MACHINE LEARNING MODELS FOR PREDICTIVE ANALYSIS OF PRESSURE DROP AND TEMPERATURE IN POLYMER ELECTROLYTE MEMBRANE FUEL CELL STACKS TO FIND OPTIMAL FABRICATION PARAMETERS

HANHEE LEE

ESROP GLOBAL

NATIONAL UNIVERSITY OF SINGAPORE

2024

Declaration

I hereby declare that the thesis is my original work and it has been written by
me in its entirety. I have duly acknowledged all the sources of information
which have been used in this thesis.

This thesis has also not been submitted for any degree in any university
previously

Hanhee Lee

August, 2024

Acknowledgements

Abstract

Contents

List of Figures

List of Tables

0.1 List of Symbols

Name	Description	Unit
p	Pressure	Pa
$\mathbf{u}, \mathbf{U}$	Velocity	m/s
ρ	Density	kg/m^3
$\mathbf{I}$	Identity tensor	-
$\mathbf{K}$	Viscous stress tensor	-
$\mathbf{F}$	Volume force vector	-
μ	Dynamic viscosity	Pa·s
T	Temperature	K
$\mathbf{n}$	Unit vector normal to the given surface	-
$\mathbf{G}$	Reciprocal wall distance	-
C_p	Specific heat capacity	J/(kg·K)
$\mathbf{q}$	Heat flux vector	-
Q	Heat source	W/m^3
k	Thermal conductivity	W/(m·K)
R	Thermal resistance	(m^2·K)/W
d	Thin layer thickness	m
Δ	Delta	-
ℓ	Length/Distance	m
u^+	Tangential velocity in viscous units	-
σ_w	Smoothing parameter	-
Re	Reynolds number	-
q	Quadratic loss coefficient	-
ϵ_p	Porosity	-
κ	Permeability	m^2
β_F	Forchheimer coefficient	kg/m^4
Q_m	Mass source	kg/(m^3·s)

Table 1: Variable Descriptions and Units

Name	Description
0	Standard Conditions
n	Unit vector normal to the given surface
p	Point
ted	Thermoelastic damping
b	Boundary
d	Down side
s	Solid
u	Up side
ref	Reference
w	Wall
T	Turbulent
$exit$	Exit
pc	Pressure curve

Table 2: Subscript Descriptions

Name	Description
T	Transpose

Table 3: Superscript Descriptions

Chapter 1

Introduction

1.1 Fuel Cells

Chapter 2

Neural Networks

2.1 Introduction to Machine Learning

Machine learning is a subset of artificial intelligence that focuses on developing algorithms and statistical models that enable computers to learn from and make predictions based on data. It is used in a wide range of applications, including image and speech recognition, medical diagnosis, and financial forecasting. Machine learning can be broadly categorized into three types: supervised learning, unsupervised learning, and reinforcement learning.

Machine learning is useful for tasks where it is difficult to explicitly program the rules or patterns, such as recognizing handwritten digits or detecting spam emails.

By training a model on a dataset, the model can learn the underlying patterns and relationships in the data and make predictions on new, unseen data.

2.2 Structure of Neural Networks

Neural networks are a class of machine learning models inspired by the structure and function of the human brain. They consist of interconnected nodes, or neurons, organized in layers. Each neuron receives input, processes it, and produces an output that is passed to the next layer. The connections between neurons are represented by weights, which are learned during the training process.

The basic building block of a neural network is the perceptron, which takes a set of inputs, applies weights to them, and passes the weighted sum through an activation function to produce an output. Multiple perceptrons are connected in layers to form a neural network. The most common type of neural network is the feedforward neural network, where information flows in one direction from the input layer to the output layer.

Neural networks can have multiple layers, with each layer performing a different

transformation on the input data. The input layer receives the input data, the hidden layers process the data, and the output layer produces the final output. The number of layers and the number of neurons in each layer are hyperparameters that can be tuned to optimize the performance of the network.
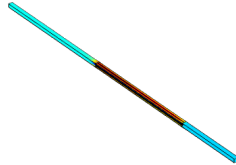
Figure 2.1: Structure of a feedforward neural network.

