Physics-Informed Neural Networks (PINNs) for Solid Mechanics

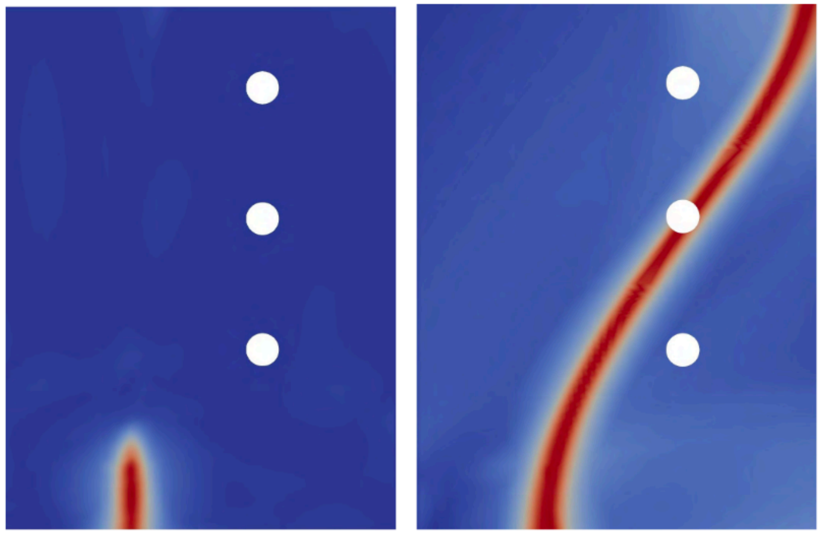
Eric M. Stewart

Final Project for 18.337

Massachusetts Institute of Technology

Background

- In my field of research — **computational solid mechanics** — the Finite Element Method (FEM) reigns supreme among all PDE solution procedures.
  - In fact, FEM traces its roots in mechanics back **over 80 years** to variational elasticity papers by Hrennikoff (1941) and Courant (1943).
- In **the last 3 years** however, PINNs have mounted a challenge in the solid mechanics literature. Applications include:
  - Phase-field fracture mechanics (Goswami et al., 2020) **(fig. right)**
  - Small-strain elastodynamics (Rao et al., 2021)
  - Finite deformation hyperelasticity (Fuhg and Bouklas, 2022)
  - Finite deformation plasticity (Niu et al., 2023)



PINN simulation of crack propagation path in a plate with three holes.

- None of these applications has used Julia, instead using TensorFlow or PyTorch.
- The goal of my project is to construct solid mechanics PINNs using Julia and document my challenges and victories, drawing comparisons to FEM along the way.

PINNs in one slide

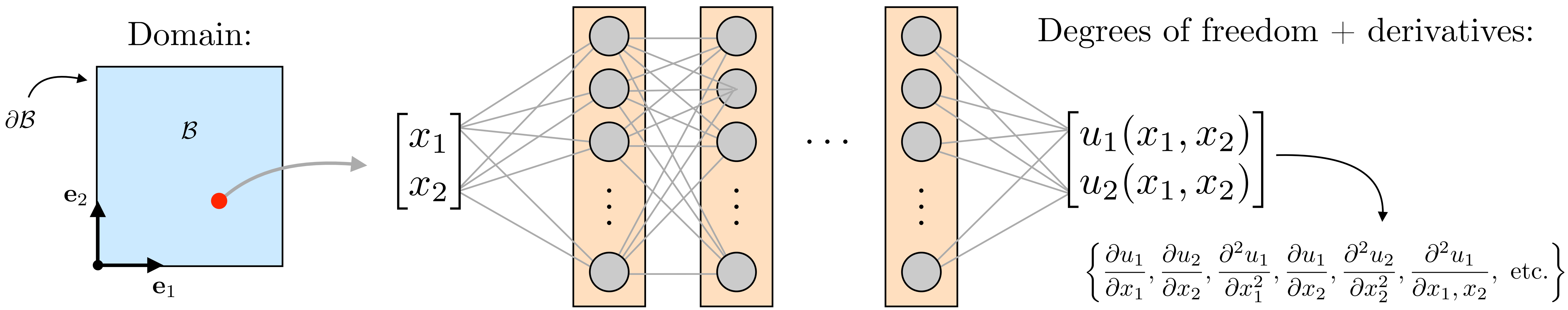
1. Loss function (the "PI" in **PI**NN): $\boxed{(\text{ ModelingToolkit.jl })}$

$$\mathcal{L} \overset{\text{def}}{=} \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BCs}}, \quad \text{where}$$

$$\mathcal{L}_{\text{PDE}} = \int_{\mathcal{B}} f\left(u_i, \frac{\partial u_i}{\partial x_i}, \frac{\partial u_i}{\partial x_j}, \frac{\partial^2 u_i}{\partial x_i^2}, \frac{\partial^2 u_i}{\partial x_i\, \partial x_j}, \text{etc.}\right) dv, \quad \text{and}$$

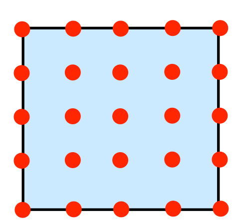
$$\mathcal{L}_{\text{BCs}} = \int_{\partial\mathcal{B}} (\text{ error in BCs }) \, da.$$

2. Differentiable Neural Network DOFs (the "NN" in PI**NN**): $\boxed{(\text{ Flux.jl / Lux.jl })}$



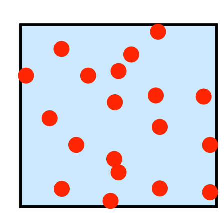
3. Estimate $\mathcal{L}(\mathbf{u})$.

