CHINA FOUNDRY Modeling of Casting and Solidification Processes – MCSP2017
Vol.14 No.5 September 2017
https://doi.org/10.1007/s41230-017-7172-1

GPU based numerical simulation of core shooting process

Yi-zhong Zhang[1], Gao-chun Lu[2], Chang-jiang Ni[1], *Tao Jing[1], Lin-long Yang[2], and Qin-fang Wu[2]

1. School of Materials Science and Engineering, Tsinghua University, Beijing 100084, China

2. Suzhou Mingzhi Technology Co., Ltd., Suzhou 215217, China

Abstract: Core shooting process is the most widely used technique to make sand cores and it plays an important role in the quality of sand cores. Although numerical simulation can hopefully optimize the core shooting process, research on numerical simulation of the core shooting process is very limited. Based on a two-fluid model (TFM) and a kinetic-friction constitutive correlation, a program for 3D numerical simulation of the core shooting process has been developed and achieved good agreements with in-situ experiments. To match the needs of engineering applications, a graphics processing unit (GPU) has also been used to improve the calculation efficiency. The parallel algorithm based on the Compute Unified Device Architecture (CUDA) platform can significantly decrease computing time by multi-threaded GPU. In this work, the program accelerated by CUDA parallelization method was developed and the accuracy of the calculations was ensured by comparing with in-situ experimental results photographed by a high-speed camera. The design and optimization of the parallel algorithm were discussed. The simulation result of a sand core test-piece indicated the improvement of the calculation efficiency by GPU. The developed program has also been validated by in-situ experiments with a transparent core-box, a high-speed camera, and a pressure measuring system. The computing time of the parallel program was reduced by nearly 95% while the simulation result was still quite consistent with experimental data. The GPU parallelization method can successfully solve the problem of low computational efficiency of the 3D sand shooting simulation program, and thus the developed GPU program is appropriate for engineering applications.

Key words: graphics processing unit (GPU); Compute Unified Device Architecture (CUDA); parallelization; core shooting process

CLC numbers: TP391.99 **Document code:** A **Article ID:** 1672-6421(2017)05-392-06

Cold box core-making is widely used in sand core manufacture, and the core shooting process in this core-making method plays a decisive role in the quality of sand cores [1]. In the core shooting process, core sand properties (density, size, shape, and binder ratio), tooling design (vents, core box, shooting head, and blow tubes) and technological parameters (pressure, time) will affect the filling sequence of sand flow and packing degree of core sand, and then decide flaws[2, 3]. Thus, systematic studies of the sand flow under the influence of these factors in the core shooting process will help in predicting flaws and improving the quality of sand cores. In recent years, research and development have been carried out to increasingly improve the quality of sand cores [4], but most efforts have gone to the development of a new binder system, and studies on the flow dynamics of the core shooting process are limited.

In-situ experiments are needed to unveil the mechanism of core shooting considering these complex influence factors and the limited studies on the flow dynamics of the core shooting process. Given that intrusive techniques will influence the behavior of sand flow, non-intrusive techniques are needed to synchronously investigate the flow behavior of core sand. High speed photography has been applied in the core shooting by a few researchers [5]. Wu et al. [6] and Winartmo et al. [7] applied a high-speed camera to photograph sand flow behavior, and their results provided qualitative validation. Systematic experiments were carried out by Ni et al. and practical conclusions were drawn based on quantitive results [2, 3].

Besides experimentation, a numerical simulation program based on the two-fluid model (TFM) with a kinetic-friction constitutive correlation was developed to study the flow dynamics of the core shooting process [8, 9].

*** Tao Jing**

Male, born in 1965, Professor. His research interests mainly focus on materials processing technology and integrated computational materials engineering (ICME). He has published more than 100 papers.

E-mail: jingtao@mail.tsinghua.edu.cn

Received: 2017-08-01; Accepted: 2017-08-15

The sand flow behaviors and the pressure distribution were well predicted, but the calculating efficiency was too low to match the needs in engineering.

