Reduced Basis Method With Time-Deforming Domains
MWE: Linear Heat Equation

Enrique Millán Valbuena
463 426 8

Abstract

The research objective is to build a Reduced Order Model (ROM) for a parametrized one-dimensional heat equation with a deforming boundary.

The main goal is to have a simple problem to define and tackle all the implementation details of the reduction procedure, to later on solve an actual problem with industrial or clinical application.

Both the main body of the PDE, the geometrical definition of the boundary and the boundary conditions will be parametrized.

Index Terms

Reduced Order Model, Moving Domain, FEM, DEIM, POD, Galerkin-Projection

CONTENTS

1 FULL ORDER MODEL: HEAT EQUATION

The Full Order Model for the parametrized one-dimensional linear heat equation is derived. Due to its simplicity, this problem is not the focus of the thesis, yet it serves well to set a benchmark and test our ideas and implementations without the noise and effort required by non-linear terms.

The model will be derived in the continuous, semi-discrete and fully discrete contexts for a generic parametrization and forcing term. We shall use the Galerkin projection principle to find a weak form, which we later discretize using the Finite Element Method. At the end of the document two specific problems will be presented, to implement and test the solver and reduction algorithms.

If the variational or Finite Element problem is not correctly stated in formal terms yet, we apologize in advance. These kind of language formalities take time to settle and this is still a draft. The reader will be able to fill in any notational or definition gaps for the moment.

We define the vector $\mu \in \mathcal{P}$ to collect all the parameters present in the formulation. Parameters can be present in the PDE's body, in the boundary conditions, or in the geometrical definition of the domain. We shall define our problem within a deforming domain in time, whose movement is known and not part of the solution,

$$\Omega(t,\mu) := \{x \in \mathbb{R} : x \in [0, L(t,\mu)]\}.$$

The function $L(t,\mu)$ is a real-valued smooth function in the time and parameter space. From now on, we drop the dependency on time and the parameters unless it is strictly necessary.

1.1 Continuous Problem

We start with the definition of the problem in the continuous setting: a differential model given by a PDE, referred to as *strong formulation*; and its weak formulation derived with the Galerkin principle.

1.1.1 Strong Formulation

The differential model that captures heat transfer for a function $u = u(x,t)$ is given by the following PDE, boundary and initial conditions,

$$\left.\frac{\partial u}{\partial t}\right|_x - \alpha \Delta u = f(x,t;\mu), \tag{1a}$$
$$u(0,t) = b_0(t;\mu), \tag{1b}$$
$$u(L,t) = b_L(t;\mu), \tag{1c}$$
$$u(x,0) = u_0(x;\mu), \tag{1d}$$

where we assume all the terms to be present in their non-dimensional form with respect to the original problem. The notation $\left.\frac{\partial u}{\partial t}\right|_x$ indicates that the derivative takes place in the physical moving domain.

We have the boundary conditions and the forcing term to depend on time and the parameter vector without loss of generality. Again, from now on we drop the dependency to have a clear notation.

For the one-dimensional scenario, the Laplacian operator reduces to the simple expression of the second derivative,

$$\Delta u = \frac{\partial^2 u}{\partial x^2}. \tag{2}$$

1.1.2 ALE Formulation

Despite the fact that we will be solving the problem in the physical moving domain, we still need the two basic ingredients stemming at the root of the ALE method:

- A smooth mapping between domains.
- A mesh velocity vector.

We introduce the ALE mapping $\mathcal{A}$ that connects a point in the fixed reference domain $\mathcal{X}$ with a point in the physical domain x:

$$x = \mathcal{A}(\mathcal{X}, t), \tag{3a}$$
$$\mathcal{X} = \mathcal{A}^{-1}(x, t), \tag{3b}$$

which we assume to be regular enough.

We define the mesh velocity as the time derivative of the spatial coordinate, which will coincide with the time derivative of the ALE map:

$$w(x,t) = \frac{\partial x}{\partial t} = \frac{\partial \mathcal{A}}{\partial t}(x,t) = \frac{\partial \mathcal{A}}{\partial t}(\mathcal{A}^{-1}(\mathcal{X},t),t). \tag{4}$$

Time-Derivative in the Reference Domain

If the equations are going to be solved in the physical domain, we only need to adapt the time derivative in the physical domain $\left.\frac{\partial u}{\partial t}\right|_x$. By application of the chain rule we get

$$\left.\frac{\partial u}{\partial t}\right|_{\mathcal{X}} = \left.\frac{\partial u}{\partial t}\right|_x + w\frac{\partial u}{\partial x}, \tag{5}$$

from where we get the necessary modification to be done to the strong form (1a) of the PDE,

$$\left.\frac{\partial u}{\partial t}\right|_{\mathcal{X}} - w\frac{\partial u}{\partial x} - \alpha\Delta u = f \tag{6}$$

Geometric Conservation Laws

1.1.3 Weak Formulation

Since we will be working with the Galerkin procedure to solve PDEs, we define the $L^2(\Omega)$ inner product to transform the strong formulation into a weak, variational one,

$$\langle u, v\rangle_{(t,\mu)} = \int_{\Omega(t,\mu)} uv \, d\Omega. \tag{7}$$

This inner product induces the so called *eyeball* norm, since it checks if two functions look alike,

$$\|f - g\|_{(t,\mu)} = \sqrt{\int_{\Omega(t,\mu)} (f-g)^2 \, d\Omega}. \tag{8}$$

Eventually, we will also be interested in other norms, such as the $H^1(\Omega)$ norm, which captures the differences in the gradients too. This is important when computing stress-related values from the solved field.

With this inner product, we project the residual of the strong formulation onto a given fuction $v \in V$, where V is a suitable Hilbert space,

$$\left\langle \frac{\partial u}{\partial t}, v \right\rangle - \langle w\boldsymbol{\nabla} u, v\rangle + \langle \alpha \boldsymbol{\nabla} u, \boldsymbol{\nabla} v\rangle = \langle f, v\rangle \tag{9a}$$
$$u(0,t) = b_0(t), \tag{9b}$$
$$u(L,t) = b_L(t), \tag{9c}$$
$$u(x,0) = u_0(x). \tag{9d}$$

We have exploited the integration by parts rule to lower by one the derivative degree of the laplacian operator.

1.1.4 Dirichlet Lifting

For reasons that will become apparent later, it is preferable to work with a homogeneous problem in the Dirichlet boundary conditions.

To obtain so, we introduce a *lifting* $g(x,t)$ of the Dirichlet boundary conditions. We express the solution of our problem like the linear combination of the solution of the homogeneous problem and the lifting function:

$$u(x,t) = \hat{u}(x,t) + g(x,t). \tag{10}$$

There are two conditions to be met by the lifting of the boundary conditions:

1) To reach the prescribed values at the boundary nodes.
2) To be sufficiently smooth within the domain.

In a one-dimensional setting, the definition of a lifting function $g(x,t)$ is straigtforward, with a simple linear interpolation of the boundary values:

$$g(x,t) = b_L(t)\left(\frac{x}{L}\right) + b_0(t)\left(\frac{L-x}{L}\right). \tag{11}$$

In higher dimensional settings this procedure is valid too. However, due to the arbitrary shape the domain can take, the construction of the lifting function becomes more laborious. We shall skip that for the moment, since we are dealing with a one-dimensional problem, where we can build the extension of the boundary conditions analytically.

