Oil & Gas Science and Technology – Rev. IFP Energies nouvelles **75**, 86 (2020)

https://doi.org/10.2516/ogst/2020076

Available online at:
ogst.ifpenergiesnouvelles.fr

REGULAR ARTICLE OPEN ACCESS

GPU-accelerated hydraulic simulations of large-scale natural gas pipeline networks based on a two-level parallel process

Yue Xiang[1], Peng Wang[2,*], Bo Yu[2], and Dongliang Sun[2]

[1] Sichuan Energy Internet Research Institute, Tsinghua University, 610042 Chengdu, PR China
[2] School of Mechanical Engineering, Beijing Key Laboratory of Pipeline Critical Technology and Equipment for Deepwater Oil & Gas Development, Beijing Institute of Petrochemical Technology, 102617 Beijing, PR China

Received: 10 March 2020 / Accepted: 15 September 2020

Abstract. The numerical simulation efficiency of large-scale natural gas pipeline network is usually unsatisfactory. In this paper, Graphics Processing Unit (GPU)-accelerated hydraulic simulations for large-scale natural gas pipeline networks are presented. First, based on the Decoupled Implicit Method for Efficient Network Simulation (DIMENS) method, presented in our previous study, a novel two-level parallel simulation process and the corresponding parallel numerical method for hydraulic simulations of natural gas pipeline networks are proposed. Then, the implementation of the two-level parallel simulation in GPU is introduced in detail. Finally, some numerical experiments are provided to test the performance of the proposed method. The results show that the proposed method has notable speedup. For five large-scale pipe networks, compared with the well-known commercial simulation software SPS, the speedup ratio of the proposed method is up to 57.57 with comparable calculation accuracy. It is more inspiring that the proposed method has strong adaptability to the large pipeline networks, the larger the pipeline network is, the larger speedup ratio of the proposed method is. The speedup ratio of the GPU method approximately linearly depends on the total discrete points of the network.

Nomenclature

d	Pipe diameter, m
g	Gravitational acceleration, m/s^2
m	Mass flowrate, kg/s
p	Pressure, Pa
t	Time, s
x	Spatial coordinate
A	Cross-sectional area, m^2
M	Number of pipelines in the pipeline network
N	Number of sections into which the pipelines are divided
N_{st}	Max number of reduced steps, $N_{\mathrm{st}} = \log_2^{N+1}$
U	General variable

Greek symbols

ρ	Density of gas, kg/m^3
θ	Inclination angle of the pipe, rad
λ	Friction coefficient
Δx	Mesh spatial size, m
Δt	Time step, s

Superscripts

n	Time level
k	kth reduced

Subscripts

i	Node number or pipeline number
in	Flow in
out	Flow out

1 Introduction

With the increasing demand for clean energy resources worldwide, natural gas plays a more important role than previously in the energy structure [1, 2]. Pipeline networks are the most common means of natural gas transportation. They are thus being constructed on a large scale in recent

* Corresponding author: wangp@bipt.edu.cn

years [3]. The scale of natural gas pipeline networks is becoming increasingly larger and the topological structure is becoming increasingly more complex [4].

Prediction of the flow parameters in a natural gas pipeline network is important to managers who create the operating schedule and forecast potential risk [5, 6]. Hydraulic simulation is the most common way to analyze the flow in a gas pipeline network [7, 8]. As early as the 1950s, hydraulic simulation for gas pipelines began with some simplified mathematical models [9, 10]. Some terms in the governing equations of pipeline flow, such as the inertia term, the transient term or the friction term, are usually ignored in these models. In later years, some numerical methods were proposed to solve the non-simplified mathematical model, such as the method of characteristics [11], the method of lines [12], and the finite difference method [13]. Among these methods, the implicit finite difference method is one of the most commonly used methods because of its high stability and accuracy.

However, because all the equations should be solved simultaneously at each time step, there is a large computational cost for the implicit finite difference method. To solve this issue, studies have been done to accelerate the computation. Luongo [14] proposed a novel linearization method for the partial differential equations in a gas pipeline, which can save 25% of the computational time. Behbahani-Nejad and Shekari [15] presented a reduced-order modeling approach based on the implicit finite model for natural gas transient flow in pipelines, showing a 71.14% time saving compared with the direct simulation of the origin nonlinear model. To further improve the efficiency of the implicit finite difference method, an adaptive implicit finite difference method for natural gas pipeline transient flow was introduced [16]. The results show that the calculation speed of the slow transient simulation is accelerated and the computation accuracy of the fast transient simulation is high. What is more, the Decoupled Implicit Method for Efficient Network Simulation (DIMENS) method was proposed in our previous study [17, 18]. DIMENS is a rapid method for the hydraulic simulation of natural gas pipeline networks. In DIMENS, the pipelines in the networks are decoupled and solved independently. Comparison with the well-known commercial simulation software SPS, the speedup ratio is 2.5. Although these methods improve the efficiency of the hydraulic simulation of gas pipeline networks, simulation efficiency is lacking.

In recent years, Graphic Processing Unit (GPU) acceleration has attracted considerable research interest in scientific computing. Many studies in various fields, e.g., fluid simulation [19–21], weather prediction [22, 23], seismic damage simulation [24], and sheet forming simulation [25], show great performance with GPU computing. GPU computing is the use of a GPU as a coprocessor to accelerate CPUs for general-purpose scientific and engineering computing. Compared with a CPU, on a single PC, GPU uses thousands of smaller and more efficient cores for a massively parallel architecture aimed at handling multiple functions at the same time. It provides superior processing power, memory bandwidth and efficiency over its CPU counterparts.

