TFM_PIKAN_GPU

June 20, 2025

[1]:
```
!pip install tensorboard
```

```
Collecting tensorboard
  Using cached tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Collecting absl-py>=0.4 (from tensorboard)
  Using cached absl_py-2.3.0-py3-none-any.whl.metadata (2.4 kB)
Collecting grpcio>=1.48.2 (from tensorboard)
  Downloading grpcio-1.73.0-cp312-cp312-win_amd64.whl.metadata (4.0 kB)
Collecting markdown>=2.6.8 (from tensorboard)
  Downloading markdown-3.8.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: numpy>=1.12.0 in
c:\users\jorge\anaconda3\envs\kan_pytorch\lib\site-packages (from tensorboard)
(2.0.1)
Requirement already satisfied: packaging in
c:\users\jorge\anaconda3\envs\kan_pytorch\lib\site-packages (from tensorboard)
(24.2)
Collecting protobuf!=4.24.0,>=3.19.6 (from tensorboard)
  Using cached protobuf-6.31.1-cp310-abi3-win_amd64.whl.metadata (593 bytes)
Requirement already satisfied: setuptools>=41.0.0 in
c:\users\jorge\anaconda3\envs\kan_pytorch\lib\site-packages (from tensorboard)
(75.8.0)
Requirement already satisfied: six>1.9 in
c:\users\jorge\anaconda3\envs\kan_pytorch\lib\site-packages (from tensorboard)
(1.17.0)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard)
  Using cached tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard)
  Using cached werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\jorge\anaconda3\envs\kan_pytorch\lib\site-packages (from
werkzeug>=1.0.1->tensorboard) (2.1.5)
Using cached tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
Using cached absl_py-2.3.0-py3-none-any.whl (135 kB)
Downloading grpcio-1.73.0-cp312-cp312-win_amd64.whl (4.3 MB)
   ---------------------------------------- 0.0/4.3 MB ? eta -:--:--
   ---------------------------------------- 4.3/4.3 MB 43.8 MB/s eta 0:00:00
Downloading markdown-3.8.2-py3-none-any.whl (106 kB)
Using cached protobuf-6.31.1-cp310-abi3-win_amd64.whl (435 kB)
```

```
Using cached tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Using cached werkzeug-3.1.3-py3-none-any.whl (224 kB)
Installing collected packages: werkzeug, tensorboard-data-server, protobuf,
markdown, grpcio, absl-py, tensorboard
Successfully installed absl-py-2.3.0 grpcio-1.73.0 markdown-3.8.2
protobuf-6.31.1 tensorboard-2.19.0 tensorboard-data-server-0.7.2 werkzeug-3.1.3
```

[2]:
```
import torch
from torch import autograd
from torch.utils.tensorboard import SummaryWriter
```

[3]:
```
device = "cuda:0" if torch.cuda.is_available() else "cpu"
```

[4]:
```
from tqdm import tqdm
import matplotlib.pyplot as plt
from kan import KAN, LBFGS

device = torch.device(device)
print("Using device:", device)

rho = torch.tensor(1.0, device=device, requires_grad=False)
nu = torch.tensor(0.01, device=device, requires_grad=False)
eps = torch.tensor(1e-8, device=device, requires_grad=False)

width, height = 10.0, 2.0
num_points_x, num_points_y = 100, 20

x = torch.linspace(0, width, num_points_x, device=device, requires_grad=False)
y = torch.linspace(0, height, num_points_y, device=device, requires_grad=False)
X, Y = torch.meshgrid(x, y, indexing='ij')
coordinates = torch.stack([X.flatten(), Y.flatten()], dim=1).to(device)
coordinates.requires_grad = True  # Ensure coordinates require grad

model = KAN(width=[2,3,3, 3], grid=5, k=10, grid_eps=1.0,
            noise_scale=0.25).to(device)

def batch_jacobian(func, x, create_graph=False):
    def _func_sum(x):
        return func(x).sum(dim=0)
    return autograd.functional.jacobian(_func_sum, x,␣
 ↪create_graph=create_graph).permute(1, 0, 2)

def batch_hessian(func, x):
    jacobian = batch_jacobian(func, x, create_graph=True)
    hessians = []
    for i in range(jacobian.size(1)):
```