2.2.1 Types of Layers

Neural networks have multiple types of layers, including:

- Input Layer: The first layer of the network that receives the input data.

- Hidden Layers: Intermediate layers that process the input data and extract features.

- Output Layer: The final layer that produces the output of the network.

2.2.2 Shallow and Deep Neural Networks

Neural networks can be classified as shallow or deep based on the number of hidden layers they contain. Shallow networks have a small number of hidden layers, while deep networks have many hidden layers. Deep neural networks are capable of learning complex patterns and representations in the data but are more computationally intensive to train

2.3 Process of Supervised Learning

Supervised learning is a type of machine learning where the model is trained on labeled data. The objective is to learn a function that maps input data to the correct output based on the provided labels. The general process involves the following steps:

1. **Data Collection**: Gather a dataset consisting of input-output pairs. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be the set of input vectors and $\mathbf{Y} = \{y_1, y_2, \ldots, y_n\}$ be the corresponding set of output values.

2. **Data Preprocessing**: Clean and preprocess the data to remove noise, handle missing values, and normalize the features. This step ensures that the data is in a suitable format for training the model.

3. **Model Selection**: Choose a neural network architecture suitable for the problem. This includes deciding on the number of layers, the number of neurons per layer, and the type of activation functions.

4. **Initialization**: Initialize the weights $\mathbf{W}$ and biases $\mathbf{b}$ of the network. This is typically done using small random values.

5. **Forward Propagation**: Compute the predicted output $\hat{y}$ by passing the input $\mathbf{x}$ through the network.

6. **Loss Computation**: Calculate the loss $\mathcal{L}(\hat{y}, y)$ which measures the difference between the predicted output $\hat{y}$ and the actual output y.

7. **Backward Propagation**: Compute the gradients of the loss with respect to the weights and biases.

8. **Weight Update**: Update the weights and biases using an optimization algorithm such as Stochastic Gradient Descent (SGD).

9. **Model Evaluation**: Evaluate the performance of the model on a validation set to tune hyperparameters and avoid overfitting.

Mathematically, the process can be summarized as follows:

Given an input vector $\mathbf{x}$, the network's output $\hat{y}$ is computed as:

$$\hat{y} = f(\mathbf{x}; \mathbf{W}, \mathbf{b}) \tag{2.1}$$

where f is the function represented by the neural network, parameterized by weights $\mathbf{W}$ and biases $\mathbf{b}$.

The loss function $\mathcal{L}$ is defined to measure the discrepancy between $\hat{y}$ and the true output y:

$$\mathcal{L}(\hat{y}, y) \tag{2.2}$$

The gradients of the loss with respect to the parameters are computed during backpropagation:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}} \tag{2.3}$$

Finally, the parameters are updated using an optimization algorithm:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \quad \mathbf{b} \leftarrow \mathbf{b} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}} \tag{2.4}$$

where η is the learning rate.
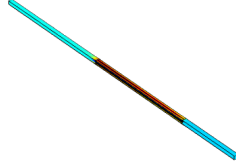
Figure 2.2: Overview of the supervised learning process in neural networks.

2.4 Forward Propogation

Forward propagation is the process by which the input is passed through the neural network to obtain the output. It involves the computation of outputs at each layer of the network until the final output is achieved. The process can be described as follows:

Given an input vector $\mathbf{x}$, the output of the first layer is computed as:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \tag{2.5}$$

$$\mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)}) \tag{2.6}$$

where $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ are the weights and biases of the first layer, $\mathbf{z}^{(1)}$ is the linear combination of inputs and weights, and σ is the activation function.

