Neural-Network Closures of Entropy-Based Moment Models of Linear Kinetic Equations

Research Summary

William Porteous

Advisors & Senior Authors: Cory Hauck, Paul Laiu

Oak Ridge National Laboratory: CSM Department

February 2, 2021

To The Reader These notes motivate and summarize my research at ORNL-CSM on a novel entropy-based closure technique for moment-models of kinetic equations. Cory Hauck and Paul Laiu were my research advisors and primary authors of our developing paper, who generously allowed me to investigate this idea with their consistent help and guidance. Here I present an outline of our method, including mathematical construction of our technique, and some key numerical results. Since our preprint for publication is in development (we are writing now), I intend not to preempt that by including highly-detailed numerical findings here.

Abstract We present a new closure technique to implement entropy-based ($\mathrm{M_N}$) moment models for kinetic equations which employs artificial neural-networks (ANNs), and we comparatively examine the numerical performance of our method in the context of a linear kinetic equation with homogeneous background interaction for systems with low moment number ($N \leq 2$). Our method is developed with the intention to supplant a particular step in a kinetic solution scheme for such moment models, where one updates the numerical flux of the solution state via a convex optimization problem; our technique replaces this optimization problem with a convex-approximation problem, where we develop network training schemes to ensure both local accuracy in various metrics, while retaining global properties of the original map, such as convexity and symmetry relations. This approximation technique leads to empirical improvements in total computation time for numerical test-cases of at least 95% over the standard optimization technique.

Contents

1 Introduction

1.1 The Moment-Model Framework: A Velocity Discretization

Consider a generic kinetic equation for a system of charge neutral particles which interact with each other through collision, and perhaps with a background medium. The kinetic equation of this system evolves a kinetic density (or distribution) function $f\colon [0,\infty)\times X\times V\longrightarrow[0,\infty)$ according to:

$$\partial_t f(t,x,v)+v\cdot\nabla_x f(t,x,v)=\mathcal{C}(f(t,x,\cdot))(v) \tag{1.1.1}$$

where $\mathcal{C}$, the "collision operator", introduces interaction effects. This is well-posed only when accompanied by specific initial and boundary conditions.

We intend to discretize (1.1.1) in the velocity variable by examining the evolution of the velocity moments of the underlying kinetic distribution f implied by (1.1.1). To that end, let $\{\varphi_n\}_{n\in\mathbb{N}}$ be an orthonormal basis on $L^2(V)$, and make the following definitions:

$$\mathbf{m}_N(v)=(\varphi_0(v),\varphi_1(v),...,\varphi_N(v))$$

and

$$\mathbf{u}\colon [0,\infty)\times X\longrightarrow\mathbb{R}^{N+1}$$

$$\mathbf{u}^i(t,x):=\int_V f(t,v,x)\mathbf{m}_N^i(v)dv \text{ for } 0\ \leq i\ \leq N$$

In short, $\mathbf{u}$ is a vector of the first $N+1$ moments of f according to the first $N+1$ moment basis functions. To obtain an equation for the evolution of these moments according to (1.1.1), we multiply across the kinetic equation by $\mathbf{m}$ and integrating in v to obtain:

$$\partial_t\langle\mathbf{m}f\rangle+\partial_x\langle v\mathbf{m}f\rangle+\sigma_t=\sigma_s\tilde{Q}\mathbf{u} \tag{1.1.2}$$

where $\tilde{Q}$ is the unique matrix for $\mathbf{m}$ for which $\tilde{Q}\langle \mathbf{m}g\rangle = \frac{\langle \mathbf{m}g\rangle}{2}$. We can express (1.1.2) as

$$\partial_t \langle \mathbf{u}_N\rangle + \partial_x \langle \mathbf{u}_{N+1}\rangle + \sigma_t = \sigma_s \tilde{Q}\mathbf{u_N} \tag{1.1.3}$$

Note that this is not "closed" in the first N moments of f, since the flux term is proportional, in the monomial basis, to a higher moment of F. Hence, we approximate f with some ansatz $\mathbf{F}_{\mathbf{u}(v,x,t)}$, which depends on these moments in some fashion. In a traditional linear Galerkin method, would be a linear combination of the moments $\mathbf{m}$ and yields velocity moments $\mathbf{u}$. In any case, this gives us an equation:

$$\partial_t \langle \mathbf{m}F_{\mathbf{u}}\rangle + \nabla_x \langle v \cdot \mathbf{m}F_{\mathbf{u}}\rangle = \langle \mathbf{m}\mathcal{C}(F_{\mathbf{u}})\rangle \tag{1.1.4}$$

with the demand that $F_{\mathbf{u}(t,x)}$ satisfies the constraint

$$\int_V F_{\mathbf{u}(t,x)}(t,x)\mathbf{m}(v)dv = \mathbf{u}(t,x)$$

However, we want that our ansatz F satisfies physical properties of our original kinetic equation (1.1.1). For this reason, we require that F in fact satisfies an entropy-minimization criterion.

1.2 Entropy-Based Closures: The M_N System

One such form for F which preserves physically-desireable properties of (1.1.1) is the entropy-based closure, where we assume F is a nonnegative function with the prescribed moments $\mathbf{u}(x,t)$ which minimizes the total entropy of the velocity distribution. We take a convex, C^2 function, $\eta\colon \mathbb{R}_+ \longrightarrow \mathbb{R}$, called the kinetic entropy density, and make the following definition, writing $\langle f\rangle$ to mean $\int_V f dv$:

Definition (Entropy-Based Closure)**.** *We demand in (1.1.4) that $F_{\mathbf{u}(x,t)}$ is the solution to the following minimization problem:*

$$\begin{gathered} \textit{mininimize } \langle \eta(g)\rangle,\ g \in L^1(V) \\ \textit{subject to } \langle \mathbf{m}g\rangle = \mathbf{u}(x,t) \end{gathered}$$

When we take η to be the Maxwell-Botlzmann entropy, $\eta(r) = r\log(r) - r$, and restrict to the first $N+1$ moments, i.e. $\mathbf{u}(x,t) = \langle f \cdot \mathbf{m}_N\rangle$, this is called the M_N closure framework. The corresponding nonlinear Galerkin discretization of (1.1.4) is called the M_N system.

Definition (M_N System)**.** *In (??), we define $F_{\mathbf{u}_N}$ as the extremum of the following (entropy) minimization*

$$\begin{gathered} \textit{mininimize } \langle \eta(g)\rangle,\ g \in L^1(V) \\ \textit{subject to } \langle \mathbf{m}_N g\rangle = \mathbf{u}_N \end{gathered} \tag{1.2.1}$$

where $\langle\cdot\rangle$ denotes integration with respect to the velocity domain V, and $\eta(r) = r\log(r) - r$ is the Maxwell-Botlzmann Entropy. This gives the M_N System:

$$\partial_t \mathbf{u} + f(\mathbf{u}) + \sigma_t \mathbf{u} = \sigma_s \tilde{Q}\mathbf{u} \tag{1.2.2}$$

where $\mathbf{f}(\mathbf{u}) = \langle v\mathbf{m}F_{\mathbf{u}(x,t)}\rangle$ is the flux, and $F_{\mathbf{u}}$ is the entropy ansatz for $\mathbf{u}$.

2 Entropy-Based Closure: Structure and Numerical Solution Scheme

2.1 Realizability

For the M_N system, the appropriate domain for moment vectors $\mathbf{u}$ must be specified, particularly to establish valid initial and boundary conditions for (1.1.4). To start, we consider those vectors $\mathbf{u} \in \mathbb{R}^{N+1}$ that at least are the moments of some non-negative Lebesgue-integrable function g. Note that this is not the same as the image of the feasible set in (1.2.1), since non-negativity is weaker than the constraints of (1.2.1).

