Programmation Scientifique 2 - TP2 - Méthodes des éléments finis en une dimension

Yann LE GUILLY

April 5, 2016

1 Introduction

Nous allons développer, valider et analyser un code Fortran mettant en œuvre la méthode des éléments finis. Cette méthode permet notamment de résoudre, de façon approchée, des équations différentielles ordinaires.

2 Formulation du problème

2.1 Énoncé du problème

On s'intéresse donc à la résolution approchée d'une équation différentielle ordinaire (EDO) avec les conditions aux limites suivantes:

$$\begin{cases} -u'' + u = f \quad \text{dans} \quad]0,1[\\ u'(0) = \alpha \\ u(1) = 0 \end{cases} \tag{1}$$

Où α est une constante dans $\mathbb{R}$, f est une fonction donnée définit sur $]0,1[$ et u l'inconnue. On considère la fonction f suffisamment régulière pour permettre de lui appliquer les opérations mathématiques que nous verrons plus loin.

2.2 Formulation variationnelle

Pour appliquer la méthode des éléments finis, la première étape consiste à établir la formulation variationnelle du problème. Pour l'obtenir, nous allons, dans un premier temps, multiplier l'équation (1.a) par une fonction test v.

$$\int_\omega -u''v + \int_\omega uv = \int_\omega fv$$

Ensuite nous intégrons sur le domaine $]0,1[$ en utilisant l'intégration par parties (2ème formule de Green en 1D). Nous obtenons ainsi la nouvelle formulation du problème:

$$\begin{cases} \text{Trouver} \quad u \in V \quad \text{telle que} \quad \forall v \in V : \\ \\ \int_{]0,1[} u'v' + \int_{]0,1[} uv = \int_{]0,1[} fv - \alpha v(0) \end{cases} \tag{2}$$

Cette forme est appelé formulation variationnelle du problème (1). Les conditions aux limites de type Neumann ou naturelle (1.b) sont incluent dans la partie

droite de l'équation (2). Concernant les conditions de Dirichlet homogènes (1.c), elles doivent être incluent dans l'espace fonctionnelle V. De plus, toutes les fonctions $v \in V$ satisfont $v(1) = 0$. V étant un espace d'Hilbert, on se place dans le cadre de Lax-Milgram (espace complet) qui nous permet d'assurer l'existence et l'unicité d'une solution pour $v \in V$.

3 Méthode des éléments finis

Maintenant que nous avons posé le cadre mathématique, nous allons nous attacher à la méthode des éléments finis en elle-même. Ainsi, les sections qui vont suivre sont associés à un code Fortran.

3.1 Discrétisation du domaine - sous-routines mesh

Afin d'appliquer la méthode des éléments finis, nous allons, dans un premier temps, discrétiser le domaine géométrique. Pour cela, nous considérons $n+1$ nœuds notés x_i avec $i \in 1, ..., n+1$ répartis uniformément sur Ω de sorte que $x_{i+1} - x_i = \frac{1}{n} = h$ pour $i = 1, ..., n$. Les segments $[x_i, x_{i+1}]$ sont appelés *éléments*.
La sous-routine **mesh** contenue dans la librairie **discretisation.f90** est chargée réaliser cette étape.

3.2 Discrétisation de la formulation variationnelle

u est approchée par $u_h \in Vect\{\varphi_i, i = 1, ..., n\}$. Ainsi, on réécrit la formulation variationnelle pour tout les éléments de la base $\{\varphi_i, i = 2, ..., n)\}$. Et par changement d'inconnue, on peut se ramener à $u_{\Gamma_D} = 0$. Ainsi, on écrit:

$$u_h = \sum_{j=1}^{n} u_j \varphi_j$$

Dans notre cas, les n fonctions φ_i sont définies telles que:

- φ_i est une fonction définie par morceaux, polynomiale de degré 1 sur chaque éléments $[x_i, x_{j+1}]$, $j = 1, ..., n$.
- $\varphi_i(x_j) = \delta_{ij}$, pour $j = 1, ..., n$.

Pour $\varphi_i(x)$ nous obtenons ainsi:

$$\begin{cases} \frac{x}{h} - i + 2 \quad \text{si} \qquad x \in [x_{i-1}, x_i] \\ \\ \frac{-x}{h} + i \quad \text{si} \qquad x \in [x_i, x_{i+1}] \\ 0 \quad \text{sinon} \end{cases}$$

Et pour les dérivées de $\varphi_i(x)$:

$$\begin{cases} \frac{1}{h} - i + 2 \quad \text{si} \qquad x \in [x_{i-1}, x_i] \\ \\ \frac{-1}{h} \quad \text{si} \qquad x \in [x_i, x_{i+1}] \\ 0 \quad \text{sinon} \end{cases}$$

Les fonctions **fbase** et **dfbase** contenues dans la librairie **functions.f90** sont chargées réaliser cette partie.