Obviously, if GPU technology is introduced into the simulation of natural gas pipeline networks, the computational efficiency would be improved substantially. However, to the best of authors' knowledge, it is hard to see any literature focusing on the GPU-accelerated simulations of natural gas pipeline networks due to the lack of suitable parallel algorithm. Fortunately, the DIMENS method in our previous work [17, 18] is a highly parallel algorithm. The core idea of this method is that the pipelines are first decoupled and then solved independently, thus the parallel solution process is very suitable for GPU computing. In other word, with the combination of proposed DIMENS method with GPU technology, the computational speed of natural gas pipeline networks simulation would be greatly accelerated. Therefore, to further improve the simulation efficiency of natural gas pipeline networks, our DIMENS method is extended to a two-level parallel process, and then it is implemented on GPU. A novel two-level parallel simulation process and the corresponding parallel numerical method for hydraulic simulation of natural gas pipeline networks are proposed. The proposed method can effectively improve the simulation efficiency of large-scale natural gas pipeline network, and can be used to guide decisions regarding the design and operation of the pipeline network system.

The outline of this paper is the following. In Section 2, the mathematical model and the DIMENS method are briefly introduced and analyzed. In Section 3, the idea of the two-level parallel process and the corresponding parallel numerical method for the hydraulic simulation of large-scale networks are proposed. In Section 4, the implementation of the two-level parallel process with GPUs is presented in detail. In Section 5, numerical experiments are presented to verify the accuracy and efficiency of the proposed GPU-accelerated method. The conclusions are summarized in Section 6.

2 Mathematical model and the DIMENS method

In this section, the mathematical model for a gas pipeline network and the DIMENS method are reviewed. For a brief and clear description, a network that includes only the pipeline, the supply and the demand, are studied. The mathematical model of complex networks, which also includes a compressor and a valve can be found in the study [17, 18].

2.1 Mathematical model

2.1.1 Pipeline model and discretization

According to our previous study, the hydraulic governing equations of a gas pipeline, incorporating the continuity equation and momentum equation, is written as a general formula [17, 18] as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{B} \cdot \frac{\partial \mathbf{U}}{\partial x} = \mathbf{F}, \tag{1}$$

where $\mathbf{U} = \begin{bmatrix} p \\ m \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 & \frac{1}{A}\left(\frac{\partial p}{\partial \rho}\right)_T \\ \left[A - \frac{m^2}{A\rho^2}\left(\frac{\partial \rho}{\partial p}\right)_T\right] & \frac{2m}{A\rho} \end{bmatrix}$,

$\mathbf{F} = \begin{bmatrix} \left(\frac{\partial p}{\partial T}\right)_\rho \frac{\partial T}{\partial t} \\ -\frac{\lambda}{2}\frac{m|m|}{dA\rho} - A\rho g \sin\theta + \frac{m^2}{A\rho^2}\left(\frac{\partial \rho}{\partial T}\right)_p \frac{\partial T}{\partial x} \end{bmatrix}$.

The hydraulic equation is a nonlinear partial difference equation, and its direct solution is usually unstable and inefficient [7, 26]. Therefore, equation (1) is linearized as equation (2) based on the Taylor expansion [27] as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \overline{\mathbf{B}} \cdot \frac{\partial \mathbf{U}}{\partial x} + \overline{\mathbf{G}} \cdot \left(\mathbf{U} - \overline{\mathbf{U}}\right) = \overline{\mathbf{F}} + \overline{\mathbf{S}} \cdot \left(\mathbf{U} - \overline{\mathbf{U}}\right), \quad (2)$$

where, the elements (i, j) of matrices $\mathbf{G}$ and $\mathbf{S}$ are $[\mathbf{G}]_{i,j} = \sum_{l=1}^{l=2}\left(\frac{\partial \mathbf{B}}{\partial u_j}\right)_{i,l}\frac{\partial u_l}{\partial x}$ and $[\mathbf{S}]_{i,j} = \frac{\partial F_i}{\partial u_j}$, respectively. The detailed expressions of $\mathbf{G}$ and $\mathbf{S}$ can be found in Appendix A. u_i is the ith corresponding component of $\mathbf{U}$. $\overline{\mathbf{B}}$, $\overline{\mathbf{G}}$, $\overline{\mathbf{F}}$, $\overline{\mathbf{S}}$ and $\overline{\mathbf{U}}$ are calculated using the results of the previous time step.

The implicit finite difference method is adopted to discretize equation (2) on a uniform pipe mesh, as shown in Figure 1. The space grid and time grid are divided horizontally and vertically, and Δx and Δt denote the space step and the time step, respectively.

For the ith section in Figure 1, the following discretization is conducted [17, 18, 28, 29]:

$$\mathbf{U} = \frac{\mathbf{U}_{i+1}^{n+1} + \mathbf{U}_i^{n+1}}{2}, \quad (3)$$

$$\overline{\mathbf{U}} = \frac{\mathbf{U}_{i+1}^{n} + \mathbf{U}_i^{n}}{2}, \quad (4)$$

$$\frac{\partial \mathbf{U}}{\partial t} = \frac{\mathbf{U} - \overline{\mathbf{U}}}{\Delta t}, \quad (5)$$

$$\frac{\partial \mathbf{U}}{\partial x} = \frac{\mathbf{U}_{i+1}^{n+1} - \mathbf{U}_i^{n+1}}{\Delta x}. \quad (6)$$