Definition (The Realizable Set)**.** *The set of all possible N+1 moment vectors given V and moment-basis $\mathbf{m}_N$:*

$$\mathcal{R}_{\mathbf{m}} := \{\mathbf{u} \in \mathbb{R}^{N+1} \colon \mathbf{u} = \langle f \mathbf{m}_N \rangle,\ f \in L^1(V), f \geq 0\}$$

For generic $\mathbf{v} \in \mathcal{R}_{\mathbf{m}}$, it may be that the (1.2.1) has no solutions or fails uniqueness. In the case that V is unbounded, an example is simple, since L^1 continuity fails for the objective function. To continue, we assume that V is a bounded convex subset of $\mathbb{R}$.

2.1.1 Characterizing the Realizable Set

When the moments $\mathbf{m}_N$ are monomials, there is a simple characterization of the realizable set $\mathcal{R}_{\mathbf{m}}$.

Theorem. *Let* $m_N = \begin{pmatrix} 1 \\ v \\ \vdots \\ v^N \end{pmatrix}$ *then* $\mathbf{u}$ *is Realizable if and only if*

1. *If N is odd, that the matrices $P^{\pm}$ defined by*

$$P^{\pm}_{ij} := \mathbf{u}_{i+j} \pm \mathbf{u}_{i+j+1}$$

are positive definite.

2. *If N is even, that the matrices P^0 and P^1 defined by*

$$P^0_{ij} := \mathbf{u}_{i+j} \pm \mathbf{u}_{i+j+1}$$

and

$$P^1_{ij} = \mathbf{u}_{i+1} - \mathbf{u}_{i+j+2}$$

are positive definite.

In our case, we take $\mathbf{m}_N$ is the vector of the first $N+1$ Legendre polynomials (including the 0th order constant of 1). Since this basis is simply an invertible linear transformation of the monomial basis, it follows from the above theorem that there is a simple way to check realizability in our case. $\mathbf{m}_N$.

2.1.2 Invariance Under Dynamics

We must ask whether it is true, even when assuming that V is bounded and convex, that for a given initial conditions $\mathbf{u}(0, x)$ (and appropriate boundary conditions), that the solution to (1.2.2) has $\mathbf{u}(x, t) \in \mathcal{R}_{\mathbf{m}_N}$ for all $t > 0$. It turns out that this is an open problem for generic conditions, but it is possible to enforce this numerically for particular solution schemes; this informs our choice of the kinetic solution scheme in the next section.

2.2 Calculating the Entropy-Ansatz: Lagrange Duality

In the case that $V = [-1, 1]$, or any bounded domain for that matter, the objective function of the primal entropy minimization problem is a continuous, convex functional on $L^1(V)$. This problem has a unique minimizer, and we can in fact solve for it explicitly via the Lagrange dual problem. This will enable us to construct a kinetic solution scheme which preserves realizability. The theory of this section is exposited in greater generality in Appendix section 1, which also justifies the calculations and lemmas here.

First, define a map which takes a realizable moment $\mathbf{u}$ to the minimal entropy for functions $f \in L^1(V)$ in the feasible set where $\langle \mathbf{m} f \rangle = \mathbf{u}$ (this is legitimate since the solution is guaranteed to exist).

Definition (Minimal Entropy Map)**.**

$$\begin{aligned} h &\colon \mathcal{R} \longrightarrow \mathbb{R} \\ h(\mathbf{u}) &= \inf \{ \int_V \eta(g) \colon g \in L^1(V) \} \\ &\textit{subject to } g \in L^1(V),\ \langle \mathbf{m} g \rangle = \mathbf{u},\ \textit{and } g > 0 \end{aligned} \tag{2.2.1}$$

where $\eta(s) = s \log s - s$ is the Maxwell-Botlzmann entropy.

Again, the infimum is a minimum when V is bounded, which is the case we restrict to. To solve this minimization problem of the M_N system, we calculate Lagrange dual problem, which is unconstrained; see Appendix Part 1 for the justification and background on this calculation. Here, we will write η_* for the legendre-transform of η.

Lemma (Minimal Entropy Dual Problem)**.** *The dual problem to (1.2.1) is given by:*

$$\begin{aligned} &\textit{minimize } \langle \eta_*(\alpha^t \mathbf{m}) \rangle - \alpha^t \cdot \mathbf{u}\ , \alpha \in \mathbb{R}^{N+1} \\ &\alpha \in \mathbb{R}^{N+1} \end{aligned} \tag{2.2.2}$$

where $h_*(\alpha) = \langle \eta_*(\hat{\alpha}^t(\mathbf{u}) \cdot \mathbf{m}) \rangle$.

We will denote the solution to the dual problem for a given moment $\hat{u}$, as $\hat{\alpha}(\mathbf{u})$.

Definition (Dual Solution $\hat{\alpha}(\mathbf{u})$)**.** *For an admissable moment vector u*

$$\hat{\alpha}(\mathbf{u}) := \textit{argmin } \langle \eta_*(\alpha^t \mathbf{m}) \rangle - \alpha^t \cdot \mathbf{u}\ , \alpha \in \mathbb{R}^{N+1}$$

By necessary conditions of Lagrange duality, the solution $\hat{\alpha}(\mathbf{u})$ to the dual problem obtains a solution to the primal problem (1.2.1). Recall from §1.3 that we denote the solution to the dual as $F_{\mathbf{u}}$. With the knowledge of the following lemmas, we define a new map

$$G_{\mathbf{v}} \colon V \longrightarrow \mathbb{R}_+$$

for any vector $\mathbf{v} \in \mathbb{R}^{N+1}$, which will be identical to $F_{\mathbf{u}}$ when $\mathbf{v} = \hat{\alpha}(\mathbf{u})$.

Lemma (Primal Solution)**.** *When $\hat{\alpha}(\mathbf{u})$ is the solution to the dual problem (2.2.2), necessary conditions for Lagrange duality imply that the solution to the primal problem (1.2.1), denoted $F_{\mathbf{u}}$, is:*

$$F_{\mathbf{u}} = \eta_*'(\hat{\alpha}(\mathbf{u})^t \cdot \mathbf{m}) \tag{2.2.3}$$

which, when η is the Maxwell-Botlzmann entropy is

$$F_{\mathbf{u}}(v) = \exp(\alpha^t \mathbf{m}) \tag{2.2.4}$$

Lemma (First-Order Conditions)**.** *First order conditions for Lagrange duality imply that the gradient of h is equal to the dual optimum*

$$\nabla h(\mathbf{u}) = \hat{\alpha}(\mathbf{u}) \tag{2.2.5}$$

and moreover that $\mathbf{u}$ *is reconstructed from the dual solution* $\hat{\alpha}(\mathbf{u})$ *as follows*

$$\mathbf{u} = \langle \hat{\alpha}(\mathbf{u})^t \cdot \mathbf{m}) \rangle \tag{2.2.6}$$

With these lemmas in mind, we make the following definition:

Definition (G_α)**.** *For any* $\alpha \in \mathbb{R}^{N+1}$ *define*

$$G_\alpha \colon V \longrightarrow \mathbb{R}_+$$

$$G_\alpha(v) = \eta_*'(\alpha \cdot \mathbf{m})$$

and for the case of M_N*:*

$$G_\alpha(v) = \exp(\alpha \cdot \mathbf{m})$$

When $\alpha = \hat{\alpha}(\mathbf{u})$ *we have that*

$$G_{\hat{\alpha}(\mathbf{u})} = F_{\mathbf{u}}$$

The key is to note that the vectors $\mathbf{v} \in \mathbb{R}^{N+1}$, like $\hat{\alpha}$, are Lagrange multipliers, and for each $\mathbf{v}$ we can calculate a moment vector, $\hat{u} = \langle v^t \cdot \mathbf{m} \rangle$ and the distribution which solves the associated entropy-minimization problem, $G_{\mathbf{v}}$. From the above first-order conditions, it immediately follows that $\hat{\alpha}$ is therefore an invertible function, and we let its inverse be denoted by $\hat{\mathbf{u}}(\alpha)$ for admissible vectors $\alpha \in \mathbb{R}^{N+1}$.

