Internship report
from February, 18 to July, 31 2014,
at the *California Institute of Technology*,
1200 E California Blvd, Pasadena, CA 91125, United States.

Numerical instabilities in hyperbolic system solvers

Author:
Hugo Lavenant

Supervisors:
Dr. Oscar P. Bruno
Dr. Edwin N. Jimenez

École Normale Supérieure

California Institute of Technology

November 23, 2014

Abstract

The numerical resolution of hyperbolic systems with *Fourier Continuaton* or high order finite difference methods on uniform grids and with *explicit* time marching leads to instabilities due to the enforcement of boundary conditions. In this report, we give a theoretical understanding of these instabilities and present a local filtering method that enables us to recover stability on general multi-dimensional domains with a minimal loss of accuracy.

Contents

1 Introduction **1**

2 The Fourier Continuation method **3**
2.1 General Principle 3
2.2 The Gram Fourier Continuation 3

3 Instabilities in the 1D case **4**
3.1 Equation and discretization 4
3.2 Stability analysis 5
3.3 Singular value decomposition of D_1 6
3.4 Importance of the enforcement of the boundary conditions 8

4 The finite difference local filter **8**
4.1 The problem of the FC filter 10
4.2 The finite difference filter 11
4.2.1 The 1D case 11
4.2.2 The 2D case 12
4.3 Making the filter local 12
4.4 Results in 1D 13
4.5 Results in 2D 14
4.6 Multi-domain in 2D 15
4.7 Finite difference differentiation operator 17

5 Conclusions and further work **17**

A Gradient and divergence in the local basis **17**
A.1 Gradient 17
A.2 Divergence 17

1 Introduction

The goal of this study was to understand and to remove numerical instabilities that arise when solving numerically hyperbolic systems with explicit time marching and *Fourier Continuation* (that we will denote by FC and present in section 2) to compute spatial derivatives.

These instabilities arise with many different hyperbolic PDEs (the typical examples are the ones in electromagnetic), but we will concentrate on the most simple hyperbolic system: the wave system in d dimensions which can be written as

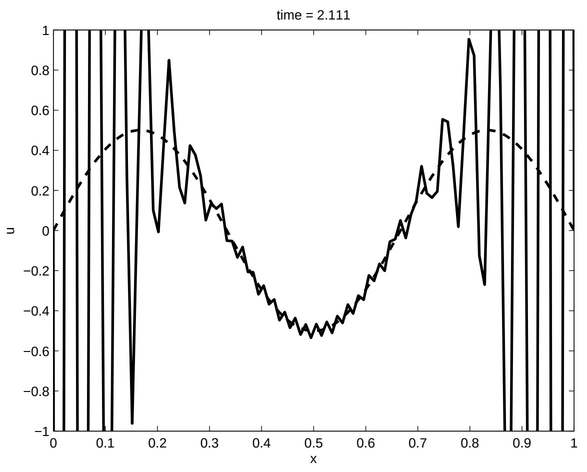


Figure 1: Numerical (solid line) and exact (dashed line) solution of the equation (1) in 1D (only u is plotted) at time $t = 2.111$ with explicit multi-step time marching and FC.

$$\begin{cases} u_t = \nabla.\mathbf{v} \text{ in } \Omega \\ \mathbf{v}_t = \nabla u \text{ in } \Omega \\ u = 0 \text{ on } \partial\Omega \end{cases}, \tag{1}$$

where Ω is a bounded domain in $\mathbf{R}^d$ with a smooth boundary $\partial\Omega$, u, defined over $\mathbf{R}_+ \times \Omega$ is a scalar function and $\mathbf{v} : \mathbf{R}_+ \times \Omega \to \mathbf{R}^d$ is a vector field.

Given any smooth initial condition $(u_0, \mathbf{v}_0)$ such that u_0 vanishes on $\partial\Omega$, there exists a unique solution to (1). Moreover, u satisfies the wave equation

$$u_{tt} = \Delta u, \tag{2}$$

an both u and $\mathbf{v}$ are uniformly bounded over time in $L^2(\Omega)$.

One can easily try to solve (1) numerically using explicit time marching method and FC (or high order[1] finite difference formula) to compute the spatial derivatives, but it leads to an unstable scheme. Growing oscillations arise in the numerical solution (see figure 1) and prevent one to solve (1) with high order accuracy. That's why one wants a way understand these instabilities, and remove them while keeping the high order accuracy and the excellent dispersion relation provided by FC.

We restricted our self to *explicit* time marching method (in order to avoid the costly inversion of a linear system), and more precisely the *Adam Bashforth of order four* (AB4). As far as the spatial discretization is concerned, the only constraint was to use a *uniform* grid, in order to avoid a too strict CFL condition.

This report is organized as follows : we present briefly the FC method, then introduce the spectral analysis in the 1D case that gives an understanding of the instabilities. This is followed by the technique we developed to fix the problem: the application of a finite difference filter that is localized only near the boundaries.

[1]of order greater or equal to four.

2 The Fourier Continuation method

2.1 General Principle

If one has a smooth periodic function (or more precisely the sampled values of a smooth periodic function), the Discrete Fourier Transform (DFT) can be used to compute quickly the derivatives of the function with spectral accuracy and no dispersion. However, this method fails if the function is not periodic: the Gibbs phenomenon occurs near the boundaries, damaging severely the accuracy. To recover the high order accuracy when working with a non periodic function, one can extend it on a larger interval in such a way that the extended function is smooth and periodic, and then use all the tools available for periodic functions. The article [1] gives a description of the many ways to extend a non periodic function, with the theoretical and numerical questions raised by this problem.

2.2 The Gram Fourier Continuation

We will present briefly the *Fourier Continuation* (FC) developed by O. Bruno, a complete description can be found in [3, 5]. Let us assume that we have a smooth non periodic function f, defined on $[0,1]$ (given by its values $(F_j)_{1\leq j\leq m} = \left(f\left(\frac{j-1}{m-1}\right)\right)_{1\leq j\leq m}$ on a equidistant grid), that we want to extend on a larger interval $[0,a]$, with $a > 1$. The extension is computed by using only the values at n_Δ points near the boundaries (meaning $(F_j)_{1\leq j\leq n_\Delta}$ and $(F_j)_{m-n_\Delta+1\leq j\leq m}$). An accurate extension of the two polynomials (one for each boundary) that interpolate f at these n_Δ points near the boundaries is computed thanks to the SPIC-FU described in [1, section 4.1]. We take this extension as the extension of the original function (see figure 2).

As the interpolation is n_Δ order accurate and the SPIC-FU extension is spectrally accurate, the global extension is of order n_Δ (meaning that the extended function will be n_Δ times differentiable as long as this is also the case for the original function). Moreover, one can precompute and store the extensions of all the polynomials of degree less than $n_\Delta - 1$ (by linearity, only n_Δ extensions need to be computed[2]) so that the whole extension process will have a constant cost (independent of m the total number of points in the original interval)

Using these ideas, the algorithm to compute the derivative of a given function can be described as follows :

1. `Compute the two polynomials that interpolate` f `near the boundaries.`
2. `Use the precomputed extension of these polynomials to get the function in the extended region` $[1,a]$.
3. `Compute the derivative of the whole periodic function (defined on` $[0,a]$`) using the FFT.`
4. `Restrict the derivative of the extended function to the original interval` $[0,1]$.

The whole process gives an approximation of order n_Δ and the number of operations needed is $O(m \log m)$ (the bottle neck is the computation of the FFT, all other computations have a constant or linear cost), where m is the number of discretization points. Moreover, this method has very few dispersion compared to high order finite difference formula.

