Time reversal of wave propagation in one dimension

CANET Mathieu
JARRIER Antoine
MYTYCH François-Joseph

under the direction of
ROUSSELET Bernard and JUNCA Stéphane.

January 30, 2008

Acknowledgements

We would like to take this opportunity to thank Bernard ROUSSELET and Stéphane JUNCA who have supervised us to this report. We hope that we have been up to their expectation.

Contents

I Introduction 3
II The spring equation 4
1 Explicit solution of the equation 4
2 Numerical solution of the equation 5
3 Time reversal of the equation with opposite speed 9
4 Resolution of the equation : $\ddot{u} + c^2 u = u_1 \delta_0$ 10
5 Spring with strength measure 11
6 System of springs 14
III A link between the two parts of the project 19
IV In 1-dimension : the wave equation 20
1 Numerical resolution of the wave equation 20
2 Time reversal with opposite speed 24
3 Time reversal with the same speed and boundary conditions 30
V Conclusion 34
Bibliographie 35

Annexe **36**

I Introduction

In this project, we will deal with the wave equation and its time reversal. We can do a time reversal only on the reversible equations.

In the first part, we will begin with an easier example : a spring system. that allow us to understand the time reversal effects in a real situation. Then, we will complicate this system by adding others springs in the equation, that will allow us to do the link with the wave equation.

In the second part, we will solve numerically the wave equation. Then, we will do a time reversal, in a first time with good initial conditions (right initial speed) and in a second time with bad initial conditions (wrong initial speed). Finally, we will show the limits of our numerical resolution for the time reversal.

The purpose of our project is to find the initial position (source) or speed using the "revolutionary principle" of time reversal introduced by Mathias FINK. We will treat this problem in 0-dimension (spring equation) and 1-dimension (the vibrating string equation) only.

II The spring equation

1 Explicit solution of the equation

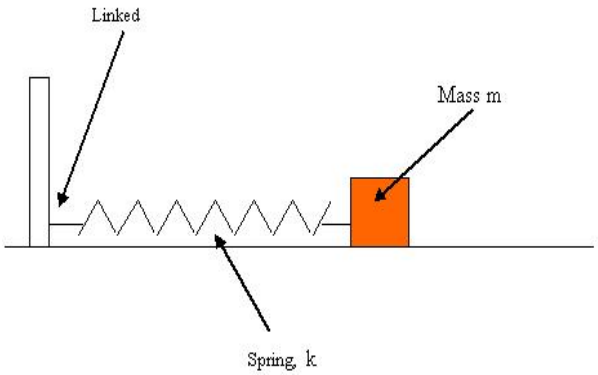

Figure 1: spring system

$$\Sigma F = m\gamma \Leftrightarrow -ku = m\ddot{u}$$

$$\begin{cases} \ddot{u} + c^2 u = 0, \quad \text{with } c = \sqrt{\frac{k}{m}} \\ u(0) = u_0 \\ \dot{u}(0) = u_1 \end{cases}$$

We know that the solution u is of the following form :

$$u(t) = A\cos(ct) + B\sin(ct)\,.$$

Now, using the initial data,

$$\begin{cases} u(0) = A\cos(0) + B\sin(0) \;=\; A \;=\; u_0 \\ \dot{u}(0) = -cA\sin(0) + cB\cos(0) \;=\; cB \;=\; u_1 \end{cases}$$

Therefore,

$$u(t) = u_0\cos(ct) + \frac{u_1}{c}\sin(ct) \tag{1}$$

2 Numerical solution of the equation

2.1 The scheme

We have chosen the following scheme :

$$\frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} + c^2 u^n = 0\,,$$

with,

$$\begin{cases} u^0 = u_0 - \frac{\Delta t}{2} u_1 \text{ for } t = -1/2 \\ u^1 = u_0 + \frac{\Delta t}{2} u_1 \text{ for } t = 1/2 \end{cases}$$

Thus we obtain,

$$u^{n+1} = u^n(2 - (c\Delta t)^2) - u^{n-1}\,, \quad \text{for} \quad t = (n + 1/2)\Delta t\,.$$

2.2 The stability conditions

Now, look at the stability conditions :

$$u^{n+1} = u^n(2 - c^2\Delta t^2) - u^{n-1}$$

Therefore :

$$\begin{pmatrix} u^{n+1} \\ u^n \end{pmatrix} = \begin{pmatrix} 2 - c^2\Delta t^2 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u^n \\ u^{n-1} \end{pmatrix}$$

We calculate the spectral radius of this matrix whose eigenvalues are given by the equation :

$$\lambda - \lambda(2 - c^2\Delta t^2) + 1 = 0$$

So,

$$\Delta = c^2\Delta t^2(c^2\Delta t^2 - 4)$$

If $\Delta > 0 \Leftrightarrow c^2\Delta t^2 > 4$ then

$$\lambda_{1,2} = \frac{2 - c^2\Delta t^2 \pm |c\Delta t|\sqrt{c^2\Delta t^2 - 4}}{2}$$

$$\lambda_{max} = 1 - \frac{1}{2}(c^2\Delta t^2 + |c\Delta t|\sqrt{c^2\Delta t^2 - 4})$$

stable if and only if $0 < c^2\Delta t^2 + |c\Delta t|\sqrt{c^2\Delta t^2 - 4} < 4$, but $c^2\Delta t^2 > 4$ and $|c\Delta t|\sqrt{c^2\Delta t^2 - 4} > 0$ then instability.

If $\Delta < 0 \Leftrightarrow c^2\Delta t^2 < 4$ then,

$$\lambda_{1,2} = \frac{2 - c^2\Delta t^2 \pm i\sqrt{-c^2\Delta t^2 + 4}|c\Delta t|}{2}$$

$$|\lambda|^2 = \frac{4 - 4c^2\Delta t^2 + (c^2\Delta t^2)^2}{4} + \frac{c^2\Delta t^2(4 - c^2\Delta t^2)}{4} = \frac{4}{4} = 1$$

We conclude that this scheme is stable on condition that $\boxed{|c\Delta t| \leq 2}$.

2.3 The order

What is the order of this scheme?

Let $u(t_n)$ be a solution of the differential equation. Using the taylor expansion of $u(t_n + \Delta t)$ and $u(t_n - \Delta t)$, we have :

$$\begin{aligned}\tfrac{1}{\Delta t^2}u(t^n)(2-2) + \tfrac{1}{\Delta t^2}\tfrac{\partial u(t^n)}{\partial t}(\Delta t - \Delta t) + \tfrac{1}{\Delta t^2}\tfrac{\partial^2 u(t^n)}{\partial t^2}(\tfrac{\Delta t^2}{2} + \tfrac{\Delta t^2}{2}) + \tfrac{1}{\Delta t^2}\tfrac{\partial^3 u(t^n)}{\partial t^3}(\tfrac{\Delta t^3}{6} - \tfrac{\Delta t^3}{6}) \\ + \tfrac{1}{\Delta t^2}\tfrac{\partial^4 u(t^n)}{\partial t^4}(\tfrac{\Delta t^4}{24} + \tfrac{\Delta t^4}{24}) + \tfrac{\sigma(\Delta t^4)}{\Delta t^2} + c^2 u = \tfrac{\partial^2 u}{\partial t^2} + c^2 u + \sigma(\Delta t^2)\end{aligned}$$

where u is a solution of differential equation. We conclude that the $\boxed{\text{order of this scheme is 2}}$.

We have to start the scheme with an equation keeping the same order,

$$\begin{cases} u^0 = u_0 - \frac{\Delta t}{2}u_1 \\ u^1 = u_0 + \frac{\Delta t}{2}u_1 \end{cases}$$

Now,

$$\begin{aligned} u(\tfrac{-1}{2}) &= u(0) - \tfrac{\Delta t}{2}\dot{u}(0) + \sigma(\Delta t^2) = u^0 + \sigma(\Delta t^2) \\ u(\tfrac{1}{2}) &= u(0) + \tfrac{\Delta t}{2}\dot{u}(0) + \sigma(\Delta t^2) = u^1 + \sigma(\Delta t^2) \end{aligned}$$

Then, $\boxed{\text{the order of the starter is 2}}$.

2.4 The results

For $u_0 = u_1 = c = 1$ and a number of time steps equals to 100, on an interval $[0, 7]$, we have $c\Delta t = 0.07 < 2$. So, we obtain the following curves (value of the final error : 0.003575),

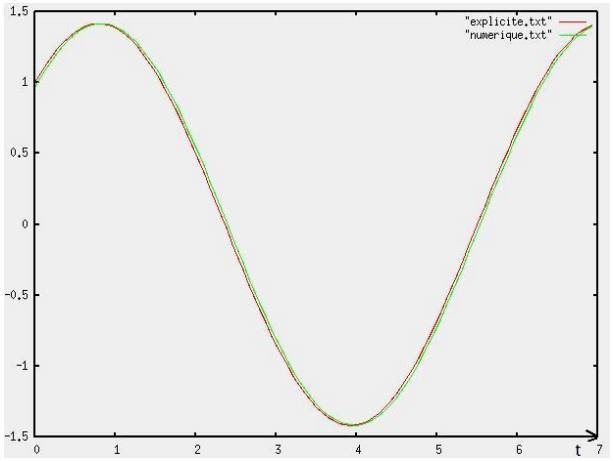

Figure 2: $u(t)$ and u^n

For $u_0 = u_1 = c = 1$ and a number of time steps equals to 200, on an interval $[0, 7]$, we have $c\Delta t = 0.035 < 2$. So, we obtain the following curves (value of the final error : 0.001755),

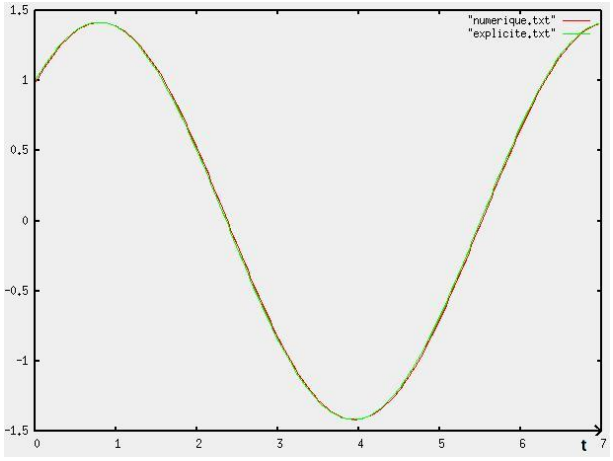

Figure 3: $u(t)$ and u^n

For $u_0 = 0$, $u_1 = c = 1$ and a number of time steps equals to 100, on an interval $[0, 7]$, we have $c\Delta t = 0.07 < 2$. So, we obtain the following curves (value of the final error : 0.025679),

