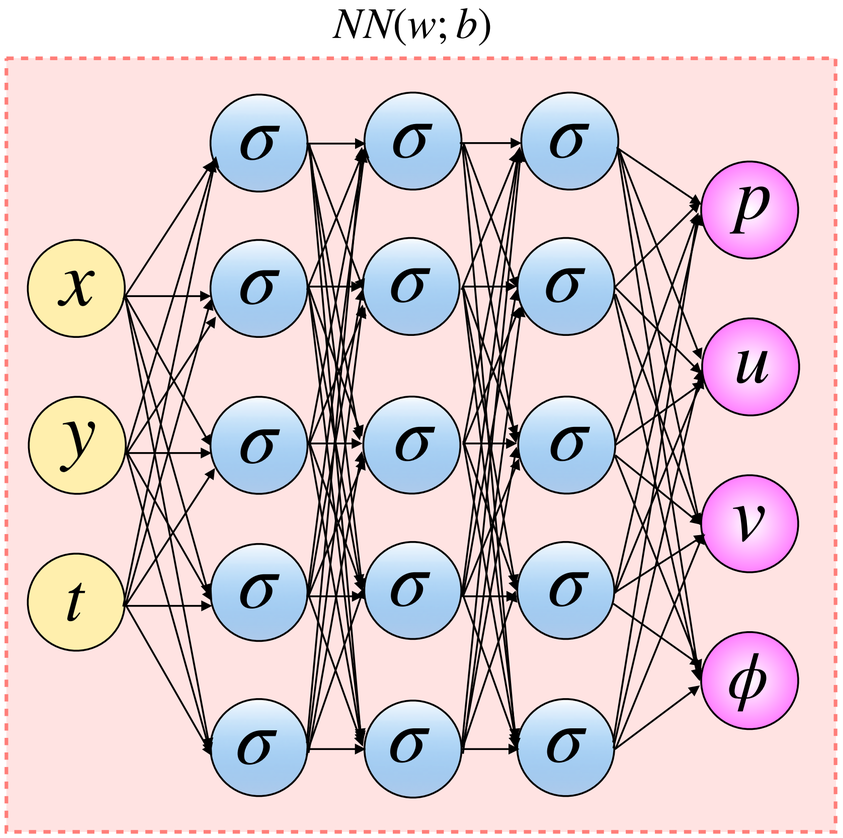
Data-driven inference of micro-bubble dynamics with physics-informed deep learning

Hanfeng Zhai

April 12, 2021

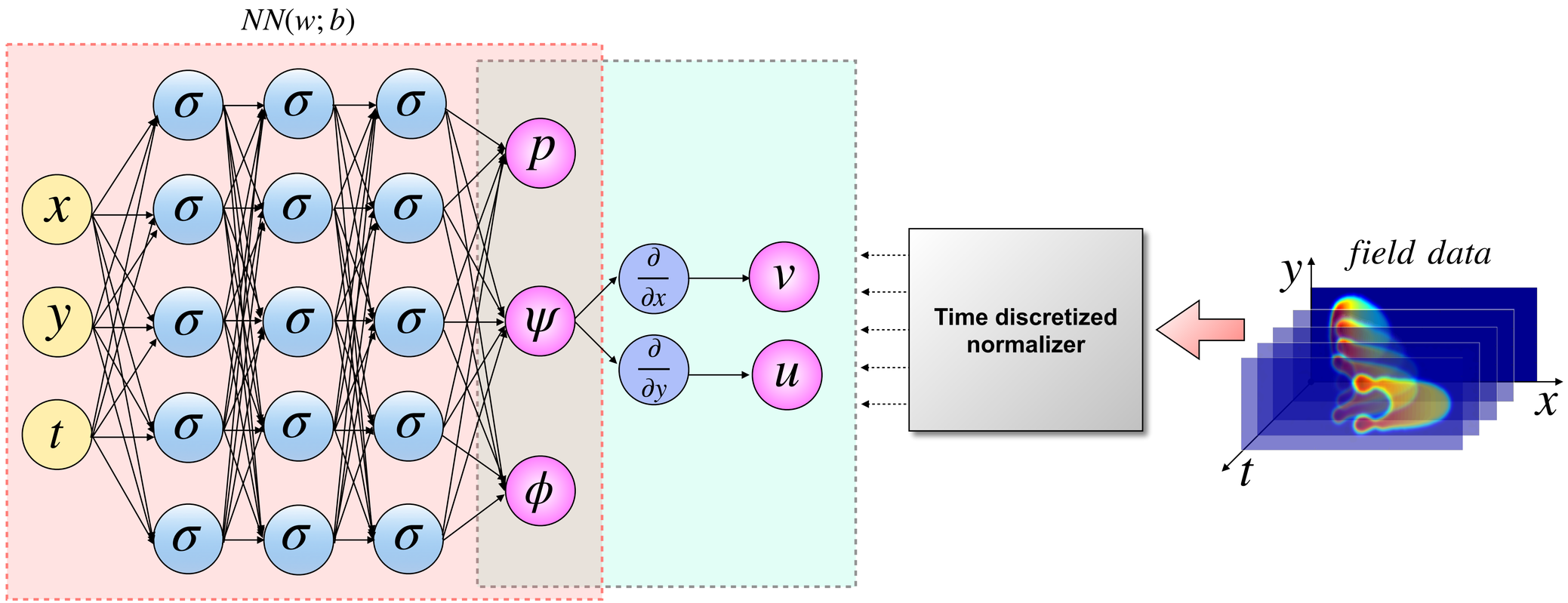
Methods



Deep Neural Networks

Methods

BubbleNet: Physics-Informed Neural Networks for general bubble dynamics



Methods

Traditional DNNs

Algorithm 1 DNN for predicting bubble dynamics

function DEEPNEURALNET($self,\ x,\ y,\ t,\ u,\ v,\ p,\ \phi,\ layers$)
  $(\hat{x},\ \hat{y},\ \hat{t},\ \hat{u},\ \hat{v},\ \hat{p},\ \hat{\phi}) =$ UPDATE($x,\ y,\ t,\ u,\ v,\ p,\ \phi$)
  $(\hat{weights},\ \hat{biases},\ \hat{layers}) = self.$INITIALIZENN($weights,\ biases,\ layers$)
  $self.$Loss $=$ `MSE`$[(u - u_{pred}) + (v - v_{pred}) + (p - p_{pred}) + (\phi - \phi_{pred})]$
  $u_{pred} = self.\text{Net}_\text{u}(x,\ y,\ t)$
  $v_{pred} = self.\text{Net}_\text{v}(x,\ y,\ t)$
  $p_{pred} = self.\text{Net}_\text{p}(x,\ y,\ t)$
  $\phi_{pred} = self.\text{Net}_\phi(x,\ y,\ t)$
  Optimization method 'L-BFGS-B' & Optimizer: Adam
  def INITIALIZENN($self,\ layers$)
    Initialize all the *weights* & *biases* for Net_u, Net_v, Net_p, Net_ϕ.
  def NEURALNET($self,\ weights,\ biases$)
    Build NN for $u,\ v,\ p,\ \phi$ with four sets of *weights* & *biases*.
  def $\{\text{Net}_\text{u}, \text{Net}_\text{v}, \text{Net}_\text{p}, \text{Net}_\phi\}$ ($self,\ x,\ y,\ t$)
    $\{u,\ v,\ p,\ \phi\} = self.$NEURALNET($x,\ y,\ t,\ weights,\ biases$)
  def TRAIN($self,\ iterations$)
    Obtain training time & Losses; train the NN with Adam optimizer.
  def PREDICT $\{u,\ v,\ p,\ \phi\}$ ($self,\ iterations$)
    $\{u_{pred},\ v_{pred},\ p_{pred},\ \phi_{pred}\} = self.$ `sess`.run($x,\ y,\ t$)
end function
Input $= \{x,\ y,\ t\}$, Output $= \{u,\ v,\ p,\ \phi\}$
Hidden layers = [30 neurons $\times$ 9 layers]
Load fields data of micro-bubble system dynamics simulation.
Set training sets $= \{x_{train},\ y_{train},\ t_{train},\ u_{train},\ v_{train},\ p_{train},\ \phi_{train},\ layers\}$
  = MaxMinScaler(Simulation Data)
`model` = DEEPNEURALNET(training sets)
`model`.TRAIN(10000)
Set target prediction time as t_{pred}
Obtain $\{u_{pred},\ v_{pred},\ p_{pred},\ \phi_{pred}\}$ = `model`.PREDICT($x,\ y,\ t$) at t_{pred}.
Save all the data & post-processing.