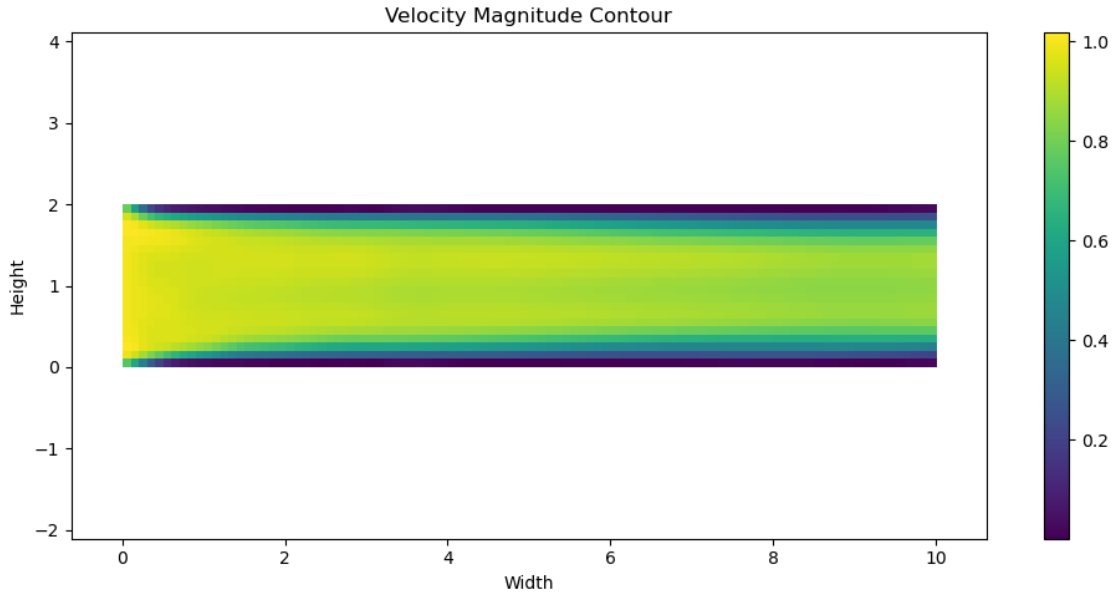
```
        grad = autograd.grad(jacobian[:, i].sum(), x, create_graph=True,␣
 ↪retain_graph=True)[0]
        hessians.append(grad.unsqueeze(1))
    return torch.cat(hessians, dim=1)

def navier_stokes_residuals(coords):
    coords = coords.clone().detach().requires_grad_(True)  # Ensure coords␣
 ↪require grad
    y_pred = model(coords)
    grads = batch_jacobian(model, coords, create_graph=True)
    hessians = batch_hessian(model, coords)

    u, v, p = y_pred[:, 0], y_pred[:, 1], y_pred[:, 2]
    u_x, u_y = grads[:, 0, 0], grads[:, 0, 1]
    v_x, v_y = grads[:, 1, 0], grads[:, 1, 1]
    p_x, p_y = grads[:, 2, 0], grads[:, 2, 1]

    u_xx, u_yy = hessians[:, 0, 0], hessians[:, 0, 1]
    v_xx, v_yy = hessians[:, 1, 0], hessians[:, 1, 1]

    continuity = u_x + v_y + eps * p
    x_momentum = u * u_x + v * u_y + (1 / rho) * p_x - nu * (u_xx + u_yy)
    y_momentum = u * v_x + v * v_y + (1 / rho) * p_y - nu * (v_xx + v_yy)

    no_slip_mask = (coords[:, 1] == 0) | (coords[:, 1] == height)
    inlet_mask = (coords[:, 0] == 0)
    outlet_mask = (coords[:, 0] == width)

    no_slip_loss = torch.mean(u[no_slip_mask] ** 2 + v[no_slip_mask] ** 2)
    inlet_loss = torch.mean((u[inlet_mask] - 1) ** 2)
    outlet_pressure_loss = torch.mean(p[outlet_mask] ** 2)

    bc_loss = no_slip_loss + inlet_loss + outlet_pressure_loss
    total_loss = torch.mean(continuity ** 2 + x_momentum ** 2 + y_momentum **␣
 ↪2) + bc_loss
    return total_loss

writer = SummaryWriter()

def train():
    optimizer = LBFGS(model.parameters(), lr=1,
                      history_size=10, line_search_fn="strong_wolfe",␣
 ↪tolerance_grad=1e-32, tolerance_change=1e-32, tolerance_ys=1e-32)

    steps = 20 # 20 steps are enough 200 in the original 2h9min version
    pbar = tqdm(range(steps), desc='Training Progress')
```

```
    for step in pbar:
        def closure():
            optimizer.zero_grad()
            loss = navier_stokes_residuals(coordinates)
            loss.backward()
            return loss

        optimizer.step(closure)
        if step % 5 == 0:
            current_loss = closure().item()
            pbar.set_description("Step: %d | Loss: %.3f" %
                                 (step, current_loss))
            writer.add_scalar('Loss/train', current_loss, step)

train()

writer.close()
```

```
Using device: cuda:0
checkpoint directory created: ./model
saving model version 0.0
```