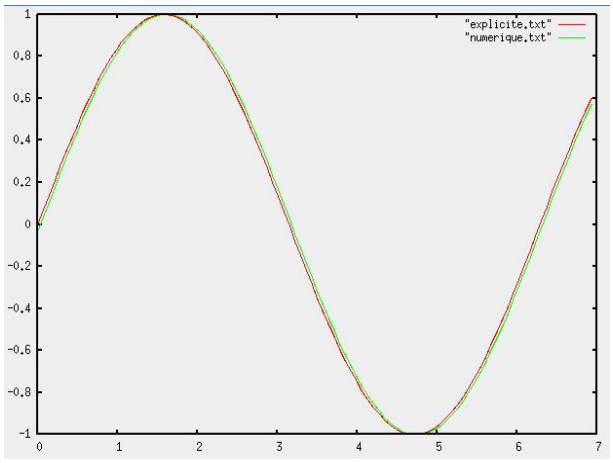

Figure 4: $u(t)$ and u^n

For $u_0 = 0$, $u_1 = c = 1$ and a number of time steps equals to 200, on an interval $[0, 7]$, we have $c\Delta t = 0.035 < 2$. So, we obtain the following curves (value of the final error : 0.013023),

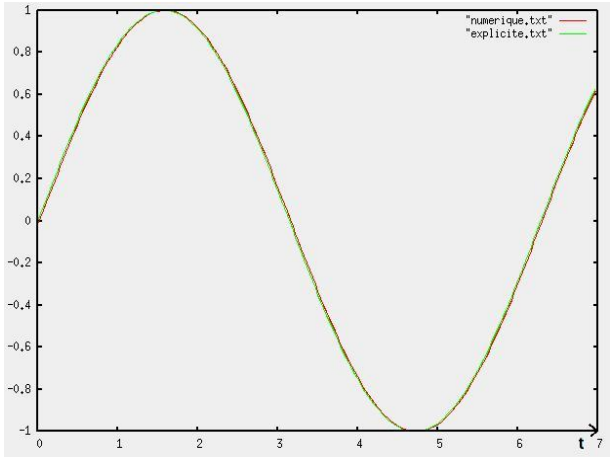

Figure 5: $u(t)$ and u^n

3 Time reversal of the equation with opposite speed

After the resolution of the differential equation on $[0, T]$. We use the following system,

$$\begin{cases} \ddot{v}(t) + c^2 v(t) = 0 \\ v(0) = u(T) = u_0 \cos(cT) + \frac{u_1}{c} \sin(cT) = v_0 \\ \dot{v}(0) = -\dot{u}(T) = c u_0 \sin(cT) - u_1 \cos(cT) = v_1 \end{cases}$$

However, for practical purposes, it is impossible to stop a spring system and to reverse speed. But it is interesting to see that the solution of this system gives us all the past of the spring. We know that the solution of this system is formed of :

$$v(t) = A \cos(ct) + B \sin(ct)\,,$$

We find $A = v_0 = u(T)$ and $B = \frac{v_1}{c} = \frac{-\dot{u}(T)}{c}$.
Therefore, we obtain the explicit solution,

$$v(t) = u(T) \cos(ct) + \frac{-\dot{u}(T)}{c} \sin(ct)\,.$$

We have $v(T) = u(0)$, after computation.
For $u_0 = 2$, $u_1 = 1$, $c = 4$, on an interval $[0, 8]$, we obtain the following curves,

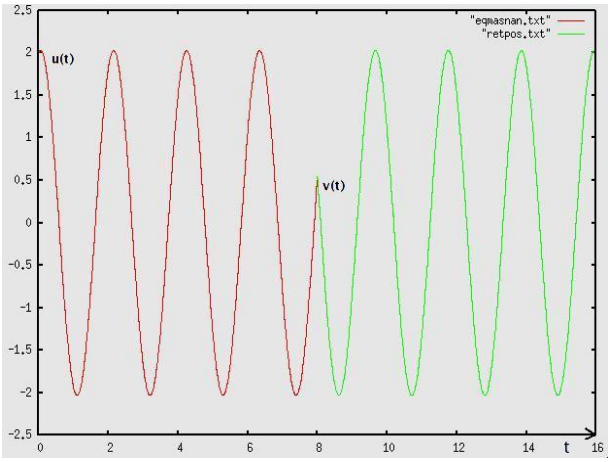

Figure 6: $u(t)$ et $v(t)$

4 Resolution of the equation : $\ddot{u} + c^2 u = u_1 \delta_0$

First of all, we fix $\forall t < 0$, $u(t) = 0$. So we can write u as : $u(t) = U(t)H(t)$. With $H(t) = 0$, $\forall t < 0$ and $H(t) = 1$, $\forall t > 0$. We have that $\dot{H} = \delta_0$ by definition.

$$\begin{cases} u{=}UH \\ \dot{u}{=}\dot{U}H + \dot{H}U \ = \ \dot{U}H + U(0)\delta_0 \\ \ddot{u}{=}\ddot{U}H + \dot{U}(0)\delta_0 + U(0)\delta_0' \end{cases}$$

Whence,

$$\ddot{u} + u = (U + \ddot{U})H + \dot{U}(0)\delta_0 + U(0)\delta_0' \, .$$

And by designation, we obtain the following system :

$$\begin{cases} \ddot{U} + U = 0 \\ U(0) = 0 \\ \dot{U}(0) = u_1 \end{cases}$$

That is the first solved system, with $c = 1$ and $u_0 = 0$.

5 Spring with strength measure

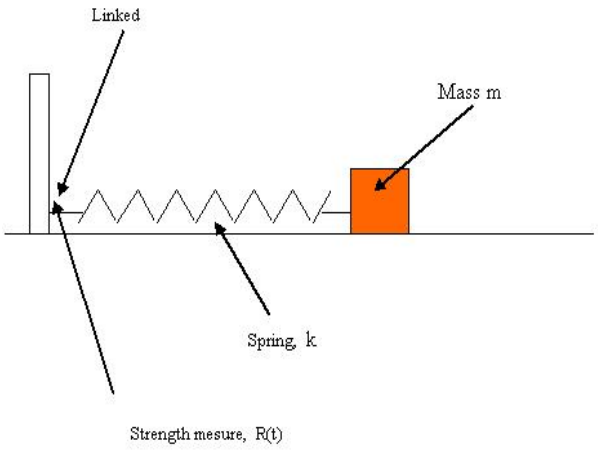

Figure 7: spring system

For this system, we only use an explicit resolution (also in time reversal) to solve the following equation.

$$m\ddot{u}_1 = -k(u_1 - u_0) + \delta_0$$

We take $u_0 = 0$, and the δ_0 in the second member is equivalent of these initial conditions:

$$\begin{cases} u_1(0)\boxed{= 0} \\ \dot{u}_1(0) = 1 \end{cases}$$

Resolution : the equation (1) gives us $u(t) = \frac{1}{c}\sin(ct)$ with $u_0 = 0$ and $\frac{u_1}{c} = \frac{1}{c}$.

Time reversal : thanks to the strength measures, we write :

$$\begin{cases} \ddot{v} + c^2 v = R(T-t), \text{ with } R(t) = ku(t) \\ v(0) = u(T)\boxed{= 0} \\ \dot{v}(0) = \dot{u}(T) \end{cases}$$

We underline that the condition $v(0) = u(T) = 0$ give us the condition on T. It is a condition to have a good result. Then, we have to choose cT a multiple of π.

Particular solution :

$$\begin{cases} v_p(t) = At\cos(ct) + Bt\sin(ct) \\ \dot{v}_p(t) = -cAt\sin(ct) + A\cos(ct) + B\sin(ct) + cBt\cos(ct) \\ \ddot{v}_p(t) = -c^2At\cos(ct) - 2cA\sin(ct) + 2cB\cos(ct) - c^2Bt\sin(ct) \end{cases}$$

We inject the solution into the equation to find the undertermined coefficients A and B :

$$\cos(ct)\left[-c^2At+2cB\right]-\sin(ct)\left[2cA+c^2Bt\right]+c^2At\cos(ct)+c^2Bt\sin(ct)=R(T-t)=\frac{k}{c}\sin(c(T-t))$$

$$\begin{aligned}\cos(ct)\,[2cB]-\sin(ct)\,[2cA]&=\tfrac{k}{c}\left(\sin(cT)\cos(ct)-\sin(ct)\cos(cT)\right)\\&=\cos(ct)\left[\tfrac{k}{c}\sin(cT)\right]-\sin(ct)\left[\tfrac{k}{c}\cos(cT)\right]\end{aligned}$$

We obtain :

$$A=\frac{k}{2c^2}\cos(cT)\quad\text{and}\quad B=\frac{k}{2c^2}\sin(cT)\tag{2}$$

Then : $v(t)=I\cos(ct)+J\sin(ct)+At\cos(ct)+Bt\sin(ct)$, with A and B given in (2). To find I and J, we use the initial conditions :

$$\dot v(t)=-cI\sin(ct)+cJ\cos(ct)-cAt\sin(ct)+A\cos(ct)+B\sin(ct)+cBt\cos(ct)$$

$$\begin{aligned}v(0)=u(T)&\Leftrightarrow\boxed{I=\tfrac{1}{c}\sin(cT)}\\\dot v(0)=\dot u(T)&\Leftrightarrow cJ+A=\cos(cT)\Rightarrow\boxed{J=\tfrac{\cos(cT)-A}{c}}\\v(t)&=I\cos(ct)+J\sin(ct)+At\sin(ct)+Bt\sin(ct)\end{aligned}$$

The condition $v(0)=0$ gives us that cT is a multiple of π.

