Projet S8
Contrôlabilité à zéros de l'équation des ondes 1D

Professeurs encadrants

CHAUTARD Alexandre — Mme DELOURME

RANDIMBIARISON Sarobidy — M. AUDUSSE

2020 - 2021

Table des matières

1 Introduction

Ce projet porte sur la contrôlabilité à zéros de l'équation des ondes unidimensionnelle, qu'on retrouve par exemple dans la modélisation d'une corde de guitare. L'objectif est de trouver la fonction contrôle frontière v qui permet d'amener la solution d'une équation des ondes à 0 en un temps T. $v \in L^2(0,T)$ est telle que la solution de l'équation des ondes :

$$\begin{cases} \partial_{tt}^2 y - \partial_{xx}^2 y = 0 \qquad t > 0, x \in]0,1[\\ y(t,0) = 0 \qquad t > 0 \\ y(t,1) = v(t) \qquad t > 0 \\ y(0,x) = y_0 \qquad x \in [0,1] \\ \partial_t y(0,x) = y_1 \qquad x \in [0,1] \end{cases} \tag{1}$$

vérifie

$$y(T,x) = 0 \quad \text{et} \quad \partial_t y(T,x) = 0 \qquad x \in]0,1[\tag{2}$$

où $T \geq 2$ et y_0, y_1 sont des fonctions très régulières définies sur [0,1] et s'annulant en 0 et en 1.

L'existence d'une telle fonction v est admise pour $T \geq 2$ et on cherche celle dont la norme L^2 est minimale, c'est-à-dire celle qui minimise la fonctionnelle

$$J(v) = \frac{1}{2} \int_0^T \eta(t)(v(t))^2 dt.$$

η est une fonction régulière : on peut par exemple la choisir égale à 1 ou vérifiant

$$\eta(0) = \eta'(0) = 0 \qquad \eta(T) = \eta'(T) = 0$$

et

$$\eta(t) = 1 \quad t \in]\delta, T - \delta[\quad 0 < \delta < 0,5.$$

En fait, la théorie de la dualité en optimisation nous permet de montrer que le contrôle v^* peut être obtenu en minimisant la fonctionnelle

$$J_1(e_0, e_1) = \frac{1}{2} \int_0^T \eta(t)(\partial_x u(t,1))^2 dt + \int_0^1 y_0(x) \partial_t u(0,x) dx - \int_0^1 y_1(x) u(0,x) dx \tag{3}$$

où u est solution de :

$$\begin{cases} \partial_{tt}^2 u - \partial_{xx}^2 u = 0 \qquad t > 0, x \in [0,1] \\ u(t,0) = 0 \qquad t > 0 \\ u(t,1) = 0 \qquad t > 0 \\ u(0,x) = e_0 \qquad x \in [0,1] \\ \partial_t u(0,x) = e_1 \qquad x \in [0,1] \end{cases} \tag{4}$$

Pour avoir le contrôle v^*, on trouve le minimum $e^* = (e_0^*, e_1^*)$ de J_1. Une fois qu'on a ce minimum, v est donné par :

$$v = -\eta(t)\partial_t u^*(0, x)$$

où u^* est la solution de l'équation (4) avec comme données initiales (e_0^*, e_1^*). On va voir dans la suite que e^* s'avère être également l'unique solution de l'équation linéaire :

$$\Lambda e^* = f \tag{5}$$

où

$$f = (-y_1(x), y_0(x))$$

et Λ est un opérateur que l'on définira dans la partie II. L'objectif de ce projet est de résoudre l'équation (5) numériquement.

2 Construction de l'opérateur Λ

2.1 Equation des ondes avec données initiales

Dans cette partie, on cherche à définir l'opérateur Λ pour résoudre plus tard l'équation linéaire $\Lambda e^* = f$.

Pour cela, on se donne comme donnée initiale le couple (e_0, e_1) et on construit u une solution de l'équation des ondes :

$$\begin{cases} \partial_{tt}^2 u - \partial_{xx}^2 u = 0 \qquad t > 0, x \in [0,1] \\ u(t,0) = 0 \qquad t > 0 \\ u(t,1) = 0 \qquad t > 0 \\ u(0,x) = e_0 \qquad x \in [0,1] \\ \partial_t u(0,x) = e_1 \qquad x \in [0,1] \end{cases} \tag{6}$$

Par ailleurs, pour u solution de (7), on peut montrer que l'énergie E définie par :

$$E(t) := \frac{1}{2}\int_0^1 |\partial_t u(t,x)|^2 dx + \frac{1}{2}\int_0^1 |\partial_x u(t,x)|^2 dx$$

se conserve au cours du temps, c'est-à-dire : $\forall t > 0, E(t) = E(0)$.
On a :

$$\begin{aligned} E'(t) &= \frac{1}{2}\frac{d}{dt}\int_0^1 |\partial_t u(t,x)|^2 dx + \frac{1}{2}\frac{d}{dt}\int_0^1 |\partial_x u(t,x)|^2 dx \\ &= \frac{1}{2}\int_0^1 2\partial_t u(t,x)\frac{d}{dt}(\partial_t u(t,x))dx + \frac{1}{2}\int_0^1 2\partial_x u(t,x)\frac{d}{dt}(\partial_x u(t,x))dx \\ &= \int_0^1 \partial_t u(t,x)\partial_{tt} u(t,x)dx + \int_0^1 \partial_x u(t,x)\partial_{xt} u(t,x)dx. \end{aligned}$$

