Thermal Boundary Layer Energy Equation for Liquid Metal Flow

Corbin Foucart

1 Problem Statement

We investigate boundary layer heat transfer of fluid flowing over a heated flat plate. The governing equations are derived in [1] as:

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \tag{1}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \mu\frac{\partial^2 u}{\partial y^2} \tag{2}$$

$$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{k}{\rho c_p}\frac{\partial^2 T}{\partial y^2} + \mu\left[\frac{\partial u}{\partial y}\right]^2 \tag{3}$$

Where u, v are the x and y velocities respectively, p is the pressure, T is the spatially-varying fluid temperature in the boundary layer, and κ, ρ, c_p are the thermal conductivity, density, and specific heat of the fluid, respectively. We are interested in the temperature profile close to the heated plate, and so we make the following assumptions:

1. No vertical component of velocity: $v = 0$
2. Specified flow profile: $u(y)$
3. Neglect viscous dissipation term: $\mu\left[\frac{\partial u}{\partial y}\right]^2$ small compared to other terms

Then the governing equation in this regime reduces to:

$$[u(y)]\frac{\partial T}{\partial x} = \frac{k}{\rho c_p}\frac{\partial^2 T}{\partial y^2} \tag{4}$$

1.1 Non-dimensionalization of governing equations

Since we don't know what physical parameter values the user is going to provide, we don't want to run into scaling issues; therefore we can accept as input the physical parameters, but solve a non-dimensionalized version of the problem.

We define $\alpha = \frac{k}{\rho c_p}$, and note the following characteristic problem parameters: the characteristic length scale L of the plate, the free-stream velocity U_∞, the heat flux q in [W/m^2], and free stream temperature T_∞. We consider the non-dimensional quantities

$$x^* \equiv \frac{x}{L} \qquad y^* \equiv \frac{y}{L} \tag{5}$$

$$u^* \equiv \frac{u}{U_\infty} \qquad T^* \equiv \frac{k(T - T_\infty)}{qL} \tag{6}$$

where $\mathrm{Pe} = \mathrm{Re}_L\mathrm{Pr} = \frac{LU_\infty}{\alpha}$ is the Peclet number. The above definitions give rise to the following non-dimensional governing equation

$$u^* \frac{\partial T^*}{\partial x^*} = \frac{1}{\mathrm{Pe}} \frac{\partial^2 T^*}{\partial {y^*}^2} \tag{7}$$

For the boundary conditions, in the case of the plate being heated by a resistance heater or the like, we impose a constant flux boundary condition on Γ_1

$$\left[k \frac{dT}{dy} \right]_{\Gamma_1} = q \qquad \rightarrow \qquad \left. \frac{dT^*}{dy^*} \right|_{\Gamma_1} = 1 \tag{8}$$

similarly, the far-field condition at Γ_2 is that $T = T_\infty \Rightarrow T^* = 0$. Having completed the non-dimensionalization of the problem, we are prepared to state the strong form.

1.2 Strong form

We can therefore write the complete strong form of the problem as:

$$u^* \frac{\partial T^*}{\partial x^*} = \frac{1}{\mathrm{Pe}} \frac{\partial^2 T^*}{\partial {y^*}^2} \quad \text{on } \Omega = (\Gamma_1, \, \Gamma_2) \tag{9}$$

subject to boundary conditions

$$\left. \frac{dT^*}{dy^*} \right|_{\Gamma_1} = 1, \; T^*(x^*, \, \Gamma_2) = 0 \tag{10}$$

$$\tag{11}$$

Which is the problem which will be discretized and solved numerically. The user will enter as inputs to the software L, U_∞, q, T_∞, Pe. The only concern is that the user must enter an L which is large enough to contain the thermal boundary layer.

1.3 Approximation of $u(y)$ and the computational domain

The last piece of the problem statement is the types of velocity profiles to be considered. In the physical regime of $\mathrm{Pr} \ll 1$, the thermal boundary layer grows much faster than the momentum boundary layer, and we can approximate the velocity field u^* as steady in x. We consider the two approximations of $u^* = U_\infty$ and a linear profile $u^* = y^*$.

