Berkeley
UNIVERSITY OF CALIFORNIA

ASSIGNMENT

UNIVERSITY OF CALIFORNIA, BERKELEY

DEPARTMENT OF MECHANICAL ENGINEERING

Introduction to the Finite Element Method Assignment 4

Author:
Théophile Sautory

Email:
tsautory@berkeley.edu

Date: November 3, 2021

1 Introduction

In this section, we introduce the problem and the goal of our analysis, providing key technical definitions and high-level overview of the of the components of the results.

The goal of this report is to present the implementation of an FEM solver that can solve a wave Equation (1):

$$\ddot{u} - c^2 \nabla \cdot \nabla \boldsymbol{u} = \boldsymbol{f} \quad \text{in } \Omega \times I, \tag{1}$$

subject to initial and boundary conditions (BC):

$$\begin{aligned} u &= u_0 \quad \boldsymbol{x} \in \Omega, t = 0 \\ \dot{u} &= v_0 \quad \boldsymbol{x} \in \Omega, t = 0 \\ u &= g \quad \text{on } \Gamma_g \times I \\ c^2 \nabla u \cdot \hat{n} &= h \quad \text{on } \Gamma_h \times I \end{aligned}$$

where Ω is the space domain, I the time domain, c is the the wave speed and $\hat{n}$ is the outward normal to $\partial\Omega$.

1.1 Problem set-up

We are interested in solving this equation in a provided domain, in which the flow must be discretized. The mesh is pre-built using linear triangular elements and has a homogeneous density of elements throughout.

1.1.1 Boundary Conditions

In this section, we present the necessary boundary conditions (BC) defined to solve the problem in the discretized domain.

The domain boundary is composed of Γ_q, Γ_h where q refers to the Dirichlet boundary and h to the Neumann boundary. From the assignment, we have that these BCs respectively are:

$$\begin{cases} g = A \sin(2\pi\omega t) \\ h = 0 \end{cases},$$

with the amplitude $A = 0.05$, and the frequency $\omega = 3$, scaled four our bi-unit domain dimension. This ensures that we are creating waves from the Dirichlet boundary, which simulates Young's two-slit experiment.

1.1.2 System Of Equations

Equation (1) is an equation of second order in the displacement $\boldsymbol{u}$. In this assignment, we are interested in comparing the performance of the first order θ family of time-steppers. Hence, we re-write the equation in a system of first order differential equations. Doing so yields the following system:

$$\begin{cases} \dot{u} = v \\ \dot{v} = \nabla \cdot \nabla u \end{cases}, \tag{2}$$

where we have set $c = 1$ and $f = 0$ for our problem.

1.2 Goals

We have now presented the problem set-up and necessary BCs, which allow us to derive the weak and matrix form of our problem statement. Once these equations are built, we will solve for the values of the vertical displacement and velocity across the domain, from which we can compare the effects of different time stepping models.

The user of the program should be able to select the value of θ, the time step Δt, and whether lumping is to be used to accelerate the computations. The goal of the analysis is then to investigate aspects of each models such as the stability, qualitative accuracy and numerical dissipation. This example will highlight some advantages and disadvantages of each scheme.

2 Method

This section presents the solution strategy through the FEM theoretical aspect and resulting algorithmic implementation. Firstly, we will go through the steps deriving the matrix form from the strong form of the problem, before presenting the required approximations used to write the solution as an the algorithm. We will then present key steps of the code such as the assembly loop of the stiffness and mass matrix, and mass lumping.

2.1 Strong Form to the Matrix Form

To solve the problem using FEM, we must first express the strong form of the problem and rewrite into its weak form, before expressing the matrix form.

To derive the weak form, we multiply Equations by a weighting function w such that:

$$w \in \mathcal{V} \equiv \left\{ w | w \in \mathcal{H}^1, w(\boldsymbol{x}) = 0, \boldsymbol{x} \in \Gamma_g \right\}$$

Doing so we get the following system of equation for all $w \in \mathcal{V}$:

$$\begin{cases} \int_{\Omega \times I} w \left(\dot{u} - v \right) d \left(\Omega \times I \right) = 0 \\ \int_{\Omega \times I} w \left(\dot{v} - \nabla \cdot \nabla u \right) d \left(\Omega \times I \right) = 0 \end{cases}, \tag{3}$$

noting that $w(\boldsymbol{x}) = 0 \forall \boldsymbol{x} \in \gamma_g$. Note that we define ∇ as the gradient with respect to the global coordinates.