On fait une intégration par parties en espace dans la deuxième intégrale :

$$\begin{aligned} E'(t) &= \int_0^1 \partial_t u(t,x)\partial_{tt} u(t,x)dx + \underbrace{\left[\partial_t u(t,x)\partial_x u(t,x)\right]_0^1}_{=0} - \int_0^1 \partial_t u(t,x)\partial_{xx} u(t,x)dx. \\ &= \int_0^1 \partial_t u(t,x)\underbrace{(\partial_{tt} u(t,x) - \partial_{xx} u(t,x))}_{=0} dx \\ &= 0 \end{aligned}$$

Donc E est constante au cours du temps.

On va maintenant discrétiser l'équation (6) par une méthode de différences finies centrées. On obtient le schéma suivant :

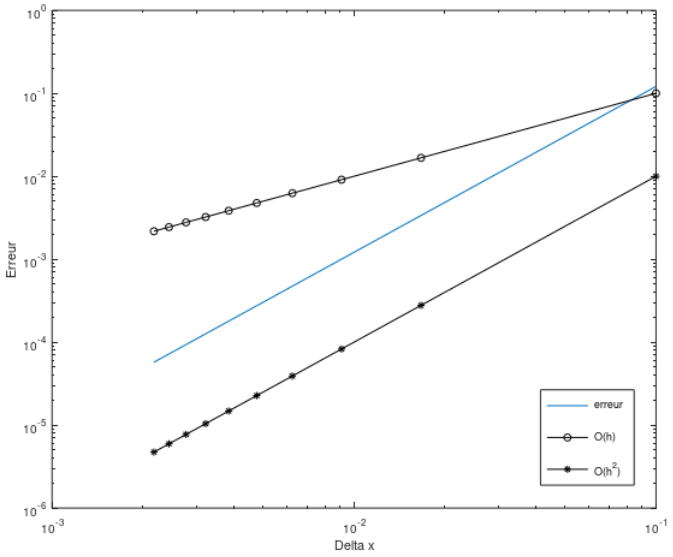
$$\begin{cases} \frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\Delta t^2} - \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} = 0 \\ u_0^n = u_N^n = 0 \qquad \forall n \geq 0 \\ u_j^0 = e_0(x_j) \\ \frac{u_j^1 - u_j^0}{\Delta t} - \frac{\Delta t}{2\Delta x^2}(u_{j+1}^0 - 2u_j^0 + u_{j-1}^0) = e_1(x_j) \end{cases}$$

Ce schéma explicite est stable sous la condition $\Delta t \leq \Delta x$. On veillera à la respecter quand on implémentera le code. En fait, notre fonction Octave dépendra du nombre de points en espace N_x et d'un réel a compris entre 0 et 1, avec $\Delta t = a\Delta x$.

Notre fonction Octave s'écrit :

```
function [t,x,u] = EquationOnde(EDP,a,Nx)

  delta_x = (EDP.b - EDP.a)/Nx;
  delta_t = a*delta_x;
  Nt = ceil((EDP.T - EDP.t0)/delta_t);
  t = zeros(Nt+1,1);
  x = zeros(Nx+1,1);
  u = zeros(Nx+1,Nt+1);
  alpha = (delta_t^2)/(delta_x^2);

  t = EDP.t0:delta_t:Nt*delta_t;
  x = EDP.a:delta_x:EDP.b;

  v = alpha*ones(1,Nx);
  Dh = diag(v,1);
  Db = diag(v,-1);
  Mat = 2*(1-alpha)*eye(Nx+1,Nx+1) + Dh + Db;
  Mat(1,1) = 1;
  Mat(1,2) = 0;
  Mat(Nx+1,Nx) = 0;
  Mat(Nx+1,Nx+1) = 1;
  u(:,1) = EDP.e0(x);
  for j = 2:Nx
    u(j,2) = u(j,1) + delta_t*EDP.e1(x(j)) + (delta_t^2/(2*delta_x^2))*(u(j+1,1)-2*u(j,1)+u(j-1,1));
  end

  for i = 3:Nt+1
   u(:,i) = Mat*u(:,i-1) - u(:,i-2);
   u(1,i) = EDP.ua(t(i));
   u(Nx+1,i) = EDP.ub(t(i));
  end
endfunction
```