To illustrate this resolution, we plot the direct and the reversal solution on $[0, T]$ interval,

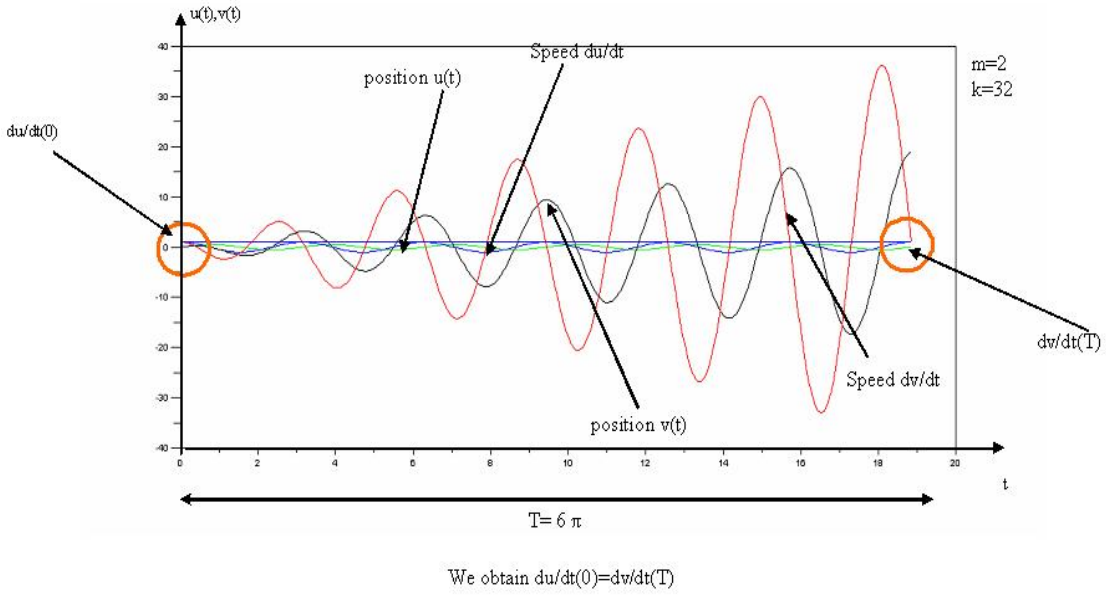

Figure 8: $u(t)$, $\dot{u}(t)$, $v(t)$ and $\dot{v}(t)$

6 System of springs

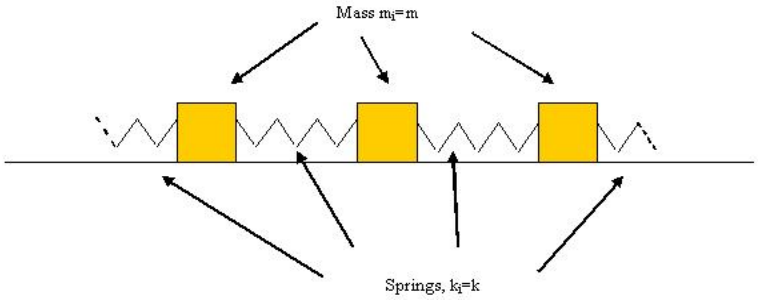

Figure 9: springs system

$$\Sigma F = m\gamma$$

To build the problem, we have a system of p-equations with the same spring between each mass :

$$\begin{cases} m_1\ddot{u}_1 = & -2ku_1 + ku_2 \\ m_3\ddot{u}_3 = & ku_1 - 2ku_2 + ku_3 \\ \vdots & = \quad \vdots \\ m_i\ddot{u}_i = & ku_{i-1} - 2ku_i + ku_{i+1} \\ \vdots & = \quad \vdots \\ m_p\ddot{u}_p = & -k(u_p - u_{p-1}) \end{cases}$$

We use matrices and vectors to sum up the system : $\boxed{M\ddot{U} = KU}$

$$\underline{\text{with :}}\ M = \begin{pmatrix} m_1 & & \\ & \ddots & \\ & & m_p \end{pmatrix}\ ;\ K = -k \begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{pmatrix}\ ;\ U^n = \begin{pmatrix} u_1^n \\ \vdots \\ u_p^n \end{pmatrix}$$

6.1 Numerical scheme for right direction

We use a scheme (order 2) to approach the second derivative of U :

$$\frac{U^{n+1} - 2U^n + U^{n-1}}{\Delta t^2}$$

(We have the demonstration of the stability and the order in II.1)
Therefore, $M\ddot{U} - KU = 0 \Leftrightarrow \ddot{U} - M^{-1}KU = 0$, (M is invertible because it is a diagonal matrix and there is no 0 in the diagonal). To implement the scheme, we use a matrix in which the columns at the different times and it is the line i which interests us to plot the position and the speed,

$$\begin{pmatrix} u_0^0 & u_0^1 & \cdots & u_0^n \\ \vdots & \vdots & & \vdots \\ u_i^0 & u_i^1 & \cdots & u_i^n \\ \vdots & \vdots & & \vdots \\ u_p^0 & u_p^1 & \cdots & u_p^n \end{pmatrix}$$

To start the scheme, we consider rest masses at $t = 0$. On a chosen mass, we applicate a local strength (dirac pulse). So we can use the following equations to start our scheme:

$$\begin{cases} U_i^0 = X_{0,i} - \frac{\Delta t}{2} X_{1,i} \\ U_i^1 = X_{0,i} + \frac{\Delta t}{2} X_{1,i} \end{cases}$$

where $X_{0,i}$ is the initial move so $X_{0,i} = 0$, $\forall i$ and $X_{1,i}$ is the initial speed $X_{1,i} = 0$, $\forall i \neq k$ and $X_{1,k} = u_1$.
We approach the speed $\dot{U}_i(t)$ by :

$$\frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t}$$

To determinate the stability conditions of the scheme, we need to use Scilab. We have,

$$\ddot{U} = M^{-1}KU$$

Therefore :

$$U^{n+1} = 2U^n - U^{n-1} + \Delta t^2 M^{-1} K U^n$$

Whence, $\begin{pmatrix} U^{n+1} \\ U^n \end{pmatrix} = \left(\begin{array}{c|c} \Delta t^2 M^{-1}K + 2I_p & \text{-I}_p \\ \hline \text{I}_p & 0 \end{array}\right) \begin{pmatrix} U^n \\ U^{n-1} \end{pmatrix}$

If the spectral radius of the following matrix is between -1 and 1, the scheme is stable

$$\left(\begin{array}{cccc|ccc} 2\text{-}2\alpha & \alpha & & 0 & \text{-}1 & & 0 \\ \alpha & \ddots & \ddots & & & \ddots & \\ & \ddots & 2\text{-}2\alpha & \alpha & & \ddots & \\ 0 & & \alpha & 2\text{-}\alpha & 0 & & \text{-}1 \\ \hline 1 & & & 0 & 0 & & \\ & \ddots & & & & \ddots & \\ & & \ddots & & & \ddots & \\ 0 & & & 1 & & & 0 \end{array}\right)$$

For example, with these values : we take 5 masses, all spring constants (k) equal to 32 and $\Delta t = 0.07$. So $a = -\frac{-k}{m}\Delta t^2 \simeq 0.0104$. With Scilab, we find,

```
a  =

   0.0104533
mat  =

   1.9790933    0.0104533    0.           0.           0.         - 1.    0.    0.    0.
   0.0104533    1.9790933    0.0104533    0.           0.           0.  - 1.    0.    0.
   0.           0.0104533    1.9790933    0.0104533    0.           0.    0.  - 1.    0.
   0.           0.           0.0104533    1.9790933    0.0104533    0.    0.    0.  - 1.
   0.           0.           0.           0.0104533    1.9895467    0.    0.    0.    0.
   1.           0.           0.           0.           0.           0.    0.    0.    0.
   0.           1.           0.           0.           0.           0.    0.    0.    0.
   0.           0.           1.           0.           0.           0.    0.    0.    0.
   0.           0.           0.           1.           0.           0.    0.    0.    0.
   0.           0.           0.           0.           1.           0.    0.    0.    0.
eigenvalues  =

   0.9807528 + 0.1952537i
   0.9807528 - 0.1952537i
   0.9852042 + 0.1713846i
   0.9852042 - 0.1713846i
   0.9910343 + 0.1336075i
   0.9910343 - 0.1336075i
   0.9963921 + 0.0848687i
   0.9963921 - 0.0848687i
   0.9995766 + 0.0290979i
   0.9995766 - 0.0290979i
norm_eigenvalues  =

   1.
   1.
   1.
   1.
   1.
   1.
   1.
   1.
   1.
   1.
```

Figure 10: Numerical applications

$$\rho(\text{matrix}) = 1 \quad \Rightarrow \quad \text{the scheme is stable.}$$

Using numerical implementation, we have the following curves for the third mass with $u_1 = 1$,

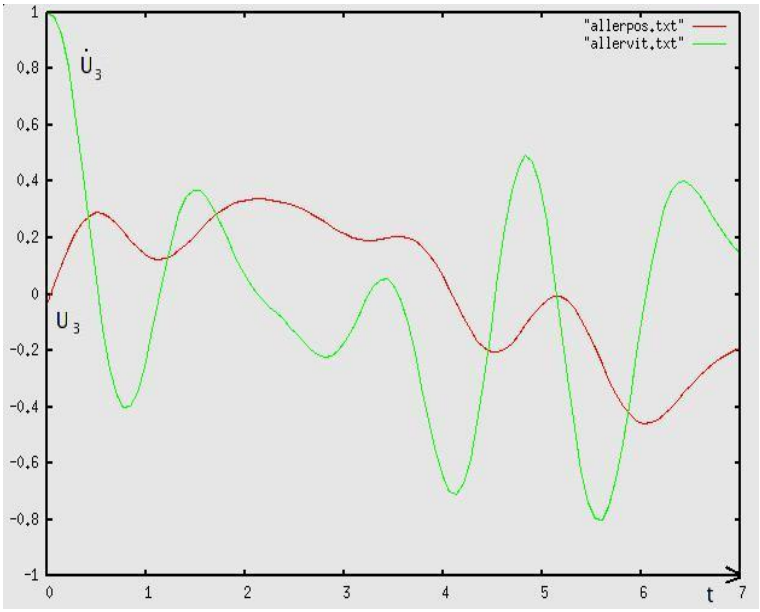

Figure 11: U_3 and $\dot{U}_3$

6.2 Time reversal with opposite speed

Like in section 3, we will do a time reversal with a opposite speed. At the time T, the numerical scheme is the same that in the right direction but the initial conditions change,

$$\begin{cases} V(0)=U(T) \\ \dot{V}(0)=\boxed{-\dot{U}(T)} \end{cases} \quad \text{we take an opposite speed}$$

Thus, to start the scheme, we use the following initial conditions,

$$\begin{cases} V_i^0=U_i^T-(-\dot{U}_i^T)\frac{\Delta t}{2} \\ \dot{V}_i^1=U_i^T+(-\dot{U}_i^T)\frac{\Delta t}{2} \end{cases}$$

Like before, we approach the speed by,

$$\dot{V}_i^n=\frac{V_i^{n+1}-V_i^{n-1}}{2\Delta t} \quad \text{and} \quad \dot{U}_i^n=\frac{U_i^{n+1}-U_i^{n-1}}{2\Delta t}$$

Now, we can solve numerically the time reversal.

