Accelerated thermomechanical modeling of additive manufacturing using laser-based powder bed fusion

Juha Jeronen, Tero Tuovinen and Matti Kurki
JAMK University of Applied Sciences, Jyväskylä, Finland

This research was funded by the ERDF grants
A77069 iADDVA (Adding Value by Creative Industry Platform)
A77973 coADDVA (ADDing VAlue by Computing in Manufacturing)

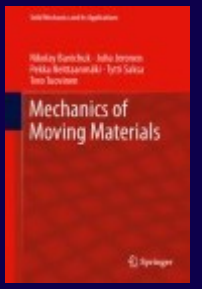
jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

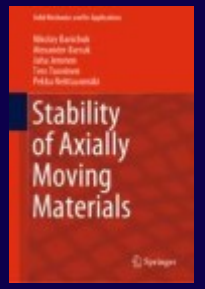
CSAI 2023 ECCOMAS Thematic Conference, Trondheim, Norway, 28 August 2023

My perspective

$$\psi = \frac{1}{\sqrt{3}}|(\mathrm{IT}, \mathrm{CS})\rangle + \frac{1}{\sqrt{3}}|\mathrm{M}\rangle + \frac{1}{\sqrt{3}}|(\mathrm{P}, \mathrm{ES})\rangle$$

* appropriate normalization, assuming orthogonal states.





- where
  IT = information technology, CS = computer science
  ...with some software engineering
  M = mathematics
  P = physics, ES = engineering sciences
- Software engineer 1999–2007, Jutel Oy, Oulu, Finland
- MSc 2008, University of Oulu, mathematics
- PhD 2011, University of Jyväskylä (JYU), information technology
- *Postdoc researcher* 2012–2018, mostly at JYU; at Tampere University of Technology (TUT) 2017–2018;
  Calculation engineer at Global Boiler Works, Oulu, Finland (2019–, currently on leave);
  University researcher at Tampere University (TUNI) 2020–2021;
  Project engineer at JAMK University of Applied Sciences, Jyväskylä, 2021–2022/06;
  Researcher at JAMK 2022/07–
- Research topics: **axially moving materials** [1], energy harvesting [2], thermomechanics of steel [3]
- Programming background: C [4], C++, Perl, Java, MATLAB, Racket, Python (2012– [5] [6]), Julia (2020– [3])
- Metaprogramming [7] [8] [2]

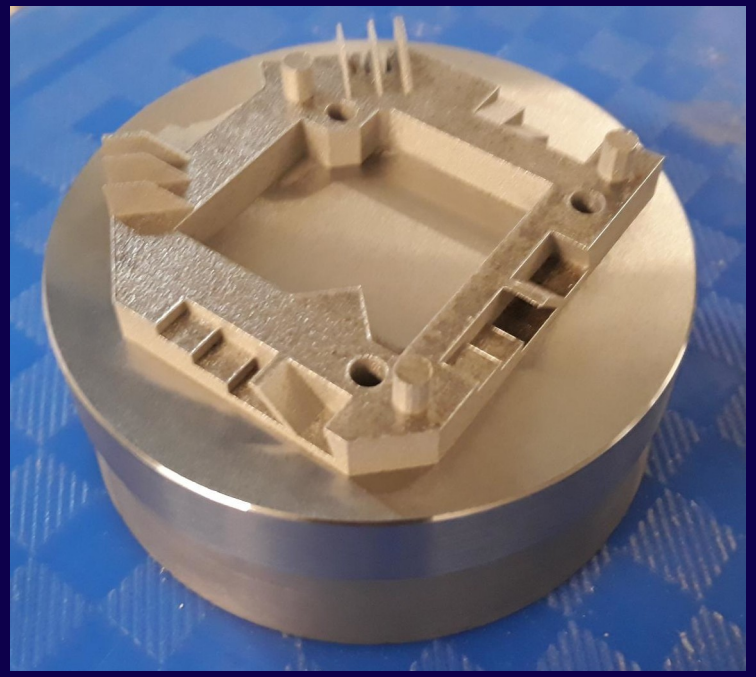
[1] Major contributions to SMIA vol. 207 and 259. Springer.
[2] https://github.com/TUTElectromechanics/mm-codegen
[3] https://github.com/JuliaFEM/Materials.jl
[4] https://code.videolan.org/videolan/vlc/-/tree/master/modules/video_filter/deinterlace IVTC and Phosphor modes
[5] https://github.com/Technologicat/pydgq – time-discontinuous Galerkin for ODE systems
[6] https://github.com/Technologicat/python-wlsqm – meshless interpolator and differentiator
[7] https://github.com/Technologicat/mcpyrate – advanced macro expander and language lab
[8] https://github.com/Technologicat/unpythonic – kitchen-sink language extension

** Silly gamer alias due to GitHub's one account per person policy.*

jamk

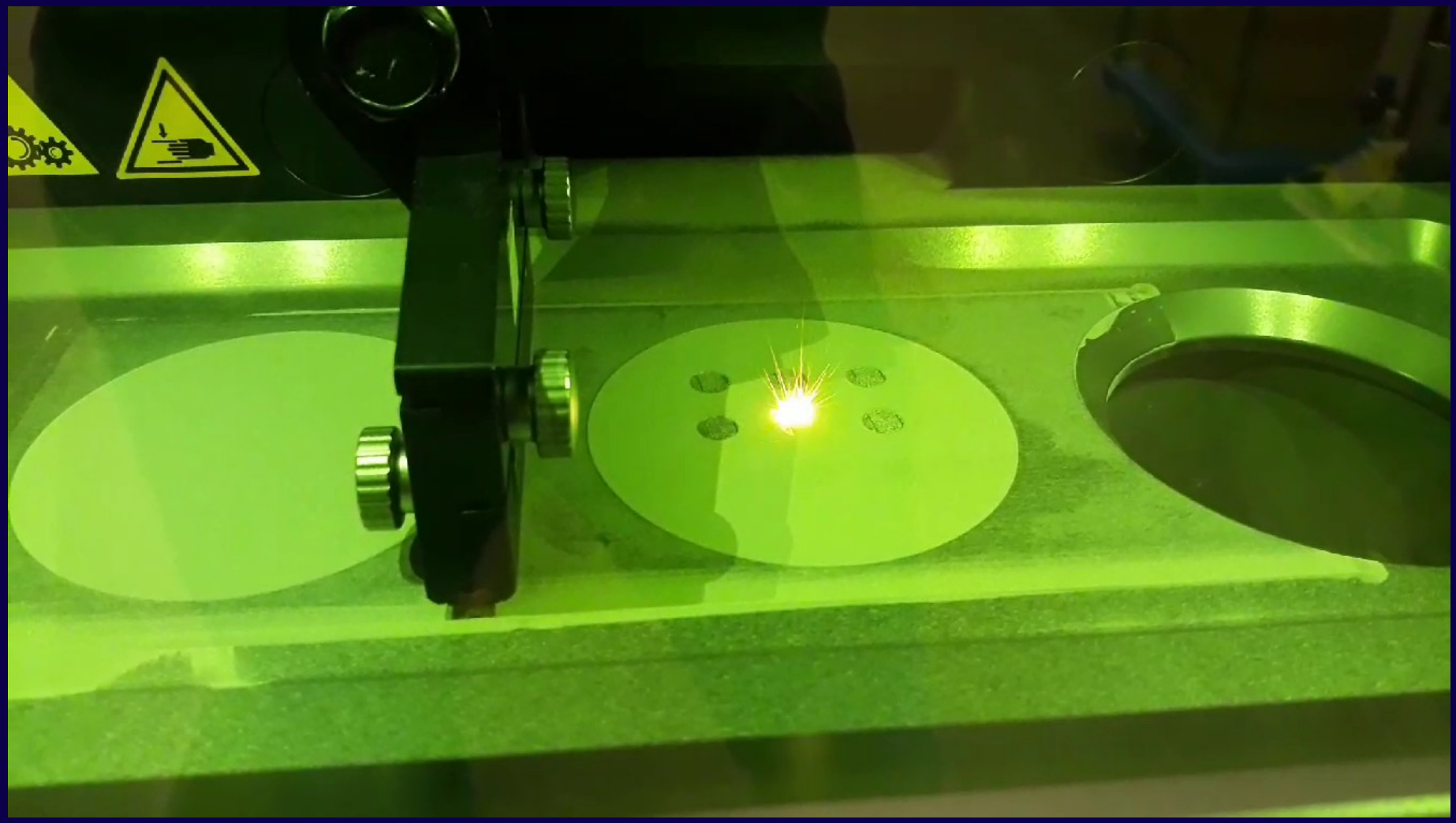
Part 1 – PDE model for additive manufacturing for metals

- Additive manufacturing of metals is a relatively new technology (2010s).
- **Advantages** of additive manufacturing:
  - Can manufacture shapes not achievable by other methods.
  - Can build small batches.
    → Mass customization for long-tail items.
  - **Green** manufacturing: helps reduce manufacturing waste.
- Based on **melting** or **sintering** (partially melting) the metal.
- Several varieties exist, for example:
  - Powder Bed Fusion (**PBF**) vs. Directed Energy Deposition (DED)
  - Laser based vs. electron beam based
- See e.g. the book by Milewski (2017).
- JAMK has a *Trumpf TruPrint 1000* laser-based powder bed fusion (**L-PBF**) 3D printer for metals.



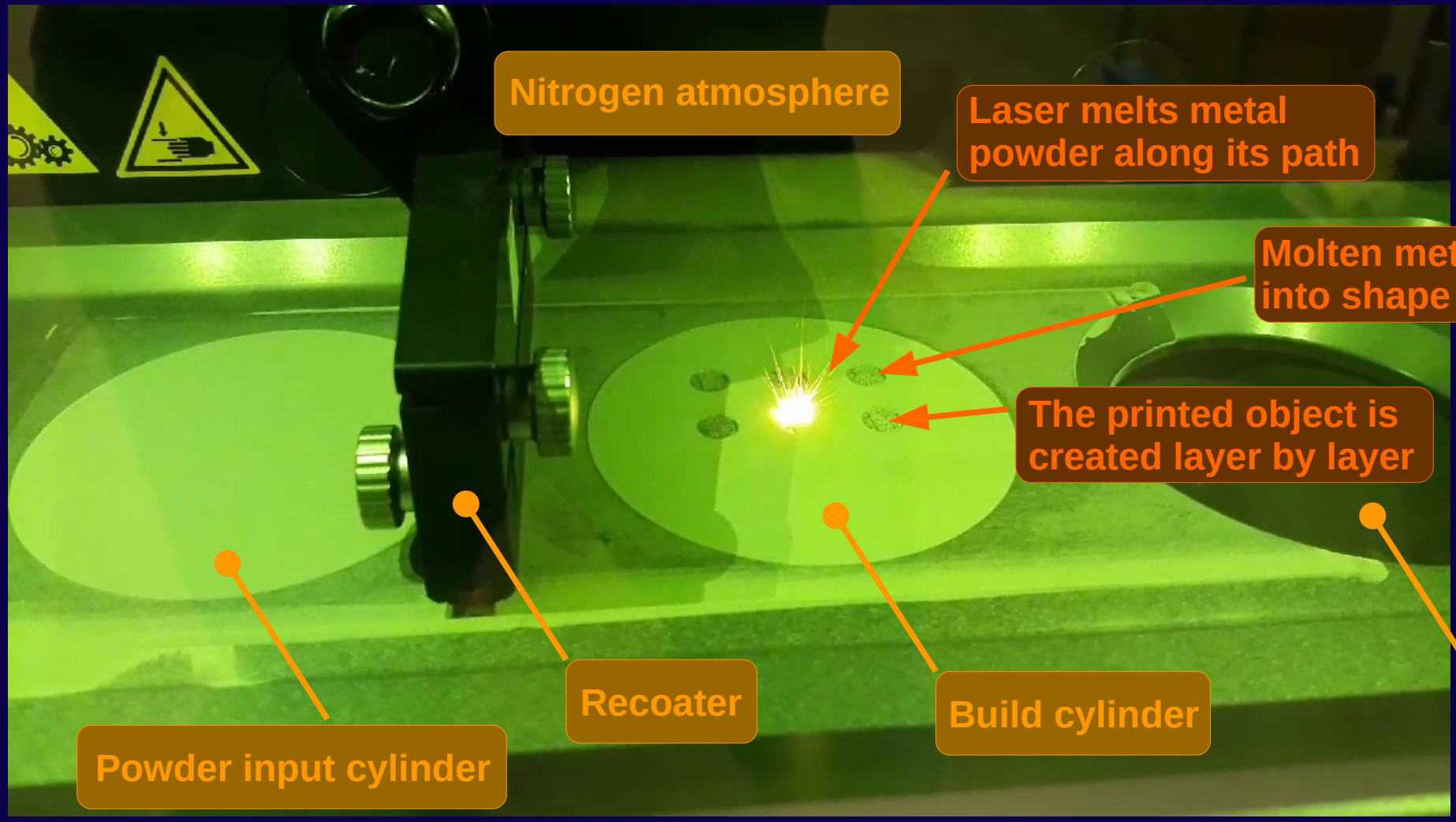
jamk

Laser-based Powder Bed Fusion (L-PBF)



jamk

Laser-based Powder Bed Fusion (L-PBF)

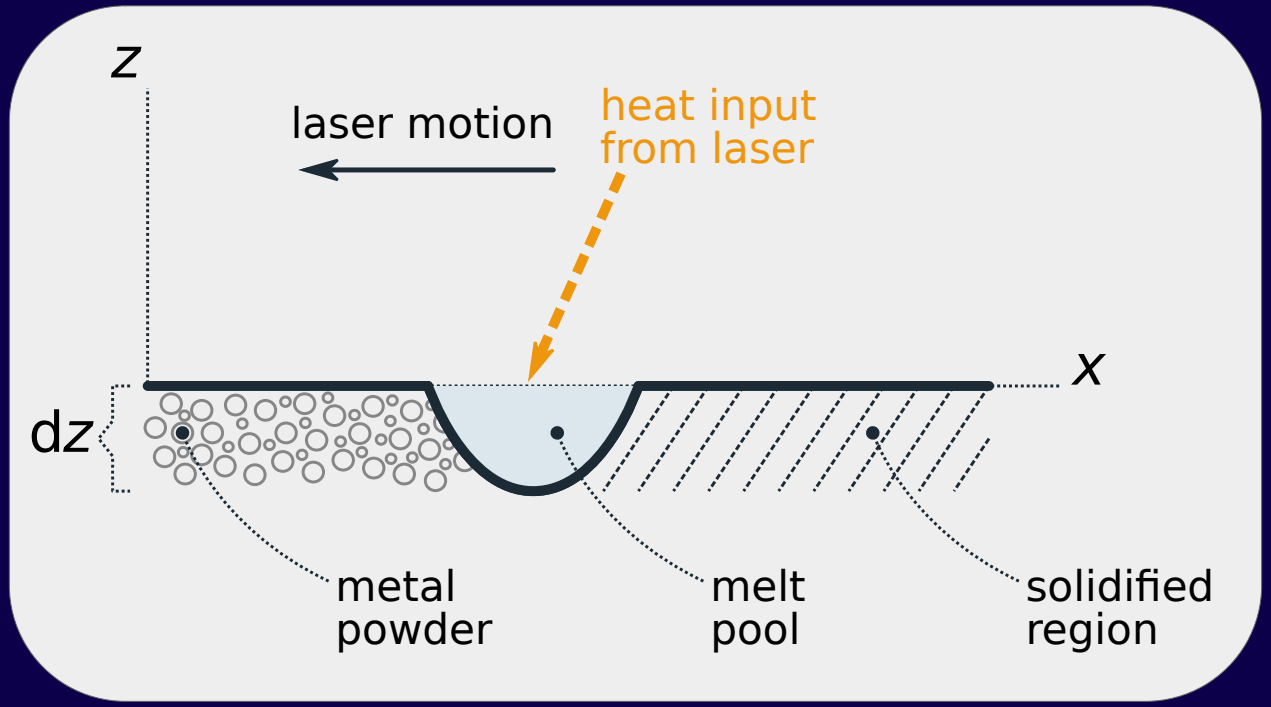


jamk

Consider printing metal in a straight line...

Why?

