

The Finite Element Method

Mikkel Paltorp (mpasc@dtu.dk), 11 May 2026

Department of Applied Mathematics and Computer Science, Technical University of Denmark

The aim of the Finite Element Method (FEM) is to solve (or more precisely approximate the solution of) a Partial Differential Equation (PDE) using a computer. In general the solution to a PDE is a *function*. Unfortunately it is computationally intractable to task a computer with finding this underlying function. The idea to resolve this issue is to parametrize a family of functions with the aim of computing the parameters through solving a linear system of equations. There is a plethora of ways to get to such a linear system of equations, with the most common being the so-called Galerkin approach.

1. Getting the computer to understand functions

In the most basic terms, computers are only capable of understanding numbers, which means that they are inherently unable to solve equations where the unknowns are functions. This is a problem when trying to solve differential equations. To solve this problem, the functions are instead approximated using parameterizations for which the coefficients (numbers) are unknown. Intuitively, these numbers are exactly what the computer is asked to find. For element methods, this parameterization is chosen to be the simplest possible: a linear combination of functions

$$p(\mathbf{x}) \approx \mathbf{T}(\mathbf{x})\mathbf{p} = \begin{bmatrix} T_1(\mathbf{x}) & T_2(\mathbf{x}) & \dots & T_{n(\mathbf{x})} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}, \quad (1)$$

where p is the unknown function being approximated. Note that the linearity is with respect to the unknown parameters $\mathbf{p}$, but not necessarily in the known basis functions $\mathbf{T}(\mathbf{x})$.

One might ask: *How does the above relate to the Finite Element Method?* The answer is that the functions T_i are chosen to be simpler functions with support equal to only a few subdomains of the original domain. These subdomains are commonly referred to as elements.

Interactive figure 1 (Piecewise polynomial approximation) In one dimension an approximation of the form Equation 1 is easy to visualize. The graph shows the function p (dashed) and the approximation $\mathbf{T}(x)\mathbf{p}$ (solid) on N elements (every other one shaded). The coefficients $p_i = p(x_i)$ are the large dots, and each basis function T_i , scaled by its p_i , is drawn faint. There are two ways to improve the approximation: more elements¹ or basis functions of a higher polynomial order p .²

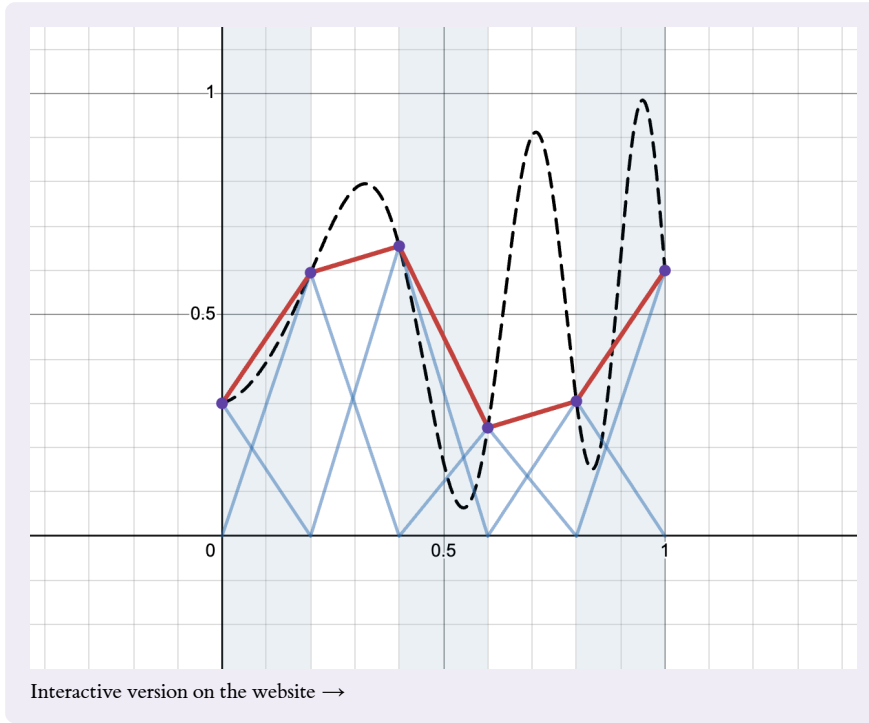
Contents

1. Getting the computer to understand functions
 2. The Finite Element Method for the Helmholtz equation
 3. What is an element?
 4. Computing the stiffness and mass matrices
 5. Examples
- Bibliography

Interactive figure 1: Piecewise polynomial approximation

¹ Called h -refinement, as it decreases the element size $h = 1/N$.

² Called p -refinement. Here p is the order, not the function being approximated.