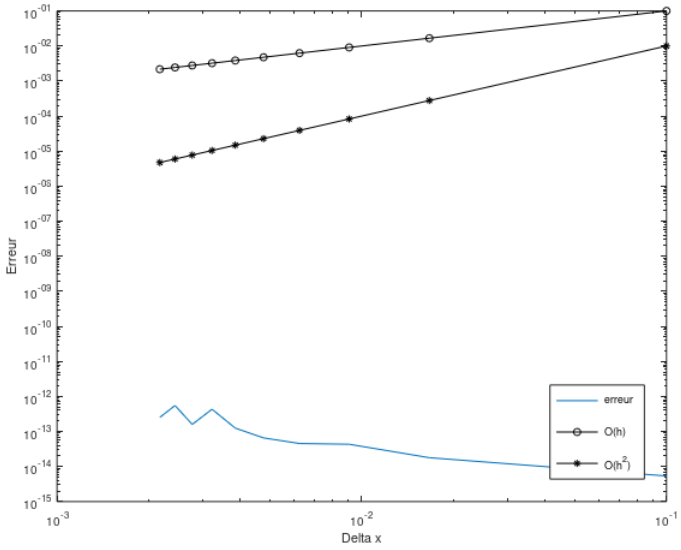
Après avoir fait la résolution numérique de l'équation (6) par le schéma ci-dessus, on observe pour différentes valeurs croissantes du pas d'espace Δx les erreurs suivantes :



Erreur du schéma (7) en fonction du pas d'espace Δx, $a = 0,5$

On retrouve bien l'ordre 2 du schéma en espace et en temps. On précise qu'on a pris ici $u : (t, x) \mapsto sin(\pi x)cos(\pi t)$ comme solution exacte.

Pour $\Delta t = \Delta x$, on observe :



Erreur du schéma (7) en fonction du pas d'espace Δx, $a = 1$

2.2 Equation des ondes rétrograde

Connaissant u solution de l'équation des ondes (6), on calcule y solution de l'équation des ondes "rétrograde" (car on va de T en 0 pour la résoudre) :

$$\begin{cases} \partial_{tt}^2 y - \partial_{xx}^2 y = 0 \qquad t < T, x \in]0,1[\\ y(t,0) = 0 \qquad t < T \\ y(t,1) = -\eta(t)\partial_x u(t,1) \qquad t < T \\ y(T,x) = 0 \qquad x \in]0,1[\\ \partial_t y(T,x) = 0 \qquad x \in]0,1[\end{cases} \tag{7}$$

On écrit le schéma de différences finies correspondant au problème (8) :

$$\begin{cases} \frac{y_j^{n+1} - 2y_j^n + y_j^{n-1}}{\Delta t^2} - \frac{y_{j+1}^n - 2y_j^n + y_{j-1}^n}{\Delta x^2} = 0 \\ y_0^n = 0 \qquad \forall n > 0 \\ y_{Nx}^n = -\eta(t^n)\Big(\frac{u_{Nx}^n - u_{Nx-1}^n}{\Delta x}\Big) \\ y_j^0 = 0 \\ \frac{y_j^{Nt} - y_j^{Nt-1}}{\Delta t} = 0 \end{cases}$$

On l'implémente ensuite sous Octave de la même façon que le précédent, avec en paramètres la structure EDP, Δx et a :

```
function [t,x,y] = EquationOnde2(EDP,a,Nx,u)

  delta_x = (EDP.b - EDP.a)/Nx;
  delta_t = a*delta_x;
  Nt = ceil((EDP.T - EDP.t0)/delta_t);
  t = zeros(Nt+1,1);

  x = zeros(Nx+1,1);
  y = zeros(Nx+1,Nt+1);
  alpha = (delta_t^2)/(delta_x^2);

  t = EDP.t0:delta_t:Nt*delta_t;
  x = EDP.a:delta_x:EDP.b;

  v = alpha*ones(1,Nx);
  Dh = diag(v,1);
  Db = diag(v,-1);
  Mat = 2*(1-alpha)*eye(Nx+1,Nx+1) + Dh + Db;
  Mat(1,1) = 1;
  Mat(1,2) = 0;
  Mat(Nx+1,Nx) = 0;
  Mat(Nx+1,Nx+1) = 1;
  y(:,Nt+1) = EDP.e0(x);
  y(:,Nt) = y(:,Nt+1) - delta_t*EDP.e1(x)';

  for i = Nt-1:1
   y(:,i) = Mat*y(:,i+1) - y(:,i+2);
   y(1,i) = EDP.ua(t(i));
   y(Nx+1,i) = EDP.ub(t(i))*(u(Nx+1,i) - u(Nx,i)/delta_x);
  end
endfunction

```

On peut définir à présent l'opérateur Λ :

$$\Lambda : (e_0, e_1) \mapsto (-\partial_t y(0,x), y(0,x))$$

On retrouve par le calcul les composantes de Λe en résolvant le problème

$$\min_{(e_0,e_1)} J_1$$

Pour cela, on calcule la différentielle de J_1 :

$$\lim_{\tau \to 0} \frac{J_1(e_0 + \tau\tilde{e}_0, e_1 + \tau\tilde{e}_1) - J_1(e_0, e_1)}{\tau}$$

