

# key aspects, Governing Equations, Boundary Conditions for CFD simulations of Crystal Growth

---

*Research on the thermal–fluid coupling in the growth process of Czochralski silicon single crystals based on an improved physics-informed neural network*

---

## Thermal–Fluid Coupling PINN for Czochralski Silicon Crystal Growth

---

*(SI–Loss Adapted Physics-Informed Neural Network)*

---

### 1. Governing Equations (SI–Loss Adapted PINN)

Solving the **steady-state, incompressible, thermal–fluid coupling problem** for the silicon melt during Czochralski (CZ) crystal growth.

The formulation is **two-dimensional axisymmetric** and based on the **Boussinesq approximation**.

---

#### 1.1 Continuity Equation (Mass Conservation)

$$[\nabla \cdot \mathbf{u}] = 0$$

or in Cartesian form (used in the paper’s 2D model):

$$\left[ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right] = 0$$

This equation enforces incompressibility of the melt.

---

#### 1.2 Momentum Conservation (Navier–Stokes Equations)

The melt flow is governed by steady incompressible Navier–Stokes equations with buoyancy coupling.

$$[\rho (\mathbf{u} \cdot \nabla) \mathbf{u}] = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g} \beta (T - T_0)$$

In component form:

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

$$\bullet \quad \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

$$u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial y} + \nu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$$

$$\bullet \quad \nu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$$

- $\rho \beta (T - T_0)$

where:

- $(\mathbf{u} = (u, v))$ : velocity
- $(p)$ : pressure
- $(\nu)$ : kinematic viscosity
- $(\beta)$ : thermal expansion coefficient
- $(T)$ : temperature

### 1.3 Energy Equation (Heat Transfer)

$$\rho c_p (\mathbf{u} \cdot \nabla T) = k \nabla^2 T$$

or equivalently:

$$u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} = \alpha \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

where:

- $(\alpha = \frac{k}{\rho c_p})$ : thermal diffusivity

### 1.4 PINN Output Variables

The neural network approximates:

$$\{\mathcal{N}_\theta(x, y) \rightarrow \{u(x, y), v(x, y), p(x, y), T(x, y)\}\}$$

All PDE residuals are enforced using **automatic differentiation**.

## 2. Physical and Modeling Assumptions

The improved PINN relies on the following assumptions, consistent with the physics of CZ crystal growth:

### 2.1 Incompressible Melt

- Density variations are neglected **except** in the buoyancy term.
- Valid because silicon melt velocities are much smaller than the speed of sound.

### 2.2 Boussinesq Approximation

- Density variation is linearized as:  $\rho(T) \approx \rho_0 [1 - \beta(T - T_0)]$
- Buoyancy force couples temperature and momentum.
- Enables thermal–fluid coupling without full compressible flow modeling.

### 2.3 Laminar Flow Regime

- Reynolds number is **moderate**.
- Turbulence is **not modeled**.
- Flow is assumed steady and laminar, which is reasonable for typical CZ operating conditions.

---

## 2.4 Two-Dimensional Axisymmetric Approximation

- The 3D CZ furnace is reduced to a 2D axisymmetric domain.
- No azimuthal velocity component is considered.
- This greatly reduces computational cost while preserving dominant physics.

---

## 2.5 Steady-State Approximation

- Time dependence is neglected.
- The model represents a quasi-steady growth stage of the crystal.

---

## 2.6 Constant Material Properties

- Viscosity, thermal conductivity, density, and heat capacity are treated as constants.
- Radiation and phase change dynamics are simplified into boundary conditions.

---

# 3. Training Process of the Improved PINN

The paper introduces an **Improved Physics-Informed Neural Network** consisting of:

- **Spatial Information (SI) embedding**
- **Adaptive loss balancing (LB)**

---

## 3.1 Training Data

**No experimental or CFD data are used for training.**

Training data consist of:

- Interior **collocation points** for PDE enforcement
- Boundary points for BC enforcement

Inputs:

( $x$ ,  $y$ )

Outputs:

( $u$ ,  $v$ ,  $p$ ,  $T$ )

---

### 3.2 Loss Function Structure

The total loss is a weighted sum of multiple physics-based losses:

$$[\mathcal{L} = \mathcal{L}_{\text{cont}}]$$

- $\mathcal{L}_{\text{mom}}$
- $\mathcal{L}_{\text{energy}}$
- $\mathcal{L}_{\text{BC}}$  ]

where each term corresponds to the squared residual of a governing equation.