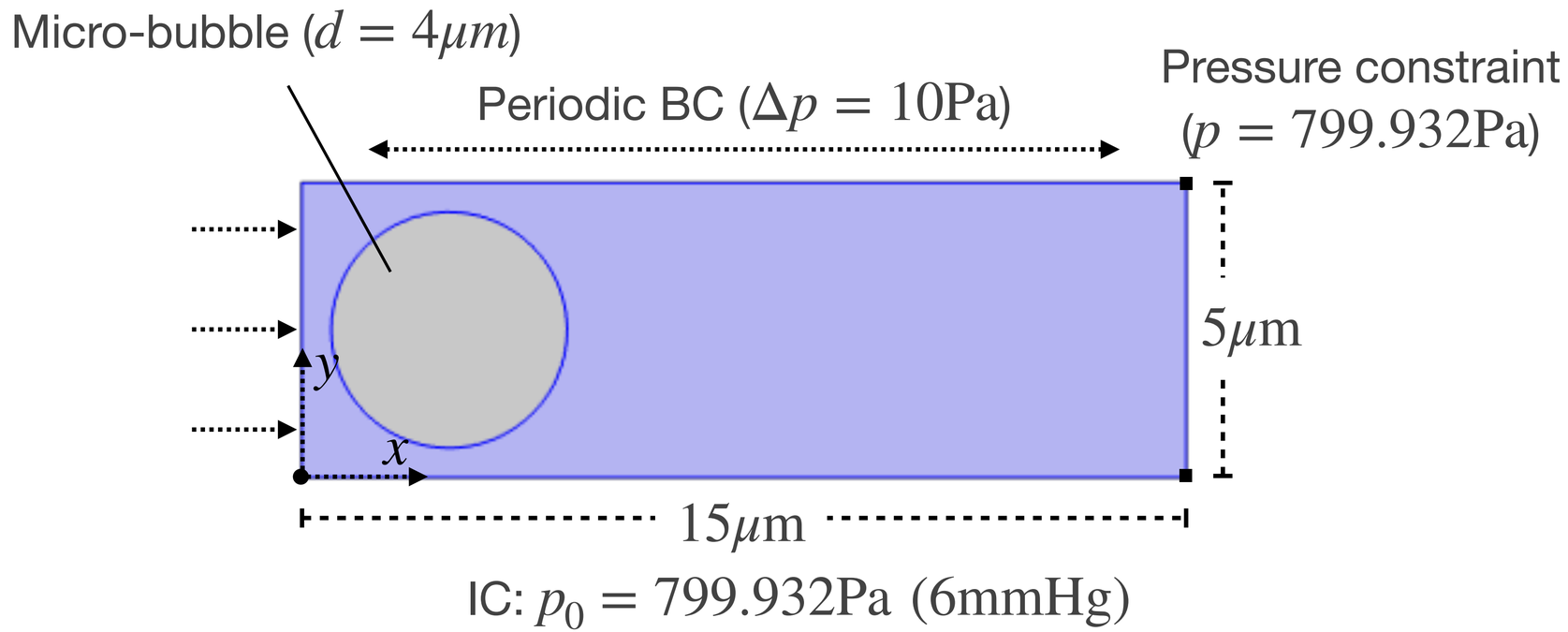
Methods

BubbleNet

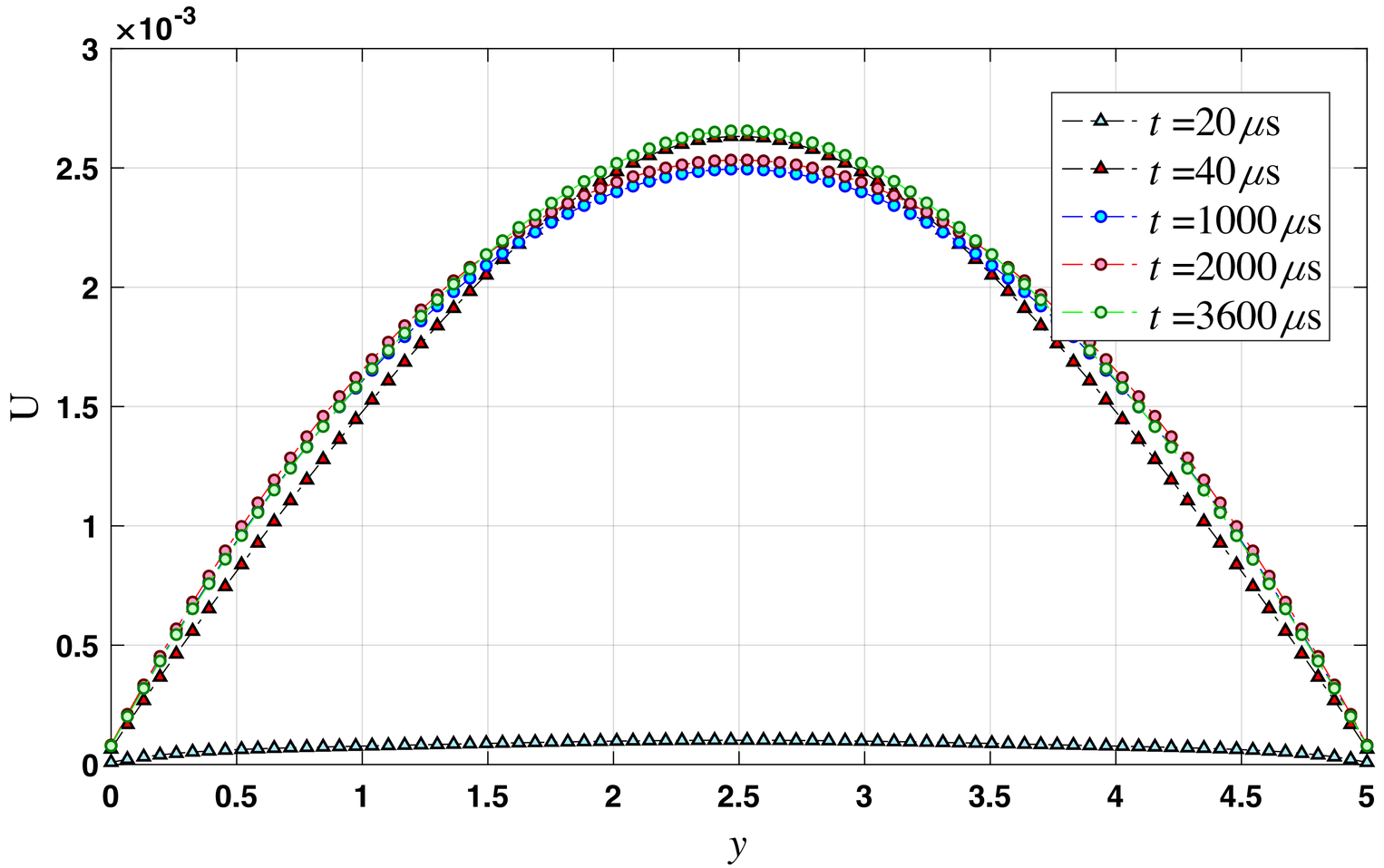
Algorithm 2 BubbleNet: physics-informed neural network for bubble dynamics

function BUBBLENET($self,\ x,\ y,\ t,\ u,\ v,\ p,\ \phi,\ layers$)
  $(\hat{x},\ \hat{y},\ \hat{t},\ \hat{u},\ \hat{v},\ \hat{p},\ \hat{\phi}) =$ UPDATE$(x,\ y,\ t,\ u,\ v,\ p,\ \phi)$
  $(\hat{weights},\ \hat{biases},\ \hat{layers}) = self.$INITIALIZENN$(weights,\ biases,\ layers)$
  $self.$Loss $=$ `MSE`$[(u - u_{pred}) + (v - v_{pred}) + (p - p_{pred}) + (\phi - \phi_{pred})]$
  $\{u_{pred}, v_{pred}, p_{pred}, \phi_{pred}\} = self.\{\mathrm{Net}_\psi, \mathrm{Net_p}, \mathrm{Net}_\phi\}(x,\ y,\ t)$
  Optimization method 'L-BFGS-B' & Optimizer: Adam
  def INITIALIZENN$(self,\ layers)$
    Initialize all the *weights* & *biases* for Net_ψ, $\mathrm{Net_p}$, Net_ϕ.
  def NEURALNET$(self,\ weights,\ biases)$
    Build NN for $\psi,\ p,\ \phi$ with four sets of *weights* & *biases*.
  def $\{\mathrm{Net}_\psi, \mathrm{Net_p}, \mathrm{Net}_\phi\}$ $(self,\ x,\ y,\ t)$
    $\{\psi,\ p,\ \phi\} = self.$NEURALNET$(x,\ y,\ t,\ weights,\ biases)$
    $u = \partial_y \psi \quad \& \quad v = -\partial_x \psi$
  def TRAIN$(self,\ iterations)$
    Obtain training time & Losses; train the NN with Adam optimizer.
  def PREDICT $\{u,\ v,\ p,\ \phi\}$ $(self,\ iterations)$
    $\{u_{pred},\ v_{pred},\ p_{pred},\ \phi_{pred}\} = self.$ `sess`.run$(x,\ y,\ t)$
end function
Set training sets $= \{x_{train},\ y_{train},\ t_{train},\ u_{train},\ v_{train},\ p_{train},\ \phi_{train},\ layers\}$
  $=$ TimeDiscretizedNormalization(Simulation Data, *timestep*)
`model` $=$ BUBBLENET(training sets)
`model`.TRAIN(10000)
Rest procedures same as **Algorithm 1**

