Mise en œuvre des approches Différence Fini et Volume Fini

B. Nkonga

February 13, 2009

Contents

1 Différence Fini pour Schrödinger nonlinéaire 1D.

Le but de ce projet est de modèliser numériquement la propagation d'un faisceau laser à l'aide de l'équation de Schrödinger non linéaire:

$$i\partial_t u + \partial_x^2 u = \varepsilon |u|^{2\sigma} u, \quad t \in [0,T], \quad x \in \mathbb{R}, \tag{1}$$

où $u(t,x) : [0,T] \times \mathbb{R} \to \mathbb{C}$, $\varepsilon = \pm 1$ et $\sigma > 0$. $|z|$désigne le module du nombre complexe z. On se donne $u_0(x) := u(0,x)$ pour $x \in \mathbb{R}$.
Pour simplifier l'étude, nous allons nous restreindre au cas où pour tout temps t, l'application $x \mapsto u(t,x)$ est $2L$ périodique pour un certain $L > 0$ si bien qu'il suffit d'étudier u sur l'intervalle $[-L, L]$.

1.1 Deux identités.

On se donne u de classe $\mathcal{C}^2([0,T] \times [-L,L])$ solution de (1).
a. Montrer que pour tout $t \geq 0$

$$\int_{-L}^{L} |u(t,x)|^2 dx = \int_{-L}^{L} |u_0(x)|^2 dx.$$

(Indic: dériver la première intégrale par rapport à t et remplacer $\partial_t u$ par sa valeur donnée par l'équation (1)).
b. Montrer que pour tout $t \geq 0$

$$\int_{-L}^{L} \frac{1}{2}|\partial_x u(t,x)|^2 + \frac{\varepsilon}{2\sigma+2}|u(t,x)|^{2\sigma+2} = \int_{-L}^{L} \frac{1}{2}|\partial_x u_0(x)|^2 + \frac{\varepsilon}{2\sigma+2}|u_0(x)|^{2\sigma+2}.$$

(Indic.: multiplier (1) par $\partial_t \bar{u}$, intégrer sur $[-L,L]$ en x et prendre la partie réelle du résultat.

2.0. Montrer que pour tout $\omega > 0$, $(t,x) \mapsto \dfrac{e^{i\omega t}\sqrt{2\omega}}{\cosh(\sqrt{\omega}t)}$ est solution de (1).

1.2 Cas quadratique, schéma de Crank-Nicolson.

Dans ce paragraphe, on se fixe $\sigma = 1$. Nous allons mettre en place une méthode différences finies pour ce problème. Soit $T > 0$. On se donne N_t et N_x des nombres de pas de temps et d'espace et on pose $\delta t = \dfrac{T}{N_t}$ et $\delta x = \dfrac{2L}{N_x}$ les pas de temps et d'espace. On note u_j^n (pour $j = 1$ à N_x et $n = 0$ à N_t) une valeur approchée de $u(n\delta t, j\delta x)$. Afin de satisfaire la condition de périodicité on pose $u_0^n = u_{N_x}^n$ pour tout n et $u_{N_x+1}^n = u_1^n$ pour tout n. Le schéma aux différences

finies s'écrit

$$i\frac{u_j^{n+1}-u_j^n}{\delta t}+\frac{1}{\delta x^2}\left[\frac{(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})+(u_{j+1}^n-2u_j^n+u_{j-1}^n)}{2}\right] \\ =\varepsilon\frac{|u_j^{n+1}|^2+|u_j^n|^2}{2}\left(\frac{u_j^{n+1}+u_j^n}{2}\right) \tag{2}$$

pour $j=1$ à N_x et $n=0$ à N_t.

On pose $U^n=\begin{pmatrix} u_1^n \\ \vdots \\ u_{N_x}^n \end{pmatrix}$ et

$$M=\frac{1}{\delta x^2}\begin{pmatrix} -2 & 1 & \dots 0\dots & \dots 0\dots & 1 \\ 1 & -2 & 1 & \dots 0\dots & \dots 0\dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots 0\dots & 1 & -2 & 1 & \dots 0\dots \\ \dots 0\dots & \dots 0\dots & 1 & -2 & 1 \\ 1 & \dots 0\dots & \dots 0\dots & 1 & -2 \end{pmatrix}$$

Pour $X=\begin{pmatrix} X_1 \\ \vdots \\ X_{N_x} \end{pmatrix}$ et pour toute fonction $f:\mathbb{C}\to\mathbb{C}$, on note $f(X)$ le vecteur $\begin{pmatrix} f(X_1) \\ \vdots \\ f(X_{N_x}) \end{pmatrix}$ et si X et Y sont dans $\mathbb{R}^{N_x}$, on note (comme en fortran 90!) $X.*Y$ le produit terme à terme c'est à dire le vecteur de composantes X_iY_i.