2. The Finite Element Method for the Helmholtz equation

In acoustics we are often interested in solving the Helmholtz equation, given as

$$\Delta p(\mathbf{x}) + k^2 p(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \quad (2)$$

where k is most often referred to as the wavenumber. In its current form the basis functions $\mathbf{T}(\mathbf{x})$ must be twice differentiable; piecewise linear functions, for example, have a vanishing second derivative inside each element and cannot be used. This constraint can be relieved using the so-called weak formulation. In order to do so we multiply the Helmholtz equation with a so-called test function φ followed by an integration over the domain Ω

$$\int_{\Omega} \varphi(\mathbf{x}) (\Delta p(\mathbf{x}) + k^2 p(\mathbf{x})) \, d\mathbf{x} = 0. \quad (3)$$

Utilizing integration by parts on the term including the Laplacian we can move derivatives from $p(\mathbf{x})$ onto $\varphi(\mathbf{x})$ as

$$\int_{\Omega} (\nabla \varphi(\mathbf{x}))^{\top} \nabla p(\mathbf{x}) \, d\mathbf{x} - k^2 \int_{\Omega} \varphi(\mathbf{x}) p(\mathbf{x}) \, d\mathbf{x} - \int_{\partial\Omega} \varphi(\mathbf{x}) \frac{\partial p}{\partial n}(\mathbf{x}) \, d\mathbf{x} = 0. \quad (4)$$

In the above the constraint on the smoothness of $p(\mathbf{x})$ has been reduced to first order. Furthermore the last term

$$\int_{\partial\Omega} \varphi(\mathbf{x}) \frac{\partial p}{\partial n}(\mathbf{x}) \, d\mathbf{x} \quad (5)$$

can be used to include the boundary conditions into the equation. For simplification purposes we will in the following assume that the above is equal to zero. As such we have

$$0 = \int_{\Omega} (\nabla \varphi(\mathbf{x}))^{\top} \nabla p(\mathbf{x}) \, d\mathbf{x} - k^2 \int_{\Omega} \varphi(\mathbf{x}) p(\mathbf{x}) \, d\mathbf{x}. \quad (6)$$

We will now use the Galerkin approach to discretize the above equation. Note that “Galerkin” simply refers to the approach where $\varphi(\mathbf{x})$ and $p(\mathbf{x})$ are discretized using the same basis functions ($\mathbf{T}(\mathbf{x})$). In short this means that we introduce

$$p(\mathbf{x}) \approx \mathbf{T}(\mathbf{x})\mathbf{p}, \quad \varphi(\mathbf{x}) \approx \mathbf{a}^{\top} \mathbf{T}(\mathbf{x})^{\top}, \quad (7)$$

where $\mathbf{a} \in \mathbb{C}^n$ is a set of arbitrary coefficients that parametrize a whole *family* of test functions and $\mathbf{p} \in \mathbb{C}^n$ are the coefficients that we aim to find. Inserting the approximations it follows that

$$\begin{aligned} 0 &= \int_{\Omega} (\nabla \varphi(\mathbf{x}))^{\top} \nabla p(\mathbf{x}) \, d\mathbf{x} - k^2 \int_{\Omega} \varphi(\mathbf{x}) p(\mathbf{x}) \, d\mathbf{x} \\ &\approx \int_{\Omega} \mathbf{a}^{\top} (\nabla \mathbf{T}(\mathbf{x}))^{\top} \nabla \mathbf{T}(\mathbf{x}) \mathbf{p} \, d\mathbf{x} - k^2 \int_{\Omega} \mathbf{a}^{\top} \mathbf{T}(\mathbf{x})^{\top} \mathbf{T}(\mathbf{x}) \mathbf{p} \, d\mathbf{x} \\ &= \mathbf{a}^{\top} \left(\int_{\Omega} (\nabla \mathbf{T}(\mathbf{x}))^{\top} \nabla \mathbf{T}(\mathbf{x}) \, d\mathbf{x} \right) \mathbf{p} - k^2 \mathbf{a}^{\top} \left(\int_{\Omega} \mathbf{T}(\mathbf{x})^{\top} \mathbf{T}(\mathbf{x}) \, d\mathbf{x} \right) \mathbf{p} \\ &= \mathbf{a}^{\top} \left(\int_{\Omega} (\nabla \mathbf{T}(\mathbf{x}))^{\top} \nabla \mathbf{T}(\mathbf{x}) \, d\mathbf{x} - k^2 \int_{\Omega} \mathbf{T}(\mathbf{x})^{\top} \mathbf{T}(\mathbf{x}) \, d\mathbf{x} \right) \mathbf{p}. \end{aligned} \quad (8)$$

We can make the above equality hold for all possible test functions, i.e. for all $\mathbf{a} \in \mathbb{C}^n$, if we find $\mathbf{p}$ such that

$$\left(\int_{\Omega} (\nabla \mathbf{T}(\mathbf{x}))^{\top} \nabla \mathbf{T}(\mathbf{x}) \, d\mathbf{x} - k^2 \int_{\Omega} \mathbf{T}(\mathbf{x})^{\top} \mathbf{T}(\mathbf{x}) \, d\mathbf{x} \right) \mathbf{p} = \mathbf{0}. \quad (9)$$

This is exactly what we do in Finite Element computations. As a shorthand the above can be written as

$$(\mathbf{K} - k^2 \mathbf{M})\mathbf{p} = \mathbf{0}, \quad (10)$$

where $\mathbf{K}$ is referred to as the *stiffness matrix* and $\mathbf{M}$ as the *mass matrix*. In the literature the equation can also be found as

$$(c^2 \mathbf{K} - \omega^2 \mathbf{M})\mathbf{p} = \mathbf{0}, \quad (11)$$

where c is the propagation speed and ω is the angular frequency.