By substituting equations (3)–(6) into equation (2) and rearranging the terms, the discretization is obtained as follows:

$$\mathbf{CE}_i \cdot \mathbf{U}_i^{n+1} + \mathbf{DW}_i \cdot \mathbf{U}_{i+1}^{n+1} = \mathbf{H}_i, \quad (7)$$

where $\mathbf{CE}_i = \frac{1}{2\Delta t}\mathbf{I} - \frac{1}{\Delta x}\overline{\mathbf{B}} + \frac{1}{2}\left(\overline{\mathbf{G}} - \frac{\overline{\partial \mathbf{F}}}{\partial \mathbf{U}^T}\right)$, $\mathbf{DW}_i = \frac{1}{2\Delta t}\mathbf{I} + \frac{1}{\Delta x}\overline{\mathbf{B}} + \frac{1}{2}\left(\overline{\mathbf{G}} - \frac{\overline{\partial \mathbf{F}}}{\partial \mathbf{U}^T}\right)$, $\mathbf{H}_i = \overline{\mathbf{F}} + \left(\overline{\mathbf{G}} - \frac{\overline{\partial \mathbf{F}}}{\partial \mathbf{U}^T} + \frac{1}{\Delta t}\right)\overline{\mathbf{U}}$, $\mathbf{I}$ is a 2×2 identity matrix.

2.1.2 Boundary conditions

The supply and demand are referred to as the gas source. The mathematical model of the gas source is shown in equations (8)–(9). They are also called the external boundary conditions of pipeline network:

$$p = p(t), \quad (8)$$

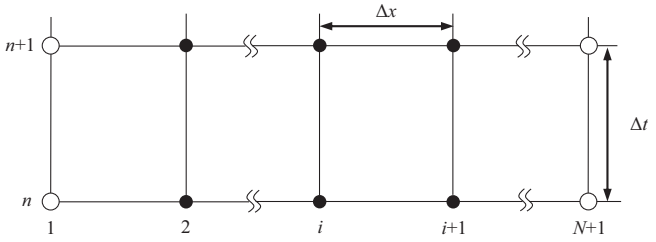

Fig. 1. Schematic diagram of the uniform mesh.

$$m = m(t). \quad (9)$$

At the junction node of the components, the laws of mass conservation and pressure balancing should be satisfied. The mathematical model of a junction node is shown in equations (10)–(11), which are also called internal boundary conditions, as follows:

$$\sum m_{\text{in}} = \sum m_{\text{out}}, \quad (10)$$

$$p_{\text{in},1} = p_{\text{in},2} = \ldots = p_{\text{out},1} = p_{\text{out},2} = \ldots. \quad (11)$$

Equations (7)–(11) are the hydraulic discretized equations of a natural gas pipeline network. An Equation Of State (EOS) which expresses the density in terms of pressure and temperature needs to be close to these equations. Several different EOSs, such as Peng–Robinson (PR) and Soave–Redlich–Kwong (SRK), are commonly used in natural gas industry [30]. Solving this system of hydraulic discretized equations, the pressure and mass flowrate in the pipe can be obtained.

2.2 DIMENS method

The DIMENS method is proposed based on the idea of divide-and-conquer in our previous study [17, 18]. In the DIMENS method, the gas pipeline network is divided into several pipelines and a set of junction nodes, then the equations of the pipeline and those of junction nodes are solved independently. In this way, the system of origin discretized equations is split into two parts: the system of discretized equations for the pipe in equation (7) and the system of boundary equations in equations (8)–(11). In this paper, we call them PDES and BES, respectively. There are as many PDESs in the system of discretized equations as there are pipelines in a network.

To describe the decoupling of the DIMENS method, a simple pipeline network of 9 nodes and 12 pipes is used as an example, as shown in Figure 2, where red circles represent nodes and black lines represent pipes. The blue points indicate the spatial discretized points of the pipe. Using the DIMENS method, the simple network is divided into 9 nodes and 12 pipes. Using this method, one set of original equations with $9 + 12N$ size is divided into one BES with 9 size and twelve block PDES with N size.

The solution process of the PDES and the BES using the DIMENS method can be shown in Figure 3. More details about the DIMENS method can be found in the literature [17, 18].

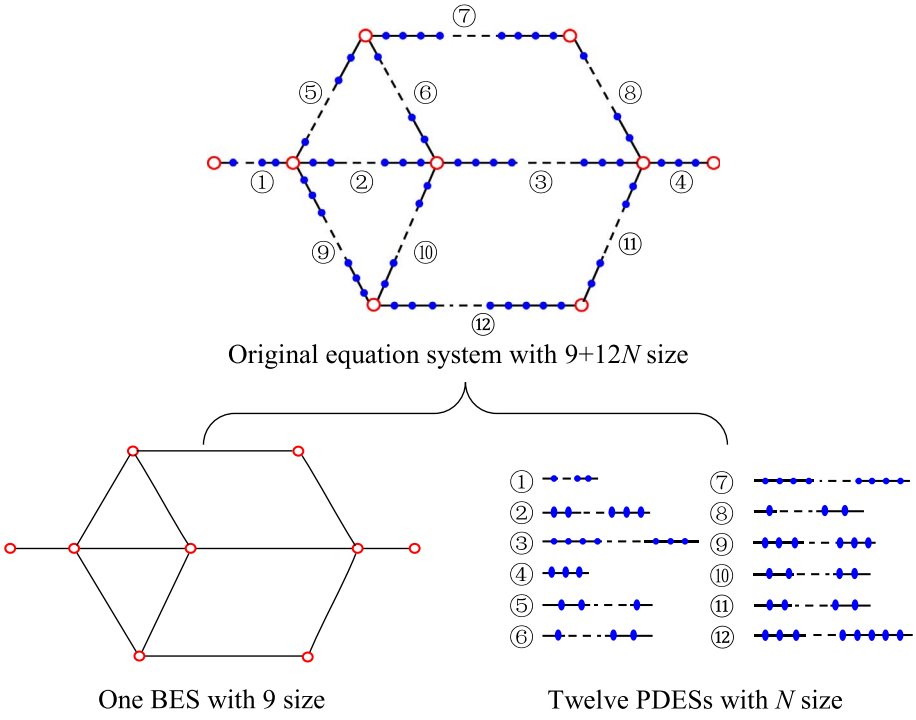

Fig. 2. Decoupling sketch of the simple network of 9 nodes and 12 pipes.