In practice, if n_Δ is too large, the Runge phenomenon can occur when interpolating the function leading to instabilities, so we used only $n_\Delta = 5$. Moreover, the parameter a was chosen in such a way there were exactly 25 points in the continuation region, keeping the step size equal to the one in the original interval $[0,1]$.

[2]The extensions are computed only for the polynomials of the Gram basis.

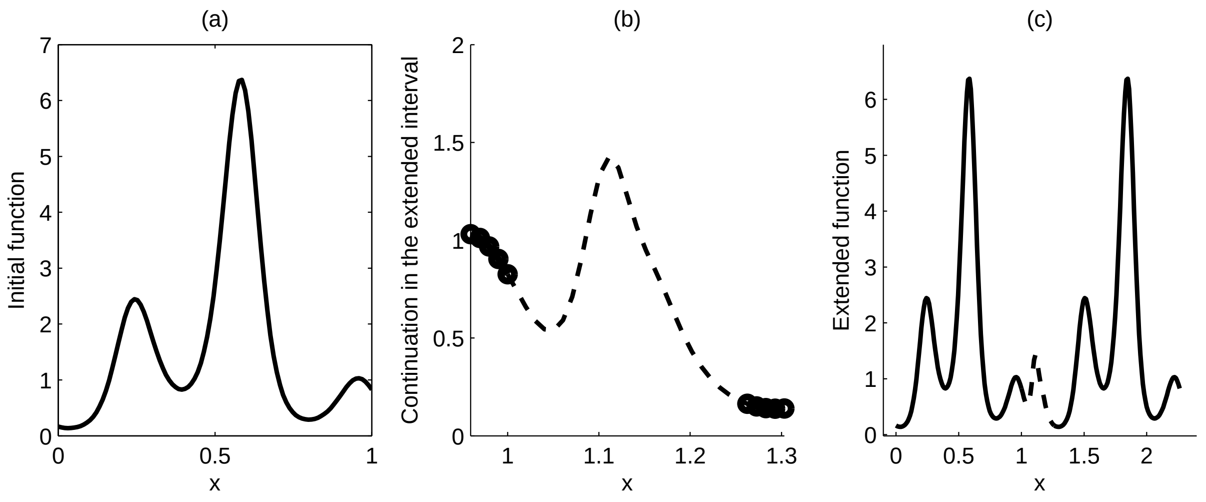


Figure 2: (a) Initial smooth non periodic function (b) Continuation of the function in the extended interval (dashed line) using only n_Δ near boundaries values (represented by circles) (c) the continued function is periodic, the extension being in dashed line.

3 Instabilities in the 1D case

3.1 Equation and discretization

Taking $d = 1$ in (1), we come up with the following PDE system:

$$\begin{cases} u_t = v_x \\ v_t = u_x \\ u(t,0) = u(t,1) = 0 \ \forall t \geq 0 \end{cases}, \tag{3}$$

where $(t, x) \in \mathbf{R} \times [0, 1]$ and u, v are scalar function. In this case, both u and v will satisfy the wave equation :

$$u_{tt} = u_{xx} \text{ and } v_{tt} = v_{xx}. \tag{4}$$

Let us give some notations about the way we discretize (4): the spatial discretization is done on a uniform grid, the total number of points in the grid will be denoted by m, and the discretization points will be the $x_j = \frac{j-1}{m-1}$ for $1 \leq j \leq m$. If w is a function defined over $[0, 1]$, then we will note $W = (w(x_j))_{1 \leq j \leq m} \in \mathbf{R}^m$ the vector containing the sampled values of w[3]. D_1 will denote an approximation of the first differentiation operator: it is a linear operator acting on $\mathbf{R}^m$ such that $D_1 W$ is an approximation of w'.

The time is sampled with a constant time step dt, so we will denote $n\mathrm{d}t \equiv t^n$, and superscript indices will correspond to time: the quantity U_j^n will be an approximation of $u(x_j, t^n)$. The coefficients of the AB4 scheme will be denoted by a_0, a_1, a_2 and a_3. Eventually, the discrete operator enforcing the boundary conditions will be denoted by B : $B(U_1, U_2, \ldots U_m) = (0, U_2, \ldots, U_{m-1}, 0)$. A fully discretized version of (3) can therefore be written as

[3]More generally, lower case letters stand for functions and upper case letters for vectors containing sampled values of functions.

$$\begin{cases} U^{n+1} = B\left(U^n + \mathrm{d}tD_1\left(a_0V^n + a_1V^{n-1} + a_2V^{n-2} + ab_3V^{n-3}\right)\right) \\ V^{n+1} = V^n + \mathrm{d}tD_1\left(a_0U^n + a_1U^{n-1} + a_2U^{n-2} + a_3U^{n-3}\right) \end{cases} . \tag{5}$$

3.2 Stability analysis

As long as the operator D_1 is consistent, the overall scheme will be consistent. Therefore to prove the convergence the stability is a sufficient and necessary condition. We recall that the scheme is stable if and only if, for every initial condition, the sequences $(U^n)_{n\geq 0}$ and $(V^n)_{n\geq 0}$ are bounded. A simple analysis is to write (5) as a linear first order recurrence relation on $\mathbf{R}^{8m}$, and then to study the spectrum of the $8m \times 8m$ matrix associated to this recurrence relation. However, following a discrete analogue of the derivation of the scalar wave equation (4) from the system (3), and using the fact that $B^2 = B$, we can show that $(U^n)_{n\geq 0}$ satisfies an eight order recurrence relation:

$$\frac{U^{n+1} + U^{n-1} - 2U^n}{\mathrm{d}t^2} = BD_1^2\Big(a_0^2U^{n-1} + 2a_0a_1U^{n-2} + (a_1^2 + 2a_0a_2)U^{n-3} + (2a_0a_3 + 2a_1a_2)U^{n-4} + (2a_1a_3 + a_2^2)U^{n-5} + 2a_2a_3U^{n-6} + a_3^2U^{n-7}\Big). \tag{6}$$

The left hand side of (6) is an approximation of u_{tt} and the right hand side of u_{xx}. To solve this equation, we project it on the eigenvectors of BD_1^2, so we are left with m dis coupled scalar eight order linear recurrence relations. And a sufficient condition for a scalar eight order linear recurrence relation to be stable is that all the roots of the characteristic polynomial are of modulus less than 1.