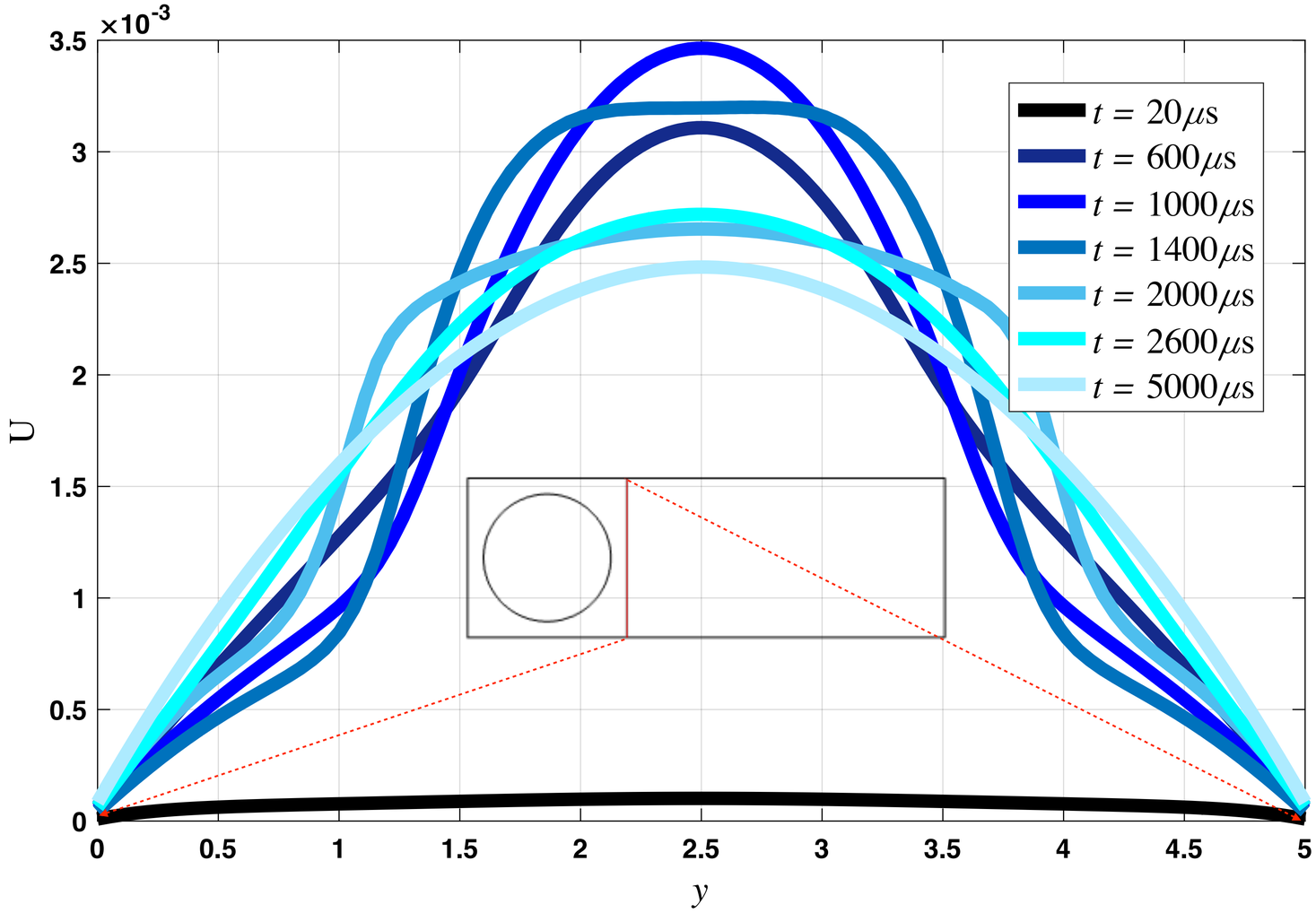
Case 1: single bubble movement



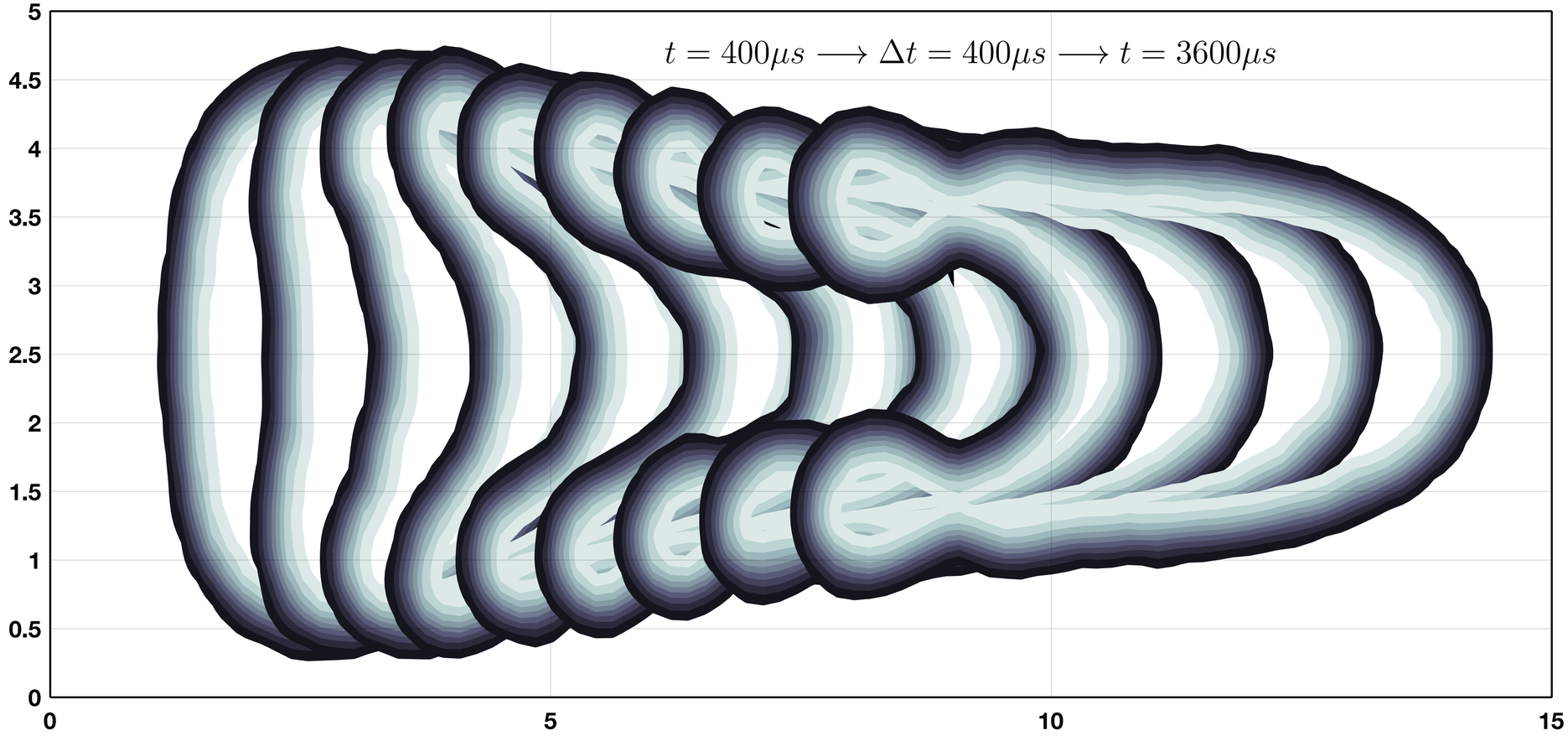
Case 1: single bubble movement



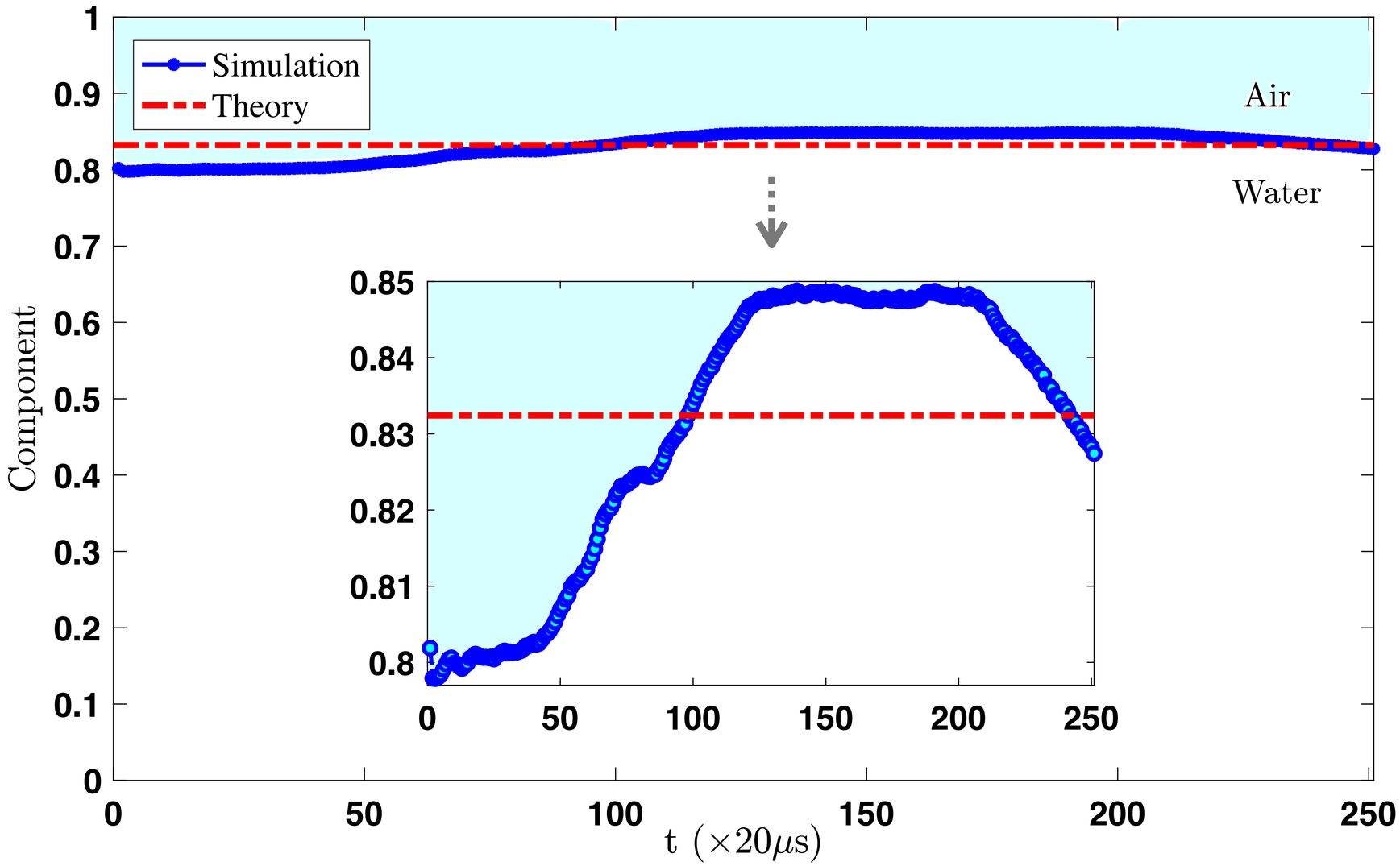
Case 1: single bubble movement



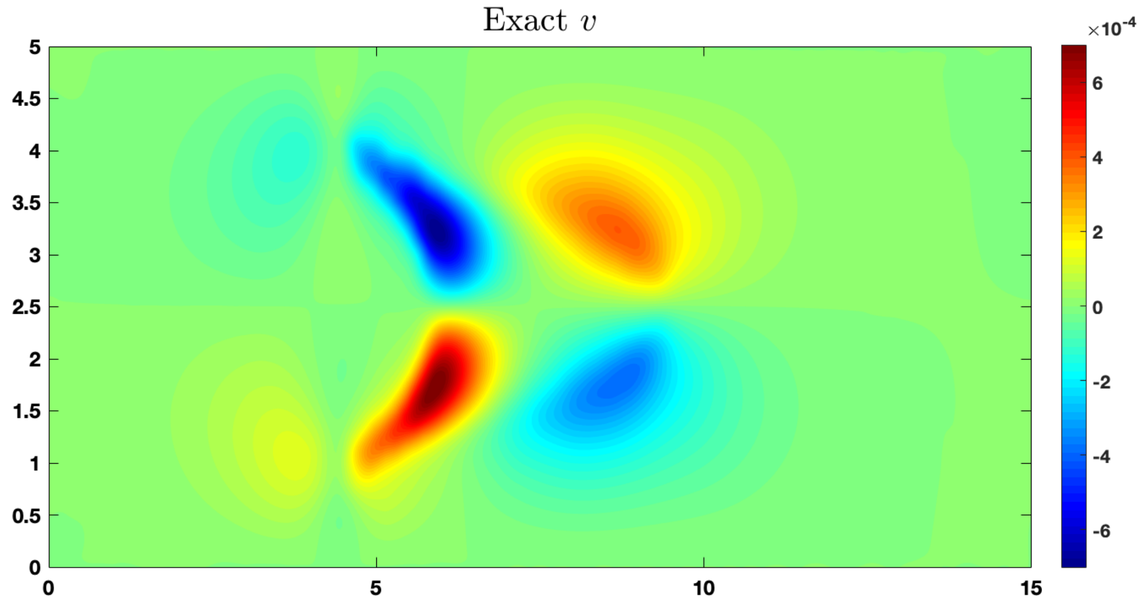
Results

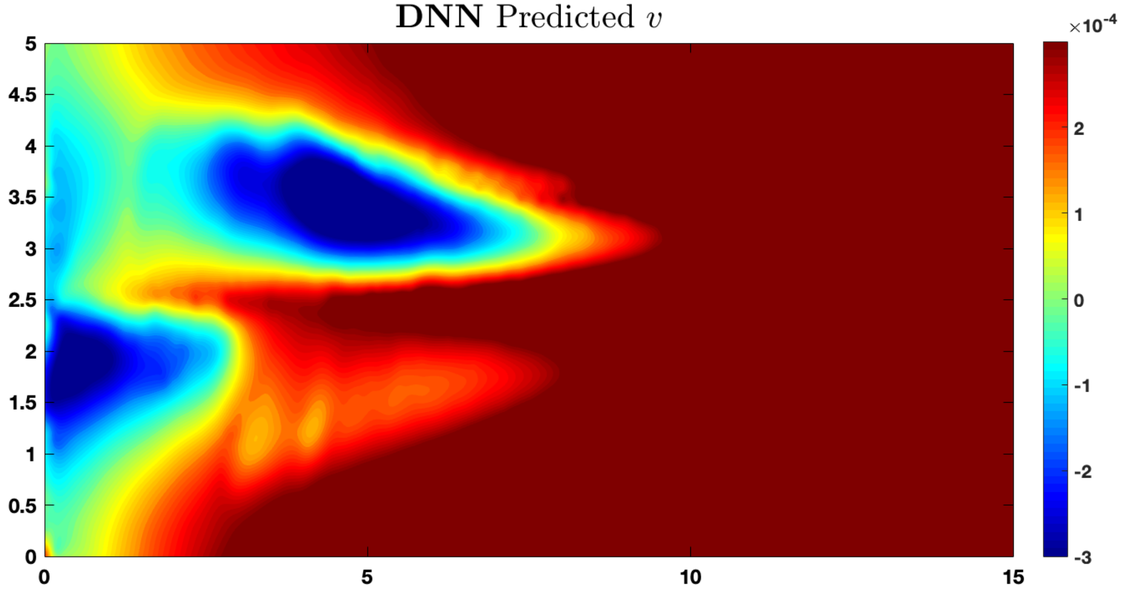

$t = 400\mu s \longrightarrow \Delta t = 400\mu s \longrightarrow t = 3600\mu s$
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15