This process is repeated for each subsequent layer. For the l-th layer, the computations are:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \tag{2.7}$$

$$\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)}) \tag{2.8}$$

Finally, the output of the network is obtained:

$$\hat{y} = \mathbf{a}^{(L)} \tag{2.9}$$

where L is the number of layers in the network.
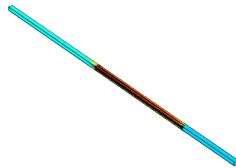
Figure 2.3: Forward propagation through a neural network.

2.4.1 Activation Functions

Activation functions introduce non-linearity into the neural network, allowing it to learn complex patterns. Common activation functions include:

Sigmoid

The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.10}$$

It maps any real-valued number into the range (0, 1).

Hyperbolic Tangent (Tanh)

The tanh function is defined as:

$$\sigma(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.11}$$

It maps any real-valued number into the range (-1, 1).

Rectified Linear Unit (ReLU)

The ReLU function is defined as:

$$\sigma(z) = \max(0, z) \tag{2.12}$$

It outputs the input directly if it is positive; otherwise, it outputs zero.

Leaky ReLU

The Leaky ReLU function is defined as:

$$\sigma(z) = \begin{cases} z & \text{if } z \geq 0 \\ \alpha z & \text{if } z < 0 \end{cases} \tag{2.13}$$

where α is a small constant.

2.5 Backward Propagation

Backward propagation, or backpropagation, is the process by which neural networks update their weights and biases to minimize the loss function. It involves calculating the gradient of the loss function with respect to each weight by the chain rule, iterating backward from the output layer to the input layer. The main steps are as follows:

1. **Compute the loss**: Calculate the loss $\mathcal{L}$ between the predicted output $\hat{y}$ and the actual output y.
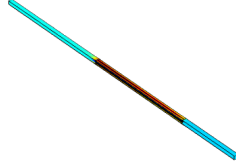
Figure 2.4: Common activation functions used in neural networks.

2. **Calculate the gradient of the loss with respect to the output layer**: For the output layer, compute the gradient of the loss with respect to the activations.

3. **Propagate the gradient backward through the network**: Use the chain rule to compute the gradient of the loss with respect to the weights and biases of each layer.

4. **Update the weights and biases**: Use the gradients to update the weights and biases in a direction that reduces the loss.

Mathematically, the gradient of the loss $\mathcal{L}$ with respect to the weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$ in layer l is computed as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} \mathbf{a}^{(l-1)} \tag{2.14}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \tag{2.15}$$

where $\delta^{(l)}$ is the error term for layer l and $\mathbf{a}^{(l-1)}$ is the activation of the previous layer.

The error term $\delta^{(l)}$ is computed as:

$$\delta^{(l)} = \begin{cases} (\mathbf{a}^{(L)} - y) \odot \sigma'(\mathbf{z}^{(L)}) & \text{for the output layer} \\ (\mathbf{W}^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(\mathbf{z}^{(l)}) & \text{for hidden layers} \end{cases} \tag{2.16}$$

where $\odot$ denotes the element-wise multiplication and σ' is the derivative of the activation function.

2.5.1 Loss Functions

Loss functions, also known as cost functions, measure how well the neural network's predictions match the actual target values. Common loss functions include:

Mean Squared Error (MSE)

The Mean Squared Error is used for regression tasks and is defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2.17}$$
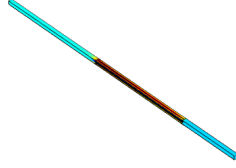
Figure 2.5: Illustration of common loss functions.

2.5.2 Methods to Update Weights

Updating the weights and biases of a neural network is a crucial part of the training process. Various methods can be employed to perform these updates:

Stochastic Gradient Descent (SGD)

SGD updates the weights using a single training example at a time:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \tag{2.18}$$

where η is the learning rate.

Batch Gradient Descent

Batch Gradient Descent computes the gradient using the entire training dataset:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}} \tag{2.19}$$

Mini-Batch Gradient Descent

Mini-Batch Gradient Descent is a compromise between SGD and Batch Gradient Descent. It uses a small random subset (mini-batch) of the training data to compute the gradient:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{m} \sum_{i=1}^{m} \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}} \tag{2.20}$$

where m is the mini-batch size.