A parallelization method is needed to accelerate the program. Parallel computing based on a graphics processing unit (GPU) has been applied in fields such as science computation, image rendering [10, 11] and signal processing [12], and great improvement of computing efficiency has been approved using GPU parallelization method. Compute Unified Device Architecture (CUDA) platform belonging to Nvidia Corporation provides an easy access to realize GPU parallel algorithm by using extended C programming language (called CUDA C)[13], and it popularized the GPU parallelization program into personal computers with a Nvidia GPU. The discrete method of the core-shooting numerical simulation program is finite difference. Krakiwsky et al. studied the finite difference time domain method with a GPU parallelization method, and good performance of GPU parallelization method has been proved in the improvement of computing efficiency and reliable simulation results [14, 15].

In the casting field, simulation studies based on GPU have achieved some breakthroughs in the solidification process[16,17], but the core shooting process is close to an isothermal process while the studies about the core shooting process are relatively limited. Thus, the parallelization of the core shooting program is still worthy of study. A parallel program was developed to simulate a relatively simple test-piece, and the algorithm was improved in the process. In the GPU parallelization method, data delay and model capability of parallelization have a great influence on the improvement of computing efficiency[10–15]. After optimizing the parallel algorithm, the computing time was greatly reduced, and the simulation result was compared with experimental data. Then, a combustor was simulated, and results were discussed.

1 Experiment and model

1.1 Experiment apparatus

In order to analyze the behavior of sand flow, experiments were carried out by Ni et al. [2, 3] using the experimental apparatus shown in Fig. 1. The behavior of sand flow was synchronously photographed with a computer-controlled high-speed camera (Mega speed MS55K). Pressure sensors (Keyence AP-C30C) and a data acquisition computer (PC) were used to collect the pressure variation in the shooting head, nozzle and core box.

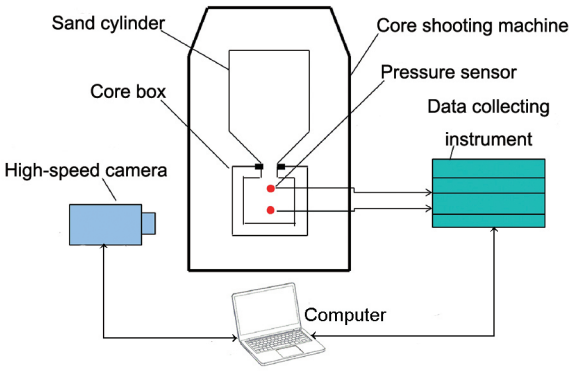

Fig. 1: Schematic of experimental apparatus

1.2 Model development

A two-fluid model (TFM) based on the kinetic theory of granular flow (KTFG) has been developed to simulate the core shooting process [9], and the TFM is based on the fundamental concept of interpenetrating continua for the gas-solid mixture. In fact, the sand flow can be divided into two different regimes: the viscous regime and the frictional regime by different sand volume

