Une méthode multirésolution adaptative pour les systèmes de réaction-diffusion raides

Thierry Dumont

Institut Camille Jordan

14 juin 2022

Plus de détails :

`https://smai-jcm.centre-mersenne.org/item/SMAI-JCM_2017__3__29_0/`

Auteurs : Descombes, Duarte, Dumont, Guillet, Louvet, Massot.

$$\frac{\partial u_i}{\partial t}(x,t) - \operatorname{div}(\varepsilon_i(x)\ \operatorname{grad}\ u_i(x,t)) = f_i(u(x,t)), \quad x \in \Omega \subset \mathbb{R}^d,\ t > 0,$$

$$u_i(x,0) = u_i^0(x), \qquad x \in \Omega;$$

Systèmes...

$$\frac{dU}{dt} = A_\varepsilon U + F(U),$$

On s'intéresse aux cas (fréquents) où :

1. Les systèmes d'EDOs

$$\frac{dU}{dt} = F(U)$$

   sont raides.
2. Diffusion « petite ».
3. n équations $n = 2, 5, \ldots, 20, \ldots 100 \ldots$.

Splitting

- Diffusion D :

$$\frac{dV}{dt} = A_\varepsilon V.$$

- Réaction R :

$$\frac{dW}{dt} = F(W).$$

Splitting

- Diffusion D :

$$\frac{dV}{dt} = A_\varepsilon V.$$

- Réaction R :

$$\frac{dW}{dt} = F(W).$$

Schéma de Strang :

$$U_{n+1} = R_{\Delta t/2} \circ D_{\Delta t} \circ R_{\Delta t/2} U_n.$$

Remarques 1) la réaction R.

Une fois discrétisé en espace, $R_{\Delta t/2}$ est calculable en parallèle.
Mais les problèmes sont **raides :**

- solveur adapté : **Radau5**.
- temps calcul fortement variable d'un point à un autre et en temps (ondes progressives) : $=>$ parallélisme difficile à équilibrer

Remarques 2) la diffusion D.

Diffusion « faible » dans les problèmes intéressants.
$=>$ Utiliser des méthodes de Runge-Kutta explicites et stabilisées :

- ordre n, mais (beaucoup) plus de n pas (Exemples : Rock2, **Rock4**).
- Uniquement des produits matrice $\times$ vecteur (et des combinaisons linéaires) $=>$ aisément parallélisable.

Problèmes types, exemples

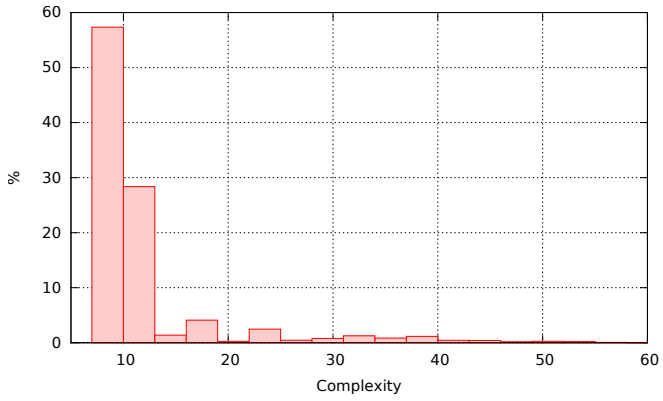
- La réaction de Belusov-Zhabotinsky (BZ). $m = 3$ inconnues. Terme réactif

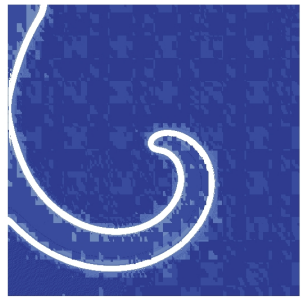
$$\begin{aligned} f_1(u_1, u_2, u_3) &= 10^5\ (-2\ 10^{-2} u_1 - u_2 u_1 + 1.6\ u_3), \\ f_2(u_1, u_2, u_3) &= 10^2\ (u_2 - u_2^2 - u_1(u_2 - 2\ 10^{-2})), \\ f_3(u_1, u_2, u_3) &= u_2 - u_3. \end{aligned}$$

- Un modèle d'AVC. $m = 21$ équations, particulièrement raide. Le Jacobien vers $F(U) = 0$ a des valeurs propres dans $[-10^8, 0[$ (modélisation de canaux ioniques).