9

Adaptive Methods

Adaptive methods like AdaGrad, RMSProp, and Adam adjust the learning rate based on the history of the gradients. For example, the Adam optimization algorithm updates the weights as follows:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \tag{2.21}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \left(\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \right)^2 \tag{2.22}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \tag{2.23}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \tag{2.24}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \tag{2.25}$$

where β_1 and β_2 are hyperparameters, $\mathbf{m}_t$ and $\mathbf{v}_t$ are the first and second moment estimates, and ϵ is a small constant.
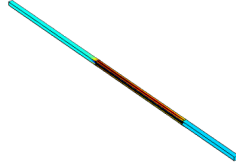
Figure 2.6: Comparison of different methods to update weights.

2.6 Hyperparameters

Hyperparameters are parameters whose values are set before the learning process begins. Unlike model parameters, which are learned during training, hyperparameters control the training process and influence the performance of the neural network. Common hyperparameters include learning rate, batch size, number of epochs, network architecture, activation functions, weight initialization, dropout rate, and regularization parameters.

2.6.1 Common Hyperparameters

Learning Rate

The learning rate η controls the size of the steps taken during gradient descent to update the weights. A smaller learning rate can lead to more precise convergence but slower training, while a larger learning rate can speed up training but might overshoot the optimal solution.

Batch Size

Batch size determines the number of training examples used to calculate the gradient in one forward/backward pass. It influences the stability and speed of the training process. Common choices are small (stochastic gradient descent), large (batch gradient descent), or in between (mini-batch gradient descent).

Number of Epochs

The number of epochs defines how many times the entire training dataset passes through the network. More epochs typically improve learning, but excessive epochs can lead to overfitting.

Network Architecture

Network architecture includes the number of layers, the number of neurons per layer, and the type of layers used (e.g., dense, convolutional, recurrent). These choices significantly impact the capacity and capability of the network.

Activation Functions

Different activation functions (e.g., Sigmoid, Tanh, ReLU) can be used in different layers of the network. The choice of activation function affects the network's ability to capture non-linear patterns.

Weight Initialization

Weight initialization affects the starting point of the training process. Common initialization methods include random initialization, Xavier initialization, and He initialization, each suitable for different types of activation functions.

Dropout Rate

Dropout is a regularization technique where a fraction of neurons is randomly set to zero during training. The dropout rate controls this fraction and helps prevent overfitting by promoting the independence of neurons.

Regularization Parameters

Regularization parameters, such as L1 and L2 regularization, add a penalty to the loss function to constrain the model complexity. They help in preventing overfitting by discouraging overly complex models.
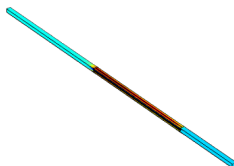
Figure 2.7: Overview of common hyperparameters in neural networks.

2.6.2 Hyperparameter Optimization

Hyperparameter optimization involves finding the optimal set of hyperparameters that result in the best performance of the neural network. Common methods for hyperparameter optimization include:

Grid Search

Grid search involves specifying a set of values for each hyperparameter and training the model on all possible combinations of these values. It is computationally expensive but exhaustive.

$$\text{Optimal Hyperparameters} = \arg\min_{\eta, \text{batch size},...} \text{Validation Loss} \qquad (2.26)$$

Random Search

Random search samples random combinations of hyperparameters from a predefined distribution. It is often more efficient than grid search and can find good hyperparameters with fewer iterations.

$$\text{Optimal Hyperparameters} = \arg\min_{\eta, \text{batch size},...} \text{Validation Loss} \qquad (2.27)$$

Bayesian Optimization

Bayesian optimization builds a probabilistic model of the objective function and uses it to select the most promising hyperparameters to evaluate next. It is more sophisticated and can find optimal hyperparameters more efficiently.

$$\text{Optimal Hyperparameters} = \arg\max_{\eta,\text{batch size},\ldots} P(\text{Low Validation Loss} \mid \eta, \text{batch size}, \ldots)$$
$$(2.28)$$

Hyperband

Hyperband is a method that combines random search with early stopping. It evaluates many configurations with a small number of iterations and progressively increases the budget for the most promising configurations.