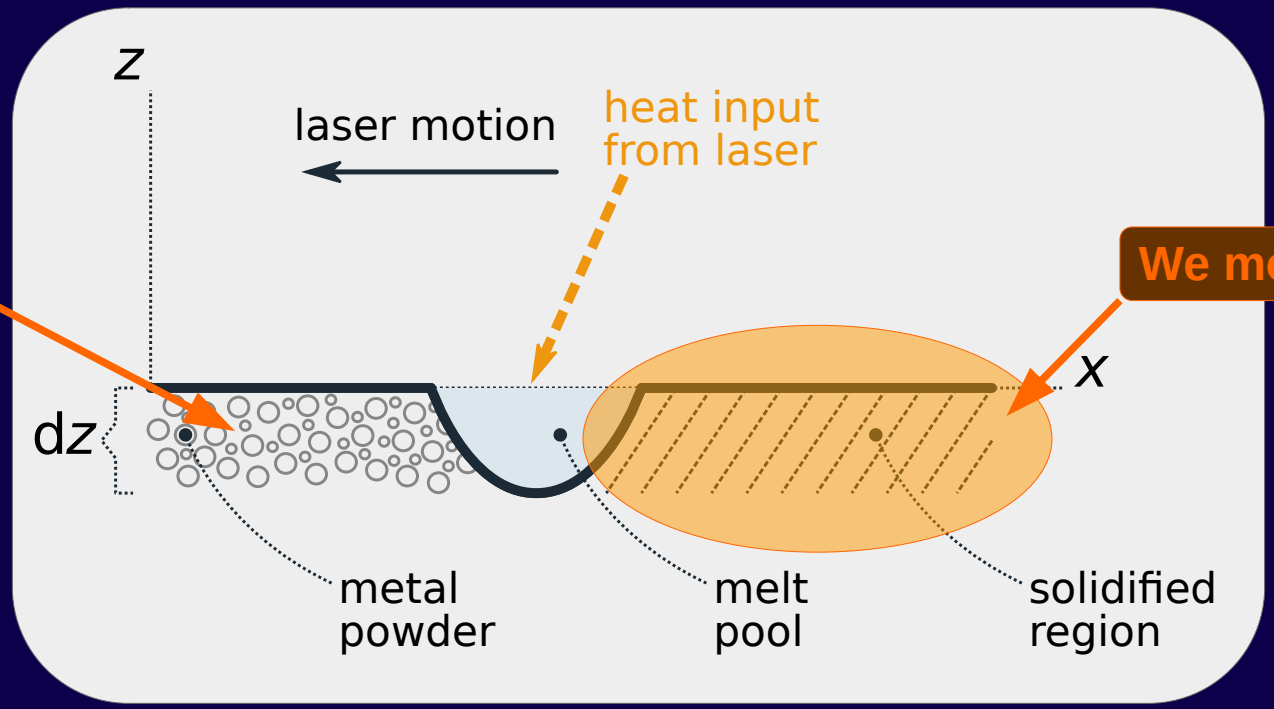
- **Simplify** (to the extreme) **to understand the fundamentals** before tackling more complex cases.
- **Explore** the space of models **systematically**, as afforded by simulation.



One layer of a straight line of metal being printed using an L-PBF printer.

Consider printing metal in a straight line...

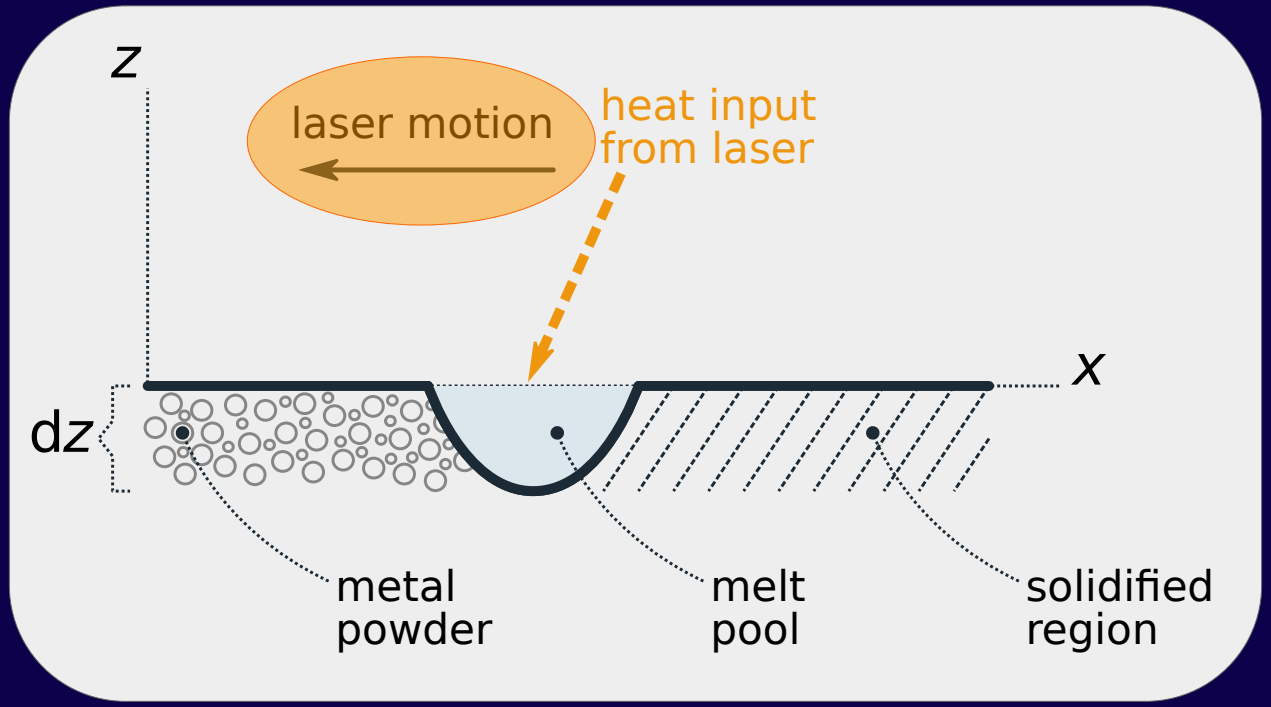
Grains of various sizes
(largest ≈ 20 µm)



Heat transfer away from the solidified region is a limiting factor for the manufacturing process.

jamk

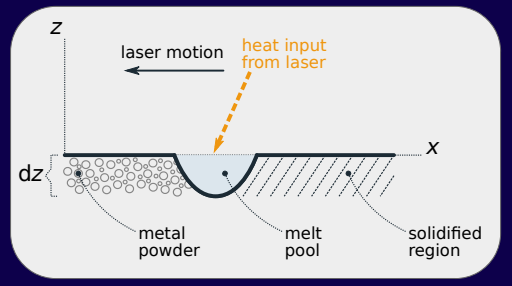
Consider printing metal in a straight line...

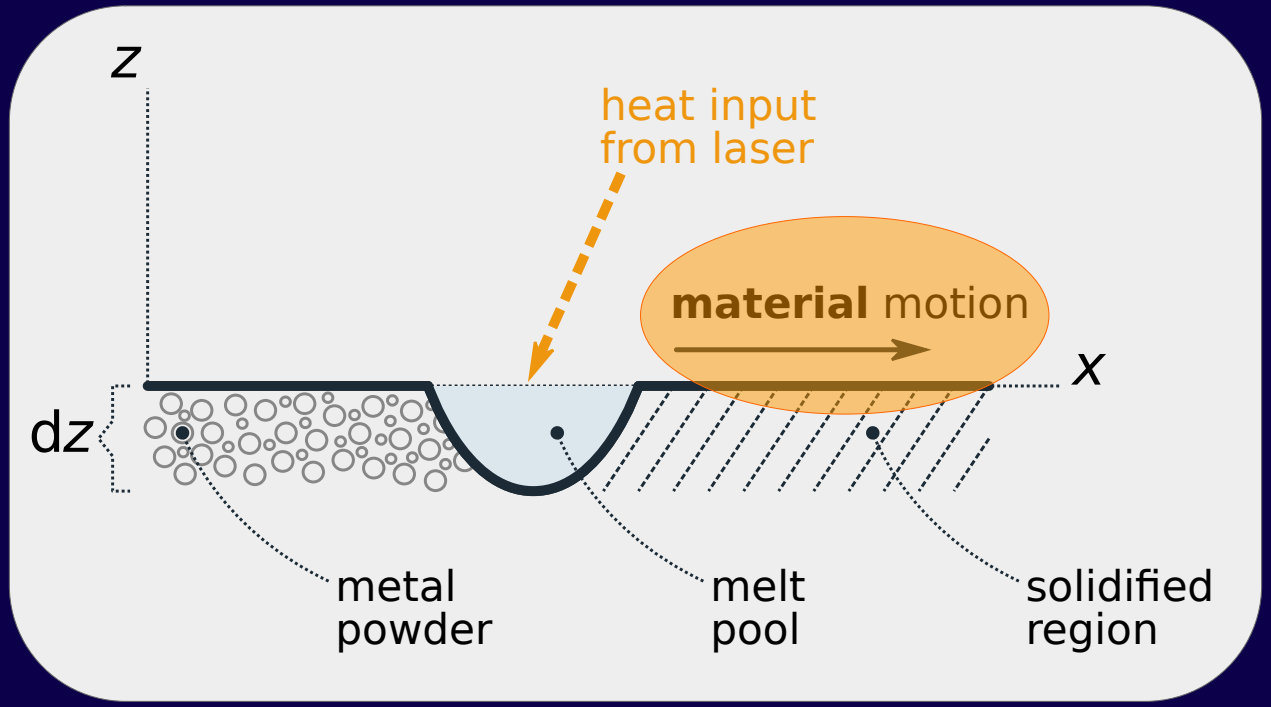


In this setup, **the laser moves toward the left**.

jamk

Consider printing metal in a straight line...

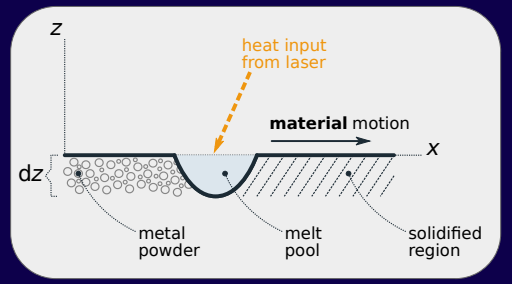




But if we follow the focus spot of the laser, then by Galilean relativity, **the metal moves toward the right**.

jamk

Why and how to follow the laser focus spot?

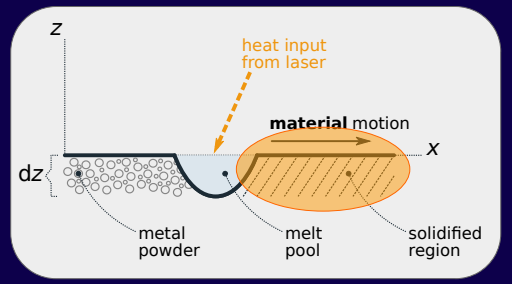


- The **change of perspective** may allow us to see features we would otherwise miss.
  - This allows to easily look at **what happens around the melt pool** as the laser moves.
- Often a **non-stationary steady state** exists for process models like this.
  - To an observer looking (staring intently) at the build cylinder, all quantities vary in time. This requires a transient analysis, and modeling many situational specifics.
  - To an observer following the focus spot of the laser, some simple configurations, such as the one considered here, admit a state where all quantities stay **constant in time** (as seen by that observer) even as the printing process proceeds.
- **Simplicity**: fewer parameters; expose the core essence of the process.
  - Can use a 2D model (add depth, *z* direction) to account for the presence of already printed layers (printing a thin fin).
- **How**: **axially moving materials**
  - Essentially, an Eulerian perspective to solids. Started by Skutch (1897).
  - Applied widely in the process industry, especially in papermaking.
  - For an introduction, see e.g. Banichuk et al. (2020, chapter 5), and our upcoming book, *Fundamental Mathematical Modeling of Additive Manufacturing*.

* *We have here an atypical application of axially moving materials. Typically, in the process industry, the solid physically flows through the machine.*

jamk

Construction of the model



- **Modularize** to facilitate agile model space exploration.
- **Model components**: **balance laws**, **kinematic relation**, **constitutive model**
  - For simplicity, we model the **solidified region** only.

** Thermodynamic conditions apply.*

- The four **fundamental balance laws** of continuum mechanics:
  - **Linear momentum** ⇒ Equation of motion (≈ **mechanical model**)

$$\rho \frac{d\boldsymbol{V}}{dt} - \nabla \cdot \boldsymbol{\sigma}^T = \rho \boldsymbol{b}$$

  - **Internal energy** (enthalpy) ⇒ Heat equation for axially moving material (≈ **thermal model**)

$$\rho \frac{de}{dt} = \nabla \cdot \boldsymbol{q} + \boldsymbol{\sigma} : \nabla \boldsymbol{V} + \rho h$$

  - **Angular momentum** ⇒ Cauchy stress tensor **σ** is symmetric (as usual).
  - **Mass** ⇒ No constraints, because trivial constant-velocity axial flow through domain, and in the small-deformation regime, density ρ is approximately constant.
- **Kinematic relation**: **small deformations**, for simplicity.

$$\boldsymbol{\varepsilon} = \frac{1}{2}[\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T]$$

CAUTION: *displacement* ***w.r.t. constant-velocity axial translation***.

jamk

Construction of the model

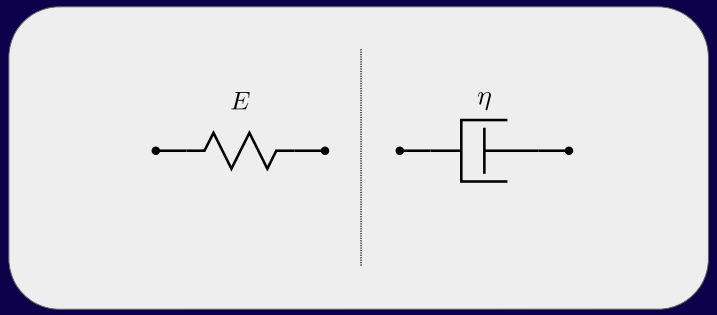
*Only certain kinds of **mechanical** strain create stress.*

1D linear constitutive law

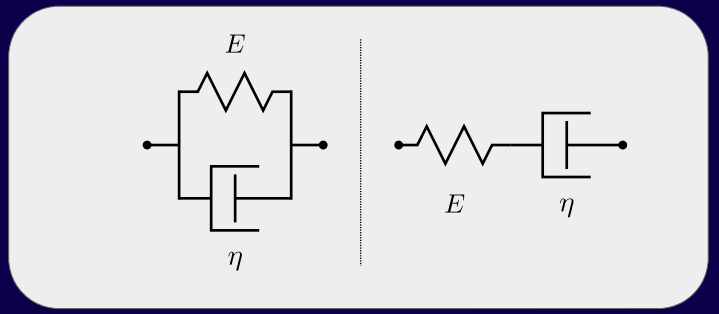
$$\left[a_0 + a_1 \frac{d}{dt}\right]\sigma = \left[b_0 + b_1 \frac{d}{dt}\right]\varepsilon_{mech}$$

***CAUTION**: **material** derivative.*

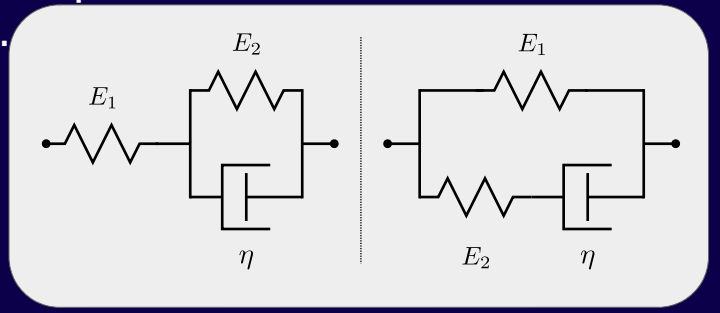
- **Constitutive relation** (stress-strain relation)
  - Material parameters depend on the absolute temperature T
  - We restrict to the class of models described by a **linear first-order differential equation**
  - This class includes, notably:
    - **Linear elastic** and **linear viscous** models
    - The **Kelvin-Voigt** and **Maxwell** models
    - The **standard linear solid** (**SLS**).
- We allow any elastic and viscous symmetry groups.
  - **Isotropic**
  - **Transverse isotropic** (layered)
  - **Orthotropic** (layered, plus along laser path vs. perpendicular to it)



Linear elastic solid Linear viscous fluid



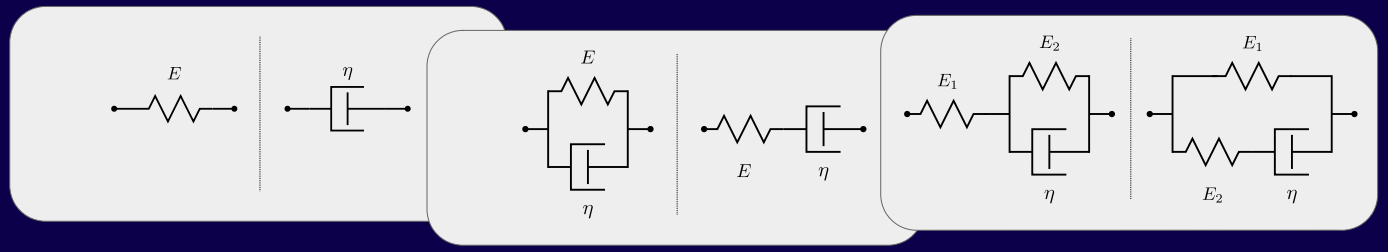
Kelvin–Voigt solid Maxwell fluid



“Poynting–Thomson” SLS “Zener” SLS

jamk

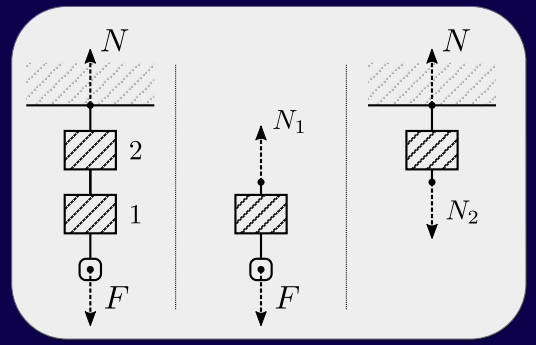
Construction of the model

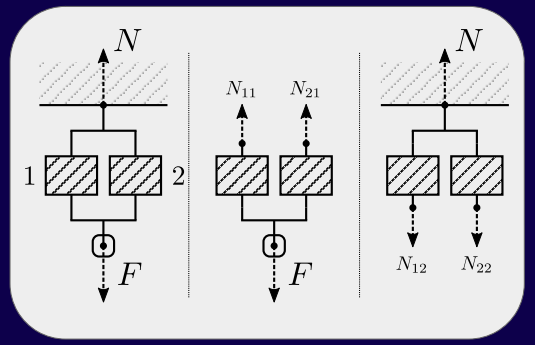


- **Multidimensional networks** are constructed by abstract analogues of the well-known 1D **composition rules**:

$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_2,\ \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2$

Serial connection





$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_1 = \boldsymbol{\varepsilon}_2,\ \boldsymbol{\sigma} = \boldsymbol{\sigma}_1 + \boldsymbol{\sigma}_2$

Parallel connection

- Result (**K**: stiffness, **C**: compliance, $\mathbf{I}_s$ symmetric rank-4 identity; subscripts *E*, *η* indicate diagram components):

$\boldsymbol{I}_s : \boldsymbol{\sigma} = \boldsymbol{K} : \boldsymbol{\varepsilon}_{mech}$ linear elastic solid

$\boldsymbol{I}_s : \boldsymbol{\sigma} = \boldsymbol{K} \frac{d}{dt} : \boldsymbol{\varepsilon}_{mech}$ linear viscous fluid

Starting from these and the composition rules

$$[\boldsymbol{A}_0 + \boldsymbol{A}_1 \frac{d}{dt}] : \boldsymbol{\sigma} = [\boldsymbol{B}_0 + \boldsymbol{B}_1 \frac{d}{dt}] : \boldsymbol{\varepsilon}_{mech}$$

2D and 3D linear constitutive law

Each network component is allowed to have any symmetry group, separately.

$$\boldsymbol{I}_s : \boldsymbol{\sigma} = [\boldsymbol{K}_E + \boldsymbol{K}_\eta \frac{d}{dt}] : \boldsymbol{\varepsilon}_{mech}$$

Kelvin–Voigt solid

$$[(\frac{\partial \boldsymbol{C}_E}{\partial T}\frac{dT}{dt} + \boldsymbol{C}_\eta) + \boldsymbol{C}_E \frac{d}{dt}] : \boldsymbol{\sigma} = [\boldsymbol{I}_s \frac{d}{dt}] : \boldsymbol{\varepsilon}_{mech}$$

Maxwell fluid

$$[(\boldsymbol{I}_s + \boldsymbol{K}_{E_2} : \boldsymbol{C}_{E_1} + \boldsymbol{K}_\eta : [\frac{\partial \boldsymbol{C}_{E_1}}{\partial T}\frac{dT}{dt}]) + \boldsymbol{K}_\eta : \boldsymbol{C}_{E_1} \frac{d}{dt}] : \boldsymbol{\sigma} = [\boldsymbol{K}_{E_2} + \boldsymbol{K}_\eta \frac{d}{dt}] : \boldsymbol{\varepsilon}_{mech}$$

"Poynting–Thomson" SLS (left side of diagram)

$$\boldsymbol{A}_0 := \frac{\partial \boldsymbol{C}_{E_2}}{\partial T}\frac{dT}{dt} + \boldsymbol{C}_\eta\ ,\ [\boldsymbol{A}_0 + \boldsymbol{C}_{E_2}\frac{d}{dt}] : \boldsymbol{\sigma} = [(\boldsymbol{A}_0 : \boldsymbol{K}_{E_1} + \boldsymbol{C}_{E_2} : \frac{\partial \boldsymbol{K}_{E_1}}{\partial T}\frac{dT}{dt}) + (\boldsymbol{I}_s + \boldsymbol{C}_{E_2} : \boldsymbol{K}_{E_1})\frac{d}{dt}] : \boldsymbol{\varepsilon}_{mech}$$

"Zener" SLS (right side of diagram)

jamk

Construction of the model

- **Thermomechanical** coupling
  - **Total strain**:

$$\varepsilon = \varepsilon_{mech} + \varepsilon_{th} \quad \Rightarrow \quad \varepsilon_{mech} = \varepsilon - \varepsilon_{th}$$

$$\left[a_0 + a_1 \frac{d}{dt}\right]\sigma = \left[b_0 + b_1 \frac{d}{dt}\right]\varepsilon_{mech}$$

where the **thermal strain** describes thermal expansion:

$$\varepsilon_{th} = \alpha\left[T - T_0\right]$$

where **α** is the **thermal expansion tensor** (which is part of the **constitutive** model).

- We allow any thermal symmetry group, separately from the elastic and viscous symmetry groups.
  - **Isotropic**
  - **Transverse isotropic**
  - **Orthotropic**

- For example, for a **thermally isotropic** material,

$$\boldsymbol{\alpha} = \alpha\,\mathbf{1}$$

where α is the coefficient of thermal expansion (unit 1/K), and **1** is the rank-2 identity tensor.

jamk

Governing equations – general strong form, for analytical treatment

Material: axially moving thermoviscoelastic Kelvin–Voigt, with arbitrary symmetry groups.

- **Linear momentum** balance, Eulerian view:

$$(\star) \qquad \rho\left[\frac{\partial^2 \boldsymbol{u}}{\partial t^2} + 2(\boldsymbol{v}\cdot\nabla)\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{v}\cdot\nabla)(\boldsymbol{v}\cdot\nabla)\boldsymbol{u}\right] - \nabla\cdot transpose(\boldsymbol{\sigma}) = \rho\boldsymbol{b}$$

where

$$(\star\star) \qquad \begin{aligned} \boldsymbol{\sigma} &= \boldsymbol{K}_E : symm(\nabla\boldsymbol{u}) - [\boldsymbol{K}_E : \boldsymbol{\alpha}][T - T_0] \\ &+ \boldsymbol{K}_\eta : \left(\frac{\partial}{\partial t} + (\bar{\boldsymbol{V}}\cdot\nabla)\right) symm(\nabla\boldsymbol{u}) \\ &- \left[\boldsymbol{K}_\eta : \left[\frac{\partial \boldsymbol{\alpha}}{\partial T}[T - T_0] + \boldsymbol{\alpha}\right]\right]\left(\frac{\partial}{\partial t} + (\bar{\boldsymbol{V}}\cdot\nabla)\right) T \end{aligned}$$

- **Internal energy** balance, Eulerian view:

$$\rho\frac{\partial c}{\partial T}T\left[\frac{\partial T}{\partial t} + (\bar{\boldsymbol{V}}\cdot\nabla)T\right] + \rho c\left[\frac{\partial T}{\partial t} + (\bar{\boldsymbol{V}}\cdot\nabla)T\right] - \nabla\cdot(\boldsymbol{k}\cdot\nabla T) = \boldsymbol{\sigma} : \nabla\bar{\boldsymbol{V}} + \rho h$$

If $\partial c/\partial T \neq 0$, this is **nonlinear**.

In these equations, $\bar{V}$ is the first-order approximate material parcel velocity as measured in the frame that follows the laser focus spot:

$$\bar{\boldsymbol{V}} := \boldsymbol{v} + \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{u}$$

To a first approximation, we may replace it by the laser velocity $\boldsymbol{v}$, which is a constant.

(Actually, we have made that replacement in the inertia terms; these result from approximating the true material parcel acceleration d**V**/d*t*, as measured in the frame that follows the laser focus spot.)

Inserting ($\star\star$) into ($\star$), final strong **primal formulation** in index notation:

$$\begin{aligned} &\rho\partial_t^2 u_i + 2\rho v_m[\partial_m\partial_t u_i] + \rho[v_m\partial_m][v_n\partial_n]u_i \\ &-\left[[\partial_m T][\partial_T(\boldsymbol{K}_E)_{milk}]\partial_l u_k + (\boldsymbol{K}_E)_{imlk}[\partial_m\partial_l u_k]\right] \\ &+[\partial_m T][\partial_T(\boldsymbol{K}_E)_{mikl}]\alpha_{kl}[T - T_0] + \left[[T - T_0][\partial_T\alpha_{kl}] + \alpha_{kl}\right](\boldsymbol{K}_E)_{klim}[\partial_m T] \\ &-\left[[\partial_m T][\partial_T(\boldsymbol{K}_\eta)_{milk}](\partial_t + v_n\partial_n)\partial_l u_k + [\partial_m(\partial_t + v_n\partial_n)\partial_k u_l](\boldsymbol{K}_\eta)_{klim}\right] \\ &+[\partial_m T][\partial_T(\boldsymbol{K}_\eta)_{mikl}]\left[[\partial_T\alpha_{kl}][T - T_0] + \alpha_{kl}\right](\partial_t T + \bar{V}_n\partial_n T) \\ &+\left[[T - T_0][\partial_T^2\alpha_{kl}] + 2[\partial_T\alpha_{kl}]\right](\boldsymbol{K}_\eta)_{klim}[\partial_m T](\partial_t T + \bar{V}_n\partial_n T) \\ &+\left[[T - T_0][\partial_T\alpha_{kl}] + \alpha_{kl}\right](\boldsymbol{K}_\eta)_{klim}\left[[\partial_m\partial_t T] + [\partial_m\bar{V}_n][\partial_n T] + [\bar{V}_n\partial_n][\partial_m T]\right] = \rho b_i \end{aligned}$$

jamk

Governing equations – 1D

Material: axially moving thermoviscoelastic Kelvin–Voigt, 1D.

- **Solve a simple variant first**, to obtain a point of reference for more complex solutions.
- Restrict the previous equations to one space dimension.

- With the material parameters allowed to be arbitrary functions of the absolute temperature T, the **equation of motion** in 1D is:

$$\begin{aligned}
&\rho\frac{\partial^2 u}{\partial t^2}+2\rho v\frac{\partial^2 u}{\partial x\,\partial t}+\rho v^2\frac{\partial^2 u}{\partial x^2} && \text{inertia}\\
&-\left[\frac{\partial T}{\partial x}\frac{\partial E}{\partial T}\frac{\partial u}{\partial x}+E\frac{\partial^2 u}{\partial x^2}\right] && \text{elastic response}\\
&+\left[\frac{\partial T}{\partial x}\frac{\partial E}{\partial T}\alpha[T-T_0]\right]+\left[[T-T_0]\frac{\partial \alpha}{\partial T}+\alpha\right]E\frac{\partial T}{\partial x} && \text{thermoelastic response}\\
&-\left[\frac{\partial T}{\partial x}\frac{\partial \eta}{\partial x}\left(\frac{\partial^2 u}{\partial x\,\partial t}+v\frac{\partial^2 u}{\partial x^2}\right)+\left(\frac{\partial^3 u}{\partial x^2\,\partial t}+v\frac{\partial^3 u}{\partial x^3}\right)\eta\right] && \text{viscous response}\\
&+\left[\frac{\partial T}{\partial x}\frac{\partial \eta}{\partial x}\left[\frac{\partial \alpha}{\partial T}[T-T_0]+\alpha\right]\left(\frac{\partial T}{\partial t}+\bar{V}\frac{\partial T}{\partial x}\right)\right] && \text{thermoviscous response}\\
&+\left[[T-T_0]\frac{\partial^2 \alpha}{\partial T^2}+2\frac{\partial \alpha}{\partial T}\right]\eta\frac{\partial T}{\partial x}\left(\frac{\partial T}{\partial t}+\bar{V}\frac{\partial T}{\partial x}\right) && \text{thermoviscous response}\\
&+\left[[T-T_0]\frac{\partial \alpha}{\partial T}+\alpha\right]\eta\left[\frac{\partial^2 T}{\partial x\,\partial t}+\frac{\partial \bar{V}}{\partial x}\frac{\partial T}{\partial x}+\bar{V}\frac{\partial^2 T}{\partial x^2}\right]=\rho b && \text{thermoviscous response}
\end{aligned}$$

- If $\partial c/\partial T \neq 0$, we obtain this **nonlinear heat equation**:

$$\rho\frac{\partial c}{\partial T}T\left[\frac{\partial T}{\partial t}+\bar{V}\frac{\partial T}{\partial x}\right]+\rho c\left[\frac{\partial T}{\partial t}+\bar{V}\frac{\partial T}{\partial x}\right]-\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)=\sigma\frac{\partial \bar{V}}{\partial x}+\rho h$$

jamk

Governing equations – 1D, constant material parameters

Material: axially moving thermoviscoelastic Kelvin–Voigt, 1D.

- With **constant material parameters**, the 1D governing equations become:

$$\rho\frac{\partial^2 u}{\partial t^2}+2\rho v\frac{\partial^2 u}{\partial x\,\partial t}+[\rho v^2-E]\frac{\partial^2 u}{\partial x^2}-\left(\frac{\partial^3 u}{\partial x^2\,\partial t}+v\frac{\partial^3 u}{\partial x^3}\right)\eta=\rho b-\alpha\left[E\frac{\partial T}{\partial x}+\eta\left[\frac{\partial^2 T}{\partial x\,\partial t}+v\frac{\partial^2 T}{\partial x^2}\right]\right]$$

Effective body force due to thermal shrinkage.

$$\rho c\left[\frac{\partial T}{\partial t}+v\frac{\partial T}{\partial x}\right]-k\frac{\partial^2 T}{\partial x^2}=\rho h$$

- **Third order** in u in the general case, where $\eta v \neq 0$.
  - Axially moving viscous materials typically produce an equation of motion one order higher than in the corresponding classical (not axially moving) case. (E.g. Kurki et al. 2016; Banichuk et al., 2020, chapter 5.)
- In the 1D model, must separately model heat escaping through exposed surface (not a domain boundary!). Heat sink, **Newton's law of cooling**:

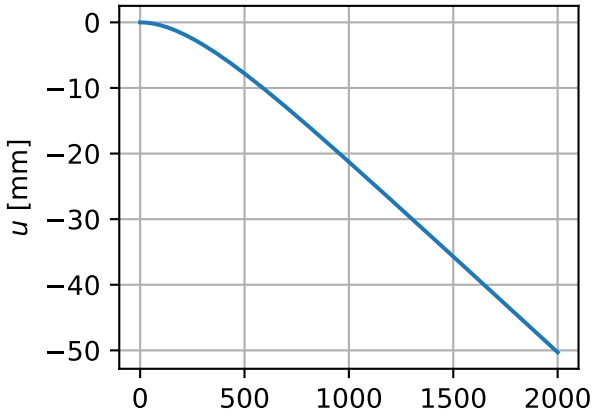
$$h(x)=-r\left[T-T_{ext}\right]$$

- Consider the steady state ($\partial/\partial t \rightarrow 0$):
  - For u, introduce auxiliary variable $\varepsilon = \partial u/\partial x$. Then integrate once. This gets rid of two differentiations. Obtain **linear first-order ordinary differential equation**, which has an analytical solution even with arbitrary function coefficients (see e.g. the handbook by Polyanin and Zaitsev, 2003).
  - For T, analytically solvable **linear second-order ordinary differential equation**.
- One-way coupling: T can be solved first.
- Inserting $T(x)$ into the linear momentum balance produces an explicit analytical solution for the displacement $u(x)$.

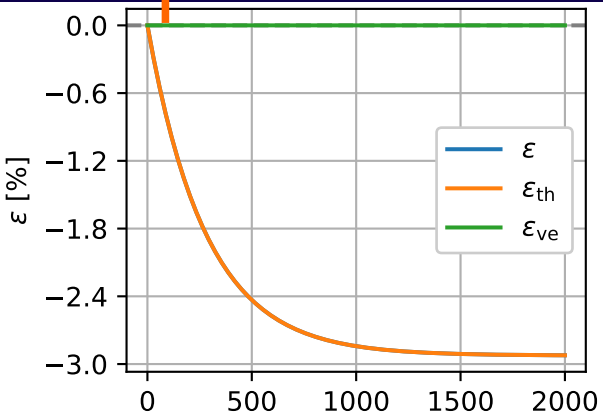
1D results, 316L steel

JJ, TT, MK. One-Dimensional Thermomechanical Model for Additive Manufacturing Using Laser-Based Powder Bed Fusion. Computation 2022, 10(6), 83, doi: 10.3390/computation10060083

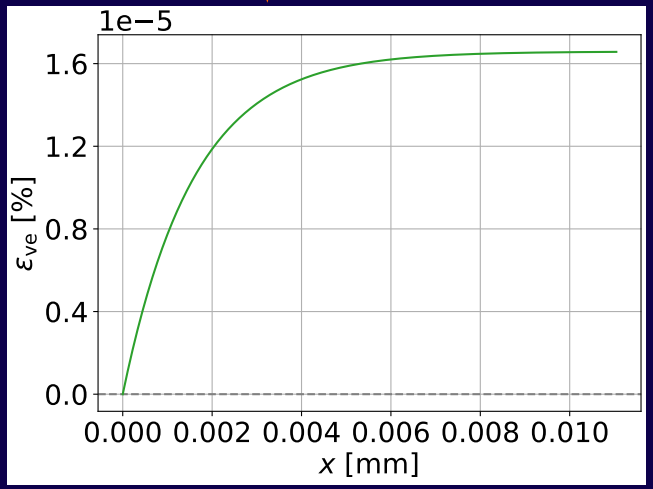
Axial displacement w.r.t. constant-velocity translation



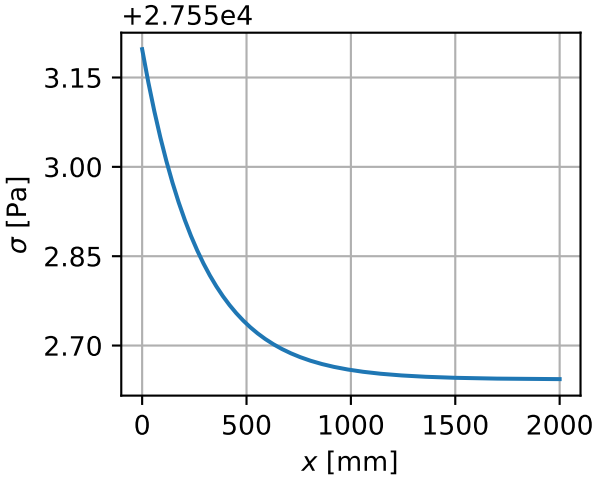
Axial strain



total = thermal + viscoelastic

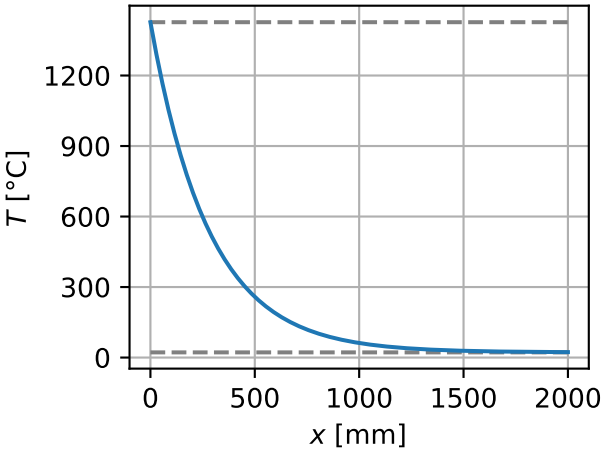


Viscoelastic strain. Note the scales.