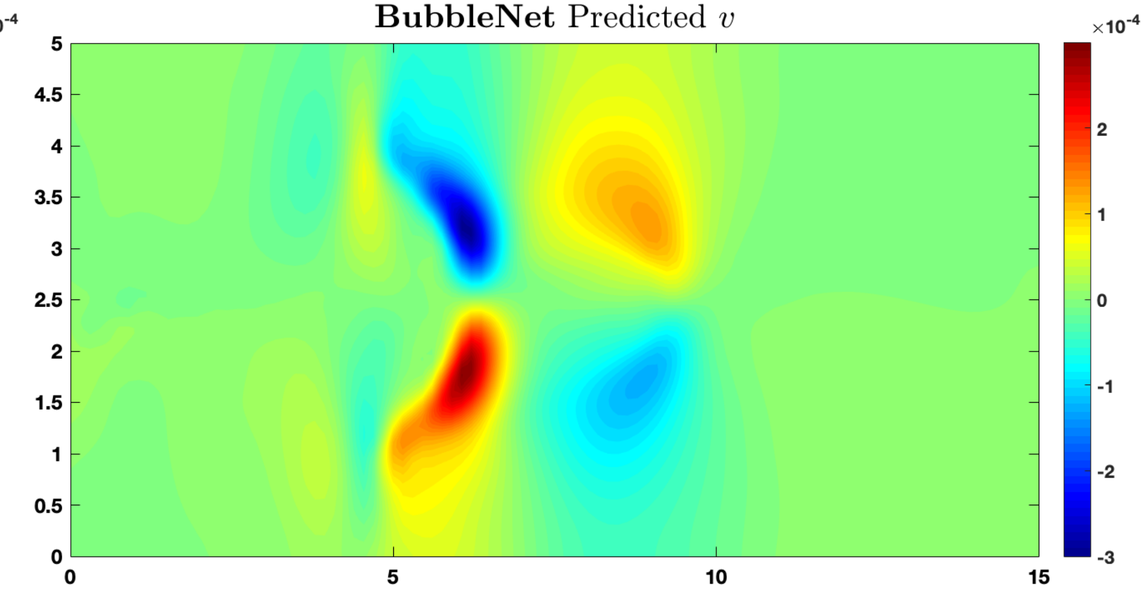
Results


Simulation
Theory
Air
Water
Component
t $(\times 20\mu s)$
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
0.8
0.81
0.82
0.83
0.84
0.85

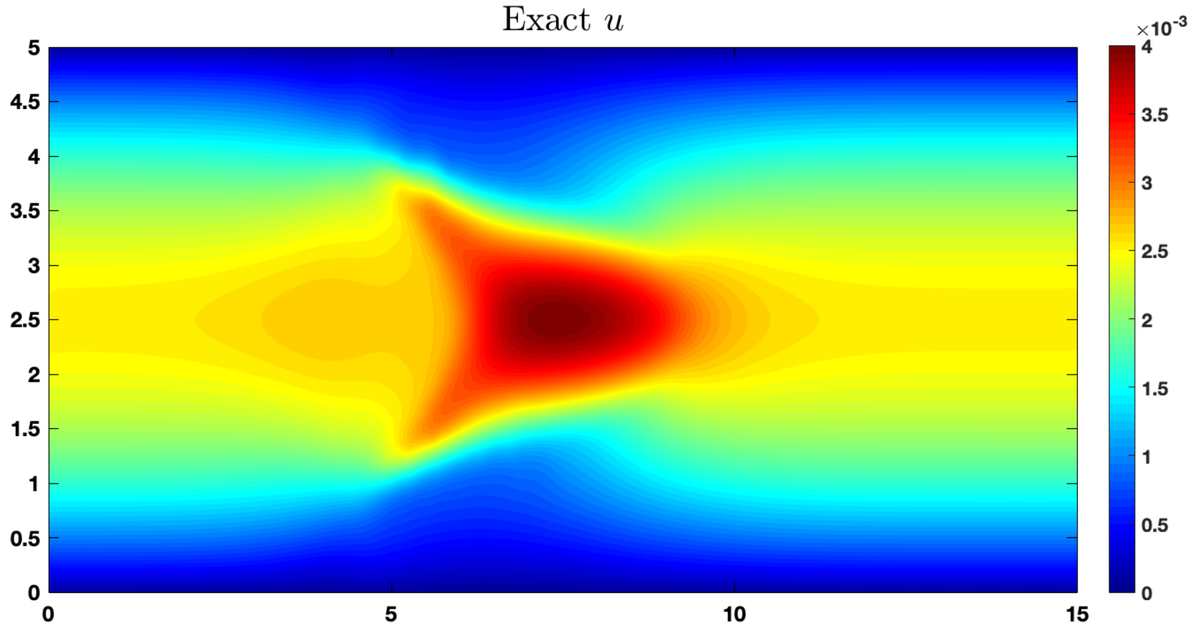
Results

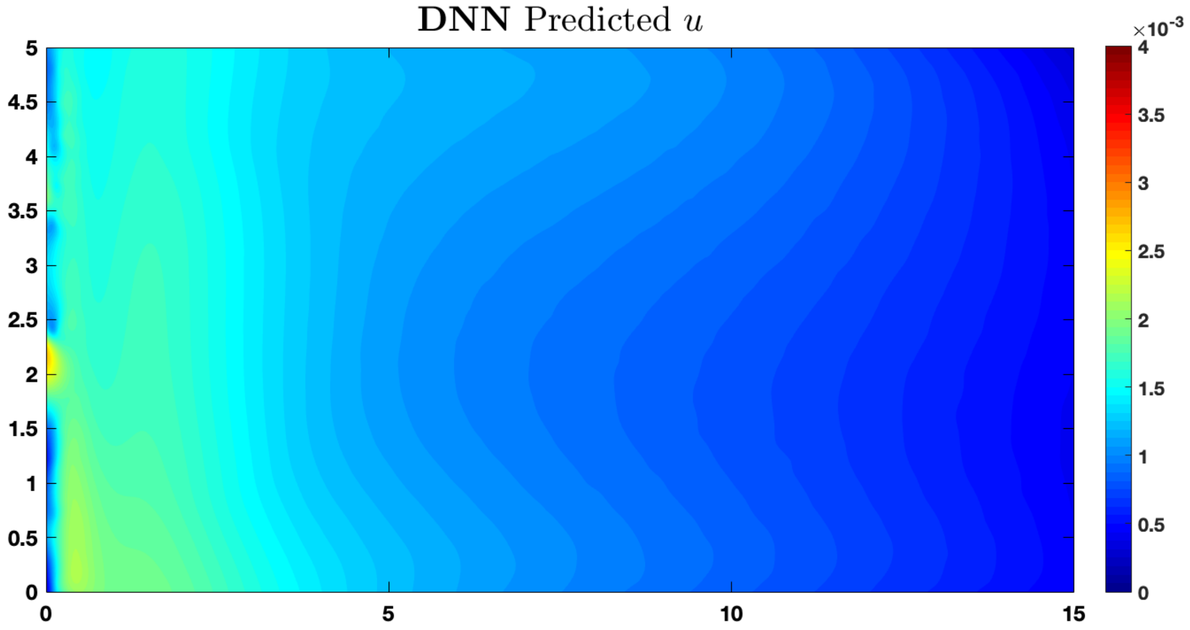

Exact v
$\times 10^{-4}$
6
4
2
0
-2
-4
-6
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
5
10
15