Therefore the first step is to compute the eigenvalues of BD_1^2, and then, for every eigenvalue λ, we have to check whether the roots of the polynomial

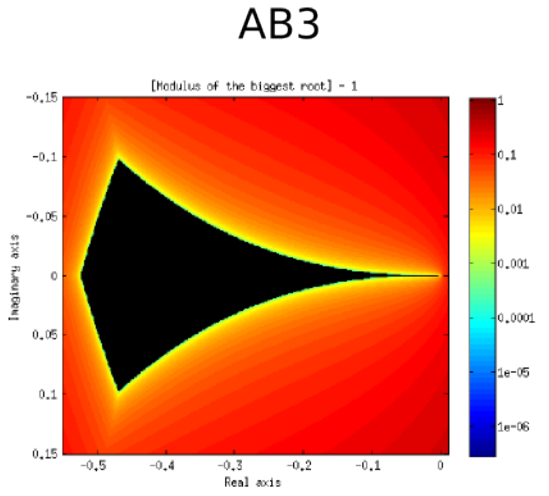
$$X^8 + X^6 - 2X^7 = \mathrm{d}t^2\lambda\Big(a_0^2X^6 + 2a_0a_1X^5 + (a_1^2 + 2a_0a_2)X^4 + (2a_0a_3 + 2a_1a_2)X^3 + (2a_1a_3 + a_2^2)X^2 + 2a_2a_3X + a_3^2\Big) \tag{7}$$

are of modulus less than one. The set of all $z = \mathrm{d}t^2\lambda$ such that the latter property holds is called the *stability region* of the scheme. The stability regions of both the AB4 and AB3 method are plotted in figure 3. We can see that they are very closed to the real negative axis, which is the stability region of the wave scalar equation (4):

$$u_{tt} = zu. \tag{8}$$

Indeed, one can check that the solutions of (8) are bounded if and only if z is real and negative. The tininess of the stability region explains why discrete hyperbolic systems are likely to be unstable. Indeed, in the continuous case, the fact that the eigenvalues of the laplacian are real and negative comes from the fact that this operator is self-adjoint and negative. But for a discrete operator, even if the eigenvalues are close to the one of the laplacian, it does not mean that they lie in the stability region. Moreover, it tells us that the choice of the time marching method (as long as it is explicit) does not affect stability: in any case the stability region will be close to the stability region of (8), and an eigenvalue outside of the stability region for one method also lies outside for another.

The equation (6) also reveals the difference between solving the system (3) and the scalar equation (4). In the second case, we can directly use an approximation of the second derivative operator D_2



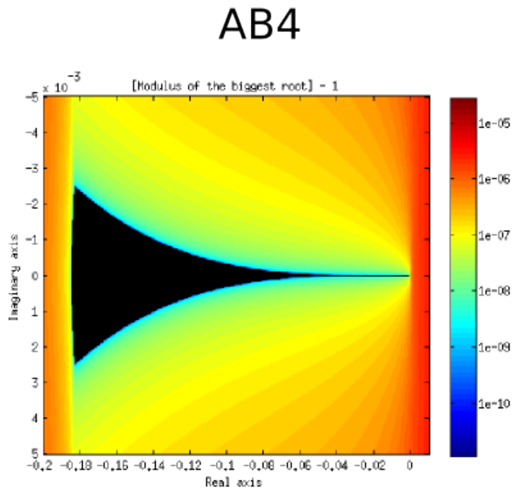


Figure 3: Difference between the biggest modulus of the roots of the characteristic polynomial and 1 for AB3 and AB4. (when this difference is negative it is in black). Watch out the different scales on the real and imaginary axis.

(either with FC or finite difference), whereas in the first case, the second derivative is approximated by the discrete square of the discrete differentiation operator D_1.

Figures 4 and 5 show the spectrum of BD_1^2 and BD_2 (the one that matters if we solve (4) numerically). The difference is striking, we can clearly see in figure 4 two eigenvalues that are outside of the stability region, eigenvalues that we do not see in the spectrum associated to the scalar wave equation.

Moreover, we can plot (figure 6) the unstable modes of the equation (5), more precisely the eigenvectors associated to the eigenvalues that lie outside the stability region. At each time step, the component of these modes will be multiplied by a factor of modulus strictly greater than one, factor that we can compute by looking at the roots of (7).

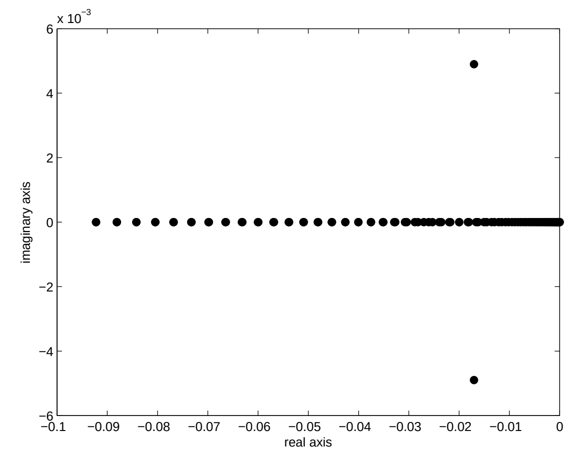
To check these results, we present in figure 7 the error (expressed in the infinity norm) committed when solving the wave equation with FC, the initial condition being $u(x, 0) = \sin(\pi x)$ and $v(x, 0) = 0$ (an analytic solution is known for this initial condition). Before $t = 1$, the component on the unstable modes is smaller than truncation error so it cannot be seen, but then we clearly see an exponential growth of the error which is due to the exponential growth of the component on the unstable modes. The ”experimental” exponential rate can be computed in figure 7 and it matches the ”theoretical” value computed thanks to the equation (7) and the spectrum of BD_1^2 (figure 4).

3.3 Singular value decomposition of D_1

Basically, to enforce stability, we have to modify either B or D_1 in such a way that the spectrum of BD_1^2 have only real and negative eigenvalues. The problem is that the way the spectrum is modified when we change D_1 is not obvious.

However, it seems to have a link with the *Singular Value Decomposition* (SVD) of D_1. Indeed, figure 8 presents the SVD of D_1, the presence of two large (compare to the other ones) singular values is striking. Moreover, figure 9 which presents one of the singular vector associated to these two large singular values is very similar to the one of figure 6.

We removed these two large singular values (by changing the diagonal matrix in the SVD), it gave us a matrix $\hat{D}_1$ that happened to produce a stable scheme when we replaced D_1 by $\hat{D}_1$ in the



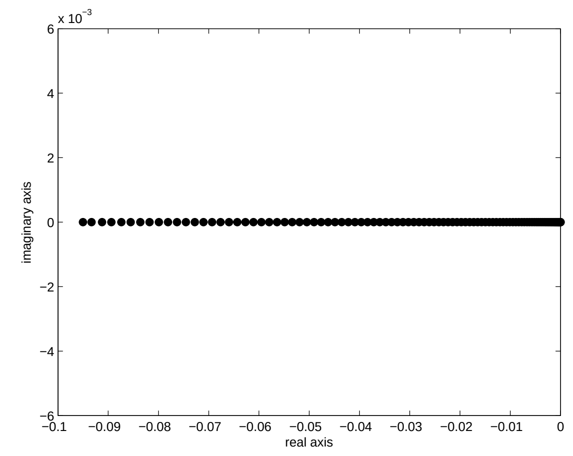


Figure 4: Spectrum of the matrix $\mathrm{d}t^2 B D_1^2$ with $\mathrm{d}t = 10^{-3}$, and D_1 the FC differentiation operator with $m = 100$ points.

Figure 5: Spectrum of the matrix $\mathrm{d}t^2 B D_2$ with $\mathrm{d}t = 10^{-3}$, and D_2 the FC second differentiation operator with $m = 100$ points.

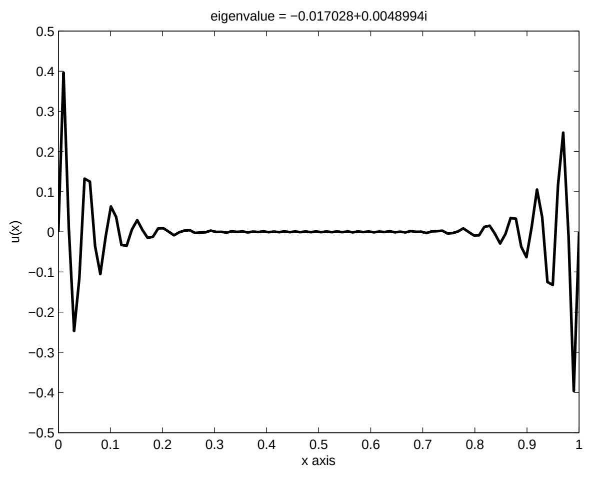


Figure 6: Eigenvector for one of the two complex eigenvalues of the figure 4. Only the real part is plotted.