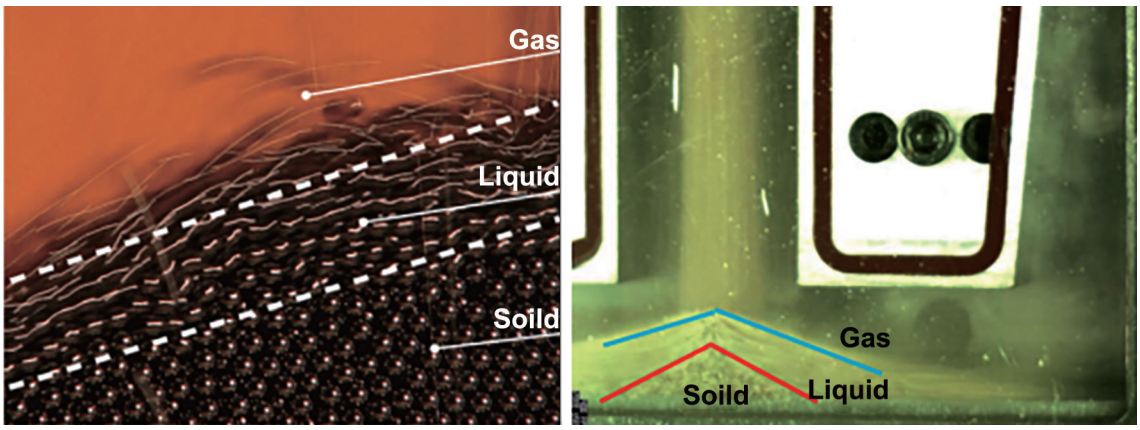

Fig. 2: Different sand phases in core shooting process

fractions (α_s), as shown in Fig. 2. The viscous regime includes the gas phase and a part of the liquid phase where sand volume fraction is low ($\alpha_s < 0.5$), and the frictional regime includes the remaining part of the liquid phase and the solid phase where sand volume fraction is high ($\alpha_s > 0.5$)[18]. KTFG is not suitable to describe the fractional action between sand particles in the friction regime, so a kinetic-frictional constitutive correlation was added considering frictional and kinetic stresses of the particle-particle interactions in different sand volume fractions.

As for the boundary condition, the gas phase applied a no slip condition and the sand phase applied the partial slip condition proposed by Johnson and Jackson [19].

The simulation program of core shooting was developed with FORTRAN and this program used a single CPU process.

A test piece (the problem scale is 36,000, and the real size is 150×150×25 mm³) was simulated. The comparisons of experimental photo sequences with simulated maps of sand volume fraction are shown in Fig. 3.

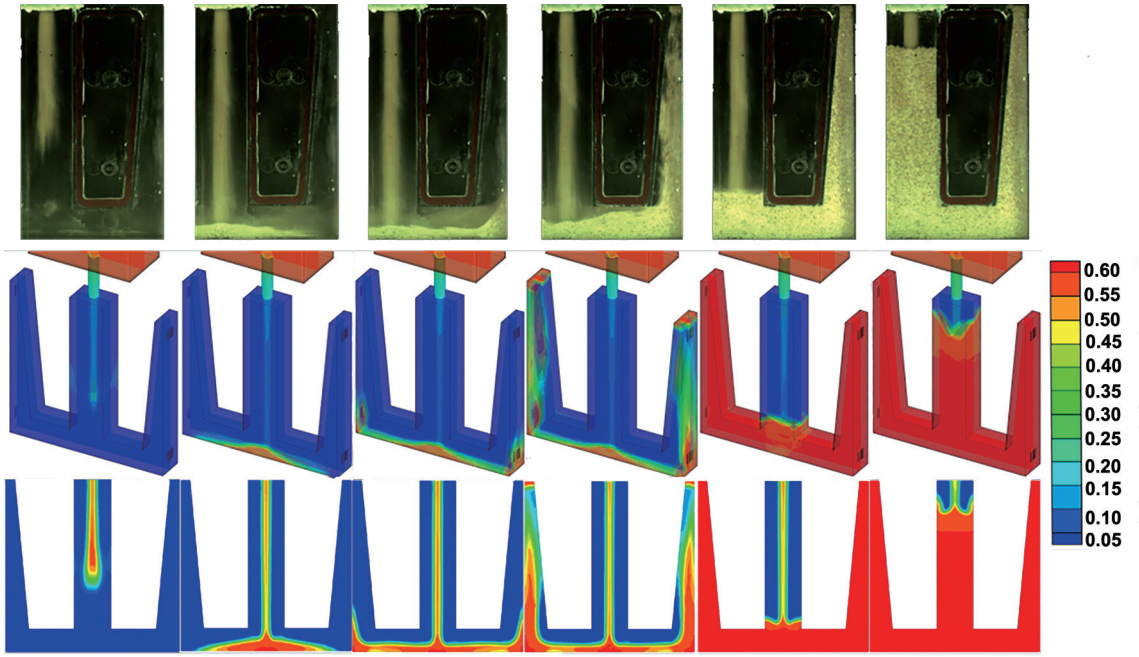

Fig. 3: Visualized photo sequences of test piece and predicted flow behavior of sand in core box

1.3 Results

Through some simulations with different parameters, it was found that simulation results with modeling parameters in Table 1 agree well with experimental photo sequences (some parameters can't be measured without suitable devices). The specularity coefficient of 0.002 refers to the free slip boundary condition for the sand phase, and simulation results with a restitution coefficient ranging from 0.3 to 0.9 maintain good consistency with experimental results.

