Skipping The Jacobian

Hyperreduced Order Models For Moving Domains

Enrique Millán Valbuena

463 426 8

Abstract

We build a Reduced Order Model (ROM) for a one-dimensional gas dynamics problem: a moving piston. The main body of the PDE, the geometrical definition of the moving boundary, and the boundary conditions are parametrized. A concise description of the reducing procedure is provided, together with a priori convergence rates for basis size estimation and a posteriori error estimators to certify the use of the Reduced Order Model. Numerical examples to showcase computational costs and implementation details are designed, implemented and validated with the Manufactured Solutions Method.

Index Terms

Reduced Order Model, Moving Domain, FEM, DEIM, POD, Galerkin projection

CONTENTS

TODO LIST

To go to Rome is little profit;
to go to Rome is little profit, endless pain.
The master that you seek in Rome,
you find at home, or seek in vain.

FOREWORD

This kind of works typically set the ending of a cycle in life. In my case, it marks a restart.

I was very close to dropping out from my masters, contempt with having my bachelor and partially frustrated for multiple reasons not worth developing here. Taking into account the fact that my professional life seemed to have drifted away from Aerospace Engineering, I struggled to see the point at completing it, let alone finding the time such task required.

However, two ideas made me get back to work. First, it is easier to explain an elongated but completed academic course, rather than an unfinished one. Second, and most important, I do actually like the subject. In fact, during the last year I have realized how much I enjoy *Applied Mathematics*, a field by which Aerospace Engineering is widely nurtered.

In this regard I would like to thank professors A. Quarteroni and A. Manzoni for giving me the opportunity to work with them in their research group in Milan. Apart from learning mathematics, I met great colleagues and picked up one of Europe's most beautiful languages. However, working with the FEM code available there, LifeV, was tough, and probably its complexity had to do with the growing frustration I was already experiencing in my academic life: too much time spent debugging, rather than understanding the problem with pen and paper. Nevertheless, with them I discovered a way of using mathematics that I had not learnt before, one that suits my mind and approach to scientific modelling.

I am grateful for the warmth I received from the PhD students and postdocs at the office where we worked together everyday: Federica, Dani (south), Dani (north), Abele, Ludovica, Stefano, and a countable infinite more. I keep good memories playing volley and climbing with you all. Last, but not least, I would like to thank Niccolò Dal Santo for his patience, time and knowledge; which he generously dedicated to me despite not being officially assigned as my supervisor. You are definitely among the most clever and kindest people I have met in my life.

At TU Delft, I would like to thank professor Steven for his joyful encouragement during round two of this thesis. When I first wrote him back after a year and a half since he last heard from me, I thought he would (righteously) no longer want to have anything to do with this work. I was gladly surprised to receive all of the contrary, I warm welcome back and a pragmatic view to reach the end at the fastest pace, whilst doing a good job.

At last, although they are completely outside of the academic scope, I would like to thank my colleagues at work and my close friends. My two supervisors, Ana and Sarah, from whom I have adquired the pragmatism industrial problems require to be completed in time and form. To Maximiliano, again a smart and kind person across my path in life, eager to teach me professional coding skills and how to structure creativity. Emmanuel, my housemate, who despite being a lawyer would kindly ask me every now and then how my convergence rates were going on. Miquel, a friend turned brother, for your unwearing support and

advice. And to all of the remaining, with whom I spend great quality time. You influence my life more than you are probably aware.

As the reader will see, this is quite a simple work. Coding it has been laborious, but the content remains simple. Yet, its simplicity has allowed me to understand the fundamentals of two versatile and powerful mathematical tools: Finite Elements and Reduced Order Models. This has motivated me to keep on working at it once this is over, until I end up solving the real-life problem I set to solve in the first place: the fluid mechanics problem of the human heart.

Madrid (España), 2021.

EXECUTIVE SUMMARY

1 Introduction

When a painter sets out to paint, she will probably use most of the available basic colours. However, if she knew beforehand she was only going to paint landscapes, she would fare well with a farsighted palette: greens, browns, blues, whites, etc. Such is the nature of Reduced Order Models, to find a subset among the combinations of colours to represent the solution to the problem of interest.

In this context, the basic colours are the classical mathematical Lagrangian finite element basis functions, generic and with local support to represent most functions of interest. Instead, the landscape palette will be ad-hoc functions, with global support, good at capturing details only specific to landscapes.

Additionally, she will not need all sorts of brushes, simply the ones with the right thickness and width for mountains and trees. The brushes represent the algebraic operators that arise from the finite element discretization. In a similar fashion as with the colours, we can find a subset of combinations of brushes that suit our problem. That is, we can find a basis for each algebraic operator to build them efficiently.

Finally, since she is a vanguard painter, the domain of our problem, her canvas, will be allowed to change in time, as she paints. The landscape colours and brushes we select will need to take this into account.

Mention that our goal is to avoid the computation of the jacobian matrix.

Mention that we will be using ALE formulation.

2 One-Dimensional Gas Dynamics

The Full Order Model for the parametrized one-dimensional piston problem is derived.

We depart from the compressible isentropic Navier Stokes equations to end up with a non-linear Burgers-like equation. Therefore, this problem contains all the necessary ingredients to show how the ROM behaves in the presence of a non-linear term within a moving domain.

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.

We define the vector $\vec{\mu} \in \mathcal{P}$ to collect all the parameters present in the formulation. Parameters will be present in the PDE's body, in the boundary conditions, or in the geometrical definition of the domain.

We deal with our problem within a deforming domain in time, whose movement is known and not part of the solution:

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

From now on, we drop the dependency on time and the parameters unless it is strictly necessary.

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.

2.1 Physical Derivation

The piston movement $L(t)$ is a real-valued smooth sinusoidal function,

$$L(t) = L_0 \left[1 - \delta \left(1 - \cos\left(\omega t\right)\right)\right], \tag{1}$$

where ω is the frequency at which it oscillates and $\delta \ll L_0$ is a scale variable to adjust how much it is

displaced from its original position. The length L_0 is defined to keep physical dimensions sound, but it will remain fixed to $L_0 = 1$ for the remaining of the problem.

To derive the equations of motion of the fluid inside the piston, we depart from the conservation of mass, momentum and the isentropic relation between pressure and density,

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = 0, \tag{2a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} = 0, \tag{2b}$$

$$p = k\rho^{\gamma}. \tag{2c}$$

Body forces and viscosity have been neglected for the sake of simplicity[1]. A difficulty we find in this system for the piston application is the determination of the boundary condition for density at the piston location. Ideally, we only want to solve an equation for the velocity, where boundary conditions are easy to set.

To do so, we take the following steps:

1) Remove the pressure gradient by means of the isentropic relation.
2) Relate explicitly velocity u to density ρ.
3) Collect all the above into one equation in terms of velocity u.

Removal of the pressure gradient

To remove the pressure gradient, we start by taking derivatives in the isentropic relation (2c):

$$\frac{\partial p}{\partial x} = k\gamma\rho^{\gamma-1}\frac{\partial \rho}{\partial x} \tag{3}$$

Then, we recognize that the coefficients $k\gamma\rho^{\gamma-1}$ multiplying the spatial derivative of density are in fact the squared speed of sound. From thermodynamics, the speed of sound a squared is the derivative of pressure with respect to density at constant entropy,

$$a^2 = \left.\frac{\partial p}{\partial \rho}\right|_S = k\gamma\rho^{\gamma-1}. \tag{4}$$

1. Viscosity terms will be introduced later on for numerical stability.

Hence, the expression for the pressure gradients becomes

$$\frac{\partial p}{\partial x} = a^2\frac{\partial \rho}{\partial x}, \tag{5}$$

which can be plugged directly into the momentum equation. The system becomes

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = 0, \tag{6a}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{a^2}{\rho}\frac{\partial \rho}{\partial x} = 0, \tag{6b}$$

$$a = \sqrt{k\gamma}\rho^{\frac{\gamma-1}{2}}. \tag{6c}$$

Compatibility Condition between u and ρ

Our next step is to find a compatibility condition between the mass and momentum equations. We define $u := \mathtt{V}(\rho)$, which, by application of the chain rule, leads to the following equalities between the derivatives of u and ρ :

$$\frac{\partial u}{\partial t} = \mathtt{V}'\frac{\partial \rho}{\partial t}, \tag{7a}$$

$$\frac{\partial u}{\partial x} = \mathtt{V}'\frac{\partial \rho}{\partial x}, \tag{7b}$$

where $\mathtt{V}'$ represents differentiation with respect to density. Introducing these relations into the mass and the momentum equations to remove u, we obtain

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x}\left(\rho\mathtt{V}' + \mathtt{V}\right) = 0, \tag{8a}$$

$$\mathtt{V}'\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x}\left(\mathtt{V}\mathtt{V}' + \frac{a^2}{\rho}\right) = 0. \tag{8b}$$

Since we have two equations for two unknowns, the system can only have a solution if the determinant is zero.