We apply integration by parts and the divergence theorem on the second equation from Equation (4) as:

$$\int_\Omega w \nabla \cdot \nabla u d\Omega = - \int_\Omega \nabla w^T \nabla u d\Omega + \int_{\Gamma_q} w h d\Gamma,$$

to get the complete weak form of the system for all $w \in \mathcal{V}$:

$$\begin{cases} \int_I \left[\int_\Omega w \left(\dot{u} - v \right) d\Omega \right] dI = 0 \\ \int_I \left[\int_\Omega w \dot{u} d\Omega + \int_\Omega \nabla w^T \nabla u d\Omega - \int_{\Gamma_q} w h d\Gamma \right] dI = 0 \end{cases}. \tag{4}$$

The next step is to express the matrix formulation of the problem. We define η_g and η_q as the node IDs for the Dirichlet and Neumann BCs respectively, and η for the rest of the mesh nodes. We use the Galerkin method to provide approximations of w, u and v as the sum of basis functions N_A for all nodes A:

$$w \approx w^h = \sum_{A \in \eta}^{n} c_A(t) N_A(\boldsymbol{x}) \tag{5}$$

$$u \approx u^h = \sum_{A \in \eta}^{n} d_B(t) N_B(\boldsymbol{x}) + \sum_{B \in \eta_g} g_B(t) N_B(\boldsymbol{x}) \tag{6}$$

$$v \approx u^h = \sum_{A \in \eta}^{n} e_B(t) N_B(\boldsymbol{x}) + \sum_{B \in \eta_g} \dot{g}_B(t) N_B(\boldsymbol{x}) \tag{7}$$

We can now introduce these approximations to into the weak form to get the matrix form pf this system of equations.

2.1.1 First equation from Equation (2)

Introducing the Galerkin approximations in the first equation from Equation (2) we get:

$$\int_I \left[\int_\Omega \sum_{A\epsilon\eta} c_A(t) N_A(\boldsymbol{x}) \left(\sum_{B\epsilon\eta} \dot{d}_B(t) N_B(\boldsymbol{x}) + \sum_{B\epsilon\eta_g} \dot{g}_B(t) N_B(\boldsymbol{x}) \right) d\Omega \right] dI -$$
$$\int_I \left[\int_\Omega \sum_{A\epsilon\eta} c_A N_A(\boldsymbol{x}) \left(\sum_{B\epsilon\eta} e_B(t) N_B(\boldsymbol{x}) + \sum_{B\epsilon\eta_g} \dot{g}_B(t) N_B(\boldsymbol{x}) \right) d\Omega \right] dI = 0$$
$$\implies \int_I \left[\sum_{A\epsilon\eta} c_A(t) \int_\Omega N_A(\boldsymbol{x}) \left(\sum_{B\epsilon\eta} \dot{d}_B(t) N_B(\boldsymbol{x}) - \sum_{B\epsilon\eta} e_B(t) N_B(\boldsymbol{x}) \right) d\Omega \right] dI = 0$$

Hence, defining the following matrix:

$$\boldsymbol{M} = \int_\Omega N_A N_B d\Omega. \tag{8}$$

We then get the following equation:

$$\int_I c^T \left[\boldsymbol{M}\dot{\boldsymbol{d}} - \boldsymbol{M}\boldsymbol{e} \right] = 0.$$

where we stacked the values of c_A in a vector $\boldsymbol{c}$. Considering that this must hold for all c_A and time interval I, we obtain the following matrix equation (setting each respective c to 1 and others to 0):

$$\boldsymbol{M}\left[\dot{\boldsymbol{d}} - \boldsymbol{e}\right] = 0.$$

Introducing the time-stepping scheme:

$$\boldsymbol{d}^{i+1} = \boldsymbol{d}^i + \theta\Delta t \dot{\boldsymbol{d}}^{i+1} + (1-\theta)\Delta t \dot{\boldsymbol{d}}_i \tag{9}$$

we get:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{d}^{i+1} &= \boldsymbol{M}\boldsymbol{d}^i + \theta\Delta t \boldsymbol{M}\dot{\boldsymbol{d}}^{i+1} + (1-\theta)\Delta t \boldsymbol{M}\dot{\boldsymbol{d}}_i \\ &= \boldsymbol{M}\boldsymbol{d}^i + \theta\Delta t \boldsymbol{M}\boldsymbol{e}^{i+1} + (1-\theta)\Delta t \boldsymbol{M}\boldsymbol{e}_i, \end{aligned}$$

and therefore:

$$\implies \boxed{\boldsymbol{M}\boldsymbol{d}^{i+1} - \theta\Delta t \boldsymbol{M}\boldsymbol{e}^{i+1} = \boldsymbol{M}\boldsymbol{d}^i + (1-\theta)\Delta t \boldsymbol{M}\boldsymbol{e}_i}.$$

2.1.2 Second equation from Equation (2)

Similarly to the first equation, we proceed by introducing the finite approximations in the weak form, yielding:

$$\int_I \left[\sum_{A\epsilon\eta} c_A \left(\int_\Omega N_A \sum_{B\epsilon\eta} \dot{e}_B d\Omega + \int_\Omega \nabla N_A^T \sum_{B\epsilon\eta} \nabla N_B d_B d\Omega \right) \right] dt =$$
$$\int_I \left[\sum_{A\epsilon\eta} c_A \left(\int_{\Gamma_q} N_A h d\Gamma + \int_\Omega N_A \sum_{B\epsilon\eta_g} N_B \ddot{g}_B d\Omega + \int_\Omega \nabla N_A^T \sum_{B\epsilon\eta_g} \nabla N_B g_B d\Omega \right) \right] dt$$