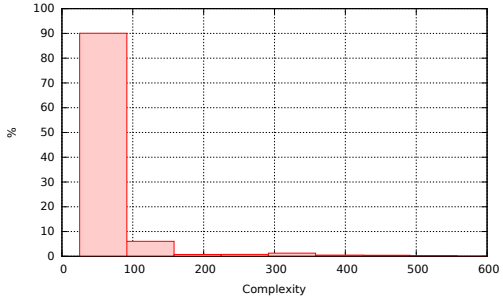
Illustrations. Grille cartésienne 1024×1024.

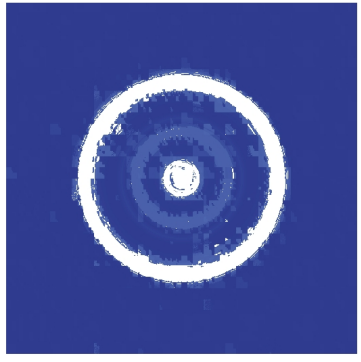
Le calcul de la réaction.





BZ : à droite : zones où le coût calcul est $> 17\times$ le coût minimum.





STROKE. À droite : zones où le coût calcul est $> 80\times$ le coût minimum.

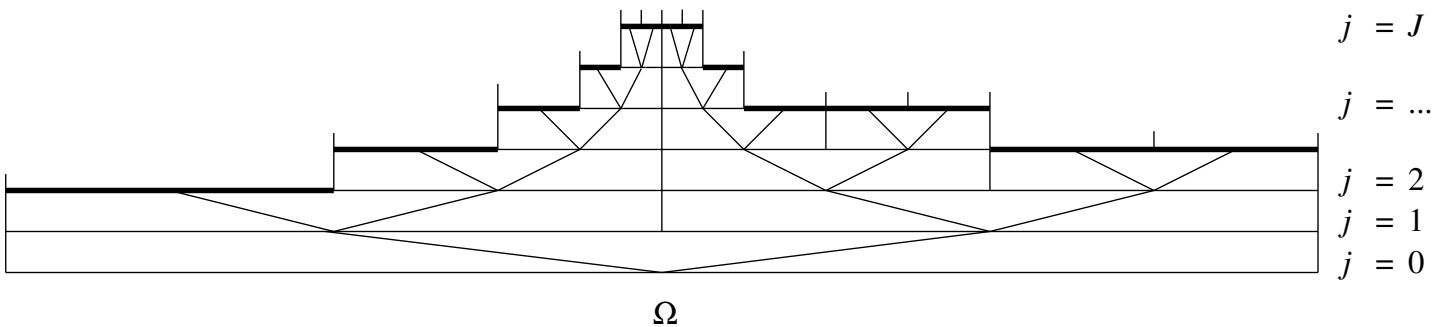
- BZ : 87 % du temps CPU consommé dans les régions où la complexité est inférieure à 17.
- STROKE : 90 % du temps CPU consommé dans les régions où la complexité est inférieure à 60.

- BZ : 87 % du temps CPU consommé dans les régions où la complexité est inférieure à 17.
- STROKE : 90 % du temps CPU consommé dans les régions où la complexité est inférieure à 60.

1. Stratégie de « Maillage adaptatif ».
2. Parallélisation : comment faire ? (les zones coûteuses sont « mobiles ».)

Adaptation de maillage : multiresolution adaptative

$\Omega = [0,1]^d$, $d = 1, 2, 3$
Division récursive dyadique du niveau $j = 1$ au niveau $j = J$.
Volumes finis associé à *toutes* les cellules (nœuds, feuilles) de l'arbre :



On s'intéresse à la solution sur les feuilles, mais la solution est stockée sur toutes les cellules.