Using the same values that in 6.1, we obtain the curves of the third mass. The right direction from $t = 0$ to T and time reversal from $t = T$ to $2T$.

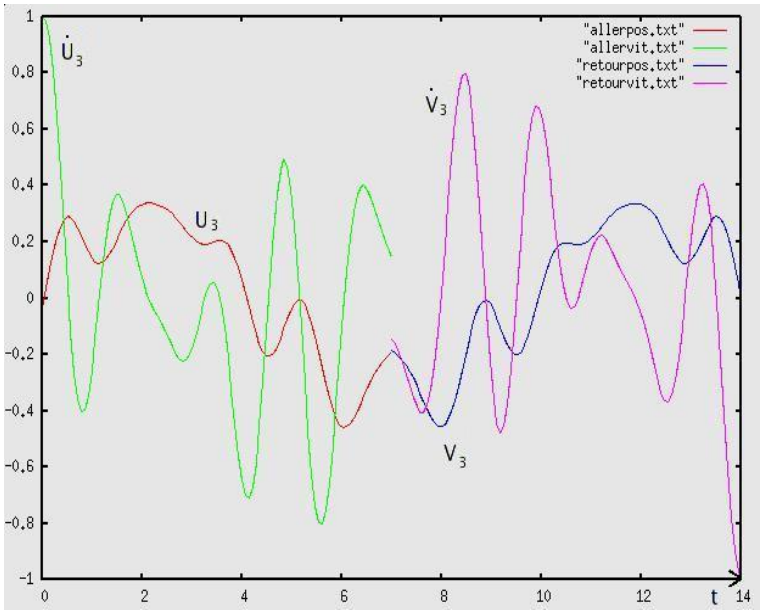

Figure 12: U_3, $\dot{U}_3$, V_3 and $\dot{V}_3$

(Although taking the opposite speed in mecanic experience is not possible, doing a time reversal like this allow us to verify that we have used a good scheme).

6.3 Tries for time reversal in physic conditions

We have tried to add a strength which correct the fact that we can't reverse the speed. We fix $R^i(t)$ the strength measured on each spring, and $R^0(t)$ the strength measured at the extremity, during the right direction. In the case where we inject $R^0(2T - t)$ at every time, or only once (at the beginnig), on the mass k or the mass 1, the results aren't good. We also tried to inject (only once and at evry time) the strength $R^i(2T - t)$ on each mass i during time reversal but the results aren't good either.In any case, we have taken the condition $V_k^0 = U_k^T = 0$. So, we haven't found how to obtain the initial speed after time reversal.

III A link between the two parts of the project

First of all, we have the 0-dimension :
It is a system of a p-springs and it can be written with the equation for the mass i:

$$m_i \ddot{u}_i = k u_{i-1} - k u_i + k u_{i+1} \, .$$

Then the scheme to approach $\ddot{u}$ is:

$$\frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} \tag{3}$$

In the second part, we will see that the equation of waves has a scheme of the same form like the 0-dimension with an additionnal dimension of space. That is to say,

$$\underbrace{\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2}}_{(4)} \quad - \quad \underbrace{c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}}_{(5)}$$

We have (3)=(4).
To conclude, we observe that passing from 0-dimension to 1-dimension is "intuitive" and the equation of springs can be viewed like a discretisation of the equation of waves.

IV In 1-dimension : the wave equation

1 Numerical resolution of the wave equation

The wave equation with an origin is defined like that :

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = \delta_{x=0} \times \delta_{t=0} & \forall (x,t) \in \Omega = [-E,E] \times [0,T] \\ u(-E,t) = g(t) & \forall t \in [0,T] \\ u(E,t) = d(t) & \forall t \in [0,T] \\ u(x,t)|_{t<0} = 0 & \forall x \in [-E,E] \end{cases}$$

This equation is reversible (i.e : if $u(x,t)$ is solution, then $u(x,-t)$ is too), but it is explicitly unsolvable, then we use a numerical scheme to find the solution.

1.1 The scheme

For this equation, we choose to use this scheme,

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} - c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = \frac{1}{2\pi\sigma^2} e^{\frac{-(i\Delta x^2 + n\Delta t^2)}{2\sigma^2}} \text{, with} 0 < \sigma << 1 \,.$$

The Gauss function in time and in space of the second member is used to approximate the ponctual source. We can change the position of this source (in this instance : $t = 0$ and $x = 0$).

1.2 The stability conditions

We check the stability of the scheme. Then, we express u_i^{n+1} function of u_i^n and u_i^{n-1} :

$$u_i^{n+1} = 2u_i^n - u_i^{n-1} + \left(\frac{c\Delta t}{\Delta x}\right)^2 (u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

Fixing $u^n(x) = u_i^n$, for $x \in [i\Delta x, (i+1)\Delta x[$. And using the Fourier transform of u^n, we have :

$$\hat{u}^{n+1} = \hat{u}^n \left(2 - 4\left(\frac{c\Delta t}{\Delta x}\right)^2 \sin^2\left(\frac{\xi \Delta x}{2}\right)\right) - \hat{u}^{n-1}$$

We fix $\beta = (\frac{c\Delta t}{\Delta x})^2 \sin^2(\frac{\xi \Delta x}{2})$ and we obtain the following amplification matrix :

$$\begin{pmatrix} \hat{u}^{n+1} \\ \hat{u}^n \end{pmatrix} = \begin{pmatrix} 2-4\beta & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u^n \\ u^{n-1} \end{pmatrix}$$

The eigenvalues of the amplification matrix are solutions of this equation :
$\lambda^2 - \lambda(2 - 4\beta) + 1 = 0$ where $\Delta = 16\beta(\beta - 1)$.
If $\Delta > 0 \Leftrightarrow \beta > 1$, then :

$$\lambda = \frac{2 - 4\beta \pm \sqrt{16\beta(\beta-1)}}{2}$$

as $2-4\beta<0$, thus

$$\lambda_{max}=1-\left(2\beta+\sqrt{4\beta(\beta-1)}\right)$$

For the scheme stability, we have to have : $0<2\beta+\sqrt{4\beta(\beta-1)}<2$.
But $2\beta>2$ and $\sqrt{4\beta(\beta-1)}>0$.
So, $2\beta+\sqrt{4\beta(\beta-1)}>2 \Rightarrow$ instability.
Let's see for $\Delta<0 \Leftrightarrow \beta<1$. We have :

$$\lambda=\frac{2-4\beta\pm i\sqrt{-16\beta(\beta-1)}}{2}$$

$$|\lambda|^2=\frac{(2-4\beta)^2}{4}+\frac{16\beta(1-\beta)}{4}$$

whence $|\lambda|=1$, thus this scheme is stable for $\beta<1$

$$\Leftrightarrow \left(\frac{c\Delta t}{\Delta x}\right)^2\sin^2\left(\frac{\xi\Delta x}{2}\right)<1$$

$\Leftrightarrow \left(\frac{c\Delta t}{\Delta x}\right)^2\leq 1$ because $0\leq\sin^2\leq 1$
This scheme is stable under the CFL's condition: $\boxed{|\frac{c\Delta t}{\Delta x}|\leq 1}$.

1.3 The order of the scheme

What is the order of this scheme? (Without the second member)

Let $u(x_i,t^n)$ be a solution of the wave equation, using the Taylor expansion, we find that :

$$\frac{1}{\Delta t^2}\{u(x_i,t^n+\Delta t)-2u(x_i,t^n)+u(x_i,t^n-\Delta t)\}$$

$$=\frac{1}{\Delta t^2}\{u(x_i,t^n)(2-2)+\frac{\partial u(x_i,t^n)}{\partial t}(\Delta t-\Delta t)+\frac{\partial^2 u(x_i,t^n)}{\partial t^2}(\frac{\Delta t^2}{2}+\frac{\Delta t^2}{2})$$
$$+\frac{\partial^3 u(x_i,t^n)}{\partial t^3}(\frac{\Delta t^3}{6}-\frac{\Delta t^3}{6})\}+\sigma(\Delta t^2)\,.$$

and $\frac{c^2}{\Delta x^2}\{u(x_i+\Delta x,t^n)-2u(x_i,t^n)+u(x_i-\Delta x,t^n)\}$

$$=\frac{c^2}{\Delta x^2}\{u(x_i,t^n)(2-2)+\frac{\partial u(x_i,t^n)}{\partial x}(\Delta x-\Delta x)+\frac{\partial^2 u(x_i,t^n)}{\partial x^2}(\frac{\Delta x^2}{2}+\frac{\Delta x^2}{2})$$
$$+\frac{\partial^3 u(x_i,t^n)}{\partial x^3}(\frac{\Delta x^3}{6}-\frac{\Delta x^3}{6})\}+\sigma(\Delta x^2)\,.$$

So, the scheme is rewritten :

$$\frac{\partial^2 u(x_i,t^n)}{\partial t^2}-c^2\frac{\partial^2 u(x_i,t^n)}{\partial x^2}+\sigma(\Delta t^2+\Delta x^2)$$

$u(x_i,t^n)$being a solution of the wave equation. Then, the order of the scheme is 2 in time and in space.

1.4 Solving area

We use this scheme in a "box" large enough to have boundary conditions equal to zero, as this picture shows you,

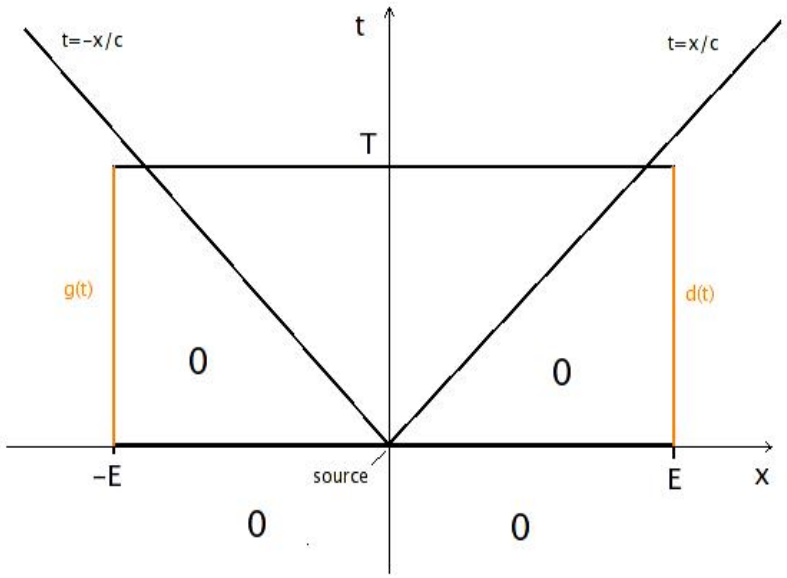

Figure 13: solving "box"

The information is located in the superior cone, and we will use $g(t) = d(t) = 0$, $\forall t \in [0, T]$, if $E > cT$, in the scheme. We have to choose a good area where we solve the equation to have boundary conditions equals to zero.