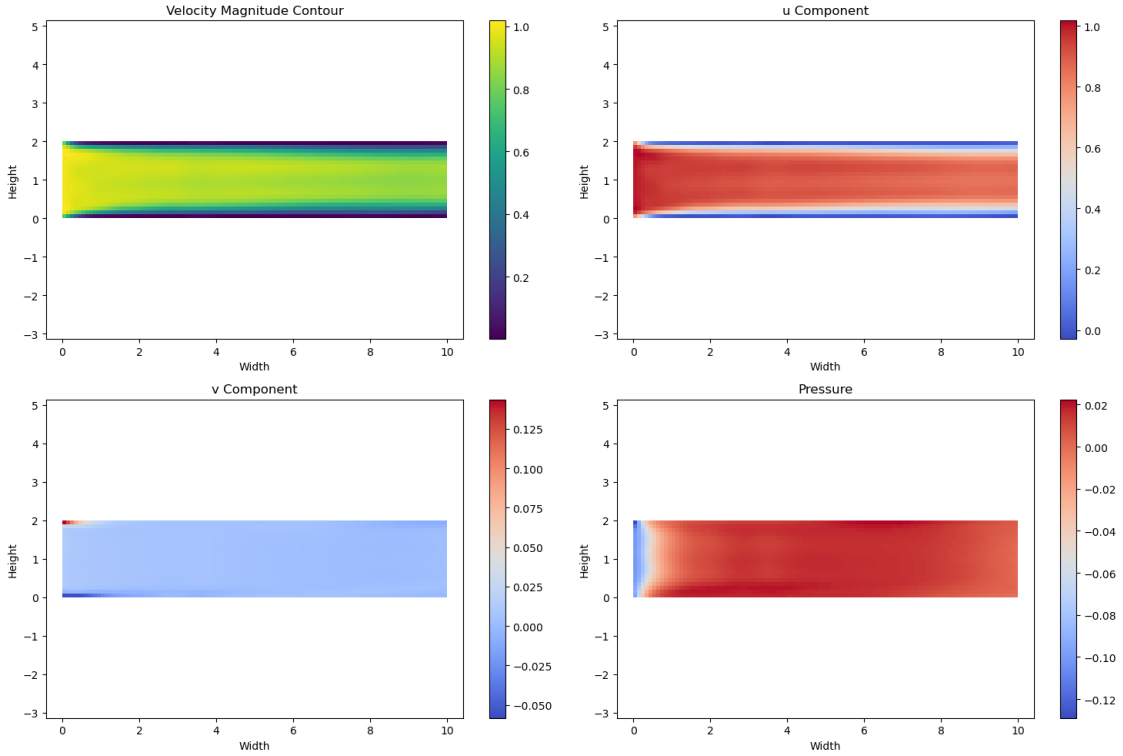
```
Step: 15 | Loss: 0.026: 100%|      | 20/20 [05:49<00:00, 17.48s/it]
```

