MATH0024 PDEs Homework 2

Maarten Arnst and Romain Boman

1 Organization

- The report must be sent in PDF format by email to both R. Boman (r.boman@uliege.be) and M. Arnst (maarten.arnst@uliege.be) before/on December 20th, 2024. Please attach your Python code.
- You are welcome to email R. Boman or M. Arnst to ask questions as well as to ask for feedback on one or more intermediate versions of your report and/or your code. It is proposed that Louis Basset, Fanny Bodart, Maxime Borbouse, Elise Boury Schmidt, Julien Brandoit, Ioan-Catalin Chirita and Adrien Daloze-Tilman contact M. Arnst, and Alessandro Demoors, Yann Paul Fotsing Takam Fonkou, Henri Gérard, Constant Gobron, Lydia Leruth, Guillaume Maertens de Noordhout, Arnaud Radermecker and Alejandro Sior contact R. Boman

2 Assignment

2.1 Local-Time-Stepping – Leap-Frog for the wave equation in 1D

Let us consider the following boundary-value problem involving the wave equation in a 1D domain extending from 0 to L:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} - c^2 \dfrac{\partial^2 u}{\partial x^2} = 0 & \text{in }]0, L[\times]0, T[\\ u(x,0) = u_0(x) & \text{on }]0, L[\times\{t=0\} \\ \dfrac{\partial u}{\partial t}(x,0) = v_0(x) & \text{on }]0, L[\times\{t=0\} \\ \dfrac{\partial u}{\partial x}(0,t) = 0 & \text{on } \{x=0\}\times]0, T[\\ \dfrac{\partial u}{\partial x}(L,t) = 0 & \text{on } \{x=L\}\times]0, T[\end{cases} \tag{1}$$

with c the wave propagation speed.

Using the finite element method and linear basis functions $\{\phi_i\}_{i=1,\ldots,N}$, the problem can be spatially discretized, leading to the following semi-discretized system of equations:

$$\begin{cases} [M]\dfrac{d^2\boldsymbol{u}}{dt^2}(t) + [K]\boldsymbol{u}(t) = 0 & \text{for } t \in]0, T[\\ \boldsymbol{u}(0) = \boldsymbol{u}_0 & \text{at } t = 0 \\ \dfrac{d\boldsymbol{u}}{dt}(0) = \boldsymbol{v}_0 & \text{at } t = 0 \end{cases} \tag{2}$$

where $[M]$ is a $N \times N$ symmetric positive definite sparse matrix, $[K]$ is a $N \times N$ symmetric positive semidefinite sparse matrix, $\boldsymbol{u}(t) \in \mathbb{R}^N$ contains the coefficients of the approximation of the solution with respect to the basis functions and $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ contain the nodal values of the initial conditions.

Defining a time step Δt and $t_n = n\Delta t$, this system can be discretized in time using the second-order "leap-frog" method:

$$[M]\frac{\boldsymbol{u}_{n+1} - 2\boldsymbol{u}_n + \boldsymbol{u}_{n-1}}{\Delta t^2} + [K]\boldsymbol{u}_n = \boldsymbol{0} \tag{3}$$

Unfortunately this method is not explicit because the matrix $[M]$ is not diagonal. Mass-lumping consists in replacing $[M]$ by a diagonal approximation:

$$\bar{M}_{ij} = \delta_{ij} \sum_{k=1}^{N} M_{ik} \tag{4}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

After this approximation, the system to be solved can be multiplied by $[\bar{M}]^{-\frac{1}{2}}$

$$[\bar{M}]^{\frac{1}{2}}\frac{d^2\boldsymbol{u}}{dt^2}(t) + \underbrace{[\bar{M}]^{-\frac{1}{2}}[K][\bar{M}]^{-\frac{1}{2}}}_{[A]}\underbrace{[\bar{M}]^{\frac{1}{2}}\boldsymbol{u}(t)}_{\boldsymbol{z}(t)} = 0 \tag{5}$$

Let $\boldsymbol{z} = [\bar{M}]^{\frac{1}{2}}\boldsymbol{u}$; the system to be solved becomes:

$$\frac{d^2\boldsymbol{z}}{dt^2} + [A]\boldsymbol{z} = 0 \tag{6}$$

Using the leap-frog time-marching method with this new system of equations on a regular mesh, it can be proven that the resulting numerical scheme is stable if

$$\frac{|c|\Delta t}{h} \leq 1 \tag{7}$$

where h is the size of the finite elements. In other words, the size of the time step must be decreased if the mesh is refined. This stability condition becomes a problem when some parts of the mesh are locally refined with very small finite elements. In this case, the stability condition involves the smallest element size $h = \min(h_i)$ and produces a small time step even if most of the elements of the mesh are large.

