Research on the temperature field model of plate hot rolling process based on CUDA

Wenhong Liu a, Siying Guo b, Siyu Zhang c

College of Information Science and Engineering Northeastern University Shenyang, China.

a liuwenhong@ise.neu.edu.cn, b 18624153317@163.com, c zhangsiyu_0301@163.com

Abstract. The current temperature field model of plate rolling process is generally based on CPU programming, due to the influence of computer performance, the accuracy and speed of the model can not be improved at the same time. To solve this problem, this paper studies the temperature field model with particle swarm optimization to modify the heat transfer coefficients, and the parallel program of this model is written under CUDA architecture, on premise of ensuring the accuracy of the model, the computation speed of the model is improved. To get the best acceleration, this paper analyzes the factors affecting GPU performance, the accuracy and speed of the temperature field model before and after modification are compared under the best acceleration configuration. Final temperature field model calculation error reduced to an average of 15.57 ℃, better than the model based on the empirical formula, the computing speed is also improved, which is 1.78 times that of CPU.

Keywords. Temperature field model, Parallel computing, CUDA architecture, particle swarm optimization algorithm

1. Introduction
In the rolling process, heat transfer coefficients are the main factors affecting the calculation accuracy of the temperature field, but they often change under the influence of complex environment. Particle swarm optimization (PSO) algorithm can find the optimal solution to make the equation more accurate through iteration. It can be used to correct heat transfer coefficients because of its simple operation, fast convergence and its applicability to the processing of real value. But the PSO algorithm of multiple iterations will increase the running time of the program, so this paper writes a parallel program under the CUDA architecture to improve the speed of the model.

CUDA is a general computing architecture based on the new parallel programming model and instruction set architecture, it can use GPU parallel computing engine to solve many complex computing tasks and is more efficient than CPU [1]. The platform architecture is shown in Fig. 1.
Due to the different construction of CPU and GPU, they have their own characteristics: GPU has no cache and uses more transistors for computing, mainly for speed, so it is suitable for problems with large amounts of computing. CPU adopts caching and complex branch prediction technology, so it is more suitable for solving complex logic problems. CUDA combines their features, greatly improving the efficiency of the program.

In this paper, the temperature field model is established by finite volume method, and PSO algorithm is used to correct heat transfer coefficients. The parallel program of the model is written by taking advantage of GPU high-speed computing. The effects of data size and GPU thread configuration on acceleration are analyzed. Finally, the accuracy and speed of the model before and after correction are compared.

2. Basic Principles

2.1. differential equation for heat conduction

Since the discrete equation derived by finite volume method has the characteristics of good adaptability to the regional shape, it is suitable for establishing temperature field of the plate which deforms in rolling area. Therefore, finite volume method is adopted to solve the temperature field in this paper. The basic idea is to divide the computational region into grids and make each grid point have a non-repeating control volume, then a set of discrete equations is obtained by integrating the differential equation over each control volume.

Considering that the temperature gradient in the thickness and width direction of the plate is much greater than that in the length direction, the temperature field of the plate can be simplified into a two-dimensional unsteady model. According to the energy conservation law, the two-dimensional heat conductivity differential equation of the rolling part in the rectangular coordinate system is:

\[
\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left( \lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( \lambda \frac{\partial T}{\partial y} \right) + S
\]

(1)

Where, \( x \) is the coordinate in the width direction of the plate, \( y \) is the coordinate in the thickness direction, and \( S \) is the internal heat source.

Fig. 2 shows a part of the grid system for the two-dimensional heat conduction problem of plate, and the shaded area is the control volume of node \( P \).
According to the basic idea of finite volume method, within the time interval \([t, t+\Delta t]\), integrate the heat conduction equation against the control volume of \(P\) as shown in Fig. 2, and the results are summarized as follows:

\[
a_p T_p = a_e T_e + a_w T_w + a_s T_s + a_n T_n + b
\]  

(2)

Where,

\[
a_e = \lambda_e \frac{\Delta y}{\Delta x}, \quad a_w = \lambda_w \frac{\Delta y}{\Delta x}, \quad a_s = \lambda_s \frac{\Delta x}{\Delta y}, \quad a_n = \lambda_n \frac{\Delta x}{\Delta y},
\]

\[
b = S_c \Delta x \Delta y + a_p T^0_p,
\]

\[
p = a_e + a_w + a_s + a_n + a_p^0 S_p \Delta x \Delta y,
\]

\[
a_p^0 = p c \Delta x \Delta y / \Delta t,
\]

\(S_c\) is the constant part of the internal source, \(S_p\) is the coefficient of \(T_p\) in the internal heat source.