[5]:
```
u_pred = model(coordinates)[:, 0].detach().reshape(
    num_points_x, num_points_y).T

v_pred = model(coordinates)[:, 1].detach().reshape(
    num_points_x, num_points_y).T


magnitude = torch.sqrt(u_pred ** 2 + v_pred ** 2)
magnitude = magnitude.detach().cpu().numpy()

plt.figure(figsize=(10, 5))  # Set the figure size as needed
plt.imshow(magnitude, extent=(0, width, 0, height), origin='lower',␣
 ↪cmap='viridis')
plt.colorbar()  # Add a colorbar to show the magnitude scale
plt.title('Velocity Magnitude Contour')
plt.xlabel('Width')
plt.ylabel('Height')
plt.axis('equal')  # Ensure the plot has equal scaling
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```

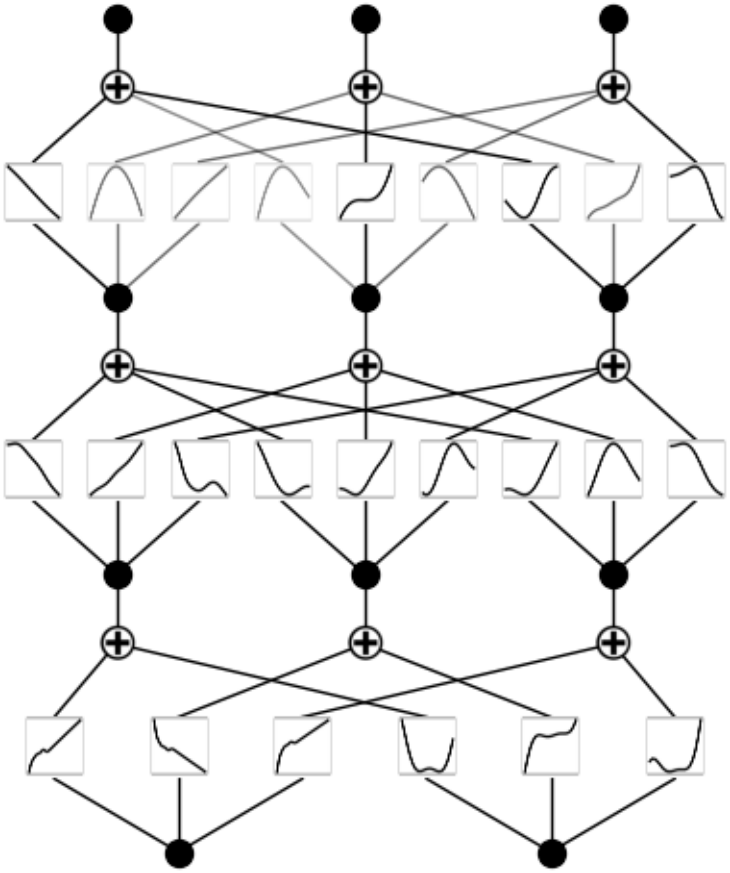


[6]:
```
# Extracting predictions
u_pred = model(coordinates)[:, 0].detach().reshape(num_points_x, num_points_y).T
v_pred = model(coordinates)[:, 1].detach().reshape(num_points_x, num_points_y).T
p_pred = model(coordinates)[:, 2].detach().reshape(num_points_x, num_points_y).T

# Velocity Magnitude
magnitude = torch.sqrt(u_pred ** 2 + v_pred ** 2)

magnitude = magnitude.detach().cpu().numpy()
u_pred = u_pred.detach().cpu().numpy()
v_pred = v_pred.detach().cpu().numpy()
p_pred = p_pred.detach().cpu().numpy()

# Plotting all subplots
fig, axs = plt.subplots(2, 2, figsize=(15, 10))

# Velocity Magnitude
im0 = axs[0, 0].imshow(magnitude, extent=(0, width, 0, height), origin='lower',␣
 ↪cmap='viridis')
fig.colorbar(im0, ax=axs[0, 0])
axs[0, 0].set_title('Velocity Magnitude Contour')
axs[0, 0].set_xlabel('Width')
axs[0, 0].set_ylabel('Height')
axs[0, 0].axis('equal')

# u Component
```

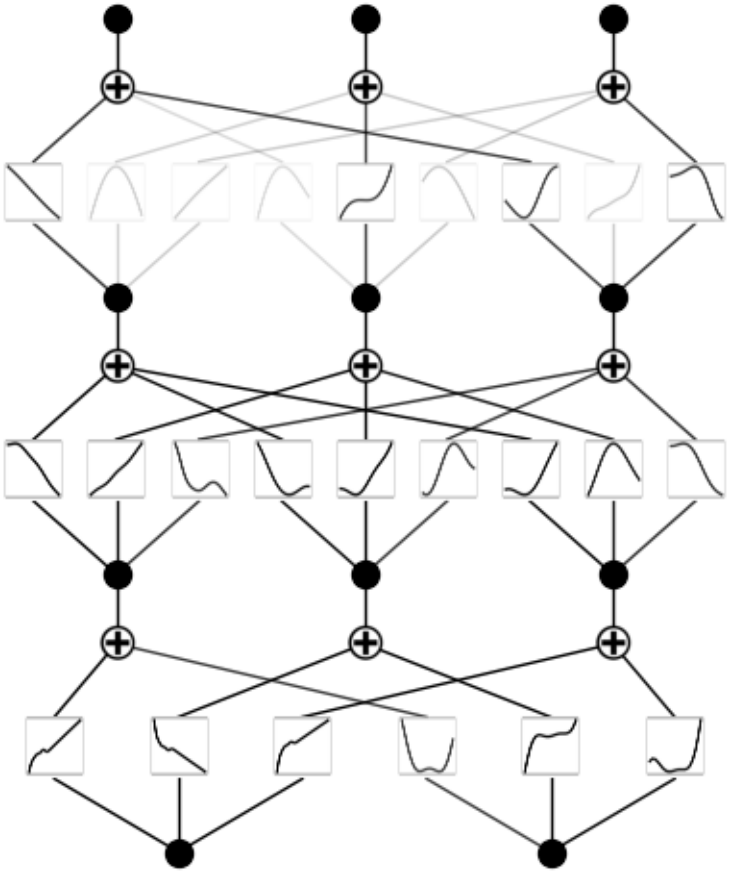
```
im1 = axs[0, 1].imshow(u_pred, extent=(0, width, 0, height), origin='lower',␣
 ↪cmap='coolwarm')
fig.colorbar(im1, ax=axs[0, 1])
axs[0, 1].set_title('u Component')
axs[0, 1].set_xlabel('Width')
axs[0, 1].set_ylabel('Height')
axs[0, 1].axis('equal')