Local time-stepping methods have been developed [1] in order to overcome this issue. The idea is to use the maximum allowable time step size for the whole problem as if it was not locally refined. Then, a sub-problem involving only the small elements is solved with a smaller time step calculated from the local stability condition. This sub-problem remains explicit and concerns a limited set of nodes. The resolution is thus much faster than solving the whole problem with a unique global reduced time step.

In the frame of the leap-frog method, the local time-stepping method can be deduced from the following equality which is valid for any function f in $\mathcal{C}^1$:

$$f(t + \Delta t) - 2f(t) + f(t - \Delta t) = \Delta t^2 \int_{-1}^{1} (1 - |\theta|) f''(t + \theta\Delta t)\, d\theta \tag{8}$$

Using this equality, the exact solution of the considered boundary value problem can be written as:

$$\boldsymbol{z}(t+\Delta t) - 2\boldsymbol{z}(t) + \boldsymbol{z}(t-\Delta t) = -\Delta t^2 \int_{-1}^{1} (1-|\theta|)\,[A]\boldsymbol{z}(t+\theta\Delta t)\,d\theta \tag{9}$$

If $[A]\boldsymbol{z}(t+\theta\Delta t)$ is approximated by $[A]\boldsymbol{z}(t)$, this expression gives the classical leap-frog method for $t = t_n$:

$$\boldsymbol{z}_{n+1} - 2\boldsymbol{z}_n + \boldsymbol{z}_{n-1} = -\Delta t^2\,[A]\boldsymbol{z}_n \tag{10}$$

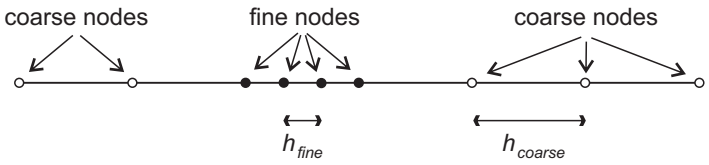

Figure 1: A refined mesh obtained by splitting some elements into several smaller elements.

In the case of a mesh which is locally refined (see fig. 1), we can partition the nodes as "coarse" nodes and "fine" nodes. The fine nodes are those which belong to the smaller elements.

The unknown vector $\boldsymbol{z}(t)$ can be thus written as a sum of 2 contributions:

$$\boldsymbol{z}(t) = ([I]-[P])\boldsymbol{z}(t) + [P]\boldsymbol{z}(t) = \boldsymbol{z}^{[\text{coarse}]}(t) + \boldsymbol{z}^{[\text{fine}]}(t) \tag{11}$$

where $[P]$ is a diagonal projection matrix. Its diagonal entries are 1 for the unknowns associated with the fine region and 0 for the ones of the coarse region.

More explicitly:

$$\begin{Bmatrix} \boldsymbol{z}_C(t) \\ \boldsymbol{z}_F(t) \end{Bmatrix} = \underbrace{\begin{bmatrix} [I] & [0] \\ [0] & [0] \end{bmatrix}}_{[I]-[P]} \boldsymbol{z}(t) + \underbrace{\begin{bmatrix} [0] & [0] \\ [0] & [I] \end{bmatrix}}_{[P]} \boldsymbol{z}(t) = \underbrace{\begin{Bmatrix} \boldsymbol{z}_C(t) \\ \{0\} \end{Bmatrix}}_{\boldsymbol{z}^{[\text{coarse}]}(t)} + \underbrace{\begin{Bmatrix} \{0\} \\ \boldsymbol{z}_F(t) \end{Bmatrix}}_{\boldsymbol{z}^{[\text{fine}]}(t)} \tag{12}$$

This partitioning can be injected into eq. 9 in order to obtain a new numerical method where "coarse" and "fine" unknowns are treated separately:

$$\begin{aligned} &\boldsymbol{z}(t+\Delta t) - 2\boldsymbol{z}(t) + \boldsymbol{z}(t-\Delta t) = \\ &\quad -\Delta t^2 \int_{-1}^{1} (1-|\theta|)\,[A]\left(\boldsymbol{z}^{[\text{coarse}]}(t+\theta\Delta t) + \boldsymbol{z}^{[\text{fine}]}(t+\theta\Delta t)\right)\,d\theta \end{aligned} \tag{13}$$

The first term of the integrand involves the "coarse" unknows. It is approximated as we did above for the classical leap-frog scheme (we consider it constant with respect to θ).