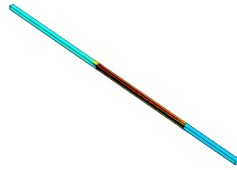$$\text{Optimal Hyperparameters} = \arg\min_{\eta,\text{batch size},\ldots} \text{Validation Loss} \qquad (2.29)$$

Figure 2.8: Comparison of different hyperparameter optimization methods.

Chapter 3

Literature Review

Chapter 4

Materials and Methods

Header	Symbol	Explanation	Unit
		Input Parameters	
Q	Q	**Heat generation** influences membrane hydration and operational efficiency; critical to prevent membrane dry-out and maintain ion conductivity.	Wm^{-2}
Tamb	T_{amb}	**Ambient temperature of the air** sets baseline thermal conditions; higher temperatures boost performance to a limit before causing potential overheating.	$^\circ C$
Uin	U_{in}	**Airflow velocity** determines oxygen supply rate, essential for maintaining optimal reaction rates and power output.	m/s
Wcc	W_{cc}	**Cathode channel width** determines oxygen flow and diffusion rates to the cathode, crucial for optimizing reaction efficiency.	mm
Hcc	H_{cc}	**Cathode channel height** affects gas flow resistance and water removal, essential for maintaining membrane hydration and preventing flooding.	mm
Lcc	L_{cc}	**Cathode channel length** influences the residence time of reactants and products along the channel, impacting overall fuel cell efficiency.	mm
Wr	W_r	**Rib width** supports mechanical stability and maximizes the active area available for reactions, balancing structural support with performance.	mm
Hr	H_r	**Rib height** controls the depth of flow channels, enhancing reactant distribution and efficient water management within the stack.	mm
		Output Performance Metrics	
Tsta	T_{stack}	**Stack temperature** is crucial for optimal reaction rates, membrane hydration, and to prolong the lifespan.	$^\circ C$
Delp	Δp	**Pressure drop** indicates the system's resistance to reactant flow; minimizing pressure drop is essential to enhance efficiency and ensure uniform distribution.	Pa

Table 4.1: Detailed Explanation of Variables

Chapter 5

COMSOL Model

5.1 Model Assumptions

1. The airflow entering the channel is turbulent. The Reynolds number exceeds 2300 at an inlet velocity (U_{in}) of 0.3 m/s.

2. Portions of the fuel cell not part of the cathode flow field are impermeable to air. While carbon paper is porous, the in-plane pressure drop is much greater than that of the cathode flow field.

3. The cathode flow field channels are modeled with a uniform height. The average offset (0.025 mm) is negligible compared to the channel height (1 mm).

4. Cathode flow fields are modeled with straight edges and right-angle folds. The fold radius (0.05 mm) is small compared to the channel width (1 mm).

5. Conservation of mass and momentum of airflow is assumed to be valid when transitioning between open spaces and cathode channels.

6. A representative unit cell approximates the entire stack, excluding edge effects near the terminals. Airflow entering the unit cell is uniform and symmetrical.

7. Channels are identical throughout the stack, resulting in no pressure drop differences between channels.

8. The channels are fully dry without condensation. The oxidant stoichiometry exceeds 100, ensuring water vapor produced is absorbed by the airflow, negating the need for accounting water saturation.

9. The losses from the fuel cell operation are converted to heat.

5.2 Workflow - Pressure Drop

The following steps outline the workflow process for the unit cell simulations conducted in COMSOL:

1. **Selection of Laminar Flow Model:**

   - This model was chosen because the airflow within the channels is laminar and is the largest contributor to the pressure drop across the stack.
   - The laminar flow model allows for faster convergence and has been shown to be accurate when compared with experimental results.