Setting the Dirichlet terms to 0, factorizing for all c values and defining the following:

$$\begin{aligned} \boldsymbol{K} &= \int_\Omega \nabla N_A^T \nabla N_B d\Omega, \\ \boldsymbol{f} &= \int_{\Gamma_q} N_A h d\Gamma, \end{aligned} \tag{10}$$

allows us to write:

$$\boldsymbol{M}\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{d} - \boldsymbol{f} = 0.$$

Introducing the time-stepping scheme, we get:

$$\begin{aligned}
&\boldsymbol{e}^i = -\boldsymbol{M}^{-1}\left(\boldsymbol{K}\boldsymbol{d}^i - \boldsymbol{f}^i\right) \\
\implies &\boldsymbol{e}^{i+1} = \boldsymbol{e}^i + \theta\Delta t\left[-\boldsymbol{M}^{-1}\left(\boldsymbol{K}\boldsymbol{d}^{i+1} - \boldsymbol{f}^{i+1}\right)\right] + (1-\theta)\Delta t\left[-\boldsymbol{M}^{-1}\left(\boldsymbol{K}\boldsymbol{d}^i - \boldsymbol{f}^i\right)\right] \\
\implies &\theta\Delta t\boldsymbol{K}\boldsymbol{d}^{i+1} + \boldsymbol{M}\boldsymbol{e}^{i+1} = (\theta-1)\Delta t\boldsymbol{K}\boldsymbol{d}^i + \boldsymbol{M}\boldsymbol{e}^i + \theta\Delta t\boldsymbol{f}^{i+1} + (1-\theta)\Delta t\boldsymbol{f}^i.
\end{aligned}$$

Combining the two obtained matrix equations we get:

$$\begin{bmatrix} \boldsymbol{M} & -\theta\Delta t\boldsymbol{M} \\ \theta\Delta t\boldsymbol{K} & \boldsymbol{M} \end{bmatrix}\begin{bmatrix} \boldsymbol{d}^{i+1} \\ \boldsymbol{e}^{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M} & (1-\theta)\Delta t\boldsymbol{M} \\ (\theta-1)\Delta\boldsymbol{K} & \boldsymbol{M} \end{bmatrix}\begin{bmatrix} \boldsymbol{d}^i \\ \boldsymbol{e}^i \end{bmatrix} + \begin{bmatrix} 0 \\ \theta\Delta t\boldsymbol{f}^{i+1} + (1-\theta)\Delta t\boldsymbol{f}^i \end{bmatrix}, \tag{11}$$

as expected from the assignment.

2.2 Basis Functions and Gradients

Please note that some of the following sections have been directly taken from my Assignment 3 as the method is exactly similar for the integrals, and resembles considerably for the code implementation.

Now that the strong form has been transformed into a matrix form, we are ready to express the basis functions used in our particular set-up.

We are due to solve the problem using a 2D triangular mesh with linear basis functions. Recall that the basis functions must be such that: $N^i(\boldsymbol{x}_j) = \delta_{ij}$. Over a triangular element, we use the right hand rule, with nodes of local coordinates positions $\boldsymbol{r} = (r,s)$:

$$\begin{aligned}
\boldsymbol{r}_1 &= (0,0) \\
\boldsymbol{r}_2 &= (1,0) \\
\boldsymbol{r}_3 &= (0,1),
\end{aligned}$$

and use the following basis functions:

$$\begin{aligned}
N_1(r,s) &= 1 - r - s \\
N_2(r,s) &= r \\
N_3(r,s) &= s.
\end{aligned} \tag{12}$$

As the equations in Equation (8;10) require integration of the basis functions with respect to the global Cartesian coordinates $\boldsymbol{x}$, we use the chain rule to express the latter as a function of the local coordinates (recall that we defined ∇ to be gradient with respect to $\boldsymbol{x}$):

$$\nabla N_A = (\boldsymbol{r}_{,\boldsymbol{x}})^T \nabla_{\boldsymbol{r}} N_A = (\boldsymbol{x}_{,\boldsymbol{r}})^{-T} \nabla_{\boldsymbol{r}} N_A.$$

We define $J = \boldsymbol{x}_{,\boldsymbol{r}}$ and note that for each element e:

$$\boldsymbol{x}^e(\boldsymbol{r}^e) = \sum_{A=1}^{3} \boldsymbol{x}^e{}_A N_A$$

which allows us to write for each element e:

$$J^e = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{bmatrix}_e = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}_e.$$

We are interested by the inverse transpose of J^e, and hence compute it's determinant:

$$j^e = [(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)]_e$$

and express:

$$J^{e^{-T}} = \frac{1}{j^e} \begin{bmatrix} y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{bmatrix}_e$$

We then define $\tilde{B}$ such that:

$$\tilde{B} = \nabla_{\boldsymbol{r}} N_{i=(1,2,3)} = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

2.3 Integral Evaluations

We are now interested in integrating the equations from Equation (10) in order to fill in the sitffness matrix and compute the values for the streamfunction using FEM. Note that for clarity, we will now remove the e superscript from the different expressions.