Introducing the lifting breakdown (10) into the weak formulation (9a), we find an additional forcing term,

$$\left\langle \frac{\partial \hat{u}}{\partial t}, v \right\rangle - \langle w\boldsymbol{\nabla}\hat{u}, v\rangle + \langle \alpha \boldsymbol{\nabla}\hat{u}, \boldsymbol{\nabla} v\rangle = \langle f, v\rangle + \langle f_{g,1}, v\rangle + \langle f_{g,2}, \boldsymbol{\nabla} v\rangle, \tag{12a}$$

$$f_{g,1} = -\frac{\partial g}{\partial t} + w\frac{\partial g}{\partial x}, \tag{12b}$$

$$f_{g,2} = -\alpha\frac{\partial g}{\partial x}, \tag{12c}$$

and the homogenization of the boundary conditions for all time t,

$$\hat{u}(0,t) = 0, \tag{13a}$$

$$\hat{u}(L,t) = 0, \tag{13b}$$

$$\hat{u}(x,0) = \hat{u}_0(x), \tag{13c}$$

The initial condition should be modified accordingly to the homogeneous problem definition,

$$u(x,0) = \hat{u}(x,0) + g(x,0), \tag{14a}$$

$$\hat{u}_0(x) := \hat{u}(x,0) = u(x,0) - g(x,0). \tag{14b}$$

At this point, we have defined the continuous problem of the heat equation in a one-dimensional moving domain.

1.2 Semi-Discrete Problem

The continuous solution changes in two directions: space and time. Eventually, we need to discretize both, but we can do it incrementally, so first, we discretize in time. To do so, we need to build an approximation for the time derivate, $\frac{\partial \hat{u}}{\partial t}$, and there is a complete body of knowledge devoted to this step of the numerical scheme CITE. Briefly, there is a trade-off between computational effort and stability in terms of the resulting matrices. For our needs, we opt for the maximization of stability, so we chose what is called an implicit scheme, which is unconditionally stable, in exchange for a more dense linear system to be solved.

The procedure is the following: a polynomial interpolation of the function $u(x,t)$ is built, using previous timesteps $u(x,t^n), u(x,t^{n-1}), \ldots$. Then, the interpolant's derivative at time t^{n+1} is used as an approximation of the actual derivative. The accuracy of the scheme (and also its complexity) is determined by the number of previous timestamps used in the interpolation.

In the coming sections, we define two implicit schemes of order one and two

1) BDF-1: also known as Backwards Euler.
2) BDF-2: also known as ...

Throughout our simulations we will use the BDF-2 scheme, but since it uses two points from the past, it cannot be used at the beginning of the simulation, in the calculation of the first step. That is where the BDF-1 comes into play, to obtain $u(x,t^1)$. Additionally, studying the BDF-1 scheme is still useful, to understand the implications of discretizing in time.

As a final note, we point out that according to each problem needs or complexities, sometimes a convergent scheme can be obtained even if certain equation terms are treated explicitly. This trick can be used in problems such as the Navier-Stokes equations, where going fully implicit leads to a non-linear algebraic system. Instead, if the convective term is treated semi-implicitly, one recovers a linear system, and yet reaches a satisfactory numerical solution. In terms of notation, from now on we use

$$\hat{u}^n := \hat{u}(x,t^n) \tag{15}$$

to define a function in space evaluated at time t^n.

1.2.1 BDF-1

To derive the BDF-1 scheme, we start by evaluating our weak formulation at time t^{n+1}, where the solution is unknown,

$$\left\langle \frac{\partial \hat{u}}{\partial t}, v \right\rangle^{n+1} - \langle w\boldsymbol{\nabla}\hat{u}, v\rangle^{n+1} + \langle \alpha \boldsymbol{\nabla}\hat{u}, \boldsymbol{\nabla} v\rangle^{n+1} = + \langle f, v\rangle^{n+1} + \langle f_{g,1}, v\rangle^{n+1} + \langle f_{g,2}, \boldsymbol{\nabla} v\rangle^{n+1}. \tag{16}$$

Then, we construct our interpolation polynomial using only one previous timestamp from the past,

$$\hat{u}(x,t) \simeq I_1(t) := \hat{u}^{n+1}\left(\frac{t-t^n}{\Delta t}\right) + \hat{u}^n\left(\frac{t^{n+1}-t}{\Delta t}\right), \tag{17}$$

$$\left.\frac{\partial \hat{u}}{\partial t}\right|^{n+1} \simeq \frac{dI_1(t)}{dt} = \frac{\hat{u}^{n+1}-\hat{u}^n}{\Delta t} \tag{18}$$

With this approximation of the time derivative, we get the following semi-discrete weak formulation,

$$\begin{aligned}\left\langle \hat{u}^{n+1}, v\right\rangle - \Delta t \left\langle w\boldsymbol{\nabla}\hat{u}^{n+1}, v\right\rangle + \Delta t \left\langle \alpha\boldsymbol{\nabla}\hat{u}^{n+1}, \boldsymbol{\nabla}v\right\rangle = \\ \left\langle \hat{u}^{n}, v\right\rangle + \Delta t \left\langle f^{n+1}, v\right\rangle \\ +\Delta t \left\langle f_{g,1}^{n+1}, v\right\rangle + \Delta t \left\langle f_{g,2}^{n+1}, \boldsymbol{\nabla}v\right\rangle,\end{aligned} \tag{19a}$$

$$\hat{u}^{n+1}(0) = 0, \tag{19b}$$

$$\hat{u}^{n+1}(L\left(t^{n+1}\right)) = 0, \tag{19c}$$

$$\hat{u}^{0}(x) = \hat{u}_0(x). \tag{19d}$$

The forcing term $\langle \hat{u}^n, v\rangle$ is an interesting one. This inner product is done at time t^{n+1}, but the solution $\hat{u}^n$ belongs to the previous time. In a way, it is a sort of L^2 projection of the previous solution into the current domain. DEVELOP.

Note that the problem is still continuous in space, but no longer in time.

1.2.2 BDF-2

To derive the BDF-2 scheme, again, we start by evaluating our weak formulation at time t^{n+1}, where the solution is unknown,

$$\begin{aligned}\left\langle \frac{\partial \hat{u}}{\partial t}, v\right\rangle^{n+1} + \langle \alpha(x)\boldsymbol{\nabla}\hat{u}, \boldsymbol{\nabla}v\rangle^{n+1} = \langle f, v\rangle^{n+1} \\ + \langle f_{g,1}, v\rangle^{n+1} + \langle f_{g,2}, \boldsymbol{\nabla}v\rangle^{n+1}.\end{aligned} \tag{20}$$

Then, we construct our interpolation polynomial using two previous timestamps from the past,

$$\hat{u}(x,t) \simeq I_2(t) := \alpha_1\hat{u}^{n+1} + \alpha_2\hat{u}^{n} + \alpha_3\hat{u}^{n-1}, \tag{21}$$

$$\left.\frac{\partial \hat{u}}{\partial t}\right|^{n+1} \simeq \frac{dI_2(t)}{dt} = \frac{\frac{3}{4}\hat{u}^{n+1} - \frac{1}{2}\hat{u}^{n} + \frac{1}{4}\hat{u}^{n-1}}{\Delta t} \tag{22}$$