1.5 The results

For $c = 1$, $T = 2$ (interval of time $[0, T]$), $E = 2$ (interval of space $[-E, E]$), $\sigma = 0.1$ and a number of time steps equals to 200 and a number of space steps equals to 400, we obtain ($\frac{c\Delta t}{\Delta x} = 1$):

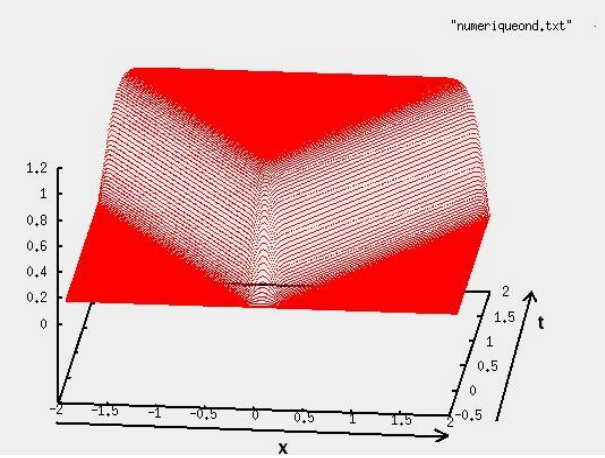

Figure 14: u_j^n

For $c = 1$, $T = 1$ (interval of time $[0, T]$), $E = 1.5$ (interval of space $[-E, E]$), $\sigma = 0.1$ and a number of time steps equals to 200 and a number of space steps equals to 400, we obtain ($\frac{c\Delta t}{\Delta x} \simeq 0.67$):

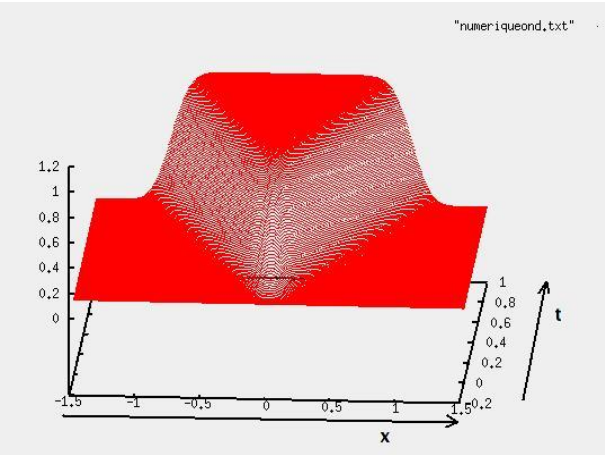

Figure 15: u_j^n

2 Time reversal with opposite speed

We are going to do a time reversal with opposite speed, to do so, we have to solve the following system :

$$\begin{cases} \frac{\partial^2 v}{\partial t^2} - c^2 \frac{\partial^2 v}{\partial x^2} = 0 & \forall (x,t) \in \Omega = [-E,E] \times [0,T] \\ v(-E,t) = u(-E, T-t) & \forall t \in [0,T] \\ v(E,t) = u(E, T-t) & \forall t \in [0,T] \\ v(x,0) = u(x,T) & \forall x \in [-E,E] \\ \frac{\partial v}{\partial t}(x,0) = \boxed{-\frac{\partial u}{\partial t}(x,T)} & \forall x \in [-E,E] \end{cases}$$

where $u(-E, T-t)$ and $u(E, T-t)$ are measures.

But experimentally, it is impossible to know the wave speed at a given instant T in all space, however this system is really interesting because the time reversal will show you all the past of the wave.

2.1 The scheme

To solve this system, we use the same scheme as before with the boundary values $g(t)$ and $d(t)$, but without the second member (because there is no source).

$$\frac{v_i^{n+1} - 2v_i^n + v_i^{n-1}}{\Delta t^2} - c^2 \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta x^2} = 0\,.$$

We obtain,

$$v_i^{n+1} = 2v_i^n - v_i^{n-1} + \frac{c^2 \Delta t^2}{\Delta x^2} \left(v_{i+1}^n - 2v_i^n + v_{i-1}^n \right)\,.$$

2.2 The results

We make the time reversal of the two previous signal and we obtain the following curves: For the first graph (fig.14),

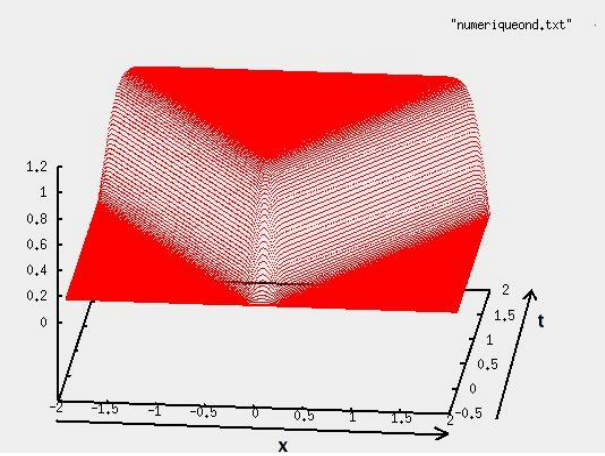

Figure 16: right direction u_j^n

And the reversal,

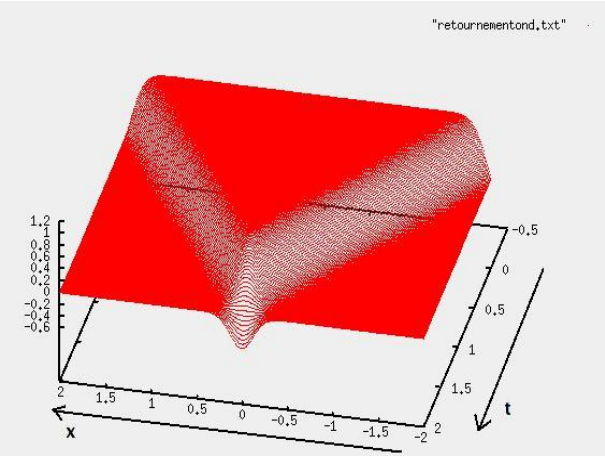

Figure 17: time reversal v_j^n

When we compare u_i^0 and v_i^T, we see a well located at $x = 0$,

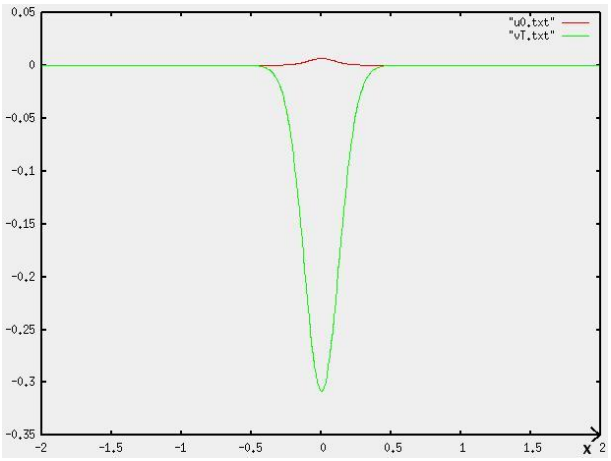

Figure 18: u_j^0 and v_j^T

For the second graph (fig.15),

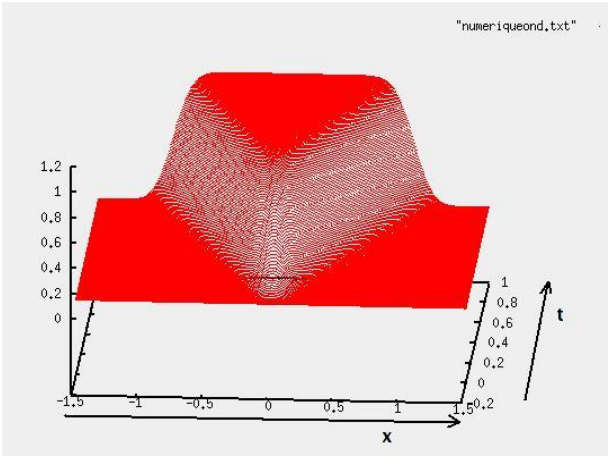

Figure 19: right direction u_j^n

And the reversal,

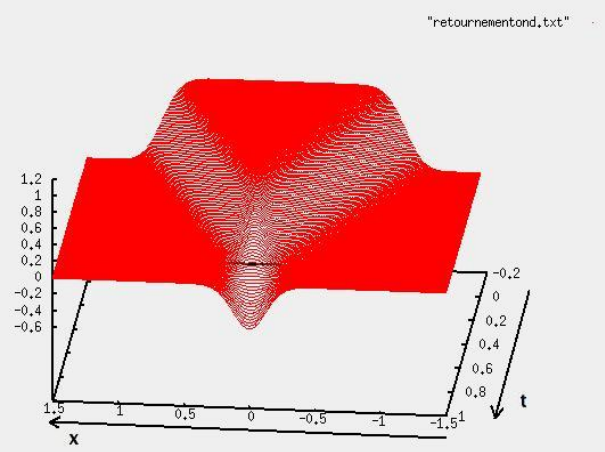

Figure 20: time reversal v_j^n

When we compare u_i^0 and v_i^T, we see a well located at $x = 0$,

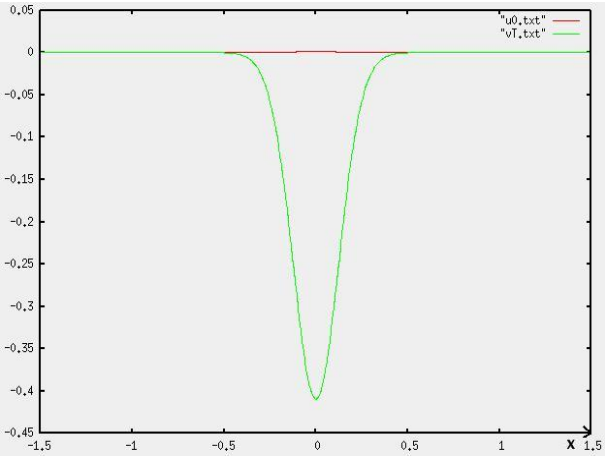

Figure 21: u_j^0 and v_j^T

If we take an other initial signal larger than the Gauss function, as :

$$\frac{1}{2\pi\sigma^2} e^{\frac{-(i\Delta x^2 + n\Delta t^2)}{2\sigma^2}} \times (1 + \alpha \sin(\beta x)) \ ,$$