# v Component
im2 = axs[1, 0].imshow(v_pred, extent=(0, width, 0, height), origin='lower',␣
 ↪cmap='coolwarm')
fig.colorbar(im2, ax=axs[1, 0])
axs[1, 0].set_title('v Component')
axs[1, 0].set_xlabel('Width')
axs[1, 0].set_ylabel('Height')
axs[1, 0].axis('equal')

# Pressure
im3 = axs[1, 1].imshow(p_pred, extent=(0, width, 0, height), origin='lower',␣
 ↪cmap='coolwarm')
fig.colorbar(im3, ax=axs[1, 1])
axs[1, 1].set_title('Pressure')
axs[1, 1].set_xlabel('Width')
axs[1, 1].set_ylabel('Height')
axs[1, 1].axis('equal')

plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```



[7]:
```
model.plot(beta=10)
```



[8]:
```
model.plot()
```



[9]:
```
model.auto_symbolic()
```

```
fixing (0,0,0) with x, r2=0.9735497236251831, c=1
fixing (0,0,1) with 0, r2=0.0, c=0
fixing (0,0,2) with 0, r2=0.0, c=0
fixing (0,1,0) with 0, r2=0.0, c=0
fixing (0,1,1) with 0, r2=0.0, c=0
fixing (0,1,2) with 0, r2=0.0, c=0
fixing (1,0,0) with sin, r2=0.9986881613731384, c=2
fixing (1,0,1) with x, r2=0.992422342300415, c=1
fixing (1,0,2) with 0, r2=0.0, c=0
fixing (1,1,0) with 0, r2=0.0, c=0
fixing (1,1,1) with 0, r2=0.0, c=0
fixing (1,1,2) with cos, r2=0.9962447285652161, c=2
fixing (1,2,0) with 0, r2=0.0, c=0
fixing (1,2,1) with 0, r2=0.0, c=0
fixing (1,2,2) with sin, r2=0.9975288510322571, c=2
fixing (2,0,0) with sin, r2=1.000000238418579, c=2
fixing (2,0,1) with sin, r2=0.9994560480117798, c=2
```

```
fixing (2,0,2) with x^2, r2=0.999975860118866, c=2
fixing (2,1,0) with 0, r2=0.0, c=0
fixing (2,1,1) with 0, r2=0.0, c=0
fixing (2,1,2) with cos, r2=0.9997799396514893, c=2
fixing (2,2,0) with 0, r2=0.0, c=0
fixing (2,2,1) with 0, r2=0.0, c=0
fixing (2,2,2) with 0, r2=0.0, c=0
saving model version 0.1
```

[10]:
```
from kan.utils import ex_round
```

[11]:
```
formula1, formula2, formula3 = model.symbolic_formula()[0]
```

[12]:
```
ex_round(formula1,2)
```

[12]:
$$1.08 \sin\left(1.71 \sin\left(0.17 x_1 - 4.46\right) + 8.52\right) + 0.48$$

[13]:
```
ex_round(formula2,2)
```

[13]:
$$-0.01 \sin\left(2.96 \sin\left(0.17 x_1 - 4.46\right) - 9.56\right)$$

[14]:
```
ex_round(formula3,2)
```

[14]:
$$-0.03 \left(1 - 0.42 \sin\left(0.17 x_1 - 4.46\right)\right)^2 - 0.05 \cos\left(0.12 x_1 + 8.95\right) - 0.01$$