With this approximation of the time derivative, we get the following semi-discrete weak formulation,

$$\begin{aligned}\frac{3}{4}\left\langle \hat{u}^{n+1}, v\right\rangle + \Delta t \left\langle \alpha(x)\boldsymbol{\nabla}\hat{u}^{n+1}, \boldsymbol{\nabla}v\right\rangle = \\ \frac{1}{2}\left\langle \hat{u}^{n}, v\right\rangle - \frac{1}{4}\left\langle \hat{u}^{n-1}, v\right\rangle \\ +\Delta t \left\langle f^{n+1}, v\right\rangle \\ +\Delta t \left\langle f_{g,1}^{n+1}, v\right\rangle + \Delta t \left\langle f_{g,2}^{n+1}, \boldsymbol{\nabla}v\right\rangle,\end{aligned} \tag{23a}$$

$$\hat{u}^{n+1}(0) = 0, \tag{23b}$$

$$\hat{u}^{n+1}(L\left(t^{n+1}\right)) = 0, \tag{23c}$$

$$\hat{u}^{0}(x) = \hat{u}_0(x). \tag{23d}$$

Note how the only changes between both discretizations are the forcing terms due to the previous solutions and the leading coefficient multipliying the inner product of the desired unknown solution $\hat{u}^{n+1}$.

1.3 Discrete Problem

To complete our discretization, we define a finite functional space $V_h \subset V$, where we can represent the solution as the linear combination of a set of Finite Elements (FE) basis functions $\varphi_i(x)$ with local support,

$$\hat{u}^n(x) \simeq \hat{u}_h^n(x) = \sum_j^{N_h} \hat{u}_{h_i}^n \varphi_j(x), \tag{24}$$

$$\hat{\mathbf{u}}_h^n = \left[\hat{u}_{h_i}^n\right]. \tag{25}$$

In this discrete setting, we define the FE vector $\hat{\mathbf{u}}_h^n$ to be the collection of coefficients $[\hat{u}_{h_i}^n]$ which multiply the basis functions. In the FE context, these coincide with the values of the function at each mesh node.

Applying the Galerkin principle to solve PDEs, we enforce the orthogonality of the residual to the functional space V_h. This is equivalent to enforcing the orthogonality to each basis function of the space, so we get an algebraic system with the same number of unknowns as equations, leading to the following algebraic system:

$$\begin{aligned}\mathbf{M}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} + \Delta t\mathbf{A}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} = \mathbf{F}_{\hat{\mathbf{u}}_h}^n \\ + \Delta t\left(\mathbf{F}_h^{n+1} + \mathbf{F}_{g,h}^{n+1}\right),\end{aligned} \tag{26a}$$

$$\hat{\mathbf{u}}_h^0 = \hat{\mathbf{u}}_{h,0}. \tag{26b}$$

The spatial boundary conditions are encoded within the matrices and the vectors. The computation of the discrete initial condition $\hat{\mathbf{u}}_{h,0}$ is addressed in Section 1.3.1.

If we collect terms and factor out the unknowns, we get a linear system to be solved at each timestep to advance the solution,

$$\mathbf{K}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} = \mathbf{b}_h^{n+1}, \tag{27a}$$

$$\mathbf{K}_h^{n+1} = \mathbf{M}_h^{n+1} + \Delta t\mathbf{A}_h^{n+1}, \tag{27b}$$

$$\mathbf{b}_h^{n+1} = \mathbf{F}_{\hat{\mathbf{u}}_h}^n + \Delta t\left(\mathbf{F}_h^{n+1} + \mathbf{F}_{g,h}^{n+1}\right), \tag{27c}$$

$$\hat{\mathbf{u}}_h^0 = \hat{\mathbf{u}}_{h,0}. \tag{27d}$$

Because the domain changes with time, even if we are solving a linear system, both the matrix and the RHS change at each timestep,

$$\left[\mathbf{M}_h^{n+1}\right]_{ij} = m_{\text{BDF}}\left\langle \varphi_j, \varphi_i\right\rangle \tag{28a}$$

$$\left[\mathbf{A}_h^{n+1}\right]_{ij} = -\left\langle w\boldsymbol{\nabla}\varphi_j, \varphi_i\right\rangle + \left\langle \alpha\boldsymbol{\nabla}\varphi_j, \boldsymbol{\nabla}\varphi_i\right\rangle \tag{28b}$$

$$\left[\mathbf{F}_h^{n+1}\right]_i = \left\langle f^{n+1}, \varphi_i\right\rangle, \tag{28c}$$

$$\left[\mathbf{F}_{g,h}^{n+1}\right]_i = \left\langle f_{g,1}^{n+1}, \varphi_i\right\rangle + \left\langle f_{g,2}^{n+1}, \boldsymbol{\nabla}\varphi_i\right\rangle, \tag{28d}$$

The assembly of the forcing terms given in Equations (28c) and (28d) are to be done with the FE vector representations of the functional expressions of each of the terms f and f_g, to be obtained by projection or interpolation, as explained in Section 1.3.1.

The change in time is implied in the inner product, which is defined for a domain that changes in time. This fact makes the integration in time quite a costly operation.

Finally, the parameter m_{BDF} changes according to the integration scheme used,

$$m_{\text{BDF}} = \begin{cases}1, & \text{BDF-1} \\ \frac{3}{4}, & \text{BDF-2}\end{cases} \tag{29}$$

as so does the vector linked to the previous timesteps,

$$\left[\mathbf{F}_{\hat{\mathbf{u}}_h}^n\right]_i = \begin{cases}\left\langle \hat{u}_h^n, \varphi_i\right\rangle, & \text{BDF-1} \\ \frac{1}{2}\left\langle \hat{u}_h^n, \varphi_i\right\rangle - \frac{3}{4}\left\langle \hat{u}_h^{n-1}, \varphi_i\right\rangle, & \text{BDF-2}\end{cases} \tag{30}$$

Although for the FOM model we can compute these inner products at each timestep, for the Reduced Order Model we will exploit an algebraic expression of these expressions. In fact, the forcing term due to the previous solution can be expressed as the product between the mass matrix and the FE representation of the previous solution,

$$\mathbf{F}^n_{\hat{\mathbf{u}}_h} = \begin{cases} \mathbf{M}_h^{n+1}\hat{\mathbf{u}}_h^n, & \text{BDF-1} \\ \frac{1}{2}\mathbf{M}_h^{n+1}\hat{\mathbf{u}}_h^n - \frac{3}{4}\mathbf{M}_h^{n+1}\hat{\mathbf{u}}_h^{n-1}, & \text{BDF-2} \end{cases} \tag{31}$$

Lastly, we point out how the timestep Δt has been intentionally left out of the discrete operators definition. There are two reasons to back this decision:

1) Conceptually, each discrete operator encodes a spatial model, in terms of a differential operator or the presence of a forcing term. The timestep Δt shows up because we first discretized the continuous problem in time. Had we gone the other way around, discretizing the problem in space in the first place, we would have found a system of ODEs with the previously defined spatial operators.
2) When we leverage the system approximation reduction technique, we will not want to have there the presence of the timestep. The reduced model could use a different timestep, or the snapshots for different μ values could be collected for different timestep values.

1.3.1 FE Representation of the Initial Condition

The initial condition in time $\hat{u}_0(x)$ needs to be expressed in terms of the basis functions $\varphi_i(x)$ to obtain the FE vector $\hat{\mathbf{u}}_{h,0}$. There are two ways to compute the values of this FE vector:

- Interpolation.
- Projection.