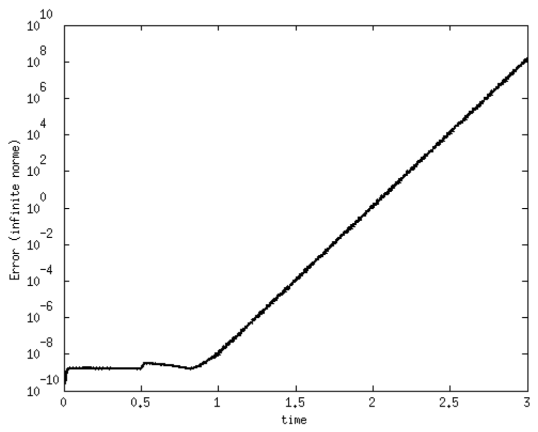


Figure 7: Error (in infinity norm) versus time for the 1D wave equation system, with FC, $m = 100$ points in space, and $\mathrm{d}t = 10^{-3}$, AB4. The initial condition is $u(x, 0) = \sin(\pi x)$ and $v(x, 0) = 0$.

recurrence relation (5): figure 10 shows the spectrum of $B\hat{D}_1$, all the eigenvalues lie in the stability region. However, to remove these large singular values damages the accuracy, the scheme is no longer consistent. Nevertheless it tells us that these large singular values may have a link with the complex eigenvalues.

The interpretation that we give is that these large singular values shows that the FC differentiation operator amplifies errors much more near the boundaries than in the center. Therefore the little discontinuities introduced when enforcing the boundary conditions are strongly amplified over time, leading to the huge oscillations.

3.4 Importance of the enforcement of the boundary conditions

To "prove" that the instabilities are coming from the enforcement of the boundary conditions, we cheated when enforcing them. In the particular case when we had an analytic expression of the exact solution (for a standing wave for example), we enforced the boundary conditions by assigning the exact value $u(x_j, t^n)$ to U_j^n not only for $j = 1$ and m (because in this case $u(x_j, t^n) = 0$) but for $j \leq p$ and $j \geq m-p+1$ for a given $p \geq 1$ (so the case $p = 1$ being the "classic" enforcement of the boundary conditions.

This way to cheat by using the exact solution in a small neighborhood of the boundaries was a way to "remove" the boundaries. And indeed, if $p \geq 2$, the scheme is stable : the instabilities are really coming from the discontinuity introduced when enforcing the boundary conditions and amplified by the inaccuracy of FC near the boundaries.

4 The finite difference local filter

We present a method that enables to fix the stability issue with a minimal loss of accuracy. This method is based on a filtering process, meaning that at each iteration, we apply a filter to the solution that smooths it and removes the high oscillatory modes. This filter is applied in such a way that it affects only the region near the boundary, so that the accuracy and the excellent dispersion relation of the FC differentiation operator are fully preserved in the middle of the domain Ω.

We first present the reasons that led us to abandon the FC filter that was used before and to switch to a finite difference filter, and then present how we apply it locally. Eventually, we present another

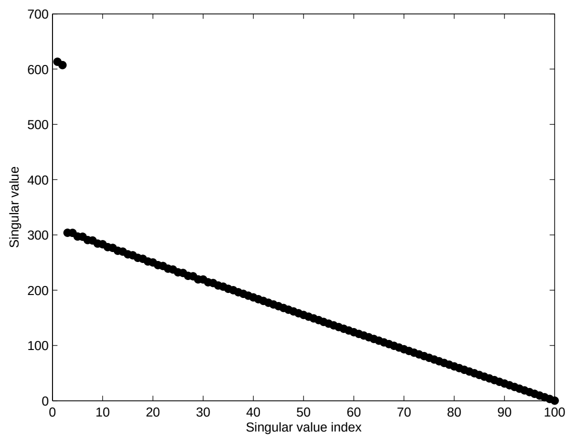


Figure 8: Singular values of the FC differentiation operator with $m = 100$ points.

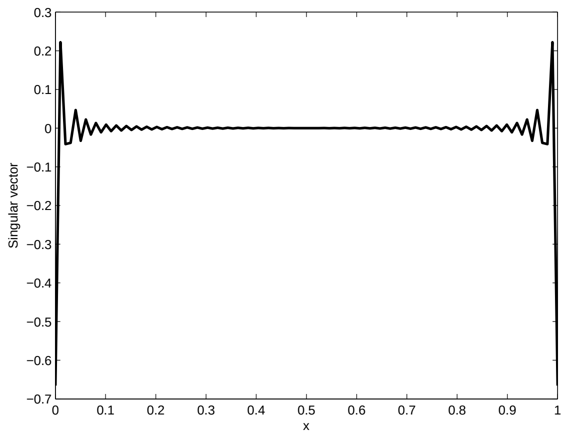


Figure 9: Right singular vector associated to the largest singular value of the FC differentiation operator, $m = 100$ points.

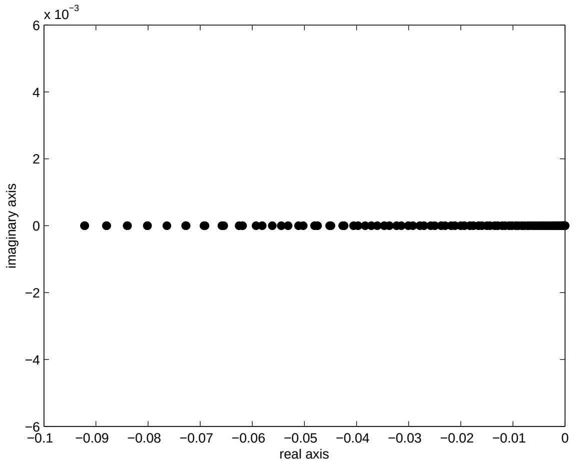


Figure 10: Spectrum of the matrix $\mathrm{d}t^2 B\hat{D}_1^2$ with $\mathrm{d}t = 10^{-3}$ and $m = 100$ points. $\hat{D}_1$ is obtained by setting the two largest singular values of D_1 to 0.

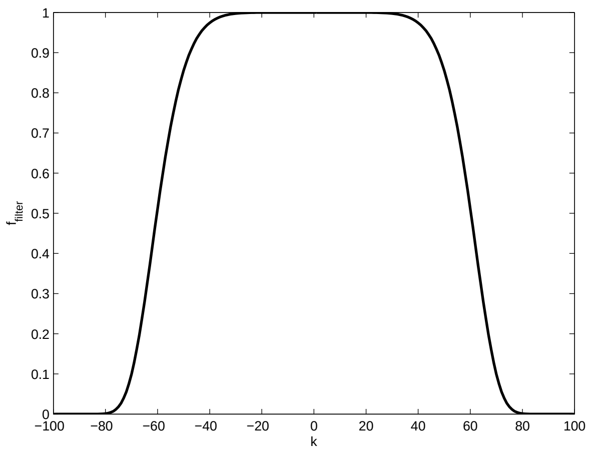


Figure 11: f_{filter} for $\alpha = 40$, $p = 4$ and $k_{\text{max}} = 100$.

condition that was necessary to enforce stability in dimension $d = 2$, mainly the basis in which to express $\mathbf{v}$. Numerical experiments in one and two dimensions are shown, the generalization to $d = 3$ being straightforward.