3. What is an element?

A key insight is that the element serves two purposes: it represents a subdomain of the original domain (also referred to as the geometry) while also describing parts of the unknown function(s) of interest. In general this subdomain is described by a parameterization, i.e. the element, as

$$\mathbf{x}^e(\mathbf{u}) = \mathbf{X}^e \mathbf{N}^e(\mathbf{u}) \in \Omega^e, \quad \forall \mathbf{u} \in \mathcal{L}^e, \quad (12)$$

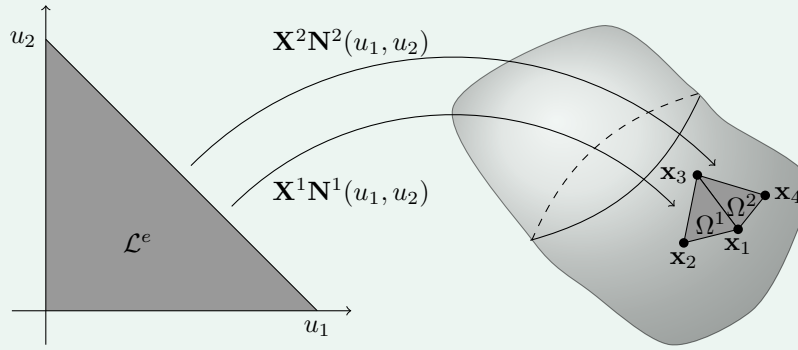
where the superscript e denotes the element number, $\mathbf{X}^e$ is a matrix with columns equal to the interpolation nodes of the geometry, $\mathbf{N}^e(\mathbf{u})$ are the so-called shape functions, Ω^e is the element in global coordinates and $\mathcal{L}^e$ are the local coordinates. In addition to the geometric interpolation of each element, we need to further define interpolations of the unknown functions, which in acoustics is usually taken as the pressure $p(\mathbf{x})$. On element e this interpolation can be done as

$$p(\mathbf{x}^e(\mathbf{u})) = \mathbf{T}(\mathbf{x}^e(\mathbf{u}))\mathbf{p} = \underbrace{\mathbf{T}(\mathbf{x}^e(\mathbf{u}))(\mathbf{L}^e)^\top}_{\mathbf{T}^e(\mathbf{u})} \underbrace{\mathbf{L}^e \mathbf{p}}_{\mathbf{p}^e} = \mathbf{T}^e(\mathbf{u})\mathbf{p}^e, \quad \mathbf{u} \in \mathcal{L}^e, \quad (13)$$

where $\mathbf{L}^e$ is a permutation-like matrix that extracts the relevant values of $\mathbf{p}$ and the relevant functions of $\mathbf{T}(\mathbf{x})$ and orders them such that they correspond to the local values $\mathbf{p}^e$ and the local basis functions $\mathbf{T}^e(\mathbf{u})$.

Example 1: Linear triangular elements

Example 1 (Linear triangular elements)



The linear shape functions for a triangular element *can* have the form

$$\mathbf{N}^e(u_1, u_2) = \begin{bmatrix} 1 - u_1 - u_2 \\ u_1 \\ u_2 \end{bmatrix}, \quad u_1 \in [0, 1], u_2 \in [0, 1 - u_1]. \quad (14)$$

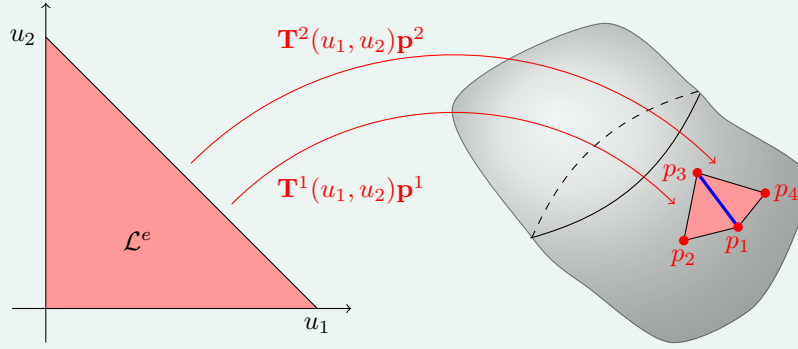
The choice in the wording is because the ordering of the columns of $\mathbf{X}^e$ can change the ordering of the rows of $\mathbf{N}^e(\mathbf{u})$ or vice versa. This is something that one should keep in mind in practice when using different mesh file formats. Taking the second element of the figure above as an example, it could be that

$$\mathbf{X}^2 = [\mathbf{x}_3 \ \mathbf{x}_1 \ \mathbf{x}_4]. \quad (15)$$

Note that extending the geometric interpolation to higher orders is as simple as adding more rows/functions to $\mathbf{N}^e(u_1, u_2)$ as well as more columns/points to $\mathbf{X}^e$.

Example 2: Basis functions for continuous linear interpolation

Example 2 (Basis functions for continuous linear interpolation)



Continuous linear basis functions on triangular elements are similar to shape functions for a linear triangular element and differ only in the fact that they are the transpose:

$$\mathbf{T}_{\text{continuous}}^e(u_1, u_2) = [1 - u_1 - u_2 \quad u_1 \quad u_2], \quad u_1 \in [0, 1], u_2 \in [0, 1 - u_1], \quad (16)$$

where the subscript “continuous” is only there to highlight that it is a continuous formulation. Again, the ordering of the columns of the row vector depends on the ordering of the element corners.

