Research Notes:
A New Surrogate Closure Technique for the M_N System

William Porteous

Oak Ridge National Laboratory: CSM Department

December 30, 2020

Contents

These notes motivate and introduce my research at ORNL-CSM, and do not divulge our numerical results. For brevity, examples are built on a 1-dimensional velocity domain. We examine a velocity-discretized modification of a kinetic equation, the M_N equations, which we intend to solve numerically with a Runge-Kutta scheme. We are interested in improving the efficiency of this solution scheme by introducing a new technique to calculate the flux-update, a key step in the scheme, and are especially concerned with the efficiency of this method as N (a positive integer) increases. Our technique will employ the same Lagrange-duality framework of M_N to calculate the flux-update, but will supplant the $N+1$ dimensional optimization problem of this framework with an N-dimensional approximation problem.

1 Introduction

1.1 The Closure Framework for Kinetic Equations

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. This kinetic equation evolves a kinetic density 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 $\mathcal{C}$, appropriately called the "collision operator", introduces all interaction effects. In order to be well-posed, this equation requires specific initial and boundary conditions. With the intent of discretizing (1.1.1) in the velocity variable, we consider an orthonormal basis $\{\varphi_n\}_{n\in\mathbb{N}}$ for $L^2(V)$ and make the definitions

$$\mathbf{m}_N(v)=(\varphi_0(v),\varphi_1(v),...,\varphi_{n-1}(v))$$
$$\mathbf{u}\colon [0,\infty)\times X\longrightarrow\mathbb{R}^{N+1}$$

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

The function $\mathbf{u}$ is a vector containing the first $N+1$ moments of f according to the moment basis functions contained in $\mathbf{m}$. The moments $\mathbf{m}$ induce a non-linear Galerkin discretization of (1.1.1) along the velocity domain as

$$\partial_t\langle\mathbf{m}F_{\mathbf{u}}\rangle+\nabla_x\langle v\cdot\mathbf{m}F_{\mathbf{u}}\rangle=\langle\mathbf{m}\mathcal{C}(F_{\mathbf{u}})\rangle \tag{1.1.2}$$

where $F_{\mathbf{u}(t,x)}$ is some approximation to f with the same velocity moments as f at each point in space and time. That is to say, we demand that $F_{u(t,x)}$ is some ansatz for f which satisfies the constraint:

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

Of course, we want this ansatz F to be physically "reasonable", satisfying properties of our original kinetic equation (1.1.1) like conservation of mass and the H-Theorem.

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 minimal total entropy along the velocity domain with the moments $\mathbf{u}$. Let's make this notion precise, and for the sake of notation, henceforth write $\langle f\rangle$ to mean $\int_V f dv$.

Consider a convex, C^2 function, $\eta\colon\mathbb{R}_+\longrightarrow\mathbb{R}$, called the kinetic entropy density.

Definition (Entropy-Based Closure)**.** *In the case of* (1.1.2)*, we demand that* $F_{\mathbf{u}(x,t)}$ *is the solution to the following minimization problem:*

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

Where we let take η to be the Botlzmann entropy, $\eta(r)=r\log(r)-r$, and restrict to $N+1$ moments, i.e. $\mathbf{u}(x,t)=\langle f\cdot\mathbf{m}_N\rangle$, this is called the M_N closure framework, and the corresponding expression for (1.1.2) is called the M_N system.

Definition (M_N System)**.** *In (1.4.2), define the function $F(\mathbf{u}_N)$ as the extremal function of the following minimization (entropy-maximization) problem*

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

where $\langle \cdot \rangle$ denotes integration with respect to the velocity domain V, and $\eta(r) = r\log(r) - r$.

Definition (The Realizable Set)**.** *For the M_N framework, we denote the set of all possible moment vectors according to V and $\mathbf{m}$ by*

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

1.3 The Lagrange-Dual to Entropy-Minimization

The problem may have no solutions, or fail uniqueness, in the case that V is unbounded. There are explicit examples which saturate each of these. To continue, we assume that V is a convex, compact subset of $\mathbb{R}$. For our purposes, we will often consider $V = [-1, 1]$, and in that case, we will take that $\mathbf{m}_N$ is the vector of the first $N+1$ Legendre polynomials (including the 0th order identity map).

Definition (Minimum Entropy Function)**.**

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

$$h(\mathbf{u}) = \min \langle \eta(g) \rangle \textit{ subject to } g \in L^1(V),\ \langle \mathbf{m} g \rangle = \mathbf{u} \tag{1.3.1}$$

The above minimization problem of the M_N system has the following unconstrained dual problem (writing η_* for the legendre-transform of η):

$$\text{minimize } \langle \eta_*(\alpha^t \mathbf{m}) \rangle - \alpha^t \cdot \mathbf{u}\ , \alpha \in \mathbb{R}^{N+1} \tag{1.3.2}$$

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

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

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

Fact (Primal Solution)**.** *By necessary conditions for duality, the solution $\hat{\alpha}(\mathbf{u})$ to the dual problem obtains the primal solution to (1.3). Explicitly, the function $G_\alpha(v)$ defined as*

$$G_\alpha(v) = \eta_*'(\alpha^t \mathbf{m}) = \exp(\alpha^t \mathbf{m}) \tag{1.3.3}$$

is the solution to (1.3)

Fact (Gradient is Dual Optimum)**.** *First order conditions for duality show that*

$$\nabla h(\mathbf{u}) = \hat{\alpha}(\mathbf{u}) \tag{1.3.4}$$

From the above fact, it immediately follows that $\hat{\alpha}$ is therefore an invertible function, let its inverse be given by $\hat{u}(\alpha)$ for admissable vectors $\alpha \in \mathbb{R}^{N+1}$.