1. Local stifness matrix.
We first discuss the integration of the local stiffness matrix $\boldsymbol{k}_{ab}$ term, where $(a,b) \in (1,2,3)$. Using the change of variable formula to compute the integral over the local triangle we can write:

$$\begin{aligned} \boldsymbol{k}_{ab} &= \int_{\Omega^e} \nabla N_a \nabla N_b \, d\Omega \\ &= \int_{\Delta} \left(J^{-T} \nabla_{\boldsymbol{r}} N_a\right)^T \left(J^{-T} \nabla_{\boldsymbol{r}} N_b\right) j \, dr ds \\ &= \int_{\Delta} (J^{-T} \tilde{B})^T (J^{-T} \tilde{B}) j \, dr ds \\ &= (J^{-T} \tilde{B})^T (J^{-T} \tilde{B}) j \int_{\Delta} dr ds \\ &= \frac{1}{2} (J^{-T} \tilde{B})^T (J^{-T} \tilde{B}) j \end{aligned}$$

as all the terms inside the integral are constant.

2. Mass matrix
For each element, we similarly define the equivalent of a local mass matrix $\boldsymbol{m}_{ab}$:

$$\begin{aligned} \boldsymbol{m}_{ab} &= \int_{\Omega^e} N_A N_B d\Omega \\ &= \int_{r=0}^{1} \int_{s=0}^{1-r} N_A N_B j \, dr ds \\ &= j \int_{r=0}^{1} \int_{s=0}^{1-r} N_A N_B \, dr ds. \end{aligned}$$

Using the basis function defined in Equation (12) and performing the integration by hand fo all combinations of $(a,b) \in (1,2,3)$ we note that we get:

$$m_{ab} = \frac{1}{24}, \text{ if } a \neq b,$$
$$m_{ab} = \frac{1}{12}, \text{ if } a = b.$$

An example of calculation is shown below:

$$\int_{r=0}^{1} \int_{s=0}^{1-r} r \cdot s \, dr ds = \int_0^1 r \left. \frac{s^2}{2} \right|_0^{1-r} dr$$

$$
\begin{aligned}
&= \int_{r=0}^{1} \frac{r\left(1-2r+r^2\right)}{2} dr \\
&= \frac{1}{2}\left[\frac{r^4}{4} - \frac{2r^3}{3} + \frac{r^2}{2}\right] \\
&= \frac{1}{24}.
\end{aligned}
$$

Similar calculations have been done to compute the other combinations of basis functions. Hence we obtain the full expression for the local stiffness matrix as:

$$
\boldsymbol{m}_{ab} = \frac{1}{24}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}.
$$

Note that we do not explicit the expression for $\boldsymbol{f}_a$, as $h = 0$ and the integral is trivially 0.

2.4 Implementation

The FEM theory and necessary tools have been set up, we are now able to describe the proposed code solution.

2.4.1 Initialization

The first step of the code is to load the mesh and BCs data. While doing so, we name the variables based on the final terms of the filename, for clarity. The user defined parameters are then inserted as defined in the introduction.

We then initialize the arrays in the **initialize_arrays** function, described in the global matrix form Equation (11). The $\boldsymbol{K}$ global stiffness matrix and $\boldsymbol{M}$ the global mass matrix are $(nn \times nn)$, where nn is the number of nodes in the mesh; the $\boldsymbol{f}$ and d vectors are $((2 * nn) \times 1)$.

2.4.2 Specific Guidance functions

We then define the functions used to compute **klocal** and **mlocal**. For each of the element to which they apply, we compute the integrals discussed in Section 2.3 using the vertices of the three nodes forming the triangle, ordered using the right hand rule (from the mesh). Note that we define a **compute_jacobian_det_inverse** function which computes the gradients of the shape functions in the local coordinate system, the determinant of the Jacobian from global to local coordinates, and its inverse.

2.4.3 Main function

We are now ready to present the main function of the code, which we name **main**.

Once the mesh data has been loaded, and the problem initialized, we are able to enter the main loop of the program where we loop over each element, extract the nodes which form them and construct the local stiffness and mass matrices. We can then assemble the latter into the global stiffness and mass matrices respectively, according to the information provided by the mesh in the connectivity array.