2 Weak Formulation and Discretization

In this section, we drop the non-dimensionalized * notation for convenience. Note that our inhomogenous Dirichlet condition in the dimensional problem reduces to a homogeneous Dirichlet condition in the non-dimensional equation. Therefore, our finite element spaces X^E and X coincide

$$X^E = X = \left\{ v \in H^1(\Omega) \mid v(\Gamma_2) = 0 \right\} \tag{12}$$

and we can write our semi-discretized weak form as

$$\text{find } w_h(x) \in X^E \text{ such that } m\left(u \frac{dw_h}{dx}, \, v \right) + a(w_h, v) = f(v) \tag{13}$$

Where the weak form operators are defined as (see Appendix A):

$$m(T, v) = \int_\Omega u(y) \frac{dT}{dx} v \, dy \tag{14}$$

$$a(T, v) = \frac{1}{\mathrm{Pe}} \int_\Omega \frac{dv}{dy} \frac{dT}{dy} \, dy \tag{15}$$

$$f(v) = \int_\Omega f_\Omega v \, dy - \frac{1}{\mathrm{Pe}} v(\Gamma_1) \tag{16}$$

We perform a backward euler discretization in time (since it is unconditionally stable). Then we can state our finite element problem for any given x as the following: find $w_h(x) \in X^E$ such that

$$m\left(u \frac{w_h^{k+1} - w_h^k}{\Delta x},\, v\right) + a\left(w_h^{k+1}, v\right) = f(v) \tag{17}$$

which is equivalent to

$$\frac{1}{\Delta x} m\left(u\, w_h^{k+1},\, v\right) + a\left(w_h^{k+1}, v\right) = f(v) - \frac{1}{\Delta x} m(u\, w_h^k,\, v) \tag{18}$$

which completes the problem discretization. Details on the implementation of the time-stepping operators can be found in Appendix B.

3 Implementation Details

The entire code for the project (numerics, verification, visualization) is written in pure `Python` in the contained iPython notebook `Project_1.ipynb`. For numerical integration, we use quadrature rules from the open source library `quadpy` [2]. The notebook additionally contains extended commentary and thorough discussion of implementation details, with the intention of walking the reader through the use and development of the code. It can be run in part or in its entirety. I have also included a static `html` file containing the output of the notebook run from beginning to end. The following section provides an abbreviated summary of notebook sections 4-6 (Verification, Re-dimensionalization, and Physical Test Cases). For the details in their entirety, consult either the output or run the notebook directly.

4 Brief Overview of Results

Verification of the code primarily involved test cases using the method of manufactured solutions over the non-dimensional computational domain. We provide a temporal convergence study and a discussion of variable "time-stepping". The user enters the problem parameters in dimensional form; `Project_1.ipynb` discusses the implementation of re-dimensionalizing the problem and returning the solution on the physical domain, as well as visualization techniques (to which we refer the interested reader). Lastly, we consider two physical test cases for low-Prandtl liquid metal flows.

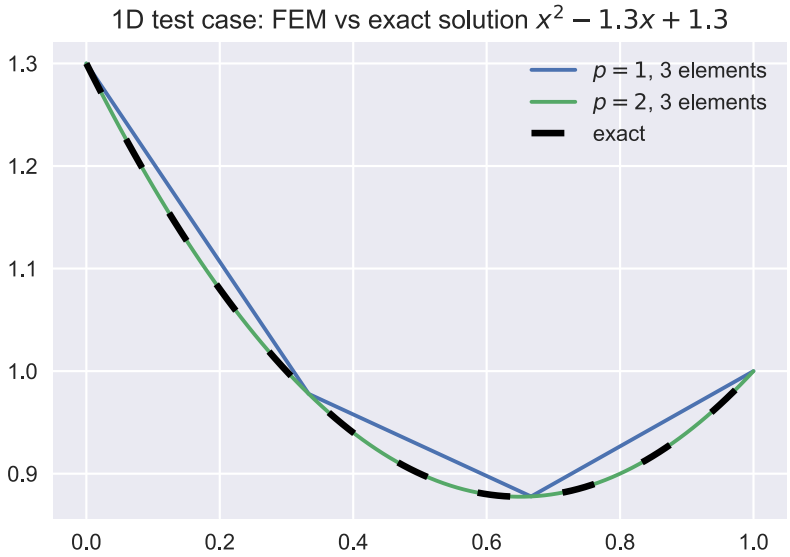
4.1 Verification

4.1.1 Verification of 1D Implicit FE Solver

Before testing the 2D code, it makes sense just to test the 1D implicit spatial solver is working correctly. We test on a small spatial subproblem with strong form