For $c = 1$, $T = 2$ (interval of time $[0, T]$), $E = 3$ (interval of space $[-E, E]$), $\sigma = 0.5$, $\alpha = 10$, $\beta = 20$, a number of time steps equals to 200 and a number of space steps equals to 400, we obtain :

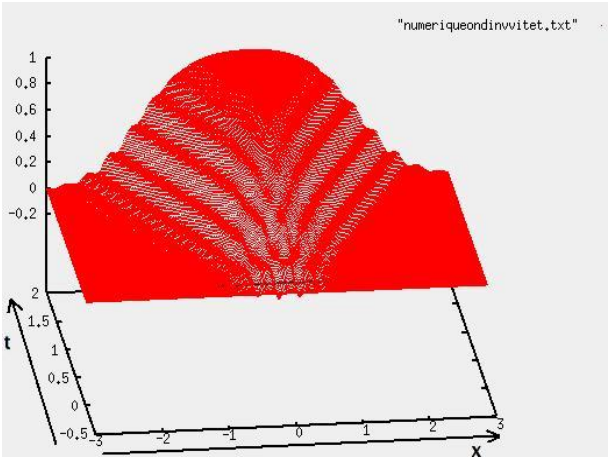

Figure 22: right direction u_j^n

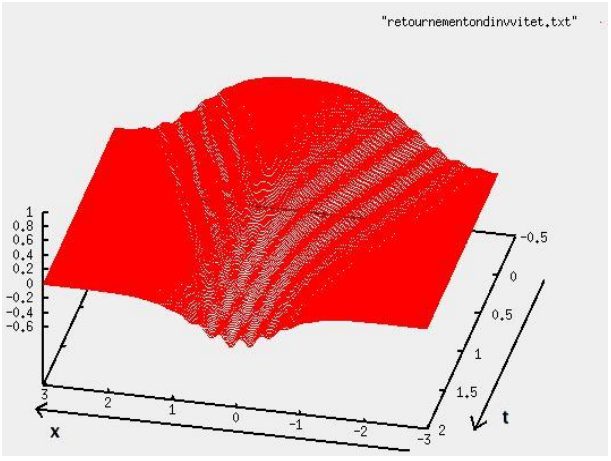

Figure 23: time reversal v_j^n

In the three cases, we see that the source is located at $x = 0$. The reversal v_i^n become negative for $n\Delta t > T$, because there is not a well at $x = 0$ to absorb the wave. The wave relive its past.

2.3 Explanation of the right reversal results

We have,

$$(6)\begin{cases} \frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = \delta_{t=0,x=0} & \forall x \ ,\forall t \in [0,T] \\ u(x,0) = u_0(x) & \forall x \\ \frac{\partial u}{\partial t}(x,0) = u_1(x) & \forall x \end{cases}$$

We fix the reversal system,

$$(7)\begin{cases} \frac{\partial^2 v}{\partial t^2} - c^2 \frac{\partial^2 v}{\partial x^2} = 0 & \forall x \ ,\forall t \in]0,T[\\ v(x,0) = u(x,T) & \forall x \\ \frac{\partial v}{\partial t}(x,0) = -\frac{\partial u}{\partial t}(x,T) & \forall x \end{cases}$$

We know that the solution is unique, because it is a Dirichlet problem.
We fix $w(x,t) = u(x,T-t)$, where u is solution of (6). Then we find,

$$\begin{cases} \frac{\partial^2 w}{\partial t^2} - c^2 \frac{\partial^2 w}{\partial x^2} = 0 & \forall x \ ,\forall t \in]0,T[\\ w(x,0) = u(x,T) & \forall x \\ \frac{\partial w}{\partial t}(x,0) = -\frac{\partial u}{\partial t}(x,T) & \forall x \end{cases}$$

Thus, w is the solution of (7). That is why the time reversal with opposite speed gives us all the past of the wave.

3 Time reversal with the same speed and boundary conditions

In this part, we try to make a time reversal without opposite speed. We hold the same speed that is experimentally executable because we don't have to change the speed at the instant T. And we reduce the solving box for having non null boundary conditions. Then, we have to solve this system :

$$\begin{cases} \frac{\partial^2 v}{\partial t^2} - c^2 \frac{\partial^2 v}{\partial x^2} = 0 & \forall (x,t) \in \Omega = [-E,E] \times [0,T] \\ v(-E,t) = u(-E,T-t) & \forall t \in [0,T] \\ v(E,t) = u(E,T-t) & \forall t \in [0,T] \\ v(x,0) = u(x,T) & \forall x \in [-E,E] \\ \frac{\partial v}{\partial t}(x,0) = \boxed{+\frac{\partial u}{\partial t}(x,T)} & \forall x \in [-E,E] \end{cases}$$

3.1 With a higher initial speed

A problem appears because if $\frac{\partial u}{\partial t}(T,x)$ is not close to zero, we start the numerical scheme with bad information for the speed but good information for boundary conditions, as we can see on this picture,

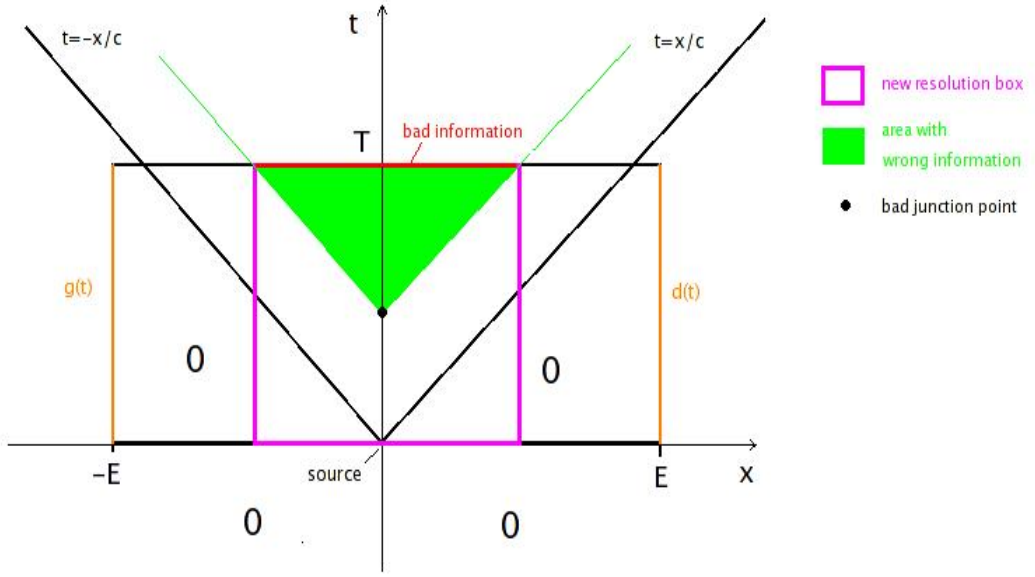

Figure 24: resolution problem

To have a large enough speed, we choose an origin signal equals to :

$$\frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+t^2)}{2\sigma^2}} \times (1+t) \ .$$

And the discretization,

$$\frac{1}{2\pi\sigma^2} e^{\frac{-(i\Delta x^2+n\Delta t^2)}{2\sigma^2}} \times (1+n\Delta t) \ .$$

For $c = 1$, $T = 1$ (interval of time $[0, T]$), $E = 1.5$ (interval of space $[-E, E]$), $\sigma = 0.5$ and a number of time steps equals to 200 and a number of space steps equals to 400 ($\frac{c\Delta t}{\Delta x} \simeq 0.67$), we obtain:

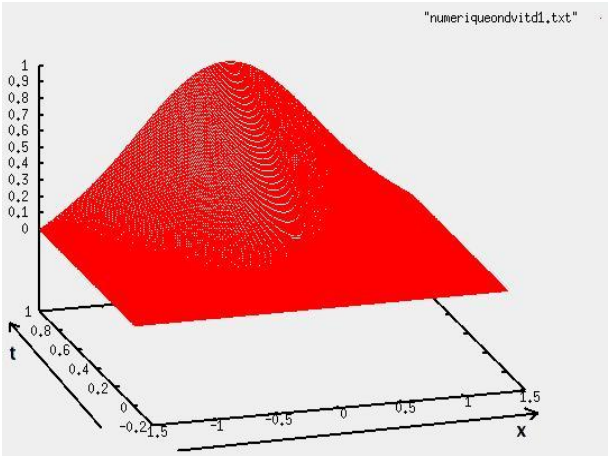

Figure 25: right direction u_j^n

We make the reversal on a new space interval $[-E_1, E_1]$, with $E_1 = 0.75$,

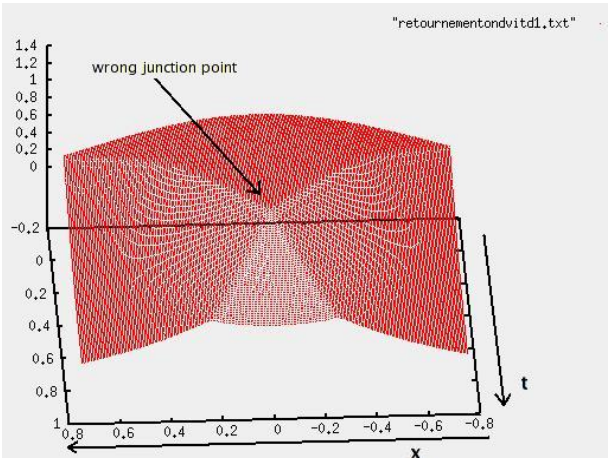

Figure 26: time reversal v_j^n

We see clearly the wrong junction point, that is to say for $t \neq T$.