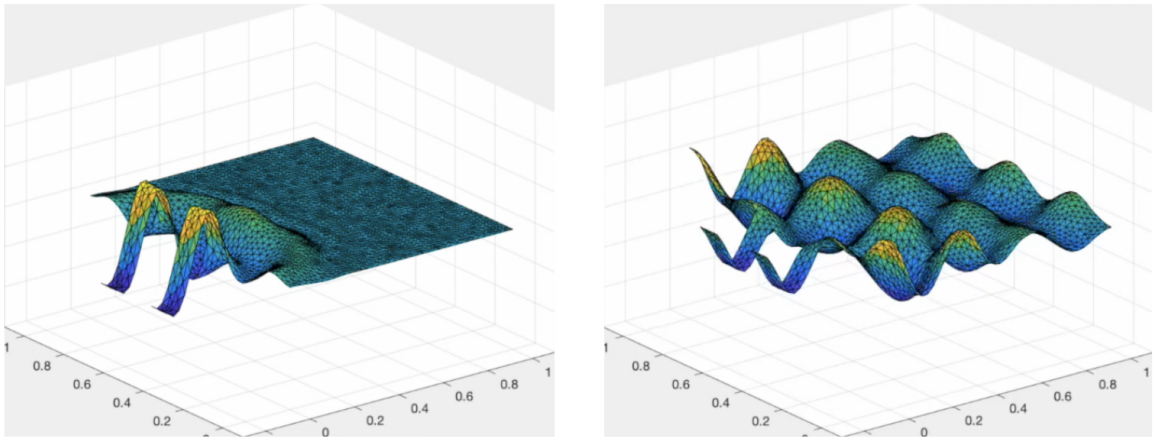
We then enter and *if* statement to select if mass lumping should be done or not. According to the input parameter, we build the matrix equation as defined in Equation (11). We then enforce the Dirichlet BCs. For the Dirichlet nodes, we set all the row of each node to 0 except the diagonal entry which is set to 1. By doing so, we are able to enforce the value as these nodes using the **g** function, which computes the value of the BC at a given time. Note that we do not consider the Neumann BC

in this code as the value h is set to 0 always, removing the term from the equation. We then inverse the left hand side part of the equation, and compute the right hand side part.

We are then interested in analyzing the results by investigating the differences in the evolution of the simulations with time and loop over the timesteps to get the evolution of the dynamics taking place, by left multiplying the right hand side of Equation (11) (which include the displacement and velocity values we have just solved for) by the inverse of the left hand side of the equation. We then save the displacement values of interest and use the template function to save the values of the frames of interest. We plot the results for the analysis using MATLAB, for exact matching between the deliverables and our work (in the file *Theophile_Sautory_videos_from_Python.m*).

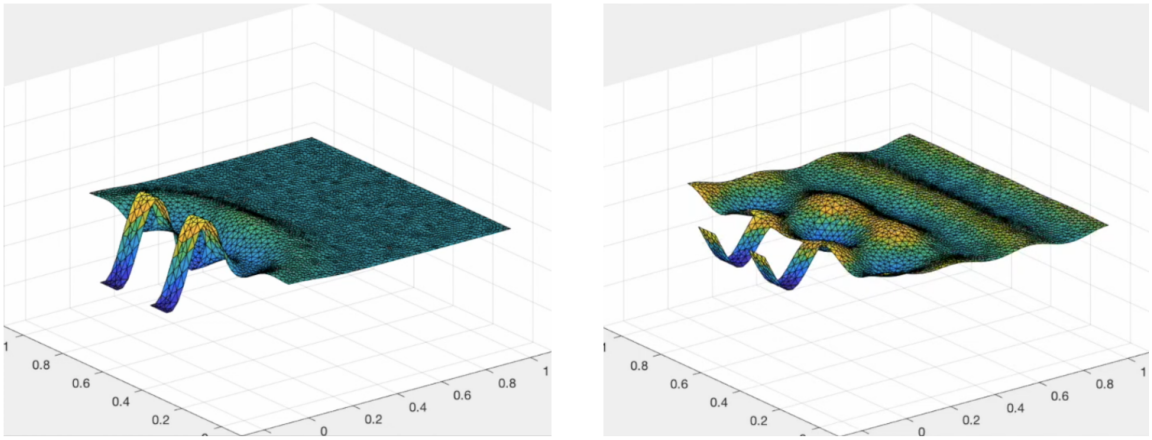
3 Results

This section presents the results of the FEM implementation solving the flow in the double slit experiment. The plots are directly constructed using the plotting template code provided in MATLAB in the assignment. We create the four videos of interest. For $\theta = 0.5$ and $\theta = 1$, we use $\Delta t = 0.005$. For $\theta = 0$ with mass lumping, we use $\Delta t = 5 \times 10^{-5}$, whilst we use $\Delta t = 1 \times 10^{-5}$. The videos are saved according to the "outfile" name provided in the template. We also plot snapshots of the runs in the following section. Please note that to respect the file sizes, we make videos of 150 frames with a rate of 15 frames per second.