Example 3: Element localization matrix

Example 3 (Element localization matrix) For a continuous linear element e all three corners correspond to a value of the global vector $\mathbf{p}$. For example, the second element in the continuous plot has local corner values given by $\mathbf{p}^2 = [p_3 \ p_1 \ p_4]^\top$. This element would have $\mathbf{L}^e$ given as

$$\mathbf{L}^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \end{bmatrix}, \quad (17)$$

so that $\mathbf{p}^2 = \mathbf{L}^2 \mathbf{p}$. Note that $\mathbf{L}^e$ is only an artifact of the mathematical description. Any reasonable implementation should use indexing instead of multiplication with $\mathbf{L}^e$.

4. Computing the stiffness and mass matrices

Using the element description of both the geometry and the pressure function the mass matrix can be computed as follows

$$\begin{aligned} \mathbf{M} &= \int_{\Omega} \mathbf{T}(\mathbf{x})^\top \mathbf{T}(\mathbf{x}) \, d\mathbf{x} \\ &\approx \sum_{e=1}^N \int_{\Omega^e} \mathbf{L}_e^\top \mathbf{L}_e \mathbf{T}(\mathbf{x})^\top \mathbf{T}(\mathbf{x}) \mathbf{L}_e^\top \mathbf{L}_e \, d\mathbf{x} \\ &= \sum_{e=1}^N \mathbf{L}_e^\top \left(\int_{\mathcal{L}^e} \mathbf{T}^e(\mathbf{u})^\top \mathbf{T}^e(\mathbf{u}) \, \text{jacobian}(\Omega^e, \mathbf{u}) \, d\mathbf{u} \right) \mathbf{L}_e \\ &\approx \sum_{e=1}^N \mathbf{L}_e^\top \left(\underbrace{\sum_{i=1}^Q w_i \mathbf{T}^e(\mathbf{u}_i)^\top \mathbf{T}^e(\mathbf{u}_i) \, \text{jacobian}(\Omega^e, \mathbf{u}_i)}_{\mathbf{M}_e} \right) \mathbf{L}_e, \end{aligned} \quad (18)$$

where the first approximation is due to the geometry being approximated by the elements, and the second is due to the quadrature with the Q points $\mathbf{u}_i$ and weights w_i . Furthermore, $\text{jacobian}(\Omega^e, \mathbf{u})$ is a function that describes the distortion from the local coordinates $\mathcal{L}^e$ onto the global element Ω^e . This distortion represents length, area, or volume depending on the dimensionality of the problem. Note that if the elements are the same size then $\mathbf{M}_e$ is constant with respect to each element (this is the case in the code below).

The stiffness matrix is computed similarly but with a difference in the fact that the gradient must have its coordinates changed from global to local. Using the chain rule it follows that

$$\nabla_{\mathbf{u}} p(\mathbf{x}(\mathbf{u})) = \mathbf{J}(\mathbf{u}) \nabla_{\mathbf{x}} p(\mathbf{x}(\mathbf{u})), \quad (19)$$

where $\mathbf{J}(\mathbf{u}) = (\partial \mathbf{x} / \partial \mathbf{u})^\top$, i.e. $J_{ij} = (\partial x_j) / (\partial u_i)$, so that $\text{jacobian}(\Omega^e, \mathbf{u}) = |\det \mathbf{J}(\mathbf{u})|$. From this we can isolate the global gradient as

$$\nabla_{\mathbf{x}} p(\mathbf{x}(\mathbf{u})) = \mathbf{J}(\mathbf{u})^{-1} \nabla_{\mathbf{u}} p(\mathbf{x}(\mathbf{u})). \quad (20)$$

Putting things together the stiffness matrix can therefore be computed as

$$\begin{aligned} \mathbf{K} &= \int_{\Omega} (\nabla \mathbf{T}(\mathbf{x}))^\top \nabla \mathbf{T}(\mathbf{x}) \, d\mathbf{x} \\ &\approx \sum_{e=1}^N \int_{\Omega^e} \mathbf{L}_e^\top \mathbf{L}_e (\nabla_{\mathbf{x}} \mathbf{T}(\mathbf{x}))^\top \nabla_{\mathbf{x}} \mathbf{T}(\mathbf{x}) \mathbf{L}_e^\top \mathbf{L}_e \, d\mathbf{x} \\ &= \sum_{e=1}^N \mathbf{L}_e^\top \left(\int_{\mathcal{L}^e} (\nabla_{\mathbf{x}} \mathbf{T}^e(\mathbf{u}))^\top \nabla_{\mathbf{x}} \mathbf{T}^e(\mathbf{u}) \, \text{jacobian}(\Omega^e, \mathbf{u}) \, d\mathbf{u} \right) \mathbf{L}_e \\ &= \sum_{e=1}^N \mathbf{L}_e^\top \left(\int_{\mathcal{L}^e} (\nabla_{\mathbf{u}} \mathbf{T}^e(\mathbf{u}))^\top (\mathbf{J}(\mathbf{u})^\top)^{-1} \mathbf{J}(\mathbf{u})^{-1} \nabla_{\mathbf{u}} \mathbf{T}^e(\mathbf{u}) \, \text{jacobian}(\Omega^e, \mathbf{u}) \, d\mathbf{u} \right) \mathbf{L}_e \\ &\approx \sum_{e=1}^N \mathbf{L}_e^\top \left(\underbrace{\sum_{i=1}^Q w_i (\nabla_{\mathbf{u}} \mathbf{T}^e(\mathbf{u}_i))^\top (\mathbf{J}(\mathbf{u}_i)^\top)^{-1} \mathbf{J}(\mathbf{u}_i)^{-1} \nabla_{\mathbf{u}} \mathbf{T}^e(\mathbf{u}_i) \, \text{jacobian}(\Omega^e, \mathbf{u}_i)}_{\mathbf{K}_e} \right) \mathbf{L}_e. \end{aligned} \quad (21)$$

In the following code we implement three different ways of assembling the mass and stiffness matrices. Note that the first two implementations are only for educational purposes as they are both based on dense matrices.