Why does the determinant need to be zero for the compatibility condition?

$$\begin{vmatrix} 1 & \mathtt{V} + \rho\mathtt{V}' \\ \mathtt{V}' & \mathtt{V}\mathtt{V}' + \frac{a^2}{\rho} \end{vmatrix} = 0 \tag{9}$$

This determinant implies that

$$\mathtt{V}'\left(\mathtt{V} + \frac{a^2}{\rho\mathtt{V}'} - \mathtt{V} - \rho\mathtt{V}'\right) = 0. \tag{10}$$

The above equation has two possible solutions, either

$$\mathtt{V}' = 0 \rightarrow \mathtt{V} = C, \tag{11}$$

which is the constant solution, valid but uninteresting, or

$$\frac{a^2}{\rho \mathtt{V}'} - \rho \mathtt{V}' = 0 \rightarrow \mathtt{V}' = \pm \frac{a}{\rho}. \tag{12}$$

The $\pm$ signs come up because there are two travelling waves, one to each side of the domain. Since our piston is located at the right boundary of the domain, we choose the $-$ sign so that waves propagate left, and continue our derivation. We can integrate $\mathtt{V}'$ with respect to ρ to obtain a relation between the flow velocity u and the speed of sound a:

$$\mathtt{V} = -\int_{\rho_0}^{\rho} \frac{a}{\rho} d\rho \tag{13}$$

where ρ_0 is some reference density. At this point it is convenient to use this reference density ρ_0 to remove the constant k from the expression of the speed of sound:

$$a = a(\rho) \rightarrow a_0 = a(\rho_0), \tag{14a}$$

$$a = \sqrt{k\gamma}\rho^{\frac{\gamma-1}{2}} \rightarrow a = a_0 \left(\frac{\rho}{\rho_0}\right)^{\frac{\gamma-1}{2}}. \tag{14b}$$

Plugging this expression into the integral, we get

$$u = \mathtt{V} = \frac{2a_0}{\gamma - 1}\left(1 - \frac{a}{a_0}\right), \tag{15a}$$

$$a = a_0 - \frac{\gamma - 1}{2} u. \tag{15b}$$

2.1.1 Burgers-like Equation

With all the above, we are now ready to obtain *one* equation which contains the three departing ones. In the momentum equation we first substitute the space derivative of density,

$$\frac{\partial u}{\partial x} = \mathtt{V}' \frac{\partial \rho}{\partial x} \rightarrow \frac{\partial \rho}{\partial x} = -\frac{\rho}{a}\frac{\partial u}{\partial x}, \tag{16a}$$

$$\frac{\partial u}{\partial t} + (u - a)\frac{\partial u}{\partial x} = 0. \tag{16b}$$

Then, we express the speed of sound in terms of velocity (15b), which leads to a PDE containing a Burgers-like nonlinear term and forced convection driven by the static speed of sound,

$$\frac{\partial u}{\partial t} + \frac{\gamma + 1}{2} u \frac{\partial u}{\partial x} - a_0 \frac{\partial u}{\partial x} = 0. \tag{17}$$

This PDE, together with its boundary conditions and the boundary's prescribed movement, represents the mathematical model of interest for our work.

2.1.2 Complete Determination of Flow Variables

Although we now have *one* equation which accounts for mass conservation and the isentropic relation between pressure and density, we still have three variables. Computing density, pressure and the speed of sound in space and time still remains useful: verification of mass conservation, computation of the force exerted by the fluid at the piston, or any other secondary derivations.

Since the speed of sound a is defined as a function of u in Equation (15b), once the latter is given, we can obtain density ρ and pressure p as a function of flow velocity u:

$$\rho = \rho_0 \left(\frac{a}{a_0}\right)^{\frac{2}{\gamma-1}}, \tag{18a}$$

$$p = p_0 \left(\frac{\rho}{\rho_0}\right)^{\gamma}, \tag{18b}$$

$$\left(\frac{a}{a_0}\right) = 1 - \frac{\gamma - 1}{2}\left(\frac{u}{a_0}\right), \tag{18c}$$

$$\left(\frac{\rho}{\rho_0}\right) = \left(1 - \frac{\gamma - 1}{2}\left(\frac{u}{a_0}\right)\right)^{\frac{2}{\gamma-1}}, \tag{18d}$$

$$\left(\frac{p}{p_0}\right) = \left(1 - \frac{\gamma - 1}{2}\left(\frac{u}{a_0}\right)\right)^{\frac{2\gamma}{(\gamma-1)}}. \tag{18e}$$

2.1.3 Mass Conservation

Before we present the complete problem and its numerical solution, we comment on the integral equation for mass conservation. This relation will not be used for the solution of the system, but it is a healthy check to perform after the calculation, to make sure we have correctly integrated problem. Mass conservation is the least of requirements we demand from fluid motion.

For a control volume whose boundary moves with the piston, the integral expression for mass conservation is

$$\frac{d}{dt}\int_{\Omega_t} \rho d\Omega + \int_{\partial\Omega_t} \rho\left(\vec{u} - \vec{u}_c\right)\cdot \vec{n} dS = 0 \tag{19}$$

At the piston location the flow moves with the piston, $\vec{u} = \vec{u}_c$, so the boundary integral vanishes. There is no flux through the walls either, so the only contribution left is the outflow. At this location, the

scalar product of the velocity and normal vectors is negative[2], which implies

$$\frac{d}{dt}\int_0^{L(t)} \rho(x,t)dx - \rho(0,t)u(0,t) = 0. \tag{20}$$

If we introduce the expression of density in terms of velocity from Equation (18d), we get

$$\begin{aligned}&\frac{d}{dt}\int_0^{L(t)} \left(1-\frac{\gamma-1}{2}\left(\frac{u(x,t)}{a_0}\right)\right)^{\frac{2}{\gamma-1}} dx \\ &-u(0,t)\left(1-\frac{\gamma-1}{2}\left(\frac{u(0,t)}{a_0}\right)\right)^{\frac{2}{\gamma-1}} = 0.\end{aligned} \tag{21}$$

To ease our task at verifying this equation, we rather calculate the integral term for each time step

$$I(t) = \int_0^{L(t)} \left(1-\frac{\gamma-1}{2}\left(\frac{u(x,t)}{a_0}\right)\right)^{\frac{2}{\gamma-1}} dx, \tag{22}$$

and only then compute time derivatives, which can be done with a second order finite difference scheme. Then we can compute numerically the mass defect $MD(t)$ between the time derivative of volume variation and mass outflow at the open boundary,

$$MD(t) = I'(t) - u(0,t)\left(1-\frac{\gamma-1}{2}\left(\frac{u(0,t)}{a_0}\right)\right)^{\frac{2}{\gamma-1}}. \tag{23}$$

We shall verify to which degree of accuracy this relation is honoured by our numerical scheme.

Formulate mass conservation as a FOM functional to be computed with the ROM basis.

2.1.4 Interesting Physical Quantities

In the craft of building reduced order models, it is certainly interesting to measure and study the error between the full and reduced solutions. However, in practical applications, one is often more concerned about aggregated magnitudes derived from raw flow variables, such as maximum flow, pressure gradients across channels, etc. A comparison between the full order and reduced order aggregated magnitudes will shed light on the quality and reproducibility of the reduction scheme.

2. Although we expect the flow to leave the domain at this point, which means a negative magnitude, this products needs to be done taking the variables positive in the direction of the axes.

We consider two magnitudes of interest:

1) The outflow at the boundary.
2) The pressure gradient across the channel.

The outflow at the boundary $F_L(t)$ is given by

$$F_L(t) = u(0,t)\rho(0,t), \tag{24a}$$

$$F_L(t) = u(0,t)\rho_0\left(1-\frac{\gamma-1}{2}\left(\frac{u(x,t)}{a_0}\right)\right)^{\frac{2}{\gamma-1}} \tag{24b}$$

The pressure gradient across the channel is the difference between pressure at the piston and at the outflow boundary,

$$\Delta_L p(t) = -\frac{p(0,t)-p(L(t),t)}{L(t)}, \tag{25a}$$

$$\begin{aligned}\Delta_L p(t) = -\frac{p_0}{L(t)}\Bigg[&\left(1-\frac{\gamma-1}{2}\left(\frac{u(0,t)}{a_0}\right)\right)^{\frac{2\gamma}{(\gamma-1)}} \\ &-\left(1-\frac{\gamma-1}{2}\left(\frac{u(L,t)}{a_0}\right)\right)^{\frac{2\gamma}{(\gamma-1)}}\Bigg].\end{aligned} \tag{25b}$$

We have defined it so that when the piston is moving forward, pushing the gas out of the chamber, the piston pressure gradient is defined positive.