1.4 The Computational Problem: An Explicit Example

Consider the case $N = 1$ and the following setup:

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

Note that the functions in $\mathbf{m}$ are orthogonal, but not normalized in $L^2(V)$. We define a kinetic distribution function $f = f(x, \mu, t)$ for $x \in \mathbb{R}, \mu \in V = [-1, 1], t \geq 0$ which evolves according to the following **linear kinetic equation**

$$\partial_t f + \mu \partial_x f + \sigma_a f = \sigma_s (\frac{1}{2} \langle f \rangle - f) \tag{1.4.1}$$

We write out what this means for our Galerkin discretization (1.1.2):

$$\partial_t \mathbf{u}_N + \partial_x \langle \mu \mathbf{m}_N f \rangle + \sigma_a \mathbf{u}_N = -\sigma_s \mathbf{Q} \mathbf{u}_N$$

This is not "closed" in the first $N + 1$ moments of f, since the second term on the lefthand-side contains higher order moments of f. We close this as an evolution of $\mathbf{u}_N$ with our M_N technique, giving:

$$\partial_t \mathbf{u}_N + \partial_x \langle \mu \mathbf{m}_N F(\mathbf{u}_N) \rangle + \sigma_a \mathbf{u}_N = -\sigma_s \mathbf{Q} \mathbf{u}_N \tag{1.4.2}$$

To numerically solve a discretization of the PDE (1.4.2) via a Runge-Kutta scheme, we must compute $G(\mathbf{u}_N)$ via $G_\alpha(v)$, which is defined by (1.3.2). Each update to the numerical $\mathbf{u}_N$, requires the computation of $\hat{\alpha}(\mathbf{u})$. This can be done accurately for each value of $\mathbf{u}_N$, for example, via a gradient-descent optimization algorithm. This method has the advantage that, with high accuracy for values of α, we are assured that $\hat{\alpha}(\mathbf{u})$ are inherited from the convex function h, ensuring our solution enjoys a variety of desireable properties. However, this is computationally slow – the optimization routine becomes much slower as the number of moments (N) increases, and similarly, if the velocity and position domains increase in dimension.

2 Proposed Method

We can approximate the entropy function h with a function $\tilde{h}$, which is convex and C^2, but which also admits a closed-form, forward-evaluation expression for its gradient $\hat{\alpha}$.

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

In principal, this can be done on the full 2 and 3-dimensional realizable sets for the M_1*equation* and M_2 equations. However, we were able to derive scaling relations which enabled us to reduce the problem to approximating h on the interior of 1-dimensional segment, and on the interior of a parabola.

2.1 Metric for Approximation Accuracy

The term which we desire to compute more efficiently in (1.4.1) 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

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

2.2 Dimension-Reduction: From Unbounded to Bounded Regions

2.2.1 Scaling Relations

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

Fact (Alpha Scaling)**.**

$$\forall \lambda > 0$$
$$\hat{\alpha}(\lambda \mathbf{u}) = \alpha(\mathbf{u}) + \begin{pmatrix} \ln(\lambda) \\ 0 \end{pmatrix} \tag{2.2.1}$$

from which it follows that

Fact (Entropy Scaling)**.**

$$\forall\ \lambda > 0$$
$$h(\lambda \mathbf{u}) = \lambda(h(\mathbf{u}) + u_0 \ln(\lambda)) \tag{2.2.2}$$

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 $\mathbf{v} = v_0 \begin{pmatrix} 1 \\ \frac{v_1}{v_0} \end{pmatrix}$*

$$h(\mathbf{v}) = v_0(h((\frac{\mathbf{v}}{v_0})) + ln(v_0)) \tag{2.2.3}$$

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 $\mathcal{D}$

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

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

2.2.2 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 $\tilde{h}$.

3 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 $\mathcal{R}$.

3.1 Function-Based Approximation: Method A

By using scaling relation (2.2.2) in the specific case of (2.2.3), 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:

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

3.2 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 (2.2.1), 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 (2.2.1). 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*

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

$$\tilde{\alpha}_s^i(\mathbf{u}) = \begin{cases} -\log \int_{-1}^{1} \exp(\nabla \mathcal{N} \cdot \mathbf{m}_N) & i = 0 \\ \nabla^i \mathcal{N}(\mathbf{u}) & i > 0 \end{cases} \tag{3.2.1}$$

Extend $\tilde{\alpha}_s$ to the whole domain $\mathcal{R}$ according to (2.2.1) 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

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

3.3 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.

□

3.4 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 (3.2.1). 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:

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

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.