



Model Order Reduction of Rarefied Gases Using Neural Networks

Zachary Schellin | Institut für Numerische Fluiddynamik



Outline

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Governing equations

Boltzmann equation with the BGK operator

$$\boxed{\text{transport}} \quad \partial_t f + v \partial_x f = \boxed{\text{BGK op.}} \quad \frac{1}{\tau} (M_f - f) \quad (1)$$

¹PhysRev.94.511.





Governing equations

Boltzmann equation with the BGK operator

$$\begin{array}{|c|} \hline \text{transport} \\ \hline \partial_t f + v \partial_x f \\ \hline \end{array} = \begin{array}{|c|} \hline \text{BGK op.} \\ \hline \frac{1}{\tau} (M_f - f) \\ \hline \end{array} \quad (1)$$

-Equilibrium solution: Maxwellian distribution

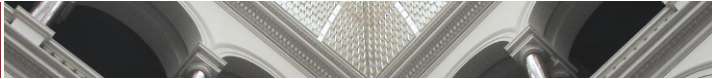
M_f

$$M_f = \frac{\rho(x, t)}{(2\pi RT(x, t))^{\frac{3}{2}}} \exp\left(-\frac{(v - u(x, t))^2}{2RT(x, t)}\right) \quad (2)$$

1

¹PhysRev.94.511.





Governing equations

Boltzmann equation with the BGK operator

$$\begin{array}{|c|} \hline \text{transport} \\ \hline \partial_t f + v \partial_x f \\ \hline \end{array} = \begin{array}{|c|} \hline \text{BGK op.} \\ \hline \frac{1}{\tau} (M_f - f) \\ \hline \end{array} \quad (1)$$

-Equilibrium solution: Maxwellian distribution
 M_f

$$M_f = \frac{\rho(x, t)}{(2\pi RT(x, t))^{\frac{3}{2}}} \exp\left(-\frac{(v - u(x, t))^2}{2RT(x, t)}\right) \quad (2)$$

Scale the duration to evolve into equilibrium:

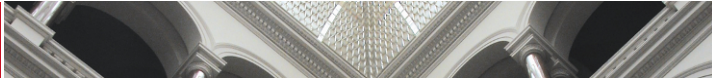
relaxation time τ

$$\tau^{-1} = \frac{\rho(x, t) T^{1-\nu}(x, t)}{Kn} \quad (3)$$

1

¹PhysRev.94.511.





Governing equations

Boltzmann equation with the BGK operator

$$\boxed{\text{transport}} \quad \partial_t f + v \partial_x f = \boxed{\text{BGK op.}} \quad \frac{1}{\tau} (M_f - f) \quad (1)$$

-Equilibrium solution: Maxwellian distribution M_f

$$M_f = \frac{\rho(x, t)}{(2\pi RT(x, t))^{\frac{3}{2}}} \exp\left(-\frac{(v - u(x, t))^2}{2RT(x, t)}\right) \quad (2)$$

Scale the duration to evolve into equilibrium:
relaxation time τ

$$\tau^{-1} = \frac{\rho(x, t) T^{1-\nu}(x, t)}{Kn} \quad (3)$$

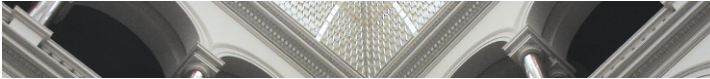
Rarefaction level: Knudsen number Kn

$$Kn = \frac{\lambda}{l} \quad (4)$$

1

¹PhysRev.94.511.





Knudsen number

- Solution is $f(x, v, t)$ in 1D and $f(x, y, v_x, v_y, t)$ in 2D and $f(x, y, z, v_x, v_y, v_z, t)$ in 3D

²NumaKUL.





Knudsen number

- Solution is $f(x, v, t)$ in 1D and $f(x, y, v_x, v_y, t)$ in 2D and $f(x, y, z, v_x, v_y, v_z, t)$ in 3D

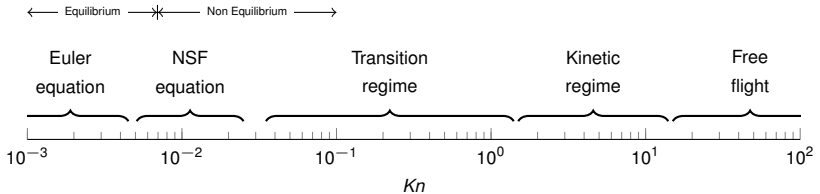


Figure: Partitioning of Kn , the Knudsen number, into levels of rarefaction.

2

²NumaKUL.





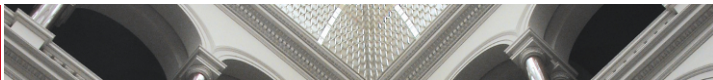
Discretization in space and velocity space in 1D

- **Space and time discretization:** $x_j = j\Delta x$ and $j \in \mathbb{Z}$, $v_k = k\Delta v$ and $k \in \mathbb{Z}$, $t^j = i\Delta t$ and $t \in \mathbb{N}$,
- **Leads to:** set of ODE's in time

$$\partial_t f_{j,k} = -(v_k)_1 D_x f|_{j,k}(t) + \frac{1}{\tau} (M_{f_{j,k}}(t) - f_{j,k}(t)). \quad (5)$$

- **KJ first-order differential equations:**
 K gridpoints in space & J number of gridpoints in velocity space
- **3D:**
 $K^3 J^3$ first-order differential equations
- **Evaluation requires:** the moments of f .





Moments/ Expected values of f

Question: How do we get the moments of f ?

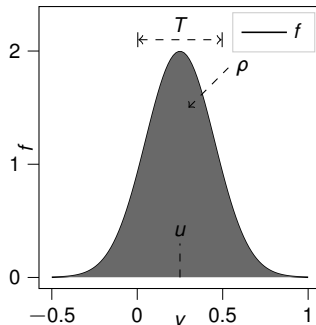


Figure: Illustration of the linkage between f and the moments of f .





Moments/ Expected values of f

Question: How do we get the moments of f ?