During the rolling process, the surface of the plate will contact with the surrounding environment and generate heat transfer. In this paper, the heat flux exchange with the outside world on the plate boundary is equivalent to the internal heat source at the adjacent internal node, at the same time, the boundary is equivalent to adiabatic boundary. This not only ensures the energy conservation, but also eliminates the unknown boundary node temperature equation from the governing equation, and only solves the algebraic equation of the inner node. As shown in Fig. 3, the heat transfer at the boundary is expressed as [2][3]:

\[
S_c = \frac{1}{\Delta y} \frac{1}{h + (\delta y) / \lambda_s} T_f
\]  

(3)

\[
S_p = -\frac{1}{\Delta y} \frac{1}{h + (\delta y) / \lambda_s}
\]  

(4)

Where, \(h\) is the heat transfer coefficient, in the rolling process, the heat transfer coefficients are mainly involved in the air cooling stage and the rolling deformation stage.

In this paper, the heat transfer coefficient in the air cooling stage is determined according to Newton’s cooling formula, and the factors such as heat convection and contact with the roller table are all converted into the emissivity:

\[
h_a = \varepsilon \delta (T^2 + T_0^2)(T + T_0)
\]  

(5)

Where, \(\delta\) is Stefan-Boltzmann constant, \(\delta = 5.67 \times 10^{-8} \text{W/(m}^2\text{K}^4)\); \(T\) is the surface temperature of the plate; \(T_0\) is the temperature of the fluid near the surface of the plate; \(\varepsilon\) is emissivity, it is a constant between 0 and 1.

The temperature change in the rolling deformation stage is mainly caused by the heat of plastic deformation, friction heat and the heat lost by the contact roller, the contact heat transfer coefficient formula model selected in this paper is [4]:

\[
h_r = (a \cdot p + b) \cdot v^{0.5}
\]  

(6)

Where, \(p\) is average unit rolling pressure; \(v\) is the rolling speed; \(a, b\) is the model coefficient.
2.2. **CUDA architecture**

Under the CUDA architecture, programs can be executed on the CPU and GPU. CPU and its memory are called the host side, GPU and its memory are called the device side, the program running on the device side is called kernel function. The CUDA architecture model is shown in Fig. 4.

![CUDA architecture model](image)

When the kernel function is started on the host side, a large number of threads will be generated in the device. Each thread executes the statement specified by the kernel with its private data. The threads generated by the startup of a kernel function are divided into many groups, each group is called a thread block, all threads within the same thread block can cooperate with each other, and all thread blocks form a thread grid. The dimensions of thread grid and thread block can be divided into one dimension, two dimensions, three dimensions, CUDA provides the built-in variables `blockDim` and `gridDim` to represent the size of thread block and thread grid respectively, and uses the built-in variables `threadIdx` and `blockIdx` to index the position of the thread in the thread block and the position of the thread block in the thread grid[5].

CUDA organizes threads on GPU by defining the configuration of the kernel functions. The kernel function is defined using the `__global__` declarator, and a new `<<<Grid, Block>>>` configuration syntax to specify the number of threads in a kernel function call [6]. Where the `Grid` is the configuration of the thread blocks in the thread grid, and the `Block` is the configuration of the threads in each thread block. The maximum number of thread blocks that a kernel can start and the maximum number of threads per thread block is determined by the computer hardware.

CUDA uses a hierarchical memory model to manage various storage resources on the device, and CUDA makes use of their differences in access speed and storage space for classification management [7]. The characteristics of different memory regions are shown in the table 1:

| Memory type        | access mode | scope    | Features and application                                      |
|--------------------|-------------|----------|----------------------------------------------------------------|
| register           | Read/write  | thread   | Fast access, store small, frequently used variables           |
| Local memory       | Read/write  | Thread   | Slow access speed, storage register overflow data             |
| Shared memory      | Read/write  | block    | Memory used by threads within the same thread block to store variables common to threads within the block |
| Global memory      | Read/write  | grid     | Used for communication between host and device, suitable for storing large amount of data |
| Constant memory    | Read        | grid     | Stores read-only parameters that need frequent access, such as invariant coefficients in a formula |
| Texture memory     | Read        | grid     | Typically used for address mapping, data filtering, and caching |
3. Establish the Temperature Field Model

3.1. serial calculation of temperature field
During the rolling process, heat transfer coefficients are often affected by the scale, rolling pressure and impurities between the contact surfaces, heat transfer coefficients calculated by empirical formula are often quite different from the actual value. In order to improve the accuracy of the model, PSO is used to modify the heat transfer coefficients.