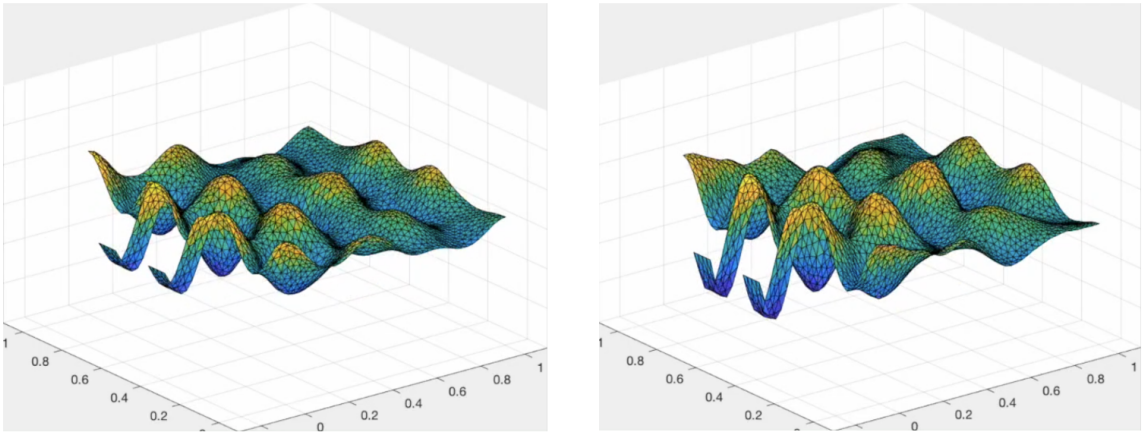
(a) Single Wavefront (b) Interferences Throughout the domain

Figure 1: Solution at different time-points with $\theta = 0.5$.



(a) Single Wavefront (b) Interferences Throughout the domain

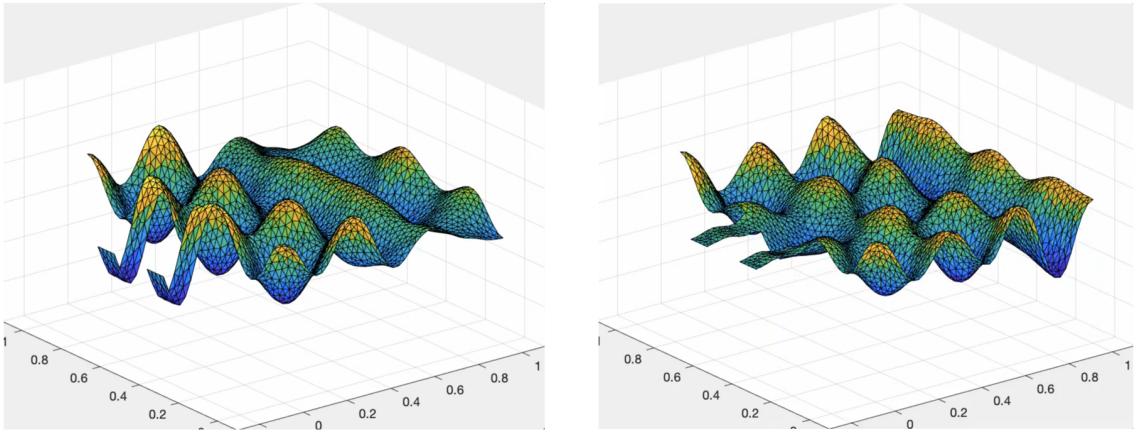
Figure 2: Solution at different time points with $\theta = 1$.



(a) Without Mass Lumping

(b) With Mass Lumping

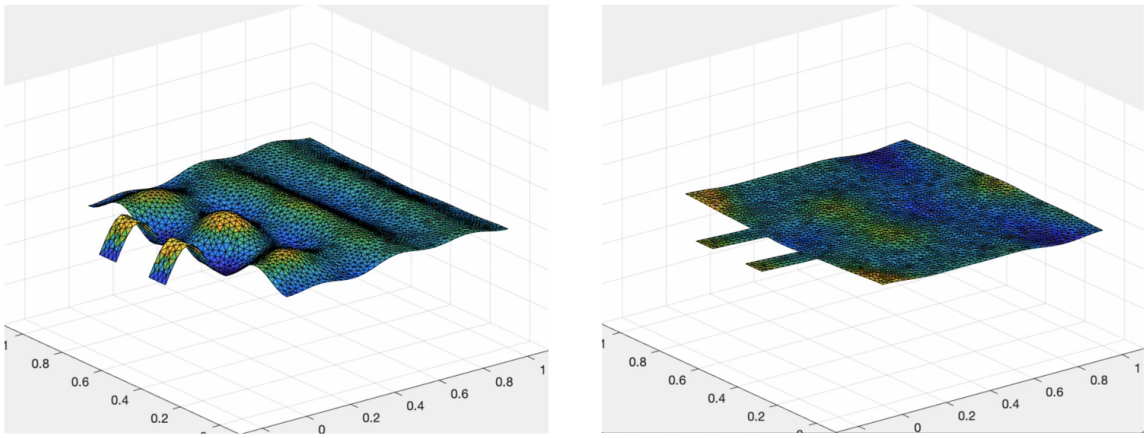
Figure 3: Example of the difference in amplitude for the solution without (a) and with mass lumping (b), using $\theta = 0$ and taken after 2.5 time units.