4.1 The problem of the FC filter

One very efficient way to filter a *periodic* function is to decompose the function as a Fourier series, and then to multiply the Fourier coefficients by a filter function that is nearly equal to 1 for the small wave numbers and very small for the large wave numbers. The filter function used was of the form

$$f_{\text{filter}}(k) = \exp\left(-\alpha\left(\frac{k}{k_{\text{max}}}\right)^{2p}\right), \tag{9}$$

where k_{max} is the maximal wave number (equal to the half of the number of discretization point) and α, p are two free parameters to adjust the width and the sharpness of the window (typical values were $\alpha = 40$ and $p = 4$, see figure 11). For a smooth function, the coefficients on the large wave numbers are very small, so that the filtered function is very close to the original one. One can even prove that if p is large enough, the order of numerical scheme can be preserved even if the function is filtered at every time step.

Following this, to filter a non periodic function, one can extend this function using FC, filter with the above method the extended periodic function and then restrict the filtered function to the physical interval. However, it happens that this method suffers from a great problem: the continuation process is ill conditioned, so the continuation of a very oscillatory function will be very large (in infinity norm), and when filtered, the huge peaks in the continuation region will spread out in the physical interval, making things worst near the boundaries for the filtered function rather than the unfiltered one (see figure 12). Numerically, with a very strong filter (meaning α large and p small) the scheme was more unstable compared to no filter at all.

That's why we abandoned this filtering method and we switched to a finite difference one to deal with non periodic functions.

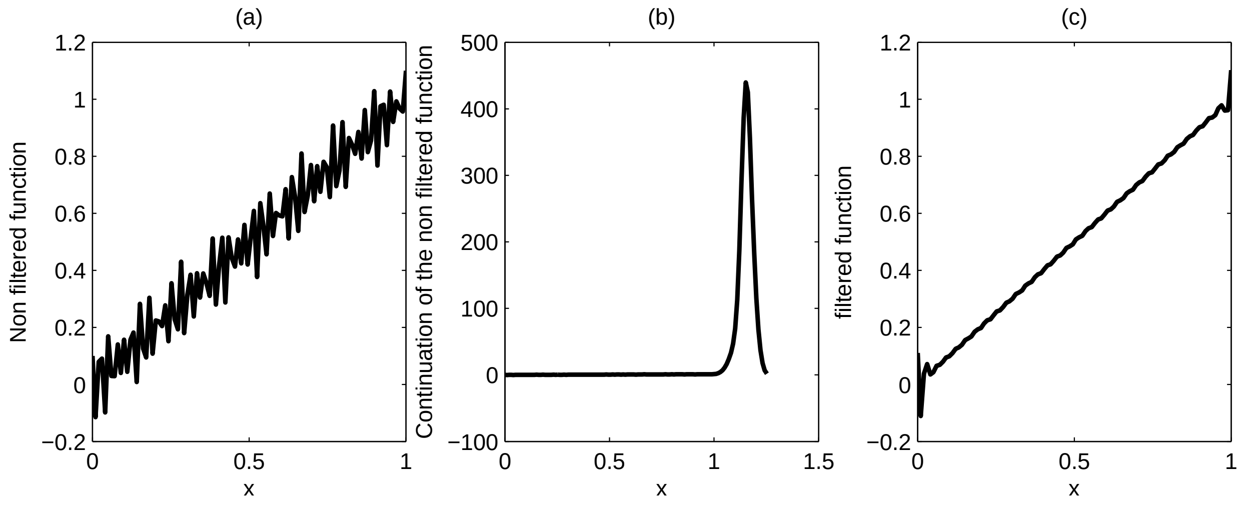


Figure 12: (a) Function with high oscillatory modes (b) Its continuation, where we notice a huge peak in the continuation region (c) the FC-filtered function, where oscillations have been eliminated everywhere except near the boundaries because of the peak in the extension region.

4.2 The finite difference filter

4.2.1 The 1D case

The finite difference filter presented in this section comes from [4]. Basically, the value of the filtered function at one point will be a local weighted average of the values of the function in the neighborhood of this point. Two parameters can be chosen : the first one is the order $2p$ of the filter (the filtered function converges to the original one at the order $2p$ as we increase the number of discretization points), that adjusts the number of points used in the average of the function, and a parameter α_f on which will depend the weights of the average. The bigger this parameter is, the weaker the filter is (see [4] for more details). We only write here the way the filtered function is computed: if $U = (U_i)_{1\leq i\leq m}$ is the unfiltered function, then the filtered function $W = (W_i)_{1\leq i\leq m}$ satisfies

$$\alpha_f W_{i-1} + W_i + \alpha_f W_{i+1} = \sum_{j=0}^{p} \frac{1}{2}\left(a_j U_{i+j} + U_{i-j}\right) \text{ for } p+1 \leq i \leq m-p, \tag{10a}$$

$$\alpha_f W_{i-1} + W_i + \alpha_f W_{i+1} = \sum_{j=1}^{2p+1} b_{i,j} U_j \text{ for } 2 \leq i \leq p, \tag{10b}$$

$$\alpha_f W_{i-1} + W_i + \alpha_f W_{i+1} = \sum_{j=1}^{2p+1} b_{i,j} U_{m-j+1} \text{ for } m-p+1 \leq i \leq m-1, \tag{10c}$$

$$W_1 = U_1, \tag{10d}$$

$$W_m = U_m, \tag{10e}$$

where the coefficients a_i and $b_{i,j}$ depend on α_f and are chosen in such a way that the method is of order $2p$. The equation (10a) holds in the middle of the interval, whereas equations (10b) and (10c) deal with the left and right boundaries respectively.

We can notice that the computation of W requires to solve a tridiagonal linear system, that can be done in $O(m)$ operations. We also note that the boundary values are left unchanged by the filter, that means that W will satisfy the same boundary conditions as U.

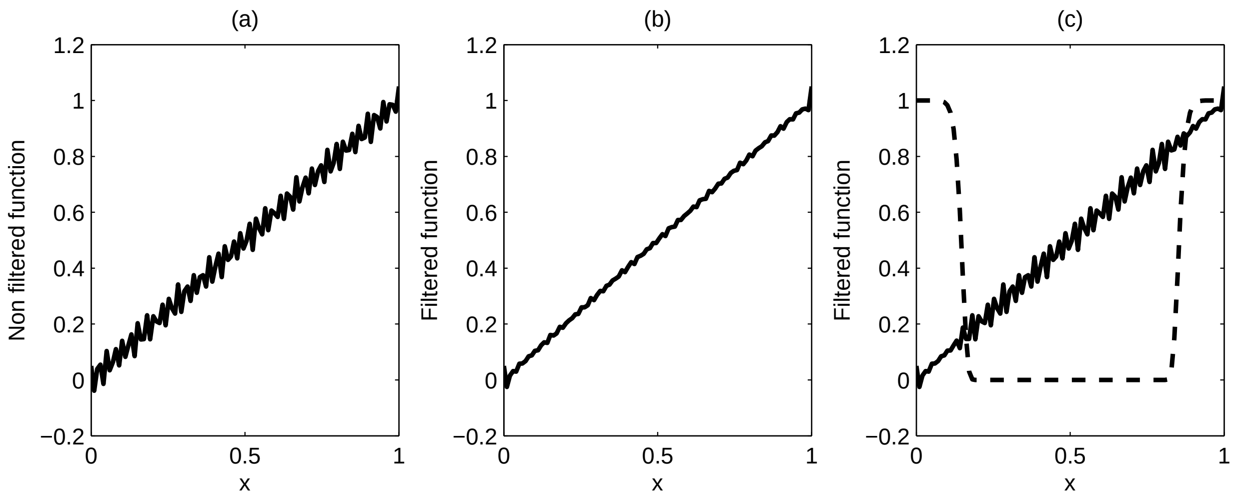


Figure 13: (a) Function with high oscillatory modes (b) the FD filtered function (c) the local FD filtered function (solid line) and the windowing function χ (dashed line), we can see that in this case the effects are located near the boundaries. Parameters of the filter are $p = 2$ and $\alpha_f = -0.3$.