The computing time of the test piece is 3.5 hours. As the core box of the test piece is symmetrical, only the right part of it is shown. It is obvious that the simulation results have a good agreement with the experimental results in some degree. For the test piece, sand particles first pile up at the bottom of the core box, and then start to pile up from the sidewalls to the entire core box, which is consistent with experimental results.

Table 1: Simulation results with modeling parameters

Property	Value
Mean sand diameter, d_s (mm)	0.24
Restitution coefficient, e	0.9
Sand density, ρ_s (kg·m^{-3})	2,650
Max sand volume fraction, α_{max}	0.63
Minimum sand volume fraction for transition to consider frictional stress, α_{min}	0.5
Angel of internal fraction, ϕ	0.3
Specularity coefficient, φ	0.002

2 GPU parallel acceleration

2.1 Parallel algorithm design

The governing equations for the conservation of mass and momentum for each separate phase are given as follows:

(1) Continuity equation

Gas phase:

$$\frac{\partial}{\partial t}(\alpha_g\rho_g) + \nabla\cdot(\alpha_g\rho_g V_g) = 0 \quad (1)$$

Solid phase:

$$\frac{\partial}{\partial t}(\alpha_s\rho_s) + \nabla\cdot(\alpha_s\rho_s V_s) = 0 \quad (2)$$

(2) Momentum equations

Gas phase:

$$\frac{\partial}{\partial t}(\alpha_g\rho_g V_g) + \nabla\cdot(\alpha_g\rho_g V_g V_g) = \alpha_g\nabla\cdot\tau_g - \beta(V_g - V_s) - \alpha_g\nabla P_g + \alpha_s\rho_s g \quad (3)$$

Solid phase:

$$\frac{\partial}{\partial t}(\alpha_s\rho_s V_s) + \nabla\cdot(\alpha_s\rho_s V_s V_s) = \nabla\cdot\tau_s + \beta(V_g - V_s) - \alpha_s\nabla P_g - \nabla P_s + \alpha_s\rho_s g \quad (4)$$

Besides, a fluctuation kinetic energy equation is added considering the fluctuation energy of solid phase:

$$\frac{\partial}{\partial t}(\alpha_s\rho_s\theta) + \nabla\cdot(\alpha_s\rho_s V_s\theta) = \frac{2}{3}[(\tau_s - P\cdot I):\nabla V_s - \nabla\cdot(k_s\nabla\theta) - \gamma_s] \quad (5)$$

where, α, ρ, V, P, τ and μ are the volume fraction, density (kg·m^{-3}), velocity (m·s^{-1}), pressure (N·m^{-1}), shear stress tensor and dynamic viscosity coefficient, and the subscripts g and s represent gas phase and sand phase. β is the gas-solid drag coefficient, g is the gravitational acceleration (m·s^{-2}), I is the unit tensor, k_s is the energy transfer coefficient of particle-phase fluctuation energy (kg·m^{-1}·s^{-1}), γ_s is the pulsation energy dissipation of the particle phase caused by particle collisions

($kg·m^{-1}·s^{-3}$), θ is granular temperature ($m^2·s^{-2}$).

The discretization method is finite difference method. The structure of the FORTRAN program and the calculating time of each part are shown in Fig. 4.

Functions of sand phase and gas phase take up 78.26 % of total time, so these two functions are the main part to realize the parallel algorithm. CUDA was used to realize the parallelization program, and the other functions were included into the dynamic link library (DLL). Single process on Intel Xeon E5-2699 v4 CPU (2.2 GHz main frequency and 128 GB memory) was used, and the GPU is Tesla K80 (562 MHz main frequency, 24 GB GDDR5 RAM memory, 4992 CUDA cores and 480 $GB·s^{-1}$ bandwidth). The algorithm is mainly to design the task allocation and process schedule. We used one stream at first, and the algorithm was designed as Table 2.