- Collision invariants $\Phi(v) = [1, v, \frac{1}{2}v^2]$

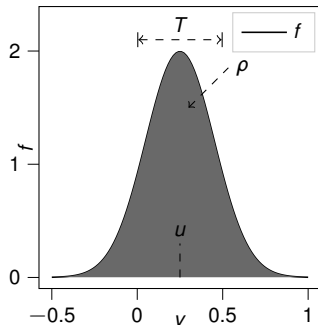


Figure: Illustration of the linkage between f and the moments of f .





Moments/ Expected values of f

Question: How do we get the moments of f ?

- Collision invariants $\Phi(v) = [1, v, \frac{1}{2}v^2]$
- The first moment/ **Density** is

$$\rho(x, t) = \int f \, dv, \quad (6)$$

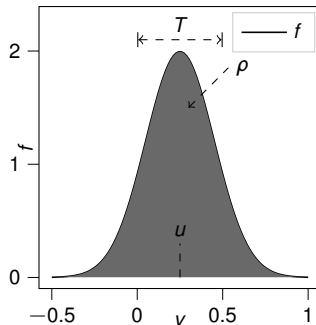


Figure: Illustration of the linkage between f and the moments of f .





Moments/ Expected values of f

Question: How do we get the moments of f ?

- Collision invariants $\Phi(v) = [1, v, \frac{1}{2}v^2]$
- The first moment/ **Density** is

$$\rho(x, t) = \int f \, dv, \quad (6)$$

- the second moment/ **Momentum** is

$$\rho(x, t)u(x, t) = \int v f \, dv, \quad (7)$$

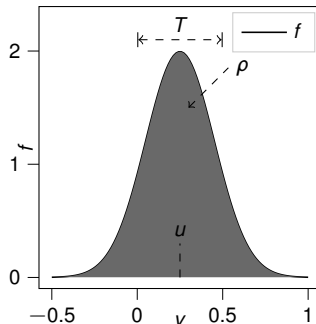


Figure: Illustration of the linkage between f and the moments of f .





Moments/ Expected values of f

Question: How do we get the moments of f ?

- Collision invariants $\Phi(v) = [1, v, \frac{1}{2}v^2]$
- The first moment/ **Density** is

$$\rho(x, t) = \int f \, dv, \quad (6)$$

- the second moment/ **Momentum** is

$$\rho(x, t)u(x, t) = \int v f \, dv, \quad (7)$$

- the third moment/ **Energy** is

$$E(x, t) = \int \frac{1}{2} v^2 f \, dv. \quad (8)$$

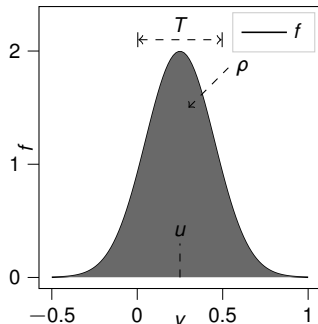


Figure: Illustration of the linkage between f and the moments of f .





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Sod's shock tube

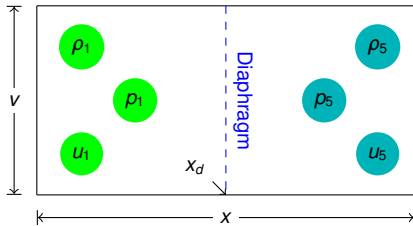


Figure: Problem setup of Sod's shock tube for the BGK model in 1D at $t = 0s$.

- **Test case** for numerical schemes solving
- **non-linear hyperbolic conservation laws** in gas dynamics (Gary A. Sod in 1978)
- **Idea:**
 - Solve problem analytically (Rankine-Hugoniot jump conditions)
 - Solve problem numerically
 - Compare results especially **resolution of discontinuities**





Sod's shock tube

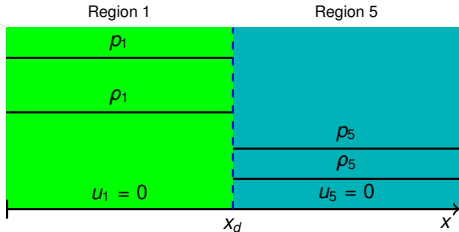


Figure: Problem setup of Sod's shock tube for the BGK model in 1D at $t = 0s$.

- **Test case** for numerical schemes solving
- **non-linear hyperbolic conservation laws** in gas dynamics (Gary A. Sod in 1978)
- **Idea:**
 - Solve problem analytically (Rankine-Hugoniot jump conditions)
 - Solve problem numerically
 - Compare results especially **resolution of discontinuities**





Sod's shock tube

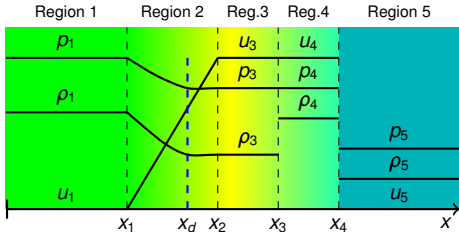


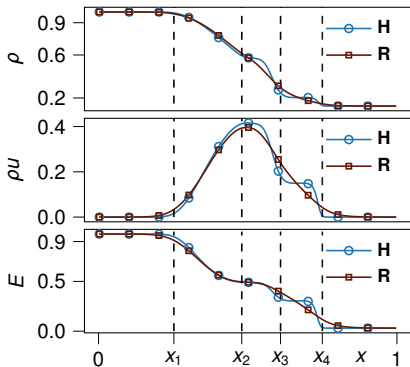
Figure: Problem setup of Sod's shock tube for the BGK model in 1D at $t > 0s$.

- **Test case** for numerical schemes solving
- **non-linear hyperbolic conservation laws** in gas dynamics (Gary A. Sod in 1978)
- **Idea:**
 - Solve problem analytically (Rankine-Hugoniot jump conditions)
 - Solve problem numerically
 - Compare results especially **resolution of discontinuities**
 - » x_1 head of rarefaction wave
 - » x_2 tail of rarefaction wave
 - » x_3 contact discontinuity
 - » x_4 position of shockwave





Two Case Studies



- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
 - **H**, $Kn = 0.00001$, "Continuum Flow"
 - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities

H	R
• x_1	• x_1
• x_2	• x_2
• x_3	• x_3
• x_4	• x_4

Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.