$$0 = \frac{d^2 u}{dx^2} - f_{ex} \tag{19}$$

where we choose f_{ex} as a constant function in space. It doesn't get much simpler than this. The general solution is $u = x^2 + bx + c$ and we can symbolically determine the coefficients for any choice of boundary conditions g_N, g_D. To check that our 1D boundary conditions are implemented correctly, we choose an exact solution with inhomogeneous boundary conditions (both for Dirichlet and Neumann boundaries). The results of the manufatured solution test are shown below.



The linear solution looks good and the quadratic solution matches exactly, as we would expect, since the exact solution is an element of our finite element approximation space. We could be a bit more rigorous about measuring the error here (by choosing an MS outside of a polynomial space and measuring error with quadrature), but we consider this overkill for 1D.

4.1.2 Verification of 2D "Temporal" FE Solver

We choose a manufactured solution which satisfies our non-dimensional equation, i.e., some field $w(x, y) = f(x)g(y)$ such that

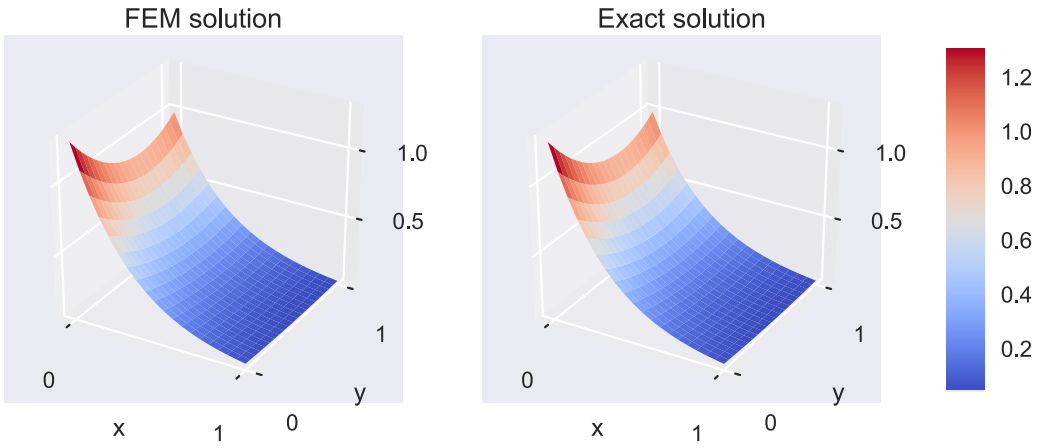
- $w(x, 1) = g_D$ (dirichlet 0 condition at $y = 1$)
- $w\frac{dw}{dy}(x, 0) = g_N$ (neumann flux condition)

For this, we can simply take our exact solution to the 1D test cast and multiply it by some $f(x)$, here we choose a decaying exponential $\exp(-3x)$. We then compute the analytical forcing function f_{ex} necessary to give us our chosen analytical solution.

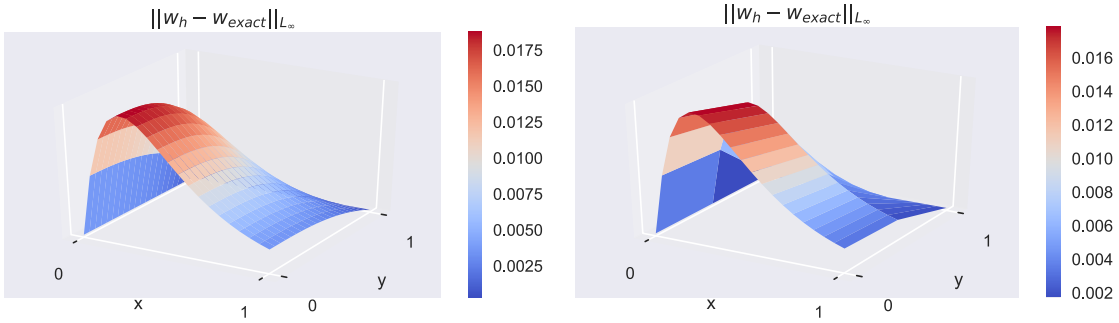
$$w(x,\, y) = \left(y^2 - 1.3y + 1.3\right) e^{-3x} \tag{20}$$