3.3 Système linéaire associé

Les considérations précédemment explicitées nous conduisent à la résolution du système linéaire:

$$Au = b$$

où $u \in \mathbb{R}$ est le vecteur des coefficients u_i approchant la solution du problème (1) et les matrices $A \in \mathbb{R}$ et $b \in \mathbb{R}$ sont définies par:

$$\begin{array}{l} A_{ij} = \int_0^1 \varphi_j'(x)\varphi_i'(x)dx + \int_0^1 \varphi_j(x)\varphi_i(x)dx \\ b_i = \int_0^1 f(x)\varphi_i(x)dx - \alpha\varphi_i(0) \end{array} \qquad \forall i, j = 1, ..., n.$$

3.4 Calculs sur les éléments finis

Afin de construire les matrices A et b, on se ramène au calcul sur chaque éléments construits précédemment. C'est à dire:

$$A_{ij} = \sum_{K \in \Gamma_h} \int_K \varphi_j'(x)\varphi_i'(x)dx + \int_K \varphi_j(x)\varphi_i(x)dx$$

$$b_i = \sum_{K \in \Gamma_h} \int_K f(x)\varphi_i(x)dx - \alpha\varphi_i(0) = \sum_{K \in \Gamma_h} b_p - \alpha\varphi_i(0)$$

Du fait que seules les fonctions φ_k et φ_{K+1} soient non nulles, la fonction **local2global** de la librairie **functions.f90** est chargée de passer de la numérotation locale à globale.
Afin d'effectuer les calculs des matrices élémentaires, nous avons développé la sous-routine **calel**. Cette sous-routine utilise d'autres sous-routines que sont **integ**, **integd** et **integf** dans la librairie **linear_system.f90**.
A ce stade, afin de valider notre code, nous comparons les résultats obtenus de façon analytique à ceux que nous donne le programme.

$$A^K = \begin{pmatrix} \frac{1}{h} + \frac{h}{3} & -\frac{1}{h} + \frac{h}{6} \\ -\frac{1}{h} + \frac{h}{6} & \frac{1}{h} + \frac{h}{3} \end{pmatrix}$$

Et pour $f = x$:

$$b_p^K = \begin{pmatrix} \frac{K}{2} - \frac{1}{3} \\ -\frac{K}{2} - \frac{1}{6} \end{pmatrix}$$

3.5 Assemblage de la matrice A et du second membre b

La sous-routine **CalAB** calcul la matrice A et le second membre b du système linéaire selon l'algorithme *Ass* décrit dans l'énoncé du TP. Encore une fois, on compare les résultats analytiques avec ceux fournis par le programme:

$$A = \begin{pmatrix} \frac{1}{h} + \frac{h}{3} & -\frac{1}{h} + \frac{h}{6} & & & \\ -\frac{1}{h} + \frac{h}{6} & \frac{1}{h} + \frac{h}{3} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \frac{2}{h} + \frac{2h}{3} & -\frac{1}{h} + \frac{h}{6} \\ & & & -\frac{1}{h} + \frac{h}{6} & \frac{2}{h} + \frac{2h}{3} \end{pmatrix}$$

Et pour $f = x$:

$$b = \begin{pmatrix} -\alpha + \frac{h^2}{6} \\ h^2 \\ 2h^2 \\ \vdots \\ (n-1)h^2 \end{pmatrix}$$

3.6 Résolution du système linéaire

Maintenant que le système linéaire est complet, nous pouvons procéder à sa résolution. Pour cela, nous utilisons l'algorithme de Thomas décrite dans les annexes fournis avec l'énoncé du TP.

```
Pour i=2,...,n
     c(i)=a(i,i-1)/a(i-1,i-1)
     a(i,i)=a(i,i)-c(i)a(i-1,i)
     b(i)=b(i)-c(i)b(i-1)
Fin pour
```

Et

```
u(n)=b(n)/a(n,n)
Pour i de n-1 à 1 par -1
     u(i)=(b(i)-a(i,i+1)*u(i+1))/a(i,i)
Fin pour
```

La sous-routine **solve** dans la libraire **solver.f90** met en œuvre cet algorithme.

4 Affichage de la solution

Pour $f = x$, la solution obtenue analytiquement est:

$$usol = -\frac{(\alpha - 1)\sinh(1) + 1}{\cosh(1)}\cosh(x) + (\alpha - 1)\sinh(x)$$

La sous-routine **read_sol** contenue dans la librairie **save_data.f90** affiche la solution et stock ces valeurs dans un fichier *sol.dat*.