$\boxed{(\text{ NeuralPDE.jl })}$



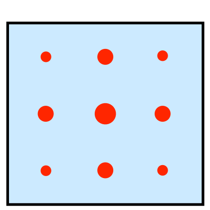
Grid methods

or



Quasi-random

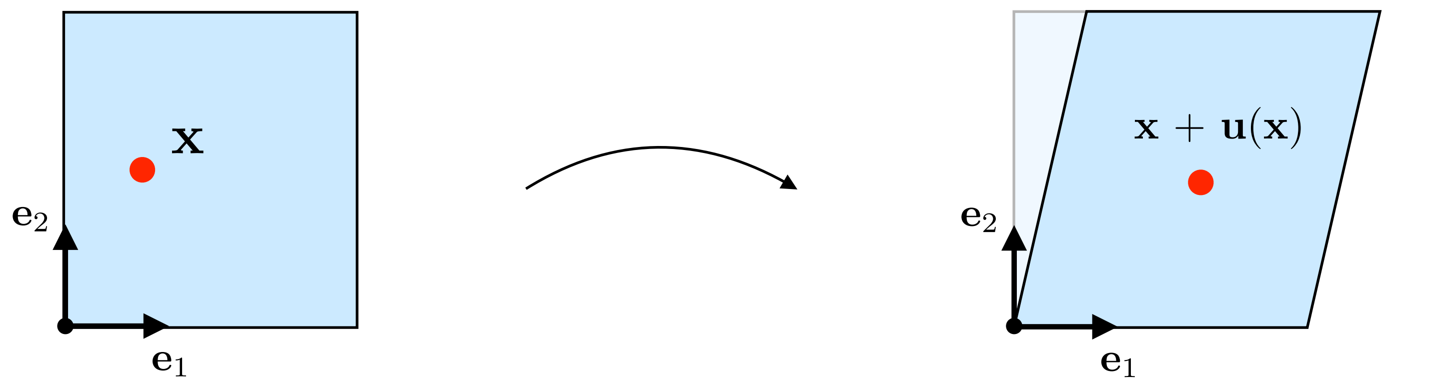
or



Gaussian quadrature

4. Find $\mathbf{u} = \arg\min_{\mathbf{u}} (\mathcal{L})$ using gradient descent, Adam, BFGS, etc. $\boxed{(\text{ Optimization.jl })}$

Continuum elasticity in one slide



Reference coordinates $\mathbf{x}$

Deformed coordinates $\mathbf{x} + \mathbf{u}(\mathbf{x})$

1. The symmetric **small strain tensor** is given by

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^\top\right].$$

2. The **constitutive relation** for stress is

$$\boldsymbol{\sigma} = 2\,G\,\boldsymbol{\varepsilon} + \left(K - \frac{2}{3}G\right)\mathrm{tr}\,(\boldsymbol{\varepsilon})\,\mathbf{1}$$

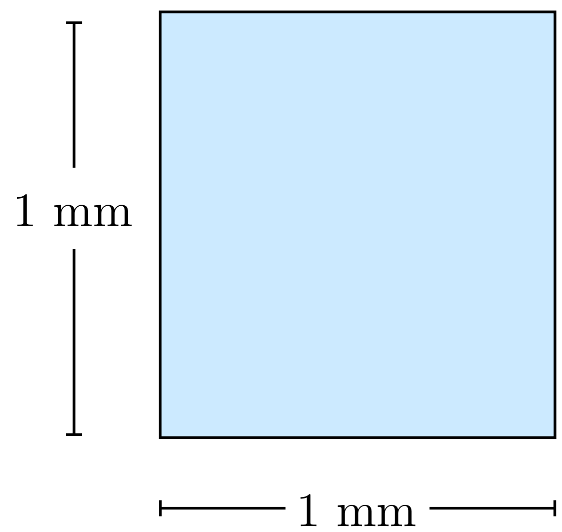
where G and K are the shear and bulk moduli respectively, which both have units of Pa.

