IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 8, 2019, accepted March 26, 2019, date of publication April 1, 2019, date of current version April 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2908522

Parallel Numerical Calculation on GPU for the 3-Dimensional Mathematical Model in the Walking Beam Reheating Furnace

ZHI YANG[1] AND XIAOCHUAN LUO[1,2], (Member, IEEE)
[1]State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China
[2]College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

Corresponding author: Xiaochuan Luo (luoxch@mail.neu.edu.cn)

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0304100, in part by the National Natural Science Foundation of China under Grant 51634002, in part by the Fundamental Research Funds for the Central Universities under Grant N170708020, and in part by the Open Research Fund from the State Key Laboratory of Rolling and Automation, Northeastern University, under Grant 2018RALKFKT008.

ABSTRACT In this paper, the parallel numerical simulations of 3-dimensional (3D) mathematical model for the walking-beam type reheating furnace have been developed and implemented on the graphics processing unit (GPU) architecture. First, the detailed heat transfer processes in the furnace are described and categorized when building the 3D mathematical model. They consist of the radiative heat exchange into the slab, the heat conduction between the stationary beams and the slabs, the heat convection between the gas flow and slab surfaces, and the heat conduction inside the slabs. Moreover, the proposed 3D mathematical model also accounts for the temperature-dependent material parameters, which is ignored by most published mathematical models. Second, the explicit finite difference method is used to discretize the proposed model to a straightforward parallel computation problem. A detailed analysis of the 3D boundary conditions for the proposed model is introduced and presented. The parallel computing problem is realized by programming on GPU via the platform CUDA in Tesla P100. Finally, the proposed model is verified with industry measurements and the comparison between the GPU-implementation model and the CPU-implementation model is also given to validate the great acceleration. The experimental results prove that the proposed GPU-implementation model declines the computation time from hours to seconds. It is not only orders of magnitude faster but also highly accurate.

INDEX TERMS Reheating furnace, heat transfer model, graphics processing unit (GPU), CUDA, PDE, explicit finite difference method.

I. INTRODUCTION

In the process chain of steel industry, continuous reheating furnaces are often used for reheating or heat treatment of steel products before and after the rolling mill, as illustrated in Fig. 1. The furnace operates with various types of fuel to raise the temperature of steel slab up to the uniform target temperature, which is approximately 1300-1500 K at the exit of furnace [1]. Basically, the reheating furnace is a high energy consumption unit in the hot strip mill. In a normal year, the furnace processes 750 000 t of steel and consumes 1250 TJ of primary energy [2]. In [3], it is estimated that reheating furnace accounts for approximately 60% of energy consumption of the steel industry. Regardless of the furnace type employed, the task of the reheating operation is to obtain a desired discharging temperature with good uniformity in slabs when slabs are discharged from the reheating furnace. To achieve this purpose, modeling, optimizing, and controlling of the furnace temperature are the key for the reheating furnace. The modeling of furnace is the cornerstone of the whole control process. Since the temperature distribution of the slab in the reheating furnace is very difficult to detect, the most common method is establishing a mathematical model to calculate it. Thus, the established heat transfer model plays a crucial role by providing the temperature distribution data of steel slabs. And the accurate and rapid heat transfer

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Touriño.

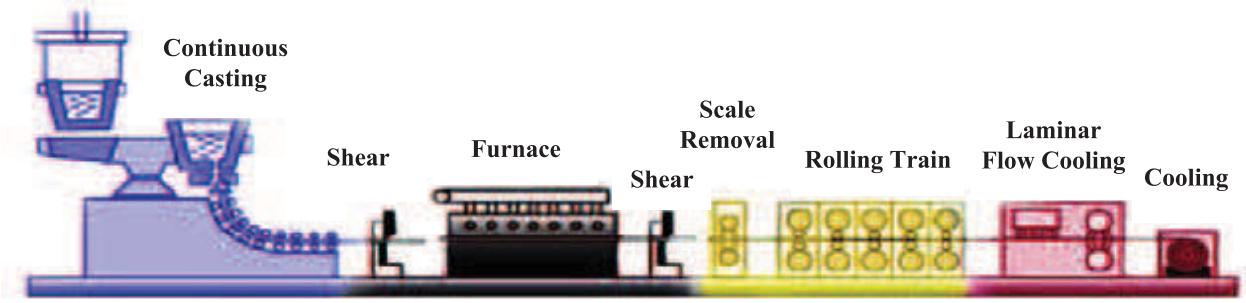


FIGURE 1. The main process of the hot rolling line.

mathematical model of the reheating furnace is of great importance for a successful application of control system for the reheating furnace.

A. EXISTING MODELS FOR THE REHEATING FURNACE

Based on the incorporated physical theory, models of reheating furnaces may be divided into two categories: the black box models and the white box models [4]. The black box models are often built by system identification tools. The disadvantage of such models is the restricted validity to the considered systems [5]. In contrast, the white box heat transfer mathematical models are adapting the fundamental equations of physical phenomena to reflect the heat transfer inside the reheating furnace. Hence, their structures and model parameters have a clear nexus to phenomena and quantities observed in the real system. Then, a simple overview of existing white box mathematical models from relevant publications will be given.

Firstly, the 1-dimensional (1D) heat transfer model is widely accepted in many studies. In general, the temperature profile of the slab is assumed as 1 dimensional along the thickness direction. Assumptions, which simplify the complex and accurate models to the 1-dimensional heat transfer model, can be found in [2]. In essence, the 1D models are usually computationally inexpensive, which makes them suitable for optimization tasks. For instance, A. Steinboeck et al. propose a reduced 1D model which counts only for radiative heat exchange in the pusher-type furnace and heat conduction inside the slabs in the paper [6]. It is proved that the reduced model is suitable for optimization and control applications in the scheduling and controlling the furnace process. Yang et al. [7] propose a 1D mathematics heat transfer model for the reheating furnace, and they solve the PDE optimal control problem for the steel slabs in the walking beam reheating furnace on the steady-state working operating. Model validation and comparison between the 1D model and the experiment results prove the effectiveness of the proposed 1D model.

Secondly, the 2-dimensional (2D) temperature profiles in the slabs are predicted and provided in some published models by a number of other researchers. According to the practical engineering application, more information of the slab's temperature profile (e.g., the influence of skids on the inhomogeneity of the slab temperature profile) should be obtained in these models. For instance, Ezure et al. [8] propose a 2D (slab thickness direction and slab length direction) model for slab temperature calculation. The temperature distribution of the slab is obtained by using a finite difference approximation for the reheating furnace. Li et al. [9] make the simultaneous estimation of heat fluxes through the upper, side and lower surface of a steel slab based dynamic matrix control (DMC) in a walking beam type reheating furnace. Luo et al. [10] build the 2D temperature prediction model rather than a precise 1D model and introduce the first-optimize-then-discretize approach to solve the 2D optimal control problems. Model validation and comparison indicate that the present 2D model works well for the prediction of thermal behavior about the slab in the reheating furnace. Kim [11] propose the finite volume method (FVM) and a 2D unsteady heat transfer model to analyze the transient heating of a slab. They calculate the radiation heating exchange among the furnace, the slab and the combustion gas along the slab's marching direction in the reheating furnace.

Thirdly, the full 3-dimensional (3D) heat transfer model is also simulated by the following researchers. Most of them are applied in commercial software, such as Fluent, STAR-CD, and so on. For instance, Kim et al. [12] apply the FLUENT software to perform 3D CFD analysis for the turbulent reactive flow and radiative heat transfer in a walking beam type reheating furnace. They calculate the heat fluxes through the upper and lower surface of the slabs and temperature distribution in the furnace. Hsieh et al. [13] analyze a 3D turbulent combusting flow with the radiative heat transfer analysis in a walking beam type reheating furnace with the commercial STAR-CD code. Jaklic et al. [14] propose the Monte Carlo method for radiation and 3D finite-difference method for heat conduction in the billets. Then, they obtain the relationship between the space of billets and the productivity of a continuous walking-beam furnace. Dubey et al. [15] present 3D heat conduction model for steel billet heating in the reheating furnace. Conduction heat transfer within the billets is modeled by the Finite Difference Method (FDM) and fully implicit spatial discretization approximation is used for three dimensional heat diffusion equation of billet.

According to the researches before, only relatively simple 1D or 2D computational models can be used in real-time

applications, because they require the less computational time. Most frequently, the fast computational speed and high accuracy of a 3D mathematical model are often conflicted. Depending on the applications in practice, if the 3D heat transfer model for real-time simulation can be solved fast enough, it will be a better choice for modeling of the reheating furnace. Therefore, it is very necessary to create an effective 3D computational model, which is physically advanced, accurate, reliable, and also very fast to evaluate.

B. GPU DEVELOPMENT

A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. Although primarily designed for the computer graphics and image rendering, GPUs are nowadays widely used for massive parallel computing in science and engineering. In recent years, GPUs have encountered a vast development and a steep rise of their computational performance. Now, GPU accelerators power the energy-efficient data centers in government labs, universities, enterprises, and small-and-medium businesses around the world. Therefore, more and more software developers, scientists and engineers are using GPUs instead of CPUs as a cost-effective high-performance computing platform in various computational tasks.