Interpolation

Since we are dealing with a nodal basis[1], the interpolation procedure simply assigns to each vector entry the value of the function at the node corresponding to that entry.

In a sense, it is what we have done with the lifting of the boundary conditions in Section 1.1.4. Since we are dealing with a nodal basis, we can express the lifting FE vector $\mathbf{g}_h$ as a linear combination of the FE basis functions,

$$g(x,t) \simeq g_h(x,t) = \sum_i^{N_h} g_{h_i}(t)\varphi_i(x), \tag{32}$$

where each coefficient $\mathbf{g}_h = [g_{h_i}]$ is the value taken by the analytical interpolation of the boundary conditions given in Equation (11).

1. We recall a nodal basis is one where each node of the mesh has a basis function assigned, such that the value of the coefficient which multiplies each basis function corresponds to the value of the function at that node.

Projection

Instead, the projection procedure aims at solving the approximation problem

$$\hat{u}_h^0(x) = \sum_j^{N_h} \hat{u}_{h_i}^0 \varphi_j(x) = \hat{u}_0(x) \tag{33}$$

with a variational approach within V_h. That is, the approximation error is set to be orthogonal to the function space V_h, yielding the weak formulation

$$\left\langle \hat{u}_h^0(x), v \right\rangle = \left\langle \hat{u}_0(x), v \right\rangle, \forall\, v \in V_h, \tag{34}$$

which leads to a linear system of equations if we enforce the orthogonality to each basis function,

$$\mathbf{M}_h \hat{\mathbf{u}}_{h,0} = \mathbf{p}_h, \tag{35a}$$
$$[\mathbf{M}_h]_{ij} = \left\langle \varphi_j, \varphi_i \right\rangle, \tag{35b}$$
$$[\mathbf{p}_h]_i = \left\langle \hat{u}_0(x), \varphi_i \right\rangle. \tag{35c}$$

In practice, the projection method is to be used, because even within the FE context we might not have a nodal basis (this would be the case if exotic function spaces are used).

However, when the Method of Manufactured Solutions is being used to certify the code implementation, only the interpolation procedure will allow us to reach machine accuracy when computing the error.

1.3.2 Conclusions

With all of the above, the generic Finite Element Method for the one-dimensional linear heat equation with a time-deforming domain is defined.

The problem has been explained within three levels of abstraction: continuous with strong and weak formulations, semi-discrete in time and fully discretized in time and space. An implicit single-step time discretization scheme is used, to obtain an inconditionally stable algebraic system.

A lifting of the Dirichlet boundary conditions has been introduced, which lead to the appearence of an additional forcing term and the cancellation of the boundary conditions. Thus, we will focus in the solution of an homogeneous boundary value problem. This fact will prove useful when we get into the implementation details of the reduction procedure.

1.4 Deformation of the Space-Time Domain

TBD.

Comment on how in the (x,t) domain we actually have a deforming domain.

This could be solved with space-time FE, but it involves complex tensor products.

Explain how what we have done is a good approximation of the latter.

1.5 Generalized Transformation

If the domain is made fixed, how does the equation change? What would be the expression of the algebraic system?

2 REDUCED ORDER MODEL: HEAT EQUATION

We present now the formulation for the Reduced Order Model (ROM) of our FE problem. Our aim is to be able to, for any unseen parameter value, assemble and solve a smaller problem in terms of the involved algebraic operators dimensions. In order to do so, first we need to obtain certain algebraic structures which capture the essence of the problem at hand. We do so by sampling the original problem at certain parameter values and processing snaphsots of the solution and the operators, obtained from the solution of the FOM problem. This is called the *offline phase*.

Later on, we exploit these static structures to build reduced operators which capture the dynamics of the problem sufficiently well, solve a smaller algebraic system and then recover the solution in the original mesh variables, in order to postprocess it. This is called the *online phase*.

We will use the Reduced Basis Method (RB) to construct an ad-hoc problem-based basis to represent our ROM, and the Discrete Empirical Interpolation Method (DEIM) and its matrix version (MDEIM) to build suitable approximations of the algebraic operators involved.

The continuous reduced problem formulation is skipped since it will not be used. Therefore, we jump directly into the discrete problem. We recall that we are focused at reducing the problem for the homogeneous component $\hat{u}(x)$ of our solution, i.e., the weak formulation given by the system of equations (12).

2.1 Reduced Basis Method: A Naive Galerkin Approach

We have included in the title the word *naive* because we are going to define the reduction problem in a very blunt way, where many ineffiencies will show up. We do so to motivate the additional reduction procedures we will include later on, specially regarding the operators.

In the reduced context, we use a finite space $V_N \subset V_h$, where we can represent the solution as the linear combination of a set of orthonormal[2] problem-based basis functions $\psi_i(x)$,

$$\hat{u}_N(x) = \sum_j^N \hat{u}_{N_j} \psi_j(x). \tag{36}$$

These basis functions $\psi_i(x)$ have global support, since their goal is to capture the problem dynamics. This is why we usually expect or desire to have $N \ll N_h$, since we want to reduce the number of basis functions we need to represent our solution.

To maintain focus and in favor of generality, let us assume at this point that the global basis functions are given, and that they are zero at the boundary. We will give details on how to obtain them later on.

2.1.1 Reduced Space Projection

Since we can represent any function in terms of our FE basis functions $\varphi_i(x)$, we have a linear mapping between the problem-based functions and the nodal basis functions. This allows us to establish the following relation between the problem solution in the reduced and original spaces,

$$\hat{\mathbf{u}}_h = \mathbb{V}\hat{\mathbf{u}}_N. \tag{37}$$

2. If they were not, we can always make them so via Gram-Schmidt.

The entries of the $\mathbb{V}$ matrix represent the coefficients of the global basis representation in the FE basis.

$$\psi_i(x) = \sum_j [\mathbb{V}]_{ji} \varphi_j(x) \tag{38}$$

2.1.2 Discrete Reduced Problem Assembly

In theory, to assemble the reduced problem operators, one could actually compute the inner products defined in Equations (28) with these new basis functions $\psi_i(x)$. In practice, it is more convenient to project the algebraic FOM operators with matrix-matrix and matrix-vector products into the reduced space,

$$\mathbf{M}_N^{n+1} = \mathbb{V}^T \mathbf{M}_h^{n+1} \mathbb{V}, \tag{39a}$$
$$\mathbf{A}_N^{n+1} = \mathbb{V}^T \mathbf{A}_h^{n+1} \mathbb{V}, \tag{39b}$$
$$\mathbf{F}_N^{n+1} = \mathbb{V}^T \mathbf{F}_h^{n+1}, \tag{39c}$$
$$\mathbf{F}_{g,N}^{n+1} = \mathbb{V}^T \mathbf{F}_{g,h}^{n+1}, \tag{39d}$$

since the assembly of matrices based on the FE basis is easily done in parallel due to their local support, whereas the integration of functions with global support is not necessarily computationally efficient.