Two Case Studies

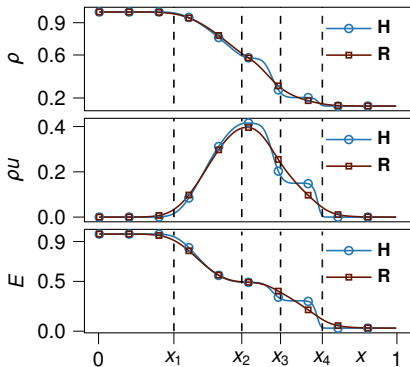


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
 - **H**, $Kn = 0.00001$, "Continuum Flow"
 - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities

H	R
• X₁	• X₁
• X₂	• X₂
• X ₃	• X ₃
• X ₄	• X ₄





Two Case Studies

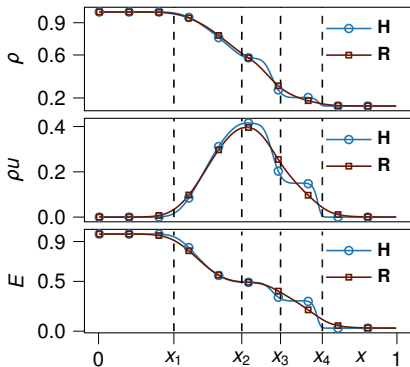
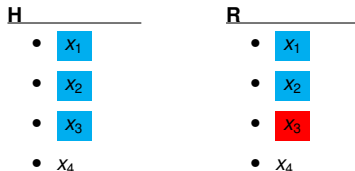


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
 - **H**, $Kn = 0.00001$, "Continuum Flow"
 - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities





Two Case Studies

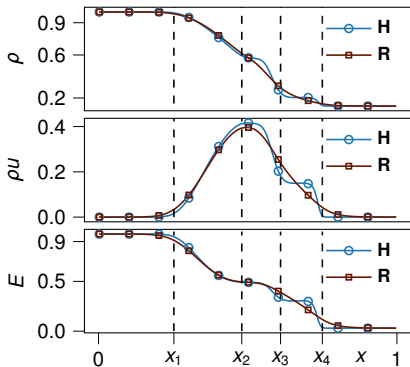
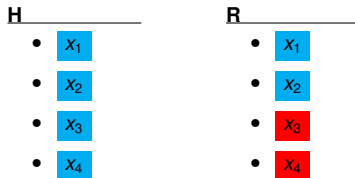


Figure: Moments of **H** and **R** at $t = 0.12s$ and $v = v_0$ in Sod's shock tube.

- Two solutions of the BGK model in Sod's shock tube
- Two levels of rarefaction
 - **H**, $Kn = 0.00001$, "Continuum Flow"
 - **R**, $Kn = 0.001$, "Slip flow"
- Present discontinuities





Two Case Studies

- Spatial resolution $J = 200$
- Temporal resolution $I = 25$
- Velocious resolution $K = 40$

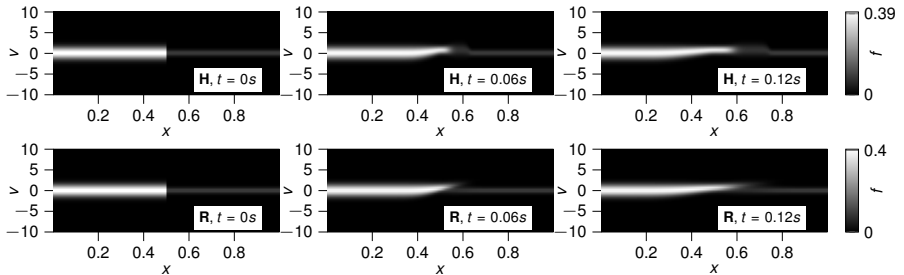


Figure: Solution f for **H** and **R** in x and v .



Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

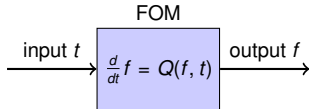
Discussion



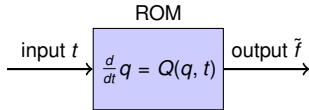


Model Order Reduction

- **Goal:** Reduce computational cost



(a) Evolving the FOM in time.



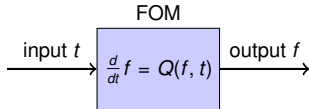
(b) Evolving the ROM in time.

Figure: In the online phase the operator Q is different for the FOM and the ROM.

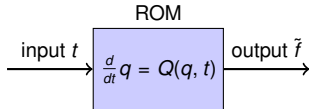




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

– **Goal:** Reduce computational cost

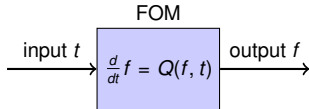
- $f(x, v, t)$ with KJ ODE's in time for 1D

Figure: In the online phase the operator Q is different for the FOM and the ROM.

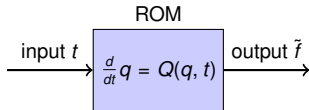




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

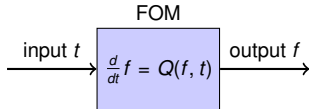
- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm

Figure: In the online phase the operator Q is different for the FOM and the ROM.

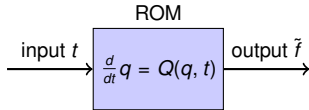




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

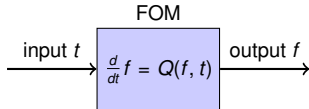
- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)

Figure: In the online phase the operator Q is different for the FOM and the ROM.

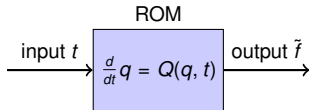




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

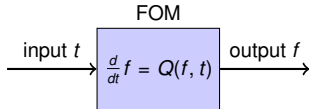
- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)
- **Require:** Solution of f (only few timesteps)

Figure: In the online phase the operator Q is different for the FOM and the ROM.

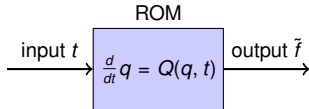




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

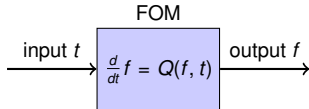
- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)
- **Require:** Solution of f (only few timesteps)
- **Reduce:** $POD(f(x, v, t)) = q(x, n, t)$

Figure: In the online phase the operator Q is different for the FOM and the ROM.