2.3 Calculating Entropy-Ansatz: An Explicit Example

While in the previous section we showed that Lagrange duality gives us an explicit means by which to calculate the minimum entropy-ansatz, here we demonstrate that in an example which gives solutions expressible in elementary functions. We assume, however, that we are in possession of the solution to the dual problem (2.2.2). We restrict to M_N for $N = 1$ and consider

$$V = [-1, 1], \ \mathbf{m} = \begin{pmatrix} 1 \\ v \end{pmatrix}$$

Note that the functions in $\mathbf{m}$ are orthogonal but not normalized in L^2.

Fact. *For* $N = 1$ *and* $V = [-1, 1]$*, the minimum entropy map (defined in (2.2.1)) is given by*

$$h(\mathbf{u}) = 2 \exp \alpha_0 [(\alpha_0 - 2) \frac{\sinh(\alpha_1)}{\alpha_1} + \cosh(\alpha_1)] \tag{2.3.1}$$

where $\alpha \equiv \hat{\alpha}(\mathbf{u})$

Proof. We recover the primal solution, G_α, by duality conditions on the dual solution $\hat{\alpha}(\mathbf{u})$, to the dual problem

$$\text{minimize } \langle \exp(\alpha^t \mathbf{m}) \rangle - \alpha^t \cdot \mathbf{u}, \ \alpha \in \mathbb{R}^2$$

Write $\alpha \equiv \hat{\alpha}(\mathbf{u})$ from here on, in this proof, for brevity. By virtue of first order duality conditions (using calculus of variations), the explicit primal solution is, in this $N = 1$ case:

$$G_\alpha(v) = \exp(\alpha_0 + \alpha_1 v)$$

$$h(\mathbf{u}) = \int_{-1}^{1} G_\alpha(v)\log(G_\alpha(v)) - G_\alpha(v)dv = \int_{-1}^{1} \exp(\alpha_0 + v\alpha_1)\cdot(\alpha_0 + \alpha_1 \cdot v - 1)dv$$

To compute h from α, let $w = \alpha_0 + v\alpha_1$, $dw = \alpha_1 dv$, so that for $\alpha_1 \neq 0$:

$$\int_{-1}^{1} \exp(\alpha_0 + v\alpha_1)\cdot(\alpha_0 + \alpha_0 v - 1) = \int_{\alpha_0-\alpha_1}^{\alpha_0+\alpha_1} \exp w(w-1)\frac{dw}{\alpha_1}$$

Since an antiderivative for $F'(w) = \exp w(w-1)$ is $F(w) = \exp(w)(w-2)$ we have:

$$h(\mathbf{u}) = \frac{1}{\alpha_1}\exp w(w-2)|_{w_{-1}}^{w_1} = \frac{1}{\alpha_1}[\exp(\alpha_0+\alpha_1)(\alpha_0+\alpha_1-2) - \exp(\alpha_0-\alpha_1)(\alpha_0-\alpha_1-2)] =$$

$$\frac{1}{\alpha_1}[(\alpha_0-2)(\exp(\alpha_0+\alpha_1) - \exp(\alpha_0-\alpha_1)) + \alpha_1(\exp(\alpha_0+\alpha_1) + \exp(\alpha_0-\alpha_1))]$$

Finally as $2\sinh(s) = e^s - e^{-s}$ and $2\cosh(s) = e^s + e^{-s}$, simplify to obtain:

$$h(\mathbf{u}) = 2\exp\alpha_0[(\alpha_0-2)\frac{\sinh(\alpha_1)}{\alpha_1} + \cosh(\alpha_1)]$$

Notice that this formula continuously extends to the $\alpha_1 = 0$ case by taking the $\alpha_1 \to 0$ limit to obtain:

$$h(\mathbf{u}) = 2 * \exp\alpha_0(\alpha_0 - 1)$$

□

The key takeaway from this section is that the forward-evaluation of the entropy function $h(\mathbf{u})$ can be expressed as elementary functions for $N = 1$ and $\mathbf{m}$ as the Legendre polynomials, and moreover, that it is simple to calculate $h(\mathbf{u})$ given that one is in possession of $\hat{\alpha}(\mathbf{u})$. In fact, calculating this dual optimum, $\hat{\alpha}$, becomes the central challenge of our solution scheme.

2.4 Choice of Numerical Scheme: Realizability Preserving

In order to develop a numerical scheme for solving (**??**) which preserves realizability, we exploit the fact that given any vector α, the map G_α yields the entropy ansatz for the realizable vector $\mathbf{u} = \langle \alpha^t \cdot \mathbf{m}\rangle$. In short, we employ a kinetic scheme. The key challenge, however, is the same as in sections §1.5-1.7: calculating $F_{\mathbf{u}} = G_{\hat{\alpha}(\mathbf{u})}$ requires us to obtain $\hat{\alpha}(\mathbf{u})$ by solving the dual problem (2.2.2). To this end, for we define

Definition ($\tilde{\alpha}$)**.** *We define $\tilde{\alpha}$ to be some approximation of $\hat{\alpha}(\mathbf{u})$ by a process which is to be determined in context. For reference, in the standard numerical implementation of the M_N system, this is merely the numerical solution to some computational implementation of the dual optimization problem (2.2.2). In short:*

$$\tilde{\alpha} \simeq \hat{\alpha}(\mathbf{u})$$

Here, we illustrate our scheme in a linear spatial domain $X = (-a, b)$ where $a, b \in [-\infty, \infty]$, and again, $V = [-1, 1]$. We start with a numerical initial condition $\mathbf{u}^0$, and describe how we calculate our numerical solution.

2.4.1 Generic Scheme for Time-Integration

To evolve forward in time from a numerical initial condition $\mathbf{u}^0$, we employ a second-order Runge-Kutta method (SSP-RK2), known as the improved Euler method, with a uniform mesh in time and space where t_n is the n-th time step.This scheme is second order in both space and time. The vector containing the numerical solution at time t_n, denoted $\mathbf{u}^n$, is updated to $\mathbf{u}^{n+1}$ at time t_{n+1}, by the following procedure, where L is a nonlinear operator, defined subsequently:

$$\mathbf{u}_{s0} := \mathbf{u}^n,\ \mathbf{u}_{s1} := \Delta t \cdot L(\mathbf{u}_{s0}),\ \mathbf{u}_{s2} := \Delta t \cdot L(\mathbf{u}_{s1})$$
$$\mathbf{u}^{n+1} := \frac{1}{2}(\mathbf{u}_{s1} + \mathbf{u}_{s2})$$

2.4.2 Construction of L

To construct the nonlinear operator L, first choose the mesh. Define constants Δx and Δt for a uniform mesh where $\Delta x = \frac{x_R - x_L}{N_x}$, where N_x represents the total number of physical points within the spatial domain, and x_R and x_L are the right and left-most point within the physical domain. This gives mesh points $\{x_j\}_{-j=1}^{N_x+2}$ and continugous spatial cells I_j as

$$x_j = x_L + \frac{2j-1}{2}\Delta x$$
$$t^i := i \cdot \Delta t$$
$$I_j = (x_{j-1/2}, x + j + 1/2)$$

For $j \in \{-1, 0, N_x + 1, N_x + 2\}$ the cells I_j are referred to as Ghost cells, and are used to hold information of the boundary conditions. Then, for each spatial cell I_j and each time t, we construct a numerical approximation $u_j(t)$, which approximates the average of the true solution $\mathbf{u}(x,t)$ at the time t. That is, we intend our approximation to be:

$$\frac{1}{\Delta x}\int_{I_j} \mathbf{u}(x,t)dx \simeq u_j(t)$$

Then, we construct the nonlinear operator $\tilde{G}_k$. For integer values of k,

Definition $(\tilde{G}_k)$**.** *For $k \in \{-1, 0, ..., N_x + 1, N_x + 2\}$,*

$$\tilde{G}_k := G_{\tilde{\alpha}(\mathbf{u}_k)} \tag{2.4.1}$$

This is the approximate entropy ansatz with prescribed moments $\mathbf{u}_k(t)$ up to the approximation $\tilde{\alpha}$.