$$f_{ex} = 3y\left(y^2 - 1.3y + 1.3\right) e^{-3x} - 2e^{-3x} \tag{21}$$

For good measure, we choose a velocty profile $u(y) = y$ to test that our quadrature rules are integrating the mass operator with the velocity field well. The results of a test at reasonable resolution are below.



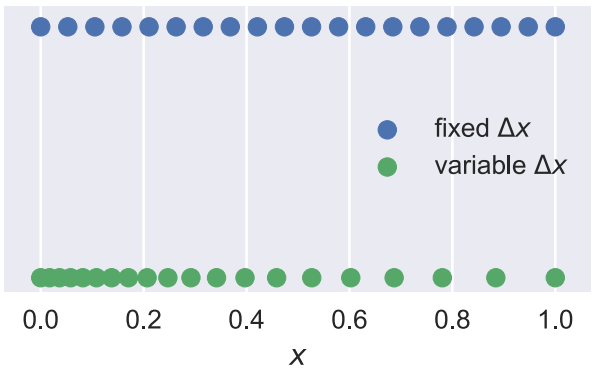
Additionally, we can plot the L_∞ error; computing other error norms is dicey since the time and space domains are treated differently – at least it isn't clear that we can meaningfully talk about a finite element solution defined at x points not at Δx increments. As we expect, the error is close to location corresponding to the beginning of the plate in the numerical domain. We also note that our spatial discretization is enormous overkill. Simply taking a single $p = 2$ element should reproduce the exact solution for each 1D problem, since the solution of every spatial problem lies in our $p = 2$ finite element space by construction.



Indeed, we see that the L_∞ error is unaffected by coarsening the spatial mesh to a single element. It remains to investigate the temporal error.

4.1.3 Variable "Timestepping"

Because of the particular features of our problem, we expect numerical issues to occur at the leading edge of the heated plate. We would like to investigate the effect of changing the computational "temporal" domain in x to be more refined closer to the leading edge of the plate.

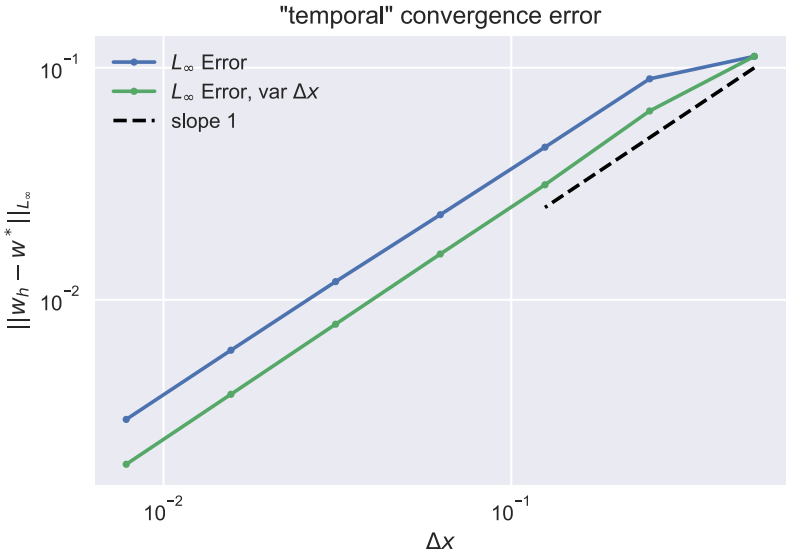


Above, we see sample output of a warping function which warps the grid according to an exponential trans-

form. The code is written sufficiently generally that we can provide a variable timestep.