1. Particle swarm optimization
Suppose that in a target search space of \(D\)-dimensions, there are \(M\) particles forming a group, then the position of the \(i^{th}\) particle in the \(D\)-dimensional search space is expressed as \(X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})\). The group searches for the global optimal solution in the \(D\)-dimensional space, \(X_i\) can be substituted into the objective function to calculate its fitness value, so as to measure the quality of particles. The fitness function in this paper is:

\[
J(h) = \sum_{i=1}^{M} \left[ T_{m}^i - T_{c}^i \right]^2
\]  

(7)

Where, \(M\) is the sequence length of temperature value; \(T_{m}^i\) and \(T_{c}^i\) are the measured temperature values and the calculated temperature values of different measuring points.

The velocity of a particle is an important parameter used to adjust its own direction to ensure its flight to the optimal solution. The flight speed of the \(i^{th}\) particle is denoted as \(v_i\), and PSO algorithm updates the particle speed by the following formula:

\[
\begin{align*}
vid(t+1) &= wvid(t) + c_1 r_1 (pid - xid(t)) + c_2 r_2 (pgd - xid(t)) \\
xid(t+1) &= xid(t) + vid(t+1)
\end{align*}
\]  

(8)

\[
\begin{align*}
v_i^d(t) &= \begin{cases} 
    v_{\text{max}}^d, & v_i^d(t) > v_{\text{max}} \\
    -v_{\text{max}}^d, & v_i^d(t) < -v_{\text{max}}
\end{cases}
\end{align*}
\]  

(10)

Where, \(c_1, c_2\) are learning factors and non-negative acceleration constants, usually between \([0,2]\); \(r_1, r_2\) are random numbers uniformly distributed in \([0,1]\); \(t\) is the number of current iterations; \(w\) is inertia weight.

The inertia weight \(w\) represents the degree of retention of the particle's original velocity, if the value is too large, the global convergence ability is strong, but it is easy to fall into the local optimal value. In order to balance the convergence, \(w\) can be dynamically adjusted in the search process. The algorithm starts to assign a larger initial value to \(w\), so that the particle can first detect a better region in the global scope with a large speed. As the search is carried out, \(w\) gradually decreases, so that the particle can conduct a fine search around its pole value. The \(w\) expression adopted in this paper is as follows:

\[
w = w_{\text{max}} - (w_{\text{max}} - w_{\text{min}}) \cdot \frac{t}{T_{\text{max}}}
\]  

(11)

Where, \(T_{\text{max}}\) is the maximum evolutionary algebra; \(w_{\text{min}}\) is the minimum inertia weight, most take 0.4, \(w_{\text{max}}\) is the maximum inertia weight, most take 0.9.

2. Temperature field model based on standard particle swarm optimization
When the PSO algorithm is applied to the temperature field model, the position of each particle is a potential solution of heat transfer coefficients, in each iteration, the particle updates its state through the optimal solution found by the particle itself and the optimal solution found by the entire population.

The steps for establishing the temperature field model based on the standard PSO algorithm are as follows:
(1) Initialize the particle swarm, including swarm size $M$, randomly initialize the initial position and speed of the search point $x_i$, $v_i$; variable range of search space; learning factor $c_1$ and $c_2$; maximum particle speed $v_{max}$; the optimal position that each particle has experienced so far $g_{best}$, find the global extremum from the individual extremum, the best value of the particle is set to $z_{best}$.

(2) Calculate the fitness of each particle according to the formula, update the best position experienced by each particle $g_{best}$, update the best position experienced by the group $z_{best}$.

(3) Update the current particle's speed and position according to the above formula.

(4) Check to see if the end condition is met (In order to compare the computing time of GPU and CPU, this paper takes the iteration number of PSO algorithm as the ending condition), if the condition is satisfied, stop the operation, and output the optimal particle and the optimal value.

3.2. Parallel calculation of temperature field

In the process of solving the discrete equation, when solving the temperature value at a certain grid point, only the temperature values at the previous moment of this point and its four surrounding points can be used, and there is no sequential relation with the solution of other points. Therefore, the temperature values of all temperature points at a certain moment can be calculated in parallel. Suppose the temperature field has $nx$ grid points along the width direction of the plate and $ny$ grid points along the thickness direction of the plate, then the parallelism degree of a temperature field is $nx \times ny$.