Axial stress

solidus temp.



Temperature

room temp.

Almost all of the **total** strain is thermal shrinkage, but only the very small **viscoelastic** contribution produces stress.

Thermal shrinkage is in principle **reversible**, but the reference temperature is the solidus temperature of the steel. The printed object will never again be heated back to that temperature, making the thermal shrinkage **effectively permanent**.

Laser scan speed v = 50 mm/s.

x is the distance from the edge of the solidified region.

jamk

2D solution strategy – mechanical subproblem

Material: axially moving viscoelastic Kelvin–Voigt, isotropic.

- Solid material; 2nd order in time. Take a velocity field as an auxiliary unknown to reduce to 1st order system.
- Classically, one extra order in space due to axial motion of a viscous solid: $\mathrm{d}/\mathrm{d}t \rightarrow \partial/\partial t + \mathbf{v}\cdot\nabla$
  - A steady state for this mechanical model was computed in Kurki et al. (2016) using C^1 continuous FEM.

- Choose the **co-moving rate** $\mathbf{V} = \mathrm{d}\mathbf{u}/\mathrm{d}t$ (**not** the Eulerian rate $\partial\mathbf{u}/\partial t$) as the auxiliary variable.
  - This synergizes with the form of the constitutive law, **avoiding the increase in the spatial order of the PDE**. This allows us to use **standard C^0 continuous FEM**.
  - Our auxiliary variable becomes **physically meaningful**: it is the actual physical velocity of the material parcels with respect to the co-moving frame (i.e., w.r.t. the uniform axial motion).
  - **u** and **V** are connected by a first-order linear transport PDE (i.e. pure advection).
    - Standard problem; standard numerical stabilizations are easy to apply.

- **Primal formulation** with **u** and **V** only, by inserting **σ** into the linear momentum balance.
  - Avoids the need for an inf-sup condition (LBB condition); can choose FEM element types more easily.
  - Drawback: not applicable to most models that have no explicit expression of the form $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{u}, \mathbf{V})$.

- Weak (iterative) coupling between mechanical and thermal parts of the multiphysics problem.

- For solving the linear systems, MUMPS. Krylov methods tested, did not converge.

Governing equations – 2D, mechanical subproblem, basic idea

Material: axially moving viscoelastic Kelvin–Voigt, isotropic.

With **V** := d**u**/d*t*, the **constitutive law** looks as if no axial motion was present! C^0 elements are now fine.

Strong form

Linear momentum balance

$$\rho \frac{d\boldsymbol{V}}{dt} - \nabla\cdot\boldsymbol{\sigma} = \rho\boldsymbol{b}$$

Auxiliary variable

Material parcel velocity

$$\boldsymbol{V} = \frac{d\boldsymbol{u}}{dt} \equiv \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{u}$$

Laser velocity

Constitutive law

Elastic and viscous stiffness tensors (rank 4)

$$\begin{aligned}
\boldsymbol{\sigma} &= \boldsymbol{E} : \boldsymbol{\varepsilon} + \boldsymbol{\eta} : \frac{d\boldsymbol{\varepsilon}}{dt} \\
&= \boldsymbol{E} : symm(\nabla \boldsymbol{u}) + \boldsymbol{\eta} : \frac{d}{dt} symm(\nabla \boldsymbol{u}) \\
&= \boldsymbol{E} : (symm\,\nabla)\boldsymbol{u} + \boldsymbol{\eta} : \frac{d}{dt}(symm\,\nabla)\boldsymbol{u} \\
&= \boldsymbol{E} : (symm\,\nabla)\boldsymbol{u} + \boldsymbol{\eta} : (symm\,\nabla)\frac{d\boldsymbol{u}}{dt} \\
&= \boldsymbol{E} : (symm\,\nabla)\boldsymbol{u} + \boldsymbol{\eta} : (symm\,\nabla)\boldsymbol{V} \\
&= \boldsymbol{E} : (symm\,\nabla)\boldsymbol{u} + \tau\,\boldsymbol{E} : (symm\,\nabla)\boldsymbol{V} \\
&=: \mathscr{L}(\boldsymbol{u}) + \tau\,\mathscr{L}(\boldsymbol{V})
\end{aligned}$$

KV retardation time, $\boldsymbol{\eta} =: \tau\,\boldsymbol{E}$

* Thermomechanical terms not shown here.

$$\boldsymbol{E} = 2\mu\,\mathbf{1}_S + 3\lambda\,\mathbf{1}_V$$

Elastic stiffness tensor for isotropic material (in Lamé form)

$$[\mathbf{1}_S]_{ijk\ell} := \frac{1}{2}[\delta_{ik}\delta_{j\ell} + \delta_{i\ell}\delta_{jk}]$$

$$[\mathbf{1}_V]_{ijk\ell} := \frac{1}{3}\delta_{ij}\delta_{k\ell}$$

Weak form

Linear momentum balance – "equation of **u**"

$$\begin{aligned}
&\rho\int \frac{\partial \boldsymbol{V}}{\partial t}\cdot\boldsymbol{\varphi}\,d\Omega + \rho C(\boldsymbol{v};\boldsymbol{V},\boldsymbol{\varphi}) \\
&\quad + \int \boldsymbol{\sigma} : symm(\nabla\boldsymbol{\varphi})\,d\Omega \\
&\quad - \int \boldsymbol{\varphi}\cdot\boldsymbol{\sigma}_0\cdot\boldsymbol{n}\,d\Gamma = \rho\int \boldsymbol{b}\cdot\boldsymbol{\varphi}\,d\Omega \quad \forall\,\boldsymbol{\varphi}
\end{aligned}$$

Auxiliary variable – "equation of **V**" (defining **V** in terms of **u**)

$$\int \boldsymbol{V}\cdot\boldsymbol{\psi}\,d\Omega = \int \frac{\partial \boldsymbol{u}}{\partial t}\cdot\boldsymbol{\psi}\,d\Omega + C(\boldsymbol{v};\boldsymbol{u},\boldsymbol{\psi}) \quad \forall\,\boldsymbol{\psi}$$

Just an L^2 projection – no BCs.

φ: test function of **u** (i.e. virtual displacement δ**u**)
ψ: test function of **V**

To go into a steady state, take $\partial/\partial t \to 0$.

Skew-symmetric weak form of $\boldsymbol{v}\cdot\nabla\boldsymbol{u}$ for a divergence-free advection velocity $\boldsymbol{v}$; $\boldsymbol{\varphi}$ is a test function. Including the boundary term as-is avoids generating a BC on boundaries through which there is flow.

$$C(\boldsymbol{v};\boldsymbol{u},\boldsymbol{\varphi}) := \frac{1}{2}\int [[(\boldsymbol{v}\cdot\nabla)\boldsymbol{u}]\cdot\boldsymbol{\varphi} - [(\boldsymbol{v}\cdot\nabla)\boldsymbol{\varphi}]\cdot\boldsymbol{u}]\,d\Omega + \frac{1}{2}\int [\boldsymbol{n}\cdot\boldsymbol{v}][\boldsymbol{u}\cdot\boldsymbol{\varphi}]\,d\Gamma$$

* Improves numerical stability.

* We additionally stabilize by *streamline upwinding Petrov-Galerkin* (SUPG), not shown here.

jamk

Governing equations – 2D, thermal subproblem

Material: axially moving viscoelastic Kelvin–Voigt, isotropic.

Strong form

Eulerian!
(Not mixed Lagrangean–Eulerian)

Internal energy balance

$$\rho \frac{\partial c}{\partial T} \Theta T + \rho c \Theta - \nabla \cdot (\boldsymbol{k} \cdot \nabla T) - \boldsymbol{\sigma} : \nabla \boldsymbol{V} = \rho h$$

Nonlinear heat equation, with advection.

Auxiliary variable

Material rate of temperature

$$\Theta = \frac{dT}{dt} \equiv \frac{\partial T}{\partial t} + ([\boldsymbol{v} + \boldsymbol{V}] \cdot \nabla) T$$

Material parcel velocity w.r.t. **laboratory** frame

Laser velocity

Material parcel velocity w.r.t. **co-moving** frame

This **splits off advection**, simplifying the implementation.

Also, we now get d*T*/d*t* as a model output.

Constitutive law, with also **thermomechanical terms** shown:

$$\begin{aligned}
\boldsymbol{\sigma} &= \boldsymbol{E} : (symm \nabla) \boldsymbol{u} + \boldsymbol{\eta} : (symm \nabla) \boldsymbol{V} - \left[\boldsymbol{E} : \boldsymbol{\alpha} [T - T_0] + \boldsymbol{\eta} : \left[\frac{\partial \boldsymbol{\alpha}}{\partial T} [T - T_0] + \boldsymbol{\alpha} \right] \Theta \right] \\
&= \boldsymbol{E} : (symm \nabla) \boldsymbol{u} + \tau \boldsymbol{E} : (symm \nabla) \boldsymbol{V} - \left[\boldsymbol{E} : \boldsymbol{\alpha} [T - T_0] + \tau \boldsymbol{E} : \left[\frac{\partial \boldsymbol{\alpha}}{\partial T} [T - T_0] + \boldsymbol{\alpha} \right] \Theta \right] \\
&=: \mathscr{L}(\boldsymbol{u}) + \tau \mathscr{L}(\boldsymbol{V}) - \mathscr{T}(T, \Theta)
\end{aligned}$$

Static part

Part dependent on the temperature material rate Θ

Weak form

Internal energy balance – "equation of *T*"

$$\begin{aligned}
&\rho \int \frac{\partial c}{\partial T} \Theta T \, w \, d\Omega + \rho \int c \Theta \, w \, d\Omega + \int [\boldsymbol{k} \cdot [\nabla T]] \cdot [\nabla w] \, d\Omega \\
&- \int \boldsymbol{n} \cdot \boldsymbol{q}_0 \, d\Gamma - \int [\boldsymbol{\sigma} : \nabla \boldsymbol{V}] w \, d\Gamma = \rho \int h w \, d\Omega \quad \forall w
\end{aligned}$$

Feed in **σ** and **V** from the mechanical subproblem.

Auxiliary variable – "equation of Θ"
(defining Θ in terms of *T*)

$$\int \Theta Q \, d\Omega = \int \frac{\partial T}{\partial t} Q \, d\Omega + C(\boldsymbol{v} + \boldsymbol{V}; T, Q) \quad \forall Q$$

Just an L^2 projection – no BCs.

* We additionally stabilize by SUPG.

w: test function of *T*
Q: test function of Θ

To go into a steady state, take $\partial/\partial t \to 0$.

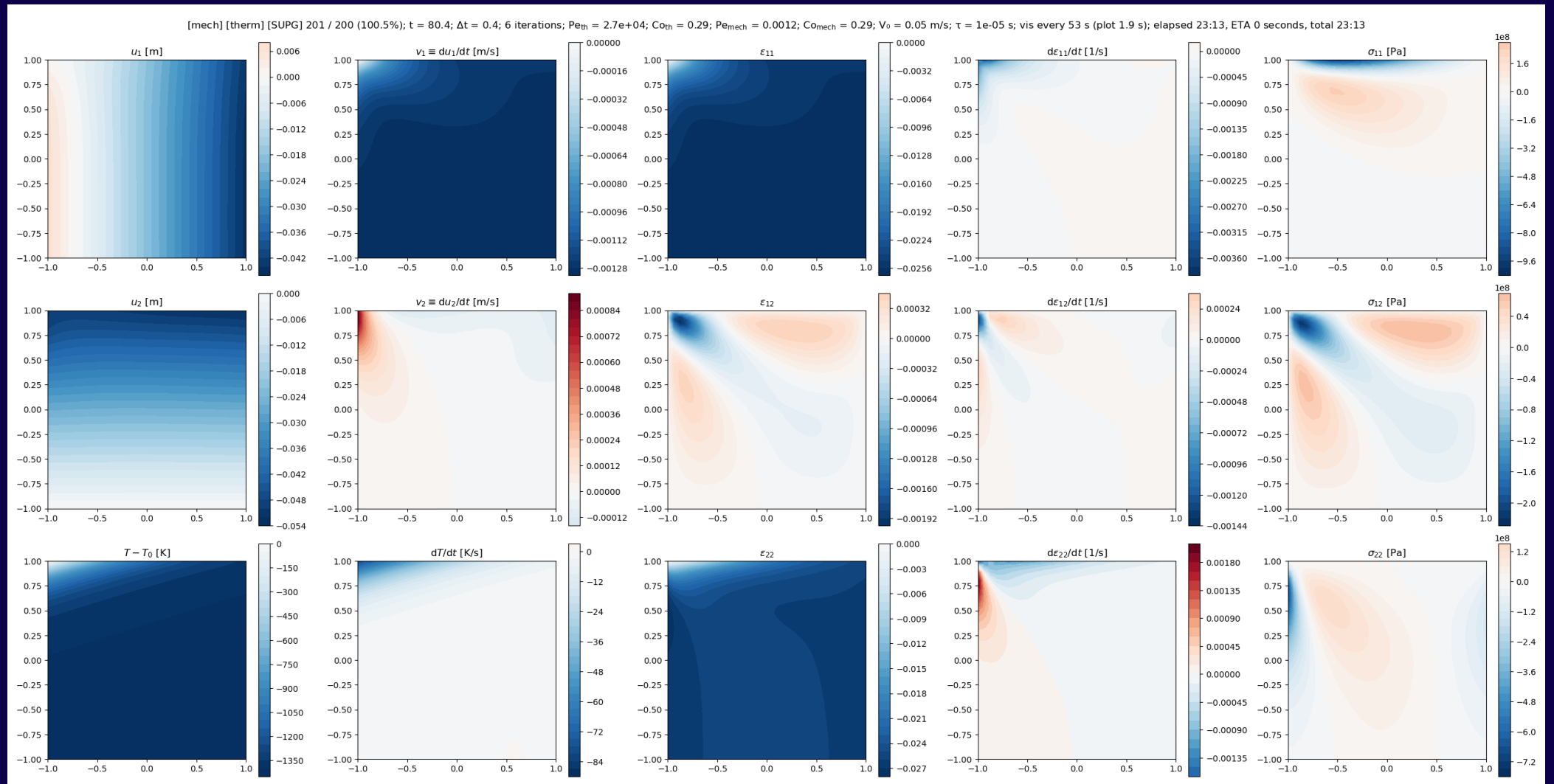
***CAUTION:* *Axially moving material*!**
Even a steady state exhibits advection effects in the laboratory frame.

jamk

2D thermomechanical simulation, FEM

Material: axially moving viscoelastic Kelvin–Voigt, isotropic.

Solver and some agility utilities for FEniCS on GitHub:
https://github.com/Technologicat/extrafeathers
⊳ *Eulerian linear solid mechanics*



Axially moving 316L steel, final state.

UL corner:
$u_1 = 0$

Bottom edge:
$u_2 = 0$

Otherwise:
$\boldsymbol{\sigma} \cdot \mathbf{n} = 0$

Inlet T profile:
0D cooling sim.

P2 elements.

jamk

Part 2 – AI acceleration

- **Goal**: investigate modern PDE acceleration techniques, keeping an eye on possible **digital twin** applications.
  - *A **digital twin** is defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, monitoring, control and optimization of the asset for improved decision making throughout the life cycle of the asset and beyond.*
    A. Rasheed, O. San, T. Kvamsdal. Digital twin: Values, Challenges and Enablers from a Modelling Perspective. IEEE Access, 8: 21980–22012, 2020.
- A high-fidelity FEM simulation is too slow for a digital twin, so we need to accelerate the computation.
- **Reduced order modeling** (ROM):
  - Represent the solution using only a small number of degrees of freedom (DOFs), but use an algorithm that chooses them wisely.
  - Obtain an **interpolator** that can quickly compute solutions in the class it was trained for.
  - Classical ROM methods:
    - **POD** (*proper orthogonal decomposition*): truncated SVD (*singular value decomposition*) of snapshot matrix.
    - **PGD** (*proper generalized decomposition*): grow an optimal Galerkin basis, one basis function at a time.
    - Both yield a **small Galerkin basis** that approximates the solution space.