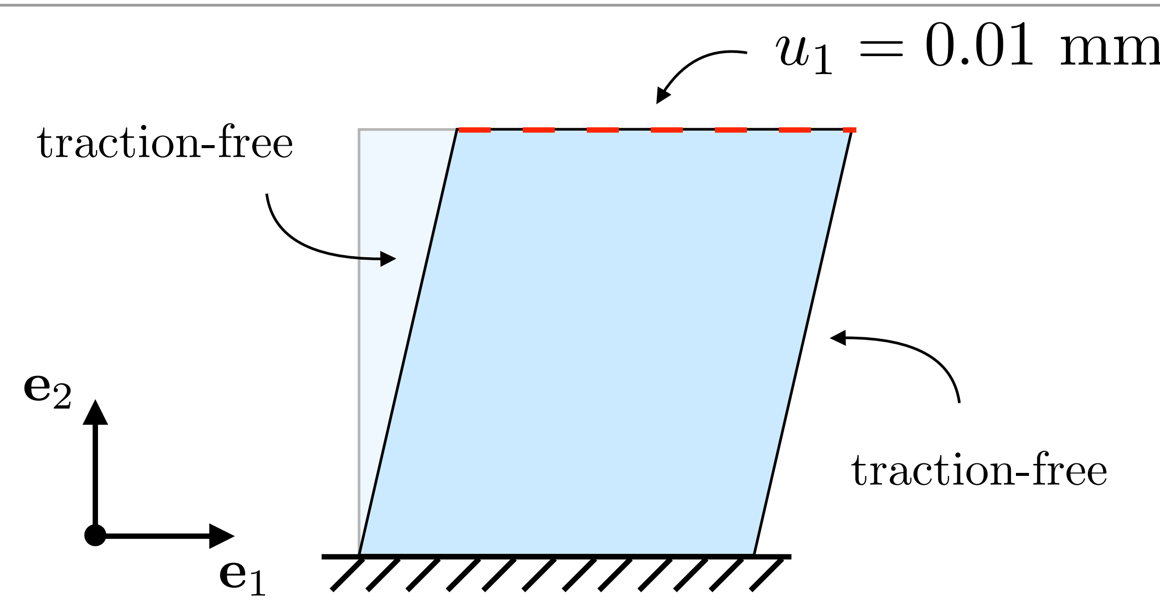
3. The **equation of motion** for continuum bodies is given by

$$\mathrm{div}\,\boldsymbol{\sigma} + \mathbf{b} = \rho\ddot{\mathbf{u}},$$

where $\mathbf{b}$ is a body force per unit volume e.g. gravity, and ρ is the mass density in kg/m^3.

Validation problem: Quasi-static simple shear





It's an intuitive picture, but complex to describe in terms of $\{u_1, u_2\}$:

$$
\underbrace{\begin{aligned}
\left(K + \frac{1}{3}G\right)\left(\frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_2}{\partial x_1\,\partial x_2}\right) + G\left(\frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_1}{\partial x_2^2}\right) &= 0 \\
\left(K + \frac{1}{3}G\right)\left(\frac{\partial^2 u_1}{\partial x_1\,\partial x_2} + \frac{\partial^2 u_2}{\partial x_2^2}\right) + G\left(\frac{\partial^2 u_2}{\partial x_1^2} + \frac{\partial^2 u_2}{\partial x_2^2}\right) &= 0
\end{aligned}}
$$

Governing PDEs in terms of **2nd derivatives of displacement field**.

$$
\left.\begin{aligned}
u_1(x,0) &= 0 \\
u_2(x,0) &= 0
\end{aligned}\right\} \text{Bottom edge fixed}
$$

$$
\left.\begin{aligned}
u_1(x,1) &= 0.01 \\
u_2(x,1) &= 0
\end{aligned}\right\} \text{Top edge shear}
$$

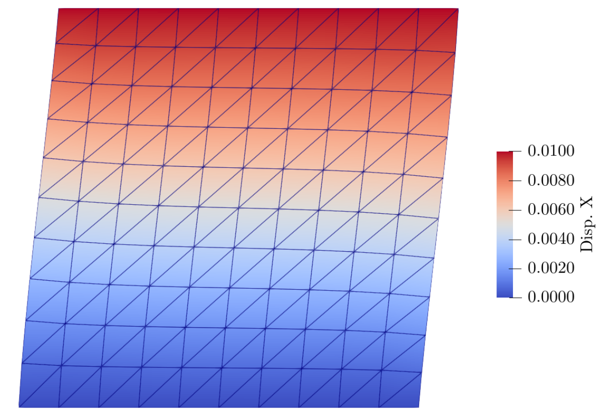
$$
\left.\begin{aligned}
&\left[\left(K - \frac{2}{3}G\right)\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + G\left(\frac{\partial u_1}{\partial x_1}\right)\right]\Bigg|_{(0,y)} = 0 \\
&G\left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2}\right)\Bigg|_{(0,y)} = 0
\end{aligned}\right\} \text{Traction-free left edge}
$$