On a par linéarité :

$$u(t,x) = u_0(t,x) + \tilde{u}(t,x)$$

D'où :

$$\begin{aligned} J_1(e_0 + \tau\tilde{e}_0, e_1 + \tau\tilde{e}_1) =& \frac{1}{2}\int_0^T \eta(t)(\partial_x u_0(t,1))^2 dt + \tau\int_0^T \eta(t)\partial_x u_0(t,1)\partial_x\tilde{u}(t,1)dt + \frac{1}{2}\tau^2\int_0^T (\partial_x\tilde{u}(t,1))^2 dt \\ &+ \int_0^1 y_0(x)e_1 dx + \tau\int_0^1 y_0(x)\tilde{e}_1 dx - \int_0^1 y_1(x)e_0 dx - \tau\int_0^1 y_1(x)\tilde{e}_0 dx \end{aligned}$$

Soit :

$$\begin{aligned}
J_1(e_0+\tau\tilde{e}_0, e_1+\tau\tilde{e}_1) - J_1(e_0,e_1) &= \frac{1}{2}\int_0^T \eta(t)(\partial_x u_0(t,1))^2 dt + \tau\int_0^T \eta(t)\partial_x u_0(t,1)\partial_x\tilde{u}(t,1)dt \\
&+ \frac{1}{2}\tau^2\int_0^T (\partial_x\tilde{u}(t,1))^2 dt + \int_0^1 y_0(x)e_1 dx + \tau\int_0^1 y_0(x)\tilde{e}_1 dx - \int_0^1 y_1(x)e_0 dx \\
&- \tau\int_0^1 y_1(x)\tilde{e}_0 dx - \frac{1}{2}\int_0^T \eta(t)(\partial_x u_0(t,1))^2 dt - \int_0^1 y_0(x)e_1 dx + \int_0^1 y_1(x)e_0 dx \\
&= \tau\int_0^T \eta(t)\partial_x u_0(t,1)\partial_x\tilde{u}(t,1)dt + \frac{1}{2}\tau^2\int_0^T (\partial_x\tilde{u}(t,1))^2 dt + \int_0^1 y_0(x)\tilde{e}_1 dx \\
&- \int_0^1 y_1(x)\tilde{e}_0 dx
\end{aligned}$$

Donc :

$$\begin{aligned}
&\lim_{\tau\to 0}\frac{J_1(e_0+\tau\tilde{e}_0, e_1+\tau\tilde{e}_1) - J_1(e_0,e_1)}{\tau} = 0 \\
\iff \quad &\int_0^T \eta(t)\partial_x u_0(t,1)\partial_x\tilde{u}(t,1)dt + \int_0^1 y_0(x)\tilde{e}_1 dx - \int_0^1 y_1(x)\tilde{e}_0 dx = 0 \\
\iff \quad &\langle\Lambda(e_0,e_1),(\tilde{e}_0,\tilde{e}_1)\rangle = \int_0^1 y_0(x)\tilde{e}_1 dx - \int_0^1 y_1(x)\tilde{e}_0 dx \\
\iff \quad &\int_0^1 (\Lambda e)_1\tilde{e}_0 dx - \int_0^1 (\Lambda e)_2\tilde{e}_1 dx = \int_0^1 y_0(x)\tilde{e}_1 dx - \int_0^1 y_1(x)\tilde{e}_0 dx
\end{aligned}$$

En prenant $\tilde{e}_1 = 0$, puis $\tilde{e}_0 = 0$ et par des arguments de densité, on retrouve

$$\begin{cases} (\Lambda e)_1 = -y_1(x) \\ (\Lambda e)_2 = y_0(x) \end{cases}$$

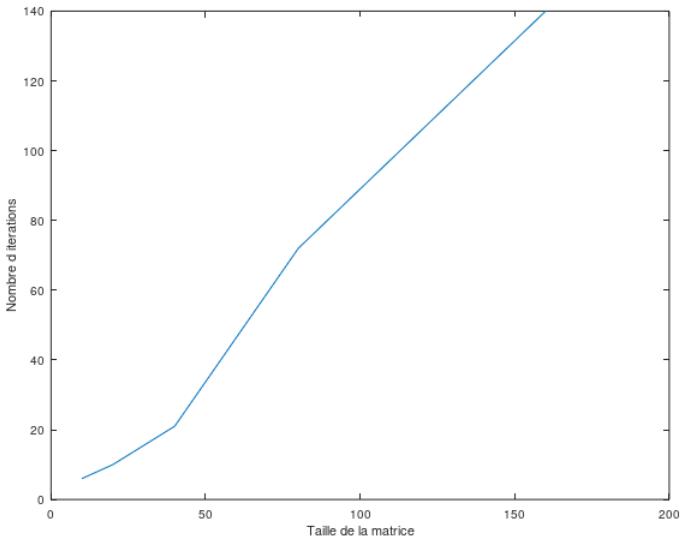
3 Résolution par gradient conjugué

3.1 Algorithme du gradient conjugué

Dans cette seconde partie, nous essayerons de résoudre l'équation linéaire (5) par une méthode de gradient conjugué. Tout d'abord, écrivons le code Octave du gradient conjugué pour une matrice A quelconque :

```
function [x,k] = GradientConjugue(A,b,x0,tolerance,maxiter)
  x = x0;
  r = b - A*x0;
  p = r;
  k = 1;
  while(k<maxiter && norm(r)>tolerance)
    a = r'*r/((p'*A)*p);
    x(:,k+1) = x(:,k) + a*p;
    r1 = r - a*A*p;
    beta = r1'*r1/(r'*r);
    r = r1;
    p = r + beta*p;
    k=k+1;
  endwhile
endfunction
```

On regarde maintenant comment évolue le nombre d'itérations en fonction de la taille de la matrice afin de vérifier la convergence de notre algotrithme.