AI-powered reduced order modeling

- Leverage **deep learning**: training “deep” neural networks, i.e. with many hidden layers (extensive overview by Goodfellow et al., 2016, https://www.deeplearningbook.org/)
  - Artificial neural networks (ANNs) invented in the 1980s; deep networks enabled by 2010s hardware.
  - **Exponential gains from depth**: up to a problem-specific depth limit, deep networks with small layers perform better than shallow networks with large layers. (He et al., 2015b, https://arxiv.org/abs/1502.01852)

- **Discriminative** vs. **generative** modeling [statistics and machine learning]:

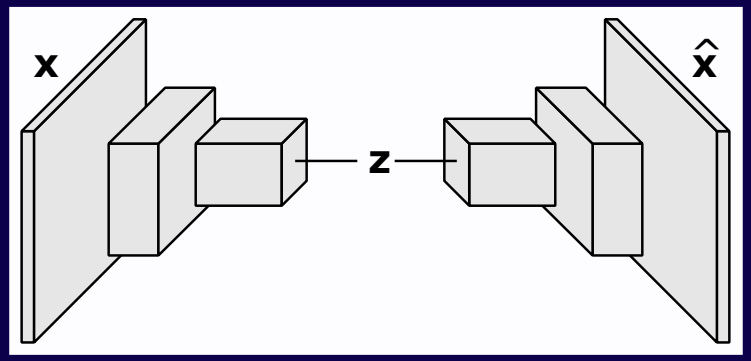
  *While in **discriminative modeling** one aims to learn a predictor given the observations, in **generative modeling** one aims to solve the more general problem of learning a joint distribution over all the variables. A generative model estimates how the data is generated in the real world.*
  (Kingma and Welling, 2019, https://arxiv.org/abs/1906.02691)

- AI concepts particularly interesting for ROM applications:
  - **Representation learning**: automatic feature detection and classification
  - **Generative models** that use **latent representations**
    - For example: VAE, *variational autoencoder*; and LDM, *latent diffusion model*
  - **Physically informed neural network** (PINN)
    - Incorporate prior information (from the known PDE) into the NN training process.

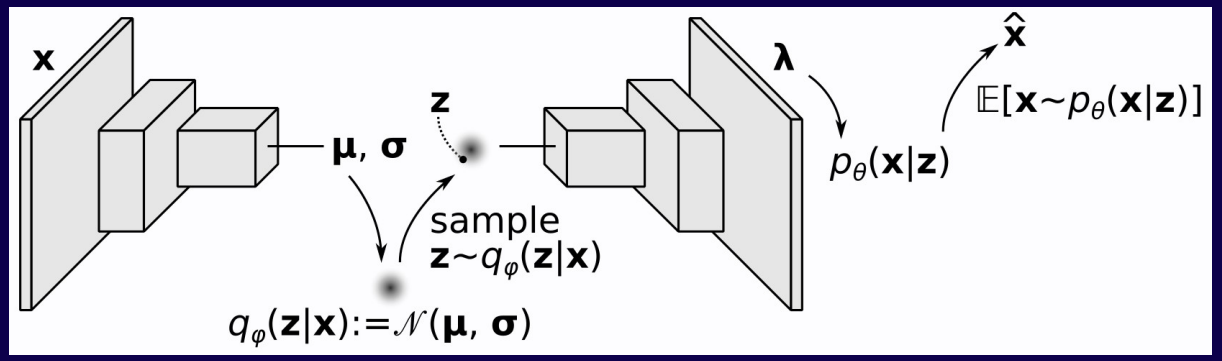
Autoencoders

- **Autoencoder** (AE): Take a data sample as input, and attempt to reconstruct that sample as output.
  - **Low-dimensional bottleneck** forces the AE to learn to represent high-level **features** of the data, because the architecture does not have the capacity to learn the identity mapping.
  - However, a basic AE does not care about its latent space layout: as **x** varies, **z** tends to vary as spaghetti.
    - Technically, the code may be δ-ε continuous; and even Lipschitz-continuous, but with a uselessly large constant. Cf. the resolution of Loschmidt's and Zermelo's paradoxes in thermodynamics (see *Rothstein, 1974, pp. 88–89*).

AE



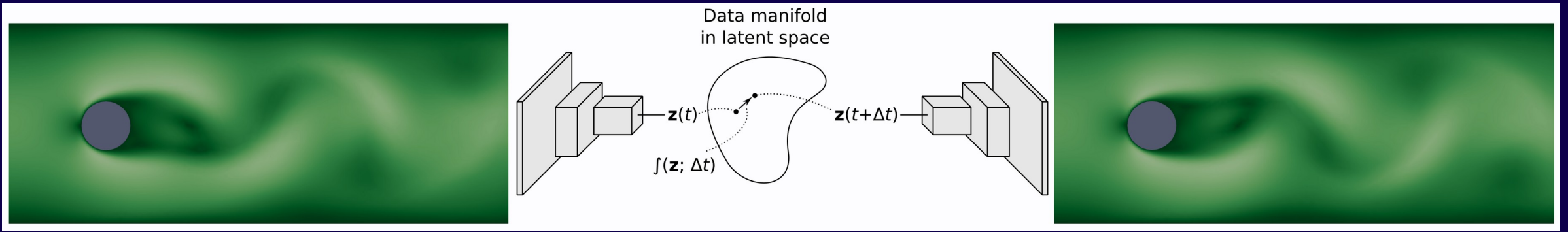
VAE



- **Variational autoencoder** (VAE; Kingma and Welling, 2013, https://arxiv.org/abs/1312.6114) (tutorial paper: Kingma and Welling, 2019, https://arxiv.org/abs/1906.02691)
  - Stochastic, **Bayesian autoencoder**, which produces a **latent representation with a clean layout**.
    - The encoder learns to map similar training samples near each other in the latent space.
  - Three distributions: **inference** / **encoder** / **variational posterior** ($q_\varphi(\mathbf{z}|\mathbf{x})$, data → latent), **observation** / **decoder** ($p_\theta(\mathbf{x}|\mathbf{z})$, latent → data), and the **latent prior** ($p(\mathbf{z})$, latent layout regularization).
  - Trained by maximizing the **evidence lower bound** (ELBO), a variational estimate of the **model evidence** $\log p_\theta(\mathbf{x})$, i.e. the marginal log-likelihood of the training data under the model.

AI accelerator overview

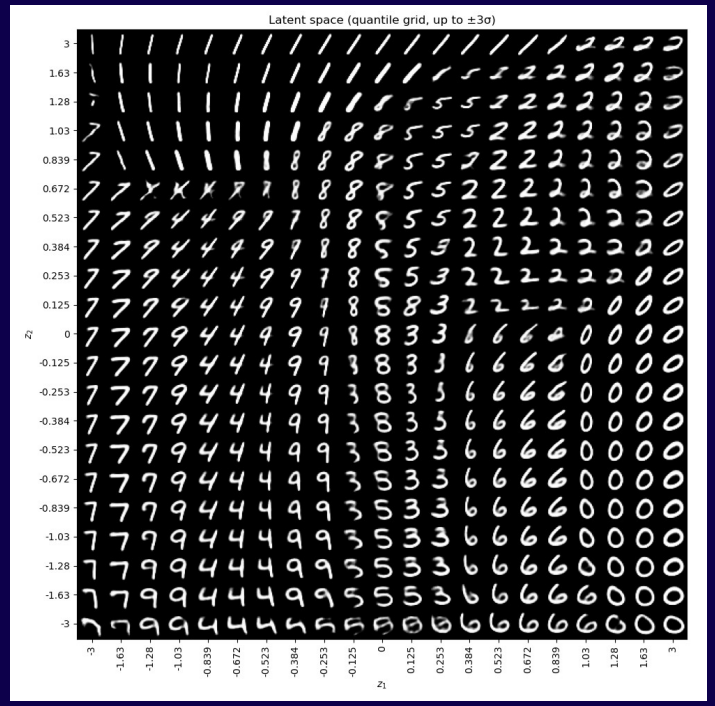
- **Two AI components**:
  - **Dimension reduction and restoration**
    - A **VAE** with deep **convolutional neural networks** (CNN, *convnet*) for model parameter estimation.
    - CNNs are a machine vision technology. PDE solutions projected to a cartesian grid are images.
    - This component provides the low-dimensional latent representation.
  - **Time evolution in latent space** (ROM integrator)
    - Integration of the PDE over a small timestep → a small local hop in latent space.
    - This component provides fast time integration, by operating on the low-dimensional latent representation.



jamk

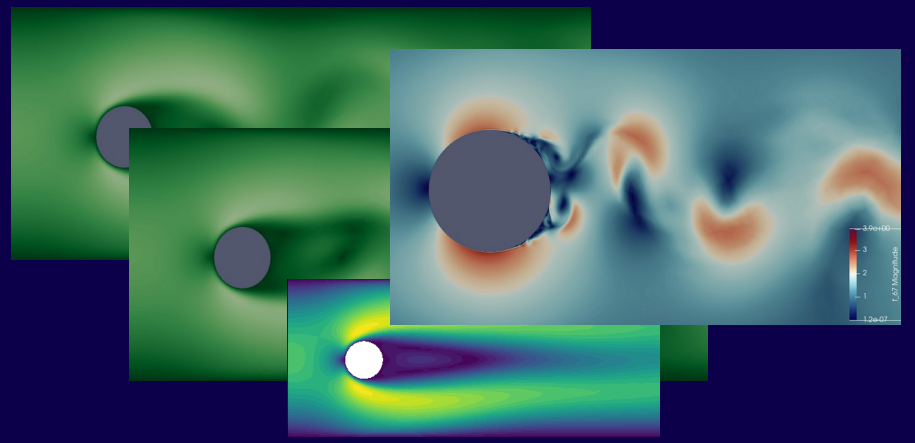
Development plan

Step 1: Prototype VAE on a well-studied dataset



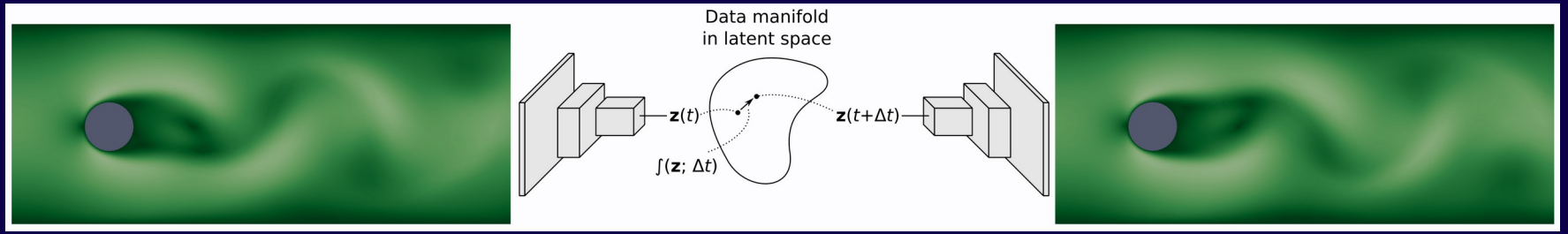
Step 2: Apply to a well-studied PDE

Produce FEM simulation data

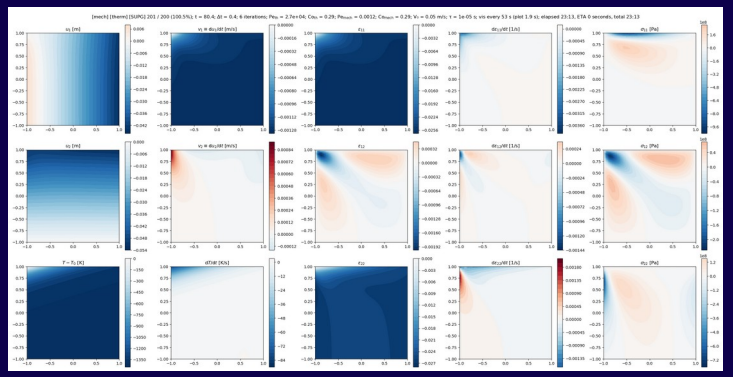


Formulate physically informed loss function

Develop time-evolution AI (ROM integrator)



Step 3: Deploy on target problem



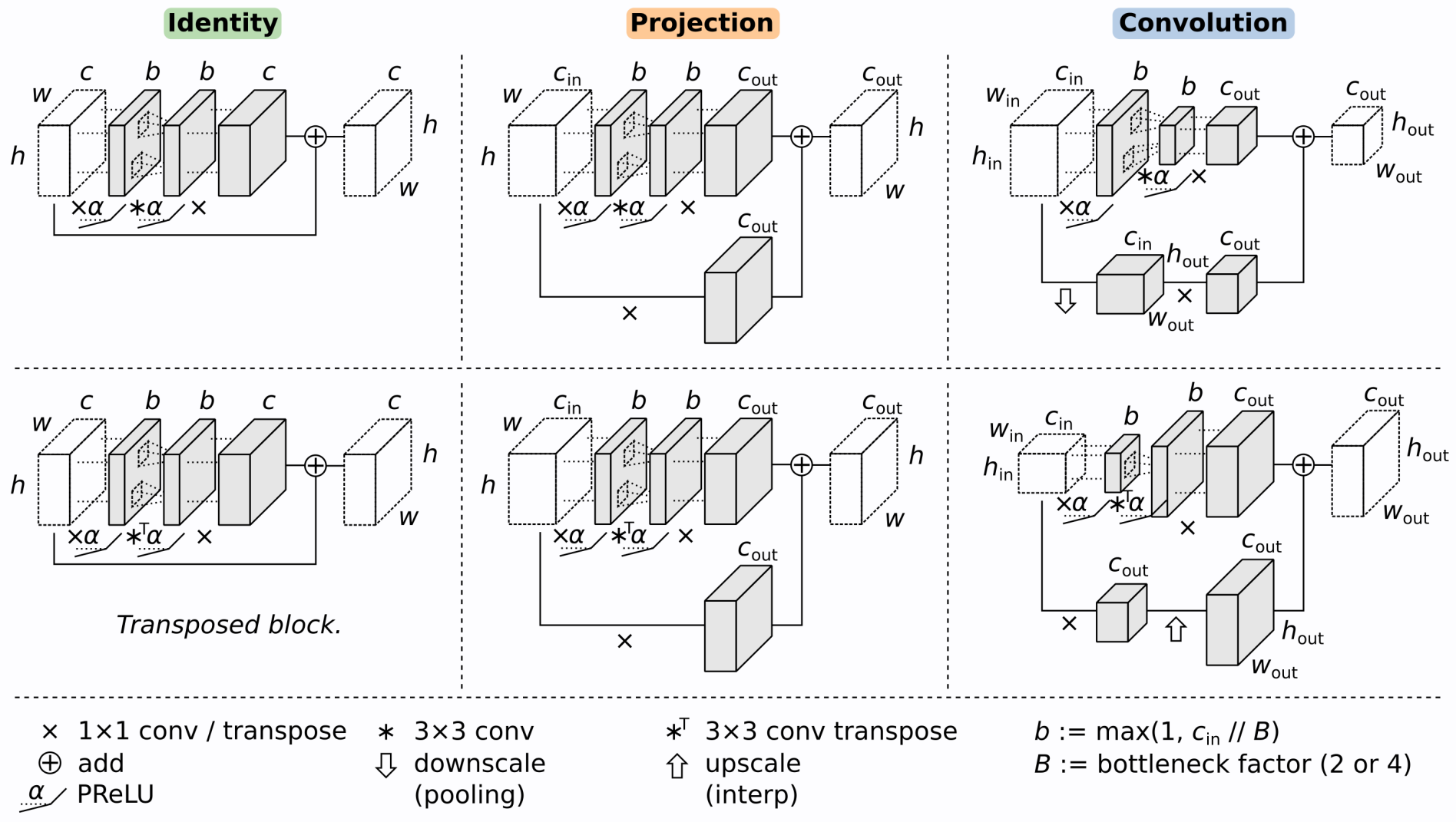
jamk

NN design

- **Bottleneck residual blocks** (He et al., 2015a, https://arxiv.org/abs/1512.03385v1)
  - Instead of $f(\mathbf{x})$, learn the **residual** $g(\mathbf{x}) := f(\mathbf{x}) - \mathbf{x}$. (In practice, add the input $\mathbf{x}$ to the block output.)
  - Allowing information to flow through the **skip-connections** avoids the vanishing gradient problem.
  - Other alternatives: highway networks, and skip-connections with concatenation (instead of addition).
- **He normal initialization** of convolution kernels (He et al., 2015b, https://arxiv.org/abs/1502.01852)
  - Improves the chances of getting a reasonable initial state for training.
- **PReLU** activation (He et al., 2015b)
  - Essentially, trainable leaky ReLU.
  - Performs slightly better than the basic ReLU, at almost no extra computational cost.
- **Channel dropout regularization** (Tompson et al., 2015, https://arxiv.org/abs/1411.4280)
  - **Dropout** = randomly disable parts of the NN during each training epoch.
  - Original neuron dropout: Hinton et al., (2012, https://arxiv.org/abs/1207.0580), Srivastava et al. (2014, https://jmlr.org/papers/v15/srivastava14a.html)
  - In convolution networks, where nearby pixels are correlated, better to drop entire channels.
    - Recommended channel retain ratio 0.9 (Cai et al., 2020, https://arxiv.org/abs/1904.03392)
  - Why dropout **improves generalization**:
    - Selective pressure against high-order interactions (Lengerich et al., 2022, https://arxiv.org/abs/2007.00823)
    - Approximate powerset of 2^N subnetworks, leveraging the ensemble effect (Labach et al., 2021, https://arxiv.org/abs/1904.13310; Ganaie et al., 2022, https://arxiv.org/abs/2104.02395)
    - Bayesian explanation (Gal and Ghahramani, 2016, https://arxiv.org/abs/1506.02142)