4.2.2 The 2D case

In 2D we will assume that U is a 2D array $(U_{i,j})_{1\leq i\leq m_1,\ 1\leq j\leq m_2} \in \mathbf{R}^{m^2} = \mathbf{R}^m \otimes \mathbf{R}^m$. If we denote by F^1 the 1D finite difference filter, than the we can define the filter along the first and second direction F_1^2 and F_2^2 by

$$\forall 1 \leq j \leq m_2,\ ((F_1^2 U)_{i,j})_{1\leq i\leq m_1} = F^1(U_{i,j})_{1\leq i\leq m_1}, \tag{11}$$

and

$$\forall 1 \leq i \leq m_1,\ ((F_2^2 U)_{i,j})_{1\leq j\leq m_2} = F^1(U_{i,j})_{1\leq j\leq m_2}. \tag{12}$$

To apply the 2D filter $F2$ is to filter in both directions:

$$F^2 = F_1^2 \circ F_2^2 = F_2^2 \circ F_1^2, \tag{13}$$

it simply means that the 2D filter is the tensor product of the two 1D filters : $F^2 = F^1 \otimes F^1$.

4.3 Making the filter local

As the instabilities appear near the boundaries, we filter the function only there, in order to avoid to damage the accuracy of the numerical solution in the interior. More precisely, we can write the linear operator F corresponding at the filtering process as as $F = I + \hat{F}$. Then, after enforcing the boundary condition, we filter the numerical solution by assigning the following value to U :

$$U \leftarrow U + \chi \hat{F} U, \tag{14}$$

where χ is a smooth function, null in the interior and equal to 1 near the boundaries. Therefore, near the boundaries the filtered function FU is assigned to U, and the latter is left unchanged in the interior. The figure 13 compares the effect of the local and "classic" finite difference filter.

Then if we call F_χ the local finite difference filter, the whole scheme solving the wave system (1) can be written, for each time step as:

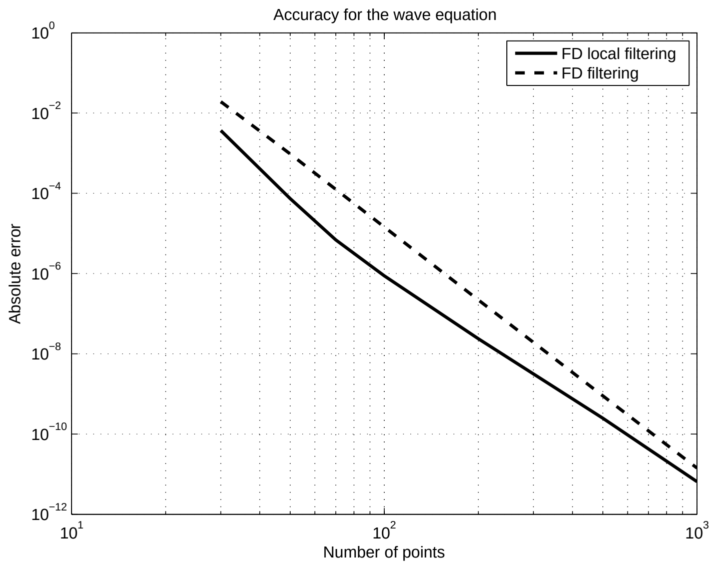


Figure 14: Error in infinity norm in space and time ($0 \leq t \leq 1$) for (3) solved with AB4, FC and local (solid line) and global (dashed line) FD filtering in function of the number of discretization points. We used the manufactured solution $u(t, x) = \sin(2\pi x)\cos(2\pi t)$ and $v(t, x) = \cos(2\pi x)\sin(2\pi t)$.

1. Compute U^{n+1} and $\mathbf{V}^{n+1}$ using AB4 and the FC differentiation operator.
2. Apply the boundary conditions to U^{n+1} by setting the value at the boundary points to 0.
3. Assign to U^{n+1} the value $F_\chi U^{n+1}$.

Indeed, numerical experiments showed that it is enough to filter only u and not $\mathbf{v}$ to enforce stability.

4.4 Results in 1D

To check for stability, the same kind of analysis as the one performed in 3.2 is not possible, (basically because the filtering operator is not a projection). That's why we checked the stability only numerically: we ran a simulation with random initial condition (in order to trigger all modes) on a long time (of the order of 10^7 time steps), and we checked if the numerical solution remain bounded. If yes, it does not necessarily mean that the scheme is rigorously stable, but it shows that instabilities will not be observed for every practical case.

With the filter of order 6, $\alpha_f = 0.3$, and a window of width 5 points, the numerical experiments showed that the scheme is stable. Moreover, to check for accuracy, we used a manufactured solution (a stationary wave), the convergence plot is presented in figure 14, exhibiting a 5th order convergence (the bottle neck being the order of the FC differentiation operator).

To show the advantage of the local FD filter compared to the global one, we ran the following experiment: we solved the wave system (3) starting with a very oscillatory wave presented in figure 15. We waited then one period during which the wave bounces against the boundaries and goes back to its initial value. We then computed the error (in infinity norm) between the initial condition and the numerical solution after one period, and we plotted the results in function of the number of discretization points, this is figure 16. The plateau at the beginning is due to the fact that with not

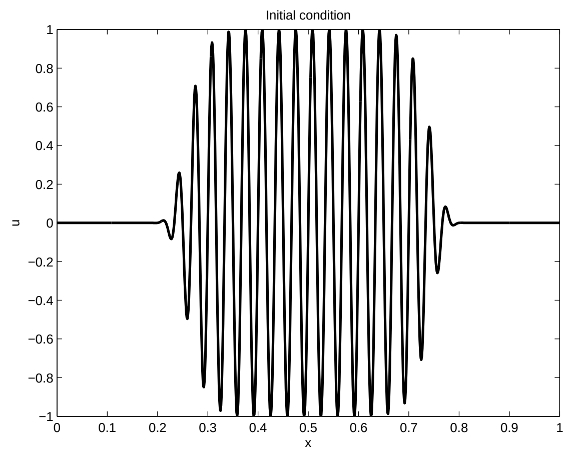


Figure 15: Initial condition u (very oscillatory wave with an envelope), $v = 0$.