3.2 With a initial speed close to zero

With a initial speed close to zero, we have good initial conditions, because $\frac{\partial u}{\partial t}(T,x) \sim -\frac{\partial u}{\partial t}(T,x)$. Thus, we have a good focalisation.
With a signal source equals to,

$$\frac{1}{2\pi\sigma^2} e^{\frac{-(i\Delta x^2 + n\Delta t^2)}{2\sigma^2}}$$

For $c = 1$, $T = 2$ (interval of time $[0,T]$), $E = 2.1$ (interval of space $[-E,E]$), $\sigma = 0.1$ and a number of time steps equals to 200 and a number of space steps equals to 400 ($\frac{c\Delta t}{\Delta x} \simeq 0.95$), we obtain:

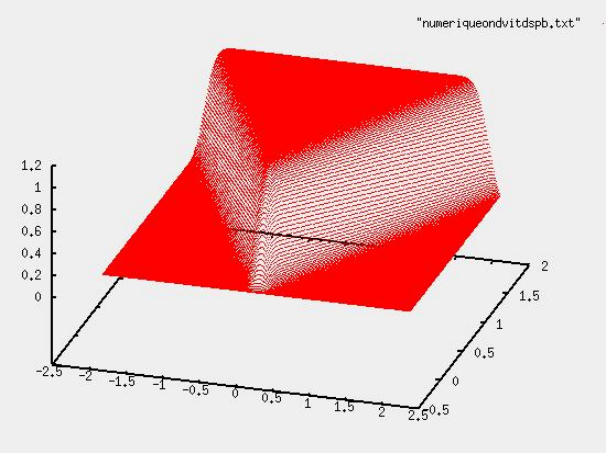

Figure 27: right direction u_j^n

We make the reversal on a new space interval $[-E_1, E_1]$, with $E_1 = 1.6$,

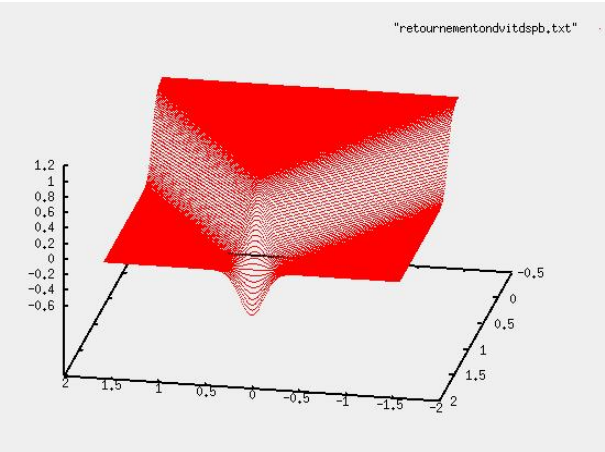

Figure 28: time reversal v_j^n

V Conclusion

We were interested in the time reversal after the Mathias FINK's colloquium. This project was an opportunity for us to work on a pleasant project. We have got the possibility to work on two main parts to approach the problem of reversibility (through an example in 0-dimension: the springs, and in 1-dimension: the vibrating string).

We observed that time reversal can be used on theoretical problems but in pratice, we need to fix specific conditions (lower speeds, particular periods, etc ...). Using time reversal to solve physical experience wasn't a success, the obtained results introduce problems which could be solved with time and many tests to find solutions without using an opposite speed. In fact, physicists and engineers don't try to find exactly the initial conditions but a good enough localization spatiotemporal by dint of a focalization. It's a reaction that we have seen in the Mathias FINK's colloquium with the silicium faceplate example..

Indeed the obtained results introduce problems which could be solved with time and many tests to find solutions without using an opposite speed. Moreover this project gives us numerical methods to solve complex systems thanks to schemes used to approximate the second derivative.

Eventually we worked together to increase the performance of our team, on each problems met (often on the modelisation for the strength and second member in equations). Thanks to the professors for their implication in this very interesting project.

Bibliography

[1] Mathematical foundations of the time reversal mirror, C. BARDOS and M. FINK.

[2] http://en.wikipedia.org/wiki/Time_Reversal_Signal_Processing

[3] Asymptotic analysis 29, p. 157-182, (2002)

[4] Methods of Mathematical Physics, COURANT and HILBERT, 1953.

[5] Methodes Mathématiques pour les Sciences Physiques, 1965, Hermann, seconde édition 1987