2.2 Continuous Formulation

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

We introduce the ALE formulation, to account for the movement of the mesh. Then, we will introduce a scaling to work with non-dimensional variables. This will complete the strong formulation.

Then, once we have obtained a weak formulation via the Galerkin projection principle, we will introduce a Dirichlet lifting of the boundary conditions. To solve the FOM this is not mandatory, since the boundary conditions can be directly tweaked into the right-hand side of the algebraic vectors. However, it is a paramount step for the construction of our reduced space, to ensure the basis serves all boundary parametrizations.

2.2.1 Strong Formulation

At this point of the derivation, we allow ourselves to introduce a diffusion term which was not present in the physical derivation of the model. The viscosity value will be small enough so that it disappears with the mesh size,

Review diffusion coefficient value for mesh consistency.

$$\varepsilon \sim (\Delta x)^2 . \tag{26}$$

Thus, the differential model in the physical space is given by the following PDE, boundary and initial conditions,

$$\left.\frac{\partial u}{\partial t}\right|_x + \frac{\partial u}{\partial x}\left(\frac{\gamma+1}{2}u - a_0\right) - \varepsilon\frac{\partial^2 u}{\partial x^2} = 0, \tag{27a}$$

$$u(L,t) = L'(t). \tag{27b}$$

At the left boundary, we will impose a natural outflow condition. At the right boundary, the moving piston, a Dirichlet condition sets the flow velocity equal to the one of the moving wall,

Natural outflow condition means evaluating the weak form at the boundary. This is compatible with the small diffusion coefficient. This is equivalent to imposing an homogeneous Neumann boundary condition.

$$L'(t) = -\delta L_0 \omega \sin(\omega t). \tag{28}$$

The notation $\left.\frac{\partial u}{\partial t}\right|_x$ indicates that the derivative takes place in the physical moving domain. This is relevant for the ALE formulation, which we explain in the following section.

2.2.2 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:

$$x = \mathcal{A}(\mathcal{X}, t), \tag{29a}$$

$$\mathcal{X} = \mathcal{A}^{-1}(x, t), \tag{29b}$$

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:

$$w(x,t) = \frac{\partial x}{\partial t} = \frac{\partial \mathcal{A}}{\partial t}(x,t) = \frac{\partial \mathcal{A}}{\partial t}(\mathcal{A}^{-1}(\mathcal{X},t),t). \tag{30}$$

Time-Derivative in the Reference Domain

Since the equations are going to be solved in the physical domain, we need to adjust the time derivative in the physical domain $\left(\left.\frac{\partial u}{\partial t}\right|_x\right)$, so that it takes into account the movement of the mesh nodes. By application of the chain rule we get

$$\left.\frac{\partial u}{\partial t}\right|_{\mathcal{X}} = \left.\frac{\partial u}{\partial t}\right|_x + w\frac{\partial u}{\partial x}, \tag{31}$$

from where we get the necessary modification to be done to the strong form (27a) of the PDE,

$$\left.\frac{\partial u}{\partial t}\right|_{\mathcal{X}} + \left(\frac{\gamma+1}{2}\right)u\frac{\partial u}{\partial x} - (a_0 + w)\frac{\partial u}{\partial x} - \varepsilon\frac{\partial^2 u}{\partial x^2} = 0. \tag{32}$$

An additional convective term shows up, to take into account the movement of the nodes. We will show in the results section how, if we remove this term, mass is no longer conserved.

Quote paper with FSI piston.

Define ALE map and mesh velocity for the piston problem.

2.2.3 Nondimensional Equations

Finally, as it is a good practice, we carry out a nondimensionalization of the velocity and the spatial coordinate,

$$\tilde{x} = \frac{x}{L_0}, \qquad \tilde{u} = \frac{u}{a_0}, \tag{33a}$$

with respect to the static speed of sound a_0 and the piston's initial length L_0. This leads to the PDE and boundary condition

$$\frac{\partial \tilde{u}}{\partial t} + \left(\frac{\gamma+1}{2}\right)\left(\frac{a_0}{L_0}\right)\tilde{u}\frac{\partial \tilde{u}}{\partial \tilde{x}} - \left(\frac{\varepsilon}{L_0^2}\right)\frac{\partial^2 \tilde{u}}{\partial \tilde{x}^2} - \left(\frac{a_0}{L_0}\right)\left(1+\frac{w}{a_0}\right)\frac{\partial \tilde{u}}{\partial \tilde{x}} = 0, \tag{34a}$$

$$\tilde{u}(L(t),t) = \frac{L'(t)}{a_0} = b_L(t). \tag{34b}$$

For the sake of clear notation, we condense the coefficients

$$b_0 = \frac{a_0}{L_0}\left(\frac{\gamma+1}{2}\right), \tag{35a}$$

$$b_L(t) = -\frac{\delta L_0 \omega}{a_0}\sin(\omega t), \tag{35b}$$

$$c(x,t) = \left(\frac{a_0}{L_0}\right)\left(1-\frac{w}{a_0}\right), \tag{35c}$$

and drop the $\tilde{u}$ notation to prevent an overloaded notation.

The coefficient $\left(\frac{\delta L_0 \omega}{a_0}\right)$ multiplying the boundary condition is the relation between the piston motion and the static speed of sound[3]. As we will see in the results section, this coefficient will be the one driving the response of the fluid to the motion of the piston. The larger it is, the stronger the nonlinear response will become. We shall exploit this fact to build our reduced space, by sampling smartly through the parameter space.

With all of this done, we obtain a familiar PDE structure with diffusion, linear and non-linear convection terms:

$$\left.\frac{\partial u}{\partial t}\right|_{\mathcal{X}} + b_0 u\frac{\partial u}{\partial x} - c(x,t)\frac{\partial u}{\partial x} - \varepsilon\frac{\partial^2 u}{\partial x^2} = 0. \tag{36}$$

Before moving on to the derivation of the weak formulation, we apply the scaling to the mass conservation and derived variables expressions.

3. a proxy for the average speed at which information propagates.

2.2.4 Mass Conservation

If this rescaling is applied, Equation (23) for mass conservation needs to be updated, to take into account the fact that the velocity variable is now scaled with the speed of sound.

$$I(t) = \int_0^{L(t)} \left(1 - \frac{\gamma-1}{2}u(x,t)\right)^{\frac{2}{\gamma-1}} dx, \tag{37a}$$

$$MC(t) = I'(t) - a_0 u(0,t)\left(1-\frac{\gamma-1}{2}u(0,t)\right)^{\frac{2}{\gamma-1}}. \tag{37b}$$

2.2.5 Derived Variables

The derived magnitudes of interest, Equations (25a) and (24a), have to be rescaled too: The outflow at the boundary $F_L(t)$ is given by

$$F_L(t) = a_0\rho_0 u(0,t)\rho(0,t). \tag{38}$$

The pressure gradient across the channel is the difference between pressure at the piston and at the outflow boundary,

$$\Delta_L p(t) = -\frac{p(0,t) - p(L(t),t)}{L(t)}. \tag{39}$$

We have defined it so that when the piston is moving forward, pushing the gas out of the chamber, the piston pressure gradient is defined positive.

2.2.6 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,

$$\langle u, v\rangle = \int uv\, d\Omega. \tag{40}$$

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

$$\|f-g\| = \sqrt{\int (f-g)^2\, d\Omega}. \tag{41}$$

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,

Is this the right way to formulate a weak form?

$$\left\langle \frac{\partial u}{\partial t}, v \right\rangle + \langle b_0 u \boldsymbol{\nabla} u, v \rangle - \langle c \boldsymbol{\nabla} u, v \rangle + \langle \varepsilon \boldsymbol{\nabla} u, \boldsymbol{\nabla} v \rangle = 0, \tag{42a}$$

$$u(x,0) = 0, \tag{42b}$$

$$u(L,t) = b_L(t), \tag{42c}$$

$$b_L(t) = -\frac{\delta L_0 \omega}{a_0} \sin \omega t. \tag{42d}$$

The natural boundary condition, which for this problems translates into an homogeneous Neumann condition, has already been taken into account in the integration by parts procedure of the diffusion term.

2.2.7 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:

$$u(x,t) = \hat{u}(x,t) + g(x,t). \tag{43}$$

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:

$$g(x,t) = b_L(t) \left(\frac{x}{L} \right) \tag{44}$$

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 analytically the extension of the boundary conditions.