4.1.4 Convergence Study

Since we have no spatial discretization error, we can measure temporal error to see if we can recover order 1 convergence in the L_∞ norm. We can also compare how variable timestepping does against regular timesteps. Note that we are running with $p = 2$ elements where we know the spatial solution is analytically quadratic in order to make sure spatial error is unimportant.



We see that both are converging to first order in time, but that the coefficient on the error is a bit lower for the warped grid. As a word of warning, though, we can only get this nice feature because we knew where we expected the solution have a steep gradient in time. If we knew nothing about our problem, using the variable timestepping in this manner would not help. It just so happens that for our client application, we expect the area of interest to be the start of the plate.

4.2 Physical Test Cases

Now that we are confident that our code is giving the correct solution to the appropriate convergence rates for problems slightly more general than our problem of interest, we can consider some physical test cases.

4.2.1 Liquid Mercury

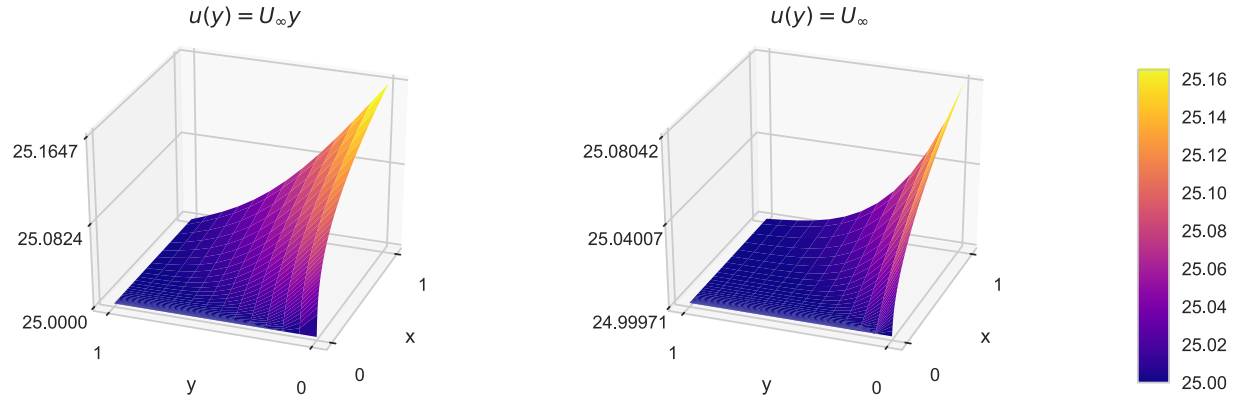
Test case setup:

```
# liquid mercury -- low Prandtl number
L     = 1.0      # meter
U_inf = 5e-2     # m/s
k     = 8.18     # W / (m deg C)
q     = -2       # W/m^2
T_inf = 25       # degrees C
mu    = 0.16e-2  # N s / m^2
rho   = 13595    # kg/m^3
cp    = 0.1404   # J/kgK

# compute Pr number, make sure it's low
Pr = cp * mu / k
```

```
ReL = rho * U_inf * L / mu
Pe = ReL * Pr
```

From where we see that Pr $= 1.75e-5$, Pe $= 11.7$, in the physical regime in which our approximations comfortably hold. We solve the numerical problem on a warped grid, and visualize the boundary layer temperature with both height and color. The results can be seen below.



We can see that when $u = U_\infty$ is constant, the boundary layer growth is slower, as we would physically anticipate.