By doing so, we find the following ROM for the time evolution problem,

$$\mathbf{M}_N \hat{\mathbf{u}}_N^{n+1} + \Delta t \mathbf{A}_N \hat{\mathbf{u}}_N^{n+1} = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t \mathbf{F}_N^{n+1} + \Delta t \mathbf{F}_{g,N}^{n+1}, \tag{40a}$$
$$\hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}. \tag{40b}$$

If we collect terms and factor out the unknowns we get a linear system, this time in the reduced space, to be solved for each timestep to advance the solution,

$$\mathbf{K}_N^{n+1} \hat{\mathbf{u}}_N^{n+1} = \mathbf{b}_N^{n+1}, \tag{41a}$$
$$\mathbf{K}_N^{n+1} = \mathbf{M}_N^{n+1} + \Delta t \mathbf{A}_N^{n+1}, \tag{41b}$$
$$\mathbf{b}_N^{n+1} = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t \mathbf{F}_N^{n+1} + \Delta t \mathbf{F}_{g,N}^{n+1}, \tag{41c}$$
$$\hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}. \tag{41d}$$

Time-Discretization Forcing Term

The forcing term $\mathbf{F}_{\hat{\mathbf{u}}_h}^n$ due to the time discretization has been intentionally left out in the previous section. We could naively project it too,

$$\mathbf{F}_{\hat{\mathbf{u}}_N}^N = \mathbb{V}^T \mathbf{F}_{\hat{\mathbf{u}}_h}^n, \tag{42}$$

but this would force us to reconstruct the FE vector of the ROM at each time-step, making the integration cumbersome. To go around this issue, we can exploit the algebraic representation of $\mathbf{F}_{\hat{\mathbf{u}}_h}^n$, given in Equation (31) in terms of the mass matrix. Consequently, the forcing term due to the time discretization takes the following form in the ROM:

$$\mathbf{F}_{\hat{\mathbf{u}}_N}^n = \begin{cases} \mathbf{M}_N^{n+1} \hat{\mathbf{u}}_N^n, & \text{BDF-1} \\ \frac{1}{2}\mathbf{M}_N^{n+1} \hat{\mathbf{u}}_N^n - \frac{3}{4}\mathbf{M}_N^{n+1} \hat{\mathbf{u}}_N^{n-1}, & \text{BDF-2} \end{cases} \tag{43}$$

2.1.3 Boundary and Initial Conditions

The computation of the initial condition $\hat{\mathbf{u}}_{N,0}$ is to be done in two steps. First, the initial condition $\hat{\mathbf{u}}_{h,0}$ in the FOM space needs to be computed as a FE vector by means of the techniques explained in Section 1.3.1. Then, to obtain $\hat{\mathbf{u}}_{N,0}$, the FE representation $\hat{\mathbf{u}}_{h,0}$ needs to be projected unto the

reduced space. Since we are dealing with an orthonormal basis, we can use the matrix $\mathbb{V}$ to project the FE vector departing from the FOM-ROM relation given in Equation (37),

$$\mathbb{V}^T \hat{\mathbf{u}}_{h,0} = \underbrace{\mathbb{V}^T \mathbb{V}}_{\mathbb{I}} \hat{\mathbf{u}}_{N,0} \rightarrow \hat{\mathbf{u}}_{N,0} = \mathbb{V}^T \hat{\mathbf{u}}_{h,0}. \tag{44}$$

Regarding the spatial boundary conditions, at this point of the naive reduction scheme, two facts come into play, which we first present and then put together:

1) The weak form has homogeneous boundary conditions.
2) The ad-hoc basis $\psi_i(x)$ take value zero at the boundaries.

These two facts imply that the projection of the operators, system of Equations (39), does not break the original constraint of homogeneous boundary conditions; and that the linear expansion of the solution $\hat{u}_N(x)$ in the span of V_N, Equation (36), is always true.

Once the reduced homogeneous solution $\hat{u}_N(x)$ is obtained, it can be brought back to the original space V_h via Equation (37), and then add on top of it the FE representation of the Dirichlet lifting, to obtain the final solution,

$$u(x,t;\mu) \simeq u_h(t,\mu) = \mathbb{V}\hat{u}_N(t,\mu) + g_h(t,\mu). \tag{45}$$

2.1.4 Conclusions

At this point, the naive reduction scheme for the RB-ROM has been defined and explained. However, some aspects of it remain unattended:

- The construction of the problem-based basis functions $\psi_i(x)$. There are many methods to build them, and which combination of them we chose defines which reduction method we are applying, along with their advantages and requirements.
- The *offline-online decomposition*: to uncouple the usage of FOM operators in as much as possible from the integration of the ROM. Ideally, no FOM structures should be assembled during the online phase. In this naive scheme, we are clearly not meeting such requirement, since we need to assemble all the FOM operators and then project them into V_N *for each timestep*.

Thus, some additional sections need to be brought up, in order to complete our definition of the reducing scheme. We now treat the construction of the basis functions, Section 2.2; and the approximation of the algebraic operators, Section 2.3.

2.2 Reduced Basis Construction

Hereby we layout the details of the basis construction, that is, the definition of the $\psi_i(x)$ functions. There are many techniques to build such basis, but since we are laying out a toy problem, we explain an automatic and out-of-the-box technique: the nested POD approach.

On paper, the most simple basis anyone could come up with is a collection of solution snaphsots for different parameter values and timesteps,

$$\begin{aligned}\Psi_{\hat{u}} &:= [\psi_i] \\ &= [\hat{u}_h(t^0;\mu_0), \hat{u}_h(t^1;\mu_0), \dots, \hat{u}_h(t^j;\mu_i), \dots], \\ &= [\Psi_{\hat{u}}(\mu_0), \Psi_{\hat{u}}(\mu_1), \dots, \Psi_{\hat{u}}(\mu_{N_\mu})]\end{aligned} \tag{46}$$

Yet, this basis $\Psi_{\hat{u}}$ is unpractical from a computational point of view: we could not possibly store all the basis vectors and neiter compute efficiently all the required algebraic operations with them. Additionally, it would probably lead to ill-posed linear systems, since the vectors are almost linearly dependent.

However, since all these vectors arise from the same PDE (although for different parametrizations of it), and for each timestep the solution is close to the previous one, in a way, we could expect there to be a lot of repeated information inside each $\Psi_{\hat{u}}(\mu_i)$, and consequently, inside $\Psi_{\hat{u}}$. We can exploit this fact by using a compression algorithm, such as the Proper Orthogonal Decomposition, to find a set of vectors which capture sufficiently good the span of $\Psi_{\hat{u}}$, and yet do so with less number of basis vectors.

We call this the *method of snapshots*.

2.2.1 POD Space Reduction

This section is left intentionally short for the moment. Many high quality references exist to explain what is the Proper Orthogonal Decomposition (POD), why does it work and which limitations does it have.

We can see it as an enhanced Gram-Schmidt orthonormalization procedure: not only an orthonormal basis is obtained, but additionally the output basis vectors recover hierarchically the span of the original space given by the input vectors.

Let us define the $POD : X \rightarrow Y$ function between two normed spaces, such that $Y \subseteq X$, which takes as inputs a collection of N_S vectors and a prescribed tolerance error ε_{POD},

$$[\psi_i]_{i=1}^{N_{POD}} = POD\left([\varphi_i]_{i=1}^{N_S}, \varepsilon_{\text{POD}}\right). \tag{47}$$

with $\varphi_i \in X$ and $\psi_i \in Y$. This function returns a collection of orthonormal N_{POD} vectors, whose span is a subset of the input vector span.