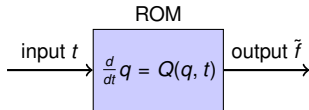




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

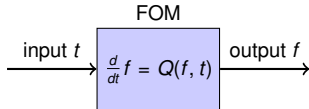
- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)
- **Require:** Solution of f (only few timesteps)
- **Reduce:** $POD(f(x, v, t)) = q(x, n, t)$
 - P is # n with $P \ll K$

Figure: In the online phase the operator Q is different for the FOM and the ROM.

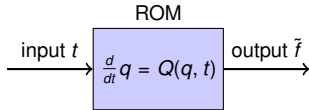




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

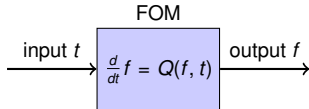
- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)
- **Require:** Solution of f (only few timesteps)
- **Reduce:** $POD(f(x, v, t)) = q(x, n, t)$
 - P is # n with $P \ll K$
 - KJ ODE's vs. PJ ODE's

Figure: In the online phase the operator Q is different for the FOM and the ROM.

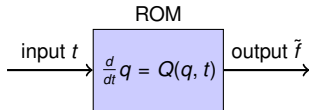




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

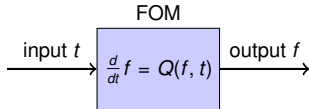
Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)
- **Require:** Solution of f (only few timesteps)
- **Reduce:** $POD(f(x, v, t)) = q(x, n, t)$
 - P is # n with $P \ll K$
 - KJ ODE's vs. PJ ODE's
- **Evolve in time** $\rightarrow Q(q, t) = \tilde{f}$

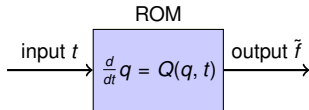




Model Order Reduction



(a) Evolving the FOM in time.



(b) Evolving the ROM in time.

Figure: In the online phase the operator Q is different for the FOM and the ROM.

- **Goal:** Reduce computational cost
 - $f(x, v, t)$ with KJ ODE's in time for 1D
- **Require:** Reduction algorithm
 - Proper Orthogonal Decomposition (**POD**)
 - Neural Networks (**NN**)
- **Require:** Solution of f (only few timesteps)
- **Reduce:** $POD(f(x, v, t)) = q(x, n, t)$
 - P is # n with $P \ll K$
 - KJ ODE's vs. PJ ODE's
- **Evolve in time** $\rightarrow Q(q, t) = \tilde{f}$
- $f - \tilde{f} < \epsilon$





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Proper Orthogonal Decomposition

- Solution of a PDE is $f(x, v, t)$ can be obtained





Proper Orthogonal Decomposition

- Solution of a PDE is $f(x, v, t)$ can be obtained
 - Discretization into a system of ODE's





Proper Orthogonal Decomposition

- Solution of a PDE is $f(x, v, t)$ can be obtained
 - Discretization into a system of ODE's
 - Separation of variables ansatz





Proper Orthogonal Decomposition

- Solution of a PDE is $f(x, v, t)$ can be obtained
 - Discretization into a system of ODE's
 - Separation of variables ansatz

$$\gg f(t, v, x) = \sum_{i=1}^n a_i(t) \Phi_i(x, v) \quad (9)$$






Proper Orthogonal Decomposition

- How to get Φ_i ?






- How to get Φ_i ?

- » $X =$ 



- How to get Φ_i ?

- » $X =$ 

- **Truncation:** $\tilde{X} = \begin{matrix} & \Phi \\ \tilde{U} & \end{matrix} \tilde{\Sigma} \tilde{V}^* \quad (10), \quad \tilde{U} = \Phi = [\Phi_1, \Phi_2, \dots, \Phi_p] \quad (11)$





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Terminology

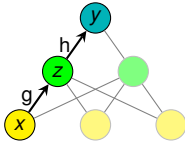


Figure: Example of a simple network.

- Network with three layers



Terminology

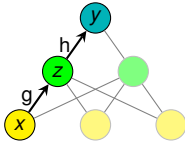


Figure: Example of a simple network.

– Network with three layers

- Input layer
- Hidden layer
- Output layer





Terminology

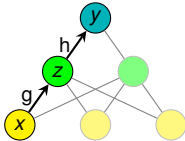


Figure: Example of a simple network.

– **Layer:** Stage of computation

– Network with three layers

- Input layer
- Hidden layer
- Output layer





Terminology

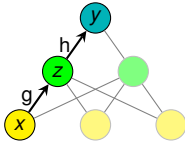


Figure: Example of a simple network.

- **Layer:** Stage of computation
- **Computations/ Forward pass**
 - $g(x) = g(xW + b) = z$
 - $h(z) = h(zW + b) = y$
 - » $h(g(x)) = y$

– Network with three layers

- Input layer
- Hidden layer
- Output layer





Terminology

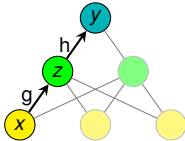


Figure: Example of a simple network.

– Network with three layers

- Input layer
- Hidden layer
- Output layer

- **Layer:** Stage of computation
- **Computations/ Forward pass**
 - $g(x) = g(xW + b) = z$
 - $h(z) = h(zW + b) = y$
 - » $h(g(x)) = y$
- **Neuron:** Entry in 'Tensor'





Terminology

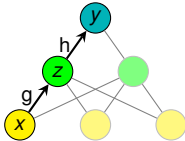


Figure: Example of a simple network.

– Network with three layers

- Input layer
- Hidden layer
- Output layer

- **Layer:** Stage of computation
- **Computations/ Forward pass**
 - $g(x) = g(xW + b) = z$
 - $h(z) = h(zW + b) = y$
 - » $h(g(x)) = y$
- **Neuron:** Entry in 'Tensor'
- **Trainable parameters:** W, b





Autoencoders

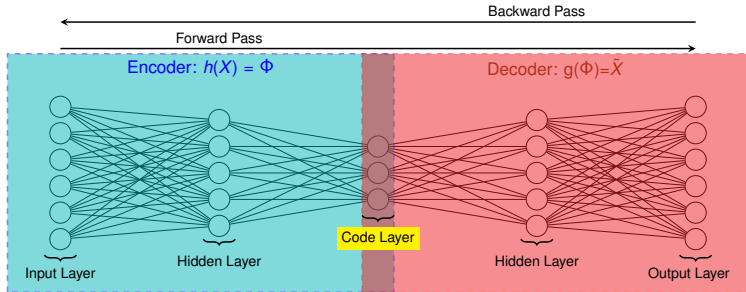


Figure: A fully connected autoencoder.