The second term is approximated by a new unknown function $\tilde{\boldsymbol{z}}(\tau)$:

$$\boldsymbol{z}(t+\Delta t) - 2\boldsymbol{z}(t) + \boldsymbol{z}(t-\Delta t) \cong -\Delta t^2 \int_{-1}^{1} (1-|\theta|)\,\left([A]([I]-[P])\boldsymbol{z}(t) + [A][P]\tilde{\boldsymbol{z}}(\theta\Delta t)\right)\,d\theta \tag{14}$$

where $\tilde{\boldsymbol{z}}(\tau)$ is the solution of the following new problem to be solved between $\tau = 0$ (corresponding in practice to $t = t_n$) and $\tau = \Delta t$ (corresponding to $t = t_n + \Delta t = t_{n+1}$):

$$\begin{cases} \dfrac{d^2\tilde{\boldsymbol{z}}}{d\tau^2}(\tau) = -[A]([I]-[P])\boldsymbol{z}(t) - [A][P]\tilde{\boldsymbol{z}}(\tau) \\ \tilde{\boldsymbol{z}}(0) = \boldsymbol{z}(t) \\ \dfrac{d\tilde{\boldsymbol{z}}}{d\tau}(0) = 0 \end{cases} \tag{15}$$

Indeed, if we inject the right-hand side of this new problem into eq. 8 we obtain:

$$\begin{aligned}\tilde{\boldsymbol{z}}(\tau+\Delta t)-2\tilde{\boldsymbol{z}}(\tau)+\tilde{\boldsymbol{z}}(\tau-\Delta t)=\\ -\Delta t^2\int_{-1}^{1}(1-|\theta|)\ ([A]([I]-[P])\boldsymbol{z}(t)+[A][P]\tilde{\boldsymbol{z}}(\tau+\theta\Delta t))\ d\theta\end{aligned} \tag{16}$$

which is the same equation as eq. 14 except that eq. 14 is an approximation and refers to $\boldsymbol{z}$ in the left-hand side instead of $\tilde{\boldsymbol{z}}$.

The important thing to notice is that the term $[A]([I]-[P])\boldsymbol{z}(t)$, which only involves the "coarse" unknowns, remains constant during the numerical integration of eq. 15. This is the most expensive operation of the algorithm as long as the mesh contains only a few small elements. This vector can be computed once at the beginning of each time step, at the beginning of the substeps.

The second term $[A][P]\tilde{\boldsymbol{z}}(\tau)$ only involves the limited number of "fine" unknowns. It is thus rather quick to evaluate.

The last part of the algorithm concerns the computation of $\boldsymbol{z}_{n+1}$ from $\tilde{\boldsymbol{z}}(\Delta t)$

Evaluated for $\tau=0$, eq. 16 gives:

$$\tilde{\boldsymbol{z}}(\Delta t)-2\tilde{\boldsymbol{z}}(0)+\tilde{\boldsymbol{z}}(-\Delta t)=-\Delta t^2\int_{-1}^{1}(1-|\theta|)\ ([A]([I]-[P])\boldsymbol{z}(t)+[A][P]\tilde{\boldsymbol{z}}(\theta\Delta t))\ d\theta \tag{17}$$

From eq. 14 and eq. 17, we deduce that

$$\boldsymbol{z}(t+\Delta t)-2\boldsymbol{z}(t)+\boldsymbol{z}(t-\Delta t)\cong\tilde{\boldsymbol{z}}(\Delta t)-2\tilde{\boldsymbol{z}}(0)+\tilde{\boldsymbol{z}}(-\Delta t) \tag{18}$$

Since the first derivative of $\tilde{\boldsymbol{z}}$ is chosen to be 0 at $\tau=0$ (the reason is omitted here), we have $\tilde{\boldsymbol{z}}(\Delta t)=\tilde{\boldsymbol{z}}(-\Delta t)$

Moreover $\boldsymbol{z}(t)=\tilde{\boldsymbol{z}}(0)$, which gives:

$$\boldsymbol{z}(t+\Delta t)+\boldsymbol{z}(t-\Delta t)\cong 2\tilde{\boldsymbol{z}}(\Delta t) \tag{19}$$

Consequently, the solution $\boldsymbol{z}$ at $t+\Delta t$ can be calculated by

$$\boldsymbol{z}(t+\Delta t)\cong-\boldsymbol{z}(t-\Delta t)+2\tilde{\boldsymbol{z}}(\Delta t) \tag{20}$$

as soon as $\tilde{\boldsymbol{z}}$ is known at the end of the time step.