(a) Physics at time t=3

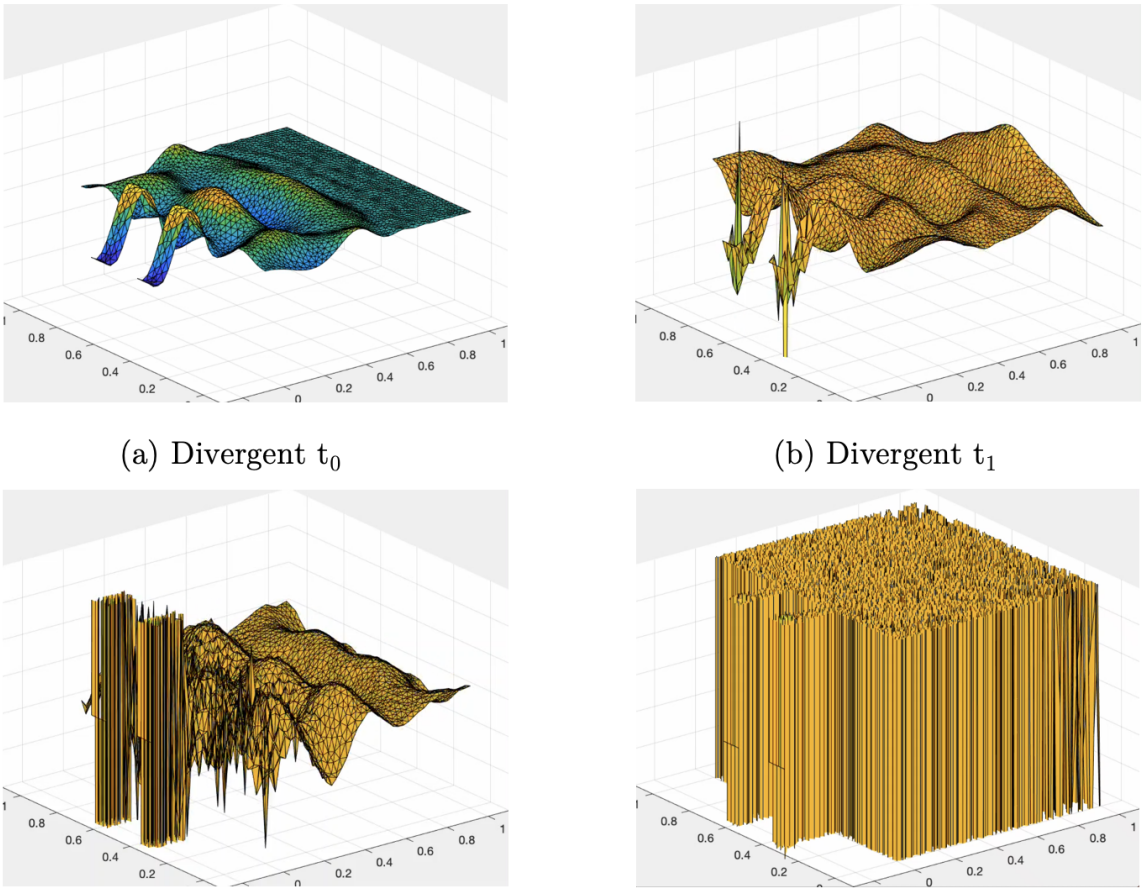
(b) Physics at time t=6

Figure 4: Evolution of the simulation turning off the sinusoidal wave at the Dirichlet nodes after $t < 3$ using the $\theta = 0.5$ scheme.



(a) Physics at time t=3 (b) Physics at time t=6

Figure 5: Evolution of the simulation turning off the sinusoidal wave at the Dirichlet nodes after $t < 3$ using the $\theta = 1$ scheme.



(a) Divergent t_0 (b) Divergent t_1

(c) Divergent t_2 (c) Divergent t_3

Figure 6: Example of the evolution a divergent solution at different time points, using $\theta = 0$ and a time step of $\Delta t = 5 \times 10^{-4}$.

4 Discussion

In this section, we discuss the results using different θ values for the time-step in terms of stability and accuracy, and the numerical dissipation which can arise form these.