It is obvious that this structure does not make full use of the GPU because the data transfers and the waiting of both the CPU and the GPU take up too much time, as shown in Fig. 5, especially as data transfer happens in every loop. The simulation

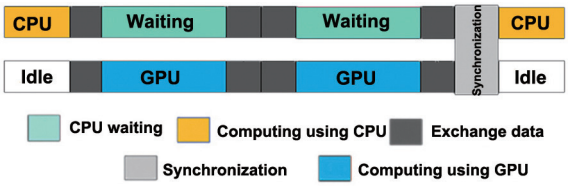

Fig. 5: Parallel scheme of core shooting program in each loop

results still agree well with the experimental results, but the increase in speed is only 2.5 times (Time is recorded when new data files generate).

2.2 Parallel program optimization

The most important reason for the limited speedup is that only 78.26% of the main structure is paralleled, as shown in Fig. 4. In fact, the remaining 21.74% DLL functions have a great impact on the improvement of computational efficiency. With 21.74 % DLL functions left, the maximal speedup can only reach to 5 times.

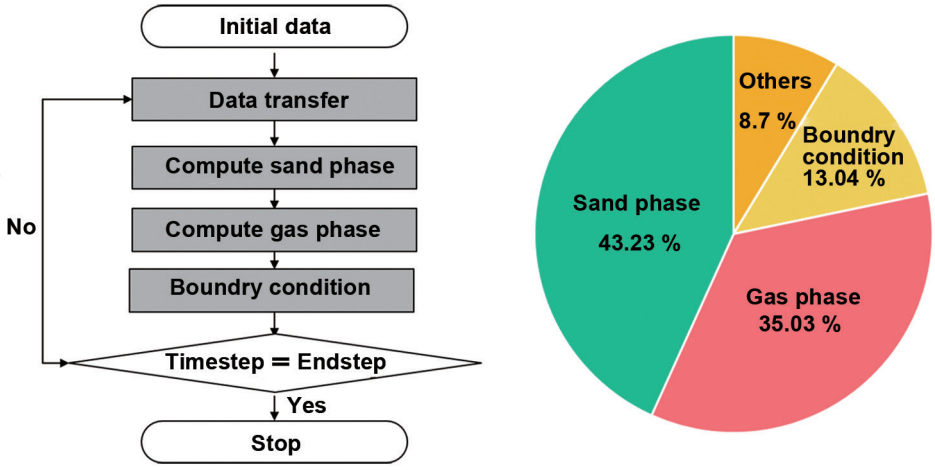

Fig. 4: Main structure of FORTRAN program and time consumption of each part

Table 2: Parameters, definations and algorithm design

Parameters		Definations
Input Parameters	P_vent:	Locations of vents
	Wall_sand:	Symbol of grids
	Cal_time:	Time of sand shooting process
Output Parameters	RHO:	The density of grids
	Ug, Vg, Wg:	The velocity of gas in 3 dimension
	Us, Vs, Ws:	The velocity of sand in 3 dimension
	Alpha_sand:	Sand volume fraction
	Theta:	Granular Temperature[19]
Variable Define	T1:	Current step
	T2:	Next Step
	Alpha_gas:	Gas volume fraction
	Stop_Flag:	The number of end step
	Current_Flag:	The current step

Algorithm design

```
Step 1: If(Cal_time == 0)
          Read data of P_vent and Wall_sand
        Else
          Read data of Ug,Vg,Wg,Us,Vs,Ws of T1
          Read data of RHO, Alpha_Sand, Theta of T1
Step 2: If (Current_Flag != Stop_Flag)
          Copy data to GPU
          Start stream0 of GPU to compute sand phase
          Return Us, Vs, Ws, Alpha_sand, Theta of T2
          Synchronize();
          Computing Alpha_gas
          Start stream0 of GPU to compute gas phase
          Return RHO, Ug, Vg, Wg of T2
          Synchronized();
          Return data to CPU
          Refresh boundry condition
          Exchange data from T2 to T1
          Current_Flag++;
        Else
          Synchronized();
          Write data files
Step 3: Check the behavior of sand flow using output data files
```

Data communication delay and the degree of parallelization of the program are two main factors to be considered, and two strategies are taken to optimize the program: Strategy 1 is to parallel the whole main structure and Strategy 2 is to use GPU asynchronous streams.

