Méthodes numériques pour les EDP.
Projet 3: parallélisation de la résolution d'un problème de contrôle.

Ce projet consiste à paralléliser le problème de contrôle :

$$\min_{c\in L^2(0,T)} J(c) := \|y(T) - y_{cible}\|^2 + \alpha \int_0^T c(t)^2 dt,$$

où l'équation reliant y et c est de type Schrödinger :

$$i\dot{y}(t) = (A + c(t)B)y(t),$$

avec un état initial y_0 fixé.

1 Stratégie de parallélisation

On souhaite paralléliser en temps la résolution de ce problème. Pour ce faire, on utilise la décomposition $[0,T] = \cup_{\ell=1}^{N}[T_{\ell-1}, T_\ell]$. Pour simplifier, et sans perdre de généralité, on définit $T_\ell = \ell T/N$. La méthode que l'on va suivre consiste à itérer trois sous-étapes :

1. Définition d'états intermédiaires λ_ℓ à chaque temps T_ℓ.
2. Résolution en parallèle des sous-problèmes de contrôle sur $[T_{\ell-1}, T_\ell]$.
3. Construction d'un nouveau contrôle par simple concaténation des contrôles partiels obtenus à la sous-étape précédente.

Il reste à préciser la définition des états intermédiaires $\Lambda = (\lambda_\ell)_{\ell=0,\cdots,N}$. Pour un contrôle c arbitraire, on pose $\Lambda^c = (\lambda_\ell^c)_{\ell=0,\cdots,N}$ avec :

$$\lambda_\ell^c = (1-\gamma^\ell)y^c + \gamma^\ell p^c, \tag{1}$$

avec de l'état y^c et de l'adjoint p^c associés associés à c et $\gamma_\ell = T_\ell/T$. Supposons maintenant à l'étape k disposer d'un contrôle c^k, on pose simplement

$$\lambda_\ell^k = \lambda_\ell^{c^k}.$$

2 Étude théorique

Pour étudier théoriquement la méthode précédente, on introduit une nouvelle fonctionnelle :

$$J_{\|}(c,\Lambda) = \sum_{\ell=1}^{N} \beta_\ell J_\ell(c),$$

avec

$$\beta_\ell = \frac{T}{T_{\ell+1} - T_\ell},\ \alpha_\ell = \frac{\alpha}{\beta_\ell}$$

et

$$J_\ell(c_\ell) = \|y_\ell(T_{\ell+1}) - \lambda_{\ell+1}\|^2 + \alpha_\ell \int_{T_\ell}^{T_{\ell+1}} c_\ell(t)^2 dt,$$

où l'équation reliant y_ℓ et c est la même que précédement, à ceci près qu'elle est seulement définie sur $[T_\ell, T_{\ell+1}]$:

$$i\dot{y}_\ell(t) = (A + c_\ell(t)B)y_\ell(t),$$

avec un état initial définit par $y_\ell(T_\ell) = \lambda_\ell$.

1. Pour un contrôle c fixé arbitrairement, démontrer que les états intérmédiaires définis par (1) sont les solutions du problème :
$$\min_\Lambda J_\parallel(c, \Lambda).$$
2. Montrer de plus que
$$J_\parallel(c, \Lambda^c) = J(c).$$
3. Expliquer brièvement pourquoi l'algorithme de parallélisation peut-être interprété comme un algorithme de "directions alternées" ou "descente alternée".

3 Étude pratique

On teste maintenant l'algorithme !

1. Implémenter l'algorithme : on simulera la parallélisation, dans le sens où on se contentera de mettre dans une boucle "for" (portant sur les sous-intervalles) la deuxième sous-étape de l'algorithme. Pour réaliser la deuxième sous-étape, on utilisera soit une méthode de gradient, soit un algorithme monotone, soit les deux (au choix).
2. À l'aide de la commande 'cputime()', mesurer le gain apporté par la parallélisation. On parle de "full efficiency" lorsque le temps de calcul est exactement divisé par le nombre de processeurs impliqués dans la parallélisation.
3. Pourquoi peut-on dire que la première sous-étape ne rentre pas dans le cadre de la parallélisation, c'est-à-dire qu'elle nécessite une résolution sur tout l'intervalle $[0, T]$? Pourquoi empèche-t-elle nécessairement l'obtention d'une "full-efficiency" ?
4. Pour résoudre ce problème, on peut procéder comme suit : sur chaque sous-intervalle, la deuxième sous-étape permet de calculer en parallèle pour chaque sous-intervalle $[T_{\ell-1}, T_\ell]$ l'opérateur P_ℓ associé à la fonction $y_\ell(T_\ell) \mapsto y_\ell(T_{\ell+1})$. Expliquer pourquoi ce calcul permet de se rapprocher de la "full-efficiency".
5. À l'aide de la commande 'cputime()', mesurer le gain apporté par la parallélisation associée à cette nouvelle idée.