Frameworks for DCE with gradient based optimisation

Matthew Macaulay
Physics Informed Machine Learning

1. PDE solvers
   - Geometry
   - Boundary Conditions
   - Flow Equations

2. Physics informed ML
   - Some Physics
   - Some Machine Learning

3. Need Big data
   - Machine Learning

Data
ML Framework for Engineering

Classical physics

Model
\[ z = M_\theta (x') \]

Physical:
PDE, CFD, DEM, FEM

Bayesian inference of latent variables
\[ p(z|x) \]

E.g. velocity field, energy distribution

Loss function
\[ L(z) \]

E.g. minimise energy, range of temperature

Input data
\[ x' \]
E.g. input controls, geometry

Real world

Digital twin

Prior
ML Framework for Engineering
Classical physics + machine learning

Measured data \( x \)
E.g. sensors, lab

Input data \( x' \)
E.g. input controls, geometry

Real world

Model
\[ z = M_\theta(x, x') \]

Physical:
PDE, CFD, DEM, FEM

ML: NN, Lasso regression

Bayesian inference of latent variables
\[ p(z|x) \]
E.g. velocity field, energy distribution

Prior

Loss function
\[ \mathcal{L}(z) \]
E.g. minimise energy, range of temperature

Digital twin
**ML Framework for Engineering**

**Optimise controls**

- **Measured data** $x$
  
  E.g. sensors, lab

- **Input data** $x'$
  
  E.g. input controls, geometry

- **Model** $z = M_0(x, x')$
  
  Physical: PDE, CFD, DEM, FEM
  
  ML: NN, Lasso regression

- **Bayesian inference of latent variables** $p(z|x)$
  
  E.g. velocity field, energy distribution

- **Loss function** $\mathcal{L}(z)$
  
  E.g. minimise energy, range of temperature

\[
\frac{\partial C}{\partial x'} = \frac{\partial z}{\partial x'} \quad \frac{\partial \mathcal{L}}{\partial z}
\]
Optimisation Framework
AutoDiff with PyTorch

- Analytical derivatives: chain rule as code runs
- Complex models
- Gradient-based optimisation
- Probabilistic models by re-parameterisation trick

\[
\frac{\partial C}{\partial x'} = \frac{\partial z}{\partial x'} \frac{\partial L}{\partial z}
\]
Opportunities for DCE
Physics informed neural net: Solve PDEs with NN

- Solve for all possible parameters
- Include data
- Design NN architecture to satisfy physical laws
- Can NNs always find good solutions?
- Uncertainty propagation through PDE
- Model calibration

\[ \mathcal{L} = w_{\text{data}} \mathcal{L}_{\text{data}} + w_{\text{PDE}} \mathcal{L}_{\text{PDE}} \]
Opportunities for DCE

Application to Digital twins

- Develop or extend a python package
  - Specifically for digital twins
  - Real-time data
  - Probabilistic or deterministic models and NN
Demonstrator Ideas
In Mineral Processing
Theory

Adapt physics informed neural nets to grains

- NN for grain kinematics (as opposed to PDEs)
  - Grain specific loss function
- Inverse problem: what is the physical law or material?
  - E.g. geotechnical: porosity, permeability, strength
  - E.g. transient response

\[ \mathcal{L} = w_{\text{data}} \mathcal{L}_{\text{data}} + w_{\text{PDE}} \mathcal{L}_{\text{PDE}} \]

\[ \mathcal{L} = w_{\text{data}} \mathcal{L}_{\text{data}} + w_{\text{Grain}} \mathcal{L}_{\text{Grain}} \]
Applications
Predict flow features

• Energy efficiency of mineral processing equipment
• Particle Mixers
• Powder “flowability”
• Granular wear and tear

Rock + slurry. Sinnott, 2017, Minerals Engineering (data 61)
Rotating drum
Optimise milling efficiency

1. Input: grain size, geometry, ...

2. Train NN to results of DEM.
   Predict: kinematics and/or energy
   [Option: Bayesian Variational Inference]

3. Optimise energy required by autodiff through NN - with gradients

\[ p^*(z) = p(z|x) \]

Posterior Distribution of Energy Efficiency

\( z = \text{controls e.g. rotating speed} \)
Rotating drum
Optimise milling efficiency

1. Input: grain size, geometry, ...

2. Train NN to results of DEM.
   Predict: kinematics and/or energy
   [Option: Bayesian Variational Inference]
   • Add sensors as inputs

3. Optimise energy required by auto-diff through NN - with gradients