2. **Construction of Unit Cell:**

   - The unit cell was constructed using blocks with dimensions defined by equations to accommodate dimensional changes in the channels.

3. **Meshing Strategy:**

   - A custom mesh was employed, focusing on the transition areas between inflow and outflow within the channels with a finer mesh.
   - A coarser mesh was used within the channel and the air space between inflow and outflow.
   - A mesh independence study was performed during the mesh fine-tuning. The results are shown in Figure ?.
   - The difference between the blue dash (custom meshing) and green dots (physics-based extra fine meshing) is less than 3%, thus the blue dash mesh was selected for its speed and accuracy.

4. **Parameter Sweep Function:**

   - The parameter sweep function was utilized to explore all possible combinations for each stack size.

5. **Exporting Simulation Data:**

   - The inputs and outputs of the simulations were exported from COMSOL for further analysis.

5.3 Workflow - Temperature Stack

The following steps outline the workflow process for the unit cell simulations conducted in COMSOL:

1. **Selection of Physics Models:**

- The unit cell simulations were done using laminar flow physics as well as heat transfer in laminar flow.
- The parameter sweep function was utilized to explore all possible combinations in the parameter space.
- This model was selected because the flow through the channel is laminar and the simulation focuses on heat transfer within the channels.

2. **Construction of Unit Cell:**

- The unit cell was constructed using blocks with dimensions defined by equations to accommodate dimensional changes in the channels.

3. **Heat Generation Simulation:**

- Heat generation was simulated as a boundary heat source placed at the cathode side of the membrane.

4. **Heat Conduction Values:**

- The heat conduction values for each layer in the fuel cell were obtained from various papers, as shown in Table 8-1.

5. **Meshing Strategy:**

- A mesh independence study was conducted during the fine-tuning of the mesh. The results are shown in Figure ?.
- The difference between the blue dash and green dots is less than 1.06%, hence the blue dash mesh was selected for its speed and accuracy.

6. **Exporting Simulation Data:**

- The inputs and outputs of the simulations were exported from COMSOL for further analysis.

5.4   Parameters

Name	Value	Unit
Cell Area	0.005	m^2
Cell Voltage	0.6	V
Compression	0.1	-
Current	40	A
Hbp	5×10^{-5}	m
Hcc	1.25×10^{-3}	m
Hcp	3.15×10^{-4}	m
Hmem	1.5×10^{-5}	m
Kcp	1.5	$W \cdot m^{-1} K^{-1}$
Kmem	0.1	$W \cdot m^{-1} K^{-1}$
Lcc	0.03	m
pRef	1.0133×10^5	Pa
Q	5040	$W \cdot m^{-2}$
Tamb	293.15	K
Test	25.2	W
Uin	7.75	$m \cdot s^{-1}$
Wcc	0.001	m
Wr	5×10^{-5}	m

Table 5.1: Parameter Values

5.5 Materials

Name	Unit
Air	
Dynamic viscosity	$\text{Pa} \cdot \text{s}$
Ratio of specific heats	-
Heat capacity at constant pressure	$\text{J}/(\text{kg} \cdot \text{K})$
Density	kg/m^3
Thermal conductivity	$\text{W}/(\text{m} \cdot \text{K})$
Carbon Paper	
Density	kg/m^3
Heat capacity at constant pressure	$\text{J}/(\text{kg} \cdot \text{K})$
Thermal conductivity	$\text{W}/(\text{m} \cdot \text{K})$
Membrane	
Thermal conductivity	$\text{W}/(\text{m} \cdot \text{K})$
Steel Grade 316L	
Density	kg/m^3
Heat capacity at constant pressure	$\text{J}/(\text{kg} \cdot \text{K})$
Thermal conductivity	$\text{W}/(\text{m} \cdot \text{K})$

Table 5.2: Material Descriptions

5.6 Laminar Flow

5.6.1 Governing Equations

$$\rho(\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla \cdot [-p\mathbf{I} + \mathbf{K}] + \mathbf{F} \tag{5.1}$$

where ρ is the density of air, $\mathbf{u}$ is the velocity vector, p is the pressure, $\mathbf{I}$ is the identity matrix, $\mathbf{K}$ is the viscous stress tensor, and $\mathbf{F}$ is the volume force vector.

$$\nabla \cdot (\rho\mathbf{u}) = 0 \tag{5.2}$$

$$\mathbf{K} = \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) - \frac{2}{3}\mu(\nabla \cdot \mathbf{u})\mathbf{I} \tag{5.3}$$

where μ is the dynamic viscosity of air.