Internally, the POD is solving an optimality approximation problem in the L^2 norm. Therefore, with a slight notation abuse, one could conceptually say

$$\text{span}\,(\varphi_i) = \text{span}\,(\psi_i) + \varepsilon_{\text{POD}}, \tag{48}$$

that is, the representation of any vector in the original space can be reconstructed to a point with the output POD basis, and the error in the L^2 norm should be less or equal to ε_{POD}.

There is a relation between the prescribed tolerance error $\varepsilon_{\text{POD}}$and the outcoming number of basis elements N_{POD},

$$N_{POD} = N_{POD}(\varepsilon_{\text{POD}}). \tag{49}$$

This functional relationship usually shows exponential decay, or a sharp drop beyond a given number of elements. That is, if a problem is reduceable, beyond a given number of elements, adding more will not improve significantly the approximation bondness.

Alternatively to a prescribed error, one could directly ask for a prescribed number of basis functions $N^*_{POD} \leq N_S$,

$$[\psi_i]_{i=1}^{N^*_{POD}} = POD\left([\varphi_i]_{i=1}^{N_S}, N^*_{POD}\right). \tag{50}$$

2.2.2 Nested POD Basis Construction

A simple procedure to leverage the compression properties of the POD function, and the nested structure of our PDE with respect to time and the parameters; is to first build certain POD basis from the snapshots of the solution at different timesteps for a fixed parameter value,

$$\begin{aligned}
\Psi_{\mu_0} &= POD_\varepsilon\left(\left[\hat{u}_h(t^0;\mu_0),\dots,\hat{u}_h(t^T;\mu_0)\right]\right),\\
\Psi_{\mu_1} &= POD_\varepsilon\left(\left[\hat{u}_h(t^0;\mu_1),\dots,\hat{u}_h(t^T;\mu_1)\right]\right),\\
&\dots,\\
\Psi_{\mu_{N_\mu}} &= POD_\varepsilon\left(\left[\hat{u}_h(t^0;\mu_{N_\mu}),\dots,\hat{u}_h(t^T;\mu_{N_\mu})\right]\right).
\end{aligned}$$

Then, all the μ-fixed POD basis, which sum up the information contained in the time evolution direction, are compressed again using a POD,

$$\mathbb{V} := \Psi = POD_\varepsilon\left(\left[\Psi_{\mu_0},\Psi_{\mu_1},\dots,\Psi_{\mu_{N_\mu}}\right]\right).$$

In the end, this gives us a unique basis $\Psi = [\psi_i] = \mathbb{V}$, which contains information for parameter variations and time evolution. And above all, it has been built in a computationally efficienty manner, at least in terms of storage.

Different error tolerances could be prescribed at the time and parameter compression stages.

We say this to be an automatic and out-of-the-box procedure because it does not require further complications beyond the storage of the snapshots and the implementation of the POD algorithm. It only demands the definition of a collection of parameter values to solve for. This can be done with random sampling techniques, or if some physically-based knowledge is available, a custom selection of the parameters for ranges in which we know the solution will strongly vary.

Naturally, more involved procedures exist to create the final basis Ψ, but they demand the capacity to evaluate during the creation of the basis how good or bad the new basis is performing at capturing the problem dynamics, or the determination of sharp *a priori* error estimators.

We leave this for later.

2.3 System Approximation

Regarding the assembly of the reduced operators, if already during the offline stage, where the FOM problem is tackled, we had to assemble all the discrete operators for each timestep; during the online stage we have to additionally project them unto the reduced space too, as shown in Equations (39), increasing the overall cost of the integration procedure.

This will be our main motivation to include a system approximation procedure, with the goal of speeding up the construction of the operators.

We talk about *parameter and time separable* problems, or the existence of an *affine decomposition*, when the spatial operators (bilinear or linear forms), which depend on time and parameter values, present the following functional form,

$$A_h(t,\mu) = \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{h,q}, \tag{51}$$

where the coefficient functions are real-valued, $\Theta_q^a(t,\mu) \in \mathbb{R}$, and the operator basis elements $A_{h,q}$ are parameter-independent.

This expansion can be used for both matrices or vectors provided the topology of the mesh does not change in time. If this is the case, the matrices can be transformed into vectors, and later on brought back to matrix form once any necessary operation has been carried out.

If we had such a decomposition, once we had computed the basis matrix $\mathbb{V}$, we could project each element of the operator basis $A_{h,q}$ to obtain an expression for the reduced operator,

$$\begin{aligned}
A_N(t,\mu) &= \mathbb{V}^T A_h(t,\mu)\mathbb{V}\\
&= \sum_q^{Q_a} \Theta_q^a(t,\mu)\mathbb{V}^T A_{h,q}\mathbb{V}\\
A_N(t,\mu) &= \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{N,q}.
\end{aligned} \tag{52}$$

Since $A_{N,q}$ is fixed, provided that we had a way to evaluate each $\Theta_q^a(t,\mu)$, we would be able to build the reduced operator for a given parameter for each timestep without having to use any FOM operator.

2.3.1 Discrete Empirical Interpolation Method

Naturally, not many problems are likely to present a separable form as the one shown above. Even our simple linear heat equation problem, due to the time-deformation of the mesh, cannot be presented in such a form.

To tackle this issue, we use the Discrete Empirical Interpolation Method (DEIM). This method is a numerical extension of its analytical sibling, the Empirical Interpolation Method (EIM). Basically, it mimicks the idea of creating a basis for the solution space, but this time centered around the operator space. By means of a nested POD as we explained in Section 2.2.2, if we replace the solution snapshots with operator snapshots, we can build the static and problem-dependent basis $A_{h,q}$.

Since we will be creating the operator basis with an approximation technique, an error is expected in the reconstruction of the actual operator, and so we introduce the notation $A_h^m(t,\mu)$ to reference the approximation of the operator via the (M)DEIM algorithm,

$$A_h(t,\mu) \simeq A_h^m(t,\mu) = \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{h,q}. \tag{53}$$

Naturally, this idea leads to the concept of approximated reduced operators,

$$A_N(t,\mu) \simeq A_N^m(t,\mu) = \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{N,q}, \tag{54}$$

which is the approximation of the reduced operators when the basis comes from an approximation of the operator snapshots.

2.3.2 Discussion on the Approximation Target

There are reasons to approximate A_h instead of directly A_N, but I still need to wrap my head around them.

2.3.3 Evaluation of the Coefficient Functions

To evaluate the $\Theta_q^a(t,\mu)$ functions, we set and solve an interpolation problem; that is, we enforce the approximation to actually match certain elements of the operator,

$$[A_h(t,\mu)]_k = [A_h^m(t,\mu)]_k = \sum_q^{Q_a} \Theta_q^a(t,\mu)[A_{h,q}]_k \tag{55}$$

for certain indices $k \in \mathcal{I}_a$ within a collection of Q_a selected indices $\mathcal{I}_a$. The notation $[A_h(t,\mu)]_k$ stands for the value of the operator at the given mesh node k. In the FE context it can be obtained by integrating the weak form locally.