Le graphique met bien en évidence le théorème de convergence du gradient conjugué, qui dit que ce dernier converge en au plus n itérations, où n est la taille de la matrice. En effet, on observe que le rapport $\frac{\text{nombre d'itérations}}{\text{taille de la matrice}}$ est toujours plus petit que 1.

On précise qu'on a pris ici comme matrice A le laplacien dont le conditionnement varie à chaque fois que la taille de A augmente. Pour optimiser notre algorithme, on peut préconditionner A avant de résoudre $Ax = b$ et on résolvera plutôt $MAx = Mb$, où MA est notre matrice bien conditionnée.

3.2 Résolution du problème $\Lambda e^* = f$

On cherche maintenant à résoudre $\Lambda e^* = f$ par une méthode de gradient conjugué. On doit donc s'assurer que Λ est symétrique. En fait, on montre la formule suivante :

$$(\Lambda e, \tilde{e}) = \int_0^1 (\Lambda e)_0 \tilde{e}_0 dx + \int_0^1 (\Lambda e)_1 \tilde{e}_1 dx = \int_0^T \partial_x u(t,1) \partial_x \tilde{u}(t,1) dt \tag{8}$$

Par définition de Λ :

$$(\Lambda e, \tilde{e}) = -\int \partial_t y(0,x) \tilde{e}_0(x) dx + \int y(0,x) \tilde{e}_1(x) dx.$$

On reconnaît des termes de bords liés à une intégration par parties en temps.
On a :

$$\int_0^T \int_0^1 (\partial_{tt} y) \tilde{u}(t,x) dx dt = -\int_0^T \int_0^1 \partial_t y \partial_t \tilde{u}(t,x) dx dt + \left[\int_0^1 (\partial_t y) \tilde{u} dx \right]_0^T.$$

Or, en $t = T$, $\partial_t y(T,x) = 0$, d'où :

$$\int_0^T \int_0^1 (\partial_{tt} y) \tilde{u}(t,x) dx dt = -\int_0^T \int_0^1 \partial_t y \partial_t \tilde{u}(t,x) dx dt - \int_0^1 \partial_t y(0,x) \underbrace{\tilde{u}(0,x)}_{\tilde{e}_0}.$$

On refait une intégration par parties :

$$\int_0^T \int_0^1 (\partial_{tt} y) \tilde{u}(t,x) dx dt = \int_0^T \int_0^1 y(t,x) \partial_{tt} \tilde{u}(t,x) dx dt - \left[\int_0^1 y \partial_t \tilde{u} dx \right]_0^T - \int_0^1 \partial_t y(0,x) \tilde{e}_0(x) dx.$$

Or, en $t = T$, $y(T,x) = 0$, d'où :

$$\begin{aligned} \int_0^T \int_0^1 (\partial_{tt} y) \tilde{u}(t,x) dx dt &= \int_0^T \int_0^1 y(t,x) \partial_{tt} \tilde{u}(t,x) dx dt + \int_0^1 y(0,x) \underbrace{\partial_t \tilde{u}(0,x)}_{e_1} dx - \int_0^1 \partial_t y(0,x) \tilde{e}_0(x) dx. \\ &= \int_0^T \int_0^1 y(t,x) \partial_{tt} \tilde{u}(t,x) dx dt + (\Lambda e, \tilde{e}). \end{aligned}$$

Or y et $\tilde{u}$ sont solutions de (7), soit :

$$\int_0^T \int_0^1 (\partial_{xx} y) \tilde{u}(t,x) dx dt = \int_0^T \int_0^1 y(t,x) \partial_{xx} \tilde{u}(t,x) dx dt + (\Lambda e, \tilde{e}).$$

On fait maintenant une intégration par parties en espace :

$$-\int_0^T \int_0^1 (\partial_x y) \partial_x \tilde{u}(t,x) dt dx + \int_0^T \underbrace{\left[(\partial_x y) \tilde{u} \right]_0^1}_{=0} dt = -\int_0^T \int_0^1 (\partial_x y) \partial_x \tilde{u}(t,x) dt dx + \int_0^T \left[(\partial_x \tilde{u}) y \right]_0^1 dt + (\Lambda e, \tilde{e})$$

$$\iff (\Lambda e, \tilde{e}) = -\int_0^T \left[(\partial_x \tilde{u}) y \right]_0^1 dt$$

Or, $y(t, x = 0) = 0$ et $y(t, x = 1) = -\int_0^T \eta(t)\partial_x\tilde{u}(t, 1)\partial_x u(t, 1)dt$, d'où (8) :

$$(\Lambda e, \tilde{e}) = \int_0^T \eta(t)\partial_x\tilde{u}(t, 1)\partial_x u(t, 1)dt.$$

Et on a bien $(\Lambda e, \tilde{e}) = (\Lambda\tilde{e}, e)$.