Introducing the lifting breakdown (43) into the weak formulation (42a), we find additional forcing terms and PDE terms, due to the cross-product between the function and its derivative,

$$u \frac{\partial u}{\partial x} = (\hat{u} + g) \left(\frac{\partial \hat{u}}{\partial x} + \frac{\partial g}{\partial x} \right). \tag{45}$$

Taking this into account, we find the following *lifted* weak formulation,

$$\begin{aligned} \left\langle \frac{\partial \hat{u}}{\partial t}, v \right\rangle - \langle c \boldsymbol{\nabla} \hat{u}, v \rangle + \langle \varepsilon \boldsymbol{\nabla} \hat{u}, \boldsymbol{\nabla} v \rangle \\ + \langle b_0 g \boldsymbol{\nabla} \hat{u}, v \rangle + \langle b_0 \hat{u} \boldsymbol{\nabla} g, v \rangle \\ + \langle b_0 \hat{u} \boldsymbol{\nabla} \hat{u}, v \rangle \\ = - \left\langle \frac{\partial g}{\partial t}, v \right\rangle + \langle c \boldsymbol{\nabla} g, v \rangle - \langle \varepsilon \boldsymbol{\nabla} g, \boldsymbol{\nabla} v \rangle \\ - \langle b_0 g \boldsymbol{\nabla} g, v \rangle, \end{aligned} \tag{46a}$$

where we have reorganized the terms to show in order: linear dependency with the solution, terms due to cross-product effects, and finally the actual non-linearity.

Thanks to the lifting, we find the homogenization of the boundary conditions for all time t,

$$\hat{u}(L,t) = 0, \tag{47a}$$

and recall that the initial condition should be modified accordingly to the homogeneous problem definition,

$$u(x,0) = \hat{u}(x,0) + g(x,0), \tag{48a}$$

$$\hat{u}_0(x) := \hat{u}(x,0) = u(x,0) - g(x,0). \tag{48b}$$

At this point, we have defined the continuous problem for the one-dimensional piston in a moving domain.

Did I modify correctly my initial condition in the code?

2.3 Semi-Discrete Formulation: Time-Discretization

The continuous solution changes in two dimensions: space and time. Eventually, we need to discretize them both, but we can do so incrementally.

First, we discretize in time. To do so, we need to build an approximation for the time derivate, $\left(\frac{\partial \hat{u}}{\partial t}\right)$. There is a complete body of knowledge devoted to this step of the numerical scheme.

Cite works on time discretization.

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 semi-implicit scheme, which is potentially 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 the solution at previous timesteps, $\{u(x,t^n), u(x,t^{n-1}), \dots\}$. 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 timesteps used in the interpolation.

In the coming sections, we define two implicit schemes of order one and two, BDF-1 (also known as Backwards Euler), and BDF-2, respectively. 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. 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 time discretization. In terms of notation, from now on we use

$$\hat{u}^n := \hat{u}(x,t^n) \tag{49}$$

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

2.3.1 Semi-Implicit Scheme

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. If the nonlinear convective term is treated semi-implicitly, one recovers a linear system, and yet reaches a satisfactory numerical solution.

Additionally, as we shall see further down in the reduced model, we will be able to assemble the reduced non-linear operator with the MDEIM procedure. If for the evaluation of the nonlinear we used u^{n+1}, we would have a non-linear system which could be solved with a Newton-Rapshon iterative method. Although this method is certainly the most robust, it is not efficient.

Thus, we opt for the approximation of its effect by considering the *convection velocity* at time t, instead of time t^{n+1},

$$\langle b_0 \hat{u} \boldsymbol{\nabla} \hat{u}, v \rangle^{n+1} \simeq \left\langle b_0 \hat{u}^n \boldsymbol{\nabla} \hat{u}^{n+1}, v \right\rangle . \tag{50}$$

With this strategy we obtain a semi-implicit scheme, which can be treated as a linear system.

2.3.2 BDF-1

Update BDF-1 proof, it is done with the heat equation. Should I move this to an appendix? Should I simplify this to be equation-independent?

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

$$\left\langle \frac{\partial \hat{u}}{\partial t}, v \right\rangle^{n+1} - \langle w \boldsymbol{\nabla} \hat{u}, v \rangle^{n+1} + \langle \varepsilon \boldsymbol{\nabla} \hat{u}, \boldsymbol{\nabla} v \rangle^{n+1} = + \langle f, v \rangle^{n+1} + \langle f_{g,1}, v \rangle^{n+1} + \langle f_{g,2}, \boldsymbol{\nabla} v \rangle^{n+1} . \tag{51}$$

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

$$\hat{u}(x,t) \simeq I_1(t) := \hat{u}^{n+1}\left(\frac{t-t^n}{\Delta t}\right) + \hat{u}^n\left(\frac{t^{n+1}-t}{\Delta t}\right), \tag{52}$$

$$\left.\frac{\partial \hat{u}}{\partial t}\right|^{n+1} \simeq \frac{dI_1(t)}{dt} = \frac{\hat{u}^{n+1}-\hat{u}^n}{\Delta t} \tag{53}$$

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

$$\begin{aligned}
&\left\langle \hat{u}^{n+1}, v\right\rangle - \Delta t\left\langle c\boldsymbol{\nabla}\hat{u}^{n+1}, v\right\rangle + \Delta t\left\langle \varepsilon\boldsymbol{\nabla}\hat{u}^{n+1}, \boldsymbol{\nabla}v\right\rangle \\
&+\Delta t\left(\left\langle b_0\hat{u}^{n+1}\boldsymbol{\nabla}g^{n+1}, v\right\rangle + \left\langle b_0 g^{n+1}\boldsymbol{\nabla}\hat{u}^{n+1}, v\right\rangle\right) \\
&+\left\langle b_0\hat{u}\boldsymbol{\nabla}\hat{u}, v\right\rangle^{n+1} \\
&= \left\langle \hat{u}^n, v\right\rangle - \Delta t\left\langle \frac{\partial g^{n+1}}{\partial t}, v\right\rangle \\
&-\Delta t\left\langle b_0 g^{n+1}\boldsymbol{\nabla}g^{n+1}, v\right\rangle - \Delta t\left\langle \varepsilon\boldsymbol{\nabla}g^{n+1}, \boldsymbol{\nabla}v\right\rangle \\
&+\Delta t\left\langle c^{n+1}\boldsymbol{\nabla}g^{n+1}, v\right\rangle
\end{aligned} \tag{54a}$$

Include BDF-2 discrete scheme.

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

2.4 Discrete Problem: Space and Time Discretization

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,

$$\hat{u}^n(x) \simeq \hat{u}_h^n(x) = \sum_j^{N_h} \hat{u}_{h_j}^n \varphi_j(x), \tag{55}$$

$$\hat{\mathbf{u}}_h^n = \left[\hat{u}_{h_j}^n\right]. \tag{56}$$

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

Applying the Galerkin principle to solve PDEs, we enforce the orthogonality of the residual to the functional space V_h. Because the domain changes with time, both the matrices and the vectors change for each timestep,

$$\left[\mathbf{M}_h^{n+1}\right]_{ij} = m_{\text{BDF}}\left\langle \varphi_j, \varphi_i\right\rangle, \tag{57a}$$

$$\left[\mathbf{A}_h^{n+1}\right]_{ij} = \left\langle \varepsilon\boldsymbol{\nabla}\varphi_j, \boldsymbol{\nabla}\varphi_i\right\rangle, \tag{57b}$$

$$\left[\mathbf{C}_h^{n+1}\right]_{ij} = -\left\langle c\boldsymbol{\nabla}\varphi_j, \varphi_i\right\rangle, \tag{57c}$$

$$\left[\mathbf{N}_h^{n+1}\right]_{ij} = b_0\left\langle u^n\boldsymbol{\nabla}\varphi_j, \varphi_i\right\rangle, \tag{57d}$$

$$\left[\hat{\mathbf{N}}_h^{n+1}\right]_{ij} = b_0\left(\left\langle g\boldsymbol{\nabla}\varphi_j, \varphi_i\right\rangle + \left\langle \varphi_j\boldsymbol{\nabla}g, \varphi_i\right\rangle\right), \tag{57e}$$

$$\begin{aligned}
\left[\mathbf{F}_{g,h}^{n+1}\right]_i = &-\left\langle \frac{\partial g}{\partial t} + b_0 g\boldsymbol{\nabla}g - c\boldsymbol{\nabla}g, \varphi_i\right\rangle \\
&-\left\langle \varepsilon\boldsymbol{\nabla}g, \boldsymbol{\nabla}\varphi_i\right\rangle,
\end{aligned} \tag{57f}$$

$$\left[\mathbf{F}_{\hat{\mathbf{u}}_h}^n\right]_i = \begin{cases} \left\langle \hat{u}_h^n, \varphi_i\right\rangle, & \text{BDF-1,} \\ \frac{1}{2}\left\langle \hat{u}_h^n, \varphi_i\right\rangle - \frac{3}{4}\left\langle \hat{u}_h^{n-1}, \varphi_i\right\rangle, & \text{BDF-2.} \end{cases} \tag{57g}$$