From this, we extend $\tilde{G}_k(v)$ to $k = j \pm 1/2$, $j \in \{-1, 0, ..., N_x + 1, N_x + 2\}$, where the definition is conditional on the sign of v:

$$\tilde{G}_{j+1/2}(v,t) := \begin{cases} \tilde{G}_j(v,t) + \frac{\Delta x}{2} \cdot \hat{s}_j(v,t) & v > 0 \\ \tilde{G}_{j+1}(v,t) - \frac{\Delta x}{2}\hat{s}_{j+1}(v,t) & v < 0 \end{cases}$$

where $\hat{s}_k$ approximates the spatial derivative of $\tilde{G}_k$ via:

$$\hat{s}_j = \text{minmod}_{1\le\theta\le 2}\{\theta\frac{G_j - G_{j+1}}{\Delta x}, \frac{G_{j+1} - G_j}{2\Delta x}, \theta\frac{G_{j+1} - G_j}{\Delta x}\}$$

The "minmod" returns the number with minimal absolute value from the convex hull of its arguments.

Finally, we define the numerical flux as

$$\mathbf{f}_{j+1/2} := \langle v\mathbf{m}(v)\tilde{G}_{j+1/2}\rangle, \text{ for } 0 \leq j \leq N_x$$

The vector $\mathbf{f}_{j\pm1/2}$ are the numerical flux vectors at the edges of the cell I_j, which depends on $\mathbf{u}_j$, $\mathbf{u}_{j-1}$, and $\mathbf{u}_{j+1}$ in a complex manner through $\tilde{G}_{j+1/2}$.

Now, assumging some boundary condition $u_j(0)$ for $j \in \{-1, ..., N_x + 2\}$, we update $u_j(t)$ on the interior of the domain (i.e. $j \in \{1, ..., N_x\}$) according to the operator L, (which is nonlinear in the spatial cells), according to the following semidiscrete scheme:

Definition (Characterization of L)**.**

$$\partial_t \mathbf{u}_j(t) = \langle \hat{e}_j, L(\mathbf{u}(t))\rangle := -(\frac{\mathbf{f}_{j+1/2} - \mathbf{f}_{j-1/2}}{\Delta x} + \sigma_t \mathbf{u}_j(t)) + \sigma_s \tilde{\mathbf{Q}} \cdot \mathbf{u}_j(t), \text{ for } 1 \leq j \leq N_x$$

2.4.3 Initial and Boundary Conditions

The spacial cells which contain boundary data, referred to as "ghost cells", are indexed with $j \in \{-1, 0, N_x + 1, N_x + 2\}$. Generic boundary conditions for M_N is still an open problem, but they can be implemented numerically in our scheme via the ghost cells. In the case that a boundary condition is specified for the underlying kinetic equation (1.1.1) at the edges of the domain x_L and x_R, we have that $f(x_L, v, t)$ is defined for incoming data, i.e. for $v > 0$ and $f(x_R, v, t)$ for $v < 0$. If we extend this info to outgoing data, $v < 0$ for $f(x_L, v, t)$ and $v > 0$ for $f(x_R, v, t)$, then we may require that our ghost cells are the moments associated to this boundary data:

$$\mathbf{u}_{-1} = u_0(t) = \langle \mathbf{m} f(x_L, \cdot, t)\rangle$$

and

$$\mathbf{u}_{N_x+1}(t) = u_{N_x+2}(t) = \langle \mathbf{m} f(x_R, \cdot, t)\rangle$$

In the case that the boundary condition is periodic, we can simply identify all 4 ghost cells to some common boundary value.

2.4.4 Preserving Realizability

For this kinetic scheme to preserve realizability of $\mathbf{u}_j^i \ \forall\, j \in \{1, ..., N_x\}$ and all $t_n \geq t_i > 0$, it suffices to impose an upper-bound on the time step value Δt, determined by the ratio $\tilde{G}_{\mathbf{u}_j^n}$ and $G_{\mathbf{u}_j^n}$, as well as Δx and θ. Letting

$$\gamma = \sup_{v,j,i}\{ \frac{\tilde{G}_{\mathbf{u}_j(t_i)}(v)}{G_{\mathbf{u}_j(t_i)}(v)} : j \in \{1, ..., N_x\}, j \leq n, v \in V\}$$

then we have the following fact:

Theorem. *If $u_j^i \in \mathcal{R}_{\mathbf{m}}$ for $j \in \{-1, 0, ..., N_x + 1, N_x + 2\}$ and Δt is such that:*

$$\Delta t \cdot (\frac{\gamma}{\Delta x}\frac{2+\theta}{2} + \sigma_t) < 1$$

then u_j^{i+1} are realizable for all $j \in \{1, ..., N_x\}$.

3 Numerical Challenge and Proposed Method

3.1 The Costs of Optimization

Note that in the construction of our numerical scheme in §2.4, to pass from $\mathbf{u}^n$ to $\mathbf{u}^{n+1}$ we require the evaluation of $\tilde{G}_{\mathbf{u}_k^n}$ for $k \in \{j, j+1, j-1\}$, for all $j \in \{0, ..., N_x+1\}$. This requires the calculation of $\tilde{\alpha}(\mathbf{u}_j^n)$, which we recall, is some approximation to the solution of the Lagrange dual problem to the entropy-ansatz. In short, it requires us for each j and n to solve a finite dimensional convex optimization on $\mathbb{R}^{N+1}$ which is distinct for each j and n, and return our approximation for the minimum, $\tilde{\alpha}(\mathbf{u}_j^n)$. Let us define the numerically implemented problem precisely, and then introduce our novel method.
Recall the dual problem to entropy minimization:

Definition (Minimal Entropy Dual Problem)**.** *The dual problem to* (1.2.1) *is given by:*

$$\begin{aligned} &\textit{minimize } \langle \eta_*(\alpha^t \mathbf{m}) \rangle - \alpha^t \cdot \mathbf{u} \ , \alpha \in \mathbb{R}^{N+1} \\ &\alpha \in \mathbb{R}^{N+1} \end{aligned} \tag{3.1.1}$$

where $h_*(\alpha) = \langle \eta_*(\hat{\alpha}^t(\mathbf{u}) \cdot \mathbf{m}) \rangle$.

As N grows, this optimization problem grows to become prohibitively expensive in terms of computation time, and moreover, the hessian of this objective function becomes arbitrarily ill-conditioned as $\mathbf{u}_j^n$ approaches the realizable boundary. These challenges are compounded by the fact that **all** the integrals in the preceding discussion, at least for $N > 1$, must be approximated by quadrature. This implied additional difficulties for the hessian of the dual objective function due to numerical precision of arithmetic means over large numbers of quadrature points, making it impossible to solve for the correct Newton direction during optimization in certain cases.