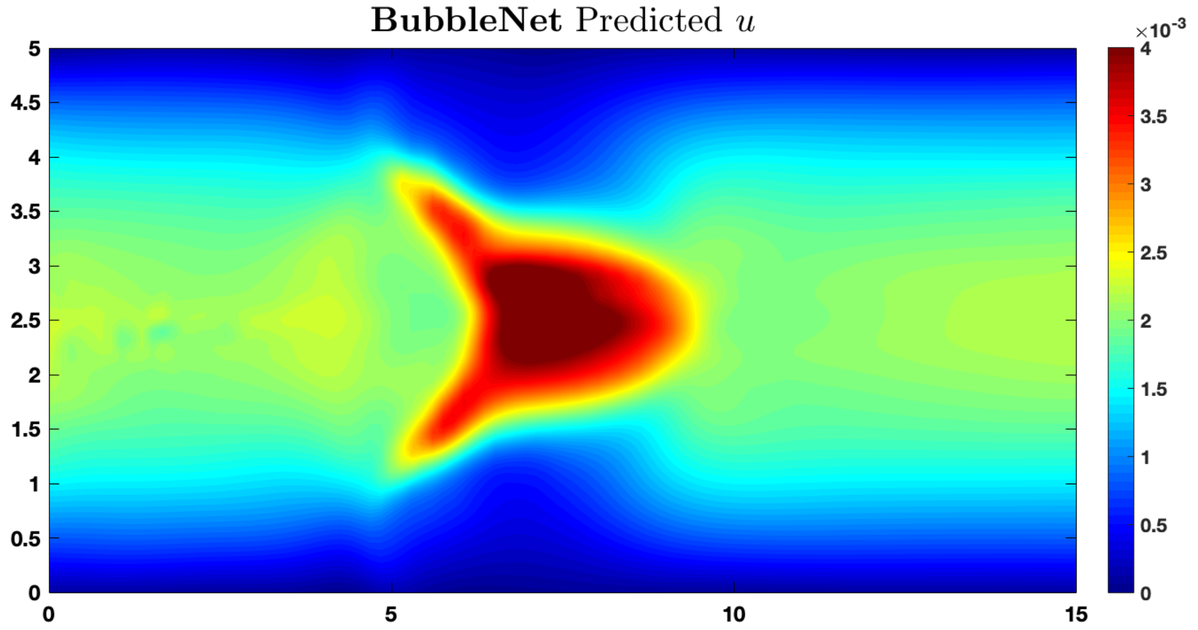

DNN Predicted v
$\times 10^{-4}$
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
5
10
15


BubbleNet Predicted v
$\times 10^{-4}$
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
5
10
15

