connaisse à un instant t_k donné **tous** les $u^k_{i,j}$, $i, j = 0, \ldots, N$. Par contre, on ne connait pas tous les $u^{k+1}_{i,j}$ à l'instant suivant t_{k+1}. Le but des prochaines questions est de pouvoir exprimer les $u^{k+1}_{i,j}$ en fonction des $u^k_{i,j}$ en discrétisant l'équation

$$\frac{\partial u}{\partial t}(x_i, y_j, t_k) - \frac{\partial^2 u}{\partial x^2}(x_i, y_j, t_k) - \frac{\partial^2 u}{\partial y^2}(x_i, y_j, t_k) = 0.$$

Cette méthode est dite d'**Euler explicite**.

---

ii. Discrétiser le terme suivant $\partial u / \partial t\ (x_i, y_j, t_k)$ en fonction uniquement de $u^k_{i,j}$, $u^{k+1}_{i,j}$ et τ.
Quel est l'ordre de l'approximation en temps ?

iii. Discrétiser le terme suivant $\partial^2 u / \partial x^2\ (x_i, y_j, t_k)$ en fonction uniquement de $u^k_{i-1,j},\ u^k_{i,j},\ u^k_{i+1,j}$.
Discrétiser le terme suivant $\partial^2 u / \partial y^2\ (x_i, y_j, t_k)$ en fonction uniquement de $u^k_{i,j-1},\ u^k_{i,j},\ u^k_{i,j+1}$.
Quel est l'ordre de l'approximation en espace ?

iv. Établir le système à résourdre pour trouver les $u^{k+1}_{i,j}$ en fonction des $u^k_{i,j}$.

Complétez le code dans `TP4_heat.sce` et visualisez l'équation de la chaleur sur une domaine carré $[0, N]^2$.

1. Les températures $u^k_{i,j}$ sont stockés dans une matrice `T`. Construire la matrice initiale `T0(i,j)`$= u^0_{i,j}$.
2. Dans une boucle : calculer la prochaine iteration `T1` avec la méthode d'Euler explicite en utilisant les valeurs dans `T0`. Passer à la prochaine iteration (`T0 = T1`).
3. Modifier le pas de temps τ (variable `dt`). Que se passe-t-il ?