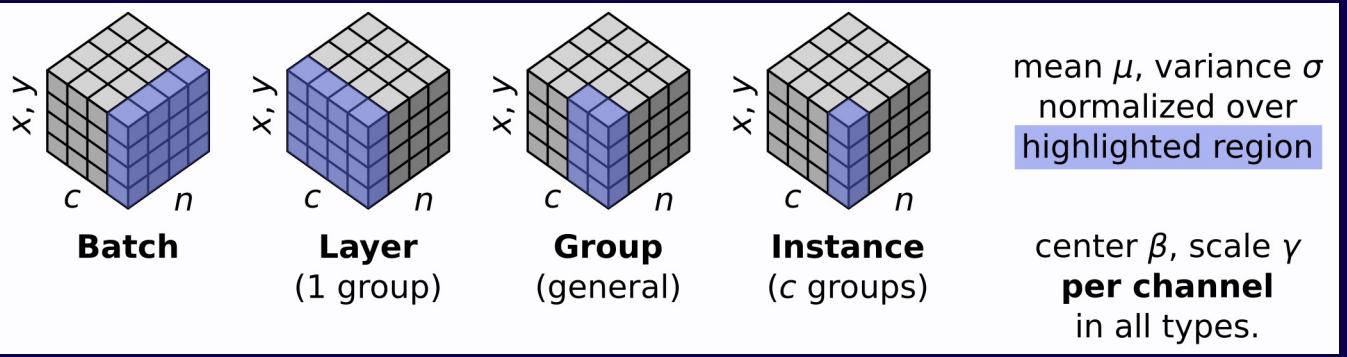
Bottleneck residual network (ResNet) blocks

* Originally introduced by (He et al., 2015a, https://arxiv.org/abs/1512.03385v1)



Normalization

- Normalization, i.e. keeping the mean near zero and variance near 1, helps against vanishing or exploding neural network output due to a bad random initialization, and helps with numerical stability in general.
- **Batch normalization (BN)** is a popular method.
  - However, BN is sensitive to the batch size, and may misfire in ways that are hard to debug. See e.g.
    - https://www.alexirpan.com/2017/04/26/perils-batch-norm.html
    - https://mindee.com/blog/batch-normalization/
- **Group normalization (GN)** (Wu and He, 2018, https://arxiv.org/abs/1803.08494) is a well-behaved alternative.
  - In GN, each sample in the input batch is treated separately.
  - GN computes the same way during both training and inference.
  - Special cases of GN are **Layer normalization (LN)** and **Instance normalization (IN)**.



- We use **Instance normalization (IN)**.
  - In our network, each channel represents an arbitrary feature; no subsets with different roles.
- **Operation ordering**: for best results, **first normalize, then dropout**, just before feeding into the next conv layer (Cai et al., 2020, https://arxiv.org/abs/1904.03392)

Hyperparameters

- **Batch size**
  - VRAM allowing, large batches compute faster (wall time per input sample) due to GPU parallelization.
    - However, this is illusory: to keep the total number of optimizer steps constant, increasing the batch size by a factor of *k* requires also increasing epochs by the same factor *k*.
  - Large batch sizes reduce noise in the loss gradient estimate, making it more accurate.
    - However, this also tends to help the optimizer find sharp minima, making the trained network sensitive to small perturbations in the input data. **This is why large batch sizes** tend to produce models that **don't generalize well** (at the same total number of gradient updates). (Keskar et al., 2017, https://arxiv.org/abs/1609.04836)
  - In practice, a batch size of **512** was observed to work well for quick prototyping.

- **Optimizer**: **ADAM**, at Keras default settings (except learning rate schedule)
  - Train until effective saturation, in practice use a fixed setting of **200 epochs**
    - One epoch = one complete pass through the training data.

- **Learning rate schedule**: **Cyclical learning rate** (CLR; Smith, 2017, https://arxiv.org/abs/1506.01186)
  - Tends to **improve convergence rate** per epoch of training, compared to a monotonically decreasing LR.
  - Best loss values are often seen at the end of a cycle (when the LR hits its minimum value), but it seems also the increasing part of the cycle (where loss tends to increase) is important.
  - Smith hypothesizes that temporarily increasing the LR helps escape saddle points and bad local optima.

Learning rate schedule

- Non-analytic C^∞ continuous CLR, with exponential decay envelope ($\gamma = 0.9$).
  - Empirically, the C^∞ continuous profile seems to reduce noise in the training ELBO.

- The half-cycle shape is based on a helper function in the construction of the standard mollifier in PDE theory:

$$s(x, m) := \psi(x, m) \,/\, [\psi(x, m) + \psi(1 - x, m)], \quad \text{where}$$
$$\psi(x, m) := \exp(-1 \,/\, x^m)\, \chi_{(0,\,\infty)}(x)$$

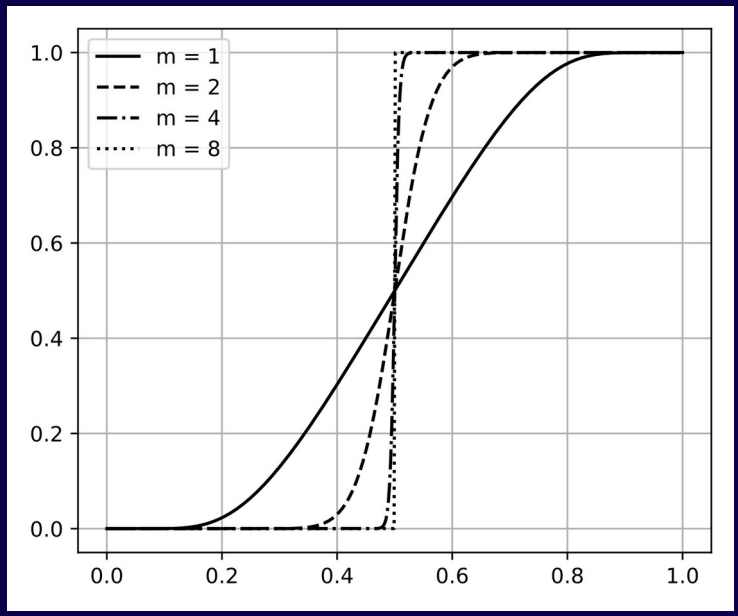
  and χ is the indicator function (1 if x belongs to the set, 0 otherwise).

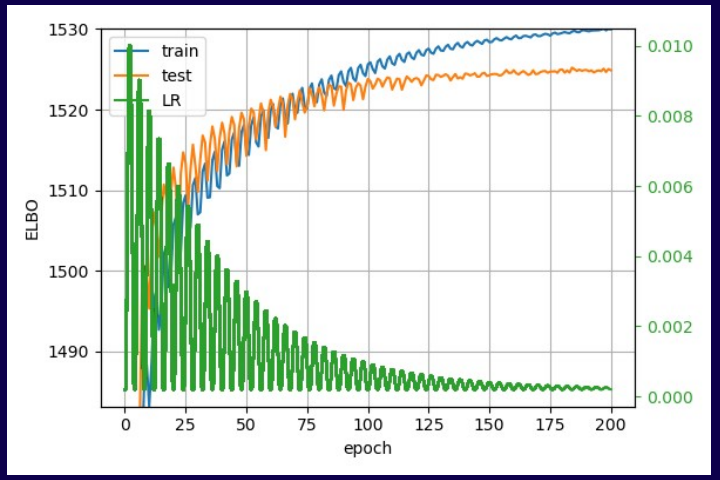
- $s \equiv 0$ for all $x < 0$, $\quad s \equiv 1$ for all $x > 1$.
- s is C^∞ continuous **everywhere**, also at the seams.
- s is **non-analytic**: non-trivial Taylor series only in $x \in [0, 1]$.

- m: transition steepness; we used ***m = 1***.

- **Example training history**: cyclic LR $\in$ [2e-4, 1e-2].

$s(x, m)$





jamk

Continuous Bernoulli VAE

- **Continuous Bernoulli** VAE (Loaiza-Ganem and Cunningham, 2019, https://arxiv.org/abs/1907.06845) performs better **for continuous-valued data**, such as pixel intensities of grayscale (or RGB) natural images, than the classical discrete Bernoulli VAE with input binarization.

- **Continuous Bernoulli distribution**:

$$p_\lambda(x) = \alpha\,[\lambda^x + (1-\lambda)^{1-x}]\,, \quad \text{where } x \in [0, 1], \text{ and } \alpha \text{ is chosen such that } \int p_\lambda(x)\,dx = 1$$

- No need to binarize the input; this model can represent the original data faithfully.
- In practice, yields **sharper reconstructed images** with the same NN design.

- The only change needed is adding a log-normalizing constant ($\log \alpha$) to the observation log-likelihood, thus converting it to a properly normalized probability density for continuous $x \in [0, 1]$.
  - Unfortunately, this makes performance statistics numbers incomparable between these two variants.

- **Does consistency matter?**
  - In the discrete Bernoulli VAE, the pixelwise expectation (the customary choice as the decoded image) is not even a sample from the observation model, which can only represent discrete pixel values {0, 1}.
  - In the continuous Bernoulli VAE, it seems it is in practice fine to just return the pixelwise distribution parameter $\lambda \in [0, 1]$ tensor as the decoded image, instead of computing the expectation.
    - Careful: for the **continuous** Bernoulli, λ itself is not the expectation.

Think like a software developer

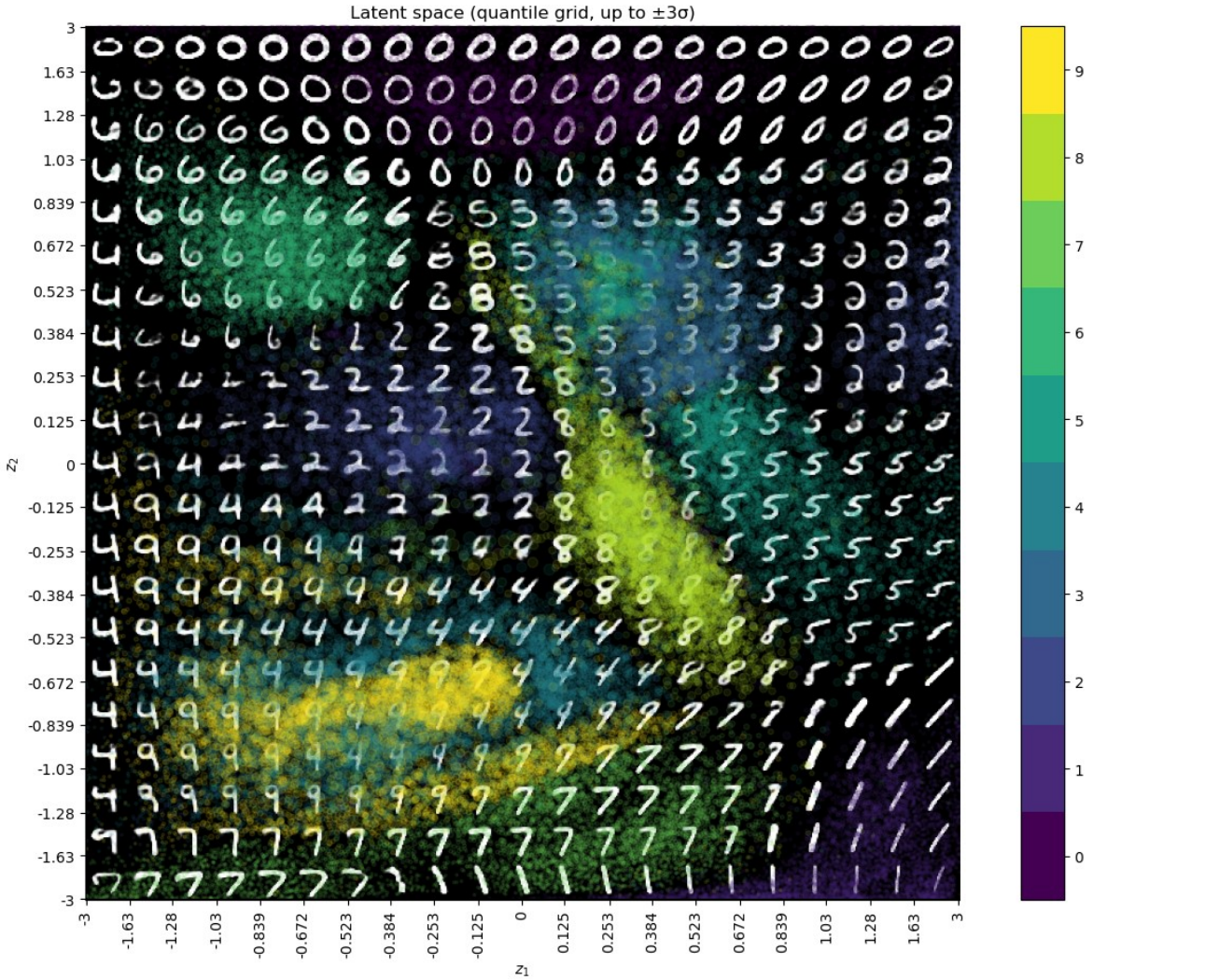
- Why is $p_\theta(\mathbf{x}|\mathbf{z})$ pixel- and channelwise independent? **So that we can easily sum the log-likelihoods.**
  - The decoder handles the pixel-level correlations, by linking the pixelwise parameters inside the NN.

- The binary cross-entropy loss is specific to discrete Bernoulli with binarized input. Avoid it.
  - Just evaluate the ELBO as $\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\varphi(\mathbf{z}|\mathbf{x})$.
  - The general way has two advantages:
    - **Transparency**: avoids an extra abstraction.
      - Each new abstraction is a learning cost for the programmer (Paul Graham: On Lisp, 1992).
      - Even if the cross-entropy happens to run faster… *premature optimization is the root of all evil* (Knuth).
    - **Modularity**: allows easily **changing the observation model** (e.g. to a gaussian, as in Lin et al., 2019, https://arxiv.org/abs/1909.03765). For best results, "know your data".

- The decoder does not need to be a mirror image of the encoder. A VAE can use **an arbitrary NN as the decoder** (Dieng et al., 2019, https://arxiv.org/abs/1807.04863).
  - How: pretend the decoder output to be not an image, but a tensor of pixelwise parameters for the observation model $p_\theta(\mathbf{x}|\mathbf{z})$. Then proceed as usual.
  - Dieng et al. use a Gated PixelCNN (van den Oord et al., 2016, https://arxiv.org/abs/1606.05328) as the decoder, with impressive results.
  - When a VAE is trained end-to-end with a ResNet encoder and a PixelCNN decoder, the PixelCNN can handle much of the pixel-level detail of the reconstruction process, freeing up more capacity of the latent representation to **capture higher-level features** from the data.

VAE, 1.7M parameters, 2D latent space [old architecture]

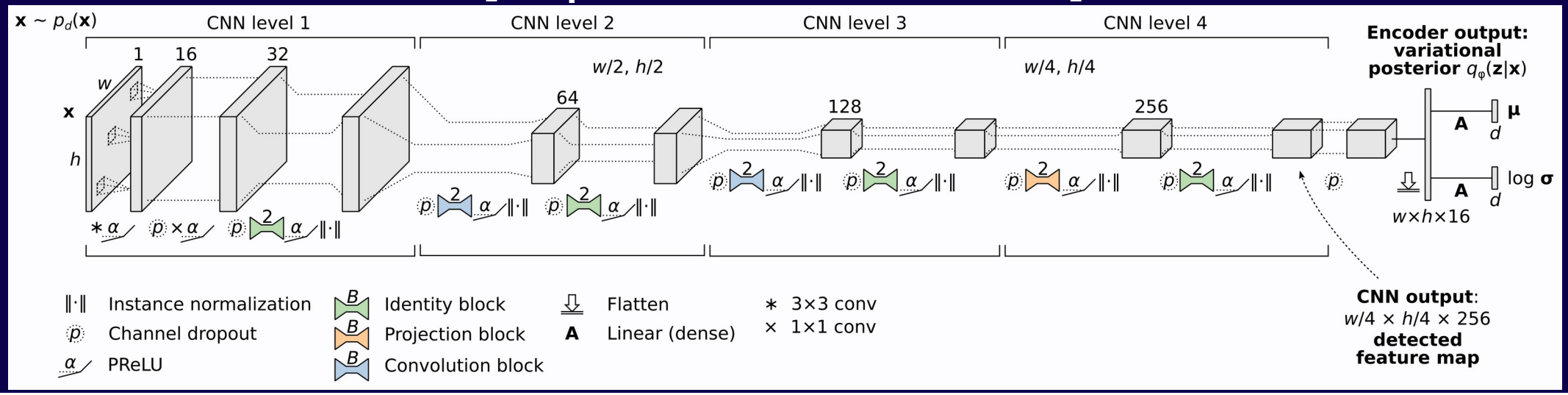
Final test ELBO ≈ 1345.