This leads to a system with the same number of unknowns as equations, where each $\Theta_q^a(t,\mu)$ is unknown. The indices $\mathcal{I}_a$ are selected during the offline stage, according to error reduction arguments of the separable form (53),

$$e_a(t,\mu) = \|A_h(t,\mu) - A_h^m(t,\mu)\|. \tag{56}$$

2.4 Hyper Reduced Basis Method

With an approximation of the reduced operators available, we can define yet another algebraic problem to integrate and obtain the reduced solution in time for a given parameter value,

$$\begin{aligned} \mathbf{M}_N^{m,n+1}\hat{\mathbf{u}}_N^{n+1} + \Delta t \mathbf{A}_N^{m,n+1}\hat{\mathbf{u}}_N^{n+1} &\qquad (57a)\\ = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t \mathbf{F}_N^{m,n+1} + \Delta t \mathbf{F}_{g,N}^{m,n+1}, &\\ \hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}. &\qquad (57b) \end{aligned}$$

If we collect terms and factor out the unknowns we get a linear system, in the reduced space and with approximated operators, to be solved for each timestep to advance the solution,

$$\begin{aligned} \mathbf{K}_N^{m,n+1}\hat{\mathbf{u}}_N^{n+1} &= \mathbf{b}_N^{m,n+1}, &(58a)\\ \mathbf{K}_N^{m,n+1} &= \mathbf{M}_N^{m,n+1} + \Delta t \mathbf{A}_N^{m,n+1}, &(58b)\\ \mathbf{b}_N^{m,n+1} &= \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t \mathbf{F}_N^{m,n+1} + \Delta t \mathbf{F}_{g,N}^{m,n+1}, &(58c)\\ \hat{\mathbf{u}}_N^0 &= \hat{\mathbf{u}}_{N,0}. &(58d) \end{aligned}$$

Each of the operators present in the problem, $\mathbf{M}_N, \mathbf{A}_N, \mathbf{F}_N$ and $\mathbf{F}_{g,N}$ will have associated an operator basis and will require the solution of the interpolation problem (55) to be solved for each timestep and parameter value. Although this could still seem like a costly procedure, if the operators are actually reduceable, the number of basis functions $Q_m, Q_a, Q_f, Q_{f,g}$ should be small, and thus way simpler problems than assembling the whole operator and the carrying out the projection.

Once the reduced homogeneous solution $\hat{u}_N^m(x)$ is obtained with approximated operators, it can be brought back to the original space V_h via Equation (37), and then add on top of it the FE representation of the Dirichlet lifting, to obtain the final solution,

$$u(x,t;\mu) \simeq u_h^m(t,\mu) = \mathbb{V}\hat{u}_N^m(t,\mu) + g_h(t,\mu). \tag{59}$$

2.5 Certification of the Reduction Procedure

Should this have some dedicated paragraphs at the beginning? I think it should.

TBD.

- Computation of error bounds between FOM and ROM without actually computing the FOM.

3 Method of Manufactured Solutions

For completeness, we define two specific problems:

- MFP-1: Reach a Steady-State Solution from Zero.
- MFP-2: Unsteady-State for All Times.

For the above we give explicit forcing terms, boundary and initial conditions. They are derived departing from a known solution

$$u_e(x,t) = \tilde{f}(t)\tilde{u}(x), \tag{60}$$

which we trim according to the implementations we want to exercise. Although this step could be seen like a loss of time, since we are not dealing with an actual problem of industrial or clinical application, it has many benefits. It gives us a known benchmark against which we can compare numerical solutions. This can not only help us find bugs in the implementation, but also certify the expected convergence rates in terms of mesh and timestep sizes. Later on, when we deal with an equation whose solution is unknown, we can be assured that the code we have written actually solves the equations written out on paper.

To simplify our equations, we chose $\tilde{u}(x)$ such that

$$\frac{\partial^2 \tilde{u}}{\partial x^2} = C. \tag{61}$$

For $\delta \in \mathbb{R}$, this can be achieved with

$$\tilde{u}(x) = 1 + \delta^2 x^2, \rightarrow C = 2\delta^2. \tag{62}$$

We set a spatial-dependent diffusion term $\alpha(x)$ with a small deviation from a fixed value,

$$\alpha(x) = \alpha_0(1 + \varepsilon x^2), \tag{63}$$

with $\alpha_0, \varepsilon \in \mathbb{R}^+$ and $\varepsilon \ll 1$. If $\varepsilon = 0$, $\alpha(x) = \alpha_0$ for the whole domain.

Our toy model is useful in exercising all the discretized operators and necessary skills to tackle the challenges the reduction of a problem alike presents.

3.1 Boundary and Initial Conditions

The boundary conditions are obtained by evaluating the function $u_e(x,t)$ at the domain endpoints, namely

$$\begin{aligned} b_0(t) &= u_e(0,t) = \tilde{f}(t)\tilde{u}(0) = \tilde{f}(t), &(64a)\\ b_L(t) &= u_e(L,t) = \tilde{f}(t)\tilde{u}(L) = \tilde{f}(t)\left(1 + \delta^2 L^2(t)\right). &(64b) \end{aligned}$$

We get the initial condition from an evaluation of $u_e(x,t)$ at time zero, namely

$$u_0(x) := u_e(x,0) = \tilde{f}(0)\tilde{u}(x). \tag{65}$$

3.2 ALE Ingredients

There are two kind of moving meshes: one that keeps fixed the domain length and simply moves the interior nodes; and one that moves the mesh according to a scaling law that changes the domain length.

3.2.1 Fixed Length

An adaptive mesh technique would want to move the mesh nodes according to some feature of the solution or its gradients. In this case, despite having a fixed domain, $L = L_0$, the interior nodes would move, leading to an ALE formulation. The mapping and mesh velocity would be

$$\mathcal{A}(\mathcal{X}, t) = \mathcal{X} - k \sin\left(\frac{2\pi\mathcal{X}}{L_0}\right) \sin(\omega t), \tag{66a}$$

$$w(\mathcal{X}, t) = -\omega k \sin\left(\frac{2\pi\mathcal{X}}{L_0}\right) \cos(\omega t) \tag{66b}$$

The drawback of this mapping is that we need to solve a non-linear equation,

$$(\mathcal{X} - x) - k \sin(2\pi\mathcal{X}) \sin\left(\frac{2\pi t}{T}\right) = 0, \tag{67}$$

to obtain $\mathcal{X} = \mathcal{X}(x, t)$ to compute the mesh velocity in physical coordinates, $w(x, t)$, which is the one that needs to be provided to the weak form in the actual calculation of the solution at the algebraic level.

3.2.2 Changing Length

The alternative scenario is one were the domain length is not fixed, $L = L(t)$. The movement of the rightmost side $L(t)$ is defined with a harmonic function,

$$L(t) = L_0 \left[1 - \sin(\omega t)\right], \tag{68a}$$

$$L'(t) = -L_0 \omega \cos(\omega t). \tag{68b}$$

where the frequency ω will be our geometrical parameter. The mapping and mesh velocity are

$$\mathcal{A}(\mathcal{X}, t) = \left(\frac{\mathcal{X}}{L_0}\right) L(t), \tag{69a}$$

$$w(\mathcal{X}, t) = \left(\frac{\mathcal{X}}{L_0}\right) L'(t), \tag{69b}$$

$$w(x, t) = \left(\frac{1}{L_0}\right) L'(t) L_0 \left(\frac{x}{L(t)}\right) = L'(t) \left(\frac{x}{L(t)}\right) \tag{69c}$$

In this scaled moving domain it is simpler to compute the mesh velocity in physical coordinates, since we do not need to solve a non-linear equation to obtain $\mathcal{X} = \mathcal{X}(x, t)$.