Après avoir implémenté sous Octave l'opérateur Λ comme suit :

```
function [L1,L2] = Lambda(y,EDP,Nt)
  L2 = y(1,:);
  dt = (EDP.T - EDP.t0)/Nt;
  L1 = (y(2,:) - y(1,:))/dt;
endfunction
```

Autrement, on réécrit notre fonction *Lambda* en y intégrant le y qu'on trouve en résolvant (7) :

```
function [L1,L2] = Lambda2(EDP,a,Nx)
  [t,x,u] = EquationOnde(EDP,a,Nx);
  [t,x,y] = EquationOnde2(EDP,a,Nx,u);
  L2 = y(1,:);
  dt = (EDP.T - EDP.t0)/Nt;
  L1 = (y(2,:) - y(1,:))/dt;
endfunction
```

On écrit le gradient conjugué sous Octave en y intégrant Λ :

```
function [e0,e1] = GradientConjugueLambda(EDP,f0,f1,maxiter,tolerance,a,Nx)
  x = zeros(Nx+1,1);
  delta_x = (EDP.b - EDP.a)/Nx;
  x = EDP.a:delta_x:EDP.b;
  e0 = zeros(size(x))'; e1 = zeros(size(x))'; %donnée initiale
  EDP.e0 = e0; EDP.e1 = e1;
  [y0,y1] = Lambda_modVec(EDP,a,Nx,[EDP.e0,EDP.e1]);%on calcule Lambda appliqué à la donnée initiale
  [f_mod0,f_mod1] = f_mod(EDP,zeros(size(x))',sin(pi*x)',Nx); %second membre
  R = [f_mod0 - y0,f_mod1 - y1];
  P = R;
  k = 1;
  while(k<maxiter && sqrt(Norme(R))>tolerance)
    EDP.e0 = e0; EDP.e1 = e1;
    [T1,T2] = Lambda_modVec(EDP,a,Nx,P);
    T = [T1,T2];
    alpha = Norme(R)/ProduitScalaire(T,P);
    sqrt(Norme(R))
    e0 = e0 + alpha*P(:,1);
    e1 = e1 + alpha*P(:,2);
    R1 = R - alpha*T;
    N1 = Norme(R1);
    beta = Norme(R1)/Norme(R);
    R = R1;
    P = R + beta*P;
    k=k+1;
  endwhile
endfunction
```

On utilise ici la norme $H_0^1 * L^2$ et le produit scalaire correspondant, qui sont donnés par :

$$||(e_0, e_1)||_{(H_0^1 * L^2)} = \int_0^1 (e_0^{'})^2(x)dx + \int_0^1 (e_1)^2(x)dx$$

$$< A, B > = \int_0^1 (A_0^{'} * B_0^{'})(x)dx + \int_0^1 (A_1 * B_1)^2(x)dx$$

on obtient la fonction norme sur Octave :

```
function S = Norme(a)
  N = length(a);
  h = (1-0)/N;
  S1 = 0;
  S2 = 0;
  for i=1:N-1
    S1 = S1 + h*((a(i+1,1)-a(i,1))/h)^2;
    S2 = S2 + h*(a(i,2)^2);
  endfor
  S2 = S2 + h*(a(N,2)^2);
  S = S1 + S2;
endfunction
```

ainsi que la fonction produit scalaire :

```
function S = ProduitScalaire(T,P)
  N = length(T);
  h = 1/N;
  S1 = 0;
  S2 = 0;
  for i=1:N-1
    S1 = S1 + (T(i+1,1)-T(i,1))*(P(i+1,1)-P(i,1))/h;
    S2 = S2 + h*T(i,2)*P(i,2);
  endfor
  S2 = S2 + h*T(N,2)*P(N,2);
  S = S1 + S2;
endfunction
```

3.3 Résolution exacte dans le cas $T = 4$

On a :

$$J_1(e_0, e_1) = \frac{1}{2}\int_0^T \eta(t)(\partial_x u(t,1))^2 dt + \int_0^1 y_0(x)\partial_t u(0,x)dx - \int_0^1 y_1(x)u(0,x)dx$$

On peut écrire y_0 et y_1 sous forme de séries de Fourier :

$$y_0 = \sum_{k=1}^{+\infty} \widehat{y}_k^0 sin(k\pi x) \qquad \text{et} \qquad y_1 = \sum_{k=1}^{+\infty} \widehat{y}_k^1 sin(k\pi x)$$

et on a

$$\sum_{k=1}^{+\infty} (k\pi)^2 |\widehat{y}_k|^0 + |\widehat{y}_k|^1 < \infty.$$

On a de plus :

$$e_0 = \sum_{k\geq 1} a_k sin(k\pi x) \qquad \text{et} \qquad e_1 = \sum_{k\geq 1} b_k sin(k\pi x).$$

u s'écrit alors :

$$u(t,x) = \sum_{k\geq 1} (a_k cos(k\pi t) + \frac{b_k}{k\pi} sin(k\pi t)) sin(k\pi x),$$

d'où :

$$\partial_x u(t,x) = -\sum_{k\geq 1} (k\pi)(a_k cos(k\pi t) + \frac{b_k}{k\pi}, sin(k\pi t)) cos(k\pi x),$$

soit :

$$\partial_x u(t,1) = \sum_{k\geq 1} (-1)^k (k\pi)(a_k cos(k\pi t) + \frac{b_k}{k\pi} sin(k\pi t)).$$