Let us denote the quadrature-implemented optimization problem as follows

Definition (Numerical Dual Problem)**.** *The numerical dual problem is given by:*

$$\begin{aligned} &\textit{minimize } \sum_{\mu_i \in \mathcal{Q}} w_i \eta_*'(\alpha^t \mathbf{m}(\mu_i)) - \alpha^t \cdot \mathbf{u} \ , \alpha \in \mathbb{R}^{N+1} \\ &\alpha \in \mathbb{R}^{N+1} \end{aligned} \tag{3.1.2}$$

where $\mathcal{Q}$ *is an integral quadrature with points* μ_i *and weights* w_i*. As always,* $\eta_*' = \exp$*. Morevover, we denote the approximate solution to the quadrature-implemented numerical dual problem as:*

$$\tilde{\alpha}_{\mathcal{Q}}(\mathbf{u}) := \textit{argmin}_{\alpha \in \mathbb{R}^{N+1}} \sum_{\mu_i \in \mathcal{Q}} w_i \eta_*'(\alpha^t \mathbf{m}(\mu_i)) - \alpha^t \cdot \mathbf{u} \tag{3.1.3}$$

This is dual to the quadrature-implemented primal problem:

Definition (Numerical Primal Problem)**.**

$$\begin{aligned} &\textit{mininimize } \sum_i \eta(g(\mu_i)) w_i, \ g \in L^1(V) \\ &\textit{subject to } \sum_i \mathbf{m}_N g(\mu_i) w_i = \mathbf{u}_N \end{aligned} \tag{3.1.4}$$

where again $\mathcal{Q}$ *is the quadrature and* $\eta = r\log(r) - r$*. Moreover, we define the minimal entropy-map:*

$$\begin{aligned} &h_{\mathcal{Q}}(\mathbf{u}) := \min \sum_i \eta(g(\mu_i)) w_i, \ g \in L^1(V), \\ &s.t. \sum_i \mathbf{m}_N g(\mu_i) w_i = \mathbf{u} \end{aligned} \tag{3.1.5}$$

Recall from (2.2.5) that in the case of the exact problem, $\nabla h(\mathbf{u}) = \hat{\alpha}(\mathbf{u})$ for all $\mathbf{u} \in \mathcal{R}$. This is the same case with the numerically implemented problem:

Lemma (First Order)**.** *For the numerical minimal entropy map $h_{\mathcal{Q}} \colon \mathcal{R} \longrightarrow \mathbf{R}$, we have that*

$$\nabla h_{\mathcal{Q}}(\mathbf{u}) = \hat{\alpha}_{\mathcal{Q}}(\mathbf{u}) \tag{3.1.6}$$

where $\hat{\alpha}$ is the true solution to the numerically implemented dual problem (as opposed to the approximate solution $\tilde{\alpha}_{\mathcal{Q}}(\mathbf{u})$.

3.2 Our Method: Local Approximation Over Optimization

Since the optimization of (3.1.2) is performed to obtain $\tilde{\alpha}_{\mathcal{Q}}(\mathbf{u})$ as an approximation to $\hat{\alpha}_{\mathcal{Q}}(\mathbf{u})$, it would be possible to supplant this optimization with the gradient calculation in (3.1.6) for each moment $\mathbf{u}$, and thereby obtain $\tilde{G}$ for the entire numerical solution scheme. This would require, in lieu of optimizing the numerical dual to the minimal entropy map at each point, an approximation to the minimal entropy map over the realizable domain.

Definition (Approximation Method, Approximate Multiplier, Entropy Approximation)**.** *We define an approximate minimal entropy map, called the* ***entropy approximation****, $\tilde{h}$ such that*

$$\begin{aligned} &\tilde{h} \colon \mathcal{R} \longrightarrow \mathbb{R} \\ &\tilde{h}(\mathbf{u}) \simeq h_{\mathcal{Q}} \end{aligned} \tag{3.2.1}$$

from which we calculate the ***approximate multiplier (alpha)*** *function* ***approximate alpha***

$$\tilde{\alpha}(\mathbf{u}) := \nabla \tilde{h}(\mathbf{u}) \tag{3.2.2}$$

Inserting the above into our numerical scheme is referred to as the ***approximation method****.*

The key requirements are that $\tilde{h}$ be which is convex and C^2, but which also admits a closed-form, forward-evaluation expression for its gradient $\tilde{\alpha}$. In our technique,

1. We approximate the entropy function $h \colon \mathcal{R} \longrightarrow \mathbb{R}$ with $\tilde{h}$ using
   (a) A fully-connected artificial neural-network
   (b) Rational-Cubic Spline
2. Compute $\nabla \tilde{h}(\mathbf{u})$ and define $\tilde{\alpha} := \nabla \tilde{h}(\mathbf{u})$

This first requires us to construct such an approximation $\tilde{h}$ on the set $\mathcal{R}$, which has dimension $N + 1$ for the M_N system. We were able to deduce scaling relations which reduced the underlying approximation problem to an N dimensional bounded, open subset of $\mathcal{R}$.

3.3 Construction of $\tilde{h}$

3.3.1 Metric for Approximation Accuracy

The term which we desire to compute more efficiently in (**??**) is the so-called "flux-update", i.e.

$$F \colon \mathcal{R} \longrightarrow \mathbb{R}^2$$

$$F(\mathbf{u}) := \langle \mu \mathbf{m}_N G_{\hat{\alpha}(\mathbf{u})} \rangle$$

Fact. *It follows from calculus that*

$$\|F(\mathbf{u}) - F(\mathbf{v})\| \leq C\|\mathbf{u} - \mathbf{v}\|$$

For an approximate closure $\tilde{h}, \tilde{\alpha}$, u and v are replaced in this estimate $\hat{u}(\tilde{\alpha}(u))$ and by $\hat{u}(\tilde{\alpha}(v))$ - the moment-reconstruction induced by our approximation to α. Since the relationship between $\tilde{\alpha}$ and $\hat{u}(\tilde{\alpha})$ is defined by the integral of an exponential, we consider the $\hat{u}(\tilde{\alpha}(u))$ to be the best metric of our accuracy.

For accuracy, we intend to obtain an approximation which minimizes the value

$$\|\hat{u}(\tilde{\alpha}(u)) - u\|_{L^2(\mathcal{R})}$$

3.3.2 Dimension-Reduction: From Unbounded to Bounded Regions

3.3.3 Scaling Relations

From the definitions in Section 1, it is straightforward to show that

Fact (Alpha Scaling)**.**

$$\forall \lambda > 0$$
$$\hat{\alpha}(\lambda \mathbf{u}) = \alpha(\mathbf{u}) + \begin{pmatrix} \ln(\lambda) \\ 0 \end{pmatrix} \tag{3.3.1}$$

from which it follows that

Fact (Entropy Scaling)**.**

$$\forall\ \lambda > 0$$
$$h(\lambda \mathbf{u}) = \lambda(h(\mathbf{u}) + u_0 \ln(\lambda)) \tag{3.3.2}$$

For the case of the M_1 equation, where the moment vectors are simply $\mathbf{u} = (u_0, u_1)$, this gives the following:

Fact (u_0 Scaling)**.** *For $v \in \mathcal{R}$, choose $\lambda = v_0$ and $\mathbf{v} = v_0 \begin{pmatrix} 1 \\ \frac{v_1}{v_0} \end{pmatrix}$*

$$h(\mathbf{v}) = v_0(h((\frac{\mathbf{v}}{v_0})) + ln(v_0)) \tag{3.3.3}$$

To approximate h, we can approximate the behavior of h on the set $\mathcal{D} = \{v \in \mathcal{R} \colon v_0 \equiv 1\}$, and extend the network to $\mathcal{R}$ using this scaling relation. That is

Definition. *We write $\tilde{h}_s$ for a function*

$$\tilde{h}_s \colon \mathcal{D} \longrightarrow \mathbb{R}$$

which aproximates the entropy function h on the set $\mathcal{D}$

$$\tilde{h} \colon \mathcal{R} \longrightarrow \mathbb{R}$$

$$\tilde{h}(\mathbf{u}) := u_0 \cdot \tilde{h}_s((1, u_1/u_0)) + u_0 \cdot \ln(u_0)$$

3.3.4 Guaranteeing Convexity

It is not immediately obvious (to me) that an approximation $\tilde{h}_s$ which is convex will induce a convex function $\tilde{h}$. This, however, turns out to be the case; we prove this by checking the positive-definiteness of the hessian matrix for $\tilde{h}$.

4 Convexity of Scaled-Approximations

In both methods here, a neural network approximation to the entropy function h at the domain $\mathcal{D}$, denoted $\mathcal{N}$ is developed first. There multiple ways to extend this to an approximation $\tilde{h}$ on the full domain $\mathcal{R}$.