2.1. Montrer que les quantités suivantes sont des invariants discrets, *i.e.* toute solution de (2) vérifie

$$\sum_{j=1}^{N_x}|u_j^n|^2=\sum_{j=1}^{N_x}|u_j^0|^2,$$

et

$$\sum_{j=1}^{N_x}\frac{|u_{j+1}^n-u_j^n|^2}{\delta x^2}+\frac{\varepsilon}{2}|u_j^n|^4=\sum_{j=1}^{N_x}\frac{|u_{j+1}^0-u_j^0|^2}{\delta x^2}+\frac{\varepsilon}{2}|u_j^0|^4.$$

2.2. Montrer que (2) s'écrit

$$i\frac{U^{n+1}-U^n}{\delta t}+M\frac{U^{n+1}+U^n}{2}=\varepsilon\frac{|U^{n+1}|^2+|U^n|^2}{2}.*\frac{U^{n+1}+U^n}{2}. \tag{3}$$

2.3 L'itération: On effectue une résolution de approchée de (3) par méthode itérative. On se donne U^N, on prend comme inconnue $Z=\dfrac{U^{n+1}+U^n}{2}$et (3)

s'écrit:

$$\frac{2i}{\delta t}(Z-u^n)+MZ=\varepsilon\left(\frac{|2Z-U^n|^2+|U^n|^2}{2}\right).*Z.$$

Pour trouver Z, on prend $Z_0=U^n$ et on construit par récurrence Z_p par

$$\frac{2i}{\delta t}(Z_{p+1}-u^n)+MZ_{p+1}=\varepsilon\left(\frac{|2Z_p-U^n|^2+|U^n|^2}{2}\right).*Z_{p+1}.$$

On pose

$$M_p=\frac{2i}{\delta t}Id+M-\frac{\varepsilon}{2}\left(\frac{|2Z_p-U^n|^2+|U^n|^2}{2}\right).$$

2.3.a Montrer que M_p est inversible (Calculer la partie imaginaire du produit scalaire de MX avec X pour tout vecteur X).
2.3.b Montrer que pour tout p

$$\sum_{j=1}^{N_x}|Z_j^p|^2=\sum_{j=1}^{N_x}|u_j^n|^2.$$

On forme

$$\varepsilon_p=\frac{\sum_{j=1}^{N_x}|Z_{p+1}^j-Z_p^j|^2}{\sum_{j=1}^{N_x}|u_j^n|^2}$$

et on arrête l'itération lorsque ε_p est assez petit.

Algorithme

- Définir L, T, N_x, N_t.
- En déduire δt, δx.
- Définir la donnée initiale u_0 par " function".
- Fabriquer M (si possible en en matrice creuse ("sparse" en scilab)).
- Se donner une tolérance sur l'itération: $tol=\eta*min(\delta t^2,\delta x^2)$.
- BOUCLE EN TEMPS :: POUR $i=1$ À N_t FAIRE
  1. résidu =1, $Z=U$,
  2. tant que résidu > tol faire:
     - Fabriquer $M_p=\frac{2i}{\delta t}Id+M-\frac{\varepsilon}{2}(|2Z-U|^2+|U|^2)$ (si possible en matrice creuse).
     - Résoudre $M_pY=\frac{2i}{\delta t}$.
     - Calculer le résidu $=\frac{\sum_{j=1}^{N_x}|Z_j-Y_j|^2}{\sum|u_j^n|^2}$.

- $Z := Y$.

3. fin
4. U=2Z-U

- FIN DE LA BOUCLE EN TEMPS.

2.4. Implémenter l'algorithme en Fortran 90 (et en scilab). A chaque pas de temps, sauver les invariants discrets définis en **II.1.**.
2.6. On prend $L = 5$, $T = 5$, $\omega = 4$, $\eta = 10^{-4}$. Calculer numériquement l'erreur L^2 produite par le schéma sur la solution précédente pour différentes valeurs de N_x et N_t (Ex.: $N_t = N_x = 50, 100, 200....$ Quel est l'ordre du schéma?
Jusifier a posteriori la définition du paramètre tol.
2.7. Etudier l'influence du paramètre η sur la précision? Sur la rapidité d'exécution?
2.8. Tracer les invariants discrets en fonction du temps. Commentez.