Autoencoders

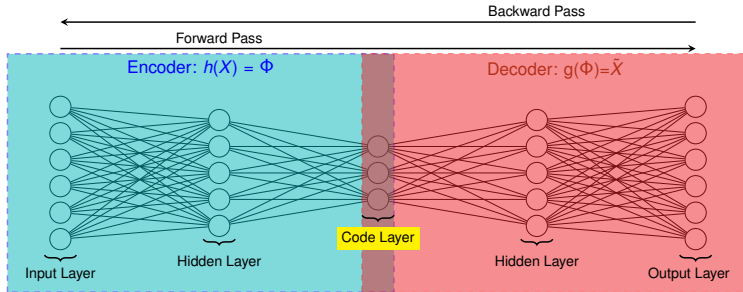
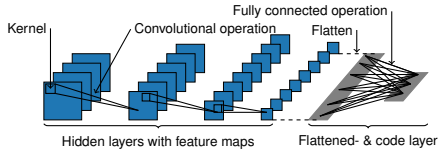


Figure: A fully connected autoencoder.

- **Structure:** Encoder & Decoder
- **Layers:** Input-, Output- and Code layer
- **Category:** Self-supervised learning
- **Main hyperparameters:**
Number & size of hidden layers
esp. size of code layer



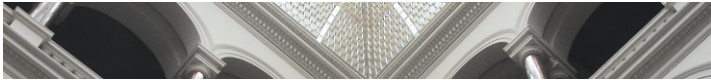
Convolutional Autoencoder



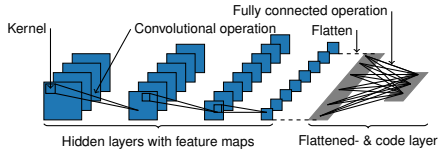
– Designed for 2D/3D input

(a) Encoder of a convolutional autoencoder without input layer.

Figure: Fundamental features of conv. networks.

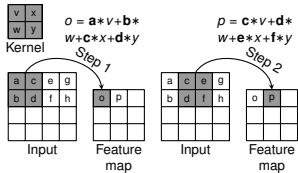


Convolutional Autoencoder



– Designed for 2D/3D input

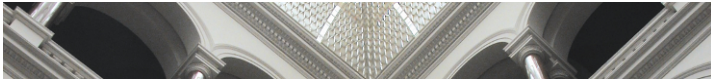
(a) Encoder of a convolutional autoencoder without input layer.



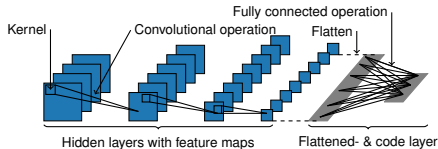
(b) Convolutional operation, 1 strided.

Figure: Fundamental features of conv. networks.



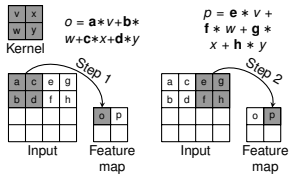


Convolutional Autoencoder



– Designed for 2D/3D input

(a) Encoder of a convolutional autoencoder without input layer.

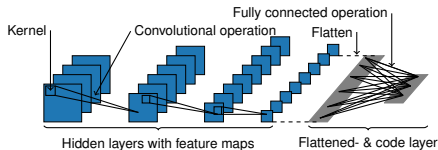


(b) Convolutional operation, 2 strided.

Figure: Fundamental features of conv. networks.

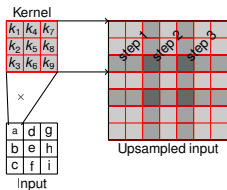


Convolutional Autoencoder



– Designed for 2D/3D input

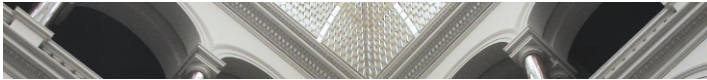
(a) Encoder of a convolutional autoencoder without input layer.



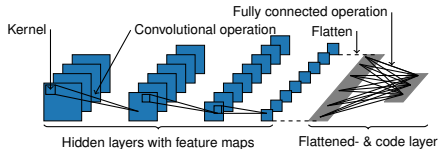
(b) Even deconvolution

Figure: Fundamental features of conv. networks.



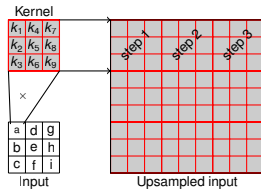


Convolutional Autoencoder



– Designed for 2D/3D input

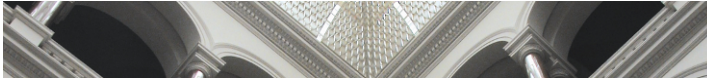
(a) Encoder of a convolutional autoencoder without input layer.



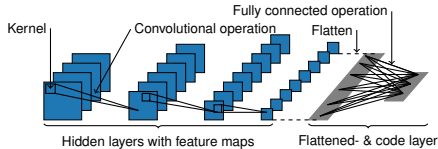
(b) Uneven deconvolution

Figure: Fundamental features of conv. networks.



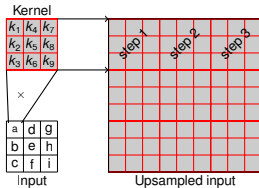


Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.

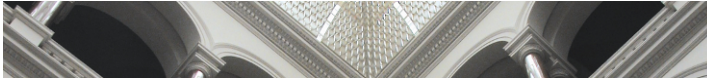
- Designed for 2D/3D input
- **Peculiarities:** Sparse connections, parameter sharing
 - Promotes generalization



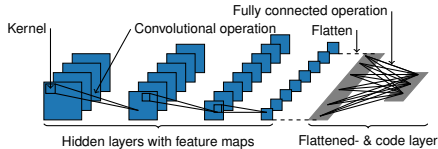
(b) Uneven deconvolution

Figure: Fundamental features of conv. networks.