4.1 Function-Based Approximation: Method A

By using scaling relation (3.3.2) in the specific case of (3.3.3), we can reduce the dimension of the approximation problem from the full set $\mathcal{R}$ to the domain $\mathcal{D}$.

Consider a neural network $\mathcal{N}$ approximating h on the domain

$$\mathcal{D} = \{\mathbf{u} \in \mathcal{R} \colon u_0 = 1\}$$

$$\mathcal{N} \colon \mathcal{D} \longrightarrow \mathbb{R}$$

We make the definition

$$\tilde{h}_s := \mathcal{D} \longrightarrow \mathbb{R}$$
$$\tilde{h}(\mathbf{u}_N) := \mathcal{N}(\mathbf{u}_N)$$

and extend this to the full domain $\mathcal{R}$ by the definition

$$\tilde{h} \colon \mathcal{R} \longrightarrow \mathbb{R}$$
$$\tilde{h}(\mathbf{u}) = u_0 \cdot (\tilde{h}_s(\frac{\mathbf{u}}{u_0}) + \log u_0)$$

In this case, we can calculate $\tilde{\alpha}$ from our approximation $\tilde{h}$:

$$\tilde{\alpha}(\mathbf{u}) := \nabla \tilde{h}(\mathbf{u})$$

For $\tilde{\alpha}$ we have the following identification:

$$\tilde{\alpha}^i(\mathbf{u}) = \begin{cases} \tilde{h}_s(\frac{\mathbf{u}}{u_0}) - \frac{\mathbf{u}_N \cdot \nabla_N \tilde{h}_s(\frac{\mathbf{u}}{u_0})}{u_0} + \log u_0 + 1 & i = 0 \\ \partial_i \tilde{h}(\frac{\mathbf{u}}{u_0}) & i > 0 \end{cases}$$

4.2 Gradient-Based Approximation: Method B

Recall the following facts from the Lagrange-duality of the entropy-minimziation framework:

$$h(\mathbf{u}) = \int_{-1}^{1} (\hat{\alpha}^t(\mathbf{u})\mathbf{m} - 1) \exp(\hat{\alpha}^t(\mathbf{u})\mathbf{m})$$

$$\nabla h(\mathbf{u}) = \hat{\alpha}(\mathbf{u})$$

$$\hat{u}_0(\alpha) = 1 \iff \alpha_0 = -\log \int_{-1}^{1} \exp(\alpha_N^t \mathbf{m_N})$$

Examining (3.3.1), it is clear that an approximation $\tilde{\alpha}_s$ to the map $\hat{\alpha}$ on the set $\mathcal{D}$, can be extended to $\mathcal{R}$ via the log scaling relationship in (3.3.1). To obtain such an initial approximation $\tilde{\alpha}_s$, we use the gradient of a neural network $\mathcal{N}$ approximating the function h at the realizable set.

First, assume that we have a neural network $\mathcal{N}$ approximating h at the set $\mathcal{D}$ to define $\tilde{\alpha}_s(\mathbf{u})$.

Definition. *Given a convex neural network $\mathcal{N}$ approximating h at the set $\mathcal{D}$, we make the definition*

$$\tilde{\alpha}_s \colon \mathcal{D} \longrightarrow \mathbb{R}^{N+1}$$

$$\tilde{\alpha}_s^i(\mathbf{u}) = \begin{cases} -\log \int_{-1}^{1} \exp(\nabla \mathcal{N} \cdot \mathbf{m}_N) & i = 0 \\ \nabla^i \mathcal{N}(\mathbf{u}) & i > 0 \end{cases} \tag{4.2.1}$$

Extend $\tilde{\alpha}_s$ to the whole domain $\mathcal{R}$ according to (3.3.1) via the definition:

Definition. *($\tilde{\alpha}$) Define $\tilde{\alpha} \colon \mathcal{R} \longrightarrow \mathbb{R}^{N+1}$ via*

$$\tilde{\alpha} = \tilde{\alpha}_s(\frac{\mathbf{u}_N}{u_0}) + \begin{pmatrix} \log u_0 \\ \mathbf{0}_N \end{pmatrix}$$

Observe that this definition has the advantage that the moment-reconstruction map always reproduces the u_0 component exactly:

$$\tilde{\alpha}(\mathbf{u}) \to \hat{\mathbf{u}}(\tilde{\alpha}(\mathbf{u})) = \begin{pmatrix} u_0 \\ \hat{\mathbf{u}}_N(\tilde{\alpha}(\mathbf{u})) \end{pmatrix}$$

For our kinetic solution scheme, we seek to ensure that this definition of $\tilde{\alpha}$ be identifiable with the gradient of some convex function. For this, no global approximation to the entropy function $\tilde{h}$ is needed, but one could additionally use this to define an approximation to the entropy function $\tilde{h} \colon \mathcal{R} \longrightarrow \mathbb{R}$ as

$$\tilde{h}(\mathbf{u}) = \int_{-1}^{1} (\tilde{\alpha}^T \mathbf{m}(v) - 1) \exp(\tilde{\alpha}^T(\mathbf{u}) \mathbf{m}(v)) dv$$

4.3 Convexity of Method A

We want to know the requirements on $\tilde{h}_s$ to ensure that $\tilde{h}$ is a convex function, in the framework where:

$$\tilde{h}(\mathbf{u}) = u_0(\ln(u_0) + \tilde{h}_s(\frac{\mathbf{u}_N}{u_0}))$$

We make the definition

$$\xi(\mathbf{u}) := u_0 \cdot \tilde{h}_s(\frac{\mathbf{u}_N}{u_0})$$

and note that, $\tilde{h}(\mathbf{u}) = u_0 \ln(u_0) + \xi(\mathbf{u})$. Since $u_0 \ln(u_0)$ is convex, it would suffice to require that ξ be convex in $\mathcal{R}$ (since the sum of convex functions is again convex). In turn, to ensure convexity of ξ on $\mathcal{R}$, it suffices that $\tilde{h}_s$ is convex and C^2 on its domain $\mathcal{D}$.

Fact. *If $\tilde{h}_s$ is convex on $\mathcal{D}$, then the map $\xi\colon \mathcal{R} \longrightarrow \mathbb{R}$ defined by*

$$\xi(\mathbf{u}) = u_0 \cdot \tilde{h}_s(\frac{\mathbf{u_N}}{u_0})$$

is convex

Proof. As ξ is C^2 (since $\tilde{h}$ is C^2), to prove that ξ is convex, we calculate the hessian and demonstrate that it is non-negative definite at all $\mathbf{u} \in \mathcal{R}$. Writing $\mathbf{D}^2$ and $\mathbf{H}$ for the total second derivative matrices on $\mathbb{R}^N$ and $\mathbb{R}^{N+1}$ respectively, $\mathbf{H}(\xi)$ is the symmetric matrix with the following components

$$\mathbf{H}_{i,j}(\xi)(\mathbf{u}) = \begin{cases} \frac{1}{u_0^3}\langle \mathbf{u}_N^T, (\mathbf{D}^2 h_s)(\frac{\mathbf{u}}{u_0}) \cdot \mathbf{u}_N \rangle & i,j = 0 \\ -\frac{u_N}{u_0^2}(\mathbf{D}^2 h_s)_j(\frac{\mathbf{u}_N}{u_0}) & i = 0, j > 0 \\ \frac{1}{u_0}(\partial_{u_i,u_j} h_s)(\frac{\mathbf{u}_N}{u_0}) & i,j > 0 \end{cases}$$

To see that $\mathbf{H}(\xi)$ is positive definite on $\mathcal{R}$, first identify the following components of $\mathbf{H}(\xi)$:

$$a := \mathbf{H}_{0,0}(\xi)$$

$$b_i := \mathbf{H}_{0,i}(\xi)(\mathbf{u}),\ i > 0$$

$$\mathbf{M}_{i,j} := \mathbf{H}_{i,j}(\xi),\ i,j > 0$$

Since $\tilde{h}_s$ is convex and C^2, and $\frac{1}{u_0}$ is positive for all $\mathbf{u} \in \mathcal{R}$, the matrix $\mathbf{M} = (\mathbf{M}_{i,j})_{i,j>0}$ is non-negative definite. With this definition of $\mathbf{M}$ observe that

$$a = \frac{\mathbf{u_N}^T}{u_0} \cdot \mathbf{M} \cdot \frac{\mathbf{u_N}}{u_0}$$