This leads to the following algebraic system:

$$\begin{aligned}
&m_{\text{BDF}}\mathbf{M}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} + \Delta t\mathbf{C}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} + \Delta t\mathbf{A}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} \\
&+\Delta t\hat{\mathbf{N}}_h^{n+1}\hat{\mathbf{u}}_h^{n+1} + \Delta t\left[\mathbf{N}_h^{n+1}\left(\hat{\mathbf{u}}_h^n\right)\right]\hat{\mathbf{u}}_h^{n+1} \\
&= \mathbf{F}_{\hat{\mathbf{u}}_h}^n + \Delta t\mathbf{F}_{g,h}^{n+1},
\end{aligned} \tag{58a}$$

$$\hat{\mathbf{u}}_h^0 = \hat{\mathbf{u}}_{h,0}. \tag{58b}$$

The spatial boundary conditions are encoded within the matrices and the vectors. The initial condition is obtain via interpolation or projection. The assembly of the forcing terms given in Equation (57f) 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.

The parameter m_{BDF} determines the time integration scheme used,

$$m_{\text{BDF}} = \begin{cases} 1, & \text{BDF-1,} \\ \frac{3}{4}, & \text{BDF-2.} \end{cases} \tag{59}$$

Regarding the forcing due to previous timesteps, $\left(\mathbf{F}_{\hat{\mathbf{u}}_h}^n\right)$, although for the FOM model we could compute the inner products at each timestep, for the Reduced Order Model we will exploit an algebraic expression of these expressions. It can be expressed

as the product between the mass matrix and the FE representation of the previous solution(s),

$$\mathbf{F}^n_{\hat{\mathbf{u}}_h} = \begin{cases} \mathbf{M}^{n+1}_h \hat{\mathbf{u}}^n_h, & \text{BDF-1,} \\ \frac{1}{2}\mathbf{M}^{n+1}_h \hat{\mathbf{u}}^n_h - \frac{3}{4}\mathbf{M}^{n+1}_h \hat{\mathbf{u}}^{n-1}_h, & \text{BDF-2.} \end{cases} \tag{60}$$

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.

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

$$\mathbf{K}^{n+1}_h \hat{\mathbf{u}}^{n+1}_h = \mathbf{b}^{n+1}_h, \tag{61a}$$

$$\hat{\mathbf{u}}^0_h = \hat{\mathbf{u}}_{h,0}, \tag{61b}$$

$$\mathbf{K}^{n+1}_h = \mathbf{M}^{n+1}_h + \Delta t \left[\mathbf{A}^{n+1}_h + \mathbf{B}^{n+1}_h + \hat{\mathbf{N}}^{n+1}_h + \mathbf{N}^{n+1}_h\left(\hat{\mathbf{u}}^n_h\right)\right], \tag{61c}$$

$$\mathbf{b}^{n+1}_h = \mathbf{F}^n_{\hat{\mathbf{u}}_h} + \Delta t \mathbf{F}^{n+1}_{g,h}. \tag{61d}$$

2.4.1 Wind-Up for the Full Order Problem

With all of the above, the generic Finite Element Method for the one-dimensional piston problem 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 semi-implicit time discretization scheme is used, to obtain a sufficiently stable algebraic system.

How does the semi-discretization play with the stability of the system?

A lifting of the Dirichlet boundary conditions has been introduced, which lead to the appearence of an additional forcing term and the homogenization of the boundary conditions. Hence, 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.

2.5 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.

2.6 Generalized Transformation

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

3 REDUCED ORDER MODEL

For any unseen parameter value, our aim is to be able to assemble and solve a smaller algebraic problem, and yet to obtain a solution close enough to that of the Full Order Model. 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; 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, that is, for the weak formulation given by the system of equations (54).

3.1 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 operator reduction procedures we will include, especially 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[4] problem-based basis functions $\psi_i(x)$,

$$\hat{u}_N(x) = \sum_j^N \hat{u}_{N_j} \psi_j(x). \tag{62}$$

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.

3.1.1 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 $\psi_i(x)$ and the nodal basis functions. This allows us to establish the following relation between the problem solution in the reduced and original spaces,

$$\hat{\mathbf{u}}_h = \mathbb{V}\hat{\mathbf{u}}_N. \tag{63}$$

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

$$\psi_i(x) = \sum_j [\mathbb{V}]_{ji} \varphi_j(x). \tag{64}$$

3.1.2 Discrete Reduced Problem Assembly

In theory, to assemble the reduced problem operators, one could actually compute the inner products defined in Equations (57) 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,

$$\mathbf{X}_N^{n+1} = \mathbb{V}^T \mathbf{X}_h^{n+1} \mathbb{V}, \tag{65a}$$

$$\mathbf{F}_N^{n+1} = \mathbb{V}^T \mathbf{F}_h^{n+1}, \tag{65b}$$

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

where $\mathbf{X}_h$ and $\mathbf{F}_h$ stand for each of the FOM operators (matrix and vector respectively). The assembly of matrices based on a FE basis can be 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,

$$\begin{aligned} m_{\text{BDF}}\mathbf{M}_N^{n+1}\hat{\mathbf{u}}_N^{n+1} + \Delta t\mathbf{C}_N^{n+1}\hat{\mathbf{u}}_N^{n+1} + \Delta t\mathbf{A}_N^{n+1}\hat{\mathbf{u}}_N^{n+1} \\ +\Delta t\hat{\mathbf{N}}_N^{n+1}\hat{\mathbf{u}}_N^{n+1} + \Delta t\left[\mathbf{N}_N^{n+1}\left(\hat{\mathbf{u}}_N^n\right)\right]\hat{\mathbf{u}}_N^{n+1} \\ = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t\mathbf{F}_{g,N}^{n+1}, \end{aligned} \tag{66a}$$

$$\hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}. \tag{66b}$$

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,

$$\mathbf{K}_N^{n+1}\hat{\mathbf{u}}_N^{n+1} = \mathbf{b}_N^{n+1}, \tag{67a}$$

$$\hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}, \tag{67b}$$

$$\begin{aligned} \mathbf{K}_N^{n+1} = \mathbf{M}_h^{n+1} + \Delta t\left[\mathbf{A}_h^{n+1} + \mathbf{B}_h^{n+1}\right. \\ \left.+\hat{\mathbf{N}}_h^{n+1} + \mathbf{N}_h^{n+1}\left(\hat{\mathbf{u}}_N^n\right)\right], \end{aligned} \tag{67c}$$

$$\mathbf{b}_N^{n+1} = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t\mathbf{F}_{g,N}^{n+1}. \tag{67d}$$

All of the previous has the same algebraic pattern as the FOM problem. The only difference is the size of the operators, much smaller due to the small size of N.

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,

$$\mathbf{F}_{\hat{\mathbf{u}}_N}^N = \mathbb{V}^T\mathbf{F}_{\hat{\mathbf{u}}_h}^n, \tag{68}$$

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 (60) in terms of the mass matrix. The forcing term due to the time discretization takes the following form in the ROM:

$$\mathbf{F}_{\hat{\mathbf{u}}_N}^n = \begin{cases} \mathbf{M}_N^{n+1}\hat{\mathbf{u}}_N^n, & \text{BDF-1}, \\ \frac{1}{2}\mathbf{M}_N^{n+1}\hat{\mathbf{u}}_N^n - \frac{3}{4}\mathbf{M}_N^{n+1}\hat{\mathbf{u}}_N^{n-1}, & \text{BDF-2}. \end{cases} \tag{69}$$

Again, these mimic the algebraic pattern obtained in the FOM.

3.1.3 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 via interpolation or projection. Then, this 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 (63),

$$\mathbb{V}^T\hat{\mathbf{u}}_{h,0} = \underbrace{\mathbb{V}^T\mathbb{V}}_{\mathbb{I}}\hat{\mathbf{u}}_{N,0} \rightarrow \hat{\mathbf{u}}_{N,0} = \mathbb{V}^T\hat{\mathbf{u}}_{h,0}. \tag{70}$$

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 elements $\psi_i(x)$ are zero at the boundaries,

$$\psi_i(x) = 0 \quad \forall\, x \in \partial\Omega.$$

These two facts imply that the projection of the operators, Equation (65), 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 (62), is always true.

There was a paper which discussed this kind of details regarding homogeneous boundary projection.