In general, the broad-ranging applications involve robots, material sciences and artificial intelligence, and so on. For instance, Hyoungseok Chu et al. [16] report some results on implementation of the Craig-Sneyd ADI scheme to solve a three-dimensional parabolic problem by using GP-GPU of NVIDIA. Its computational efficiency is confirmed that the speedup reaches 13 times in single precision and 8 times in double precision. Micikevicius et al. [17] describe a GPU parallelization approach for 3D finite difference stencil computation. The approach achieves approximately an order of magnitude speedup over similar seismic industry standard codes. They also describe the approach for utilizing multiple GPUs to solve the problem by using asynchronous communication and computation. It allows for GPU processing of large data sets in practice, often exceeding 10 GB in size. Klimes and Štětina [18] present a GPU-based model for continuous casting of steel. The computational model is implemented as highly-parallel with the use of the NVIDIA CUDA architecture. The corresponding gain of the GPU-acceleration is between 33 and 68. Cheng et al. [19] study the anomalous thermal-fluid properties of nanofluids. A simplified computational approach for isothermal nanofluid simulations is applied, and simulations are conducted by using both conventional CPU and parallel GPU implementations. Through their work on parallel GPU implementations, a conclusion is drew that the GPU simulations are approximately 1000-2500 times faster than the CPU simulations.

To the best of our knowledge, no previous studies of applying GPU-based parallel computation for the reheating furnace have been developed. Combining parallel computation technique, a 3D numerical heat transfer model implemented by GPU architecture with CUDA will be given and proved in this paper. The proposed model is both fast enough and sufficiently accurate.

The structure of this paper is constituted as follows. In section II, a detailed 3D numerical heat transfer model for the reheating furnace is proposed. And the temperature-dependent thermophysical properties for the steel slab are also given. In section III, the explicit finite difference method is used for the model implementation to approximate differential increment in temperature, space and time. Afterwards, the implementation of computational model in CUDA/C++ is introduced in detail. In section IV, the simulations are made to verify and validate the effectiveness of the proposed 3D mathematical model. Finally, the conclusion of this paper is summarized in section V.

II. 3D COMPUTATIONAL MATHEMATICAL MODEL

The main function of modeling the reheating furnace is reflecting the heat exchange into the slabs and the heat conduction inside the slabs. In general, the heat conduction inside the slabs is formulated by the Fourier's law, which is the standard approach. However, there are many different ways of reflecting the heat exchange into the slabs. They will be discussed in detail in this section. Besides, the proposed mathematical model also accounts for the temperature-dependent material parameters, which is ignored by most published mathematical models. Finally, the new 3D computational mathematical model considering the temperature-dependent material parameters can be obtained and shown as below.

A. MATHEMATICAL FORMULATION OF HEAT TRANSFER OF THE SLAB

In this analysis, the heat conduction process inside the slabs with temperature-dependent material properties can be defined by the Fourier's law as follows:

$$\rho c\left(T\right)\frac{\partial T}{\partial t}=\frac{\partial}{\partial x}\left(\lambda\left(T\right)\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\left(T\right)\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\left(T\right)\frac{\partial T}{\partial z}\right). \quad (1)$$

Here $(x, y, z, t) \in \Omega_{xyzt} = [0, l_x] \times [0, l_y] \times [0, l_z] \times [0, t_f]$, and $T = T(x, y, z, t)$ is the slab temperature, (x, y, z) is the vector of spatial coordinates, t is the reheating time, ρ is density (kg/m^3), $c(T)$ denotes specific heat $(J/(kg \cdot K))$, $\lambda(T)$ stands for thermal conductivity $(W/(m \cdot K))$, and l_x, l_y, l_z are width, thickness and length of the steel slab (m), t_f is the residence time for the steel slab in the reheating furnace. To solve the heat transfer equation (1), the initial and the boundary conditions should be given.

Firstly, the initial temperature of the steel slab before charging could be obtained by infrared pyrometer on charging door. Here, a homogeneous temperature profile inside the

steel slab is accepted. Thus,

$$T(x, y, z, 0) = T_0. \tag{2}$$

Secondly, the heat transfer on the boundary of the slabs should be considered carefully. For the slab, there are six surfaces altogether and each surface can be provided with a different boundary condition, which is a major advantage in the proposed computational mathematical model.

Considering these six surfaces, five surfaces except the bottom surface can be classified as the same category. For these five surfaces, heat flux on the boundary can be obtained by considering both the convective heat transfer and the radiative heat transfer. Because the slab reheating furnace operates at relatively high temperature levels, the radiative heat transfer, which is characterized by Stefan-Boltzmann radiation law, is the dominant mode of the whole heat exchange. In some way, almost all models account for the radiative heat transfer into the slabs, but many authors neglect the convective heat transfer. According to [2], up to approximately 5% of the total heat input into the slabs is caused by heat convection between the gas flow and surfaces. Thus, the whole total heat flux for these five surfaces can be obtained from the summation of the convective and radiative heat fluxes, which is shown as follow:

$$\begin{aligned} q_i &= q_r + q_c, \quad i = 1, 2, 3, 4, 5. \\ q_r &= \sigma\varepsilon\left(T_f^4 - T_s^4\right), \quad q_c = h_c\left(T_f - T_s\right). \end{aligned} \tag{3}$$

Here, the symbol q_c is the convective heat flux between the furnace gas and the corresponding slab surface and q_r is the radiative heat flux on the slab surface. And the symbol σ is the Stefan-Boltzmann constant, $5.67 * 10^{-8}\left(W/m^2 \cdot K^4\right)$, the symbol h_c stands for the gas convective heat transfer coefficient at the surface of the slab, 7.8 $\left(W\left/\left(m^2 \cdot K\right)\right.\right)$, the symbols T_f and T_s are the furnace temperature and slab surface temperature, respectively. Besides, the symbol ε is the total heat exchange factor at the surface of the slab, and its value can be calculated by the following equation:

$$\varepsilon = \frac{\varepsilon_g\varepsilon_s\left(1 + \varphi_{ws}\left(1 - \varepsilon_g\right)\right)}{\varepsilon_g + \varphi_{ws}\left(1 - \varepsilon_g\right)\left[\varepsilon_s + \varepsilon_g\left(1 - \varepsilon_g\right)\right]}, \tag{4}$$

where the symbol ε_s is the slab emissivity determined by steel grade and temperature. The symbol ε_g is furnace gas emissivity related to gas composition, which is determined by the proportions of CO_2 and H_2O in furnace gas. The symbol φ_{ws} is the shape factor of furnace chamber to the slab, which can be calculated by $\varphi_{ws} = \frac{L_s N_s}{2(H+B)-L_s N_s}$. Here, the symbol L_s is the length of the slab and N_s is the number of slab rows in the reheating furnace. The symbols H, B are the height and breadth of the furnace.

At the bottom surface, the slabs are supported on the stationary beams and moved by the walking beams. As shown in Fig. 2, all the walking and stationary beams are simplified to have rectangular cross sections. A skid rail is put on each stationary beam and it has a 5 mm $*$5 mm cross

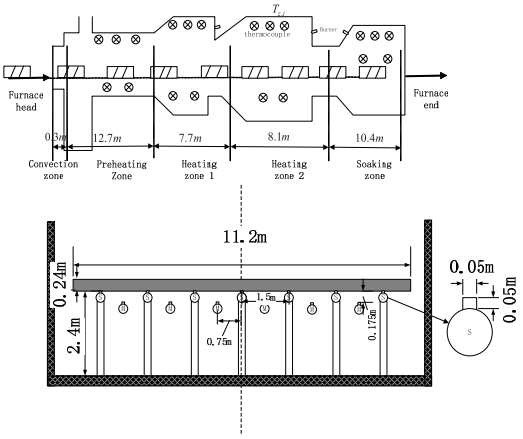


FIGURE 2. The structure of the furnace and the layout of the slab with skid support.

section normal to the axial direction. The beams in the walking beam furnace are walked as follows. Firstly, the actual walking beam will lift the slabs and move them forward. Secondly, the slabs are lowered onto the stationary beams. Thirdly, while the slabs are resting on the stationary beams, the walking beam moves underneath the slabs back to the home position ready to perform another walk. Thus, the skid mark would be formed because of the heat transfer at the area of contact between the slab and the beams. And, the whole bottom surface can be divided as walking beam skid region, stationary beam skid region and non-skid region. In the non-skid region, the calculation of the heat flux can be the same as the former 5 surfaces:

$$q_{\text{nonskid}} = q_r + q_c, (x, z) \in \Omega_{\text{nonskid}}. \tag{5}$$

For the walking beam skid region, the contact time between the walking beam and the slab is small. Then, the total heat flux on the walking beams in the skid region should add a shadow factor K_c of 0.8 to represent heat flux reduction and can be described as:

$$q_{\text{skid\_w}} = K_c q_{\text{nonskid}}, \quad (x, z) \in \Omega_{\text{skid\_w}}. \tag{6}$$

The stationary beam skid region of bottom surface for the slab is shielded and cooled by the skids, and temperature distribution adjacent to the contacting skids is expected to be depressed. Here, the heat conduction between the stationary beams and the slabs will be the main heat transfer method. A constant temperature T_w is imposed at the slabs of the cooling pipes inside the beams. The heat flux in the stationary beam skid region should be described by the following equation as:

$$q_{\text{skid\_s}} = h_c\left(T_w - T\left(x, 0, z, t\right)\right), \quad (x, z) \in \Omega_{\text{skid\_s}}. \tag{7}$$

Here, the symbols Ω_{nonskid}, $\Omega_{\text{skid\_w}}$, $\Omega_{\text{skid\_s}}$ mean the non-skid region, walking beam skid region and stationary beam skid region at the bottom surface. Besides, $T_w = 309.5$ K, which stands for the temperature of cooling water.