With these identifications available, applying $\mathbf{H}(\xi)(\mathbf{u})$ to a test vector $\mathbf{x} \in \mathbb{R}^{N+1}$ as a quadratic form gives

$$\mathbf{x}^T \cdot \mathbf{H}(\xi) \cdot \mathbf{x} = x_0^2 \cdot a + 2x_0 \sum_{i=1}^{N} x_i b_i + \mathbf{x_N}^T \cdot \mathbf{M} \cdot \mathbf{x_N}$$

Finally, replace a and b_i with their identities and let $\mathbf{v} = \frac{x_0}{u_0}\mathbf{u_N} - \mathbf{x}_N \in \mathbb{R}^N$ to see that

$$\mathbf{x}^T \cdot \mathbf{H}(\xi)(\mathbf{u}) \cdot \mathbf{x} = \frac{x_0^2}{u_0^2}(\mathbf{u}_N^T \cdot \mathbf{M} \cdot \mathbf{u}_N) - 2\frac{x_0}{u_0}(\mathbf{x_N} \cdot \mathbf{M} \cdot u_N) + (\mathbf{x}_N \cdot \mathbf{M} \cdot \mathbf{x}_N)$$

$$= \mathbf{v}^T \cdot \mathbf{M} \cdot \mathbf{v} \geq 0$$

since M is non-negative definite. As $\mathbf{x}$ is arbitrary the proof is complete.

□

4.4 Convexity Question of Method B

We want to answer the following questions:

1. What are sufficient conditions on $\mathcal{N}$ such that $\tilde{\alpha}(\mathbf{u})$ is equal to the gradient of a convex function?
2. What are sufficient conditions on $\mathcal{N}$ such that $\tilde{h}$ is a convex function?
3. What is the relation between $\nabla\tilde{h}$ and $\tilde{\alpha}$ (since they are distinct here)?

Answer To Question (1): We restrict to the case that $\mathcal{N}$ is $C^2(\mathcal{D})$ so that by the Poincare Lemma for 1-forms, i.e. vector fields, we obtain a simple sufficient condition:

Fact. *(Convex Antiderivatives) Let* $\mathbf{J}$ *denote the total derivative ("jacobian") matrix for* $N+1$ *dimensional vector field on* $\mathcal{R}$*, and* $\mathbf{D}^2$ *the total second-derivative ("hessian") matrix for* C^2 *functions on* $\mathcal{R}$*. If* $\tilde{\alpha}\colon \mathcal{R} \longrightarrow \mathbb{R}^{N+1}$ *is a* C^1 *vector field with* $\mathbf{J} = \mathbf{J}^T$ *and*

$$\mathbf{J}(\tilde{\alpha})(\mathbf{u}) \geq 0 \quad \forall\ \mathbf{u} \in \mathcal{R} \text{ (in the sense of matrices)}$$

then $\exists$ *convex* $f \in C^2(\mathcal{R}, \mathbb{R})$ *with* $\nabla f = \tilde{\alpha}$*.*

This fact says that $\tilde{\alpha}$ is the gradient of a convex function provided that $\mathbf{J}$ is symmetric, and $\mathbf{J}(\tilde{\alpha}) \geq 0$, so we seek sufficient conditions on $\mathcal{N}$ such that $\mathbf{J}(\tilde{\alpha}) \geq 0$ and $\mathbf{J} = \mathbf{J}^T$ by calculating $\mathbf{J}(\tilde{\alpha})$ explicitly (again, where $\tilde{\alpha}$ is defined via a neural network exactly as in (4.2.1). To simplify this calculation we define the **moment-reconstruction** induced by $\tilde{\alpha}$ as

$$\hat{\mathbf{u}}(\tilde{\alpha}(\mathbf{u}))$$

and note that this vector quantity satisfies the equality

$$\frac{\hat{\mathbf{u}}_N(\tilde{\alpha}(\mathbf{u}))}{\hat{u}_0(\tilde{\alpha}(\mathbf{u}))} = \frac{\int_{-1}^{1} \exp(\nabla\mathcal{N}(\frac{\mathbf{u_N}}{u_0}) \cdot \mathbf{m}_N)\mathbf{m}_N(v)dv}{\int_{-1}^{1} \exp(\nabla\mathcal{N}(\frac{\mathbf{u}_N}{u_0})\mathbf{m}_N)dv}$$

Using these identities, and writing $\hat{\mathbf{u}}$ in lieu of $\hat{\mathbf{u}}(\tilde{\alpha}(\mathbf{u}))$, it follows that $\mathbf{J}(\tilde{\alpha})$ is the the matrix with the following components:

$$\mathbf{J}_{i,j}(\tilde{\alpha})(\mathbf{u}) = \begin{cases} \frac{1}{u_0} + \frac{1}{u_0}(\frac{\mathbf{u}_N}{u_0} \cdot \mathbf{D}^2\mathcal{N}(\frac{u}{u_0}) \cdot \frac{\hat{\mathbf{u}}_N}{\hat{u}_0}) & i,j = 0 \\ \frac{-1}{u_0} \cdot \mathbf{D}_i^2(\mathcal{N})(\frac{\mathbf{u}_N}{u_0}) \cdot \frac{\hat{\mathbf{u}}_N}{\hat{u}_0} & i = 0, j > 0 \\ \frac{-1}{u_0} \cdot \mathbf{D}_i^2(\mathcal{N})(\frac{\mathbf{u}_N}{u_0}) \cdot \frac{\mathbf{u}_N}{u_0} & i > 0, j = 0 \\ \frac{1}{u_0}\mathbf{D}_{i,j}^2\mathcal{N}(\frac{u}{u_0}) & i,j > 0 \end{cases}$$

Note the asymmetry in the matrix $\mathbf{J}(\tilde{\alpha})$ occuring in the first column, and first row, due to the $\hat{\mathbf{u}}_N/\hat{u}_0$ vector. In the case where the reconstruction $\hat{\mathbf{u}}$ is "exact", then we would have that $\mathbf{J}(\tilde{\alpha})$ is not symmetric and therefore $\tilde{\alpha}$ is not the gradient of any function, let alone a convex one.

5 The Network Approximation

5.1 Defining Map

Using the approximation suggested by (3.3.3), and again recalling $D := \{u \in \mathcal{R}\colon u_0 \equiv 1\}$ we define $\mathcal{N}\colon \mathcal{D} \longrightarrow \mathbb{R}$ to be a neural network approximating h, and extend the approximation defined by $\mathcal{N}$ to all of $\mathcal{R}$ as $\tilde{h}$ by:

Definition. *(Network Approximation) Define* $\tilde{h}\colon \mathcal{R} \longrightarrow \mathbb{R}$ *via*

$$\begin{aligned} \tilde{h}(\mathbf{v}) &= \frac{v_1}{v_0} * (\mathcal{N}(\frac{v_1}{v_0}) + \log v_0) \\ \xi(v) &:= v_0\mathcal{N}(\frac{v_1}{v_0}) \text{ and } f(v) := v_0 \log v_0 \end{aligned} \tag{5.1.1}$$

In short, this means that we will only train the network on the set

$$D = \{u \in \mathcal{R} \colon u_0 \equiv 1\} = \{(1, v) \colon |v| < 1\}$$

and extend it to the rest of the space by the scaling relation above.

Given a network $\mathcal{N}$ and it's gradient $\mathcal{N}'$ at the set $u_0 \equiv 1$ which approximates h, equations (3.3.1) and (3.3.3) induce two separate ways to extend the approximate value of α, called $\tilde{\alpha}$, to the full set $\mathcal{R}$.