We could decide to stop the simulation when the domain has stretched by a factor n. If so, we have to compute the final time t_f,

$$L(t_f) = nL_0, \quad 0 < n < 1, \tag{70}$$

so that the end time t_f of our simulation should be

$$t_f = \frac{1}{\omega} \sin^{-1}(1 - n). \tag{71}$$

3.3 Forcing Term

The strong formulation forcing term $f(x, t)$ from which we depart is computed by introducing $u_e(x, t)$ into the PDE, which leads to

$$f(x, t) = \tilde{f}'(t)\tilde{u}(x) - \alpha(x)\tilde{f}(t)\tilde{u}''(x), \tag{72a}$$

$$f(x, t) = \tilde{f}'(t)\tilde{u}(x) - 2\alpha(x)\delta^2\tilde{f}(t). \tag{72b}$$

Since we did not obtain $u_e(x, t)$ from any reasoning related to the body of the PDE, the forcing term is not identically zero. This was precisely our goal, to obtain a forcing term which would lead the solution of the PDE to be $u_e(x, t)$.

3.3.1 Forcing Due to Lifting

We recall that the FE problem we solve is the one corresponding to the homogeneous solution of the problem.

Therefore, in order to implement these problems we need to define the exact expression of f_g. From Equation (12) we get that the forcing term linked to the lifting of the Dirichlet boundary conditions has the following form,

$$f_{g,1} = -\frac{\partial g}{\partial t} + w\frac{\partial g}{\partial x}, \tag{73}$$

$$f_{g,2} = -\alpha\frac{\partial g}{\partial x}. \tag{74}$$

Since the lifting g is the linear combination of both boundary conditions, Equation (11),

$$g(x, t) = b_L(t) \left(\frac{x}{L}\right) + b_0(t) \left(\frac{L - x}{L}\right), \tag{75}$$

we get that the forcing term f_g becomes the combination of

$$\frac{\partial g}{\partial x} = \frac{b_L(t) - b_0(t)}{L}, \tag{76a}$$

$$\frac{\partial g}{\partial t} = b_L'(t) \left(\frac{x}{L}\right) + b_0'(t) \left(\frac{L - x}{L}\right). \tag{76b}$$

The time derivatives of the boundary conditions are derived from Equations (64),

$$b_0'(t) = \tilde{f}'(t), \tag{77a}$$

$$b_L'(t) = \tilde{f}'(t)\left(1 + \delta^2 L^2(t)\right) + 2\tilde{f}(t)\delta^2 L(t)L'(t). \tag{77b}$$

The expression for $\tilde{f}'(t)$ will be different for each manufactured problem. Note that the second summand of $b_L'(t)$ vanishes if $L(t)$ is constant, i.e. $L'(t) = 0$.

3.4 MFP-1: Reach a Steady-State Solution from Zero

Our first problem will be built such that a zero initial condition is given, and that a steady state solution is reached.

For $\beta \in \mathbb{R}^+$, let the function $\tilde{f}(t)$ be

$$\tilde{f}(t) = 1 - e^{-\beta t}, \tag{78a}$$

$$\tilde{f}'(t) = \beta e^{-\beta t}. \tag{78b}$$

The shape of the time function $\tilde{f}(t)$,

$$\begin{cases} \tilde{f}(0) = 0, \\ \lim_{t\to\infty} \tilde{f}(t) = 1, \end{cases}$$

ensures that we reach a steady state solution for large time values,

$$\lim_{t\to\infty} u_e(x, t) = \tilde{u}(x), \tag{79}$$

and that we have a null initial condition.

If we define the number f_∞ to be smaller than unity, but very close to it, $f_\infty \sim 1$, we can estimate the final simulation time to arrive to the steady-state solution by solving for t in Equation (78a),

$$t_f \simeq -\frac{1}{\beta} \ln(1 - f_\infty). \tag{80}$$

The larger β, the quicker the steady-state arrives, but also the smaller the discrete timestep Δt will have to be to capture correctly the time evolution of the system.

The explicit expression for the forcing term is obtained from Equations (72) and (63),

$$\begin{aligned} f(x,t) = \beta e^{-\beta t}(1+\delta^2 x^2) \\ - 2\delta^2\alpha_0(1+\varepsilon x^2)\left(1-e^{-\beta t}\right). \end{aligned} \tag{81}$$

The boundary condition values, to be used to build the lifting term in Equation (11), are obtained from Equations (64),

$$b_0(t) = 1 - e^{-\beta t}, \tag{82a}$$

$$b_L(t) = \left(1-e^{-\beta t}\right)\left(1+\delta^2 L^2(t)\right). \tag{82b}$$

Collecting all the parameters used to define problem MFP-1, we find:

- α_0: constant diffusion constant.
- ε: present in the diffusion coefficient, it determines how sensitive $\alpha(x)$ is to spatial variations.
- ω: Present in the definition of the moving domain, it defines how quickly the domain boundary moves.
- δ: present in the moving boundary condition and in the forcing term. It regulates how strongly the solution changes in the spatial direction.
- β: Present in both boundary conditions and in the forcing term, it defines how quickly the steady-state is reached.

So, $\mu = [\alpha_0, \varepsilon, \omega, \delta, \beta]$ and $|\mu| = 5$.

3.5 MFP-2: Unsteady for All Times

Our second problem will be built such that we depart from an initial profile and that a steady state is never reached.

For $\omega_f \in \mathbb{R}^+$, let the function $\tilde{f}(t)$ be

$$\tilde{f}(t) = \cos(\omega_f t), \tag{83a}$$

$$\tilde{f}'(t) = -\omega_f \sin(\omega_f t). \tag{83b}$$

The shape of the time function $\tilde{f}(t)$,

$$\begin{cases} \tilde{f}(0) = 1, \\ \left\|\tilde{f}(t)\right\| < \infty, \\ \nexists \lim_{t\to\infty} \tilde{f}(t), \end{cases}$$

ensures that we start from an initial profile at the beginning of the simulation,

$$u_e(x,0) = \tilde{u}(x), \tag{84}$$

and that we never reach a steady-state solution due to the oscillatory nature of $\tilde{f}$.

Again, the explicit expression for the forcing term is obtained from Equations (72) and (63),

$$\begin{aligned} f(x,t) = -\omega_f \sin(\omega_f t)(1+\delta^2 x^2) \\ - \delta^2\alpha_0(1+\varepsilon x^2)\cos(\omega_f t). \end{aligned} \tag{85}$$

The boundary condition values, to be used to build the lifting term in Equation (11), are obtained from Equations (64),

$$b_0(t) = \cos(\omega_f t), \tag{86a}$$

$$b_L(t) = \cos(\omega_f t)\left(1+\delta^2 L^2(t)\right). \tag{86b}$$

Collecting all the parameters used to define problem MFP-2, we find:

- α_0: constant diffusion constant.
- ε: present in the diffusion coefficient, it determines how sensitive $\alpha(x)$ is to spatial variations.
- ω: Present in the definition of the moving domain, it defines how quickly the domain boundary moves.
- δ: present in the moving boundary condition and in the forcing term. It regulates how strongly the solution changes in the spatial direction.
- ω_f: Present in both boundary conditions and in the forcing term, it defines how quickly the solution changes in time.

So, $\mu = [\alpha_0, \varepsilon, \omega, \delta, \omega_f]$ and $|\mu| = 5$.