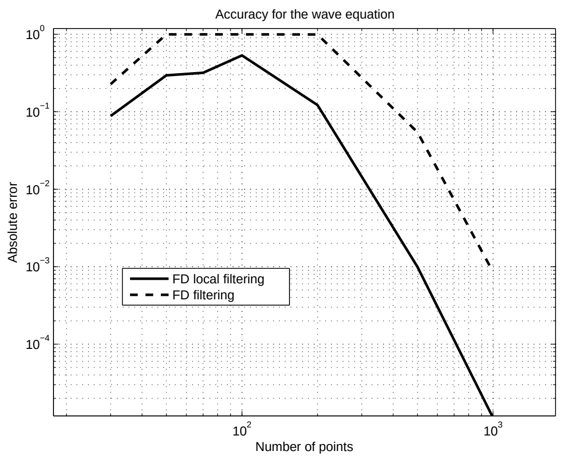


Figure 16: Error in infinity norm after one period with the initial condition given in figure 15 in function of the number of points for local (solid line) and global (dashed line) finite difference filter.

enough points, the filter will "kill" the oscillations in the initial solution. We can see with this figure that the local filter needs much less points to represent accurately a very oscillatory wave.

4.5 Results in 2D

In two dimensions, we assume that the domain Ω is parametrized by a square Γ which will either be $[0,1]^2$ or $\mathbf{R}/\mathbf{Z} \times [0,1]$ (in the second case we use periodic boundary conditions for the first variable). More precisely we assume to have a diffeomorphism

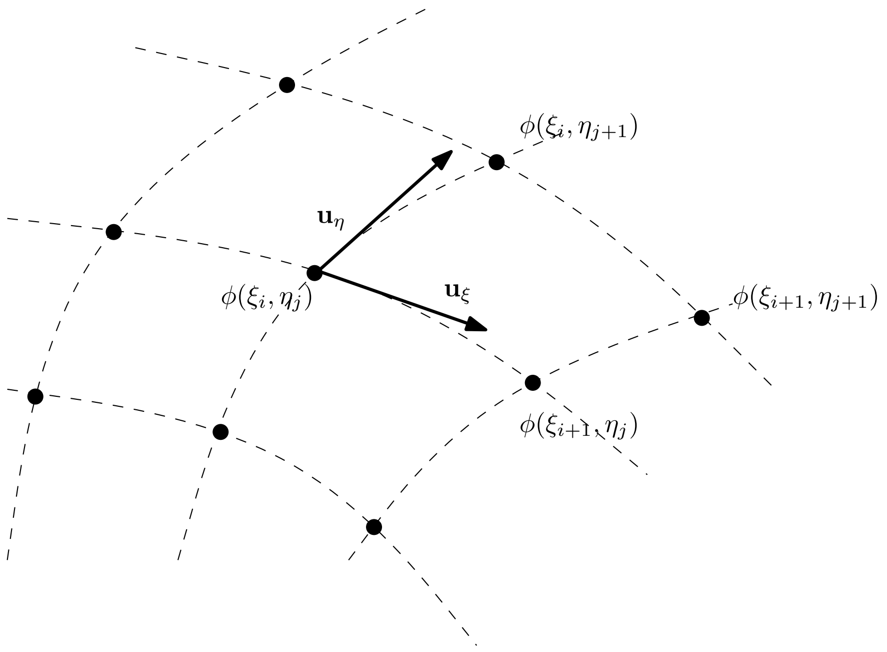
$$\begin{array}{cccc} \phi & : & \Gamma & \to & \Omega \\ & & (\xi,\eta) & \mapsto & (x(\xi,\eta), y(\xi,\eta)) \end{array} \tag{15}$$

which is numerically given by the values $(x_{i,j}, y_{i,j})$ of ϕ on a uniform discretization of Γ : with $\xi_i = \frac{i-1}{m_\xi - 1}$ for $1 \leq i \leq m_\xi$ and $\eta_j = \frac{j-1}{m_\eta - 1}$ for $1 \leq j \leq m_\eta$, then $(x_{i,j}, y_{i,j}) = \phi(\xi_i, \eta_j)$. Then instead of solving the wave system (1), we take $\tilde{u} = u \circ \phi$ and $\tilde{\mathbf{v}} = \mathbf{v} \circ \phi$ (defined over Γ) as unknown. These latter functions satisfy a system where the coefficients of the differential of ϕ^{-1} occur. But as we have the relation

$$\mathrm{d}(\phi^{-1}) = (\mathrm{d}\phi)^{-1}, \tag{16}$$

and that we can compute numerically the right hand side of (16), we end up with a way of computing without an analytic expression of ϕ the coefficients of the equation satisfied by $\tilde{u}$ and $\tilde{\mathbf{v}}$, and therefore solve (1) with only the knowledge of the grid points $(x_{i,j}, y_{i,j})$.

Let us now emphasize something that is crucial to get stability even though we don't really understand why until now: the choice of the basis on which to express $\mathbf{v}$. Indeed, in dimension 2 and more, $\mathbf{v}$ is a vector field and therefore we must choose a basis and take as unknowns the coordinates of $\mathbf{v}$ in this basis. Even though this choice doesn't matter for the consistency of the scheme, it is important as far as stability is concerned: it happens that the choice of two fix and orthogonal directions (meaning to work in Cartesian coordinates) to project $\mathbf{v}$ onto leads to an unstable scheme, even with a (global) filter.



$\phi(\xi_i, \eta_{j+1})$
$\mathbf{u}_\eta$
$\phi(\xi_i, \eta_j)$
$\mathbf{u}_\xi$
$\phi(\xi_{i+1}, \eta_{j+1})$
$\phi(\xi_{i+1}, \eta_j)$

Figure 17: Points of the grid $(x_{i,j}, y_{i,j})$ with the local basis $(\mathbf{u}_\xi, \mathbf{u}_\eta)$.

That's why we chose to work in the local basis $(\mathbf{u}_\xi, \mathbf{u}_\eta)$, whose vectors are unitary and aligned with $\partial_\xi \phi$ and $\partial_\eta \phi$, where ∂_ξ and ∂_η are respectively the differentiation w.r.t the first and second coordinates (see figure 17 for a graphic representation). The advantage of this local basis is that the expressions of the gradient and the divergence (given in annex A) lead directly to an hyperbolic system. As it is written above, we have seen numerically that to work in this local basis instead of the Cartesian one is necessary to enforce stability, but we don't have a theoretical understanding of this fact.

For the numerical experiments, and to avoid any instabilities related to the presence of corners in Ω, we take a domain (see figure 18 for examples) that can be parametrized by $\mathbf{R}/\mathbf{Z} \times [0, 1]$: one direction is periodic, so that all the boundaries locally look like the 1D boundary. In the periodic direction, we used the Fourier differentiation operator to get spectral accuracy.

To sum up, we do the following modifications compared to the naive discretization (5):

- filter U with the local finite difference filter,
- express the coordinates of $\mathbf{v}$ in the local basis,
- and take a domain periodic in one direction.

We got numerical stability (up to 10^7 time steps) for various domains like those of figure 18. Depending on the stiffness from the grid, we sometimes had to change the parameters of the filter.

Moreover, using a manufactured solution, we were able to check that this scheme is fifth order, see figure 19 for the convergence plot.

4.6 Multi-domain in 2D

Avoid corners is mainly justified by the use of a multi-domain code. Indeed, in order to use parallel computing, the domain is usually decomposed in different sub-domains with overlapping. Therefore the corners are in the overlapping regions and do not correspond to real boundaries.