1. The first, using equation (3.3.1) is:

$$\tilde{\alpha}(\mathbf{v}) := \tilde{\alpha}(\mathbf{v}/v_0) + \begin{pmatrix} \log v_0 \\ 0 \end{pmatrix}$$

2. The second, using (5.1.1) is:

$$\tilde{\alpha}(\mathbf{v}) := \nabla \tilde{h}(\mathbf{v}) = \nabla f(\mathbf{v}) + \nabla \xi(\mathbf{v})$$

since both f and ξ have global definitions.

In the second case, this gives

$$\nabla f(v) = \begin{pmatrix} 1 + \ln(v_0) \\ 0 \end{pmatrix}$$

and

$$\nabla \xi(v) = \begin{pmatrix} \mathcal{N}(\frac{v_1}{v_0}) - \frac{v_1}{v_0} \cdot \mathcal{N}'(\frac{v_1}{v_0}) \\ \frac{1}{v_0} \mathcal{N}'(\frac{v_1}{v_0}) \end{pmatrix}$$

In total this gives

$$\tilde{\alpha}(\mathbf{v}) := \nabla f(\mathbf{v}) + \nabla \xi(\mathbf{v}) = \begin{pmatrix} 1 + \ln(v_0) + \mathcal{N}(\frac{v_1}{v_0}) - \frac{v_1}{v_0} \cdot \mathcal{N}'(\frac{v_1}{v_0}) \\ \frac{1}{v_0} \mathcal{N}'(\frac{v_1}{v_0}) \end{pmatrix} \tag{5.1.2}$$

In either case, we have that restricting to the domain of definition of $\mathcal{N}$, i.e. setting $v_0 \equiv 1$, gives

$$\tilde{\alpha}((1, v_1)) = \begin{pmatrix} 1 + \mathcal{N}(v_1) - v_1 \cdot \mathcal{N}'(v_1) \\ N'(v_1) \end{pmatrix} \tag{5.1.3}$$

In either regime, we always define

$$\tilde{u}(\mathbf{v}) := \hat{u}(\tilde{\alpha}(\mathbf{v})) \tag{5.1.4}$$

In the case of the networks that follow, we will use the first definition for extending $\tilde{\alpha}$. This ensures that the global error in the approximation, $\tilde{\alpha}$, is identical to the $\tilde{\alpha}$ error at the $u_0 \equiv 1$ line.

6 Appendix

6.1 Lagrange Duality in Convex Optimization

Consider the constrained optimization problem

$$\underset{x \in V}{\text{minimize}}\ \mathcal{L}(x) \text{ subject to } Ax = \mathbf{v}$$

where $v \in \mathbb{R}^n$, $A\colon V \longrightarrow \mathbb{R}^n$ is linear, and $\mathcal{L}$ is convex (to be specified conditionally), and V is a banach space (not necessarily finite dimensional).

$$\min_x \max_\lambda \mathcal{L}(x) - \lambda(Ax - v) = \min_x \max_\lambda \mathscr{L}(x, \lambda)$$

where $\mathscr{L}$ is called the **Lagrange Function**. The solution to this problem is related to that of the **dual problem**, the solution of which is obtained by reversing the order of minimization and maximization:

$$\max_\lambda \min_x \mathcal{L}(x) - \lambda(Ax - v) = \max_\lambda \ \mathscr{G}(\lambda)$$

where $\mathscr{G}(\lambda) := \min_x \mathscr{L}(x, \lambda)$ is called the **Lagrange Dual** and the variable λ is refered to as the **dual variable** for the problem, *not to be confused with the convex dual, and dual variable associated to the legendre transform.* The legendre transform of $\mathcal{L}$, and **not** $\mathscr{L}$ does, however, occur in the definition of the Lagrange Dual.

Observe

$$\mathscr{G}(\lambda) = \min_x \mathscr{L}(x, \lambda) = \min_x \mathcal{L}(x) - \lambda(Ax - \mathbf{v}) = \lambda\mathbf{v} - \max_x\{\langle\lambda, Ax\rangle - \mathcal{L}(x)\}$$

$$= \lambda\mathbf{v} - \max_x\{\langle A^t\lambda, x\rangle - \mathcal{L}(x)\}$$

This latter term is exactly $\mathcal{L}^*(A^t\lambda)$, since x is in the dual of $A^t\lambda$.

$$\max_\lambda \langle\lambda, v\rangle - \mathcal{L}^*(A^t(\lambda))$$

In particular, it is critical that we can identify $L^*(A^t\lambda)$ in the following case:

1. $V = L^1(\mathbb{R}, m)$
2. $\eta\colon \mathbb{R} \longrightarrow \mathbb{R}$ is a convex C^2 function
3. $\mathcal{L}(f) = \int_{\mathbb{R}} \eta(f)dm$

Notice that this is very different than computing the convex dual of a mechanical lagrangian, which, in classical mechanics, would be written perhaps as $\mathcal{L}$ *inside* the integral sign. Observe that for $g \in L^\infty(\mathbb{R})$ we have

$$\mathcal{L}^*(g) = \max_{f \in L^\infty} \langle g, f\rangle - \int_{\mathbb{R}} \eta(f)dm = \max_f \int_{\mathbb{R}} (g \cdot f - \eta(f))dm$$

Now, if g and f are sufficiently smooth, the variational derivative should be computed to solve for the optimum f, since that derivative should be zero at the optimum, say, f^*. This computation yields the following: $g = \eta'(f^*)$ which simply says that $f^* = (\eta^*)'(g)$, where η^* is the legendre transform of η. Substituting this $f^*(g)$ into the equation for $\mathcal{L}^*(g)$ gives us:

$$\mathcal{L}^*(g) = \int_{\mathbb{R}} \eta^*(g)$$

This yields the dual lagrange function

$$\mathscr{G}(\lambda) = \lambda\mathbf{v} - \langle\eta_*((A^t)\lambda(\cdot))\rangle = \lambda\mathbf{v} - \mathcal{L}^*((A^t\lambda)(v))$$

There are two problems, then:

$$(P) \min_x \max_\lambda \ \mathcal{L}(x) - \lambda(Ax - v)$$

$$(DP)\max_{\lambda}\ \mathscr{G}(\lambda) = \lambda\mathbf{v} - L^*(A^t\lambda)$$

Optimization Via Dual Problem

If $\mathscr{G}$ is solved for the optimum λ, say, $\tilde{\lambda}$, one may wish to recover the optimum argument for the original problem. Fortunately, we may substitute $\tilde{\lambda}$ into $\mathscr{L}(x,\lambda)$, and an optimum x will be found at $\mathscr{L}(x,\tilde{\Lambda})$ (note that no uniqueness is asserted). We may now solve

$$\min_x \mathcal{L}(x) - \tilde{\lambda}(Ax - v) = \min_x \mathscr{L}(\tilde{x},\tilde{\lambda})$$

so that, allowing the variational derivative, the following relation is obtained:

$$\delta\mathscr{L}(\tilde{x},\tilde{\lambda}) = 0 \implies \delta\mathcal{L}|_{\tilde{x}} = \tilde{\lambda}\delta A|_{\tilde{x}}$$

This is especially important where $\mathcal{L}$ is as above, and $\mathcal{A}(g) = \langle g\mathbf{m}\rangle$ is an integral operator. Then, $\tilde{g}$ can be solved again via variational derivative, in terms of $\tilde{\lambda}$:

$$\delta L|_{\tilde{g}} - \delta A|_{\tilde{g}} = 0 \implies (\eta')(g) = \tilde{\lambda}\cdot\mathbf{m} = A^t(\tilde{\lambda}) \implies g = (\eta^*)'(\tilde{\lambda}\cdot m)$$

so that $g(\tilde{\lambda})(v) = (\eta_*)'(\tilde{\lambda}\cdot\mathbf{m})$.