On peut maintenant calculer $\int_0^T \eta(t)(\partial_x u(t,1))^2 dt$ en prenant $\eta \equiv 1$:

$$\begin{aligned}
\int_0^T (\partial_x u(t,1))^2 dt &= \int_0^T \Big(\sum_{k\geq 1} (-1)^k (k\pi)(a_k cos(k\pi t) + \frac{b_k}{k\pi} sin(k\pi t)) \Big)^2 dt \\
&= \int_0^T \Big(\sum_{k\geq 1} \Big((-1)^k (k\pi)(a_k cos(k\pi t) + \frac{b_k}{k\pi} sin(k\pi t)) \Big)^2 + 2\sum_{k\geq 1} ... \Big) dt \\
&= \sum_{k\geq 1} \int_0^T \Big((-1)^k (k\pi)(a_k cos(k\pi t) + \frac{b_k}{k\pi} sin(k\pi t)) \Big)^2 dt + 2\sum_{k\geq 1} \int_0^T ... dt
\end{aligned}$$

En fait, on peut montrer que le second membre de la somme vaut 0.
Il reste donc :

$$\begin{aligned}
\int_0^T (\partial_x u(t,1))^2 dt &= \sum_{k\geq 1} \int_0^T \Big((-1)^k (k\pi)(a_k cos(k\pi t) + \frac{b_k}{k\pi} sin(k\pi t))\Big)^2 dt \\
&= \sum_{k\geq 1} \int_0^T (k\pi a_k cos(k\pi t) + b_k sin(k\pi t))^2 dt \\
&= \sum_{k\geq 1} \Big(k^2\pi^2 a_k^2 \int_0^T cos^2(k\pi t)dt + 2k\pi a_k b_k \int_0^T cos(k\pi t)sin(k\pi t)dt + b_k^2 \int_0^T sin^2(k\pi t)dt\Big) \\
&= \sum_{k\geq 1} \Big(k^2\pi^2 a_k^2 \int_0^T \frac{1+cos(2k\pi t)}{2} dt + 2k\pi a_k b_k \int_0^T \frac{sin(2k\pi t)}{2} dt + b_k^2 \int_0^T (1 - cos^2(k\pi t))dt\Big) \\
&= \sum_{k\geq 1} \Big(k^2\pi^2 a_k^2 \Big(\frac{T}{2} + \frac{sin(2k\pi T)}{4k\pi}\Big) + 2k\pi a_k b_k \Big(-\frac{cos(2k\pi T) - 1}{4k\pi}\Big) + b_k^2 \Big(\frac{T}{2} - \frac{sin(2k\pi T)}{4k\pi}\Big)\Big)
\end{aligned}$$

Or ici $T = 4$, d'où :

$$\begin{aligned}
\int_0^T (\partial_x u(t,1))^2 dt &= \sum_{k\geq 1} \Big(k^2\pi^2 a_k^2 (2+0) + 2k\pi a_k b_k \Big(-\frac{1-1}{4k\pi}\Big) + b_k^2 (2-0)\Big) \\
&= 2(k^2\pi^2 a_k^2 + b_k^2).
\end{aligned}$$

Donc

$$J_1(a_k, b_k) = \sum_{k\geq 1} (k^2\pi^2 a_k^2 + b_k^2) + \frac{1}{2}(\widehat{y}_k^0 b_k - \widehat{y}_k^1 a_k).$$

En dérivant par rapport à a_k et b_k, on obtient :

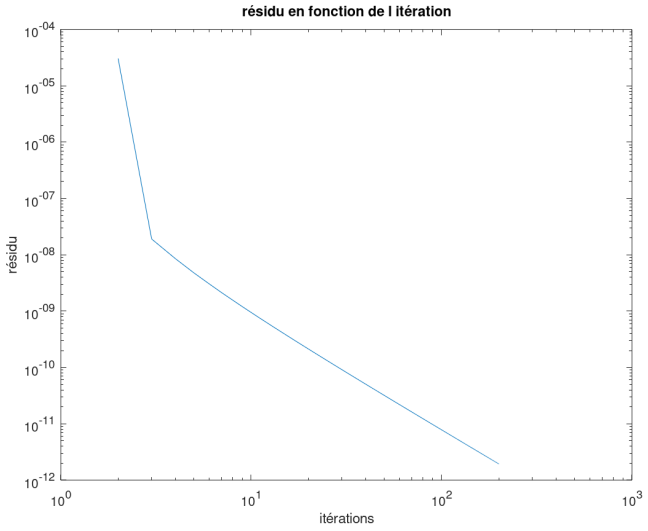
$$\nabla J_1 = 0 \iff \begin{cases} \sum_{k\geq 1} 2k^2\pi^2 a_k - \frac{1}{2}\widehat{y}_k^1 = 0 \\ \sum_{k\geq 1} 2b_k + \frac{1}{2}\widehat{y}_k^0 = 0 \end{cases}$$

d'où :

$$a_k = \frac{\widehat{y}_k^1}{4k^2\pi^2} \qquad \text{et} \qquad b_k = \frac{-\widehat{y}_k^0}{4}.$$