Finally, all boundary conditions for the proposed 3D mathematical model will be given as follows:

$$
\begin{aligned}
-\lambda(T)\left.\frac{\partial T}{\partial x}\right|_{x=l_x} &= \sigma\varepsilon\left(T_{fe}^4 - T^4\right) + h_c\left(T_{fe} - T\right),\\
\lambda(T)\left.\frac{\partial T}{\partial x}\right|_{x=0} &= \sigma\varepsilon\left(T_{fe}^4 - T^4\right) + h_c\left(T_{fe} - T\right),\\
-\lambda(T)\left.\frac{\partial T}{\partial y}\right|_{y=l_y} &= \sigma\varepsilon\left(T_{ft}^4 - T^4\right) + h_c\left(T_{ft} - T\right),\\
\lambda(T)\left.\frac{\partial T}{\partial y}\right|_{y=0} &= \begin{cases}\sigma\varepsilon\left(T_{fb}^4 - T^4\right) + h_c\left(T_{fb} - T\right),\\ K_c\left[\sigma\varepsilon\left(T_{fb}^4 - T^4\right) + h_c\left(T_{fb} - T\right)\right],\\ h_c\left(T_w - T\right).\end{cases}\\
-\lambda(T)\left.\frac{\partial T}{\partial z}\right|_{z=l_z} &= \sigma\varepsilon\left(T_{fe}^4 - T^4\right) + h_c\left(T_{fe} - T\right),\\
\lambda(T)\left.\frac{\partial T}{\partial z}\right|_{z=0} &= \sigma\varepsilon\left(T_{fe}^4 - T^4\right) + h_c\left(T_{fe} - T\right). \qquad (8)
\end{aligned}
$$

Here, the T_{fb}, T_{ft} are the bottom and top furnace temperature. And, $T_{fe} = \frac{T_{ft}+T_{fb}}{2}$, which stands for the edge furnace temperature at other surfaces. In general, the furnace temperature distribution T_f along the whole furnace will serve as the input of the proposed 3D heat transfer model.

B. MATERIAL PROPERTIES OF STEEL SLAB

In general, the temperature-dependent material properties of steel slab (the thermal conductivity, the density, and the specific heat) are important inputs to the computational model. However, in most published papers, the temperature-dependent material properties for the steel slab are often ignored, because the calculation of them will cost a lot of computational power and waste much larger running time to solve the mathematical heat transfer model. It is not worth the cost. In this paper, the temperature-dependent material properties of steel slab are considered in the proposed 3D model. The computation process is parallelizable and independent. Thus, it suits the parallel computation and allows for the acceleration of computations.

As we know from the book [2], the specific heat c and the thermal conductivity λ may explicitly depend on the location (x, y, z) or on the current local temperature $T(x, y, z, t)$ or both. And in the reheating furnace, a homogenous material is postulated for the steel slabs, which means the dependence of the material properties from (x, y, z) can be ignored. In this paper, the specific heat c and the thermal conductivity λ only depend on the current local temperature $T(x, y, z, t)$. Since experimental investigations of material properties in a working environment are generally expensive and time-consuming, the calculation formulas given by other papers are used here to describe the material properties of steel slab. For instance, the sampling steel slab is the lower carbon steel of 20MnSi in Chinese National Standard, whose yield strength is not less than 300 MPa. It is widely used in infrastructure, whose major composition is listed in Table 1. From the paper [20], the specific heat of 20MnSi slab is piecewise

TABLE 1. Steel grade composition.

	C	Si	Mn	P	S
$\leq$ (%)	0.23	0.7	1.7	0.045	0.045

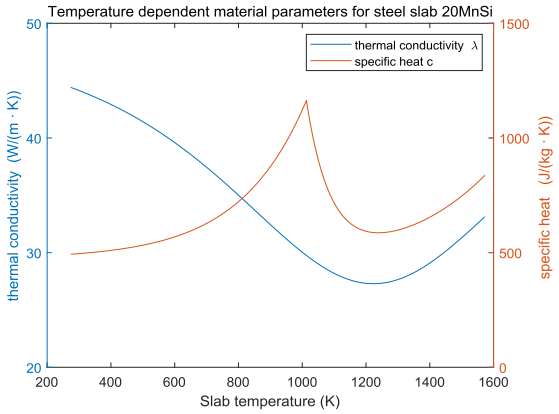


FIGURE 3. Temperature dependent material parameters for steel slab 20MnSi.

function of the slab temperature

$$
c(T) = 472.3 + 98.23\left(\frac{T}{1000}\right)^5 + 668.8e^{-a|T-740|}. \qquad (9)
$$

Here, T is the slab's temperature. If $T - 740 < 0$, $a = 0.0047$, otherwise $a = 0.0135$. So does the heat conductivity coefficient, the function is expressed as below:

$$
\lambda(T) = 48.77 - \frac{21.48}{ch[0.24\frac{T-950}{100}]}. \qquad (10)
$$

Because the density changes very small with the slab's temperature, it is constant: $\rho = 7850\,\mathrm{kg/m^3}$. Fig. 3 will show the temperature-dependence of the thermal conductivity and of the specific heat for the steel slab 20MnSi, which will be used in the model validation simulation in Section IV.

III. NUMERICAL TECHNIQUE AND PARALLEL STRATEGY

In this section, the 3D discretization heat transfer model is discretized by the explicit finite difference method. Then, it will be realized by programming on the graphics processing unit (GPU) via the platform CUDA in Tesla P100 produced by NVIDIA company. And C++ language is extended to create codes for computer formulation of the proposed 3D discretization heat transfer model.

A. DISCRETIZATION BY THE EXPLICIT FINITE DIFFERENCE METHOD

The explicit finite difference method [21] is used to approximate the differential in temperature, space, and time. Although, the finite difference method is explicit, conditionally stable, and the size of the time step is therefore limited. It is also more straightforward to parallelize an explicit discretization. As a result, the explicit finite difference method is therefore essential for launching the presented model on graphics processing units(GPU).

Defining the thermal diffusivity $\alpha(T) = \frac{\bar{\lambda}(T)}{\rho\bar{c}(T)}$. Here, the harmonic mean method is used and then the function $\bar{\lambda}(T^n)$ can be expressed as:

$$\begin{aligned}\bar{\lambda}\left(T^n\right) = &\frac{2\lambda\left(T^n_{i,j,k}\right)\lambda\left(T^n_{i+1,j,k}\right)}{3\left(\lambda\left(T^n_{i,j,k}\right)+\lambda\left(T^n_{i+1,j,k}\right)\right)} \\ &+\frac{2\lambda\left(T^n_{i,j,k}\right)\lambda\left(T^n_{i,j+1,k}\right)}{3\left(\lambda\left(T^n_{i,j,k}\right)+\lambda\left(T^n_{i,j+1,k}\right)\right)} \\ &+\frac{2\lambda\left(T^n_{i,j,k}\right)\lambda\left(T^n_{i,j,k+1}\right)}{3\left(\lambda\left(T^n_{i,j,k}\right)+\lambda\left(T^n_{i,j,k+1}\right)\right)}. \end{aligned} \tag{11}$$

Similar to the heat conductivity coefficient, the function of the thermal conductivity $c(T^n)$ follows the same idea. Thus, the heat conduction equation (1) can be rewritten as a second-order parabolic partial differential equation:

$$\frac{\partial T}{\partial t} = \alpha(T)\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right). \tag{12}$$