Once the reduced homogeneous solution $\hat{u}_N(x)$ is obtained, it can be brought back to the original

space V_h via Equation (63), and then add on top of it the FE representation of the Dirichlet lifting, to obtain the final solution,

$$u(x,t;\mu) \simeq u_h(t,\mu) = \mathbb{V}\hat{u}_N(t,\mu) + g_h(t,\mu). \quad (71)$$

3.1.4 Wind-Up for the Naive Reduction Scheme

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 choose defines which reduction method we are applying, along with their advantages and requirements.

The *offline-online decomposition*: that is, 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 *for each timestep*.

Hence, 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 3.2; and the approximation of the algebraic operators, Section 3.3.

3.2 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,

$$\begin{aligned}\Psi_{\hat{u}} &:= [\psi_i] \\ &= [\hat{u}_h(t^0;\mu_0), \hat{u}_h(t^1;\mu_0), \ldots, \hat{u}_h(t^j;\mu_i), \ldots], \\ &= [\Psi_{\hat{u}}(\mu_0), \Psi_{\hat{u}}(\mu_1), \ldots, \Psi_{\hat{u}}(\mu_{N_\mu})] \end{aligned} \quad (72)$$

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*.

3.2.1 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 $POD : X \to Y$ 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},

$$[\psi_i]_{i=1}^{N_{POD}} = POD\left([\varphi_i]_{i=1}^{N_S}, \varepsilon_{\text{POD}}\right). \quad (73)$$

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

$$\text{span}\,(\varphi_i) = \text{span}\,(\psi_i) + \varepsilon_{\text{POD}}, \tag{74}$$

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},

$$N_{POD} = N_{POD}(\varepsilon_{\text{POD}}). \tag{75}$$

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 $N^*_{POD} \leq N_S$,

$$[\psi_i]_{i=1}^{N^*_{POD}} = POD\left([\varphi_i]_{i=1}^{N_S}, N^*_{POD}\right). \tag{76}$$

3.2.2 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,

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

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

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

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.

3.3 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 (65), 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,

$$A_h(t,\mu) = \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{h,q}, \tag{77}$$

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,

$$\begin{aligned} A_N(t,\mu) &= \mathbb{V}^T A_h(t,\mu)\mathbb{V} \\ &= \sum_q^{Q_a} \Theta_q^a(t,\mu)\mathbb{V}^T A_{h,q}\mathbb{V} \\ A_N(t,\mu) &= \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{N,q}. \end{aligned} \tag{78}$$

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.

3.3.1 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 3.2.2, 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,

$$A_h(t,\mu) \simeq A_h^m(t,\mu) = \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{h,q}. \tag{79}$$

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

$$A_N(t,\mu) \simeq A_N^m(t,\mu) = \sum_q^{Q_a} \Theta_q^a(t,\mu) A_{N,q}, \tag{80}$$

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

3.3.2 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.

3.3.3 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,

$$[A_h(t,\mu)]_k = [A_h^m(t,\mu)]_k = \sum_q^{Q_a} \Theta_q^a(t,\mu)[A_{h,q}]_k \tag{81}$$

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 (79),

$$e_a(t,\mu) = \|A_h(t,\mu) - A_h^m(t,\mu)\|. \tag{82}$$

3.3.4 Reduction of the Nonlinear Term

The basis for the nonlinear operator C_h can be built in many ways, but there is one that will allows us to make it as rich as possible in an efficient way.

The nonlinearity of this operator comes from the fact that it depends on the solution from previous timesteps. Thus, one way to approach the reduction of this operator would be to collect the operator snapshots during the offline phase of the FOM. The problem with this approach is that we would be tied by the parameter space used for the reduced basis.

Instead, if we use the reduced basis, we could span a larger parameter space. We run the offline phase for the nonlinear operator once we have obtained the reduced basis.

We use a nested POD approach again, althoguh this time we have three levels.

We fix a parameter, then for each reduced basis element we collect as many snapshots as timesteps. Then we compress these snapshots to obtain a collateral basis, which we store. When we are done with all the elements in the reduced basis, we have a collection of collateral basis, which we compress. This gives us a basis which contains all the information about the solution for that parameter. We repeat this procedure for each parameter, thus obtaining a collateral basis for each parameter. Finally, we compress these collateral basis to obtain the final collateral basis with information about the solution and the parameter space.

3.4 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,

$$\mathbf{M}_N^{m,n+1}\hat{\mathbf{u}}_N^{n+1} + \Delta t\mathbf{A}_N^{m,n+1}\hat{\mathbf{u}}_N^{n+1} = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t\mathbf{F}_N^{m,n+1} + \Delta t\mathbf{F}_{g,N}^{m,n+1}, \tag{83a}$$

$$\hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}. \tag{83b}$$

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,

$$\mathbf{K}_N^{m,n+1}\hat{\mathbf{u}}_N^{n+1} = \mathbf{b}_N^{m,n+1}, \tag{84a}$$

$$\mathbf{K}_N^{m,n+1} = \mathbf{M}_N^{m,n+1} + \Delta t\mathbf{A}_N^{m,n+1}, \tag{84b}$$

$$\mathbf{b}_N^{m,n+1} = \mathbf{F}_{\hat{\mathbf{u}}_N}^n + \Delta t\mathbf{F}_N^{m,n+1} + \Delta t\mathbf{F}_{g,N}^{m,n+1}, \tag{84c}$$

$$\hat{\mathbf{u}}_N^0 = \hat{\mathbf{u}}_{N,0}. \tag{84d}$$

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 (81) 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 (63), and then add on top of it the FE representation of the Dirichlet lifting, to obtain the final solution,

$$u(x,t;\mu) \simeq u_h^m(t,\mu) = \mathbb{V}\hat{u}_N^m(t,\mu) + g_h(t,\mu). \tag{85}$$

3.5 Certification of the Reduction Procedure

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

3.5.1 Sacrifical Mode

We can use two ROMs to estimate the error with respect to the FOM solution.

$$\|u_h - \mathbb{V}_N u_N\| = \|u_h - \mathbb{V}_{N+1}u_{N+1} + \mathbb{V}_{N+1}u_{N+1} - \mathbb{V}_N u_N\|. \tag{86}$$

By the triangle inequality, we get

$$\|u_h - \mathbb{V}_N u_N\| \leq \|u_h - \mathbb{V}_{N+1}u_{N+1}\| + \|\mathbb{V}_{N+1}u_{N+1} - \mathbb{V}_N u_N\| \tag{87}$$

If we define the error function $e_N = \|u_h - \mathbb{V}_N u_N\|$, we get an upper bound of the desired error in terms of the sacrificial ROM error and the error between ROMs,

$$e_N \leq e_{N+1} + \|\mathbb{V}_{N+1}u_{N+1} - \mathbb{V}_N u_N\|. \tag{88}$$

With sufficient ROB[5] elements, it should be safe to assert that the error made with an extra mode should be smaller or equal to the one made without it,

$$e_{N+1} \leq e_N. \tag{89}$$

Thus, we get the following error bound,

$$e_N \leq \|\mathbb{V}_{N+1}u_{N+1} - \mathbb{V}_N u_N\|. \tag{90}$$

It remains to determine how sharp this error estimate is.

ROMs error: Implementation Details

The error between the two ROMs,

$$\|\mathbb{V}_{N+1}u_{N+1} - \mathbb{V}_N u_N\|, \tag{91}$$

can be expressed as a sum in terms of the ROB elements. Due to the hierarchical character of the ROB elements, $\mathbb{V}_{N+1}$ contains the same elements up to N as $\mathbb{V}_N$. Thus, the error between ROMs can be expressed like

$$\|\mathbb{V}_{N+1}u_{N+1} - \mathbb{V}_N u_N\| = \left\| u_{N+1}^{N+1}\psi_{N+1} + \sum_j^N (u_j^{N+1} - u_j^N)\psi_j \right\| \tag{92}$$

The difference between ROM coefficients $(u_j^{N+1} - u_j^N)$ should be relatively small, since it represents the difference between the two coefficients associated to the same mode. If they were notably different, it would mean that the dynamics between the original ROM and the one carrying the sacrificial mode are different. Since the basis has a hierarchical character, this effect is unlikely to happen: adding a mode should only refine the solution, not change how the previous modes are scaled. They are not strictly the same because the ROM system entries change if more modes are added to the basis, but again, they should do so in a way that somehow preserves dynamics.

5. Reduced Order Basis

4 PISTON MOTION HYPERREDUCTION

Piston probes at three different locations. It shows that the model contains nonlinearities.