We solve the problem with $\theta = 0.5$ using a time-step $\Delta = 0.005$ as this proved to have similar (qualitatively) aspects than $\Delta = 0.01$, suggesting that the solution had converged. The results are shown in the video *results-theta-0.5-ML-False-dt-0.005.mp4*, and snapshots are provided in Figure 1. As expected, the results of the simulations were stable. This is due to the implicit nature of the scheme with $\theta \geq 0.5$. In this scheme, referred to as the midpoint or Crank Nicolson scheme, we weight the current time-step derivative and future time-step derivative equally. The physics taking place in the experiment reveal similar effects that the double slit experiment from Thomas Young. Indeed, there is a wave front where the interaction of the waves coming form the two slits lead to both constructive and destructive interference, where the amplitude of the wave respectively increases and decreases (Figure 1(a)). This suggests that the sin wave Dirichlet BCs at the slits entry is well imposed. Moreover, we can observe that the the wave is reflected back off the walls which form the Neumann BC (Figure 1(b)). This physical behaviour is expected as we set these to have no flux throughout. The reflection of the wave then increases the complexity of the global physical event taking place as it introduces new interference with the new upcoming waves, which end up in a map of numerous constructive and destructive waves over the domain, with different large and small displacements (Figure 1(b)). We note that the amplitude of the waves remains large, and there does not seem to be large numerical energy dissipation.

We then run the solver using the same time-step ($\Delta t = 0.005$), with $\theta = 1$. Please find attached the video *results-theta-1-ML-False-dt-0.005.mp4*, and refer to Figure 2. By imposing $\theta = 1$, we are solely using the derivative of the future time step in Equation (9), which provides a fully implicit scheme. Anew, the results are stable, however, there is much more dissipation of the energy of the system, as highlighted by the much lower amplitudes of the resulting waves in Figure 2(b). This result was expected as the scheme is fully implicit and inherently more dissipative. We also hypothesize that the results in this simulation are much less accurate than that obtained using $\theta = 0.5$. Indeed, the large dampening of the wave amplitude using $\theta = 1$ seems less physically realistic than the simulation ran with $\theta = 0.5$. This suggests that we are in a problem where the difference in the order of the error due to the numerical schemes of Δt^3 and Δt^2 for $\theta = 0.5$ and $\theta = 1$ is respected. The accuracy seems high in the latter case, whilst the solutions are both stable.

We then solve the problem using $\theta = 0$, implying a fully explicit scheme. Please find attached the videos *results-theta-0.0-ML-False-dt-1e-05.mp4*, *results-theta-0.0-ML-True-dt-5e-05.mp4* and refer to Figure 3 for the results without and with mass lumping. In this scheme, we do not use the derivative at the next time-step to compute an estimate of the displacement at a time-step i. This is much less computationally expensive as there global stiffness matrix $\boldsymbol{K}$ is not required to be inverted any longer - and our left hand side if filled with 0s in the martix Equation (11). This is in part the rationale behind using mass lumping - which also resembles computing the matrix with nodal quadrature. However, we observe that computational advantage is counter-balanced by the requirement on the time-step size. Indeed, using mass lumping, the simulation only became fully stable when we set $\Delta t = 5 \times 10^{-5}$ (the results we nearly stable at $\Delta t = 10^{-4}$, but not exactly throughout the domain). This step size requirement is due to the explicit nature of the method, which is conditionally stable. The stable simulation without mass lumping shown in Figure 3(a) resembles that of the $\theta = 0.5$ scheme, revealing that there does not seem to be a large quantity of energy dissipation, suggesting a greater accuracy than the fully implicit scheme. On the other hand, the simulation of the model ran with $\theta = 0$ without using mass lumping (and $\Delta t = 1 \times 10^{-5}$) seems to increase the difference between neighbouring values (greater slope of the maximum and minimum of the domain displacement / sharper climbs). This is observable in Figure 3(b). We explain the latter by the fact that the mass lumping concentrates all the values to each node, and does not share/diffuse the value over different nodes. By doing so, it appears sharper as the local energy is not diffused. We also show the development of a divergent solution, where we can see that the divergence initiates at the Dirichlet nodes,

before diffusing throughout the whole domain and oscillating between $\pm\infty$ values. Figure 6 reveals these dynamics with $\Delta t = 5 \times 10^{-4}$, which is already 10 times smaller than for the $\theta = 1, 0.5$ schemes. This demonstrates that the solution us unstable for Δt not small enough.

Finally, we explore the numerical dissipation in the different methods used by integrating the solution over 6 time units, setting the Dirichlet BC to 0 from time 3 onward. Doing so essentially removes the energy input to the system, however, as the equation is hyperbolic, the solution should "conserve" its discontinuities, and we expect the energy to be conserved. Figure 4 and 5 reflect the evolution of the system with $\theta = 0.5$ and $\theta = 1$ respectively and $\Delta \sim 10^{-3}$. Running this experiment, we observe numerical dissipation for the $\theta = 1$ scheme, which is not (or barely) observable for $\theta = 0.5$. This reveals there is numerical energy dissipation caused by the implicit scheme. This scheme has the advantage of being stable, at the cost of introducing artificial diffusion on the simulation (which enables stability). This observation can be seen both with time-steps of order of magnitude $\Delta \sim 10^{-3}$ and $\Delta \sim 10^{-2}$, and we plot only one example of figures as the other relate the exact same information (for brevity).