To discretize and establish a finite-difference solution method of the partial differential equation, two steps should be made. Firstly, discretizing the continuous space domain into a grid with a finite number of grid points. At time step $n \in [1, N_n]$, the temperature at grid point (x, y, z) can be replaced by $(i \in [1, N_i], j \in [1, N_j], k \in [1, N_k])$, which is denoted as $T^n_{i,j,k}$ for the rest of the paper. Here, N_n is the number of cells in time, and N_i, N_j, N_k are the number of grid cells in the direction (x, y, z). According to central finite-difference discretization, the second-order partial derivative of T with respect to x can be expressed as

$$\begin{aligned}\frac{\partial^2 T}{\partial x^2} &= \frac{T^n_{i+1,j,k}+T^n_{i-1,j,k}-2T^n_{i,j,k}}{(\Delta x)^2}+O(\Delta x)^2 \\ &\approx \frac{T^n_{i+1,j,k}+T^n_{i-1,j,k}-2T^n_{i,j,k}}{(\Delta x)^2}. \end{aligned} \tag{13}$$

where the truncation error (TR) is $O(\Delta x)^2$. Then, a similar process can be applied to the y and z directions. Secondly, considering the time discretization problem. Here, the forward-difference with time on the left-hand side of (12) is the energy stored from time step n to $n+1$ in the control unit volume. Applying the explicit update on the right-hand side of the discretized form of (12) at time step, the following equation can be obtained

$$\frac{\partial T}{\partial t} = \frac{T^{n+1}_{i,j,k}-T^n_{i,j,k}}{\Delta t}+O(\Delta t) \approx \frac{T^{n+1}_{i,j,k}-T^n_{i,j,k}}{\Delta t}. \tag{14}$$

Thus, the heat conduction equation (12) can be discretized in the following form:

$$\begin{aligned}T^{n+1}_{i,j,k} = &\left(1-2\left(r_x+r_y+r_z\right)\right)T^n_{i,j,k} \\ &+r_x\left(T^n_{i+1,j,k}+T^n_{i-1,j,k}\right) \\ &+r_y\left(T^n_{i,j+1,k}+T^n_{i,j-1,k}\right) \\ &+r_z\left(T^n_{i,j,k+1}+T^n_{i,j,k-1}\right). \end{aligned} \tag{15}$$

Here, $r_x = \alpha(T^n)\frac{\Delta t}{(\Delta x)^2}$, $r_y = \alpha(T^n)\frac{\Delta t}{(\Delta y)^2}$, $r_z = \alpha(T^n)\frac{\Delta t}{(\Delta z)^2}$.

Finally, the following set of nonlinear equations can be obtained and shown as:

$$T^{n+1}_{i,j,k} = \Phi\left(T^n_{i,j,k}, T^n_{i\pm1,j,k}, T^n_{i,j\pm1,k}, T^n_{i,j,k\pm1}, \alpha\left(T^n\right)\right), \tag{16a}$$

$$\alpha\left(T^n\right) = \frac{\bar{\lambda}\left(T^n\right)}{\rho\bar{c}\left(T^n\right)}, \tag{16b}$$

$$\begin{aligned}\bar{\lambda} = f\Big(&\lambda\left(T^n_{i,j,k}\right), \lambda\left(T^n_{i+1,j,k}\right), \lambda\left(T^n_{i,j+1,k}\right), \\ &\lambda\left(T^n_{i,j,k+1}\right)\Big), \end{aligned} \tag{16c}$$

$$\begin{aligned}\bar{c} = g\Big(&c\left(T^n_{i,j,k}\right), c\left(T^n_{i+1,j,k}\right), c\left(T^n_{i,j+1,k}\right), \\ &c\left(T^n_{i,j,k+1}\right)\Big). \end{aligned} \tag{16d}$$

This method has second-order accuracy in space and first-order accuracy in time. From the paper [22], a stability criterion must be given for the time step Δt to keep the error in the result bounded. Hence, under the linear assumption for the above problems, the stability constraint is defined as

$$\gamma = \alpha(T)\Delta t\left(\frac{1}{(\Delta x)^2}+\frac{1}{(\Delta y.)^2}+\frac{1}{(\Delta z)^2}\right) \le \frac{1}{2}, \tag{17}$$

This inequality restricts the size of the time step Δt for the given space increments $\Delta x, \Delta y$, and Δz. More details are given and shown in the following simulations in Section IV-D.

So far, the discussed equations above are only for the points inside the slab. In the remainder of this section, the equations related to the boundary conditions will be discussed. The central-difference approximation will be used to discretize the boundary condition equations to achieve second-order accuracy. Here, the virtual temperature nodes $T^n_{0,j,k}, T^n_{N_i+1,j,k}, T^n_{i,0,k}, T^n_{i,N_j+1,k}, T^n_{i,j,0}, T^n_{i,j,N_k+1}$ are introduced by expanding the distance $\Delta x, \Delta y, \Delta z$ to external boundary at $x = 0, x = l_x; y = 0, y = l_y; z = 0, z = l_z$. To simplify the process, the top surface boundary condition is taken as representative. Then, the expression for the boundary conation at the top surface (i, N_j, k) is given by the following:

$$\begin{aligned}-\lambda\left(T^n_{i,N_j,k}\right)&\frac{T^n_{i,N_j+1,k}-T^n_{i,N_j,k}}{\Delta x} \\ &= \sigma\varepsilon\left(\left(T_{ft}\right)^4-\left(T^n_{i,N_j,k}\right)^4\right)+h_c\left(T_{ft}-T^n_{i,N_j,k}\right). \end{aligned}$$

Thus, the virtual point can be expressed as:

$$\begin{aligned}T^n_{i,N_j+1,k} = &T^n_{i,N_j,k}-\frac{h_c\Delta x}{\lambda\left(T^n_{i,N_j,k}\right)}\left(T_{ft}-T^n_{i,N_j,k}\right) \\ &-\frac{\sigma\varepsilon\Delta x}{\lambda\left(T^n_{i,N_j,k}\right)}\left(\left(T_{ft}\right)^4-\left(T^n_{i,N_j,k}\right)^4\right). \end{aligned} \tag{18}$$

The other virtual points $T^n_{0,j,k}, T^n_{N_i+1,j,k}, T^n_{i,0,k}, T^n_{i,j,0}$, and T^n_{i,j,N_k+1} can be derived in the same way. Then, these derived

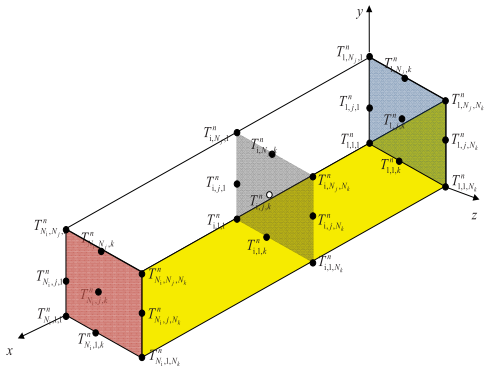


FIGURE 4. 26 different locations of the point $T^n_{i,j,k}$ on the boundary surface.

FIGURE 5. The native difference between CPU and GPU architectures.

virtual points can be used to eliminate the boundary points. For the proposed 3D model, there exists 6 surfaces, 12 edges, and 8 corner points at the boundary. Thus, there are 26 different types locations of the point $T^n_{i,j,k}$ on the boundary surface as shown in Fig. 4. For instance, $T^n_{1,N_j,k}$ shows that the location of the points is on the edge intersection of the surface $x = 0$ and surface $y = l_y$. Here, the virtual points $T^n_{0,N_j,k}$, $T^n_{1,N_j+1,k}$ will be used to calculate this edge boundary. Thus, the mathematical expression of the edge $(1, N_j, k)$ can be obtained as:

$$T^{n+1}_{1,N_j,k} = \left(1 - r_x - r_y - 2r_z\right) T^n_{1,N_j,k} + r_x T^n_{2,N_j,k} + r_y T^n_{1,N_j-1,k} + r_z \left(T^n_{1,N_j,k+1} + T^n_{1,N_j,k-1}\right) + b. \tag{19}$$

with

$$b = \sigma\varepsilon \left(a_x \left(T_{fe}\right)^4 + a_y \left(T_{ft}\right)^4 - \left(a_x + a_y\right) \left(T^n_{1,N_j,k}\right)^4\right) + h_c \left(a_x T_{fe} + a_y T_{ft} - \left(a_x + a_y\right) T^n_{1,N_j,k}\right). \tag{20}$$

Here $a_x = \frac{\Delta t}{\rho\bar{c}\left(T^n_{1,N_j,k}\right)\Delta x}$, $a_y = \frac{\Delta t}{\rho\bar{c}\left(T^n_{1,N_j,k}\right)\Delta y}$. Furthermore, the difference equations of the 26 different locations on the boundary conditions can similarly be derived by substituting virtual points for each boundary condition.

B. PARALLEL IMPLEMENTATION IN CUDA/C++

In general, programming on GPU requires a minimum understanding of the underlying mechanisms of the API and hardware features. Thus, it seems reasonable to give a brief description of GPU hardware and software. GPUs usually have something on the order of 10 to 100 times more raw compute power than the CPU. GPU computing offers a huge computational performance with a very favorable ratio between the price and the performance of GPUs [18]. In order to understand the GPU-accelerated computing, the native difference between CPU and GPU architectures is introduced, as shown in Fig. 5. From Fig. 5, it is clear that CPU collects arithmetical and logical (ALU) units, a complex unit control, and various memories with multiple levels of associated cache. Nowadays, CPUs only contain up to 12 cores, whereas GPUs contain thousands of cores, tens of thousands of threads, and can achieve the peak performance of several tera-floating-point operations per second (FLOPS). The NVIDIA Tesla P100 will be used to execute the program. The Tesla P100 GPU incorporates 3584 CUDA cores, each running at the frequency of 1.15 GHz.