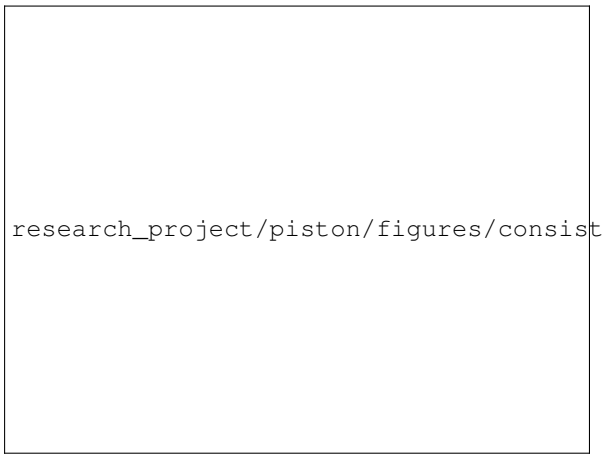

Fig. 1. Solution probes at three different locations of the piston. Top: Outflow velocity. Bottom: Piston motion (boundary condition).

4.1 Discretization Consistency

We use mass conservation as a criteria to determine whether the numerical scheme is consistent. Figure 2 shows plots for mass conservation for the BDF-1 and BDF-2 schemes.

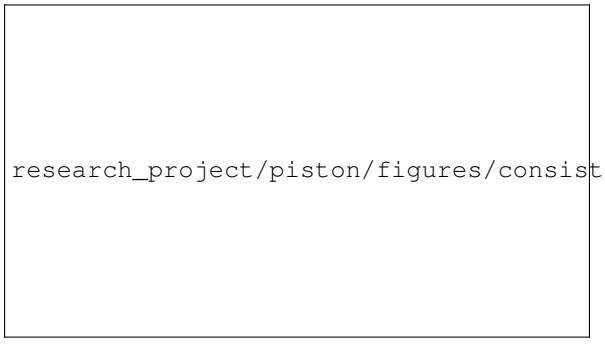

Fig. 2. Conservation of Mass, comparison between BDF-1 and BDF-2. Cont.: $d/dt \int_0^{L(t)} \rho(x,t)dx$. Dashed: Outflow, $\rho(0,t)u(0,t)$.

4.1.1 Conclusions

- The scheme seems consistent in the time variable, for the smaller we make the timestep, the smaller the error in the conservation of mass, as shown by Figure 4.
- Figures 2 and 3 show a comparison between two time marching schemes. The BDF-2 scheme seems to capture better the dynamics, for the only part where the outflow is different from the density time variation is near the upper peak.
- The fact that we do not see a slop of order two in Figure 4 could be due to the nature of the variable we are measuring, not necessarily a bug.
  - According to Figure 3, the BDF-2 scheme captures better the dynamic of mass conservation.
  - The L2 errors is computed as the differ-

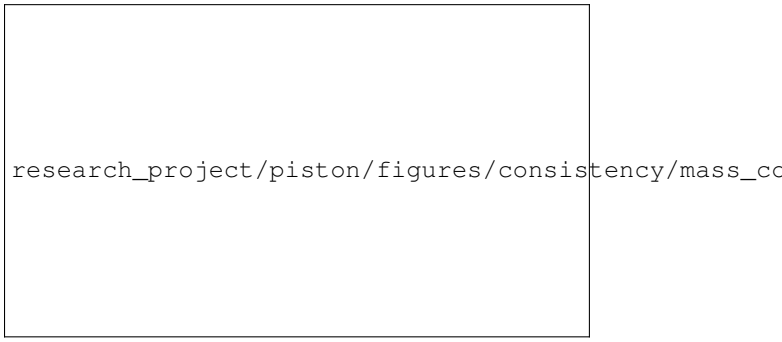

Fig. 3. Conservation of Mass, comparison between BDF-1 and BDF-2 (zoom). Cont.: $d/dt \int_0^{L(t)} \rho(x,t)dx$. Dashed: Outflow, $\rho(0,t)u(0,t)$.

Fig. 4. Conservation of Mass consistency.

ence between outflow and mass variation, divided by N_t. I think that is the reason we see the same slope twice, because we are seeing $dt \sim N_t^{-1}$ vs. N_t^{-1} (and yet, the BDF-2 curve is underneath the BDF-1).

4.2 System Approximation: DEIM and MDEIM

Due to the simplicity of the geometry (1D piston), the linear operators are reduced effortlessly. We recall that to build the collateral space (the basis for the algebraic operators), we are using a nested POD approach:

1) Select parameter, assemble in time, compress, collect basis.
2) Update parameter, repeat assembly and compression in time.
3) Compress all the collected basis for each parameter.

TABLE 1
Number of basis after the tree walk (collection of basis per parameter) and the final basis size once the final POD is applied. Five parameters were sampled for the offline phase of the collateral basis.

	basis-shape-after-tree-walk	basis-shape-final
mass	5	1
stiffness	5	1
convection	10	2
nonlinear-lifting	5	1
rhs	12	4
nonlinear	90	18
nonlinear	465	93

As we can see in Table 1, linear operators can be resolved by a few number of basis. This is so because their main nonlinearity is the implicit jacobian which is present due to the time-dependency of the domain. Since it is a very simple domain (1D piston), the jacobian is a function of time and space,

$$J \sim L'(t)x, \tag{93}$$

so its reduction is immediate.

The RHS requires a bit more of terms because it contains the action of all the operators unto the lifting function.

4.2.1 Nonlinear term

To assemble the nonlinear term, we use the reduced basis modes as a proxy for the actual solution which

is used in the discretization. This is a fair approach, since eventually the solution will be a linear combination of such modes. Thus, the number of basis after the tree walk depends not only in the number of parameters, but also in the number of reduced basis elements. Again, a three-level nested POD approach is used:

1) For a given parameter, for each timestep:
    a) Assemble for each basis element, compress, collect.
    b) Update time, repeat assemly and compression in reduced basis space.
2) Update parameter, repeat assembly and compression in time.
3) Compress all the collected basis for each parameter.

A very decent reduction ratio can be achieved.

4.2.2 Online errors

With the collateral basis, we sample unseen parameters and assemble the operators to evaluate our collateral basis. As expected given the simplicity of the implicit nonlinearity, they are reconstruced to machine accuracy, as shown in Figures 5, 6, 7, 8, 9, 10 and 11.

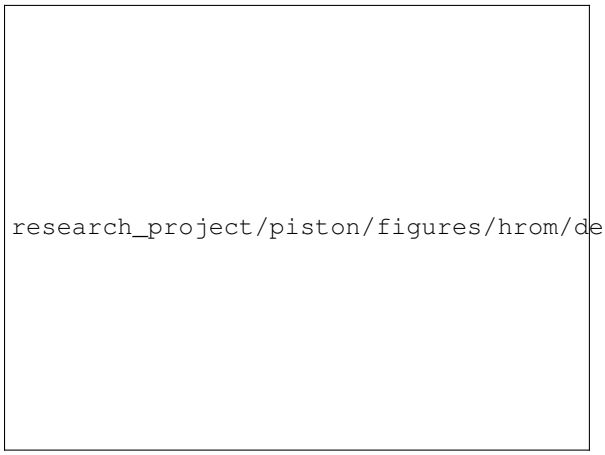

Fig. 5. Online errors: Rhs functional.

Fig. 6. Online errors: mass operator.

Fig. 7. Online errors: stiffness operator.

Fig. 8. Online errors: convection operator. It contains the speed of sound and the ALE velocity.

Fig. 9. Online errors: nonlinear lifting operator (it is the linearized nonlinear operator).

Fig. 10. Online errors: nonlinear operator with 50%-energy basis, poor reconstruction.

4.3 Model Reduction: Tolerances in the Parameter Space

We recall that to build the reduced space, we are using a nested POD approach:

- Select parameter, solve in time, compress, collect basis.
- Update parameter, repeat solution and compression in time.
- Compress all the collected basis for each parameter.

All the parameters are selected probing an uniformly distributed random variable, see Figure 12.

We ran three tests:

1) **Benchmark**: automatic basis selection, all basis elements whose singular values are $\sigma > 10^{-7}$ are kept. This is the most complete reduced basis one can get.
2) **Half-Space** (Fix tolerance in the parameter space for the solution and the nonlinear term): we keep all the basis elements which explain up 50% of the variance. This is a strong reduction criteria.
3) **Reduced Solution Space**: we keep the automatic basis selection for the collateral basis (algebraic operators) and maintain the 50% criteria for the reduced basis.

For each of them we ran the ROM unto the training parameter set and the online set, which was never seen before.

4.3.1 Benchmark

Figures 13 and 14 for ROM errors.

- The ROM achieves absolute accuracy (10^{-6}) for both validation and online sets.

4.3.2 Half-Space

In Figure 15 we plot the reduced space, with a clear non-linear character.