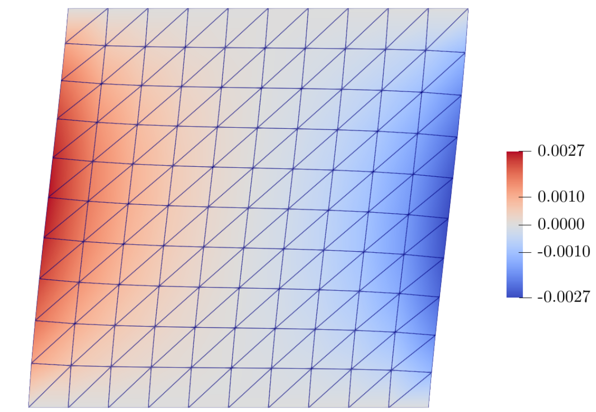
$$
\left.\begin{aligned}
&\left[\left(K - \frac{2}{3}G\right)\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + G\left(\frac{\partial u_1}{\partial x_1}\right)\right]\Bigg|_{(1,y)} = 0 \\
&G\left(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2}\right)\Bigg|_{(1,y)} = 0
\end{aligned}\right\} \text{Traction-free right edge}
$$

PINN solution versus FEM

$u_1(x_1, x_2)$ $\quad$ $u_2(x_1, x_2)$

FEM reference solution



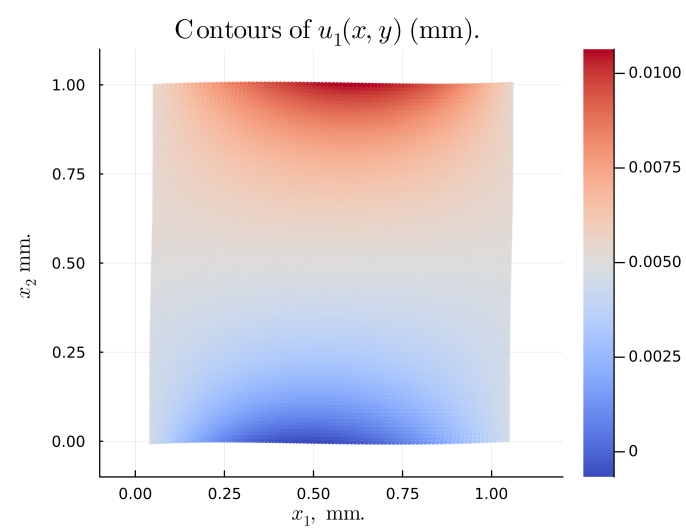


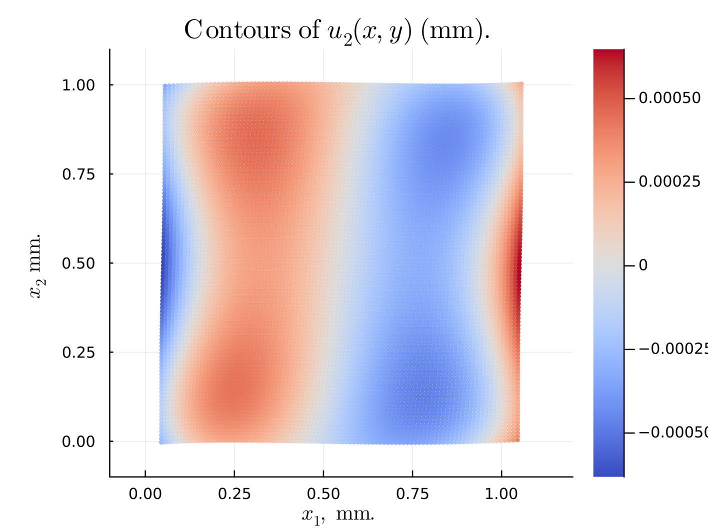
PINN solution

NNs are 2 x 20

Integral estimation:

quadrature method

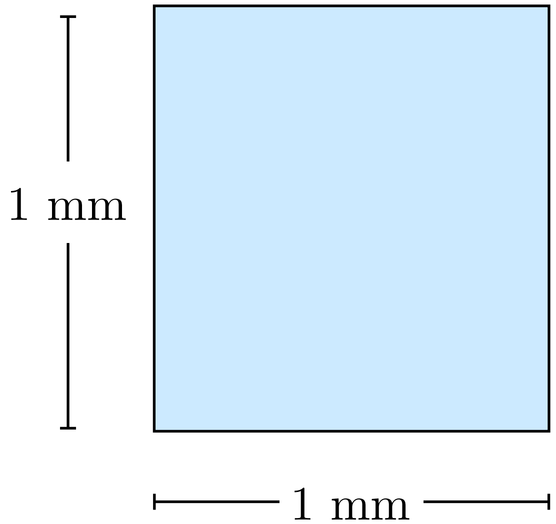


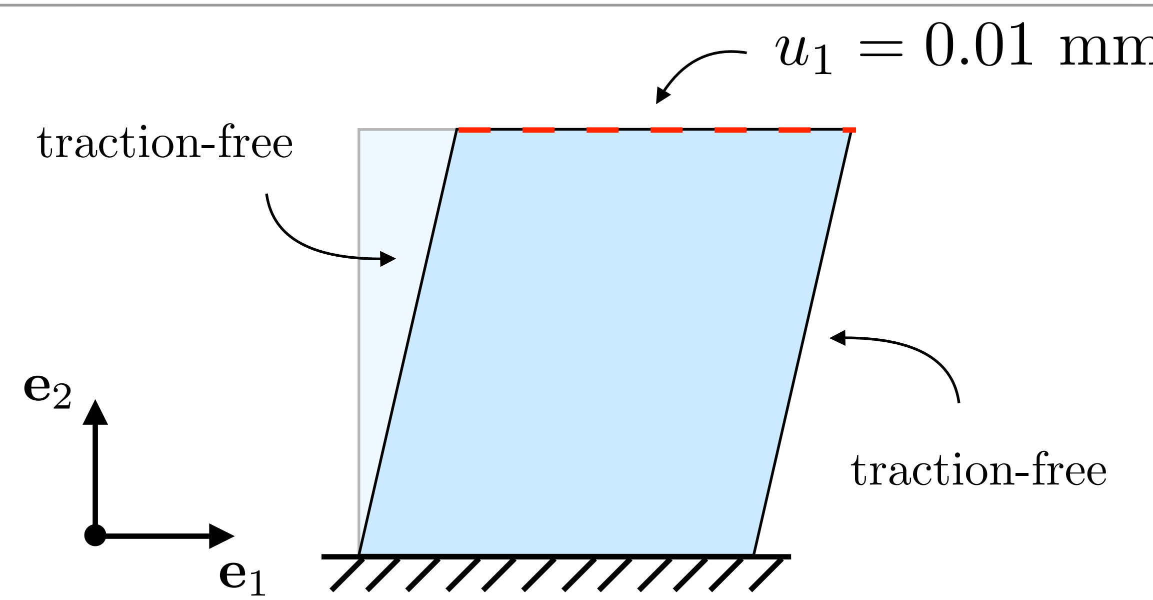


Loss converges, and seems reasonable.

But displacement contours are way off!

Recast as a “mixed formulation”





If we choose our DOFs as $\{u_1, u_2, \sigma_{11}, \sigma_{22}, \sigma_{12}\}$, the formulation simplifies:

$$\left.\begin{aligned} \frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{12}}{\partial x_2} &= 0 \\ \frac{\partial \sigma_{12}}{\partial x_1} + \frac{\partial \sigma_{22}}{\partial x_2} &= 0 \end{aligned}\right\}$$

Governing PDEs in terms of **1st derivatives of stresses.**

$$\left.\begin{aligned} \sigma_{11} &= \left(K + \frac{1}{3}G\right)\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + 2\,G\,\frac{\partial u_1}{\partial x_1} \\ \sigma_{22} &= \left(K + \frac{1}{3}G\right)\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right) + 2\,G\,\frac{\partial u_2}{\partial x_2} \\ \sigma_{11} &= G\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right) \end{aligned}\right\}$$

Stresses in terms of **1st derivatives of displacements.**

$$\left.\begin{aligned} u_1(x,0) &= 0 \\ u_2(x,0) &= 0 \end{aligned}\right\}$$ Bottom edge fixed

$$\left.\begin{aligned} u_1(x,1) &= 0.01 \\ u_2(x,1) &= 0 \end{aligned}\right\}$$ Top edge shear

$$\left.\begin{aligned} \sigma_{11}(0,y) &= 0 \\ \sigma_{12}(0,y) &= 0 \end{aligned}\right\}$$ Traction-free left edge

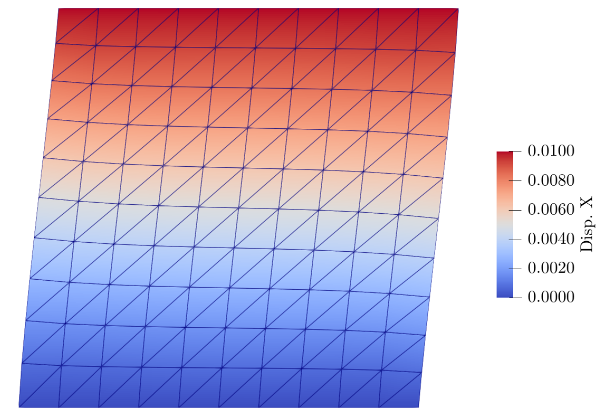
$$\left.\begin{aligned} \sigma_{11}(1,y) &= 0 \\ \sigma_{12}(1,y) &= 0 \end{aligned}\right\}$$ Traction-free right edge