Example 4: Matrix assembly implementations

Example 4 (Matrix assembly implementations)

julia

```
1 ### Importing relevant packages
2 using FastGaussQuadrature
3 using BenchmarkTools
4 using LinearAlgebra
5 using SparseArrays
6 using ForwardDiff
7 using Random
8 Random.seed!(42)
9 ## Defining geometry.
10 # Impedance tube of diameter 10 cm and length 1 m (used later)
11 D = 0.1 # 100 mm diameter
12 L = 10*D # Length of the cavity
13 ne = 600 # Number of quadratic elements
14 nnt = 2*ne+1 # Total number of nodes
15 h = L/ne # Length of the elements
```

```

16 x = Vector(0:h/2:L) # Coordinates table
17
18 ## Computing the element matrices
19 # Local basis functions (gradient by ForwardDiff: slow but easy)
20 Te(u) = [u .* (u ./ 2 - 1)/2; 1 ./ -u .^2; u .* (u ./ 2 + 1)/2]'
21 ∇Te(u) = hcat(ForwardDiff.derivative(Te,u)...)
22 # All elements are equal, so the Jacobian is the same for each.
23 # It maps [-1,1] onto [xi,x{i+1}], i.e. length 2 onto length h.
24 jacobian(u) = h/2
25 # In 1D the Jacobian function and matrix are equal (not so in 2D).
26 J(u) = h/2
27 # The element matrices, the same for all elements of equal size.
28 Q = 3 # Number of Gauss points
29 u,w = gausslegendre(Q)
30 Me = sum(i -> w[i]*Te(u[i])'*Te(u[i])*jacobian(u[i]),1:Q)
31 Ke = sum(i -> w[i]*∇Te(u[i])'*J(u[i])^(-1)*J(u[i])^(-1)*∇Te(u[i])*
32     jacobian(u[i]),1:Q)
33
34 ## Assembly 1: with the localization matrices (never do this!)
35 function assembly1(Me, Ke, nnt, ne)
36     K = zeros(nnt, nnt) # Dense matrix! Not ideal!
37     M = zeros(nnt, nnt) # Dense matrix! Not ideal!
38     for ie = 1:ne
39         Le = zeros(3, nnt)
40         Le[:, ie*2-1:ie*2+1] = Diagonal(ones(3))
41         K += Le'*Ke*Le
42         M += Le'*Me*Le
43     end
44     return K, M
45 end
46 @btime K, M = assembly1(Me, Ke, nnt, ne)
47
48 ## Assembly 2: with indexing instead of localization matrices
49 function assembly2(Me, Ke, nnt, ne)
50     K = zeros(nnt, nnt) # Dense matrix! Not ideal!
51     M = zeros(nnt, nnt) # Dense matrix! Not ideal!
52     for ie = 1:ne
53         K[ie*2-1:ie*2+1, ie*2-1:ie*2+1] += Ke
54         M[ie*2-1:ie*2+1, ie*2-1:ie*2+1] += Me
55     end
56     return K, M
57 end
58 @btime K, M = assembly2(Me, Ke, nnt, ne)
59
60 ## Assembly 3: sparse, using the compact support of the elements
61 function assembly3(Me, Ke, nnt, ne)
62     I = zeros{Int64, 4nnt - 3}
63     J = zeros{Int64, 4nnt - 3}
64     Kd = zeros{ComplexF64, length(I)}
65     Md = zeros{ComplexF64, length(I)}
66     for ie=1:ne
67         for i = 1:3
68             r = (8*(ie-1)+1 + 3*(i-1)):(8*(ie-1) + 3*i)
69             I[r] .= ie*2-1:ie*2-1+2
70             J[r] .= (ie-1)*2 + i
71             Kd[r] += Ke[:, i]
72             Md[r] += Me[:, i]
73         end
74     end
75     K = sparse(I, J, Kd)
76     M = sparse(I, J, Md)
77     return K, M
78 end
79 @btime K, M = assembly3(Me, Ke, nnt, ne)

```

5. Examples

Example 5: Impedance tube

Example 5 (Impedance tube) The procedure used in an impedance tube to measure the surface impedance and absorption coefficient of an acoustic material works as follows. A loudspeaker is installed at one end ($x = L$) and an acoustic material of thickness h and surface impedance $\hat{Z}$ is bonded onto the other rigid and impervious end, with the surface of the material at $x = 0$. Below the cut-off frequency of the tube, only plane waves propagate and the problem is amenable to a one-dimensional analysis.