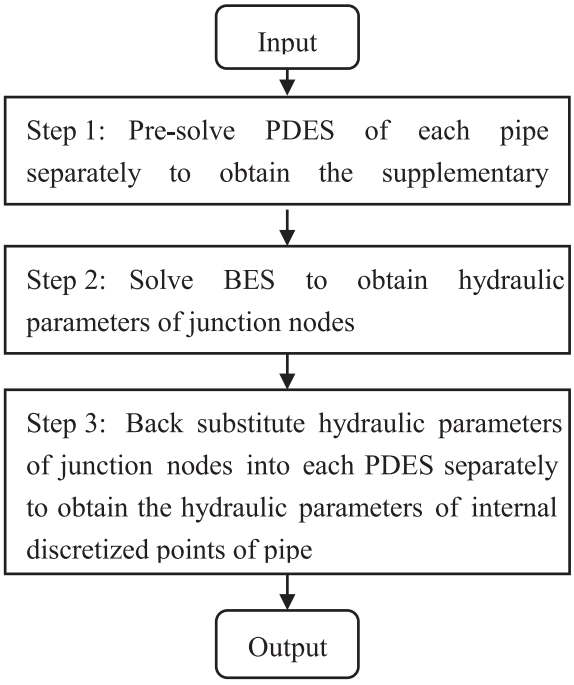

Fig. 3. Flow diagram of the DIMENS method for pipeline networks.

In the DIMENS method, the gas pipeline network is divided into several pipelines and a set of junction nodes, which can be simulated separately. The evidence in Figure 3 shows that the solution process of the PDESs in both step 1 and step 3 are separate. These analyses show that the DIMENS method is a highly parallel simulation method. However, the DIMENS method in our previous study [16, 17] is a single process. The advantages of the DIMENS method in simulation efficiency has not been fully utilized. Parallelization of the DIMENS method is an effective method to further improve the simulation efficiency in pipeline networks.

3 Two-level parallel simulation process and numerical method

In order to implement our DIMENS method on the GPU, the DIMENS method is extended to a novel two-level parallel simulation process and the corresponding parallel numerical method is introduced in detail in the following sections.

3.1 Two-level parallel simulation process

Because of the lightweight GPU computing, the parallel granularity should be fine enough to maximize the GPU utilization. Therefore, the two-level parallel simulation process is designed to be coarse and fine level processes. In the coarse level process, the task of network simulation is first divided into several solution tasks for the PDESs and the BES. In the fine level process, the solution of the PDES or the BES is then divided into parallel arithmetic operations to minimize computational effort.

3.1.1 Coarse level parallel process

In DIMENS, the solution process of the PDESs is separate; therefore, each PDES can be treated as a subtask. Because

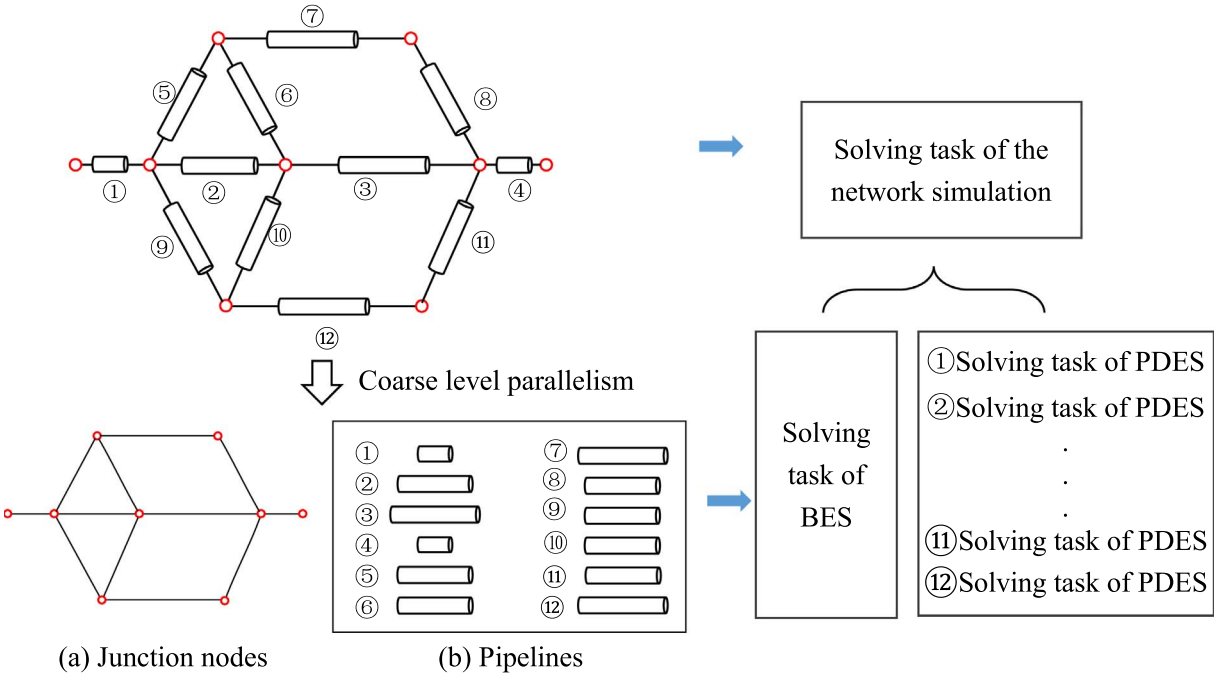

Fig. 4. Sketch of the coarse level division of a simple network.