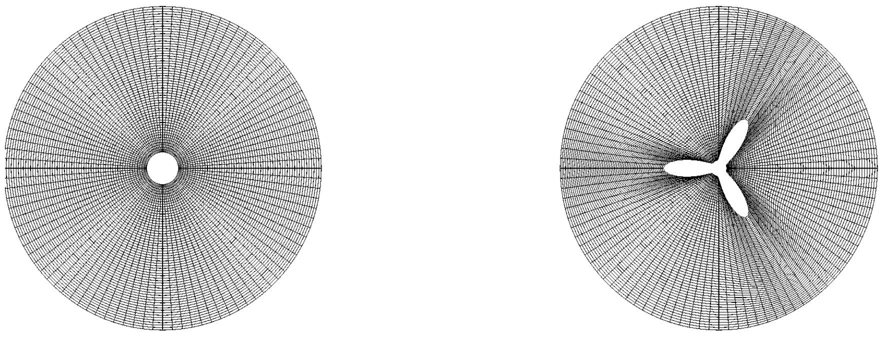


Figure 18: Two examples of domain diffeomorphic to $\mathbf{R}/\mathbf{Z} \times [0, 1]$ with their discretization.

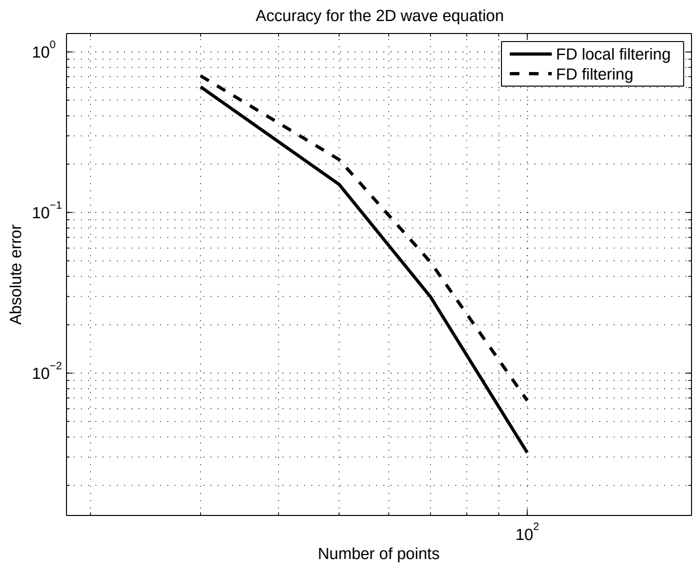


Figure 19: Error in infinity norm in space and time ($0 \leq t \leq 1$) for (3) solved with AB4, FC and local (solid line) or global (dashed line) FD filtering in function of the number of the discretization points (we took $m_\xi = m_\eta + 20$, where m_η is the number of points indicated on the plot). The domain Ω is an annulus.

Numerical experiments to gets stability with the multi-domain codes are still being performed, stability issues related to the way the solutions in different sub-domains are matched still occur.

4.7 Finite difference differentiation operator

We also used this local filtering method to stabilize the hyperbolic system solver with AB4 and a compact finite difference differentiation operator of order four (which lead to instabilities if no treatment is done).

The same stability results were obtained for 1D and 2D, and the order of convergence is four as the finite difference scheme is the bottle neck. Considering this, we strongly believe that the local filtering method can handle various high order differentiation operators.

5 Conclusions and further work

The theoretical study allows us to know precisely which modes of the equations are causing instabilities, and that the best way to fix the problem is either to change the differential operator or to filter.

Many attempts were done, and eventually the finite difference filtering method, although we do not have a really good theoretical understanding of it, seems to fix efficiently the oscillations, whereas the FC filter used previously did not. To make the filter local helps to keep a reasonable number of points per wavelength to represent accurately a function.

Work is left to be done, in particular to adapt this method for multi-mesh codes.

A Gradient and divergence in the local basis

We will denote by $\phi = (x, y)$ a diffeomorphism, going from Γ to Ω where Γ and Ω are two open and bounded domain of $\mathbf{R}^2$. We will denote by ∂_ξ and ∂_η the differentiation w.r.t the first and second coordinates respectively. We will define the local basis $(\mathbf{u}_\xi, \mathbf{u}_\eta)$ in each point by

$$\mathbf{u}_\xi = \frac{1}{n_\xi}\begin{pmatrix}\partial_\xi x\\ \partial_\xi y\end{pmatrix} \text{ and } \mathbf{u}_\eta = \frac{1}{n_\eta}\begin{pmatrix}\partial_\eta x\\ \partial_\eta y\end{pmatrix}, \tag{17}$$

where $n_\xi = \sqrt{(\partial_\xi x)^2 + (\partial_\xi y)^2}$ and $n_\eta = \sqrt{(\partial_\eta x)^2 + (\partial_\eta y)^2}$. We will also denote by $\mathcal{J}$ the jacobian of the map ϕ :

$$\mathcal{J} = \partial_\xi x \partial_\eta y - \partial_\eta x \partial_\xi y. \tag{18}$$

A.1 Gradient

If we have a function $f : \Omega \to \mathbf{R}$, its gradient $\mathbf{g} = \nabla f$ is a vector field. We are interested in the expression of the coordinates $(\tilde{g}_\xi, \tilde{g}_\eta)$ of $\mathbf{g} \circ \phi$ in the basis $(\mathbf{u}_\xi, \mathbf{u}_\eta)$. By denoting $\tilde{f} = f \circ \phi$, we have :

$$\begin{pmatrix}\tilde{g}_\xi\\ \tilde{g}_\eta\end{pmatrix} = \frac{n_\xi n_\eta}{\mathcal{J}^2}\begin{pmatrix}1 & -\langle \mathbf{u}_\xi, \mathbf{u}_\eta\rangle\\ -\langle \mathbf{u}_\xi, \mathbf{u}_\eta\rangle & 1\end{pmatrix}\begin{pmatrix}n_\eta \partial_\xi \tilde{f}\\ n_\xi \partial_\eta \tilde{f}\end{pmatrix}. \tag{19}$$

A.2 Divergence

Now if we have a vector field $\tilde{\mathbf{g}} : \Gamma \to \mathbf{R}^2$, whose coordinates $(\tilde{g}_\xi, \tilde{g}_\eta)$ are given in the local basis $(\mathbf{u}_\xi, \mathbf{u}_\eta)$, we would like to express the divergence of $\mathbf{g} = \tilde{\mathbf{g}} \circ \phi^{-1}$. We have :

$$(\nabla.\mathbf{g}) \circ \phi = \frac{1}{\mathcal{J}} \left(\partial_\xi \left(\frac{\mathcal{J}}{n_\xi} \tilde{g}_\xi \right) + \partial_\eta \left(\frac{\mathcal{J}}{n_\eta} \tilde{g}_\eta \right) \right). \tag{20}$$

References

[1] John P. Boyd. A comparison of numerical algorithms for fourier extension of the first, second, and third kinds. *Journal of Computational Physics*, 178(1):118 – 160, 2002.

[2] O.P. Bruno and A. Prieto. Spatially dispersionless, unconditionally stable fcad solvers for variable-coefficient pdes. *Journal of Scientific Computing*, 58(2):331–366, 2014.

[3] Oscar P. Bruno and Mark Lyon. High-order unconditionally stable fc-ad solvers for general smooth domains i. basic elements. *Journal of Computational Physics*, 229(6):2009 – 2033, 2010.

[4] Datta V. Gaitonde and Miguel R. Visbal. Padé-type higher-order boundary filters for the navier-stokes equations. *AIAA Journal*, 38(11):2103–2112, 2000.

[5] Mark Lyon and Oscar P. Bruno. High-order unconditionally stable fc-ad solvers for general smooth domains ii. elliptic, parabolic and hyperbolic pdes; theoretical considerations. *Journal of Computational Physics*, 229(9):3358 – 3381, 2010.