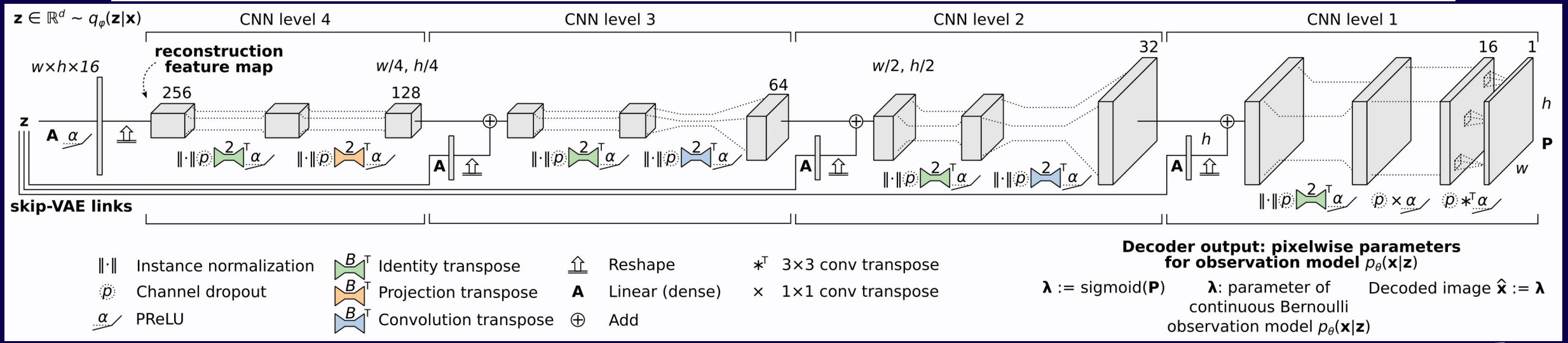


- **Dataset**: *MNIST digits*: 70 000 grayscale images, 28×28 pixels each, of handwritten digits 0 … 9, paired with the corresponding ground truth class labels.
- The VAE has **no** access to the class labels.
- **Not a single "handwritten" digit shown in the figure exists in the training data** – they are all decoded representations of points in the latent space.
- The splashes of color indicate the encoded positions of the training samples in the latent space.
- Points in unused parts of the latent space (where no training samples live) may decode into nonsense.
- 2D latent space, observation model **continuous** Bernoulli**, residual convnet, ~1.7M parameters, trained 113 epochs, batch size 64 (~1.5h, NVIDIA Quadro RTX 3000, 6 GB VRAM), standard 60k/10k training/test split. Datatype float32.

** (*Loaiza-Ganem and Cunningham, 2019, https://arxiv.org/abs/1907.06845*)

Encoder and decoder [improved architecture]



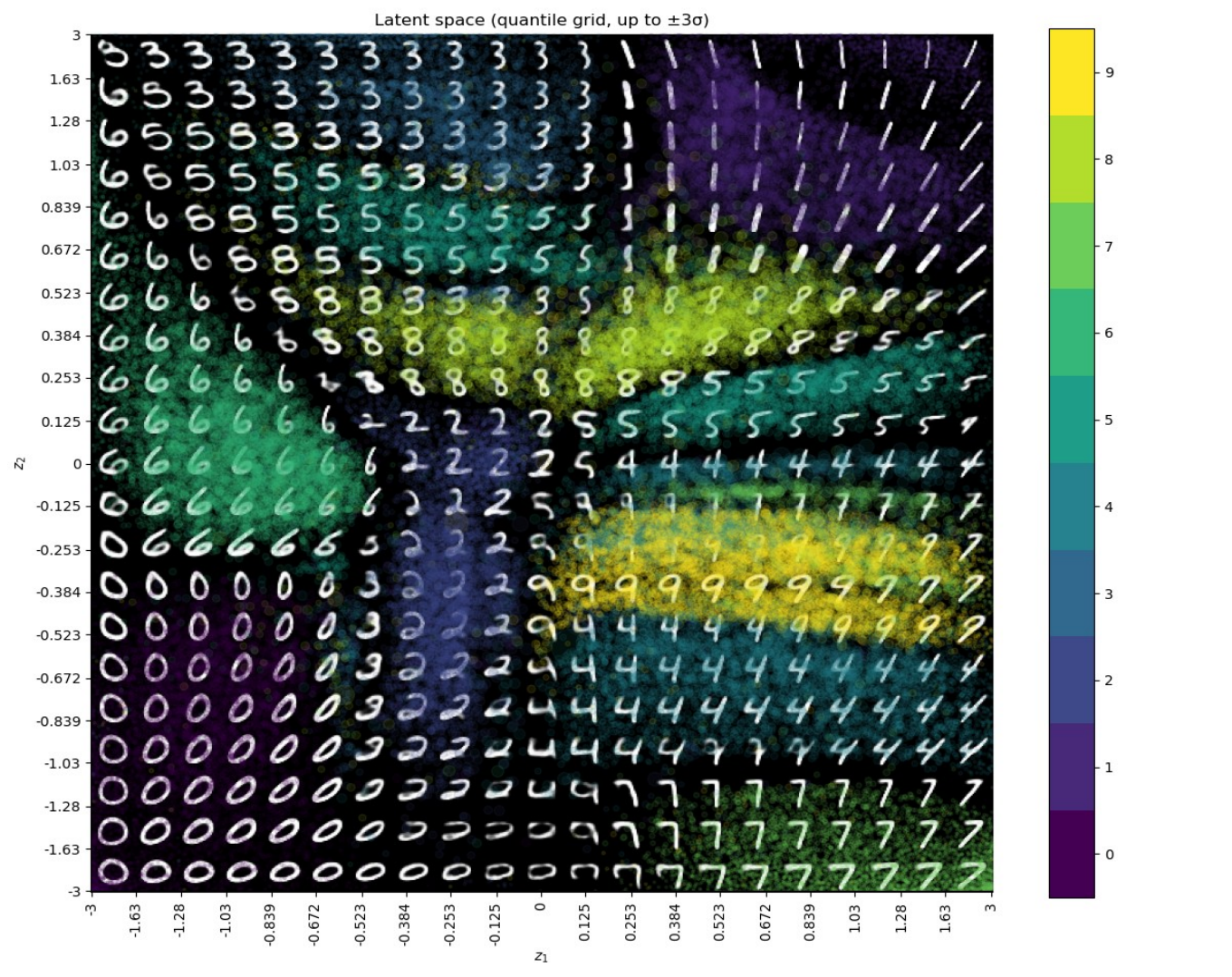


* Skip-VAE originally introduced by Dieng et al. (2019, https://arxiv.org/abs/1807.04863).
Note: **not** a U-Net (Ronneberger et al., 2015, https://arxiv.org/abs/1505.04597); no cross-skip-connections across the code bottleneck **z**.

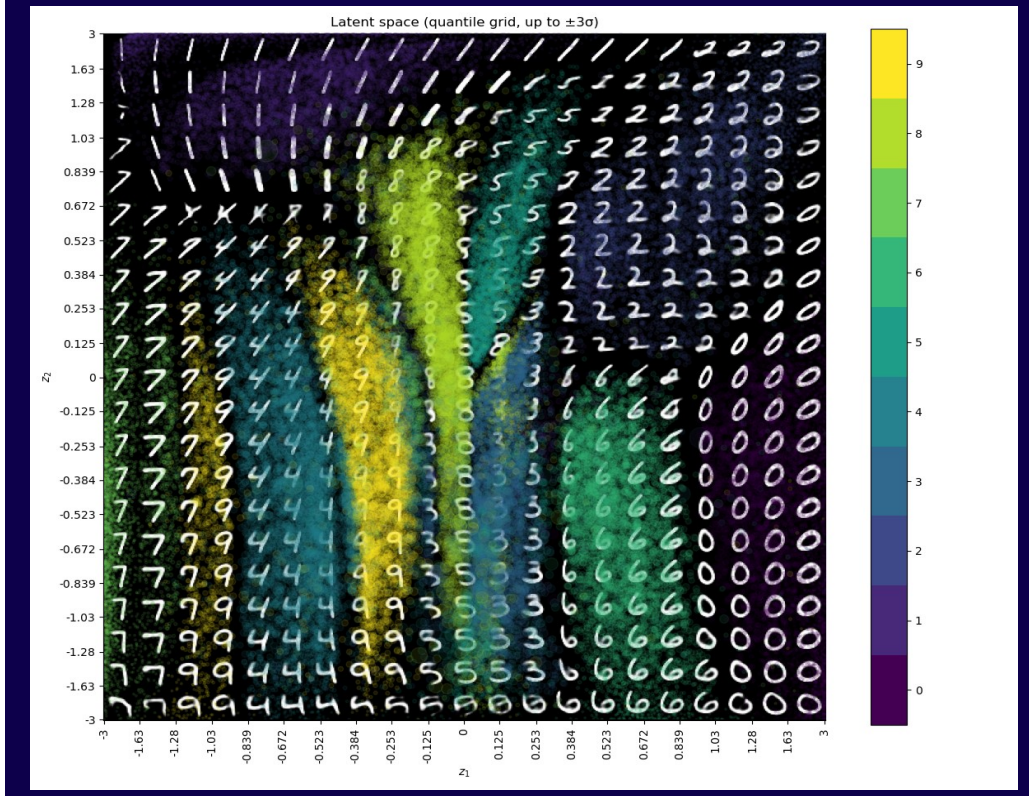
ResNet continuous Bernoulli VAE, built on top of TensorFlow:
https://github.com/Technologicat/extrafeathers/tree/master/demo/vae

VAE, 3M parameters, 2D latent space

Final test ELBO ≈ 1362.
(up from 1345)



- Using the improved architecture.
- float32 data, **float16 compute** with **gradient scaling**.
- Batch size 512.
- Two test runs shown.



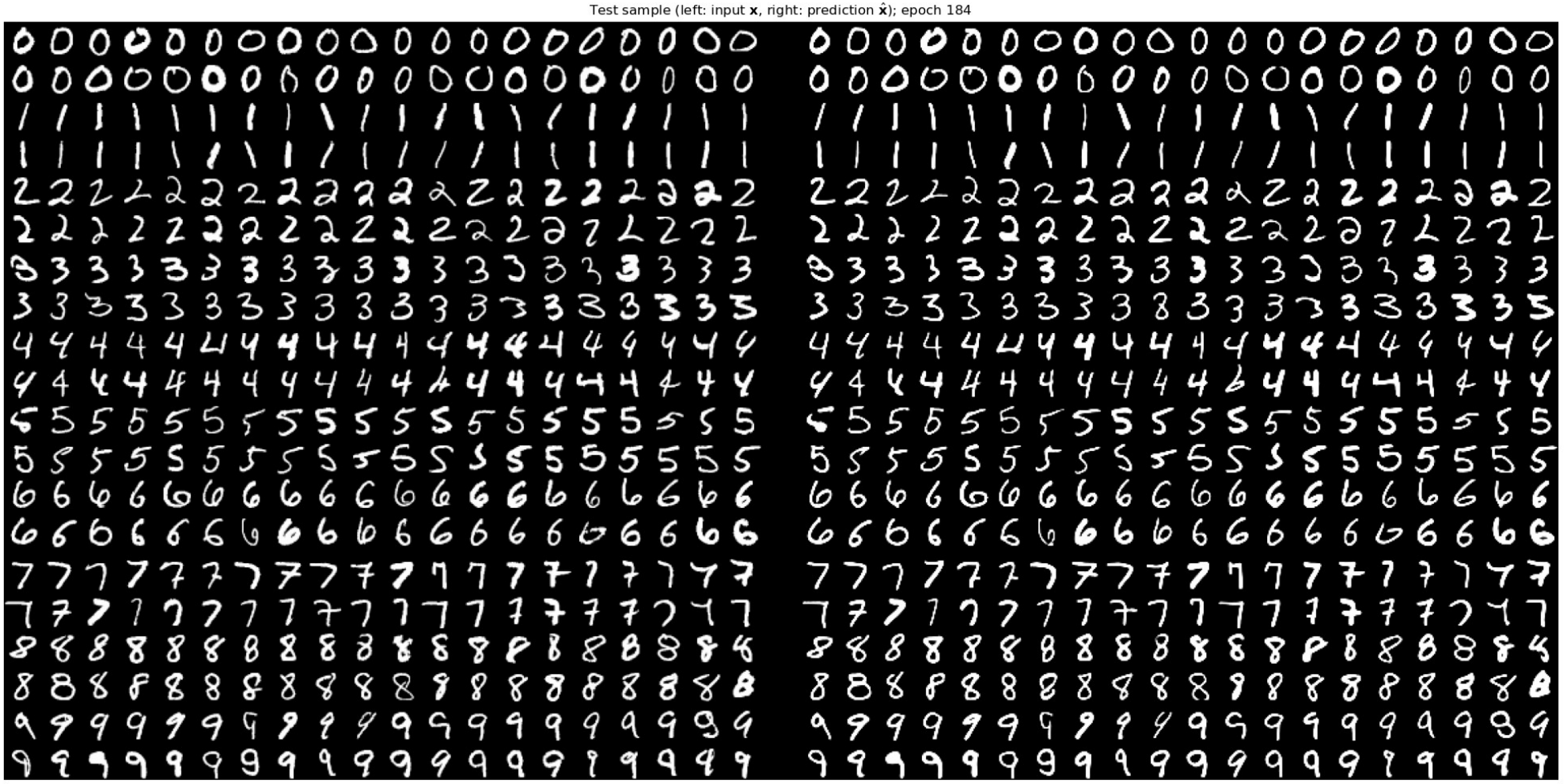
jamk

ResNet continuous Bernoulli VAE, built on top of TensorFlow:
https://github.com/Technologicat/extrafeathers/tree/master/demo/vae

VAE, 3M parameters, 20D latent space

Final test ELBO ≈ 1525.
(up from 1362)

Unseen test data **VAE reconstruction**



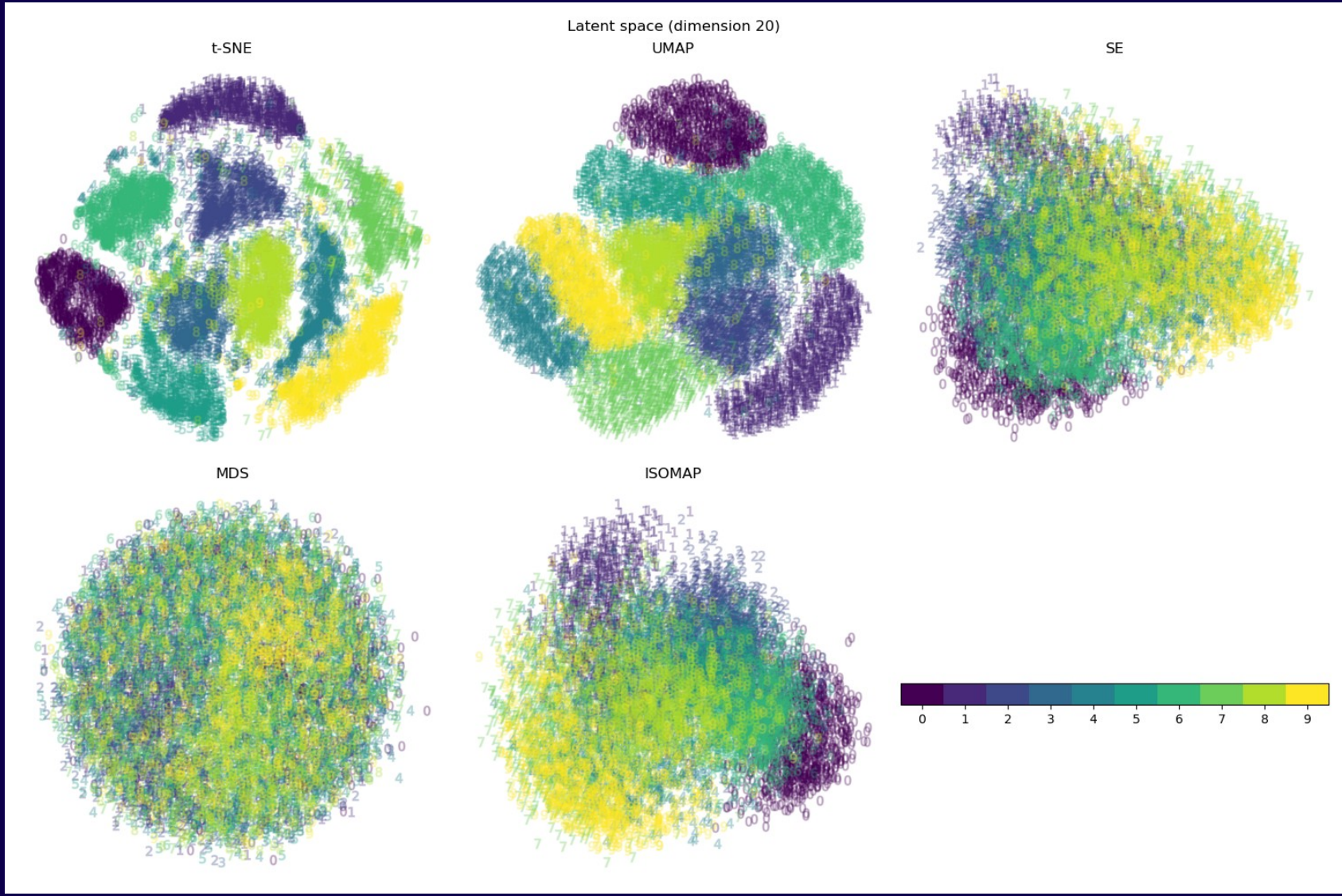
- We see the 2D latent space was a significant performance bottleneck.