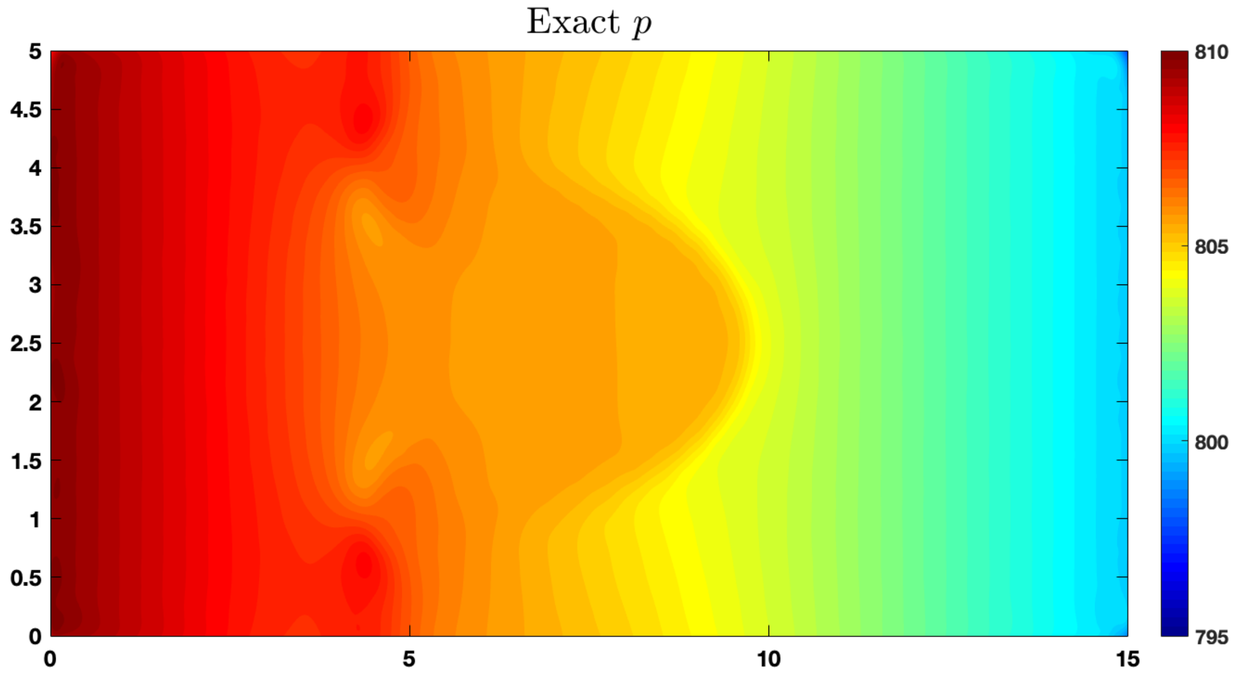
Results

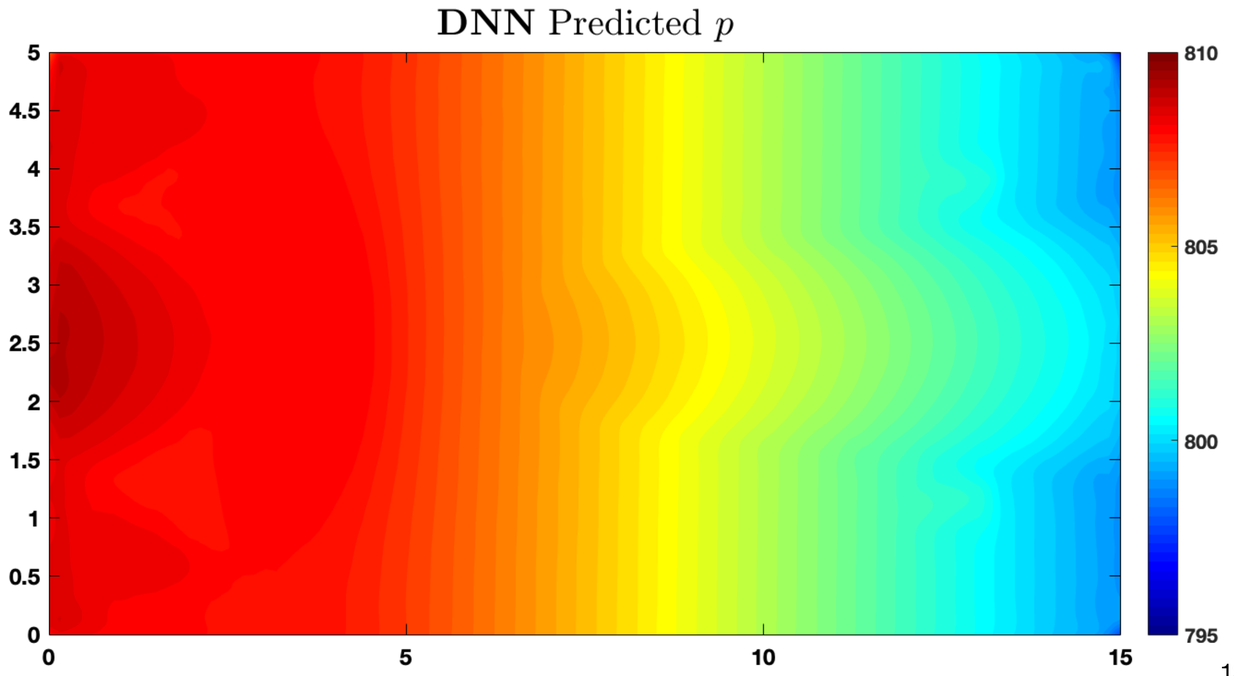

Exact u
$\times 10^{-3}$
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15

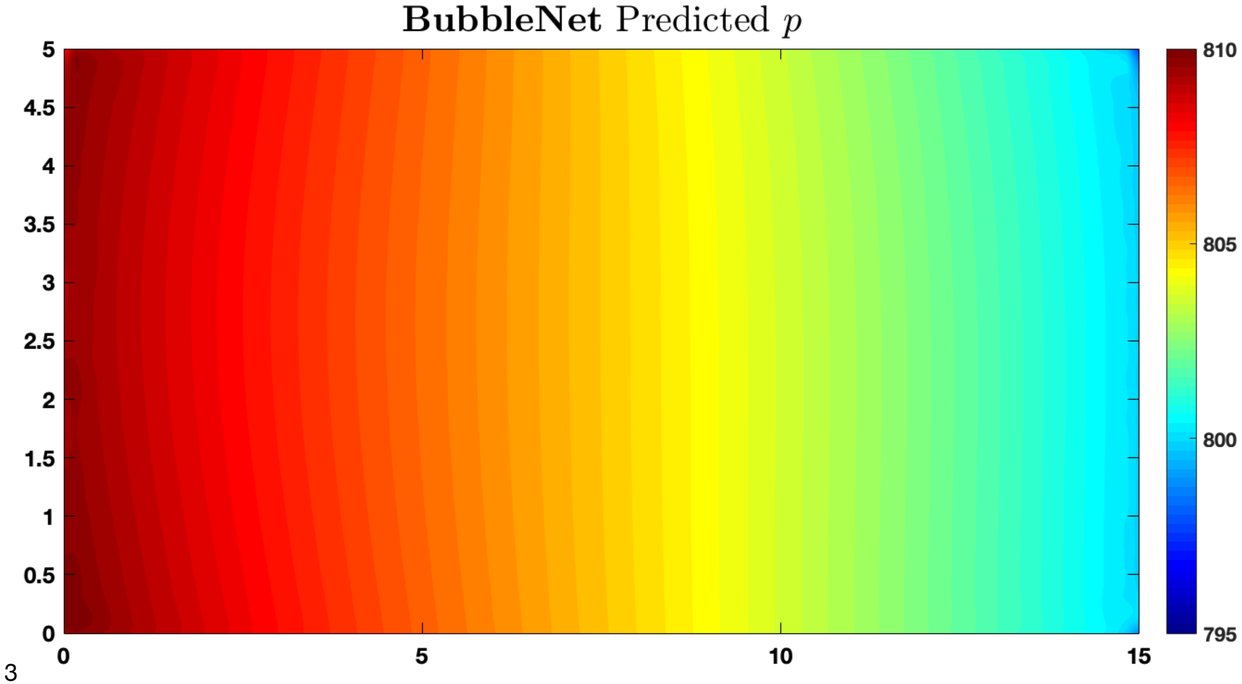

DNN Predicted u
$\times 10^{-3}$
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15


BubbleNet Predicted u