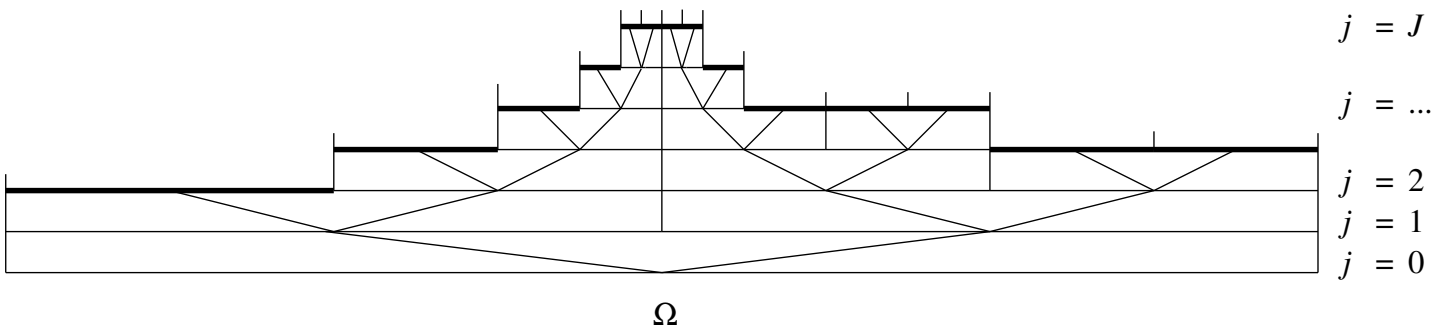
Opérateurs de prolongement et de restriction

- P_{j-1}^{j} (restriction : du niveau j vers le niveau $j-1$: moyenne.
- P_{j}^{j-1} (prolongement : du niveau $j-1$ vers le niveau j : interpolation polynomiale entre deux voisins :

$$\hat{u}_{j+1,2k} = u_{j,k} + \frac{1}{8}(u_{j,k-1} - u_{j,k+1}),$$

$$\hat{u}_{j+1,2k+1} = u_{j,k} + \frac{1}{8}(u_{j,k+1} - u_{j,k-1}).$$

En dimension > 1, produit tensoriel.



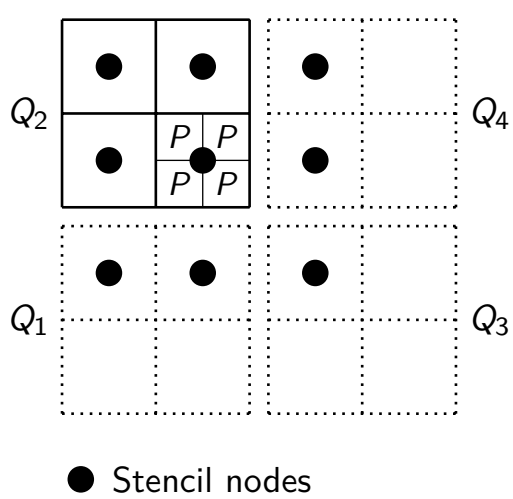
Détails, adaptation

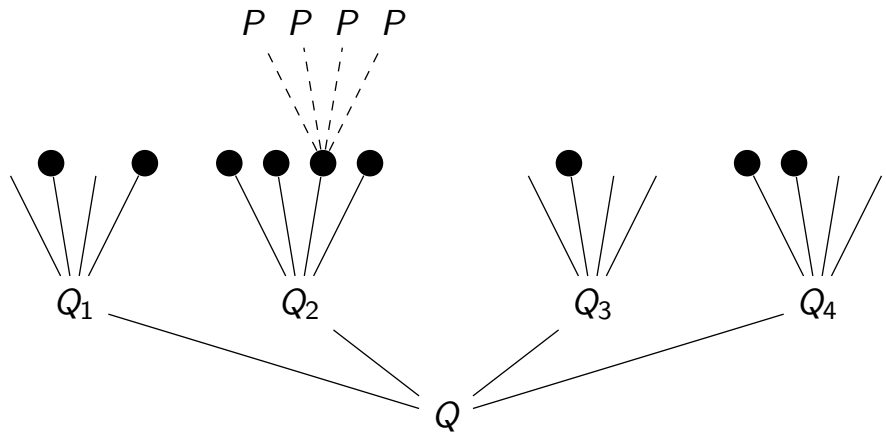
$$d_{j,k} = u_{j,k} - P_j^{j-1} \circ P_{j-1}^j u_{j,k}.$$

Adaptation basée sur la taille des détails.