Main structures and parallel schemes of the two strategies are shown in Fig. 6.

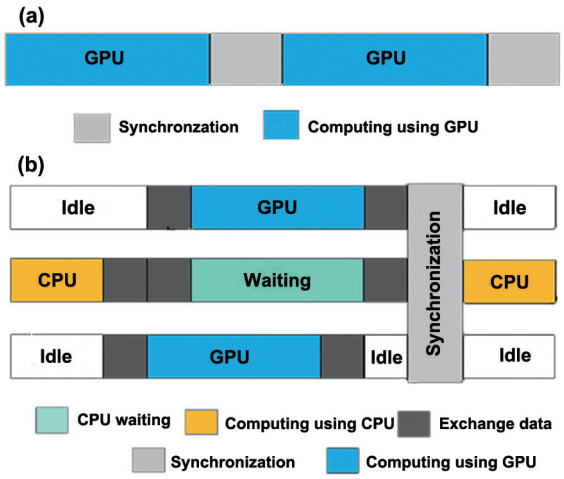

Fig. 6: Parallel scheme of strategies in each loop: (a) Strategy 1, (b) Strategy 2

The main structure of Strategy 1 is similar to the structure of the FORTRAN program, but it only transfers data between the CPU and GPU one time in the whole computing process, so the data communication delay is drastically reduced and the parallel scheme becomes so different (Fig. 6a). Besides, the maximal speed increase equals to the speed increase improved by the GPU parallelization method with the 21.74% DLL functions gone.

The main structure of Strategy 2 is also similar to the structure of the FORTRAN program. We can see that asynchronous streams are a good way to make full use of the waiting time of CPU and GPU. As for the program, the functions of sand phase and gas phase can be allocated to two separate streams, and it is notable that synchronization is necessary in asynchronous streams because the data exchange executes asynchronously.

Using asynchronous streams based on Strategy 1 is considered as another strategy, but considering the data exchange in asynchronous streams, this strategy is unrealistic.

2.3 Simulation results and discussion

The speed increase results of the two strategies are listed in Table 3. A combustor (the problem scale is 447,474, and the real size is 428 × 246 × 34 mm³) was simulated to validate the model in engineering application.

Table 3: Speedup results of two strategies

Speedup	Test piece	Combustor
Strategy 1	16.2	40.1
Strategy 2	4.3	–

For the test piece, the speed increase of Strategy 1 is 16.2 times compared with the results of the FORTRAN program (3.5 hours is reduced to 13 minutes) and the simulation results of sand flow (Fig. 7) are still consistent with experimental results. However, the speedup of Strategy 2 is 4.3 times compared with the results of the FORTRAN program (3.5 hours is reduced to 49 minutes). This indicates that the degree of parallelization of the program plays a more important role in the speedup of the program. Greater speed increases can be achieved by using asynchronous streams compared with the results in chapter 2.1, but the improvement is limited, compared with Strategy 1.

As is mentioned above, data communication delay is one of the main factors to influence the computing efficiency. Strategy 1 is aimed to optimize the communication delay between CPU and GPU, while communication delay also exists between GPU cores and memories in GPU. The global memory in GPU was chosen to save and exchange data but the share memory was not used. Shared memory has a much faster data communication speed than global memory[13], but it is necessary to divide the problem into proper sub-problems because shared memories cannot exchange data with each other and one shared memory can only support data communication in one block. Considering the program will deal with different sand cores with different shapes and different sizes, it is difficult to divide the problem into proper sub-problems. It is necessary to stress that we do not mean shared memory is useless in this problem, because it still can be used to further optimize the program within a specified range of core sand size.