2.2 Summary of the LTS-LF method

Given a regular mesh with spatial step size h_{coarse}, a time step Δt satisfying the stability condition eq. 7 is chosen. Then, some elements of this mesh are divided into p smaller elements of size $h_{\text{fine}}=h_{\text{coarse}}/p$. The local time-stepping method consists in defining p sub-steps of size $\Delta\tau=\Delta t/p$ for each time step and solving a cheap local problem involving the small finite elements and their unknowns.

Let $t=t_n$. The vectors $\boldsymbol{z}_{n-1}$ and $\boldsymbol{z}_n$ have been previously calculated. We want to compute $\boldsymbol{z}_{n+1}\cong\boldsymbol{z}(t_n+\Delta t)$

We first compute the following vector

$$\boxed{\boldsymbol{w}=[A]([I]-[P])\boldsymbol{z}_n=[A]\boldsymbol{z}_n^{[\text{coarse}]}} \tag{21}$$

This is the application of $[A]$ onto the "coarse" unknowns. This vector will be used during the substeps as a constant. In more details:

$$\begin{aligned}\boldsymbol{w} = \begin{Bmatrix}\{\boldsymbol{w}_C\}\\ \{\boldsymbol{w}_F\}\end{Bmatrix} &= \begin{bmatrix}[A_{CC}] & [A_{CF}]\\ [A_{FC}] & [A_{FF}]\end{bmatrix}\begin{bmatrix}[I] & [0]\\ [0] & [0]\end{bmatrix}\begin{Bmatrix}\{\boldsymbol{z}_C\}\\ \{\boldsymbol{z}_F\}\end{Bmatrix}\\ &= \begin{bmatrix}[A_{CC}] & [0]\\ [A_{FC}] & [0]\end{bmatrix}\begin{Bmatrix}\{\boldsymbol{z}_C\}\\ \{\boldsymbol{z}_F\}\end{Bmatrix}\\ &= \begin{Bmatrix}[A_{CC}]\{\boldsymbol{z}_C\}\\ [A_{FC}]\{\boldsymbol{z}_C\}\end{Bmatrix}\end{aligned} \tag{22}$$

Then, we perform the p sub-steps by solving the following system for $\tilde{\boldsymbol{z}}(\tau)$ between $\tau = 0$ and $\tau = \Delta t$ by the leap-frog method:

$$\begin{cases}\dfrac{d^2\tilde{\boldsymbol{z}}}{d\tau^2}(\tau) = -(\boldsymbol{w} + [A][P]\tilde{\boldsymbol{z}}(\tau)) & \tau \in [0, \Delta t]\\ \tilde{\boldsymbol{z}}(0) = \boldsymbol{z}(t) & \Rightarrow \quad \tilde{\boldsymbol{z}}_{0/p} = \boldsymbol{z}_n\\ \dfrac{d\tilde{\boldsymbol{z}}}{d\tau}(0) = 0 & \Rightarrow \quad \tilde{\boldsymbol{z}}_{-1/p} = \tilde{\boldsymbol{z}}_{1/p}\end{cases} \tag{23}$$

The first sub-step (indexed $1/p$) is different from the next ones because $\tilde{\boldsymbol{z}}_{-1/p}$ should be deduced from the initial conditions:

$$\boxed{\tilde{\boldsymbol{z}}_{0/p} = \boldsymbol{z}_n} \tag{24}$$

$$\tilde{\boldsymbol{z}}_{1/p} - 2\tilde{\boldsymbol{z}}_{0/p} + \tilde{\boldsymbol{z}}_{-1/p} = -\left(\frac{\Delta t}{p}\right)^2(\boldsymbol{w} + [A][P]\tilde{\boldsymbol{z}}_{0/p}) \tag{25}$$

$$\boxed{\tilde{\boldsymbol{z}}_{1/p} = \tilde{\boldsymbol{z}}_{0/p} - \frac{1}{2}\left(\frac{\Delta t}{p}\right)^2(\boldsymbol{w} + [A][P]\tilde{\boldsymbol{z}}_{0/p})} \tag{26}$$