the junction nodes and gas sources are independent of the pipe, BES can be similarly treated as another subtask. That is, in the coarse level parallel process, the task of network simulation is divided into several solution tasks for PDESs and BES, the solution process of a PDES or a BES is considered a subtask. Figure 4 shows a sketch of the coarse level division of a simple network. The sketch shows a simple network divided into several pipelines and a set of junction nodes. The task of network simulation is divided into twelve solution tasks for the PDESs and one solving task for the BES.

3.1.2 Fine level process

There are usually many discretized points in one pipeline and many junction nodes in a network; therefore, the solving task for PDES or BES is heavy even when using GPU, which could result in low computational speed. To solve this problem, fine level patterns for division of the above tasks are proposed. In the fine level process, the solution task for the PDES or the BES is divided into parallel arithmetic operations, and each arithmetic operation is a subtask.

1. Solution task for the PDES.
   In our previous study [17, 18], the PDES, expressed as equation (7), are transformed into the block tridiagonal equations in equation (12) as follows:

$$\mathbf{A}_i\mathbf{X}_{i-1} + \mathbf{B}_i\mathbf{X}_i + \mathbf{C}_i\mathbf{X}_{i+1} = \mathbf{D}_i, \tag{12}$$

where,

$$\mathbf{A}_i = \begin{bmatrix} \mathrm{CE}_{i-1}(2,1) & \mathrm{CE}_{i-1}(2,2) \\ 0 & 0 \end{bmatrix};$$

$$\mathbf{B}_i = \begin{bmatrix} \mathrm{DW}_{i-1}(2,1) & \mathrm{DW}_{i-1}(2,2) \\ \mathrm{CE}_i(1,1) & \mathrm{CE}_i(1,2) \end{bmatrix};$$

$$\mathbf{C}_i = \begin{bmatrix} 0 & 0 \\ \mathrm{DW}_i(2,1) & \mathrm{DW}_i(2,2) \end{bmatrix};$$

$$\mathbf{D}_i = [H_{i-1}(2), H_i(1)]^T;$$

$\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{C}_i$ and $\mathbf{D}_i$ are equation coefficients, which are all 2×2 matrices; $\mathbf{X}_i$ are the fundamental and particular solutions of equation (7); and subscript i represents the ith discretized point.

The PDES solution is the main computational stack of DIMENS. Therefore, parallelizing the solution process of equation (12) would greatly improve the computation speed. In our previous study [17, 18], equation (12) was solved by the block TDMA method. The TDMA method is a Gaussian-elimination-based method, which is usually difficult to parallelize [31] on a GPU.

In this study, the Parallel Cyclic Reduction (PCR) method [32] is adopted to parallelize the PDES solution. The PCR method was initially designed to solve scalar tridiagonal equation systems by Hockney and Jesshope in 1981. In this study, it is extended to solve the vectorial equation system in equation (12). The subtask of solving the PDES is the reduction and solution step of PCR. First,

the basic parallel process of PCR is briefly introduced, then the parallel solution process of the PDES solution is described in detail in Section 3.2.2.

2. Solution task for BES.
The solution of the BES, which also needs parallelization with fine granularity, is the other important part of the computational stack in DIMENS. The hydraulic variables of the start and end points of all pipelines are arranged by the number of pipelines. Then, BES can be written as follows:

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{b}, \tag{13}$$

where,

$$\mathbf{x}=[p_{1,1}, m_{1,1}, p_{N_1+1,1}, m_{N_1+1,1}, \cdots, p_{1,i}, m_{1,i}, p_{N_i+1,i}, m_{N_i+1,i}, \cdots, p_{1,M}, m_{1,M}, p_{N_M+1,M}, m_{N_M+1,M}]^T,$$

subscript (i, j) represents the ith point of the jth pipeline.

Because of the sparsity and irregularity of the BES, a modified Conjugate Gradient (CG) algorithm was adopted to solve it with high efficiency in our previous study [17, 18]. Because of the large number of matrix–matrix and matrix–vector multiplications in the CG method, it is readily parallelized by assigning a few steps of the arithmetic operations to each computing core. The subtasks in solving the BES are the arithmetic operations.

3.2 Parallel numerical method

3.2.1 Basic process of PCR

In PCR, the one system of original equations is reduced to a system of twice the number of reduced equations, which has only half the number of unknowns. The reduction procedure does not stop until each equation system has at most 2 unknowns. The equations of 2 unknowns are readily solved. Figure 5 shows the reduction pattern of the PCR method in a simple case 8 equations. Letters e' and e'' represent the updated equations. Equations in a yellow rectangle form a system. The parallel process is introduced briefly in the following.

Step ①: The 1st reduction is performed by reducing the even-indexed and odd-indexed unknowns, reducing the one system of 8-unknown equations to two systems of 4-unknown equations.
Step ②: The 2nd reduction is performed by the same operation, reducing the two systems of 4-unknown equations to four systems of 2-unknown equations.
Step ③: The four systems of 2-unknown equations are readily solved. There are reduction and solution processes in the PCR method.