In this paper, the "single precision" ("float" in c++ programming language) is used for the calculation. Since the background of this paper is the reheating furnace, where the accuracy is 10^{-2}. And the data from the real-world sensors do not require high precision computation. Thus, the "single precision" is enough. Besides, there are two advantages of using "single precision": reduced memory storage and faster calculations on the GPU. According to the specifications given by the vendors, theoretical peak performance of P100 is up to 4.7 TFLOPS for double precision and 9.3 TFLOPS for single precision. Afterwards, the implementation of computational model in the CUDA/C++ should be given in detail by the following.

Firstly, programming in the CUDA platform is much different from the CPU programming. In the CUDA programming, there consists two parts: the CPU host and GPU device. The CPU is referred to as a host. It sequentially controls the program flow and launches the kernel on a GPU. In contrast, the GPU performs a huge amount of computations by means of a parallel thread execution of a set of instructions referred to as a kernel [23]. The kernel is not a complete program, because it incorporates only operations performed in a parallel manner. The execution of the kernel instructs the GPU executing the same code but each thread with different data. In general, the CUDA environment launches parallel threads in a grid, which is the collection of blocks. And the block configuration tells the GPU how many threads should be used to execute the code. In summary, the programmer has to follow some steps to execute a code on the GPU: First, memory must be allocated in the GPU device in advance. Second, data are copied from CPU host to the GPU device, then the kernel function is triggered and executed on the device. Finally, the results of the device must be copied back to the host. As shown in Fig. 6, the arrangement of the GPU-based computation for the proposed problem is described.

Secondly, the details of executing the main CUDA kernel is given. The pseudocode of the main CUDA/C++ kernel reads as shown by Algorithm 1. First, when main CUDA kernel is triggered at time t^n, it creates $(N_i * N_j * N_k)$ threads.

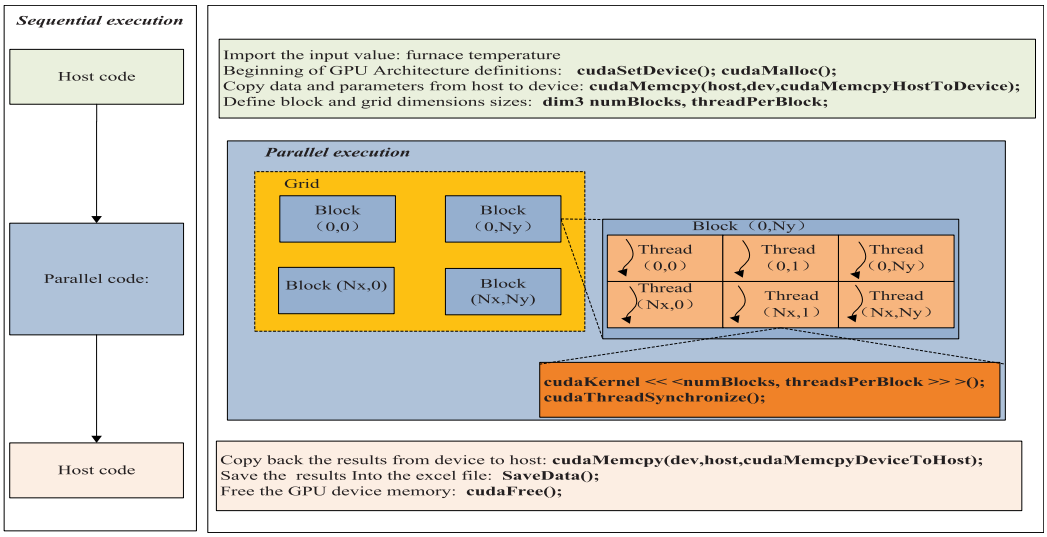


FIGURE 6. The flowchart of the GPU-implementation in CUDA/C++.

Algorithm 1 The GPU Kernel: cudaKernel()

Begin

ID : assign thread ID in the CUDA grid to a particular control volume
i = threadIdx.y,
j = (blockIdx.x * blockDim.x) + threadIdx.x,
k = blockIdx.y,
ID = i * N_j * N_k + j* N_k + k.

c, λ : update the thermophysical properties of steel slab based on the thread ID by (16c) and (16d)

α : obtain the thermal diffusivity by (16b)

T^{n+1}: determine the temperature from T^n for each ID by (16a)

end

Each thread first retrieves its identification ID according to their grid and block position. A particular node (i, j, k) can be obtained. In the next step, the thermal conductivity and the specific heat are updated based on the thread ID. The update of the heat transfer coefficient uses a multi-dimensional interpolation method and other computationally demanding operations. Once the thermophysical properties are calculated, each thread proceeds with the temperature determination of T^{n+1} by (16a), which is the last procedure of the kernel.

The most computationally demanding parts of the programming the 3D model are the iterative loop calculation of the transient temperature and the thermophysical properties calculation. From the view of computational point, nested loops belong to common bottlenecks of the computational efficiency [18]. Moreover, each node requires the thermophysical properties calculation to determine the specific heat c and the thermal conductivity λ. Obviously, the computation of thermophysical properties for each node in the discretization model is generally expensive and time-consuming.

IV. PERFORMANCE ANALYSIS

In order to explain the computationally demanding of the iterative loop calculation, the implementation of computational model on CPU is given here as comparative object. In general, a sequential CPU-implementation of the model incorporates three nested loops in the space domain, one for each spatial coordinate in 3D. However, four nested loops are needed for the computation by considering another loop required for iterations in time. Therefore, for the discretization model obtained in Section III-A, the computing unit by implementation on CPU has to perform $(N_n * N_i * N_j * N_k)$ sequential operations to complete the simulation.

Then, the proposed 3D mathematical model is carried out on both the CPU and the GPU in this section. The simulations are made and analyzed to give the verification and validation. In order to assess the efficiency, the performances are demonstrated and arranged in the following simulations: Firstly, the validation between the proposed 3D mathematical model and the industry measurements is given to determine whether the computer model provides results consistent with the behavior of the real system modelled. Secondly, the detailed temperature distribution of the slab obtained by the proposed 3D mathematical model during the entire reheating process will be indicated. Thirdly, the comparison between the implementations on GPU and CPU architectures is made to prove the necessity of implementation on GPU. Finally, the bottlenecks on the GPU acceleration factor are also analyzed.

A. VALIDATION BETWEEN PROPOSED 3D MATHEMATICAL MODEL AND THE INDUSTRY MEASUREMENTS

In order to validate the proposed model, the industry data obtained by the trial steel slab experiments are given. These data are obtained from a walking beam type reheating furnace in ShouQin company. The layout of a walking beam type reheating furnace can be found in Fig. 2. The length of this

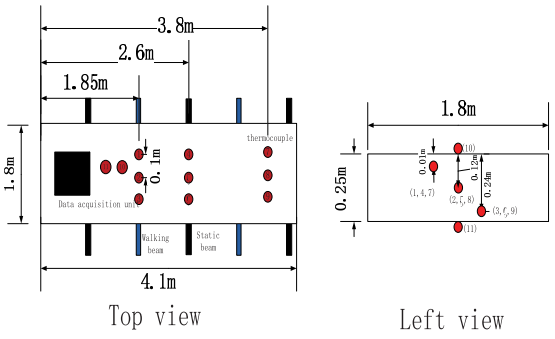


FIGURE 7. Thermocouple locations in the trial steel slab (Top view and left view).

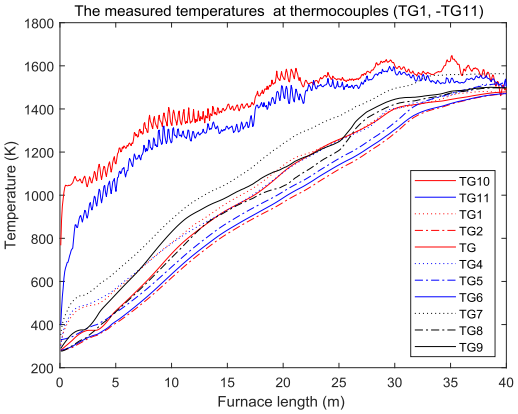


FIGURE 8. The measured furnace temperature by thermocouples (TG1- TG11).

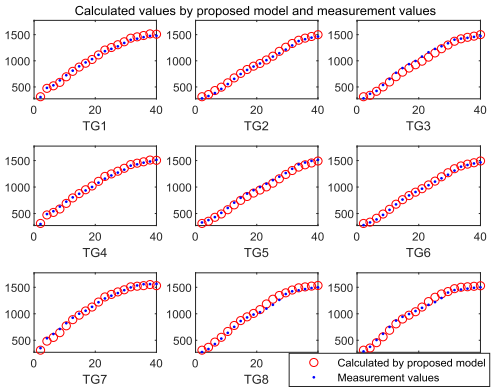


FIGURE 9. Calculated values by proposed model and measurement values of the trial slab.

furnace is 40.1 m, the width is 12.5 m, and the height is between 4.1 m and 7.4 m. The reheating furnace can be separated into 5 zones: recuperation zone, preheating zone, two heating zones, and soaking zone.