For the combustor, the computing time on the CPU is 92 days. Simulation results reflect some behaviors of sand flow in experiments, and sand behaviors are well predicted in some special positions where the sand volume fraction of the lower part is not the highest, such as a position at the top of the combustor near the plunger pistons and a position at the bottom of the combustor near the vents. Those circumstances happen because of two reasons: the first reason is that the compressed air passing the vents will lower the density near the vents and plunger pistons, the second reason is that the asymmetrical core box plays an important role in the filling sequence of sand flow. Because of these two reasons, pass-ways between the left

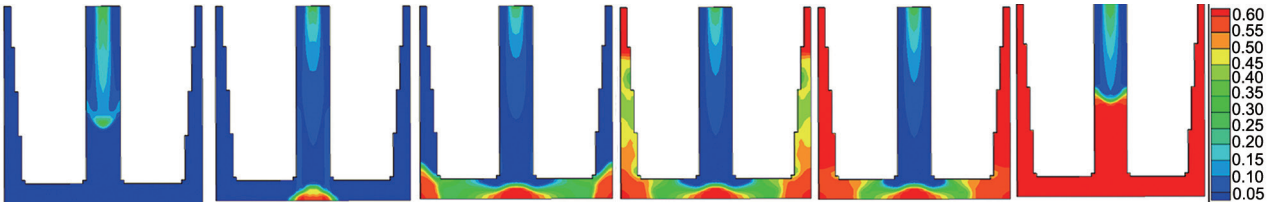

Fig. 7: Simulation results of optimized parallel program by Strategy 1

plunger pistons block first, flaws happen more easily in the pass-ways of the left, which agrees well with the experimental result. This will help analyze the flaws.

For the parallel program on GPU, the parallel program of Strategy 1 was used to simulate the sand flow, considering this strategy can result in better speed increases. The increase is 40.1 times (92 days was reduced to 2 days 7 hours) and it shows the increase change with the problem scale [20]. Some practical conclusions can be drawn under the simulations of different parameters. The proportion of sand and gas at the nozzle is an important parameter in the core shooting process. The simulation results of the combustor with 0.4 ratio of sand and gas on GPU show that pass-ways between plunger pistons block heavily because the sand volume fraction near the upper plunger pistons remains high. With the process of core shooting, sand becomes stuck in the plunger pistons and only a little sand reaches the bottom of the core box. Besides, the flaws are more obvious compared with the simulation results of which the proportion of sand and gas is 0.1. So there is a practical conclusion that increasing the proportion of sand and gas properly will help sand fill the core box and reduce flaws. However, there are some deviations in the simulation results when the problem scale increases: simulation results are in good agreement with experimental results when sand flow passes through the pass-ways at the top of the plunger pistons, but the filling becomes slower at the bottom of the combustor compared with experimental results. Considering the combustor is much more complex than the test piece, the parallel algorithm may not be stable enough in some places near the plunger pistons and then the subsequent filling is affected.

3 Conclusions

In this study, a parallel program was developed base on TFM and a kinetic-frictional constitutive correlation was added. A test piece and a combustor were simulated to investigate the model precision. With the validation of in-situ experiment results, simulation results indicate that the model can correctly reflect some behavior of sand flow in the core shooting process. Considering the long computing time when simulating some complex core boxes, a GPU parallelization method was used to accelerate the program, and great speed increases were achieved while some simulation results were still consistent with experimental results. The simulation program accelerated by GPU is applied to hopefully optimize the core shooting process.

Some limitations remain to be settled. Firstly, although a fluctuation kinetic energy equation is added into the model, whether the model can simulate turbulence is still remaining to be discussed. Secondly, experiments of the core shooting process are still limited, so the model cannot be validated with further experimental data. Last but not least, considering the deviation changing with problem scale, the GPU parallel algorithm of core shooting needs to be more stable as well.