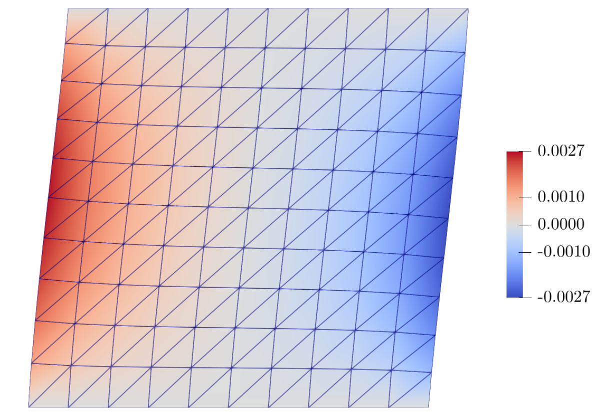
PINN solution versus FEM

$u_1(x_1, x_2)$ $u_2(x_1, x_2)$

FEM reference solution

FEniCS PROJECT



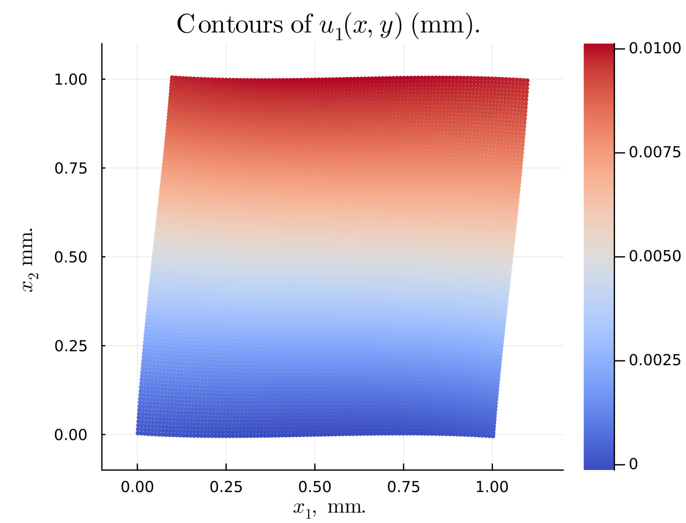


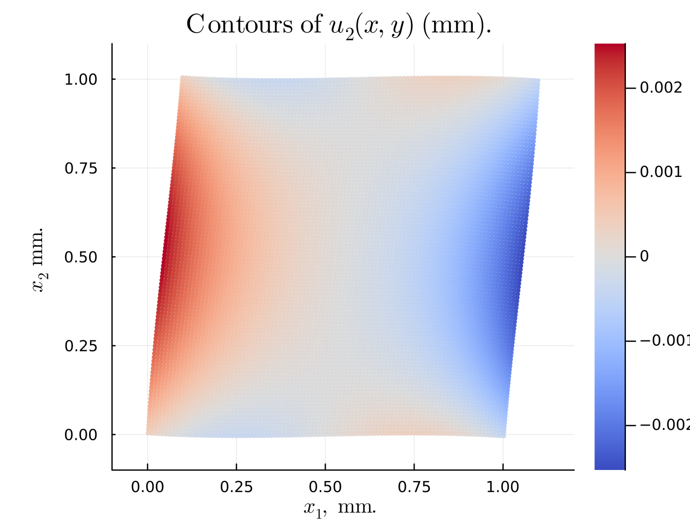
PINN solution

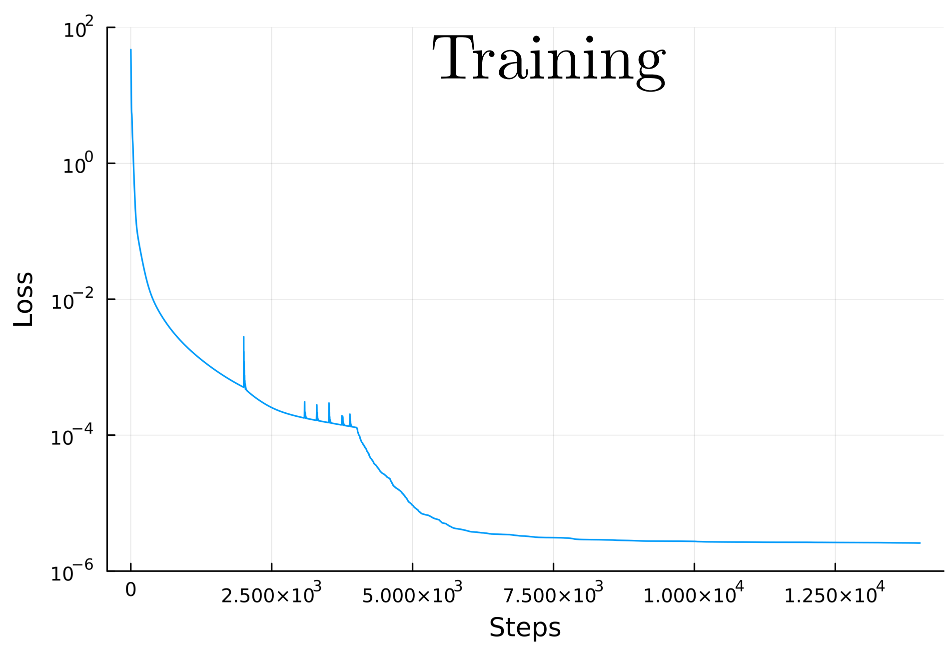
NNs are 2 x 20

Integral estimation:

quadrature method







"Mixed formulation" contours look much much better!

