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,

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

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?