5.6.2 Initial Values

1. Velocity field

$$\mathbf{u} = \mathbf{0} \tag{5.4}$$

2. Pressure

$$p = 0 \tag{5.5}$$

5.6.3 Boundary Conditions

1. Wall (No slip)

$$\mathbf{u} = \mathbf{0} \tag{5.6}$$

2. Inlet (Velocity)

$$\mathbf{u} = -U_0\mathbf{n} \tag{5.7}$$

where U_0 is the initial velocity (i.e. normal inflow velocity) and $\mathbf{n}$ is the unit normal vector.

3. Outlet (Pressure)

$$[-p\mathbf{I} + \mathbf{K}]\mathbf{n} = -\hat{p}_0\mathbf{n} \tag{5.8}$$

$$\hat{p}_0 \leq p_0, \nabla\mathbf{G} \cdot \mathbf{n} = 0 \tag{5.9}$$

where $\hat{p}_0$ is the estimated standard condition pressure, $p_0 = 0$ is the standard condition pressure (i.e. suppress backflow), $\mathbf{G}$ is the reciprocal wall distance.

4. Symmetry

$$\mathbf{u} \cdot \mathbf{n} = 0 \tag{5.10}$$

$$\mathbf{K}_n - (\mathbf{K}_n \cdot \mathbf{n})\mathbf{n} = 0, \mathbf{K}_n = \mathbf{K}\mathbf{n} \tag{5.11}$$

5.7 Heat Transfer

5.7.1 Governing Equations - Solid

$$\rho C_p\mathbf{u} \cdot \nabla T + \nabla \cdot \mathbf{q} = Q + Q_{ted} \tag{5.12}$$

where C_p is the specific heat capacity of air, T is the temperature of air, $\mathbf{q}$ is the heat flux vector, Q is the heat source, and Q_{ted} is the thermoelastic damping heat source.

$$\mathbf{q} = -k\nabla T \tag{5.13}$$

where k is the thermal conductivity.

$$Q = (1.23 - Operating\ Voltage) \times \frac{Fuel\ Cell\ Stack\ Current}{Cell\ Area} \tag{5.14}$$

5.7.2 Governing Equations - Fluid

$$\rho C_p\mathbf{u} \cdot \nabla T + \nabla \cdot \mathbf{q} = Q + Q_p + Q_{vd} \tag{5.15}$$

where Q_p is the point heat source.

$$\mathbf{q} = -k\nabla T \tag{5.16}$$

5.7.3 Governing Equations - Thin Layer

$$-\mathbf{n}_d \cdot \mathbf{q}_u = \frac{(T_u - T_d)}{R_s} + \frac{1}{2}d_s Q_s \tag{5.17}$$

where R_s is the thermal resistance, and d_s is the thin layer thickness.

$$-\mathbf{n}_u \cdot \mathbf{q}_u = \frac{(T_d - T_u)}{R_s} + \frac{1}{2}d_s Q_s \tag{5.18}$$

$$R_s = \frac{d_s}{k_s} \tag{5.19}$$

5.7.4 Initial Values

1. Temperature

$$T = T_{amb} \tag{5.20}$$

where T_{amb} is the initial ambient temperature.

5.7.5 Boundary Conditions

1. Insulation

$$-\mathbf{n} \cdot \mathbf{q} = 0 \tag{5.21}$$

2. Symmetry

$$-\mathbf{n} \cdot \mathbf{q} = 0 \tag{5.22}$$

3. Boundary Heat Source

$$-\mathbf{n} \cdot \mathbf{q} = Q_b \tag{5.23}$$

where Q_b is the boundary heat source.