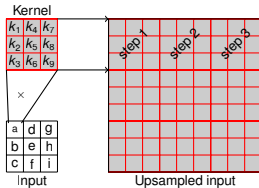




Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.



(b) Uneven deconvolution

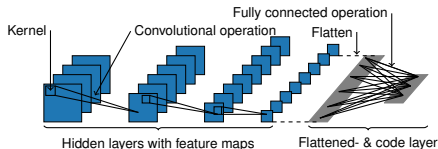
- Designed for 2D/3D input
- **Peculiarities:** Sparse connections, parameter sharing
 - Promotes generalization
- **Hyperparameters:** Number & size of layers, kernel dimensions, stride increments
 - Non-trivial influence of output dimensions & quality

Figure: Fundamental features of conv. networks.

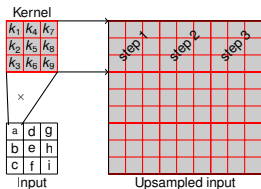




Convolutional Autoencoder



(a) Encoder of a convolutional autoencoder without input layer.



(b) Uneven deconvolution

- Designed for 2D/3D input
- **Peculiarities:** Sparse connections, parameter sharing
 - Promotes generalization
- **Hyperparameters:** Number & size of layers, kernel dimensions, stride increments
 - Non-trivial influence of output dimensions & quality

Figure: Fundamental features of conv. networks.





Training



Figure: A very simple network

– **Forward propagation:**

$$y = h(z, b) = h((g(x, a)), b) \quad (13)$$





Training



Figure: A very simple network

– **Forward propagation:**

$$y = h(z, b) = h((g(x, a)), b) \quad (13)$$

– **Loss function:**

$$L(y, \tilde{y}_0) = \frac{1}{2} (y - \tilde{y}_0)^2 = E \quad (14)$$





Training



Figure: A very simple network

- **Forward propagation:**

$$y = h(z, b) = h((g(x, a)), b) \quad (13)$$

- **Loss function:**

$$L(y, \tilde{y}_0) = \frac{1}{2} (y - \tilde{y}_0)^2 = E \quad (14)$$

- **Backpropagation:**





Training



Figure: A very simple network

– **Forward propagation:**

$$y = h(z, b) = h((g(x, a)), b) \quad (13)$$

– **Loss function:**

$$L(y, \tilde{y}_0) = \frac{1}{2} (y - \tilde{y}_0)^2 = E \quad (14)$$

– **Backpropagation:**

- $\frac{\partial E}{\partial a} = -(y_0 - y) \frac{\partial y}{\partial z} \frac{\partial z}{\partial a} \quad (15)$
- $\frac{\partial E}{\partial b} = -(y_0 - y) \frac{\partial y}{\partial b} \quad (16)$





Training



Figure: A very simple network

– **Forward propagation:**

$$y = h(z, b) = h((g(x, a)), b) \quad (13)$$

– **Loss function:**

$$L(y, \tilde{y}_0) = \frac{1}{2} (y - \tilde{y}_0)^2 = E \quad (14)$$

– **Backpropagation:**

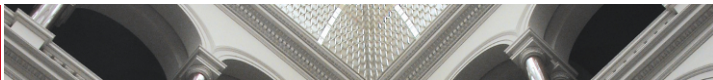
- $\frac{\partial E}{\partial a} = -(y_0 - y) \frac{\partial y}{\partial z} \frac{\partial z}{\partial a} \quad (15)$
- $\frac{\partial E}{\partial b} = -(y_0 - y) \frac{\partial y}{\partial b} \quad (16)$

– **Optimize:**

- $a_{i+1} = a_i + \epsilon \frac{\partial E}{\partial a_i} \quad (17)$

- $b_{i+1} = b_i + \epsilon \frac{\partial E}{\partial b_i} \quad (18)$





Training



Figure: A very simple network

- **Forward propagation:**

$$y = h(z, b) = h((g(x, a)), b) \quad (13)$$

- **Loss function:**

$$L(y, \tilde{y}_0) = \frac{1}{2} (y - \tilde{y}_0)^2 = E \quad (14)$$

- **Backpropagation:**

- $\frac{\partial E}{\partial a} = -(y_0 - y) \frac{\partial y}{\partial z} \frac{\partial z}{\partial a} \quad (15)$
- $\frac{\partial E}{\partial b} = -(y_0 - y) \frac{\partial y}{\partial b} \quad (16)$

- **Optimize:**

- $a_{i+1} = a_i + \epsilon \frac{\partial E}{\partial a_i} \quad (17)$

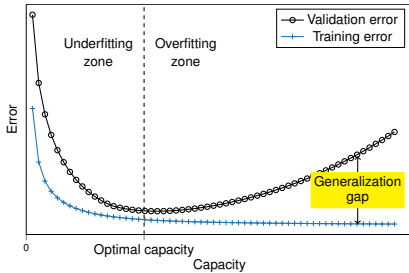
- $b_{i+1} = b_i + \epsilon \frac{\partial E}{\partial b_i} \quad (18)$

- **Hyperparameter:** learning rate ϵ





Concepts



– Over- and Underfitting:

Figure: Influence of capacity





Concepts

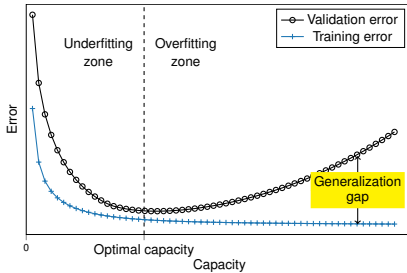


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity





Concepts

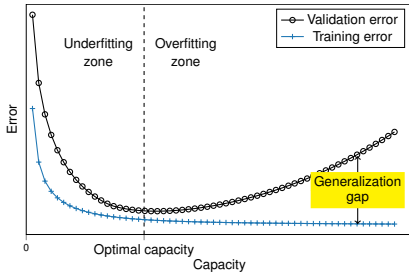


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**





Concepts

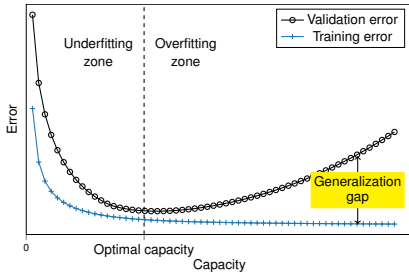


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**
 - Size of the network





Concepts

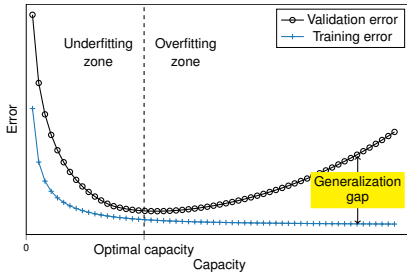


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**
 - Size of the network
 - Activation functions