---

### 3.3 Adaptive Loss Balancing (LB)

Instead of manually tuning weights, the paper introduces **learnable log-variance parameters**:

$$[\mathcal{L} = \sum_i \left( e^{-\log \sigma_i^2} \mathcal{L}_i + \log \sigma_i \right) ]$$

This:

- Prevents domination of one PDE over others
  - Stabilizes training for multi-physics coupling
  - Improves convergence robustness
- 

### 3.4 Spatial Information (SI) Embedding

To prevent loss of spatial awareness in deep networks, spatial coordinates are injected into each hidden layer:

$$[h_l = \sigma(W_l h_{l-1}) + Z_l \odot M(x,y) + (1-Z_l) \odot N(x,y) ]$$

This:

- Preserves geometric information
  - Improves learning of boundary layers and vortices
  - Reduces gradient degradation in deep PINNs
- 

### 3.5 Optimization Strategy

- **Adam optimizer** used for initial training
  - Fixed number of epochs until residual convergence
  - Automatic differentiation used for all spatial derivatives
- 

### 3.6 Model Validation

- The trained PINN solution is compared against **COMSOL Multiphysics**
- Comparison metrics:

- Velocity fields
  - Temperature contours
  - Relative ( $L_2$ ) error norms
- COMSOL data is **not used for training**, only for validation
- 

## Summary

The paper formulates Czochralski crystal growth as a **steady, incompressible, thermal–fluid PDE system** and solves it using an **improved PINN** that:

- Enforces Navier–Stokes and energy equations
  - Uses spatially enhanced deep networks
  - Automatically balances multi-physics losses
  - Avoids reliance on labeled data
  - Achieves accuracy comparable to COMSOL
- 

If you want, I can also:

- Convert this into **LaTeX**
- Add a **dimensionless formulation**
- Compare with **standard PINN mathematically**
- Map each equation directly to PyTorch code

Just tell me how you want to proceed.

---

## SI-LB-PINN — architecture & hyperparameters()

---

### Network topology

- Base network: fully-connected MLP (feedforward PINN). ([ResearchGate](#))
- Inputs: spatial coords (x,y). Outputs: (u, v, T, P) (transverse & longitudinal velocities, temperature, pressure). ([ResearchGate](#))

### Hidden layers / neurons

- **5 hidden layers, 128 neurons per hidden layer.** ([ResearchGate](#))

### Activation & optimizer

- Activation: **tanh**.
- Optimizer: **Adam**. ([ResearchGate](#))

### Learning rate schedule

- Initial learning rate: **0.003**.
- Decay: multiply by **0.9 every 1000 iterations**. ([ResearchGate](#))

## Training data / sampling

- Boundary points: **1000 random points per boundary** (4 boundaries → 4000 boundary points).
- Interior (configuration) points: **6000 LHS (Latin Hypercube Sampling)**.
- **Total training points: 10,000.** ([ResearchGate](#))

## Loss composition & adaptive balancing (the “LB” part)

- Total loss is a weighted sum of PDE residual loss ( $L_{\text{PDE}}$ ), boundary condition loss ( $L_{\text{BC}}$ ), (and data loss if used). The paper models each loss term’s weight via an uncertainty parameter ( $\varpi$ ). ([ResearchGate](#))
- Method: treat each output / loss as Gaussian with variance ( $\varpi^2$ ); the negative log-likelihood yields loss terms of the form  $\frac{1}{2\varpi^2}L + \log\varpi$ . The variances ( $\varpi$ ) (hence the effective weights ( $\omega=1/(2\varpi^2)$ )) are **learned / updated during training** via maximum likelihood (so the network adapts the per-loss weights automatically). Equations (17)–(21) in the paper show this derivation. ([ResearchGate](#))

## Spatial-information injection (the “SI” part)

- They **reintroduce the raw spatial coordinates into each hidden layer** via two learned nonlinear encoders ( $M$ ) and ( $N$ ), and an attention/gating style combination ( $Z_l$ ). Hidden-layer update (paper notation):  $(H_1=\sigma(W_1 X+b_1))$ ,  $(Z_l=\sigma(W_l H_{l-1}+b_l))$ ,  $(H_l=(1-Z_l)\odot M + Z_l\odot N)$ , where  $(M=\sigma(W_M X + b_M))$ ,  $(N=\sigma(W_N X + b_N))$ . This preserves spatial identity and boosts expressivity. ([ResearchGate](#))

## Other practical details

- Automatic differentiation used to compute PDE residuals (standard PINN practice). ([ResearchGate](#))
  - Hardware used in experiments: Intel i5-12600K CPU, 16 GB RAM, NVIDIA RTX 3080 (8 GB VRAM) — useful to estimate compute. ([ResearchGate](#))
-