1.3 Schéma de relaxation.

On utilise maintenant

$$\begin{cases} i\dfrac{u^{n+1}-u^n}{\delta t} + M\dfrac{u^{n+1}+u^n}{2} = \varepsilon\varphi^{n+1/2}.*\dfrac{u^{n+1}+u^n}{2}, \\ \dfrac{\varphi^{n+1/2}+\varphi^{n-1/2}}{2} = |u^n|^2. \end{cases} \tag{4}$$

La valeur de $\varphi^{-1/2}$ est obtenue par un demi-pas de temps d'Euler explicite:

$$i\frac{u^0-u^{-1/2}}{\delta t/2} + Mu^0 = \varepsilon|u^0|^2u^0$$

et $\varphi^{-1/2} = |u^{-1/2}|^2$.
Implémenter ce schéma. Comparer ses performances et ses résultats par rapports aux précédent (erreur par rapport à la solution exacte, invariants...)

1.4 Cas critique.

On considère maintenant $\sigma = 2$:

$$i\partial_t u + \partial_x^2 u = \varepsilon|u|^4u.$$

Adapter le schéma du **3** à ce cas. Prendre $u_0(x) = \lambda e^{-0.5*x^2}$. Faire augmenter λ jusqu'à voir une singularité. Rafiner le maillage.

2 Volume Fini pour Loi de Conservation 1D.

- La forme continue du problème: $\boldsymbol{u}(x,t)$

$$\partial_t \boldsymbol{u} + \partial_x \left(f(\boldsymbol{u})\right) = 0, \quad \boldsymbol{u}(x,0) = \boldsymbol{u}_0(x), \quad \boldsymbol{u}(x+L,t) = \boldsymbol{u}(x,t) \tag{5}$$

- Maillage structuré: $t^n = n\delta t$, $x_j = j\delta x$, $n = 0, Nmax$ et $j = 1, Ns$

$$x_1 = \delta x, \quad x_{Ns} = L, \quad \delta x = \frac{L}{Ns}$$

- La solution approchée est définie par:

$$\boldsymbol{v}(x,t) = \boldsymbol{v}_i^n \quad \forall (x,t) \in]x_{i-\frac{1}{2}},\ x_{i+\frac{1}{2}}[\times[t^n,\ t^{n+1}[, \tag{6}$$

$$\boldsymbol{v}_i^0 = \frac{1}{\delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \boldsymbol{u}_0(x)dx \simeq \boldsymbol{u}_0(x_i) \tag{7}$$

2.1 Schéma et Flux Numériques.

- Schéma conservatif:

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n - \frac{\delta t}{\delta x}\left(\boldsymbol{\phi}_{i+\frac{1}{2}} - \boldsymbol{\phi}_{i-\frac{1}{2}}\right) \quad i = 1, ..., Ns \tag{8}$$

- Flux de Lax-Friedrichs :

$$\boldsymbol{\phi}_{i+\frac{1}{2}} = \boldsymbol{\phi}^{LF}(\boldsymbol{v}_i^n, \boldsymbol{v}_{i+1}^n) = \frac{1}{2}\left(\mathbf{f}_{i+1}^n + \mathbf{f}_i^n + \frac{\delta x}{\delta t}\left(\boldsymbol{v}_i^n - \boldsymbol{v}_{i+1}^n\right)\right)$$

- Flux de Murman-Roe :

$$\boldsymbol{\phi}_{i+\frac{1}{2}} = \boldsymbol{\phi}^{MR}(\boldsymbol{v}_i, \boldsymbol{v}_{i+1}) = \frac{1}{2}\left(\mathbf{f}_{i+1}^n + \mathbf{f}_i^n + \left|\frac{\Delta \mathbf{f}_{i+\frac{1}{2}}^n}{\Delta \boldsymbol{v}_{i+\frac{1}{2}}^n}\right|\left(\boldsymbol{v}_{i+1}^n - \boldsymbol{v}_i^n\right)\right)$$

avec $\Delta X_{i+\frac{1}{2}}^n = X_{i+1}^n - X_i^n$

- Flux de Lax-Wendroff :

$$\boldsymbol{\phi}^{LW}(\boldsymbol{v}_i, \boldsymbol{v}_{i+1}) = \frac{1}{2}\left(\mathbf{f}_{i+1}^n + \mathbf{f}_i^n - \frac{\delta x \Delta \mathbf{f}_{i+\frac{1}{2}}^n}{\delta t \Delta \boldsymbol{v}_{i+\frac{1}{2}}^n}\left(\mathbf{f}_{i+1}^n - \mathbf{f}_i^n\right)\right)$$

