Enhancing Simulations with Validation Data in NVIDIA Modulus Sym

During my internship at AnK, one of the most interesting tasks I worked on was integrating validation data into models built using NVIDIA Modulus Sym. This process was crucial for improving the accuracy and performance of the simulations, and it taught me a lot about bridging commercial simulation software with AI-model.

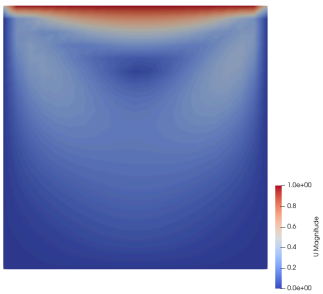
The Why and How of Validation

Validation data plays a vital role in ensuring that simulation results align with reality. NVIDIA Modulus Sym provides a handy `PointwiseValidator` for this purpose, but the real challenge lies in extracting accurate data from commercial solvers and formatting it for use in the model.

Diving into the Process

To demonstrate, I tackled two problems during my internship:

1. Simulating **lid-driven cavity flow** using OpenFOAM.
2. Solving a **transient heat distribution problem** on a 2D plate using Ansys.



Both problems required running simulations in their respective software, extracting results, and integrating them into Modulus Sym.

Learning from the Challenges

Extracting data in `.csv` format was straightforward in OpenFOAM, thanks to ParaView's intuitive interface. In Ansys, the process was slightly more complex but manageable. The real learning came when I had to map these `.csv` files to Modulus Sym's input and output variables. I also had to align the origins of the simulation data with the Modulus geometry, which involved some trial and error.

Here’s a snippet of the Python code I used:

```
from modulus.sym.domain.validator import PointwiseValidator

file_path = "/content/drive/MyDrive/Ank_internship/LDC/ldc.csv"  # Update with correct path
if os.path.exists(to_absolute_path(file_path)):
    mapping = {
        "Points:0": "x",
        "Points:1": "y",
        "U:0": "u",
        "U:1": "v",
        "p": "p"
    }
    openfoam_var = csv_to_dict(to_absolute_path(file_path), mapping)

    # Adjust origin to align with Modulus geometry
    openfoam_var["x"] += -width / 2
    openfoam_var["y"] += -height / 2

    # Separate input (invar) and output (outvar) variables
    openfoam_invar_numpy = {key: value for key, value in openfoam_var.items() if key in ["x", "y"]}
    openfoam_outvar_numpy = {key: value for key, value in openfoam_var.items() if key in ["u", "v"]}

    # Create Pointwise Validator
    openfoam_validator = PointwiseValidator(
        nodes=nodes,
        invar=openfoam_invar_numpy,
        true_outvar=openfoam_outvar_numpy,
        batch_size=1024,
    )

    # Add validator to domain
    ldc_domain.add_validator(openfoam_validator)
```

I found it fascinating how a small alignment tweak in the coordinates could make such a significant difference in the results!

Results

The following results were seen when the actual and predicted value were compared.

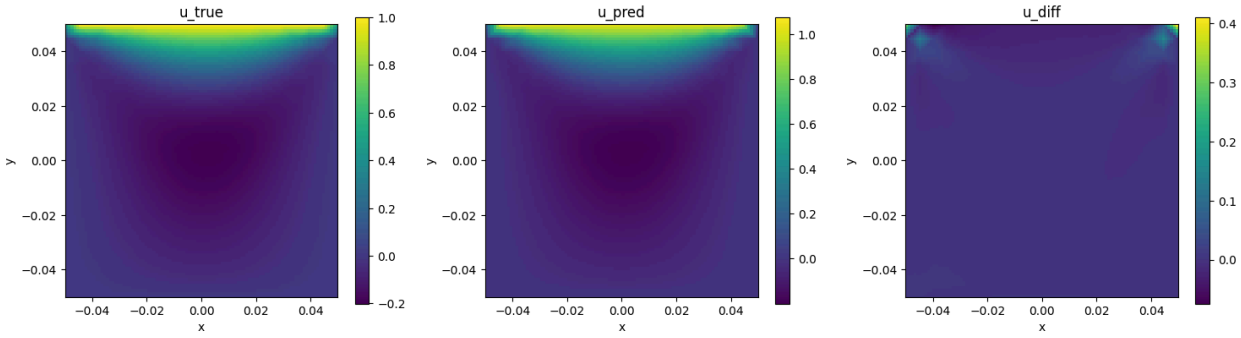


Fig- Comparison of AI prediction with the simulation results from commercial software.

Why This Matters

This task was more than just coding; it was about understanding the nuances of integrating traditional computational fluid dynamics with AI-based simulations. It highlighted how technology like NVIDIA Modulus Sym can bridge the gap between different domains, enabling smarter, faster problem-solving.

Takeaways

Looking back, this project was a fantastic opportunity to combine theory with practice. It taught me not only technical skills like Python scripting and data mapping but also how to approach problem-solving systematically. I now have a deeper appreciation for how AI can enhance engineering workflows, and I'm excited to apply these lessons to future challenges.

Detailed Tutorial

The detailed tutorial of this problem can be found at : Blog_validator_1