5 Calcul de l'erreur

Afin d'évaluer la précision de notre programme et d'en valider les résultats, nous avons développé la sous-routine **error** de la librairie **estimation_error.f90**. Il peut être intéressant également d'analyser le comportement de l'erreur en fonction de n. Une fois de plus, la sous-routine **error** affiche cette information à chaque exécution du code. Ces informations sont consignés dans les figure 1 et 2.

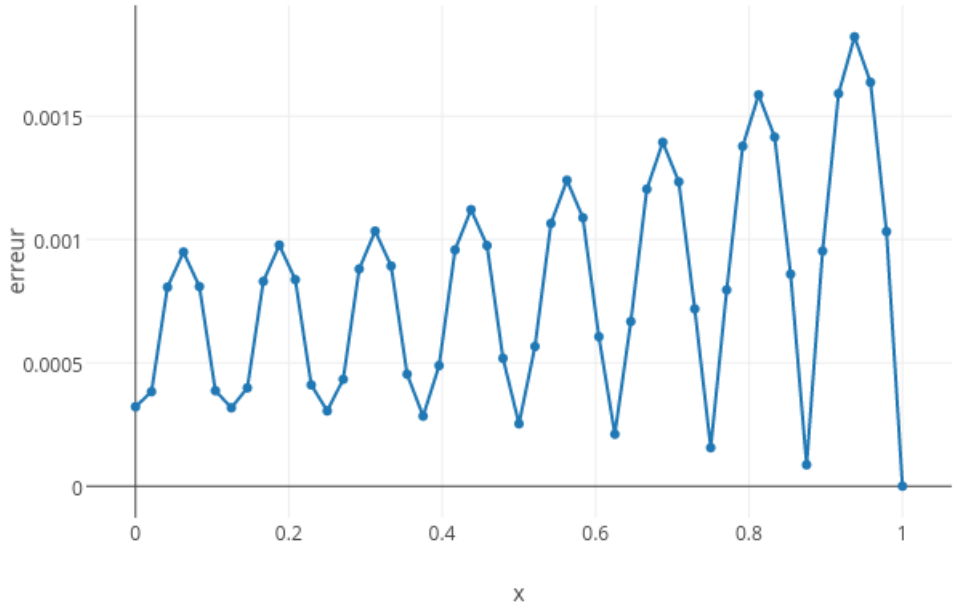

Figure 1: L'erreur pour $\alpha = 1$ et $n = 8$

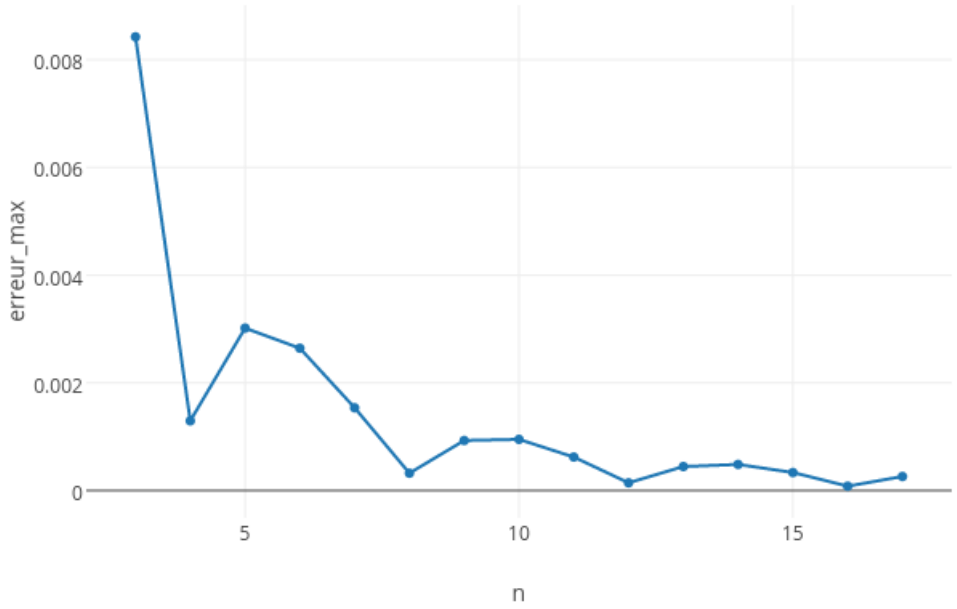

Figure 2: L'erreur maximale pour $\alpha = 1$

6 Conclusion

Lors de ce TP, nous avons pu mettre en œuvre la méthode d'éléments finis en 1D en Fortran. Nous avons comparé les résultats fournis par notre programme à nos données analytiques afin de valider et d'estimer la précision de notre code. Un effort particulier a été apporté à la modularité de notre projet et à la lisibilité du code par rapport à nos développement précédents.