In this case the surface impedance of the material can be obtained from the measurement of the transfer function $\hat{H}_{12} = \hat{p}_2/\hat{p}_1$ between two microphones adequately placed in the tube (e.g., standard ASTM E1050):

$$\frac{\hat{Z}}{\rho_0 c_0} = \frac{1 + R}{1 - R}, \quad R = \frac{\hat{H}_{12} - \exp(-ik_0 s)}{\exp(ik_0 s) - \hat{H}_{12}} \exp(2ik_0(d + s)), \quad (22)$$

where d is the distance from microphone 2 to the sample, s is the spacing between the two microphones, and $k_0 = \omega/c_0$ is the wavenumber of air. The normal-incidence absorption coefficient α is directly obtained from the reflection coefficient R : $\alpha = 1 - |R|^2$.

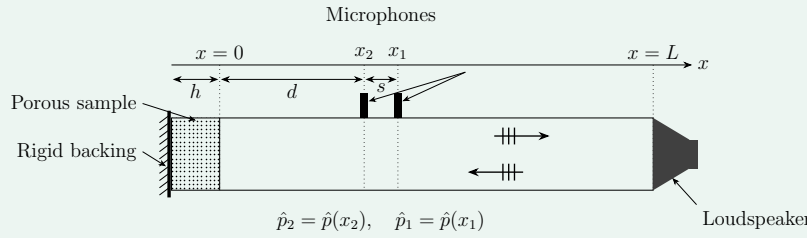


Figure 1: The impedance tube, with the surface of the material at $x = 0$ and the loudspeaker at $x = L$.

The impedance of a material of thickness h bonded onto a rigid wall (at $x = -h$) and excited by a plane wave with incidence angle $\theta = 0$ is given by [1]:

$$\hat{Z} = -i\hat{Z}_c \cot(\hat{k}_c h). \quad (23)$$

In the following the loudspeaker is modeled as a displacement of $U_L = 1$. Furthermore, the specific details of the setup are:

- Diameter of impedance tube $D = 10$ cm (and thus cut-off frequency $1.84c_0/(\pi D) \approx 2004$ Hz).
- Length of impedance tube: $L = 10D$.
- Microphone spacing: $s = D/2$.
- Distance between microphone 2 and the surface of the material: $d = D/2$.
- Thickness of the material: $h = 2$ cm.
- Characteristic impedance $\hat{Z}_c = \rho_0 c_0 [1 + 0.057X^{-0.754} - i0.087X^{-0.732}]$ (where $X = \rho_0 f/\rho_p$).
- Wavenumber $\hat{k}_c = (\omega/c_0)[1 + 0.0978X^{-0.700} - i0.189X^{-0.595}]$ (where $X = \rho_0 f/\rho_p$).
- Density and propagation speed: $\rho_0 = 1.2$ kg/m³, $c_0 = 342.2$ m/s.
- Flow resistivity of the material: $\rho_p = 10000$ Rayls/m.

A 1D model of an impedance tube with a loudspeaker at $x = L$ and a material at $x = 0$ can be described by the following set of equations

$$\begin{cases} \frac{d^2 p}{dx^2}(x) + k_0^2 p(x) = 0, & x \in [0, L] \\ \frac{dp}{dx}(L) = -i\rho_0\omega v_L = \rho_0\omega^2 U_L \\ \frac{dp}{dx}(0) = ik_0\beta p(0) \end{cases} \quad (24)$$

where $\beta = \rho_0 c_0 / \hat{Z}$ is the surface admittance of the material, normalized by the characteristic impedance of air, and U_L is the displacement of the loudspeaker, with velocity $v_L = i\omega U_L$. Here, and in the formulas above, time-harmonic quantities vary as $e^{i\omega t}$.

The PDE is modeled using the FEM as previously

$$(\mathbf{K} - k_0^2 \mathbf{M})\mathbf{p} = \mathbf{0}. \quad (25)$$

However, in this case $\int_{\partial\Omega} \varphi(x) \frac{\partial p}{\partial n}(x) dx \neq 0$. Instead we have

$$\begin{aligned} \left[\varphi(x) \frac{dp}{dx}(x) \right]_0^L &= \mathbf{a}^\top \mathbf{T}(L)^\top \frac{dp}{dx}(L) - \mathbf{a}^\top \mathbf{T}(0)^\top \frac{dp}{dx}(0) \\ &= \mathbf{a}^\top \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \rho_0 \omega^2 U_L - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} ik_0 \beta p(0) \right) \\ &= \mathbf{a}^\top \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \rho_0 \omega^2 U_L - \text{diag} \left(\begin{bmatrix} ik_0 \beta \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right) \mathbf{p} \right). \end{aligned} \quad (26)$$

In the above we utilized the prescribed boundary conditions as well as the definition of $\mathbf{T}(x)$, so that $p(0) = p_1$ is the first entry of $\mathbf{p}$. Putting things together we find that

$$\left(\mathbf{K} - k_0^2 \mathbf{M} + \text{diag} \left(\begin{bmatrix} ik_0 \beta \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right) \right) \mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \rho_0 \omega^2 U_L. \quad (27)$$