The trial steel slab has the dimension of 1.8 m∗0.25 m∗4.1m, and the heating time in the furnace is 186 min. A data acquisition unit and 11 thermocouples (Type K, ϕ = 15 mm) are equipped in the trial steel slab to record the information of slab temperature and the furnace temperature. The data acquisition unit is shown by the black package in Fig. 7. It is placed in a water cooled box and the box is wrapped with insulation material. In 11 thermocouples, 9 of them are assembled inside the trial steel slab at different locations to cover the areas of walking beam-skid, none-skid, and stationary beam-skid. The other 2 thermocouples (10 and 11), which are held by the 0.15m fixed link and assembled at the surface of the trial slab, are used to record the furnace temperature at the top and bottom surface of the slab. The locations of these thermocouples are given and shown in Fig. 7. For instance, three thermocouples (TG1, TG2,TG3) are located in the trial slab along the length direction ($z_{1,2,3}$ = 1.85 m) and each thermocouple is 100 mm from others in the width direction (x_1 = 0.8 m, x_2 = 0.9 m, x_3 = 1.0 m). And the locations for the thermocouples(TG1, TG2,TG3) along high direction of the slab are (y_1 = 0.01 m, y_2 = 0.12 m, y_3 = 0.24 m). Thus, the locations of the thermocouples(TG1, TG2,TG3) can be represented by the spatial coordinates (x, y, z): (0.8, 0.01, 1.85) , (0.9, 0.12, 1.85) , (1.0, 0.24, 1.85). Besides, the temperature information is recorded at every 20 s and saved in the recorder, which will be exported when the trial slab is discharged. As shown in Fig. 8, the measured temperatures of trial slab by the thermocouples (TG1-TG9) and the corresponding furnace temperatures at the top and bottom of the trial slab by 2 thermocouples (TG10, TG11) are all listed in this figure.

Here, we compare the programming effort and resulting performance of the CUDA program executed by using Visual Studio 2015 and CUDA 8.0. For the proposed 3D mathematical model, the spatial mesh systems for the trial slab in the space domain is chosen as $(N_i * N_j * N_k) = (180 * 25 * 410)$, which means $\Delta x = \Delta y = \Delta z = 0.01$. Based on the stability constraint equation (17), the time step Δt must be less than 1.4525. Thus, $\Delta t = 1$ and $N_n = 186*60/\Delta t = 11160$ can be obtained. Besides, the steel type of the trial slab is 20MnSi, whose temperature-dependent thermophysical properties can be obtained in Section II-B.

TABLE 2. Mean difference values mean $|T_c - T_m|$ between the calculated and measurement values of 9 thermocouples.

Thermocouple	20MnSi (K)	Constant (K)
1	7.27	50.22
2	22.92	**110.53**
3	38.11	34.30
4	7.51	50.89
5	37.93	52.73
6	14.07	97.99
7	17.44	**16.86**
8	**39.78**	107.50
9	**6.22**	61.76

For the purpose of comparison, the slab temperatures at the same locations measured by the thermocouples (TG1-TG9) and calculated by the proposed 3D model are plotted in Fig. 9. In this figure, the calculated values by proposed model are shown in the red circle, and the blue dots are the measurement values obtained by the thermocouple experiment. Considered that there exists measurement error in the measured values, which may be caused by the measurement methods and

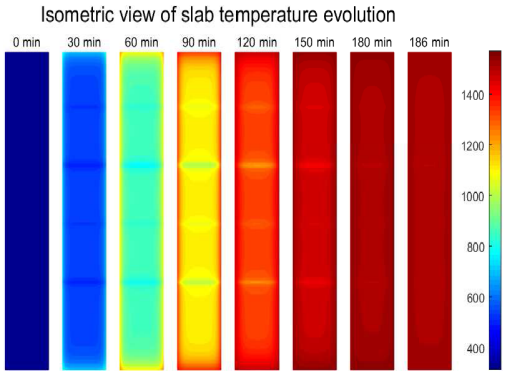


FIGURE 10. Isometric view of slab temperature evolution.

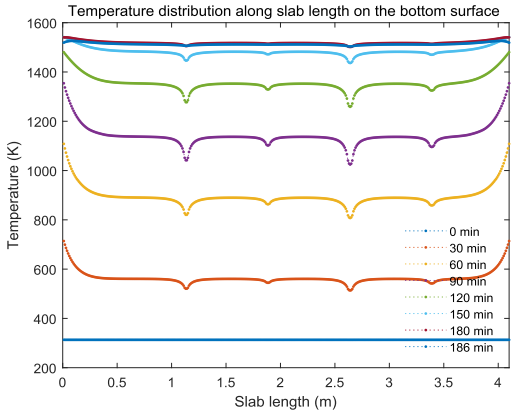


FIGURE 11. Temperature distribution along the slab length on the bottom surface.

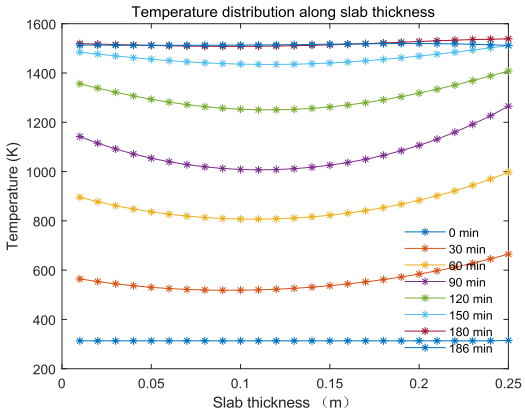


FIGURE 12. Temperature distribution along slab thickness.

environment in the furnace. It is acceptable that the predictions by the proposed model are in fair agreement with those measured by the thermocouples embedded in the trial slab.

To further compare the proposed 3D model with the thermocouple experiment results, the comparison between using 20MnSi temperature-dependent material properties and using the constant value for the specific heat c and the thermal conductivity λ are given in Table 2. Both have the same spatial mesh system, in which the number of mesh elements is 1.84 million. Here, the absolute mean difference values (mean $|T_c - T_m|$) of the 9 thermocouples between the calculated and measurement values are obtained and compared. As shown in Table 2, the minimum and maximum absolute mean difference value by using 20MnSi are 6.22 K and 39.78 K. Instead of using the constant value, both will become larger, which are 16.86 K and 110.53 K. Although, the computational time by using the constant value can be reduced from 14.30 seconds to 4.93 seconds, there will be an unacceptable decrease in accuracy of the proposed 3D mathematical model. It is not worth the cost. Therefore, it is very necessary for us to introduce the temperature-dependent material properties of steel slabs to the proposed model. Finally, it is evident that the proposed 3D model developed in the present work performs reasonably well for the slab heating process in a walking beam reheating furnace.

B. TEMPERATURE DISTRIBUTION INSIDE THE SLAB

For the reheating furnace, it is important to obtain the temperature distribution inside the slab. With the simulation by applying the proposed 3D mathematical model, the detailed temperature distribution of the slab in the reheating furnace can be demonstrated.

Firstly, the detailed temperature distribution on the slab surface or inside the slab during the entire reheating process can be clearly revealed. As shown in Fig. 10, the temperature evolution on the bottom slab surface during 186 min is demonstrated. When the slab moves from the charging door to the discharging door, the temperature field from bottom view of the slab is given to show the temperature evolution of the slab. At the beginning of the reheating process (t = 0 min), a cold slab, whose charging temperature is the room temperature 298 K, is charged into the reheating furnace. As shown by the temperature maps in Fig. 10, the skid marks on the slab, which is caused by the high temperature difference among the areas of walking beam skid region, stationary beam skid region, and non-skid region, can be clearly seen. And at the end of the reheating process (t = 186 min), the skid marks of walking beam skid region are almost disappeared.

Secondly, the effect of skid on slab surface temperature is further demonstrated in Fig. 11. In this figure, the temperature distribution on the slab bottom surface along the slab length is plotted to analyze the slab's bottom surface temperature. The temperature distribution on the slab bottom surface indicates that the skid marks have relatively lower temperature. The temperature difference between the areas with and without skid marks remain high until the slab reaches the soaking zone(t = 150 min). This phenomenon indicates that proper soaking time is required to reduce the skid mark effect.

Thirdly, another important temperature distribution is the temperature along the slab thickness, because a low core temperature at the slab mid-thickness will lead to high mill loads during hot rolling [24]. The temperature profiles along the slab thickness at different times(0, 30, 60, 90, 120, 150, 180, and 186 min) are plotted and shown in Fig. 12.

TABLE 3. Mesh parameters and results of comparison.