  (This suggests that all the handwriting variation in MNIST does not fit into two dimensions.)

jamk

VAE, 3M parameters, 20D latent space

Final test ELBO ≈ 1525.



- Various 2D embeddings of the data manifold that lives in the 20D latent space. 10k test samples.
- t-SNE (t-distributed Stochastic Neighbor Embedding) and UMAP (Uniform Manifold Approximation and Projection) cluster well.
- Spectral Embedding (SE) fails to separate the latent representation using the two leading eigenvectors; thus data manifold dimension > 2.
- MultiDimensional Scaling (MDS), which aims to preserve distance in the ambient 20D space, cannot discern any structure.
- ISOMAP, which aims to preserve distance along the manifold, produces results similar to SE.

jamk

Potential for future improvement

- **Data augmentation**
  - Use case specific, but typically significantly reduces overfitting to the training data.
  - GPU-powered online augmentation tested. As of TF 2.12, slow; 6–10× per-sample performance hit in encoder.
  - At least for MNIST, more efficient to precompute the augmented dataset and use that for training.
- **Consistently regularized VAE** (CR-VAE; Sinha and Dieng, 2022, https://arxiv.org/abs/2105.14859)
  - A small modification to the ELBO objective that causes each original data sample and its data-augmented copies to map to nearby points in latent space.
  - Needs only the classic reparameterization trick and Monte Carlo sampling, already used in any VAE.
- **Epitomic VAE** (Yeung et al., 2017, https://arxiv.org/abs/1706.03643)
  - Split responsibilities of latent channels to enforce learning different features of data. Steers away from posterior collapse.
  - Not immediately clear how to handle the discrete epitome variable in the latent-space time evolution of the accelerator.
- **Optimal latent prior** (Takahashi et al., 2019, https://arxiv.org/abs/1809.05284)
  - Avoid over-regularization incurred by the classical gaussian latent prior $p(\mathbf{z})$. This too steers away from posterior collapse.
  - The optimal prior is the aggregated variational posterior $q_\theta(\mathbf{z})$, which is slow to compute. The authors suggest a trick for how to estimate it inside the ELBO objective function.
- **Automatic fit quality estimation** (Lin et al., 2019, https://arxiv.org/abs/1909.03765)
  - Detect the amount of noise the data must be assumed to have to make the best-fit model fit the data.
  - Needs an observation model that has a variance parameter, e.g. Gaussian… which should be well-suited for PDE data.
- **Fast surrogate** for the aggregated variational posterior $q_\theta(\mathbf{z})$ (Lin et al., 2019)
  - Use a small gaussian mixture. Can be prepared by EM or variational Bayesian inference methods.
  - The true $q_\theta(\mathbf{z})$ is severely overfitted to training data, so the surrogate is better for generating (unconditional) samples from the generative model **that look different from the training samples**.

jamk

Things to be aware of

- **Hallucinations**
  - Even with a physically informed VAE, possible to get a solution of the PDE, but slightly off from the correct solution.
  - May be important for application of AI accelerators to nonlinear and/or hyperbolic PDEs.

- **Data-dependence**
  - How representative is MNIST (10 separate digit classes) as a representation learning testbed?
  - E.g., for von Kármán vortex shedding data, one would expect to find one continuous manifold describing the limit cycle.

- **Curse of recursion** (Shumailov et al., 2023, https://arxiv.org/abs/2305.17493)
  - **Don't train generative models on AI-generated data!**
  - Doing so drops outliers, which are crucial to correctly describe a natural data distribution.
  - After only a few cycles of training new generative AIs on previous-generation outputs, the output becomes useless.

- **NN phase transitions** (Dettmers, 2022, https://timdettmers.com/2022/08/17/llm-int8-and-emergent-features/)
  - Some NN designs, as a function of the network size, may exhibit sharp qualitative transitions in behavior at certain critical sizes.

- **Training data extraction attacks**
  - Important for sensitive applications of generative models, such as in the medical sector.
  - For a security testing study targeting diffusion models, see (Carlini et al., 2023, https://arxiv.org/abs/2301.13188)

What's next?

- **Publish the PDE model** in our upcoming book (in press): *Fundamental Mathematical Modeling of Additive Manufacturing*, to appear in *Tracts in Additive Manufacturing*, *Springer*.

- Quantitatively **characterize VAE performance**. Some popular performance statistics:
  - **AU** (active units), **NLL** (negative log-likelihood of data under trained model), **KL** (Kullback–Leibler divergence of variational posterior from latent prior), **MI** (mutual information between data **x** and code **z**).
  - Implemented; next step: validation.
    - Numbers from continuous-Bernoulli case not comparable with discrete Bernoulli.
    - Need to re-train with discrete Bernoulli observation model to compare against readily available known values.
    - Network details differ, the training is stochastic, and the MC estimates in the performance statistics add some noise. But the results should be in the same ballpark.

- **Facilitate AI debugging**, by visualizing what the CNN filters have learned:
  - Plot convolution kernels, plot activation maps, extract dataset images that maximally activate a specific channel, occlude different regions of input to produce a localization map of features that mattered.
    https://cs231n.github.io/understanding-cnn/
  - Use gradient descent to derive an abstract image that maximally activates a specific channel.
    https://keras.io/examples/vision/visualizing_what_convnets_learn/

- Add physically informed loss, produce von Kármán vortex shedding simulation data via FeniCS, scale up the VAE.
- Start developing the accelerator component for latent-space time evolution.
- Get an eGPU to run larger AI models on.
- Set some time aside to upgrade to TensorFlow 2.13 and fix any induced regressions.

Thank you for your attention!

juha.jeronen@jamk.fi

jamk

References [1/3] – Part 1 – PDE model

- Nikolay Banichuk, Alexander Barsuk, Juha Jeronen, Pekka Neittaanmäki, and Tero Tuovinen. 2020. *Stability of Axially Moving Materials*. Springer, Solid Mechanics and Its Applications, vol. 259. ISBN 978-3-030-23803-2, doi:10.1007/978-3-030-23803-2

- M. Kurki, J. Jeronen, T. Saksa, and T. Tuovinen. 2016. The origin of in-plane stresses in axially moving orthotropic continua. *International Journal of Solids and Structures*. doi:10.1016/j.ijsolstr.2015.10.027

- John O. Milewski. 2017. *Additive Manufacturing of Metals*. Springer Series in Materials Science, vol. 258. ISBN 978-3-319-58205-4, doi:10.1007/978-3-319-58205-4

- A. D. Polyanin and V. F. Zaitsev. 2003. *Handbook of Exact Solutions for Ordinary Differential Equations*. Chapman & Hall/CRC Press, Boca Raton. 2nd edition.

- Rudolf Skutch. 1897. Über die Bewegung eines gespannten Fadens, weicher gezwungen ist durch zwei feste Punkte, mit einer constanten Geschwindigkeit zu gehen, und zwischen denselben in Transversal-Schwingungen von gerlinger Amplitude versetzt wird. *Annalen der Physik und Chemie* **61**, 190–195.

jamk

References [2/3] – Part 2 – AI acceleration

- Shaofeng Cai, Yao Shu, Gang Chen, Beng Chin Ooi, Wei Wang, and Meihui Zhang. 2020. Effective and Efficient Dropout for Deep Convolutional Neural Networks. https://arxiv.org/abs/1904.03392
- Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting Training Data from Diffusion Models. https://arxiv.org/abs/2301.13188
- Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei. 2019. Avoiding Latent Variable Collapse With Generative Skip Models. https://arxiv.org/abs/1807.04863
- Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. https://arxiv.org/abs/1506.02142
- M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. 2022. Ensemble deep learning: A review. https://arxiv.org/abs/2104.02395
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press. http://www.deeplearningbook.org
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015a. Deep Residual Learning for Image Recognition. https://arxiv.org/abs/1512.03385v1
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015b. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. https://arxiv.org/abs/1502.01852
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. https://arxiv.org/abs/1207.0580
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. https://arxiv.org/abs/1609.04836
- Diederik P Kingma, Max Welling. 2013. Autoencoding variational Bayes. https://arxiv.org/abs/1312.6114
- Diederik P Kingma, Max Welling. 2019. An Introduction to Variational Autoencoders. https://arxiv.org/abs/1906.02691
- Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. 2021. Survey of Dropout Methods for Deep Neural Networks. https://arxiv.org/abs/1904.13310
- Benjamin Lengerich, Eric P. Xing, and Rich Caruana. 2022. Dropout as a Regularizer of Interaction Effects. https://arxiv.org/abs/2007.00823
- Shuyu Lin, Stephen Roberts, Niki Trigoni, and Ronald Clark. 2019. Balancing reconstruction quality and regularisation in evidence lower bound for variational autoencoders. https://arxiv.org/abs/1909.03765

jamk

References [3/3] – Part 2 – AI acceleration

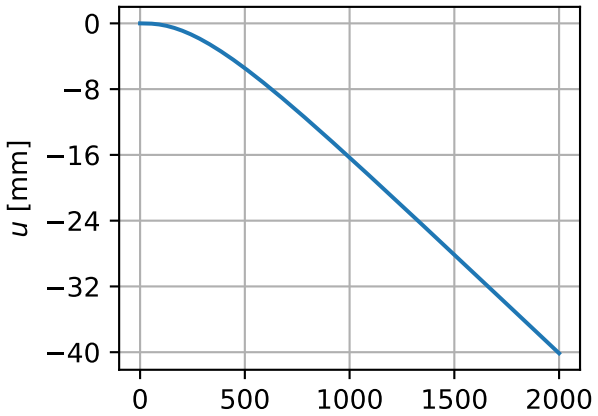
- Gabriel Loaiza-Ganem and John P. Cunningham. 2019.
  The continuous Bernoulli: fixing a pervasive error in variational autoencoders. https://arxiv.org/abs/1907.06845
- Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016.
  Conditional Image Generation with PixelCNN Decoders. https://arxiv.org/abs/1606.05328
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015.
  U-Net: Convolutional Networks for Biomedical Image Segmentation. https://arxiv.org/abs/1505.04597
- Jerome Rothstein. 1974. Loschmidt's and Zermelo's Paradoxes Do Not Exist. *Foundations of Physics* **4**(1), 83–89.
- Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2023.
  The Curse of Recursion: Training on Generated Data Makes Models Forget. https://arxiv.org/abs/2305.17493
- Samarth Sinha and Adji B. Dieng. 2022. Consistency Regularization for Variational Auto-Encoders. https://arxiv.org/abs/2105.14859
- Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. https://arxiv.org/abs/1506.01186
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR **15**(56):1929−1958. https://jmlr.org/papers/v15/srivastava14a.html
- Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. 2019.
  Variational Autoencoder with Implicit Optimal Priors. https://arxiv.org/abs/1809.05284
- Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. 2015.
  Efficient Object Localization Using Convolutional Networks. https://arxiv.org/abs/1411.4280
- Yuxin Wu and Kaiming He. 2018. Group Normalization. https://arxiv.org/abs/1803.08494
- Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. 2017.
  Tackling Over-pruning in Variational Autoencoders. https://arxiv.org/abs/1706.03643

jamk

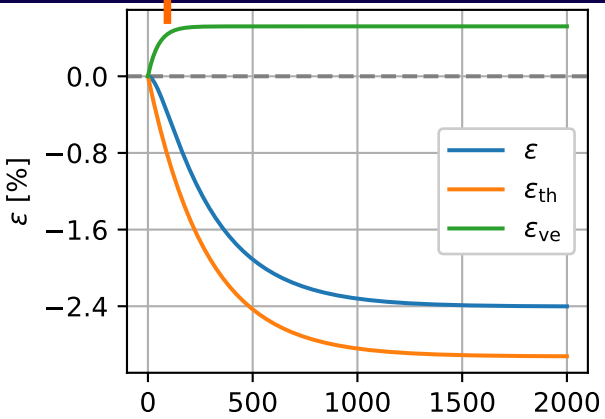
Appendix to Part 1: 1D results, very high viscosity comparison

JJ, TT, MK. One-Dimensional Thermomechanical Model for Additive Manufacturing Using Laser-Based Powder Bed Fusion. Computation 2022, 10(6), 83, doi: 10.3390/computation10060083

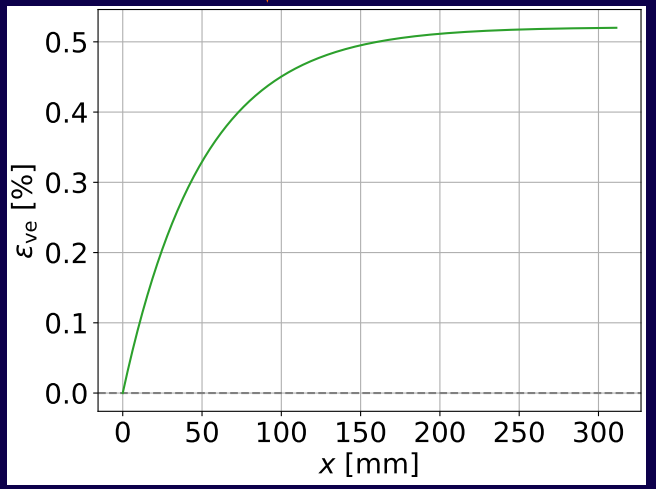
Axial displacement w.r.t. constant-velocity translation



Axial strain



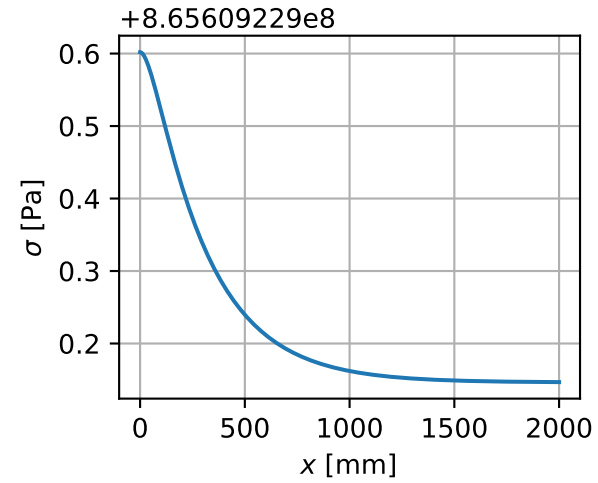
total = thermal + viscoelastic



Viscoelastic strain.

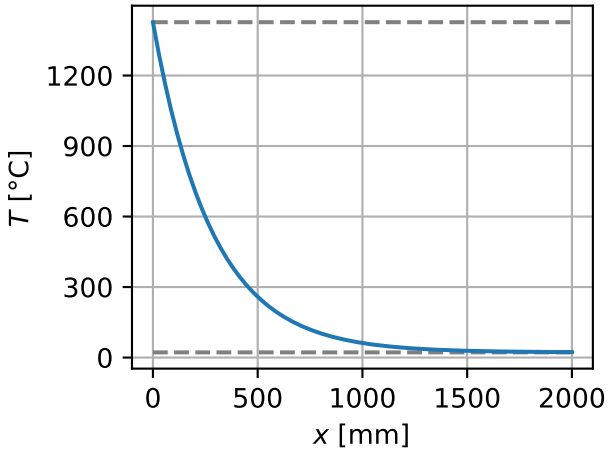
Note **high stress level**.

Laser scan speed v = 50 mm/s.



Axial stress

solidus temp.



Temperature

room temp.

jamk

Appendix to Part 1: Independent nondimensional parameters

JJ, TT, MK. One-Dimensional Thermomechanical Model for Additive Manufacturing Using Laser-Based Powder Bed Fusion. Computation 2022, 10(6), 83, doi: 10.3390/computation10060083

- The 1D model, with a Kelvin–Voigt material, is parameterized by ρ (density), E (Young modulus), η (viscosity), L (length of domain), v (laser velocity), b_0 (reference acceleration of external body force), k (heat conductivity), c (specific heat capacity), α (thermal expansion coefficient), and T_0 (reference temperature).

- The model has **ten** dimensional parameters that refer to **four** base units (kg, m, s, K). Thus, **Buckingham's π theorem** tells us there exists a set of **six independent nondimensional parameters**. One physically meaningful choice is:

$a_1 = \frac{\rho v^2}{E}$	Inertial modulus vs. elastic modulus
$a_2 = \frac{E L}{\eta v}$	Laser travel time L / v (one domain length) vs. Kelvin–Voigt retardation time η / E
$a_3 = \frac{L b_0}{v^2}$	Laser travel time L / v vs. characteristic time of external acceleration v / b_0 (how many seconds for one additional v of speed)
$a_4 = \frac{k}{c \eta}$	Thermal vs. mechanical viscosity
$a_5 = \alpha T_0$	Thermal expansion factor per unit nondimensional temperature
$a_6 = \frac{k c}{\eta \alpha^2} \left(\sqrt{E/\rho}\right)^{-4}$	Thermoviscous vs. longitudinal elastic wave characteristic velocity? (to the fourth power)

- Five of these (a_1 through a_5) appear in the nondimensional equations in a steady-state analysis.