The system of 8-unknown equations is thus solved in 3 steps using the PCR method. In addition, the process for each step in PCR is parallel. Therefore, when the PCR method is extended to solve the PDES, *i.e.*, equation (12), they can be solved by $N+1$ logical computing cores in only $N_{st}=\log_2^{N+1}$ steps. In this way, the computing task for the PDES can be lightweight and efficiently executed with GPU.

3.2.2 PCR method for PDES solution

According to the basic parallel process of PCR, the parallel solution process of the PDES solution can be divided into two steps: the reduction step and the solution step. These two steps are described in detail in the following.

1. Reduction step:
For the one system in equation (12), all even-indexed and odd-indexed equation coefficients in parallel with index i are updated as a linear combination of equation coefficients i, $i+1$ and $i-1$, obtaining the kth reduced system shown in equation (13). The number of equations in the system represented by equation (13) is $\frac{N+1}{2^k}$, where the number of equation (13) systems is 2^k:

$$\mathbf{A}_i^k\mathbf{X}_{i-2^k}+\mathbf{B}_i^k\mathbf{X}_i+\mathbf{C}_i^k\mathbf{X}_{i+2^k}=\mathbf{D}_i^k. \tag{13}$$

For the internal equation in each equation (13) system, *i.e.*, $2^k<i<N+1-2^k$, $k=1, 2, \ldots, N_{\mathrm{st}}-1$, the equation coefficients are rewritten as follows:

$$\mathbf{A}_i^k=-\mathbf{A}_i^{k-1}(\mathbf{B}_{i-2^{k-1}}^{k-1})^{-1}\mathbf{A}_{i-2^{k-1}}^{k-1};$$

$$\mathbf{B}_i^k=-\mathbf{A}_i^{k-1}(\mathbf{B}_{i-2^{k-1}}^{k-1})^{-1}\mathbf{C}_{i-2^{k-1}}^{k-1}+\mathbf{B}_i^{k-1}-\mathbf{C}_i^{k-1}(\mathbf{B}_{i+2^{k-1}}^{k-1})^{-1}\mathbf{A}_{i+2^{k-1}}^{k-1};$$

$$\mathbf{C}_i^k=-\mathbf{C}_i^{k-1}(\mathbf{B}_{i+2^{k-1}}^{k-1})^{-1}\mathbf{C}_{i+2^{k-1}}^{k-1};$$

$$\mathbf{D}_i^k=-\mathbf{A}_i^{k-1}(\mathbf{B}_{i-2^{k-1}}^{k-1})^{-1}\mathbf{D}_{i-2^{k-1}}^{k-1}+\mathbf{D}_i^{k-1}-\mathbf{C}_i^{k-1}(\mathbf{B}_{i+2^{k-1}}^{k-1})^{-1}\mathbf{D}_{i+2^{k-1}}^{k-1}.$$

For the first equation in each equation (13) system, *i.e.*, $i=2^k$, the equation coefficients are rewritten as follows:

$$\mathbf{A}_i^k=0;$$

$$\mathbf{B}_i^k=\mathbf{B}_i^{k-1}-\mathbf{C}_i^{k-1}(\mathbf{B}_{i+2^{k-1}}^{k-1})^{-1}\mathbf{A}_{i+2^{k-1}}^{k-1};$$

$$\mathbf{C}_i^k=-\mathbf{C}_i^k(\mathbf{B}_{i+2^{k-1}}^k)^{-1}\mathbf{C}_{i+2^{k-1}}^k;$$

$$\mathbf{D}_i^k=\mathbf{D}_i^{k-1}-\mathbf{C}_i^{k-1}(\mathbf{B}_{i+2^k}^{k-1})^{-1}\mathbf{D}_{i+2^k}^{k-1}.$$

For the last equation in each equation (13) system, *i.e.*, $i=N+1-2^k$, the equation coefficients are rewritten as follows:

$$\mathbf{A}_i^k=-\mathbf{A}_i^{k-1}(\mathbf{B}_{i-2^{k-1}}^{k-1})^{-1}\mathbf{A}_{i-2^{k-1}}^{k-1};$$

$$\mathbf{B}_i^k=-\mathbf{A}_i^{k-1}(\mathbf{B}_{i-2^{k-1}}^{k-1})^{-1}\mathbf{C}_{i-2^{k-1}}^{k-1}+\mathbf{B}_i^{k-1};$$

$$\mathbf{C}_i^k=0;\quad \mathbf{D}_i^k=-\mathbf{A}_i^{k-1}(\mathbf{B}_{i-2^{k-1}}^{k-1})^{-1}\mathbf{D}_{i-1}^{k-1}+\mathbf{D}_i^{k-1}.$$

The superscripts $k-1$ and k represent the $(k-1)$th and the kth reduced systems, respectively.