Using the previously implemented FEM routines we can numerically solve the problem, and use the analytical expression to check the results, as shown in Figure 2.

```
1 include("assembly_1d.jl")
2 using CairoMakie
3 ## Recomputing FEM matrices
4 K,M = assembly3(Me,Ke,nnt,ne)
5 ## Setup
6 ρ₀ = 1.2 # Fluid density
7 c₀ = 342.2 # Speed of sound
8 U_L = 1 # Piston displacement
9 D = 0.1 # Diameter of Impedance Tube
10 s = 0.5*D # Microphone spacing
11 d = D/2 # Distance between mic 2 and sample
12 ## Frequency domain
13 fc = floor(1.84*c₀/D/pi) # Cut off frequency
14 freq = Vector{Float64}(100:2:fc)
15 ω = 2*pi*freq
16 k₀ = ω/c₀
17 ## Impedance properties
18 Z₀ = ρ₀ * c₀
19 thickness = 0.02 # thickness of the material
20 σ = 10000 # flow resistivity
```

```

21 X = rho*freq/omega
22 Zc = Z0*(1 + 0.057*X.^(-0.754) - im*0.087.*X.^(-0.732))
23 kc = k0.*(1 + 0.0978*X.^(-0.700) - im*0.189.*X.^(-0.595))
24 Z = -im.*Zc.*cot.(kc*thickness) / Z0
25 beta = 1.0 ./ Z # convert to admittance
26 # Finding nodes located on the microphones
27 in2 = findall(x->x,abs.(x.-d) < 1e-6)[1]
28 in1 = findall(x->x,abs.(x.- (d+s)) < 1e-6)[1]
29 # Correcting the s and d distances to fit the numerical values
30 s = abs(x[in1] - x[in2])
31 d = x[in2]
32
33 ## Output
34 ndof = nnt
35 nfreqs = length(omega)
36 P_mic1 = zeros(ComplexF64,nfreqs)
37 P_mic2 = zeros(ComplexF64,nfreqs)
38 A = Diagonal(zeros(ComplexF64,nnt))
39 F = zeros(ComplexF64,ndof)
40
41 ## Frequency sweep
42 for i in eachindex(omega)
43     A[1,1] = im*k0[i]*beta[i]
44     F[end,1] = rho*omega[i]^2*U_L
45     S = K - k0[i]^2*M + A
46     p = S\F
47     P_mic2[i] = p[in2]
48     P_mic1[i] = p[in1]
49 end
50
51 # Calculate the normalized impedance
52 H12 = P_mic2 ./ P_mic1
53 R = (H12 - exp.(-im*k0*s))./(exp.(im*k0*s) - H12) .*
54     exp.(im*2*k0 * (d + s))
55 Z_num = (1 + R) ./ (1 - R)
56 ## Comparison with the exact solution
57 fig = Figure(size=(800,320))
58 ax_re = Axis(fig[1,1],xlabel="Frequency (Hz)",ylabel="Real part")
59 lines!(ax_re,freq,real.(Z),label="Analytical")
60 lines!(ax_re,freq,real.(Z_num),label="FEM",linestyle=:dash)
61 ax_im = Axis(fig[1,2],xlabel="Frequency (Hz)",ylabel="Imaginary part")
62 lines!(ax_im,freq,imag.(Z))
63 lines!(ax_im,freq,imag.(Z_num),linestyle=:dash)
64 axislegend(ax_re)
65 fig

```

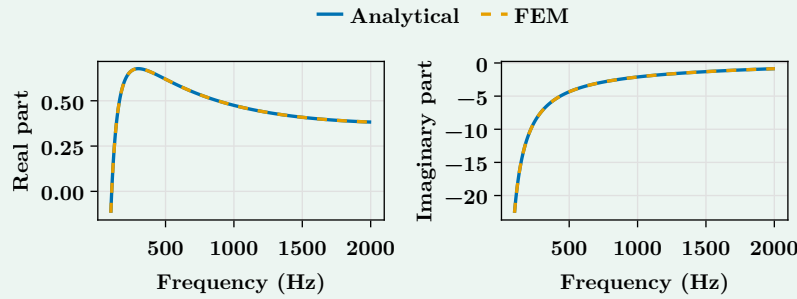


Figure 2: The normalized surface impedance $\hat{Z}/(\rho_0 c_0)$ of the material, analytical and computed with the FEM.

Example 6 (Modes of a 10×4 m room) The Finite Element Method can also be used to compute the modes, i.e. finding the eigenvalues ω^2 of the generalized eigenvalue problem

$$c^2 \mathbf{K} \mathbf{p} = \omega^2 \mathbf{M} \mathbf{p}. \quad (28)$$

For a rectangular room with side lengths L_x and L_y , here $L_x = 10$ m and $L_y = 4$ m, the frequencies of the modes can be computed analytically as

$$f_{n_x n_y} = \frac{c}{2} \sqrt{\left(\frac{n_x}{L_x}\right)^2 + \left(\frac{n_y}{L_y}\right)^2}. \quad (29)$$

The room is meshed with a regular grid of bilinear quadrilateral elements:

julia

```
1 using LinearAlgebra, CairoMakie
2 ## Defining geometry.
3 Lx = 10
4 Ly = 4
5 n = 20+1
6 m = 8+1
7 ne = (n-1)*(m-1) # Number of linear elements
8 nnt = n*m         # Total number of nodes
9 ## Plotting the elements
10 x = Vector{0:Lx/(n-1):Lx}
11 y = Vector{0:Ly/(m-1):Ly}
12 X = kron(x,ones(m))
13 Y = kron(ones(n),y)
14 scatter(X,Y,axis=(aspect=DataAspect(),))
```