Implantation : problème de localisation des données

Exemple en dimension 2 :





Nécessite une structure de donnée adaptée.

Z-order, Morton code,...

Au niveau j, les coordonnées x_i d'un nœud P exprimées en base 2 sont :

$$x_i = 0.x_{i,1}x_{i,2}\cdots x_{i,j-1}x_{i,j}.$$

L'abscisse de Morton est définie en intercalant les digits des différentes coordonnées. Exemple en dimension 2 :

$$s = x_{1,1}x_{2,1}x_{1,2}x_{2,2}\cdots x_{1,j}x_{2,j}.$$

Z-order, Morton code,...

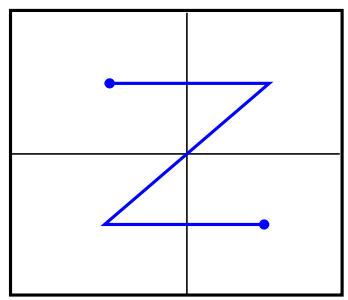
Au niveau j, les coordonnées x_i d'un nœud P exprimées en base 2 sont :

$$x_i = 0.x_{i,1}x_{i,2}\cdots x_{i,j-1}x_{i,j}.$$

L'abscisse de Morton est définie en intercalant les digits des différentes coordonnées. Exemple en dimension 2 :

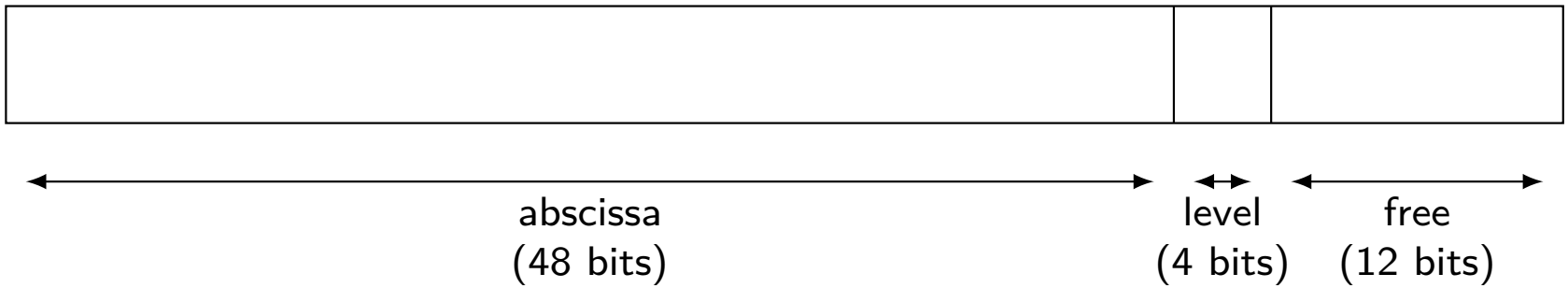
$$s = x_{1,1}x_{2,1}x_{1,2}x_{2,2}\cdots x_{1,j}x_{2,j}.$$

Peut aussi être vue comme une numérotation récursive, par quadrants successifs.



–Le Z–

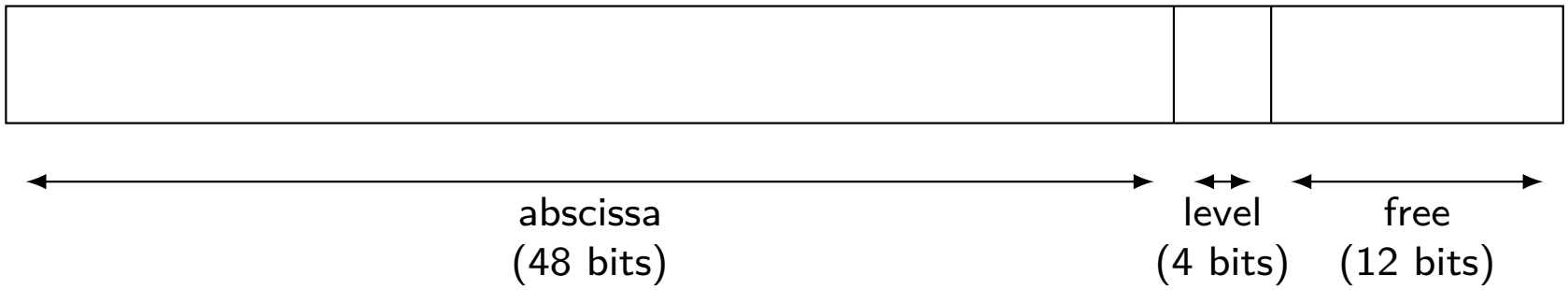
En pratique



Représentation d'un nœud sur un entier 64-bits.

ce qui permet 16 niveaux en dimension 3.