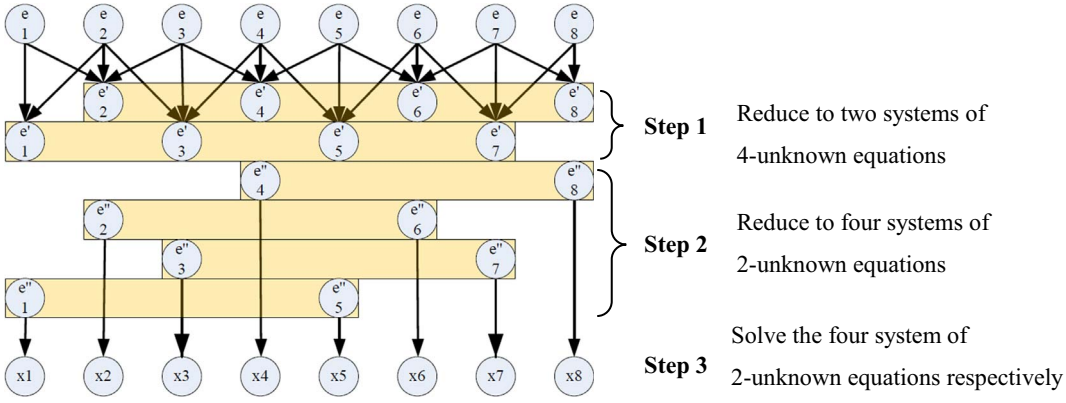

Fig. 5. Reduction pattern of PCR in the 8-unknown case.

2. Solution step:
When $k = N_{st} - 1$, the number of equations in the equation (13) system is 2. The two equations of the $(N_{st} - 1)$th reduced system are as follows:

$$\mathbf{B}_i^k \mathbf{X}_i + \mathbf{C}_i^k \mathbf{X}_{i+2^k} = \mathbf{D}_i^k, \tag{14}$$

$$\mathbf{A}_{i+2^k}^k \mathbf{X}_i + \mathbf{B}_{i+2^k}^k \mathbf{X}_{i+2^k} = \mathbf{D}_{i+2^k}^k. \tag{15}$$

Equations (10) and (11) can be readily solved and $\mathbf{X}_i$ and $\mathbf{X}_{i+2^k}$ are obtained as follows:

$$\mathbf{X}_i = [\mathbf{B}_i^k - \mathbf{C}_i^k (\mathbf{B}_{i+2^k}^k)^{-1} \mathbf{A}_{i+2^k}^k]^{-1} [\mathbf{D}_i^k - \mathbf{C}_i^k (\mathbf{B}_{i+2^k}^k)^{-1} \mathbf{D}_{i+2^k}^k], \tag{16}$$

$$\mathbf{X}_{i+2^k} = [\mathbf{B}_{i+2^k}^k - \mathbf{A}_{i+2^k}^k (\mathbf{B}_i^k)^{-1} \mathbf{C}_i^k]^{-1} [\mathbf{D}_{i+2^k}^k - \mathbf{A}_{i+2^k}^k (\mathbf{B}_i^k)^{-1} \mathbf{D}_i^k], \tag{17}$$

where $k = N_{st} - 1$, $i = 1, 2, \cdots \frac{N+1}{2}$.

Equation (13) is the reduction process, and equations (16)–(17) are the solution process. The three equations compose the computing task in the fine level process of the PDES solution. There are several steps of 2×2 matrix multiplication for each discretized point, which is a series of arithmetic operations. Therefore, the computing task is lightweight and equation (12) can be solved efficiently on GPU.

3.3 Flow diagram of the two-level parallel simulation process

Now, we establish a two-level model for the division of the computing task. Figure 6 shows a flow diagram of the two-level parallel simulation process.

4 Implementation of the two-level parallel simulation process on GPU

In this section, the implementation of the two-level parallel simulation process on GPU is described. In contrast with CPU programming, programming for GPU must consider the relation between the code and the hardware, especially thread mapping, memory organization and memory access. The implementation of these three parts are discussed below.

4.1 Thread mapping model

To manage and dispatch the GPU threads as required by the algorithm, *i.e.*, establishing a correct thread mapping model, we need a developing tool such as OpenCL or the Compute Unified Device Architecture (CUDA) to aid in the GPU programming. Between the two tools, CUDA is the most widely used GPU programming environment based on C/C++ for GPUs produced by *NVIDIA Corporation* [33]. As shown in Figure 7, CUDA uses a two-level thread mapping model, *i.e.*, the grid level and the thread block level. Host and Device indicate CPU and GPU, respectively. Kernel represents the part that can be parallelized in the problem; it is launched by CPU and executed on GPU. Each Kernel has a grid structure, consisting of multiple thread Blocks, and each thread Block also consists of multiple threads with a grid structure.

As shown in Figure 7, GPU is a double-level parallel simulation process, which perfectly integrates with the two-level parallel simulation process proposed in this paper. In general, the coarse-level and fine-level parallel computing tasks would be mapped to grid and thread blocks, respectively. Thread blocks are executed concurrently on GPU, and threads in a same block are also executed concurrently. Therefore, there are two thread mapping models: the first model is the DIMENS method, which is the coarse-level thread mapping model; and the second model is for solving PDES and BES, which is the fine-level thread mapping model.

Since the pipes are decoupled by DIMENS, the coarse-level thread mapping model is readily constructed, as shown in Figure 8. In this model, each thread block is responsible for solving the PDES of the corresponding pipe.

In the fine-level thread mapping model used to solve the PDES, the computing tasks for each discrete point, *i.e.*, equations (13), (16) and (17), are mapped to a single thread in the corresponding thread block. Figure 9 is a sketch of the thread mapping model.