Figures 16 and 17 for ROM errors.

- The ROM loses two orders of magnitude in accuracy (10^{-4}), and has the same behaviour for both the validation and online set.

4.3.3 Reduced Solution Space

Figures 18 and 19 for ROM errors.

- The ROM's accuracy is now widespread in terms of accuracy ($(10^{-6}, 10^{-4})$), behaving slightly worse for the online set.

4.4 Conclusions

Regarding discretization:

- BDF-2 is a must (even higher order terms should be considered). It allows the reduction of timesteps to use, which speeds up the offline stage, which is already quite heavy. By introducing such scheme, for $t \in [0, 1.0]$, I could go from $Nt = 1000$ to $Nt = 500$, which reduced the offline to about an hour instead of two/three hours.

Regarding model reduction:

- The model captures correctly nonlinearities.
- Linear operators are so simple a few collateral basis elements are enough to reduce them. This is due to the simplicity of the jacobian transformation between moving and reference domains.
  - No point in playing with the tolerances for linear operators.
- Reduction of the nonlinear term is a costly operation, because the integral of the basis elements cannot be efficiently computed due to its global support.
- If collateral basis elements for the nonlinear operator are neglected, the solution loses its physical accuracy.
- The reduced basis can be trimed without affecting too much the solution (not as much as the nonlinear operator).
- It would be interesting to have analytical results about the relation between parameter space tolarence and online errors.

4.5 Future Work

With this work so far we have proved that the ROM with reduced basis for the solution *and* the operators in moving domains works.

There is one flaw though, we can only assess the bondness of the ROM by computing the FOM solution too for the same parametrization. This makes

it cumbersome, since we would like to compute the error without having to compute the FOM solution too.

There are two possible paths now:

- Extend the same procedure to a 2D domain.
- Certify the ROM with the development of *a posteriori* error computations.

Calculation for a 2D domain is more number-crunching and coding work, no new knowledge will be learned. Bug prone, non-mathematical work overhead (implementation details). If achieved, none of the models are certified: online errors can only be computed by solving the FOM problem too. Adds to the body of knowledge in a can-be-done way, not learning anything we cannot learn from the 1D problem about the implementation details.

Calculation of a posteriori error computation requires error bounds and the extension of some offline structures. More uncertainty is involved, but it gives actual closure to the ROM problem. Mathematical work, adds to the body of knowledge.

4.5.1 Extension to 2D

Extension of the procedure to a 2D setting would require:

- Construction of a FOM solver. Requirements:
  - To build the final algebraic operator by the sum of diverse operators.
  - To be semi-implicit.
- Rewriting the (M)DEIM reductors, they assume a 1D setting.
- Module to create harmonic extensions for lifting and mesh displacement.
- Updating the move mesh module. Right now it simply scales the mesh.
- Learning to use Paraview to plot the solution.

4.5.2 ROM Certification: A Posteriori Errors

There is previous work on time-dependent matrices for fixed domains.

A posteriori errors would require:

- Construction of Rietz reprensentant.
- Calculation of eigenvalues.
- Literature reading.

APPENDIX

Since we are going to impose the movement of the piston, we need to make sure we do so in a physically meaningful way.

This appendix is born after making the mistake of believing that any sinusoidal function would do the job. Since our flow will depart from rest, we need to set the piston motion so that for the initial instant the piston will depart from rest too.

To derive the movement, we depart from a force defined by a sinus and cosinus functions,

$$A\cos(\omega t) + B\sin(\omega t) = m\ddot{x}(t), \tag{94a}$$

$$x(0) = 0, \tag{94b}$$

$$\dot{x}(0) = \ 0. \tag{94c}$$

Integrating in time, we get

$$\frac{A}{m\omega}\sin(\omega t) - \frac{B}{m\omega}\cos(\omega t) + C_1 = \dot{x}(t). \tag{95}$$

If we enforce the initial condition of rest to find the value of the integration constant, we will see how we arrive to an incongruency.

$$-\frac{B}{m\omega} + C_1 = 0 \rightarrow C_1 = \frac{B}{m\omega}. \tag{96}$$

If we integrate again in time to obtain the movement of the piston, due to the presence of C_1, a linear term in time $\sim\ t$ will show up. This makes no sense, given that we departed from two sinusoidal functions, which input and remove energy with fixed frequency and amount from the system. Hence a harmonic movement is expected.

This conflictive result comes from the presence of the sinusoidal term in the definition of the force moving the piston. If we set $B = 0$, the first integration constant will become null, $C_1 = 0$. When this is the case, the linear term vanishes, and we recover a harmonic piston movement,

$$\frac{A}{m\omega^2}\cos(\omega t) + C_2 = x(t). \tag{97}$$

By setting the initial location, we get the value for C_2,

$$x(t) = L_0 - \frac{A}{m\omega^2}\left(1 - \cos(\omega t)\right). \tag{98}$$

If we now define A such that $\frac{A}{m\omega^2}$ represents a fraction of the initial piston length, δL_0, we get a compact expression for the piston movement,

$$x(t) = L_0\left[1 - \delta\left(1 - \cos(\omega t)\right)\right]. \tag{99}$$

Fig. 11. Online errors: nonlinear operator with full basis. Exact reconstruction.

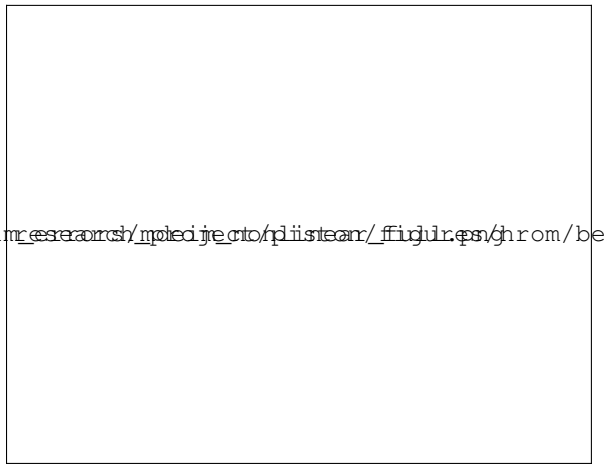

Fig. 14. (Benchmark) Online errors.

Fig. 12. Offline vs. Online parameter space for the Reduced Basis. 20 parameters were selected for each stage.

Fig. 15. (Half-Space) Reduced Space Elements. From top to bottom by decreasing level of energy. We can see how the first nodes have larger wavelengths, wereas the last ones present faster vibrations.

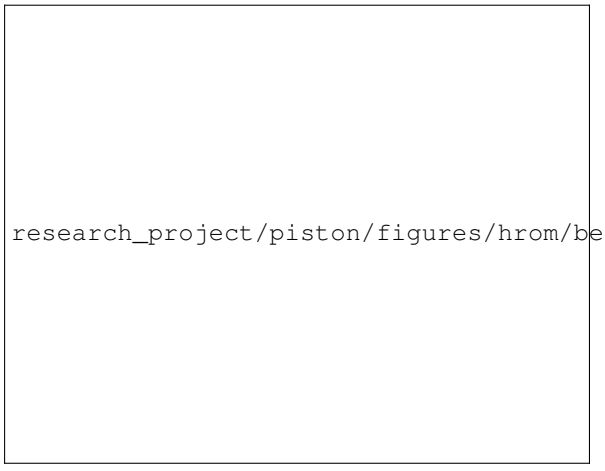

Fig. 13. (Benchmark) Validation errors.

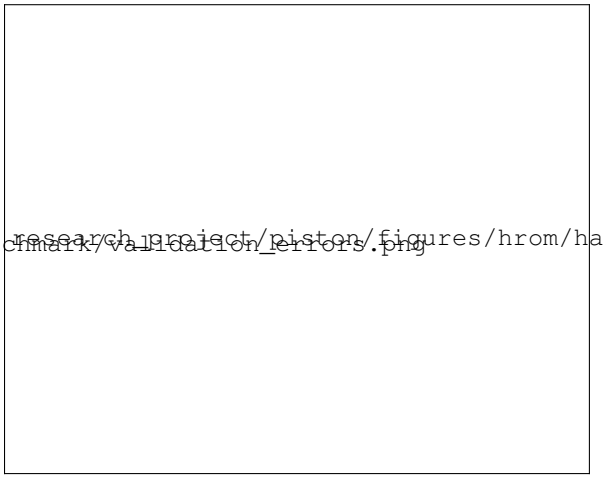

Fig. 16. (Half-Space) Validation errors.

Fig. 17. (Half-Space) Online errors.

Fig. 18. (Reduced Solution Space) Validation errors.

Fig. 19. (Reduced Solution Space) Online errors.