Number of mesh elements	Characteristic element size	Size of Δt (seconds)	Error (K)	Multiple of GPU-acceleration
$2.69 * 10^4$	0.06*0.02*0.06	14	78.86	104.09
$3.83 * 10^4$	0.05*0.02*0.05	13	63.96	135.65
$6.02 * 10^4$	0.04*0.02*0.04	11	58.04	150.59
$1.06 * 10^5$	0.03*0.02*0.03	9	35.38	308.24
$2.39 * 10^5$	0.02*0.02*0.02	5	33.91	453.74
$4.61 * 10^5$	0.02*0.01*0.02	2	30.72	706.53
$1.84 * 10^6$	0.01*0.01*0.01	1	21.25	1338.40
$3.69 * 10^6$	0.01*0.005*0.01	0.7	20.69	1110.49

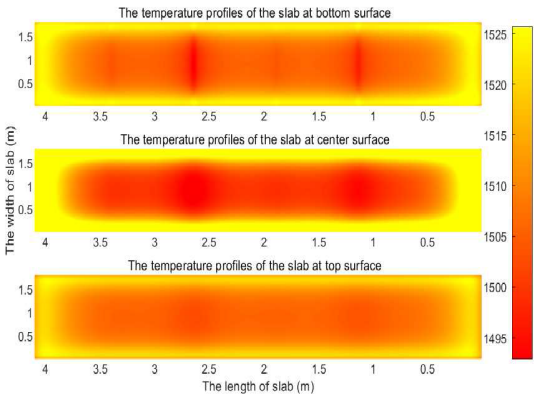


FIGURE 13. The discharging temperature profiles of the slab at top, center and bottom surface.

At the beginning, the slab is charged at the room temperature 298 K. The temperature difference between the slab center and the top (or bottom) surfaces first increased (before t = 90 min) and then decreased (after t = 90 min). It is clear that the slab surface temperature is higher than the temperature inside the slab during the heating process. At the end of the heating process, the temperature distribution becomes uniform throughout the slab thickness. So the discharged slab can meet the requirements of the following rolling mills.

Finally, one of the most important temperature distribution is the temperature distribution at the discharging time. In order to show the slab's temperature distribution at the discharging time, the slab temperature profiles at top, center, and bottom surface are plotted, as shown in Fig. 13. From this figure, it is clear that the difference between the max and min temperature value is 32.8 K, which is acceptable for the discharged slab in the temperature distribution.

C. VALIDATION OF THE IMPLEMENTATIONS ON GPU AND CPU ARCHITECTURES

The proposed computational model will be implemented on GPU and CPU architectures to obtain the comparison of the computational performance between the non-parallel CPU-implementation model and the highly parallel GPU-implementation model.

The CPU-implementation model does not include parallel processing and all the computations are implemented in nested loops. It is performed with the use of the computer with the Intel Core i5-4210H @ 2.9 GHz CPU and 6 GB RAM memory. On the contrary, the GPU-implementation model is applied in the NVIDIA Tesla P100 GPU. The simulation of the heating process lasting 186 min in the reheating furnace is used as the test case for benchmarks.

A series of simulation tests is carried out to assess the computational performance of the GPU-implementation model. Generally, eight types of computational mesh in space domain are considered for benchmarking: meshes with 26.9 thousand, 38.3 thousand, 60.2 thousand, 106 thousand, 239 thousand, 461 thousand, 1.84 million and 3.69 million mesh elements. The characteristic size of the mesh elements and the corresponding sizes of the time step are presented in Table 3 for each type of mesh. Because the explicit time discretization is used in the model formulation, the maximum size of the time step varies with the number of mesh elements according to (17). For instance, the maximum size of the time step Δt should be less than 9.22 seconds for the coarsest mesh with 106 thousand elements. Thus, the size of the time step for this computational mesh is selected as 9 seconds. From Table 3, the corresponding size of the time step decreases, when the number of mesh elements (space domain) in the mesh increases. Here, the multiple of the GPU-acceleration is defined as the ratio between the computational time of the CPU-implementation model and of the GPU-implementation model [18]. Then, the multiple of the GPU-acceleration therefore can quantify how many times the presented GPU-implementation model is faster than the common CPU-implementation model. Besides, in Table 3, the quantity "Error" is computed as the mean absolute difference between the calculated numerical solutions and the reference measured data. As the mesh elements increased, the error becomes smaller. For the coarsest mesh with 26.9 thousand elements, the error is 78.86 K, and it gradually decreases to 20.69 K for the compact grid with 3.69 million mesh elements.

To enhance the comparison of the computational performance between the CPU-implementation and the GPU-implementation, the absolute computational time for the simulations is presented in Fig. 14. The size of time step listed in Table 3 is used for both the CPU-implementation and GPU-implementation models and for the corresponding mesh. As shown in Fig. 14, which is plotted in the logarithmic scale, the GPU-implementation model provides a great acceleration of the model computation. In case of the coarse mesh with 106 thousand elements, the CPU-implementation

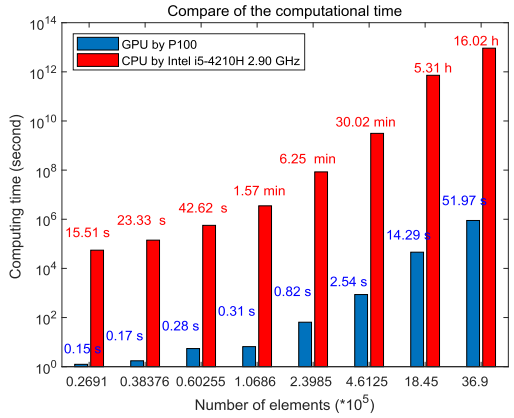


FIGURE 14. GPU vs CPU benchmarking: absolute computational time.

TABLE 4. The effect of CUDA cores on the GPU acceleration factor.

Number of mesh elements	GTX960 acceleration	P100 acceleration	The speed up ratio
$2.69 * 10^4$	30.41	104.09	3.42
$3.83 * 10^4$	36.01	135.65	3.77
$6.02 * 10^4$	44.58	150.59	3.38
$1.06 * 10^5$	66.19	308.24	4.66
$2.39 * 10^5$	70.52	453.74	6.43
$4.61 * 10^5$	79.65	706.53	8.87
$1.84 * 10^6$	108.24	1338.40	12.37
$3.69 * 10^6$	93.09	1110.49	11.93

model requires about 1.57 minutes to complete the simulation. The GPU-implementation model, however, completes the simulation in less than 0.31 seconds. The multiple of the GPU-acceleration is 308.24. Notes, the increase of the number of parallel threads therefore reduces the latency per one thread. Thus, similar conclusions can be obtained when the number of elements increases. For the compact computational mesh having 3.69 million elements, the use of the GPU-implementation model reduces the computational time from more than 16.02 hours to only 51.97 seconds. This feature is also reflected in the multiple of the GPU-acceleration presented in Table 3. The multiple of acceleration is 104.09 for the coarsest mesh with 26.9 thousand mesh elements, and it gradually increases to almost 1338.40 for the compact grid with 1.84 million mesh elements. However, there is a sudden jump down from 1338.40 to 1110.49. This may be influenced by the number of CUDA cores, memory bandwidth, latency, inter-thread communication, and so on, which will be studied in the following section.

D. THE BOTTLENECKS ON THE GPU ACCELERATION FACTOR

A bottleneck can also be called a choke-point. It's a component in the system that is too slow to allow the other components to work at their fullest potential. As shown in Table 3, the acceleration factor meets the bottleneck at the grid with 3.69 million mesh elements. To find the bottlenecks for the GPU acceleration factor, the following simulation are given.

Firstly, the effect of CUDA cores on the GPU acceleration factor is given. Two different GPUs: GTX960 and Tesla P100 are used to demonstrate the correlation between CUDA cores and GPU acceleration factor for the presented 3D model. The Tesla P100 GPU incorporates 3584 CUDA cores, and the Tesla GTX960 GPU incorporates 640 CUDA cores. The speed up ratio of cores by Tesla P100 to GTX960 is 5.6. Then, the acceleration factors by Tesla P100 and GTX960 are shown in Table 4, in which the speed up ratio of GPU acceleration factor by Tesla P100 to GTX960 is also given. The result of Table 4 includes two aspects: On the one hand, the acceleration factor of Tesla P100 ranges from 104 to 1338, whereas GTX960 has the acceleration factor of 30-108. Obviously, the acceleration factor is influenced by the number of cores. The more cores the GPU has, the bigger the acceleration factor is. On the other hand, with the increase of the mesh elements, the ratio of GPU acceleration factor becomes large from 3.42 to 12.37. However, when the computational demand is big enough, the speed up ratio also declines from 12.37 to 11.93. It can be concluded that the size of GPU acceleration factor is influenced by the CUDA cores, but it is not a bottleneck.

TABLE 5. The bottleneck of GPU acceleration performance.