In CUDA architecture, since the memory on the host and device side is independent of each other, so allocating memory for the host and the device side separately and passing data between the host and the device side must be done manually. Therefore, the specific steps of parallel operation of temperature field model combined with PSO algorithm are as follows:

(1) Initialize the temperature field on the host side, allocate enough memory for CPU and GPU respectively, and configure the thread specification of kernel function startup.

(2) Initialize the particle swarm.

(3) Copy the variables needed to calculate the temperature field and the heat transfer coefficients selected by each particle from the host side to the device side, and start the kernel function.

(4) Solve the temperature field equation in the kernel function. In this paper, the coefficients used in the calculation of each grid point are stored in the private registers of each thread, $threadIdx$ and $blockIdx$ are used to make each thread calculate the temperature data of a grid point with its own coefficients. After all threads have finished calculating, the fitness function value of the PSO algorithm is calculated and copied back to the host memory.

(5) Use the formula to update the state of each particle, and judge whether the condition of iteration stop is satisfied, if so, stop iteration; If not, go back to step 3 and recalculate the temperature field with updated particle values.

4. Results Analysis

4.1. Results analysis of serial temperature field model

The calculation results of the temperature field model before and after the correction of heat transfer coefficients by PSO algorithm are as follows:

| Table 2. Comparison of model operation results before and after coefficients correction |
|----------------------------------|----------------|----------------|----------------|
|                                | corrected model | Uncorrected model |
| number of iterations           | 10             | 50             | 100            |
| average error                  | 26.14          | 21.60          | 14.76          |
| Time (s)                       | 49.65          | 282.71         | 583.03         | 29.07         |
As can be seen from the above table, the accuracy of the model modified by PSO algorithm is higher than that of the original model, and as the number of model iterations increases, the average error of the model becomes smaller and smaller. When the number of iterations of PSO is 100, the average error of the model is within 15℃, and the time it takes to run the program is also increasing exponentially. It can be seen that the model speed tends to slow down while improving the accuracy of the model.

4.2. Effects of different grid sizes on parallel computing performance

In this paper, when calculating the discrete equation, thread blocks are organized in a two-dimensional manner. There are (32×32) threads in each thread block, and each thread is assigned to calculate the value of a grid point in the equation. In this paper, the influences of different grid numbers (the amount of data to be computed) on the GPU operation speed are compared. When the discrete equation is solved through 40 iterations, the results are shown in the following table:

| amount of data | CPU computation time(s) | GPU computation time(s) | Parallel acceleration ratio |
|---------------|-------------------------|-------------------------|-----------------------------|
| 64×64         | 0.08                    | 0.13                    | 0.73                        |
| 128×128       | 0.65                    | 0.35                    | 1.86                        |
| 256×256       | 1.81                    | 0.95                    | 1.93                        |
| 320×320       | 2.74                    | 1.39                    | 1.98                        |

The formula for calculating the acceleration ratio is:

$$S_p = \frac{T_1}{T_p}$$

(12)

Where, $T_1$ is the run time of a single processor, $T_p$ is the run time after GPU acceleration.

As can be seen from the above table, in a certain range, the acceleration ratio increases with the increase of operational data. The data to be used in the device side calculation is serially copied to the kernel function by the host side, therefore, when the amount of data is small, the speed of GPU calculation cannot make up for the lost time of data replication, this causes the program to run more slowly under the CUDA architecture. When the amount of data is large, the time saved by GPU calculation can hide the time of data replication, so the acceleration ratio will be higher.

As can be seen from the Fig. 5, with the increase of data amount, the increase of CPU running time is much faster than that of GPU. Therefore, within a certain data amount, with the increase of data size, the acceleration ratio of parallel computing will gradually increase.

4.3. Effects of different thread configurations on parallel computing performance

In this paper, the PSO algorithm is used for 100 iterations, the population size of each iteration is 48, and the grid of finite volume method is 52×62. For the model of air cooling stage, different number of
threads are enabled to test the running speed of the GPU during the rolling process. For the convenience of programming, the size of thread block in this paper is set as (26 × 32). The more thread blocks are started, the more temperature fields will be calculated at the same time. Assuming the number of temperature fields to be calculated at the same time is \( n \), the degree of parallelism is \( n \times nx \times ny \), and the process is shown as follows:

![Diagram showing the process]

**Fig. 6** Calculates \( n \) temperature fields simultaneously

When thread blocks of different sizes are started, the running results are as follows:

**Table 4.** The effect of different thread blocks on the result of calculation in the air cooling stage

| Number of thread blocks started | (4,2)  | (8,2)  | (12,2) | (16,2) |
|--------------------------------|--------|--------|--------|--------|
| running speed (s)              | 289.17 | 271.21 | 192.72 | 245.19 |
| Calculation error              | 14.26  | 16.15  | 15.57  | 13.98  |

As can be seen from the above table, the fastest running is the (12,2) specification thread block. Due to the limited resources in the GPU, when the thread block size continues to increase, it will limit the number of threads to be started at the same time, this reduces the parallelism of the computation and hence the running speed.