4. Outflow

$$-\mathbf{n} \cdot \mathbf{q} = 0 \tag{5.24}$$

5. Inlet

$$T = T_{amb} \tag{5.25}$$

5.8 Mesh

5.9 Study

5.9.1 Difference between External and Internal Sweep

An internal sweep involves varying parameters directly within the simulation model itself. This type of sweep is integrated into the simulation process, where the solver automatically adjusts the parameters as part of its internal algorithm. Internal sweeps are typically used to explore the parameter space of the model in

a more automated and controlled manner, often leveraging the solver's built-in capabilities to iterate through different parameter values.

An external sweep, on the other hand, involves varying parameters outside of the simulation model. This is usually managed by an external script or control mechanism that systematically modifies the input parameters, runs the simulation for each set of parameters, and collects the results. External sweeps provide more flexibility and control over the parameter variations and can be useful for complex scenarios where the internal capabilities of the solver might be limited.

5.9.2 Parametric Sweep

A parametric sweep is a systematic method used in simulations and computational experiments to explore the effects of varying parameters within a defined range. By adjusting one or more input parameters incrementally, researchers can observe how changes influence the outcome, enabling a comprehensive analysis of the system's behavior. This technique is essential for optimizing designs, identifying critical factors, and understanding the sensitivity of the results to different variables. Ultimately, parametric sweeps provide valuable insights that drive informed decision-making and innovation.

Name	Values	Unit
Hcc	1 1.25 1.5 1.75 2	mm
Wcc	0.5 0.75 1 1.25 1.5	mm
Lcc	30 60 90	mm

Table 5.3: Parameter Sweep Values for All Combinations

5.9.3 Auxilary Sweep

An auxiliary sweep is a computational technique used to study the influence of secondary parameters or variables that are not directly part of the main parametric study. It involves varying these auxiliary parameters to investigate their impact on the primary simulation or experiment results. This approach helps in understanding the interactions and dependencies between primary and auxiliary parameters, providing a deeper insight into the system's overall behavior. Auxiliary sweeps are particularly useful in complex models where multiple factors may indirectly affect the outcomes, aiding in fine-tuning and enhancing the accuracy of simulations.

Name	Values	Unit
Uin	1 3.25 5.5 7.75 10	m/s
Q	1272 3132 5040	W/m^{-2}
Tamb	-20 0 20 40	°C

Table 5.4: Auxilary Sweep Values for All Combinations

5.10 Results

5.10.1 Accumulated Probe Table: Set 1

1. Hcc (1 - hc)

2. Wcc (2 - wc)

3. Lcc (3 - length)

4. Tamb (4 - Tamb)

5. Q (5 - Q)

6. Uin (6 - Uin)

7. Delp (15 - Pressure (Pa), Pressure Probe 3)

Figure 5.1: Pressure Probe 3 Location

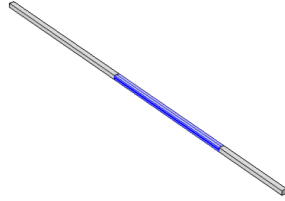8. Tsta (18 - Temperature (degC), Stack Temperature Probe 1)

Figure 5.2: Stack Temperature Probe 1 Location

5.10.2   3D Plots
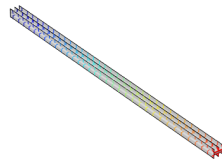
1. Pressure Drop

Figure 5.3: Pressure Drop 3D Plot

2. Temperature Stack
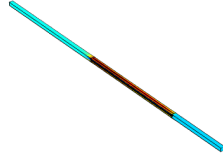

Figure 5.4: Temperature Stack 3D Plot

3. Velocity

Figure 5.5: Velocity 3D Plot

5.11   Conclusion

5.12   Outlook

5.13   References