Number of mesh elements	achieved _occupancy (%)	gld _throughput (GB/s)	gld _efficiency (%)
$2.69 * 10^4$	14.67	199.51	7.86
$3.83 * 10^4$	20.70	285.59	7.75
$6.02 * 10^4$	30.71	373.91	7.67
$1.06 * 10^5$	52.38	473.04	7.42
$2.39 * 10^5$	63.03	597.02	7.36
$4.61 * 10^5$	62.54	854.84	6.92
$1.84 * 10^6$	70.35	1149.60	6.46
$3.69 * 10^6$	52.40	937.38	6.29

Secondly, the common bottlenecks of GPU performance (memory bandwidth, latency) are studied and analyzed. Here, the nvprof command is used for the performance testing. Three possible factors are introduced and verified by using the achieved_occupancy, gld_throughput, and gld_efficiency metrics. Using the nvprof achieved_occupancy metric, we can obtain the ratio of the average active warps per active cycle to the maximum number of warps supported on a multiprocessor. As shown in Table 5, with the increase of the mesh elements, the ratio of active warps increases from 14.67 % to 70.35%. But it decreases from 70.35% to 52.40%, when the grid mesh is chosen with 3.69 million mesh elements. Similarly, the global memory load throughput can be obtained by using the nvprof achieved_occupancy metric. And, the global memory load throughput increases from 199.51 GB/s to 1149.60 GB/s, then it decreases from 1149.60 GB/s to 937.38 GB/s. However, the ratio of requested global memory load throughput to required global memory load throughput, which is obtained by the gld_efficiency, is monotonically decreasing. There is no turning point. It seems that the ratio of global memory load throughput requested to required may be inconsequential to performance. Thus, it can be concluded that the ratio of the average active warps and global memory load throughput may lead to the bottlenecks of the GPU acceleration factor.

Finally, since the acceleration factor is the ratio between the computational time of the CPU-implementation and the computational time of the GPU-implementation. The bottlenecks may be caused by other different factors: the parallelism of the programming arithmetic, CPU hardware performance, and so on. This remains a future of work.

V. CONCLUSIONS

In this paper, a rapid GPU-implementation 3D mathematical model for the walking-beam type reheating furnace has been developed. The model implements the parallel code in CUDA/C++ by the NVIDIA Tesla P100. The following conclusions can be drawn. First, the proposed 3D mathematical model is verified and validated with the industry measurements. It is proved that the proposed model is capable of predicting the temperature distribution inside a slab. Second, the detailed temperature distribution inside the slab can be demonstrated by the proposed 3D model. The detailed descriptions include the temperature evolution during the entire reheating process, the effect of skid on slab surface temperature, the temperature distribution along slab thickness and the temperature distribution at the discharging time. Third, the computational time by the developed GPU–implementation model declines from hours to seconds. The multiple of the GPU-acceleration is between 104.09 and 1338.40 in comparison to the CPU-implementation model usually used in practice. In summary, the proposed model can provide not only reasonable predictions but also rapid computation capabilities required for the real-time furnace control and optimization.

REFERENCES

[1] V. K. Singh and P. Talukdar, "Comparisons of different heat transfer models of a walking beam type reheat furnace," *Int. Commun. Heat Mass Transf.*, vol. 47, pp. 20–26, Oct. 2013.

[2] A. Steinböck, *Model-Based Control and Optimization of a Continuous Slab Reheating Furnace*, vol. 12. Aachen, Germany: Shaker, 2011.

[3] U. Leifgen, S. Ganesaratnam, and L. Croce, "A new concept for the control of reheating furnaces for slabs," in *Proc. 1st Int. Conf. Energy Efficiency CO_2 Reduction Steel Ind., (EECR STEEL)*, Düsseldorf, Germany, vol. 2, 2011.

[4] D. F. Staalman and A. Kusters, "On-line slab temperature calculation and control," *Manuf. Sci. Eng.*, vol. 4, pp. 307–314, 1996.

[5] D. R. Kreuzer and A. Werner, "Implementation of models for reheating processes in industrial furnaces," in *Proc. 8th Int. Modelica Conf.*, Dresden, Germany: Linköping University Electronic Press, 2011, pp. 376–387.

[6] A. Steinboeck, D. Wild, T. Kiefer, and A. Kugi, "A mathematical model of a slab reheating furnace with radiative heat transfer and non-participating gaseous media," *Int. J. Heat Mass Transf.*, vol. 53, nos. 25–26, pp. 5933–5946, 2010.

[7] Z. Yang and X. Luo, "Optimal set values of zone modeling in the simulation of a walking beam type reheating furnace on the steady-state operating regime," *Appl. Therm. Eng.*, vol. 101, pp. 191–201, May 2016.

[8] H. Ezure, Y. Seki, N. Yamaguchi, and H. Shinonaga, "Development of a simulator to calculate an optimal slab heating pattern for reheat furnaces," *Elect. Eng. Jpn.*, vol. 120, no. 3, pp. 42–53, 1997.

[9] Y. Li, G. Wang, and H. Chen, "Simultaneously estimation for surface heat fluxes of steel slab in a reheating furnace based on DMC predictive control," *Appl. Therm. Eng.*, vol. 80, pp. 396–403, Apr. 2015.

[10] X. Luo and Z. Yang, "Dual strategy for 2-dimensional PDE optimal control problem in the reheating furnace," *Optim. Control Appl. Methods*, vol. 39, no. 2, pp. 981–996, 2018.

[11] M. Y. Kim, "A heat transfer model for the analysis of transient heating of the slab in a direct-fired walking beam type reheating furnace," *Int. J. Heat Mass Transf.*, vol. 50, no. 19, pp. 3740–3748, 2007.

[12] J. G. Kim, K. Y. Huh, and I. T. Kim, "Three-dimensional analysis of the walking-beam-type slab reheating furnace in hot strip mills," *Numer. Heat Transf. A, Appl.*, vol. 38, no. 6, pp. 589–609, 2000.

[13] C.-T. Hsieh, M.-J. Huang, S.-T. Lee, and C.-H. Wang, "Numerical modeling of a walking-beam-type slab reheating furnace," *Numer. Heat Transf. A, Appl.*, vol. 53, no. 9, pp. 966–981, 2008.

[14] A. Jaklič, T. Kolenko, and B. Zupančič, "The influence of the space between the billets on the productivity of a continuous walking-beam furnace," *Appl. Therm. Eng.*, vol. 25, nos. 5–6, pp. 783–795, 2005.

[15] S. K. Dubey, N. Agarwal, and P. Srinivasan, "Three dimensional transient heat transfer model for steel billet heating in reheat furnace," in *Proc. ASME Summer Heat Transf. Conf.*, 2012, pp. 963–967.

[16] H. Chu, H. Lee, and D. Sheen, "Parallel ADI method for parabolic problems on GP-GPU," in *Proc. 8th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci.*, 2009, pp. 1–4.

[17] P. Micikevicius, "3D finite difference computation on GPUS using CUDA," in *Proc. 2nd Workshop Gen. Purpose Process. Graph. Process. Units*, 2009, pp. 79–84.

[18] L. Klimeš and J. Štětina, "A rapid gpu-based heat transfer and solidification model for dynamic computer simulations of continuous steel casting," *J. Mater. Process. Technol.*, vol. 226, pp. 1–14, Dec. 2015.

[19] W. L. Cheng, A. Sheharyar, R. Sadr, and O. Bouhali, "Application of GPU processing for Brownian particle simulation," *Comput. Phys. Commun.*, vol. 186, pp. 39–47, Jan. 2015.

[20] R. B. Bird, W. E. Stewart, and E. N. Lightfoot, *Transport Phenomena*. Hoboken, NJ, USA: Wiley, 2007.

[21] G. W. Recktenwald, "Finite-difference approximations to the heat equation," *Mech. Eng.*, vol. 10, pp. 1–27, Jan. 2004.

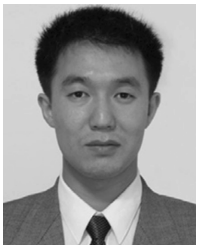
[22] N. Ozisik, *Finite Difference Methods in Heat Transfer*. Boca Raton, FL, USA: CRC Press, 1994.

[23] N. Wilt, *The CUDA Handbook: A Comprehensive Guide to GPU Programming*. London, U.K.: Pearson Education, 2013.

[24] G. Tang, B. Wu, D. Bai, Y. Wang, R. Bodnar, and C. Zhou, "CFD modeling and validation of a dynamic slab heating process in an industrial walking beam reheating furnace," *Appl. Therm. Eng.*, vol. 132, pp. 779–789, Mar. 2018.

ZHI YANG received the B.S. degree from Shandong University, Jinan, China, in 2009, and the M.S. degree in detection technology and automatic equipment from the Shenyang University of Chemical Technology, Shenyang, China, in 2013. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University.

His current research interests include modeling and optimization for the complex industrial systems, parallel programming and applications, and intelligent optimization methods.



XIAOCHUAN LUO (M'09) received the M.Eng. and Ph.D. degrees from the Harbin Institute of Technology, Harbin, China, in 1999 and 2002, respectively.

From 2002 to 2004, he held a postdoctoral position with the University of Technology of Troyes, Troyes, France. He is currently an Associate Professor with Northeastern University, Shenyang, China. His current research interests include modeling and optimization of processing industry manufacturing systems, production planning and scheduling, and optimization algorithms.

Dr. Luo is currently a Peer Review Expert of the National Natural Science Foundation of China, the IEEE Reviewer, IJPR, EJIE, and other international journal reviewers. He was selected into the 2008 Ministry of Education's New Century Excellent Talents Support Program, and was also selected into the "Million Talents Project" in Liaoning Province, in 2009.