For the next sub-steps $m = 1, \ldots, p-1$, we can use the traditional leap-frog update formula:

$$\tilde{\boldsymbol{z}}_{(m+1)/p} - 2\tilde{\boldsymbol{z}}_{m/p} + \tilde{\boldsymbol{z}}_{(m-1)/p} = -\left(\frac{\Delta t}{p}\right)^2(\boldsymbol{w} + [A][P]\tilde{\boldsymbol{z}}_{m/p}) \tag{27}$$

$$\boxed{\tilde{\boldsymbol{z}}_{(m+1)/p} = 2\tilde{\boldsymbol{z}}_{m/p} - \tilde{\boldsymbol{z}}_{(m-1)/p} - \left(\frac{\Delta t}{p}\right)^2(\boldsymbol{w} + [A][P]\tilde{\boldsymbol{z}}_{m/p})} \tag{28}$$

Note that the multiplications $[A][P]\tilde{\boldsymbol{z}}$ involve a small part of $[A]$ if there are just a few small elements in the mesh and they are thus inexpensive compared to the calculation of $\boldsymbol{w}$:

$$\begin{aligned}[A][P]\tilde{\boldsymbol{z}} &= \begin{bmatrix}[A_{CC}] & [A_{CF}]\\ [A_{FC}] & [A_{FF}]\end{bmatrix}\begin{bmatrix}[0] & [0]\\ [0] & [I]\end{bmatrix}\begin{Bmatrix}\{\tilde{\boldsymbol{z}}_C\}\\ \{\tilde{\boldsymbol{z}}_F\}\end{Bmatrix}\\ &= \begin{bmatrix}[0] & [A_{CF}]\\ [0] & [A_{FF}]\end{bmatrix}\begin{Bmatrix}\{\tilde{\boldsymbol{z}}_C\}\\ \{\tilde{\boldsymbol{z}}_F\}\end{Bmatrix}\\ &= \begin{Bmatrix}[A_{CF}]\{\tilde{\boldsymbol{z}}_F\}\\ [A_{FF}]\{\tilde{\boldsymbol{z}}_F\}\end{Bmatrix}\end{aligned} \tag{29}$$

When $\tau = \Delta t$ is reached, after p substeps, we can calculate $\boldsymbol{z}_{n+1}$ from the last value of $\tilde{\boldsymbol{z}}$:

$$\boxed{\boldsymbol{z}_{n+1} = -\boldsymbol{z}_{n-1} + 2\tilde{\boldsymbol{z}}_{p/p}} \tag{30}$$

2.3 Questions

We consider the boundary value problem defined by eq. 1 with $L = 4$, $T = 9$, $c = -1$. Deduce the initial conditions in order to get a solution which corresponds to a the following Gaussian moving to the left at constant velocity c at the very beginning of the simulation:

$$u(x,t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - ct - \frac{L}{2})^2}{2\sigma^2}\right) \tag{31}$$

where $\sigma = 0.4$. Note that this Gaussian will later be reflected by the boundary and bounce back and forth.

1. Use a von Neumann analysis on the leap-frog scheme where the spatial derivatives are discretized with a finite difference and prove the stability condition expressed by eq. 7. The scheme writes:

$$\frac{u_j^{n-1} - 2u_j^n + u_j^{n+1}}{\Delta t^2} = c^2 \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{h^2} \tag{32}$$

2. Implement the finite element method as described above, and a classical leap-frog time-marching scheme. Use a regular mesh with $h = 0.1$ and verify the stability condition expressed by eq. 7.
3. Then, define a regular mesh with $h = h_{\text{coarse}} = 0.1$ and add a local refinement inside $[1, 1 + 2h]$ (divide each of these two elements in p smaller elements). Make some numerical experiments to verify that the stability condition involves now $h_{\text{fine}} = h_{\text{coarse}}/p$ instead of h_{coarse}.
4. Implement the Local-Time-Stepping Leap-Frog scheme (LTS-LF) and solve the problem on the locally-refined mesh using the largest time step allowed by the stability condition calculated with h_{coarse}. Use sparse matrices for sub-matrices $[A_{CC}]$, $[A_{CF}]$, $[A_{FC}]$ and $[A_{FF}]$. Note that the $[P]$ matrix should not be built explicitly. Instead, you should work directly with "coarse" and "fine" vectors such as $\tilde{\boldsymbol{z}}_C$ and $\tilde{\boldsymbol{z}}_F$.

References

[1] Diaz J, Grote MJ. Energy conserving explicit local time stepping for second-order wave equations. SIAM Journal on Scientific Computing. 2009;31(3):1985–2014.