$\times 10^{-3}$
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15

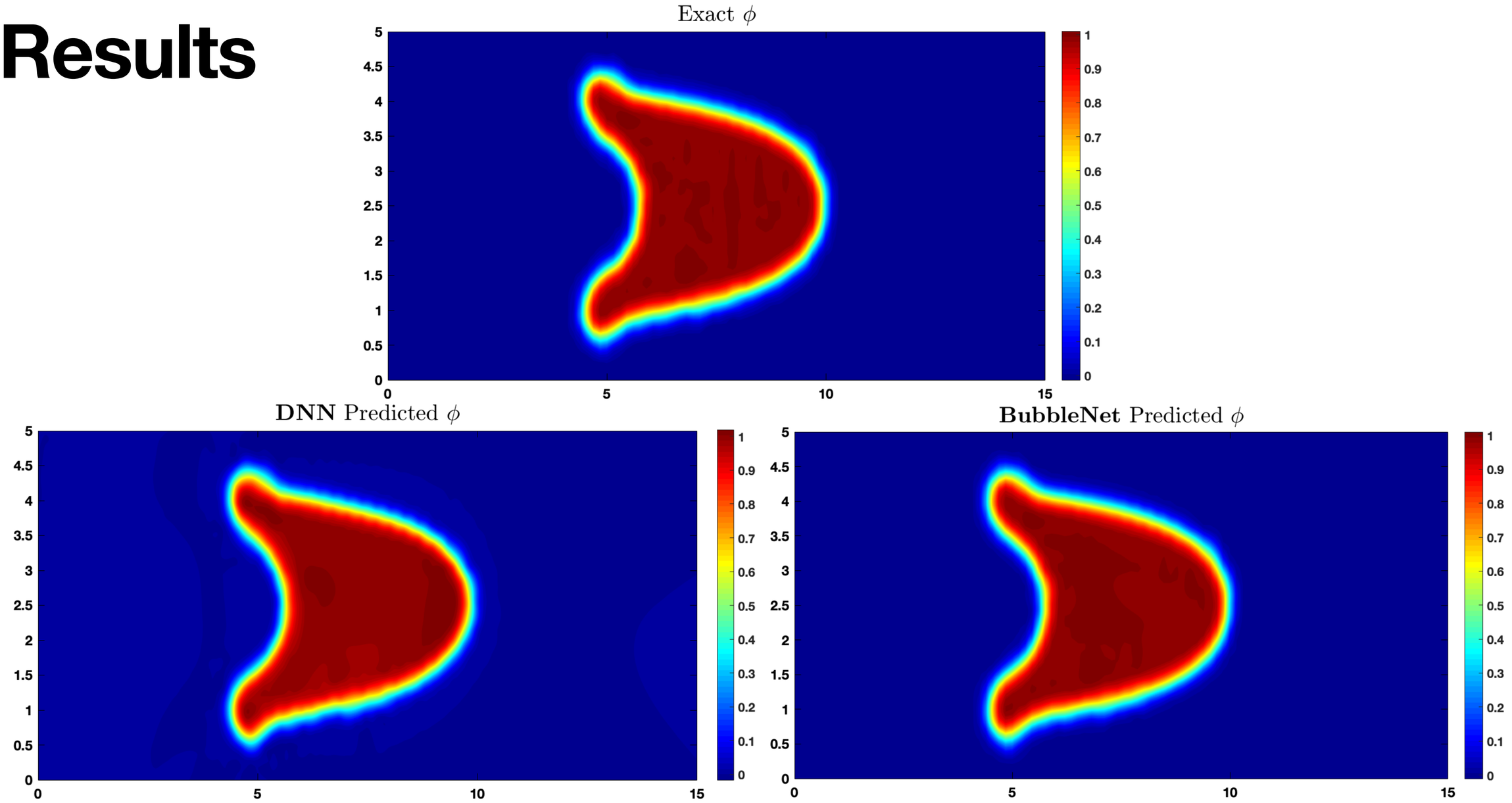
Results


Exact p
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15
810
805
800
795

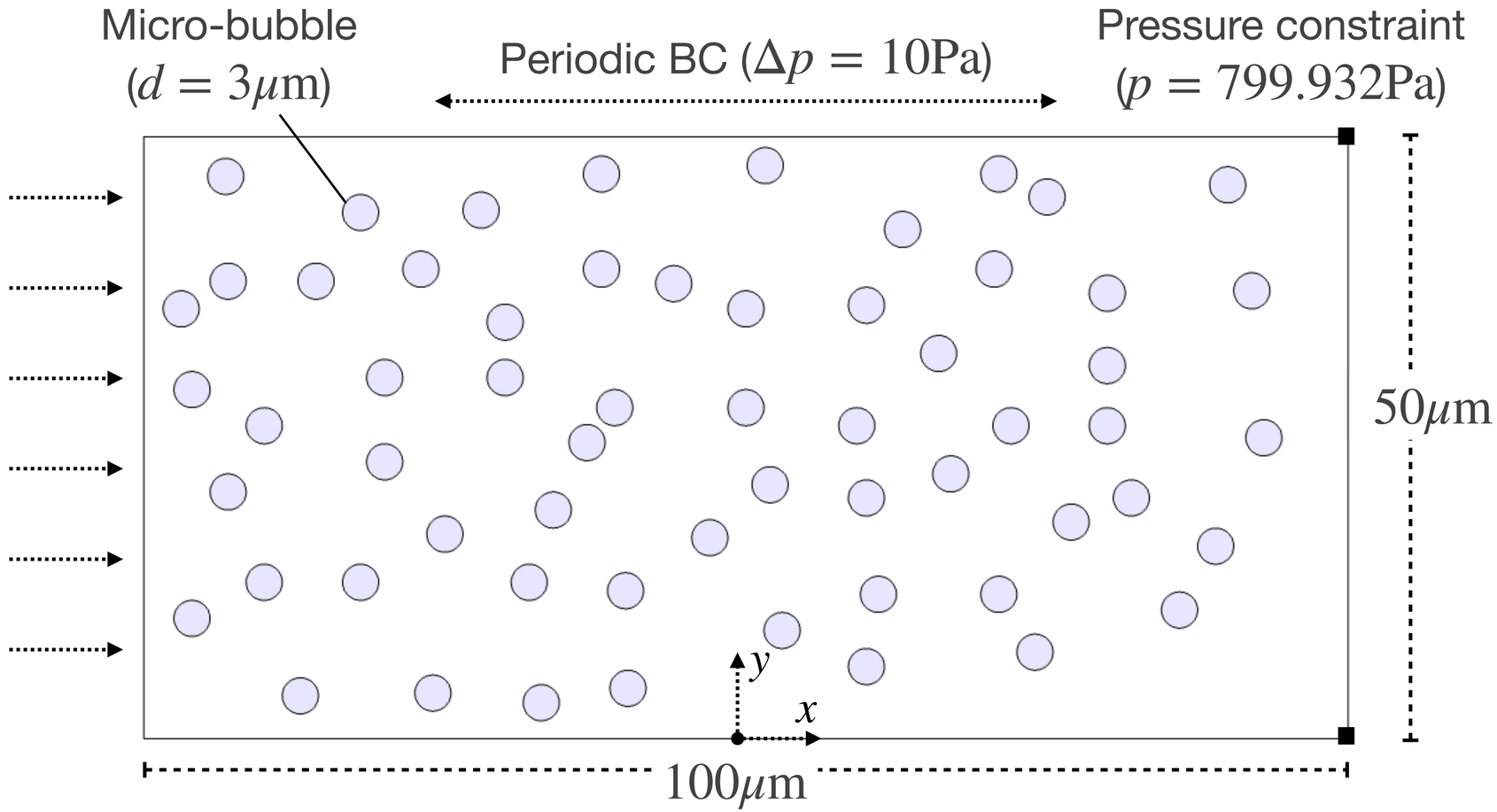

DNN Predicted p
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15
810
805
800
795