Condition de stabilité approchée:

$$\max\left|\frac{\delta t}{\delta x}\frac{\Delta \mathbf{f}_{i+\frac{1}{2}}^0}{\Delta \boldsymbol{v}_{i+\frac{1}{2}}^0}\right| = CFL < \frac{1}{2} \Longrightarrow \delta t = \frac{\delta x CFL}{\max\left|\frac{\Delta \mathbf{f}_{i+\frac{1}{2}}^0}{\Delta \boldsymbol{v}_{i+\frac{1}{2}}^0}\right|}$$

2.2 Algorithme

- $\mathbf{f}(\boldsymbol{v})$ et $\boldsymbol{u}_0(x)$ des fonctions données.
- L, Ns, CFL, T paramètres donnés, $Tl = 0$.
- Initialisations : $\delta x = \frac{L}{Ns}$,
- Initialisations : $x_j = j\delta x$ pour $j = 1, Ns$.
- Initialisations : $\boldsymbol{v}_i^n = \boldsymbol{v}_i^0 \simeq \boldsymbol{u}_0(i\delta x)$
- Initialisations : $\mathbf{f}_i^0 \simeq \mathbf{f}(\boldsymbol{v}_i^0)$ pour $i = 1, Ns$.
- Initialisations : $\delta t = \frac{\delta x CFL}{\max \left| \frac{\Delta \mathbf{f}^0_{i+\frac{1}{2}}}{\Delta \boldsymbol{v}^0_{i+\frac{1}{2}}} \right|}$,
- Initialisations : $Nmax \simeq INTEGER(T/\delta t)$.
- BOUCLE EN TEMPS: POUR $kt = 1, Nmax + 1$
  - Calculer les flux $\boldsymbol{\phi}_{i+\frac{1}{2}}$ pour $i = 1, Ns - 1$.
  - Conditions aux limites périodiques: $\boldsymbol{\phi}_{\frac{1}{2}} = \boldsymbol{\phi}_{Ns+\frac{1}{2}} = \boldsymbol{\phi}(\boldsymbol{v}_{Ns}^n, \boldsymbol{v}_1^n)$.
  - Evolution en temps: $\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n - \frac{\delta t}{\delta x}\left(\boldsymbol{\phi}_{i+\frac{1}{2}} - \boldsymbol{\phi}_{i-\frac{1}{2}}\right)$.
  - $Tl = Tl + \delta t$, si $T - Tl < \delta t$ alors $\delta t = T - Tl$
  - Sauvegardes (tous les kt multiple de $IPasSave$)
  - Impressions sur le déroulement du calcul (tous les kt multiple de $IPasImpre$)
  - Passage au temps suivant: $\boldsymbol{v}_i^n = \boldsymbol{v}_i^{n+1}$
- FIN DE LA BOUCLE EN TEMPS
- Sauvegarde de la solution au temps T.
- FIN DU PROGRAMME

2.3 Mise en œuvre En fortran 90.

Paramètre	Caractéristiques en F90			
	Type	Rang	Dim	Variable
i	INTEGER	1	1	i
j	INTEGER	1	1	j
Ns	INTEGER	1	1	Ns
n ou kt	INTEGER	1	1	n
$IPasSave$	INTEGER	1	1	IPasSave
$ISave$	INTEGER	1	1	ISave
$IPasImpre$	INTEGER	1	1	Impre
$nmax$	INTEGER	1	1	Nmax
L	REAL	1	1	L
T	REAL	1	1	T
Tl	REAL	1	1	Tl
CFL	REAL	1	1	CFL
δx	REAL	1	1	Dx
δt	REAL	1	1	Dt
x_j	REAL	1	Ns	Xp
$\phi_{j+\frac{1}{2}}$	REAL	1	0:Ns	FluxN
$\boldsymbol{u}_j^n$	REAL	1	1:Ns	Rho
$\boldsymbol{u}_j^{n+1}$	REAL	1	1:Ns	RhoNew
$\max \left\lvert \frac{\Delta \mathbf{f}^0_{i+\frac{1}{2}}}{\Delta \boldsymbol{v}^0_{i+\frac{1}{2}}} \right\rvert$	REAL	1	1	Cmax
$\boldsymbol{u}_0(x)$	REAL FUNCTION	1	1	RhoInit
$\mathbf{f}$	REAL FUNCTION	1	1	Flux
$\boldsymbol{\phi}^{LF}(\boldsymbol{v}_i, \boldsymbol{v}_j)$	REAL FUNCTION	1	1	FluxLF
$\boldsymbol{\phi}^{LW}(\boldsymbol{v}_i, \boldsymbol{v}_j)$	REAL FUNCTION	1	1	FluxLW
$\boldsymbol{\phi}^{MR}(\boldsymbol{v}_i, \boldsymbol{v}_j)$	REAL FUNCTION	1	1	FluxMR