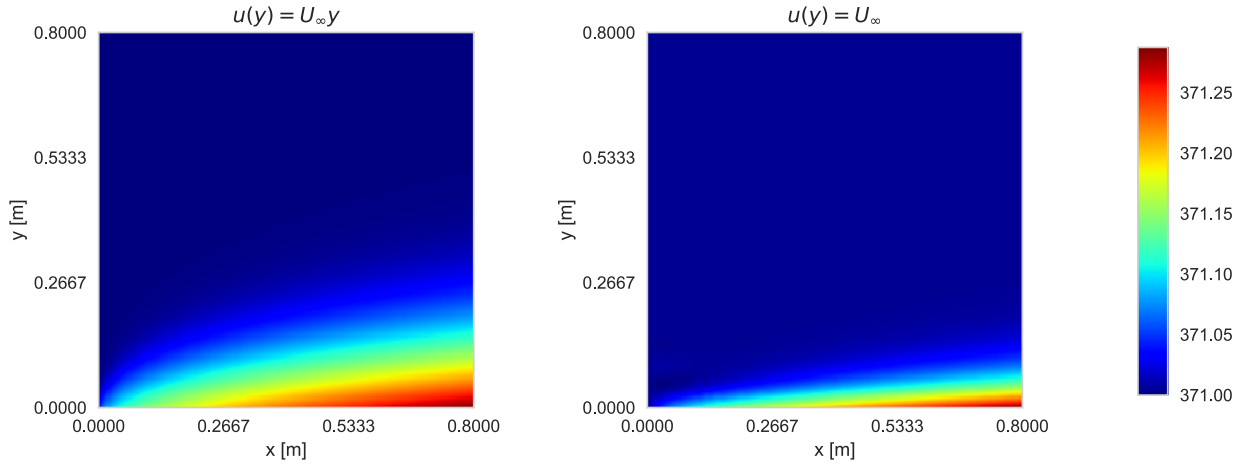
4.2.2 Molten Lead

Test case setup:

```
# molten lead -- low Prandtl number
L     = 0.8      # meter
U_inf = 2e-3     # m/s
k     = 16.1     # W / (m deg C)
q     = -20      # W/m^2
T_inf = 371      # degrees C
nu    = 2.27e-7  # m^2/s
alpha = 1.08e-5  # m^2/s
cp    = 0.159e3  # J/kgK

# compute Pr number, make sure it's low
Pr = nu / alpha
ReL = U_inf * L / nu
Pe = ReL * Pr
```

Here, Pr $= 0.021$, Pe $= 148$, again in the physical regime in which our approximations hold, although in this test case, we expect the boundary layer to stay much closer to the edge of the computational domain. We solve the problem, once again on the warped grid, and visualize the thermal boundary layer with only color (since it is so close to the wall). Further, we deliberately use a non-perception neutral colormap `jet` to accentuate differences in temperature. Results can be seen below.



There are some interesting features in this case. Although difficult to see, at the leading edge of the heated plate for the $u(y) = U_\infty$ case, the solution has small Gibbs oscillations due to the "free-slip" constant velocity profile. Note that for the velocity field $u(y) = y$, the velocity vanishes at the plate, and the oscillations are not present. In each case, the boundary layer does not develop to the extent of the computational domain.

A Weak Form: Derivation

We handle the non-dimensional problem, but omit the * notation for readability. we additionally use η to denote x^*, the pseudo-time variable, and y to denote the spatial coordinate y^*. We multiply by a test function $v \in X$ and integrate over the spatial domain[1] $\Omega = (\Gamma_1 = 0, \Gamma_2)$.

We also consider a slightly more general strong form (we add an arbitrary source term f_Ω) so that we can write a general code more easily testable by the method of manufactured solutions; namely,

$$u\frac{dT}{d\eta} = \frac{1}{\mathrm{Pe}}\frac{d^2T}{dy^2} + f_\Omega \quad \text{on } \Omega = (\Gamma_1, \Gamma_2)$$

subject to boundary conditions

$$\left.\frac{dT}{dy}\right|_{\Gamma_1} = g_N, \quad T(x, \Gamma_2) = g_D \tag{22}$$

(23)

$$\begin{aligned}
\int_\Omega u(y)\frac{dT}{d\eta}v\,dy - \frac{1}{\mathrm{Pe}}\int_\Omega \frac{d^2T}{dy^2}v\,dy &= \int_\Omega u(y)\frac{dT}{d\eta}v\,dy - \frac{1}{\mathrm{Pe}}\int_\Omega \left[\frac{d}{dy}\left(v\frac{dT}{dy}\right) - \frac{dv}{dy}\frac{dT}{dy}\right] dy \\
&= \int_\Omega u(y)\frac{dT}{d\eta}v\,dy + \frac{1}{\mathrm{Pe}}\int_\Omega \frac{dv}{dy}\frac{dT}{dy}\,dy - \frac{1}{\mathrm{Pe}}\left[v\frac{dT}{dy}\right]_{\Gamma_1}^{\Gamma_2} \\
&= \int_\Omega u(y)\frac{dT}{d\eta}v\,dy + \frac{1}{\mathrm{Pe}}\int_\Omega \frac{dv}{dy}\frac{dT}{dy}\,dy + \frac{1}{\mathrm{Pe}}v(\Gamma_1) = \int_\Omega f_\Omega v\,dy
\end{aligned}$$