u_1 RMS Error: 5.962e-5 mm $\approx$ 60 nm

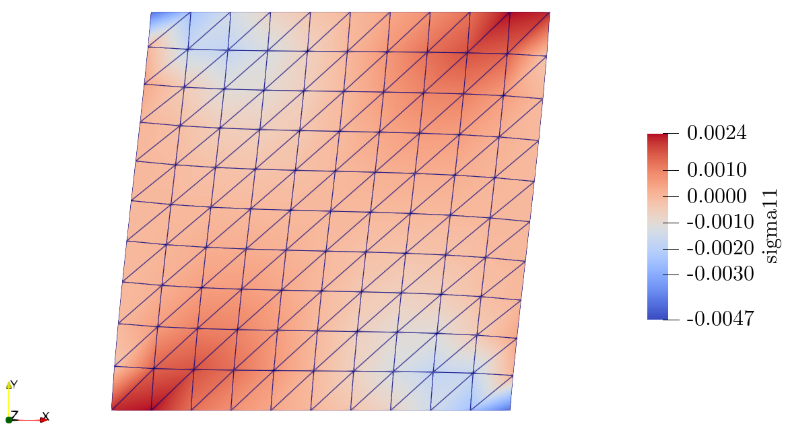
u_2 RMS Error: 1.217e-4 mm $\approx$ 120 nm

$\boxed{\text{Avoiding second derivatives seems to help a lot.}}$

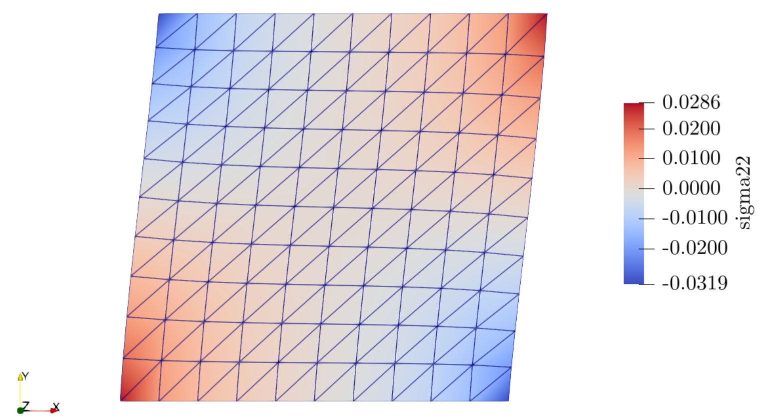
PINN solution versus FEM - stress fields