BubbleNet Predicted p
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15
810
805
800
795

Results

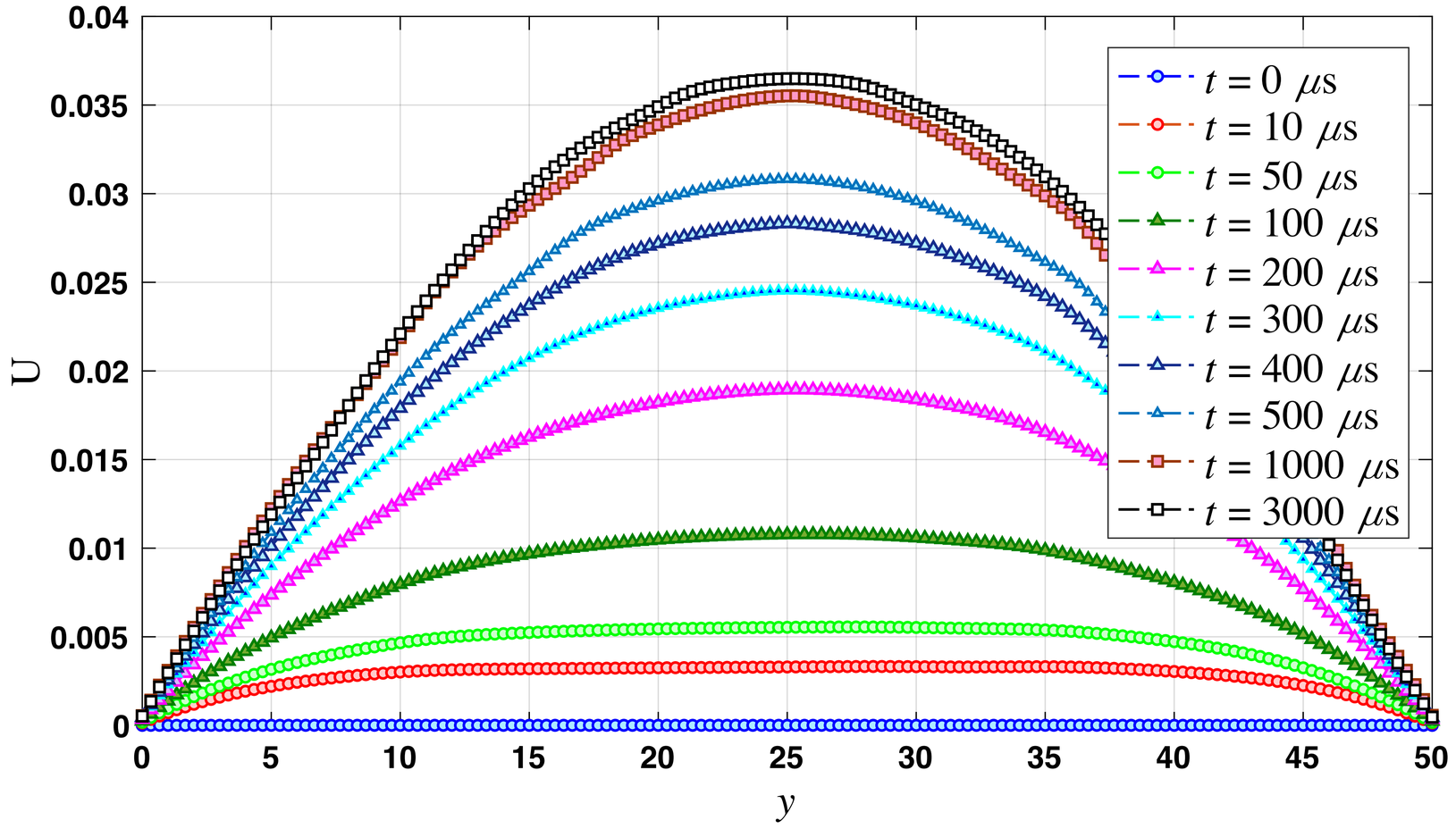


Case 2: multi bubble system

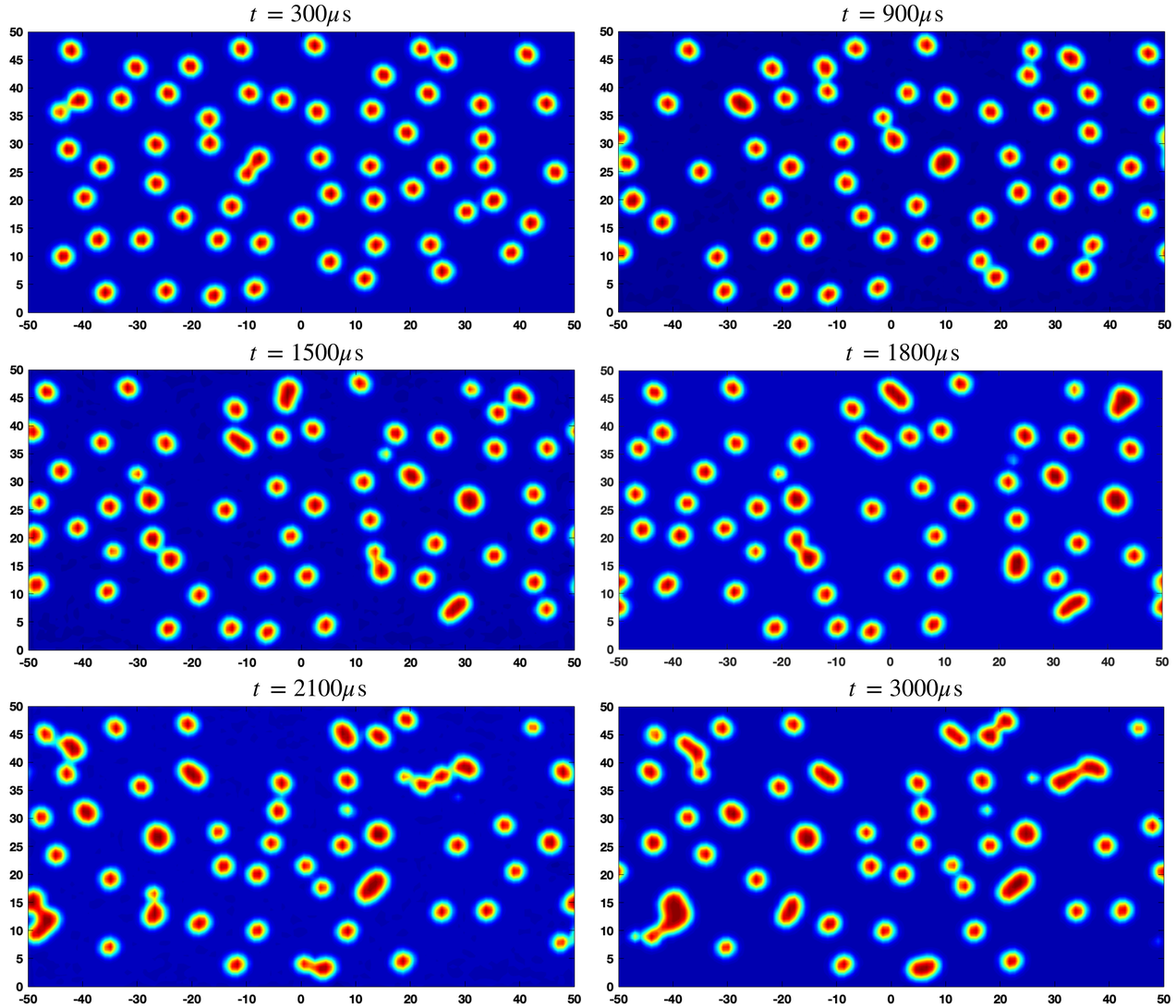


IC: $p_0 = 799.932$Pa (6mmHg)

Case 2: multi bubble system



Case 2: multi bubble system



Case 2: multi bubble system