where we have applied integration by parts and the divergence theorem. Note that $v(\Gamma_2) = 0$ by our choice of X; note that $\left.\frac{dT}{dy}\right|_{\Gamma_1} = 1$ is determined by (8). Then we recognize

[1]Note that since we are only semi-discretizing, we can write $\frac{d^2T}{dy^2}$ instead of $\frac{\partial^2 T}{\partial y^2}$. We will complete the discretization of the pseudo-time term using finite differences.

$$\underbrace{\int_\Omega u(y)\frac{dT}{d\eta}v\,dy}_{m\left(\frac{dT}{d\eta},v\right)}+\underbrace{\frac{1}{\mathrm{Pe}}\int_\Omega \frac{dv}{dy}\frac{dT}{dy}\,dy}_{a(T,v)}=\underbrace{\int_\Omega f_\Omega v\,dy-\frac{1}{\mathrm{Pe}}v(\Gamma_1)}_{f(v)} \tag{24}$$

Which completes our derivation of the weak form.

B Quadrature and implementation of $m(\cdot,\,\cdot)$ operator

We briefly remark on the implementation of the $m(\cdot,\,\cdot)$ terms. For the implicit $m(u\,w_h^{k+1})$, we compute the integral with quadrature,

$$\frac{1}{\Delta x}\int_{K_m} u\phi_j\phi_i\,dx=\frac{1}{\Delta x}\int_{\hat{K}} u\widehat{\phi}_i\cdot\widehat{\phi}_j\,|\det J|\;d\widehat{x}\approx\frac{1}{\Delta x}\sum_q w_q\;\det J\big|_{x_q}\,u_q(x_q)\cdot\widehat{\phi}_i(\xi_q)\cdot\widehat{\phi}_j(\xi_q) \tag{25}$$

For the explicit term $u\,w_h^k$ is known at time $k+1$, and it can be considered part of the body forcing; that is

$$\frac{1}{\Delta x}m(u\,w_h^k)=\int_\Omega\underbrace{\frac{u\,w_h^k}{\Delta x}}_{f_{\Delta x}}v\,dx \tag{26}$$

As a final note, we consider the implementation of $f_{\Delta x}$. In order to compute this "forcing" term with quadrature, we need the values of w_h^k at the physical-space quadrature points, whereas they are only computed at the nodal values. However, we can do so efficiently, because $w_h^k(x)=\alpha_i\phi_i(x)$, hence for physical space quadrature point x_q and corresponding reference element quadrature point ξ_q,

$$w_h^k(x_q)=\alpha_i\phi_i(x_q)=\alpha_i\phi_i(G^{-1}(x_q))=\alpha_i\phi_i(\xi_q) \tag{27}$$

where G, G^{-1} denote the isoparametric mapping from reference element to physical space and the inverse, respectively. Since the values of the shape functions ϕ_i are computed on the reference element during the problem setup, we can cheaply compute $w_h^k(x_q)$ without needing to explicitly compute the inverse transformation G^{-1}; caluculate ${S_q}^T\alpha$, where

$$S_q=\begin{bmatrix}\phi_1(\xi_{q_1}), & \dots & \phi_1(\xi_{q_n})\\ \vdots & & \vdots\\ \phi_{n_b}(\xi_{q_1}), & \dots & \phi_{n_b}(\xi_{q_n})\end{bmatrix},\qquad \alpha=\begin{bmatrix}\alpha_1,\,\dots\,\alpha_{n_b}\end{bmatrix} \tag{28}$$

References

[1] Theodore L Bergman, Frank P Incropera, David P DeWitt, and Adrienne S Lavine. *Fundamentals of Heat and Mass Transfer*. John Wiley & Sons, 2011.

[2] Nico Schloemer. `quadpy`: numerical integration in python, 2018.