FEM reference solution

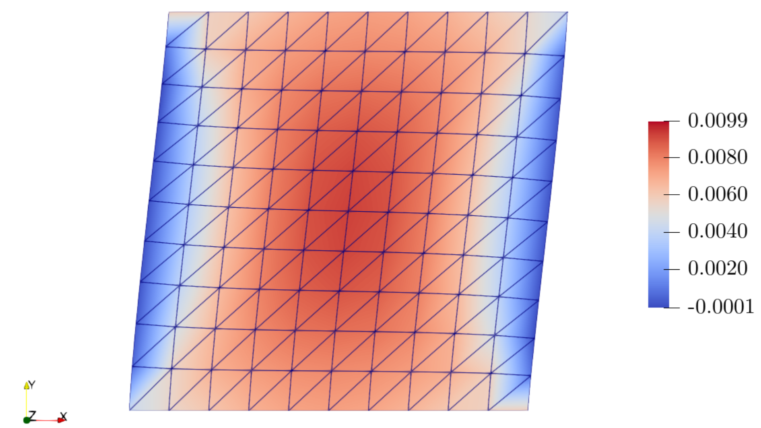
$\sigma_{11}(x_1, x_2)$



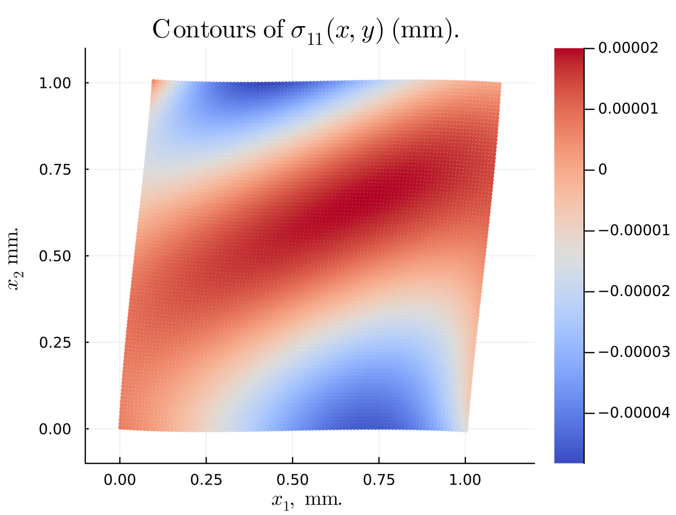
$\sigma_{22}(x_1, x_2)$

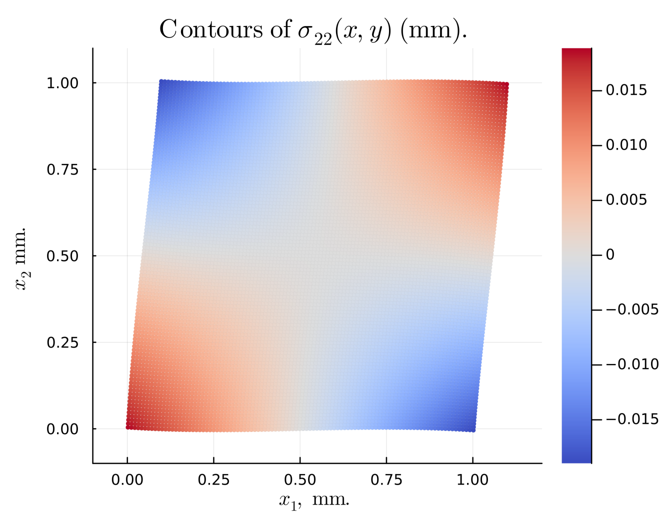


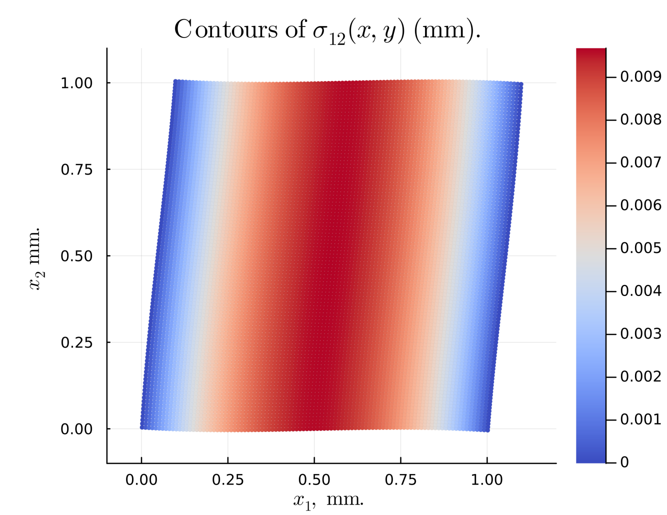
$\sigma_{12}(x_1, x_2)$



PINN solution



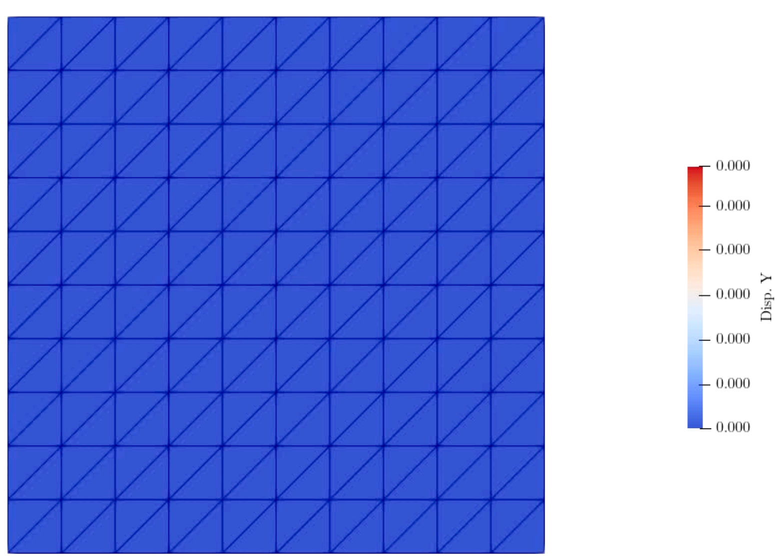




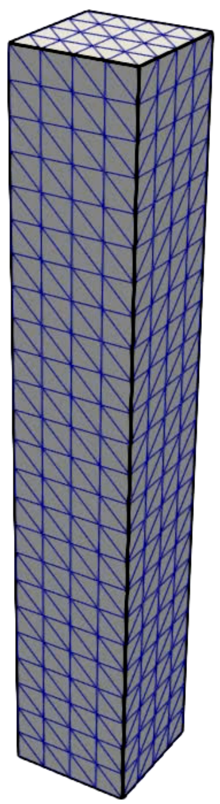
The max shear stress $\sigma_{12}^{\max}$ matches the theoretical result 0.01 MPa very well.

Next steps in progress

- In my proposal I had ambitious goals to include inertial effects, nonlinear kinematics, and rubber elastic constitutive behavior in my PINN.
    - Demonstrated in movies below.
- Progress has been much slower than expected (ML is an art), but I hope to include at least one more of these interesting behaviors in my final report.



Small strain elastodynamics in FEM



Finite strain hyperelastic dynamics in FEM

Concluding remarks

- The project has been a valuable learning experience for me about Julia and PINNs.

- I leveraged existing Julia tools a lot, and was still surprised by how lengthy and difficult it was to set up a PINN and get reasonable comparisons to FEM results.
  - PINN convergence and accuracy is **highly sensitive to order of derivatives,** as well as NN architecture, integral estimation method, and training method.
  - In terms of performance, the PINN trains in around 30 minutes versus 1 second to obtain the FEM result.

- Although PINNs are a bit cumbersome, there are some contexts where PINNs have an advantage over FEM since
  - PINNs are mesh-free.
  - PINNs have no explicit time discretization.

Which could be helpful in resolving e.g. stress singularities at a crack tip or shocks in solids.

Thank you!