2.3.1 Déclarations En fortran 90.

```
INTEGER :: Ns, Nmax,  n, i,  j, Impre, IPasSave, ISave
REAL    ::  L,   Dx, Dt, T, Tl,  Cmax, CFL

REAL, DIMENSION(:), ALLOCATABLE :: Rho, RhoNew
REAL, DIMENSION(:), ALLOCATABLE ::  Xp, FluxN

CHARACTER(LEN=50)               :: RootName =''Sorties ‘‘, NewName
```

2.3.2 Sauvegarde de la solution à un temps donné

```
IF( MOD(n,IPasSave) == 0 ) THEN
 ISave = ISave+1
 CALL  TheNewName(RootName, ISave, 4, NewName)
 Open(10,FILE=TRIM(NewName)//".xmgr")
 DO i = 1, Ns
  WRITE(10, '(1x,2(EN15.8,1x) )') X(i), V(i)
 END DO
 CLOSE(10)
END IF
```

2.4 Structuration du logiciel en fortran 90.

Organisation:

- Un module `LesFonctions` de fonctions (réutilisable dans d'autres contextes) contenant les fonctions .
- Un programme principal `LdcNonLineaire` avec un mot clé (`RootName`) donné à l'exécution du programme.
- Lectures des données dans un fichier formaté : (`RootName`)`.data`.
- Sauvegardes dans des fichiers de la forme (`RootName`)`_`(`n`)`.xmgr`.
  où `n` est le numéro de l'itération à laquelle la sauvegarde a lieu.

2.5 Fonctions Intrinèques Utililes.

- `TRIM` : Construit une nouvelle chaîne de caractère en Supprimant les blancs à la fin d'une autre chaîne de caractère. `NewName = TRIM(OldName)`
- `AJUSTL` : Supprime les blancs au début d'une chaîne de caractère. `NewName = AJUSTL(OldName)`
- `LEN_TRIM` : Donne la taille d'une chaîne de caractère, ne tenant pas en compte des blancs à la fin. `il=LEN_TRIM(OldName)`
- `WRITE` : utilisé ici pour transformer un entier en chaîne de caractère. `WRITE(OldName,*) n` mais aussi pour écrire dans un fichier ou à l'écran.

2.6 Plan du travail

(1H10) Développer un programme de résolution de l'équation (5) en utilisant le schéma numérique (6)-(7)-(8) et un flux de votre choix. Application: `L=2`π, Ns=2000, CFL=0.5, T=2, IPasSave =1000, Impre =10+, $\boldsymbol{u}_0(x) = \sin(x)$, $\mathbf{f}(\boldsymbol{v}) = \boldsymbol{v}$,

10mn Evaluer le temps de calcul du programme compilé avec les options *-r8 -O0* . Optimiser les performances du programme en utilisant d'autres options de compilation.

10mn Analyser les performances du programme en utilisant la commande *prof*. Inclure dans le programme, des appels permettant de mesurer le coût des principales composantes du programme.

2.7 Fonctions F90 à utiliser

- `CPU_TIME` temps CPU écoulé depuis le début du programme.

```
CALL CPU_TIME(TDebut)
  ........

CALL CPU_TIME(TFin)
WRITE(6,*) `` Temps dans cette partie = '', TFin-TDebut
```

- ```
  CALL GETARG(1, RootName)
  WRITE(6,*) `` Nom donne a l'execution du programme = '', RootName
  ```

- Procédure de construction de Nom de fichier `TheNewName`.

```
CALL    TheNewName(RootName, n, Nm, NewName)
```

Construit un nom de fichier `NewName` à partir d'une chaîne de caractère `RootName` et d'un nombre entier `n` représenté sur `Nm` caractères et complété au besoin par des zéros au début.

```
SUBROUTINE TheNewName(Name, n, Nm, NewName)
 CHARACTER(LEN=*), INTENT(IN)  :: Name
 INTEGER         , INTENT(IN)  :: n, Nm
 CHARACTER(LEN=*), INTENT(OUT) :: NewName
 INTEGER             :: l , m
 CHARACTER(LEN=20)   :: NB, Ze

 Ze ="000000000000000";  WRITE(NB, *) n
 NB = ADJUSTL(NB);  l= LEN_TRIM(NB) ; m=Nm-l
 IF( l > 0 ) THEN
  NewName=TRIM(Name)//"_"//Ze(1:m)//TRIM(NB)
 ELSE
  NewName=TRIM(Name)//"_"//Ze(1:Nm)
 END IF
END SUBROUTINE TheNewName
```
```