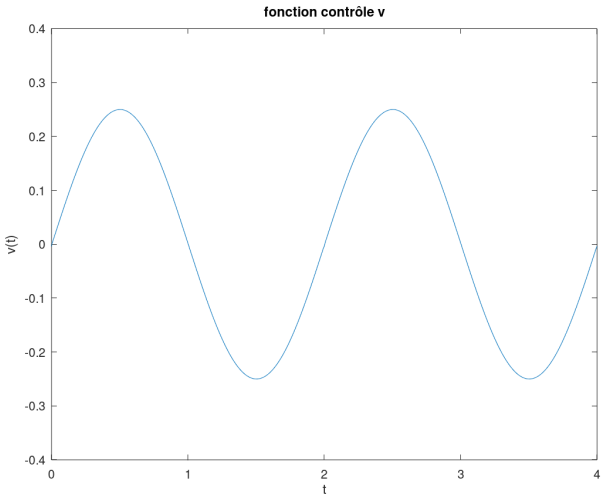
4 Conclusion

Dans ce projet, nous avons étudié la contrôlabilité à zéros de l'équation des ondes unidimensionnelle. Ainsi, à travers nos codes, nous avons pu résoudre le problème associé $\Lambda e^* = f$ en utilisant le gradient conjugué. Il serait donc intéressant de voir l'évolution du résidu en fonction du nombre d'itérations du code :

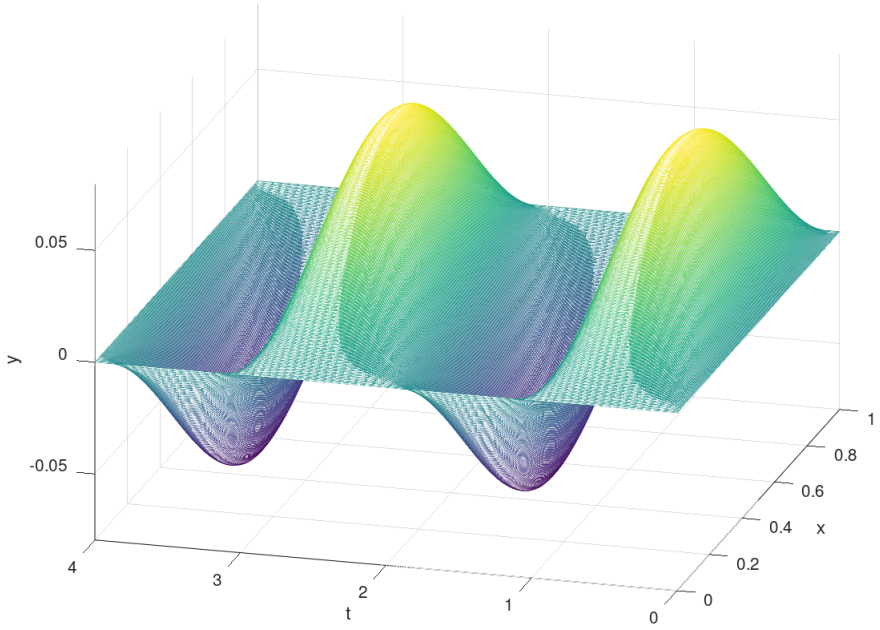


On remarque que le résidu diminue très rapidement au début puis lentement à mesure qu'on s'approche de la solution. En mettant une tolérance trop basse, l'algorithme ne s'arrête qu'au nombre d'itérations maximum permis par la condition de la boucle. Cette lenteur de convergence est due au mauvais conditionnement du problème, lui-même dû à la discrétisation de la fonction peu régulière $x \mapsto -\partial_t y(0, x)$.

En résolvant l'équation d'onde, avec comme conditions initiales les fonctions solutions du gradient conjugué, on peut construire la fonction contrôle frontière. On trace ainsi la fonction $v(t)$ obtenu pour notre exemple :



Enfin, on peut vérifier que notre fonction v est bien une fonction de contrôle de frontière en l'intégrant dans l'équation des ondes. En traçant la solution de cette équation, on retrouve bien qu'au temps $T = 4$, la solution est nulle :



5 Remerciements

Nous tenons à remercier Mme Delourme et M. Audusse pour leur encadrement de ce projet, pour le temps qu'ils nous ont accordé et toute l'aide qu'ils nous ont fournie. Les nombreuses réunions nous ont permis d'avoir un travail continu et efficace grâce aux corrections qu'ils ont apportées, toujours dans la bienveillance et la patience. Ce projet était très enrichissant pour nous et nous a permis d'appliquer la théorie que nous apprenons depuis le début de la MACS à un problème concret d'EDP, notamment dans la modélisation numérique.

"Pure mathematics is, in its way, the poetry of logical ideas" A. Einstein