The Finite Element computation for the 2D room looks similar to the one for the 1D impedance tube. One difference is the reliance on the topology (connectivity) matrix, which lists the nodes of each element. The end of the script computes the analytical frequencies for comparison:

julia

```
1 include("modes_geometry.jl")
2 using ForwardDiff, SparseArrays, LinearAlgebra
3 using FastGaussQuadrature, Arpack
4 c0 = 342.2
5 ## Creating the topology
6 T = reshape(1:nnt,m,n) # Numbering all nodes on the grid
7 topology = zeros{Int,4,ne}
8 topology[1,:] = T[1:end-1,1:end-1][:]
9 topology[2,:] = T[1:end-1,2:end][:]
10 topology[3,:] = T[2:end,2:end][:]
11 topology[4,:] = T[2:end,1:end-1][:]
12 ## Computing Element matrices
13 # Local basis functions (gradient by ForwardDiff: slow but easy)
14 T^e(u) = [(1-u[1])*(1-u[2]); (1+u[1])*(1-u[2]);
15           (1+u[1])*(1+u[2]); (1-u[1])*(1+u[2])]'/4
16 ∇T^e(u) = hcat(ForwardDiff.jacobian(T^e,u))'
17 # All elements are equal, so the Jacobian is the same for each.
18 jacobian(u) = (x[2]-x[1])/2*(y[2]-y[1])/2
19 J(u) = Diagonal([x[2]-x[1];y[2]-y[1]])/2
20 # Defining the local element matrices.
21 Q = 2
22 u,wu = gausslegendre(Q)
23 v,wv = gausslegendre(Q)
24 U = kron(u,ones(Q))
25 V = kron(ones(Q),v)
26 W = kron(wu,wv)
27 P = [U';V']
28
```

```

29 Me = sum(i -> W[i]*jacobian(P[:,i])*T^e(P[:,i])*T^e(P[:,i]),1:Q^2)
30 Ke = sum(i -> W[i]*jacobian(P[:,i])*VT^e(P[:,i])*
31     J(P[:,i])^(-1)*J(P[:,i])^(-1)*VT^e(P[:,i]),1:Q^2)
32
33 ## Assembly
34 function connected_topology(topology,nnt,ne)
35     source_connections = [zeros(Int,0) for _ in 1:nnt]
36     for element = 1:ne
37         for i = 1:4
38             node = topology[i,element]
39             append!(source_connections[node],topology[:,element])
40         end
41     end
42     sort!.(unique.(source_connections))
43 end
44 function create_I_J(connections)
45     I = zeros(Int,sum(length.(connections)))
46     J = zeros(Int,sum(length.(connections)))
47     lower = 1
48     for (idx,con) in enumerate(connections)
49         upper = lower + length(con)
50         I[lower:upper-1] .= con
51         J[lower:upper-1] .= idx
52         lower = upper
53     end
54     return I,J
55 end
56 function assembly(Me,Ke,topology,nnt)
57     ne = size(topology,2)
58     connections = connected_topology(topology,nnt,ne)
59     I,J = create_I_J(connections)
60     S = sparse(I,J,1:length(I))
61     Kd = zeros(length(I))
62     Md = zeros(length(I))
63     for ie=1:ne
64         top = topology[:,ie]
65         Kd[S[top,top]] += Ke
66         Md[S[top,top]] += Me
67     end
68     K = sparse(I,J,Kd)
69     M = sparse(I,J,Md)
70     return K,M
71 end
72 K,M = assembly(Me,Ke,topology,nnt);
73
74 using Arpack
75 F = eigs(c_0^2*K,M,which=:SM,nev=10);
76 freq = sort(sqrt.(abs.(F[1]))/2/pi);
77
78 ## The analytical frequencies, for comparison
79 nx, ny = 0:4, 0:4
80 freq_analytical = c_0/2*sqrt.((nx/Lx).^2 .+ (ny'/Ly).^2)
81 analytical_freqs = sort(freq_analytical[:])
82 println("FEM:      ", round.(freq[2:10],digits=2))
83 println("Analytical: ", round.(analytical_freqs[2:10],digits=2))

```

As seen in Figure 3, the FEM frequencies lie slightly above the analytical ones, and the difference grows with the frequency, as the mesh resolves the shorter wavelengths of the higher modes less accurately.

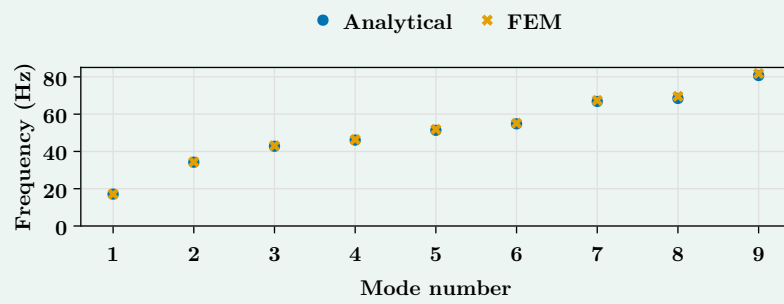


Figure 3: The first nine nonzero mode frequencies of the 10×4 m room, analytical and computed with the FEM.

Bibliography

- [1] J.-F. Allard and N. Atalla, *Propagation of Sound in Porous Media: Modelling Sound Absorbing Materials*, 2nd ed. Wiley, 2009. doi: 10.1002/9780470747339.