Concepts

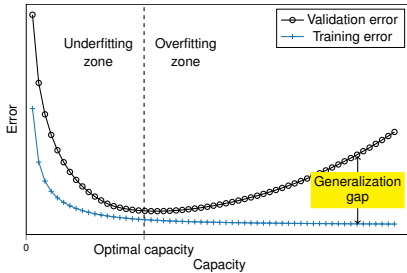


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**
 - Size of the network
 - Activation functions
 - Loss function





Concepts

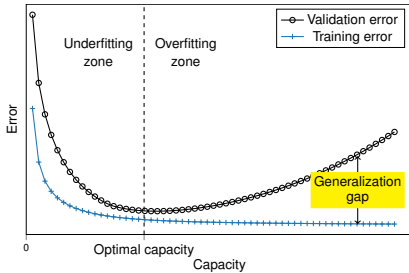


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**
 - Size of the network
 - Activation functions
 - Loss function
 - Data distortion/ add variation to existing data





Concepts

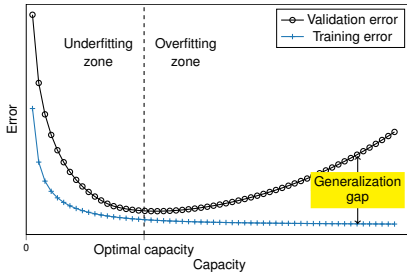
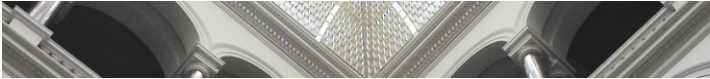


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**
 - Size of the network
 - Activation functions
 - Loss function
 - Data distortion/ add variation to existing data
 - ...
 - Any other means of regularization





Concepts

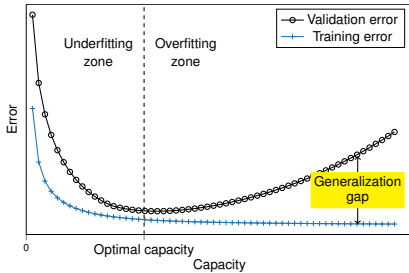


Figure: Influence of capacity

- **Over- and Underfitting:**
 - **Goal:** Reach optimal capacity
- **How to direct capacity ?:**
 - Size of the network
 - Activation functions
 - Loss function
 - Data distortion/ add variation to existing data
 - ...
 - Any other means of regularization
- **Initialization**





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





Number of intrinsic variables

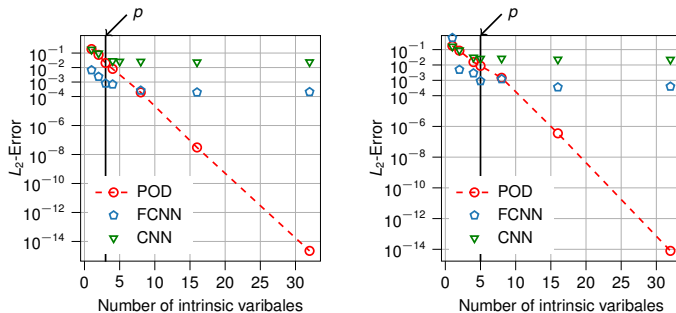


Figure: Variation of p for \mathbf{H} left and \mathbf{R} right.

– Evaluation metric:

$$L_2\text{-Error} = \frac{\|f - \tilde{f}\|_2}{\|f\|_2} \quad (19)$$





Amount of parameters

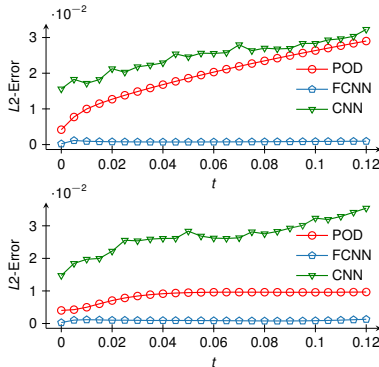
Table: Amount of parameters used to reconstruct f , the number of intrinsic variables p and the corresponding L_2 -Error for POD, the FCNNs, and the CNN.

Algorithm	Parameters		Int. variables p		L_2 -error	
	H	R	H	R	H	R
POD	15129	25225	3	5	0.0205	0.0087
FCNN	2683	3725	3	5	0.0008	0.0009
CNN	8246	8246	5	5	0.025	0.027





Time dependence of L_2 -Error



– POD:

- **H** - lin. increase of L_2
- **R** - increase & stagnation of L_2

– FCNN:

- **H & R** - no distinct time dependence L_2
- biggest value at onset

– CNN:

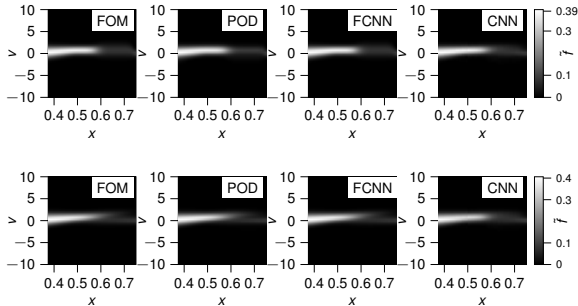
- **H & R** - similar evolution

Figure: Comparison of the L_2 -Error over time, **H** top and **R** bottom.





A detailed look at reconstructions



– POD:

- **H** - defective after $x = 0.6$:
Errors in temperature
- **R** - almost exact

– FCNN:

- **H** & **R** - almost exact

– CNN:

- **H** & **R** - average of **H** & **R**

Figure: Comparison of f and \tilde{f} at $t = t_{end}$ and $x \in [0.375, 0.75]$, **H** top and **R** bottom.