References

[1] Bakhtiyarov S I, Overfelt R A. CFD modeling and experimental study of resin-bonded sand/air two-phase flow in sand coremaking process. Powder Technology, 2003, 133 (1): 68–78.
[2] Ni Chang-jiang, Lu Gao-chun, Zhang Qing-dong, et al. Influence of core box vents distribution on flow dynamics of core shooting process based on experiment and numerical simulation. China Foundry, 2016, 13 (1): 22–29.
[3] Ni Chang-jiang, Lu Gao-chun, Jing Tao, et al. Influence of core sand properties on flow dynamics of core shooting process based on experiment and multiphase simulation. China Foundry, 2017, 14 (2): 121–127.
[4] Czerwinski F, Mir M, Kasprzak W. Application of cores and binders in metalcasting. International Journal of Cast Metals Research, 2014, 28 (3): 129–139.
[5] Liu G Q, Li S Q, Zhao X L, et al. Experimental studies of particle flow dynamics in a two-dimensional spouted bed. Chemical Engineering Science, 2008, 63 (4): 1131–1141.
[6] Wu J J, Cui Y, Li W Z. Computer simulation of core-shooting process with two-phase flow. International Journal of Cast Metals Research, 2002, 15(4): 445–449.
[7] Winartomo B, Vroomen U, Hrigpolaczek A B, et al. Multiphase modelling of core shooting process. International Journal of Cast Metals Research, 2013, 18 (1): 13–20.
[8] Ni Changjiang, Lu Gaochun, Jing Tao, Junjiao Wu. Flow dynamic analysis of core shooting process through experiment and multiphase modeling. Advances in Materials Science and Engineering, 2016, 2016(6): 1–13.
[9] Ni Changjiang, Guo Enyu, Zhang Qingdong, et al. Frictional-kinetic modeling and numerical simulation of core shooting process. International Journal of Cast Metals Research, 2016, 29 (4): 214–221.
[10] Wang B, Zhang F, Xiang M. SAR raw signal simulation based on GPU parallel computation. Geoscience & Remote Sensing Symposium, 2009, 4: IV-617–IV-620
[11] Hyunwoo K, Kyoungsu O. A GPU-based light hierarchy for real-time approximate illumination. Visual Computer, 2008, 24 (7): 649–658.
[12] Kruger J, Westermann R. Acceleration techniques for GPU-based volume rendering. IEEE Visualization, 2003: 287–292.
[13] Navarro C A, Nancy H K, Luis M. A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures. Communications in Computational Physics, 2014, 15 (2): 285–329.
[14] Krakiwsky S E, Turner L E, Okoniewski M M. Graphics processor unit (GPU) acceleration of finite-difference time-domain (FDTD) algorithm. International Symposium on Circuits & Systems, 2004, 5(5): V-265–268.
[15] Krakiwsky S E, Turner L E, Okoniewski M M. Acceleration of finite-difference time-domain (FDTD) using graphics processor units (GPU). International Microwave Symposium Digest , 2004 , 2(2) :1033–1036.
[16] Yasushi Shibuta, Munekazu Ohno, Tomohiro Takaki. Solidification in a Supercomputer: From Crystal Nuclei to Dendrite Assemblages. JOM, 2015, 67 (8): 1793–1804.
[17] Tomohiro Takaki. Takashi Shimokawabe, Munekazu Ohno, et al. Unexpected selection of growing dendrites by very-large-scale phase-field simulation. Journal of Crystal Growth Volume, 2013, 382 (6): 21–25.
[18] Gidaspow D. Multiphase Flow and Fluidization: Continuum and Kinetic Theory Descriptions. Canadian Journal of Chemical Engineering, 1995, 73 (5): 577–784.
[19] Johnson P C, Jackson R. Frictional-collisional constitutive relations for granular materials, with application to plane shearing. Journal of Fluid Mechanics, 2006, 176 (176): 67–93.
[20] Y Wang, Dou Y, Guo S, et al. CPU–GPU hybrid parallel strategy for cosmological simulations. Concurrency & Computation Practice & Experience, 2014, 26 (3): 748–765.

This work was financially supported by the National Natural Science Foundation of China (51575304).