Part

Annexe

I The spring equation

```
void explicite(int dim, float c, float u0, float u1, float T) {
  int i;
  float tmp,pas,cmp;
  FILE *f;

  pas=T/dim;
  cmp=0;

  f=fopen("explicite.txt","w");
  for(i=1;i<=dim;i++) {
    tmp=u0*cos(c*cmp)+(u1/c)*sin(c*cmp);
    fprintf(f,"%f %f\n",cmp,tmp);
    cmp=cmp+pas;
  }
  fclose(f);
}

void numerique(int dim, float c, float u0, float u1, float T) {
  int i;
  float pas,cmp;
  float num[dim];
  FILE *f;

  pas=T/dim;
  cmp=0;

  num[0]=u0-u1*pas/2;
  num[1]=u0+u1*pas/2;

  f=fopen("numerique.txt","w");

  fprintf(f,"%f %f\n",cmp,num[0]);
  cmp=cmp+pas;
  fprintf(f,"%f %f\n",cmp,num[1]);
  cmp=cmp+pas;

  for(i=1;i<dim-1;i++) {
    num[i+1]=num[i]*(2-c*c*pas*pas)-num[i-1];
    fprintf(f,"%f %f\n",cmp,num[i+1]);
    cmp=cmp+pas;
  }
  fclose(f);
  printf("mesure de l'erreur (pour eq=0) %f\n",u0*cos(c*T)+(u1/c)*sin(c*T)-(num[dim-1]*(2-c*c*pas*pas)-num[dim-2]));
}
```

```
//------------------The case of one spring without strength measures--

m=2; //weight of mass
k=32; //straightness
c=sqrt(k/m);
u0=2;
ul=1;
A=u0;
B=ul/c;
T=6*%pi; //length of the interval
N=200; //number of steps

for i=1:N+1
t(i)=(i-1)*T/N;
u(i)=A*cos(c*t(i))+B*sin(c*t(i)); //right position
ut(i)=-A*c*sin(c*t(i))+B*c*cos(c*t(i)); //right speed
end;

C=u(N+1);
D=-ut(N+1)/c;

for i=1:N+1
t(i)=(i-1)*T/N;
v(i)=C*cos(c*t(i))+D*sin(c*t(i));//opposite position
vt(i)=-C*c*sin(c*t(i))+D*c*cos(c*t(i));//opposite speed
end;


xset("window",0);
xname("équation x''+c²x=0 right & poosite direction & speed");
xtitle("équation x''+c²x= right & opposite direction & speed");
plot(t,u,'-r'); //plot of the position
plot(t,ut,'-black'); //plot of the speed
plot(t,v,'+g'); //plot of the opposite position
plot(t,vt,'+b'); //plot of the opposite speed
```

```
//------------------The case of one spring with strengh measures-------------------------
//right direction


m=2; //weight of mass
k=8; //straightness
w=sqrt(k/m);
T=6*%pi; //length of the interval
N=400; //number of steps

for i=1:N+1
t(i)=(i-1)*T/N;
y(i)=1/w*sin(w*t(i)); //position
yt(i)=cos(w*t(i)); //speed
end;

plot(t,y,"-g"); //plot right position
plot(t,yt,"-b"); //plot right speed


//----------Time reversal-------------

//the coefficients
A=1/w*sin(w*T);
C=k*cos(w*T)/(2*w^2);
B=cos(w*T)/w-C/w;
D=k*sin(w*T)/(2*w^2);

for i=1:N+1
t(i)=(i-1)*T/N;
z(i)=A*cos(w*t(i))+B*sin(w*t(i))+C*t(i)*cos(w*t(i))+D*t(i)*sin(w*t(i)); //position
zt(i)=-A*w*sin(w*t(i))+B*w*cos(w*t(i))+C*cos(w*t(i))+D*sin(w*t(i))-w*C*t(i)*sin(w*t(i))+w*D*t(i)*cos(w*t(i)); //speed
end;

plot(t,z,"-black"); //plot opposite position
plot(t,zt,"-r"); //plot opposite speed
plot(t,1,"-"); //the initial speed (to compare with zt(T))
```

```
// This is the program used to obtain the curves involving many springs.

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cmath>
using namespace std;

#define pi 3.141592654

void curves (int nbmass,float weight, int nbmes, float k,int n, float T)
{

  // In first we draw the right direction curves

  float V0=1;
  float step;
  step=T/(nbmes-1);
``

  // Matrix of masses

  float M[nbmass][nbmass];
  for(int i=0;i<=nbmass-1;i++){
    for(int j=0;j<=nbmass-1;j++){
      M[i][j]=0;
    }
  }
``
  /*
  for(int i=0;i<=nbmass-1;i++){
    cout <<"weight mass"<<i+1<<" = ";
    cin >>M[i][i];
  }
  */
``
  for(int i=0;i<=nbmass-1;i++){
    M[i][i]=weight;
  }

  // Matrix K which depends on the springs'constant as mÜ=KU

  float K[nbmass][nbmass];
  for(int i=0;i<=nbmass-1;i++){
    for(int j=0;j<=nbmass-1;j++){
      K[i][j]=0;
    }
  }
  K[0][0]=-2*k;
  K[0][1]=k;
  for(int i=1;i<=nbmass-2;i++){
    K[i][i-1]=k;
    K[i][i]=-2*k;
    K[i][i+1]=k;
  }
```

```
  K[nbmass-1][nbmass-2]=k;
  K[nbmass-1][nbmass-1]=-k;

  // Inverse matrix of masses

  float N[nbmass][nbmass];
  for(int i=0;i<=nbmass-1;i++){
    for(int j=0;j<=nbmass-1;j++){
      N[i][j]=0;
    }
  }
  for(int i=0;i<=nbmass-1;i++){
    N[i][i]=1/(M[i][i]);
  }

  // Matrix involve in the numerical scheme: P=(1/M)K

  float P[nbmass][nbmass];
  for(int i=0;i<=nbmass-1;i++){
    for(int j=0;j<=nbmass-1;j++){
      P[i][j]=0;
    }
  }
  for (int i=0;i<=nbmass-1;i++){
    for (int j=0;j<=nbmass-1;j++){
      for (int l=0;l<=nbmass-1;l++){
        P[i][j]+=N[i][l]*K[l][j];
      }
    }
  }

  // Initialization Uk(0)=0 et Uk'(0)=1

  ofstream alp("allerpos.txt");
  ofstream alv("allervit.txt");

  float U[nbmass][nbmes];

  for(int i=0;i<=nbmass-1;i++){
    if (i == (n-1))
      {
        U[i][0]=-(step/2)*V0;
        U[i][1]=(step/2)*V0;
      }
    if (i != (n-1))
      {
        U[i][0]=0;
        U[i][1]=0;
      }
  }
```

```
  alp<<0<<' '<<U[n-1][0]<<endl;
  alp<<step<<' '<<U[n-1][1]<<endl;
  alv<<0<<' '<<V0<<endl;


  // Resolution of U

  float PU[nbmass];
  for (int i=0;i<=nbmass-1;i++){
    PU[i]=0;
  }

  for(int j=2;j<=nbmes-1;j++){
    for(int i=0;i<=nbmass-1;i++){

      for (int a=0;a<=nbmass-1;a++){
        for (int b=0;b<=nbmass-1;b++){
          PU[a]+=P[a][b]*U[b][j-1];
        }
      }

      U[i][j]=2*U[i][j-1]-U[i][j-2]+step*step*PU[i];
    }
    for (int i=0;i<=nbmass-1;i++){
      PU[i]=0;
    }
    alp<<j*step<<" "<<U[n-1][j]<<endl;
  }

  // Résolution of his speed

  float speed[nbmass];
  for(int j=1;j<=nbmes-2;j++){
    for(int i=0;i<=nbmass-1;i++){
      speed[i]=(U[i][j+1]-U[i][j-1])/(2*step);
    }
    alv<<j*step<<' '<<speed[n-1]<<endl;
  }
  float Un[nbmass];
  for (int a=0;a<=nbmass-1;a++){
    for (int b=0;b<=nbmass-1;b++){
      PU[a]+=P[a][b]*U[b][nbmes-1];
    }
  }
  for(int i=0;i<=nbmass-1;i++){
    Un[i]=2*U[i][nbmes-1]-U[i][nbmes-2]+step*step*PU[i];
  }
  for(int i=0;i<=nbmass-1;i++){
    speed[i]=(Un[i]-U[i][nbmes-2])/(2*step);
  }
  alv<<(nbmes-1)*step<<" "<<speed[n-1]<<endl;
```

```
// TIME REVERSAL


// Initialization

ofstream retp("retourpos.txt");
ofstream retv("retourvit.txt");

float V[nbmass][nbmes];
for(int i=0;i<=nbmass-1;i++){
  V[i][0]=U[i][nbmes-1]+speed[i]*step/2;
  V[i][1]=U[i][nbmes-1]-speed[i]*step/2;
}

retp<<T<<' '<<V[n-1][0]<<endl;
retp<<T+step<<' '<<V[n-1][1]<<endl;
retv<<T<<' '<<-speed[n-1]<<endl;


// Resolution of V

float PV[nbmass];
for (int i=0;i<=nbmass-1;i++){
  PV[i]=0;
}

for(int j=2;j<=nbmes-1;j++){
  for(int i=0;i<=nbmass-1;i++){
      for (int a=0;a<=nbmass-1;a++){
        for (int b=0;b<=nbmass-1;b++){
          PV[a]+=P[a][b]*V[b][j-1];
        }
      }...

      // Here we can add a strength at each step of time and we can choose the
masses on which we apply a strength. An example is given in commentary later.

      if (i == (n-1))
        {
          V[i][j]=2*V[i][j-1]-V[i][j-2]+step*step*PV[i];
        }
      if (i != (n-1))
        {
          V[i][j]=2*V[i][j-1]-V[i][j-2]+step*step*PV[i];
        }
    }

    for (int i=0;i<=nbmass-1;i++){
      PV[i]=0;
    }

    retp<<T+j*step<<" "<<V[n-1][j]<<endl;
  }
```

```
  // For the Chapter I.6.3 we have used different strength in order to do time
reversal. In each test we choose T after having ascertained that U(T) is
close to 0. That's an example of one of our test:
  /*-------------------------------------------------------------
for(int j=2;j<=nbmes-1;j++){
    for(int i=0;i<=nbmass-1;i++){
......
      for (int a=0;a<=nbmass-1;a++){
        for (int b=0;b<=nbmass-1;b++){
          PV[a]+=P[a][b]*V[b][j-1];
        }
      }...
......
      if (i == 0)
        {
          V[i][j]=2*V[i][j-1]-V[i][j-2]+step*step*PV[i]+step*step*(2*k*U[i][nbm
es-1-j]-k*U[i+1][nbmes-1-j]);
        }
      if ((i != 0) && (i != (nbmass-1)))
        {
          V[i][j]=2*V[i][j-1]-V[i][j-2]+step*step*PV[i]+step*step*(-k*U[i-1][nb
mes-1-j]+2*k*U[i][nbmes-1-j]-k*U[i+1][nbmes-1-j]);
        }
      if (i == nbmass-1)
        {
          V[i][j]=2*V[i][j-1]-V[i][j-2]+step*step*PV[i]+step*step*(-k*U[i-1][nb
mes-1-j]+k*U[i][nbmes-1-j]);
        }
    }
    for (int i=0;i<=nbmass-1;i++){
      PV[i]=0;
    }
......
    retp<<T+j*step<<" "<<V[n-1][j]<<endl;
  }
-------------------------------------end of example-----------*/


  // speed of V
..
  for(int j=1;j<=nbmes-2;j++){
    for(int i=0;i<=nbmass-1;i++){
      speed[i]=(V[i][j+1]-V[i][j-1])/(2*step);
    }....
    retv<<T+j*step<<' '<<speed[n-1]<<endl;
  }
  float Vn[nbmass];
  for (int a=0;a<=nbmass-1;a++){
    for (int b=0;b<=nbmass-1;b++){
      PV[a]+=P[a][b]*V[b][nbmes-1];
    }
  }
  for(int i=0;i<=nbmass-1;i++){
    Vn[i]=2*V[i][nbmes-1]-V[i][nbmes-2]+step*step*PV[n-1];
    speed[i]=(Vn[i]-V[i][nbmes-2])/(2*step);
  }
```

```
  retv<<T+(nbmes-1)*step<<" "<<speed[n-1]<<endl;
}

int main()
{
  int nbmes,nbmass,n;
  float k,T,weight;
  cout<<"number of masses =";
  cin>>nbmass;
  cout<<"weight of masses=";
  cin>>weight;
  cout<<"number of measures =";
  cin>>nbmes;
  cout<<"springs'constant =";
  cin>>k;
  cout<<"number of the mass where we use a local strength =";
  cin>>n;
  cout<<"T=";
  cin>>T;

  curves(nbmass,weight,nbmes,k,n,T);
}
```

```
//-------------many springs --------------------------
//------------a scilab routine to give the stability---------------

n=10; // n = number of masses
k=32; // straightness
m=15; // weight of masses
DT=0.07; //step of time



a=poly(0,'a');
a=DT^2*k/m;

//the build of the matrix 'mat':
K=(-2*a+2)*eye(n,n) + a*diag(ones(n-1,1),1) +a*diag(ones(n-1,1),-1); //matrice K
K(n/2,n/2)=-a+2;
K(n/2+1,n/2)=0;
b=-diag(ones((n-n/2),1),n/2);
j=diag(ones((n-n/2),1),-n/2);
for i=n/2:n
K(i,n/2+1:n)=0;
end



mat=b+j+K



eigenvalues=spec(mat) //eigen values
norm_eigenvalues=abs(spec(mat)) //norm of eigen values
```

II In 1-dimension : the wave equation

```
void numeriqueinvvit(int dim, float c, float T,float E) {
  int n,j,i;
  float x,t,beta,gamma,pastmp,pasesp,sigma,norml2;
  float g[dim],d[dim];
  FILE *f1,*f2;
  float u[dim+2][2*dim+1],v[dim+2][2*dim+1];

  sigma=0.1;
  beta=1/(sqrt(2*pi)*sigma);

  pasesp=E/dim;

  pastmp=T/dim;

  gamma=pastmp*pastmp*c*c/(pasesp*pasesp);

  for(i=0;i<=dim+1;i++) {
    g[i]=0;
    d[i]=0;
  }

  f1=fopen("numeriqueondinvvit.txt","w");

  for(j=0;j<=2*dim;j++) {

      u[0][j]=0;
      u[1][j]=0;

    fprintf(f1,"%f %f %f\n",(j-dim)*pasesp,-pastmp,u[0][j]);
    fprintf(f1,"%f %f %f\n",(j-dim)*pasesp,0.0,u[1][j]);

  }

  for(n=2;n<=dim+1;n++) {
    for(j=0;j<=2*dim;j++) {

      if(j==0) {
        u[n][j]=g[n];
      }

      else { if(j==2*dim) {
          u[n][j]=d[n];
        }

        else {
          x=(j-dim)*pasesp;
          t=(n-2)*pastmp;
          u[n][j]=gamma*(u[n-1][j+1]-2*u[n-1][j]+u[n-1][j-1])+2*u[n-1][j]-u[n-2][j]+4*pastmp*pastmp*beta*beta*exp(-((x*x)+(t*t
))/(2*sigma*sigma));
        }
      }

      fprintf(f1,"%f %f %f\n",(j-dim)*pasesp,(n-1)*pastmp,u[n][j]);
    }
  }
  fclose(f1);
```

```
  f2=fopen("retournementondinvvit.txt","w");

  for(j=0;j<=2*dim;j++) {

    v[0][j]=u[dim+1][j];

    v[1][j]=u[dim][j];

    fprintf(f2,"%f %f %f\n",(j-dim)*pasesp,0.0,v[0][j]);
    fprintf(f2,"%f %f %f\n",(j-dim)*pasesp,pastmp,v[1][j]);

  }

  for(n=2;n<=dim+1;n++) {
    for(j=0;j<=2*dim;j++) {

      if(j==0) {
        v[n][j]=g[dim+1-n];
      }

      else { if(j==2*dim) {
          v[n][j]=d[dim+1-n];
        }

        else {
          v[n][j]=gamma*(v[n-1][j+1]-2*v[n-1][j]+v[n-1][j-1])+2*v[n-1][j]-v[n-2][j];
        }
      }

      fprintf(f2,"%f %f %f\n",(j-dim)*pasesp,n*pastmp,v[n][j]);
    }
  }
  fclose(f2);

  f1=fopen("u0.txt","w");
  f2=fopen("vT.txt","w");

  norml2=0;

    for(j=0;j<=2*dim;j++) {

      fprintf(f1,"%f %f\n",(j-dim)*pasesp,u[2][j]);
      fprintf(f2,"%f %f\n",(j-dim)*pasesp,v[dim-3][j]);

      norml2=norml2+(u[2][j]-v[dim-3][j])*(u[2][j]-v[dim-3][j])*pasesp;

    }

    fclose(f2);
    fclose(f1);

    printf("la norme l2 de u-v  est : %f\n",sqrt(norml2));

}
```