En pratique



Représentation d'un nœud sur un entier 64-bits.

ce qui permet 16 niveaux en dimension 3.

Les nœuds sont stockés dans des blocs. Le bloc k stocke les nœuds d'abscisse $\in [s_k, S_k]$.

- On s'arrange pour que les tailles des blocs soit *à peu près* constantes.
- Les nœuds ne sont pas ordonnés par abscisse croissante dans les blocs (plus efficace !).
- Un *arbre binaire* permet de gérer efficacement la collection de blocs.

Parallélisme

On peut vérifier, moyennant quelques astuces, que le calcul des détails, des raffinements et des dé-raffinements peuvent être effectués en parallèle.

Parallélisation

- Mémoire partagée.
- Threading Building Blocks (TBB, Intel, licence Apache v2.0).

Parallélisme par **vol de tâches** :

- Adaptée aux calculs hétérogènes.
- Composable (utile pour la gestion du maillage).

C++, utilisation « simple ».

Parallélisation

1. Réaction : une tâche = un bloc de nœuds.
2. Diffusion :
   Méthode Rock appliquée à un problème linéaire = appliquer un polynôme de la matrice à un vecteur.
   Schéma de Horner, parallélisation des produits matrice $\times$ vecteur (et des combinaisons linéaires).

Le stockage des inconnues

Stockage nœud par nœud ou stockage inconnue par inconnue ?

Le stockage des inconnues

Stockage nœud par nœud ou stockage inconnue par inconnue ?

L'intensité arithmétique de Radau5 est grande par rapport à celle des produits matrice $\times$ vecteur
$=>$ stockage par inconnues.

Performances

- 2d :

		J		
		8	9	10
BZ	MR	1.14×10^{-2}	2.51×10^{-2}	4.06×10^{-2}
	CM	$\mathbf{1.31 \times 10^{-2}}$ (1.2)	$\mathbf{4.06 \times 10^{-2}}$ (1.6)	$\mathbf{1.78 \times 10^{-1}}$ (4.4)
STROKE	MR	2.4×10^{-2}	4.03×10^{-2}	1.03×10^{-1}
	CM	$\mathbf{3.2 \times 10^{-1}}$ (13.3)	**1.21** (30.0)	**4.80** (46.6)

Performances

- 2d :

		J		
		8	9	10
BZ	MR	1.14×10^{-2}	2.51×10^{-2}	4.06×10^{-2}
	CM	$\mathbf{1.31 \times 10^{-2}}$ (1.2)	$\mathbf{4.06 \times 10^{-2}}$ (1.6)	$\mathbf{1.78 \times 10^{-1}}$ (4.4)
STROKE	MR	2.4×10^{-2}	4.03×10^{-2}	1.03×10^{-1}
	CM	$\mathbf{3.2 \times 10^{-1}}$ (13.3)	**1.21** (30.0)	**4.80** (46.6)

- 3d :

		J	
		8	9
BZ	MR	0.97	5.75
	CM	**3.05** (3.1)	**23.60** (4.1)
STROKE	MR	1.53	10.81
	CM	**76.68** (50.0)	**610.50** (56.5)

Simulations 3D, 10 niveaux max., 40 threads. Pourcentage de temps CPU :

	BZ	**STROKE**
Mesh Adaptation	19.23	7.20
Reaction Solver	**56.23**	**88.89**
Diffusion Solver	24.54	3.91

Quelques éléments pour conclure

- Efficacité.
- Complexité de l'implantation.
- Le vol de tâches :
  - simple en mémoire partagée.
  - Attention aux machines NUMA !
  - Généralisation en mémoire distribuée ?
  - L'avenir en mémoire partagée (cf. Rust) ?