4.4. Analysis of the calculation results of temperature field model

In this paper, when the thread block size is (12,2), the acceleration ratio and running calculation error of the overall temperature field model are compared with the results of CPU running. The results are shown in the following table:

**Table 5.** Calculation results of temperature field model

|                     | Mean error | Program run time(s) | Parallel acceleration ratio |
|---------------------|------------|---------------------|-----------------------------|
| Serial calculation  | 14.76      | 583.03              |                             |
| Parallel computing  | 15.57      | 327.54              | 1.78                        |

As shown in table 5, the error of parallel computing is not much different from that of serial computing, but the speed is much faster than serial computing. The reason is: parallel program reduces a lot of cycles in the serial program, so that a lot of data can be processed at the same time, saving a lot of time.
5. Conclusion
1. In this paper, based on the energy conservation law, the two-dimensional differential equation of heat conductivity in the rolling process is listed, and the temperature field model in rolling process is established by discretization of differential equation with finite volume method.

2. In order to improve the accuracy of the temperature field model, this paper established the inverse heat conductivity problem model based on PSO, at the same time, the running speed and calculating precision of the inverse problem model with different iteration times are compared, the results show that with the iterations of the algorithm increasing, the accuracy of the model will be higher but the speed will be slower.

3. In this paper, a parallel program of temperature field model modified by PSO is written. The effects of different data volumes and thread configurations on parallel acceleration are analyzed, and compared with the accuracy and speed of the temperature field model calculated by CPU serial. In the end, the parallel error of the temperature field model is 15.57 °C, speed ratio is 1.78, the speed and precision of the model are both improved.

Acknowledgments
This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0304100 and in part by the National Natural Science Foundation of China under Grant 61703084.

References
[1] Shuming Miao. Implementation of NS equations in parallel on GPU[D]. Shanghai: Shanghai jiaotong university, 2011.
[2] Tao W Q. Recent advances in computational heat transfer[M]. Beijing: Science press, 2000, 238-241.
[3] Tao W Q. Numerical heat transfer[M]. Xi’an: Xi’an jiaotong university press, 1988, 84-85.
[4] Zhi Huiqiang. Application of neural network and genetic algorithm in inverse heat conduction problem[J]. Journal of hebei university, 2004, 33(2): 171-176.
[5] Mingqiu Luo, Chenhong Zhu, Ning Dong, Yequan Chen. Optimized Parallelization for Full Waveform Inversion[J]. Journal of Computational Physics, 2016, 4: 233-240.
[6] Liu X Y. Research on Solidification Model Coupled with Heat and Fluid Flow for Continuous Casting Process of Steel on CUDA [D]. Shenyang: Northeastern University, 2016.
[7] Wang Zhuowei, Cheng Lianglun, Zhao Wuqing. Parallel computation performance analysis model based on GPU[J]. Computer science, 2014, 01: 31-38.
[8] Zhu Gang, Yang Yan. Parallel reconstruction and realization of 3D displacement field in digital holography using CUDA[J]. Journal of chongqing university of technology (natural science), 2018, 32(4): 111-118.
[9] Chen Y, Guo Q P. Application of parallel genetic algorithm in heat conduction inverse problem[J]. Computer and digital engineering, 2007, 35(3): 13-15, 199.
[10] Cai Yong, Li Guangyao, Wang Hu. Parallel computing of central difference explicit finite element based on GPU general computing platform[J]. Journal of computer research and development, 2013, 50(2): 412-419.
[11] Shaibu V B. Numerical analysis of work roll cooling in hot rolling process [D]. Odisha: National Institute of Technology Rourkela, 2012.
[12] Tang Guangbo, Liu Zhengdong, Kang Yonglin etc. Simulation of thermal evolution of strip and determination of heat transfer coefficient in deformation zone during hot rolling[J]. Iron and steel, 2006, 41(5): 36-40.