Moments of f and \tilde{f}

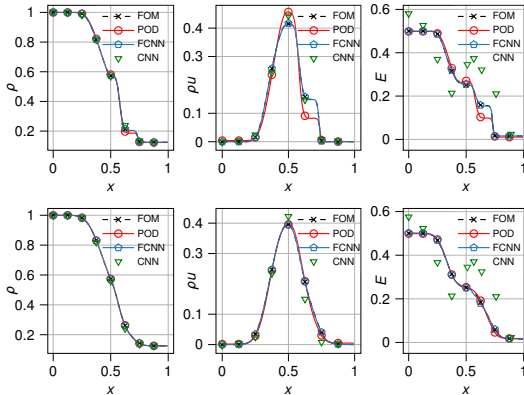


Figure: Comparison of moments of f and \tilde{f} , H top and R bottom.

– POD:

- **H** - undercuts shockwave in ρ , ρu and E
++ - ρu exceeds tail of rarefaction wave
- **R** - only small deviations at shockwave for ρu and E

– FCNN:

- **R** - severe deviation at transition: tail of rarefaction wave \rightarrow shockfront

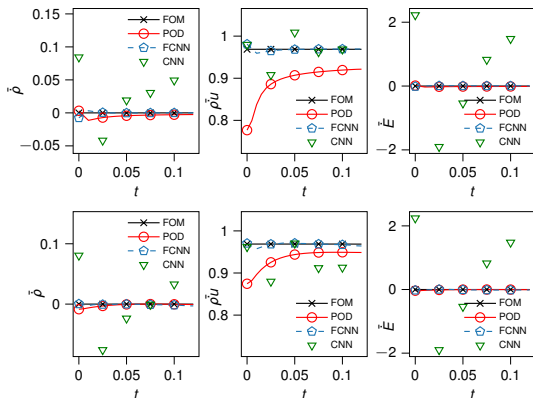
– CNN:

- **R** is copy of **H**





Physical consistency



– POD:

- H & R - mass & + mass
- H & R +++ momentum
- H & R energy

– FCNN:

- H & R mass
- H & R - momentum & + momentum
- H & R energy

– CNN:

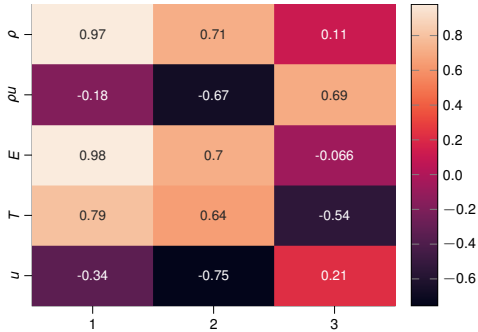
- No conservation

Figure: Conservation properties of f and \tilde{f} , \mathbf{H} top and \mathbf{R} bottom.





Interpretability



– 1:

- E : 0.98 & ρ : 0.97

– 2:

- u : -0.75 & ρ : 0.71

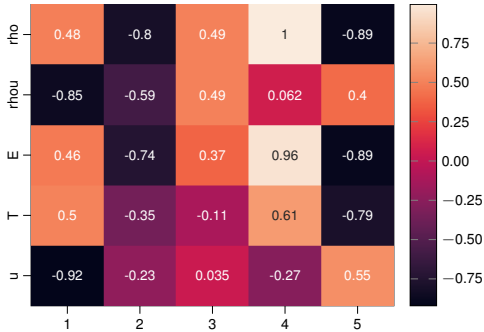
– 3:

- ρu : 0.69 & T : -0.54

Figure: Pearson correlation between of macroscopic quantities and intrinsic variables for **H**



Interpretability



- 1:
 - u : -0.92 & ρu : -0.85
- 2:
 - ρ : -0.8 & E : -0.74
- 3:
 - ρ : 0.49 & ρu : 0.49
- 4:
 - ρ : 1 & E : 0.96
- 5:
 - ρ : -0.89 & T : -0.89

Figure: Pearson correlation between of macroscopic quantities and intrinsic variables for H



Interpolation

Table: Validation and metric results for the interpolation task with 13, 9, 7 and 5 time steps.

n	Δt^*	Validation error		L_2 -error		L_2 -error	
		H^*	R^*	H^*	R^*	\bar{H}^*	\bar{R}^*
13	0.01s	2.5×10^{-8}	2.9×10^{-7}	0.0018	0.0054	0.0036	0.0058
9	0.015s	2.9×10^{-8}	9.5×10^{-8}	0.0017	0.0038	0.0067	0.0056
7	0.02s	2.5×10^{-8}	1.6×10^{-7}	0.0019	0.0042	0.0101	0.0073
5	0.025s	1.7×10^{-7}	1.6×10^{-7}	0.0039	0.0051	0.0367	0.0138

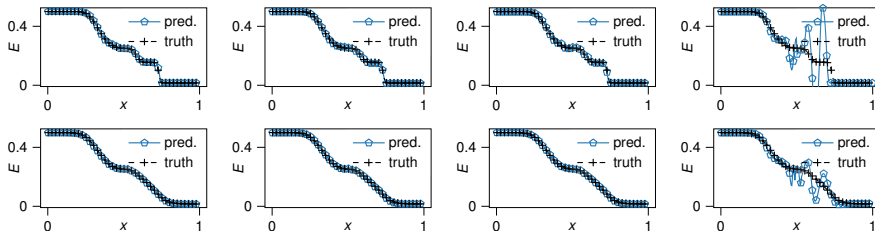


Figure: Energy after interpolation using the FCNN trained on 13,9,7 and 5 time steps, H top and R bottom.





Table of Contents

Introduction

The BGK-Model

Sod's shock tube

Model Order Reduction

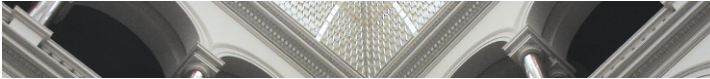
Proper Orthogonal Decomposition (POD)

Neural Networks

Results

Discussion





ToDo

- **ToDo** schreiben
- **ToDo** abarbeiten