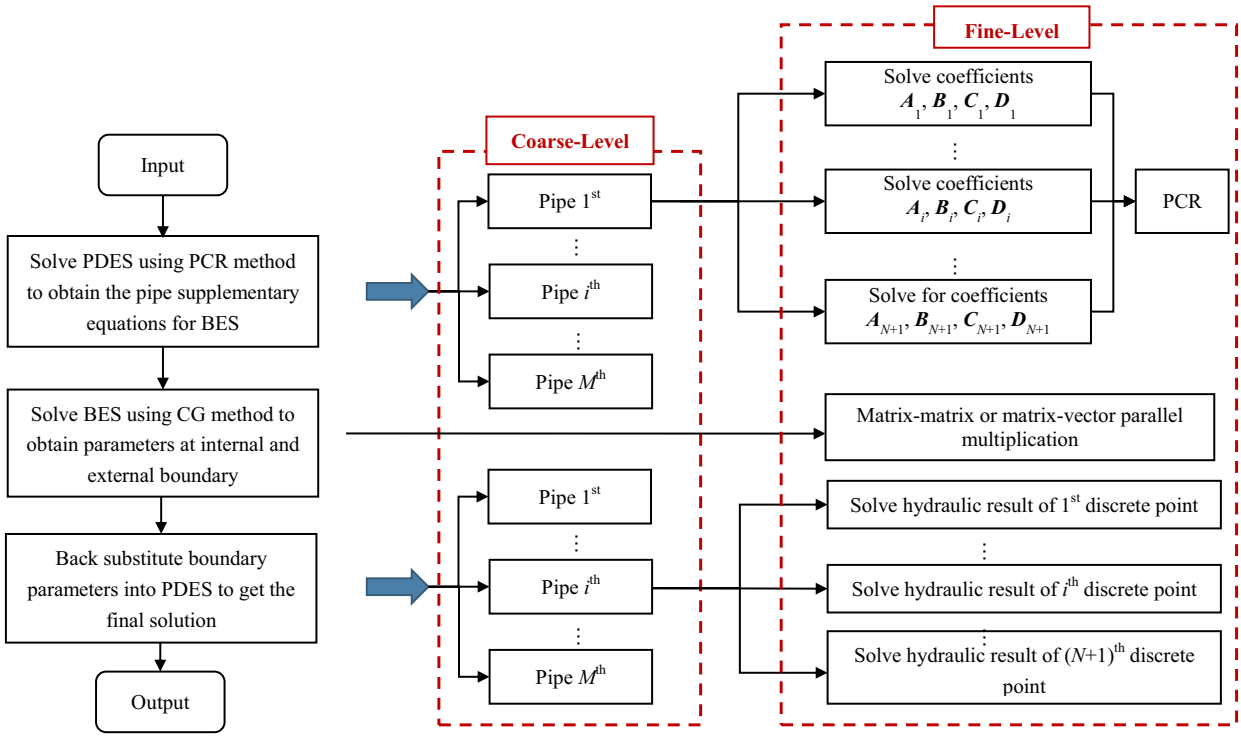

Fig. 6. Flow diagram of the two-level parallel simulation process.

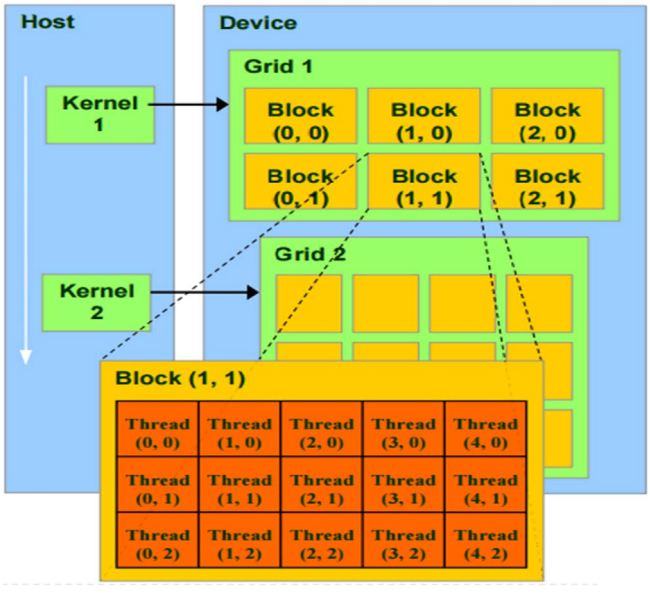

Fig. 7. CUDA programming model.

With respect to solving the BES, the thread mapping model is developed for matrix–matrix and matrix–vector multiplications. Fortunately, CUDA provides abundant libraries that have highly optimized functions for linear algebra, such as cuSPARSE, cuSOLVER, and cuBLAS in CUDA Toolkit Documentation v7. 5. In this study, the cuSPARSE library is used to accelerate the CG method for solving the BES.

4.2 Memory organization

Memory organization is the way of distributing different data to different types of memory on the GPU. There are five primary types of memory on GPU, *i.e.*, registers, shared memory, global memory, constant memory and texture memory. Figure 10 shows the memory structure on GPU, where the data stored in Register and Shared Memory are private for thread and thread block, respectively. Data stored in Global, Constant and Texture memory are public for all threads and thread blocks. Table 1 demonstrates the access latency of different types of memory.

Although registers and shared memory have excellent performance compared with the other types of memory, there are very limited resources for registers and shared memory on GPU. Therefore, the memory organization should be carefully designed such that those frequently used data can be stored in high-speed memory to increase the cache hit rate. Table 2 shows the memory organization for solving PDES on GPU.

4.3 Memory access

In addition to an appropriate memory organization, coalesced access for global memory is also required to avoid repetitive access with low efficiency [27], *i.e.*, data locality is very important. Some APIs provided by CUDA, e.g., cudaMallocPitch, have been developed to support coalesced access for data in global memory.

The performance of the above optimization is tested with different numbers of discrete points in the pipe, *i.e.*, different size of PDES. Figure 11 shows a great

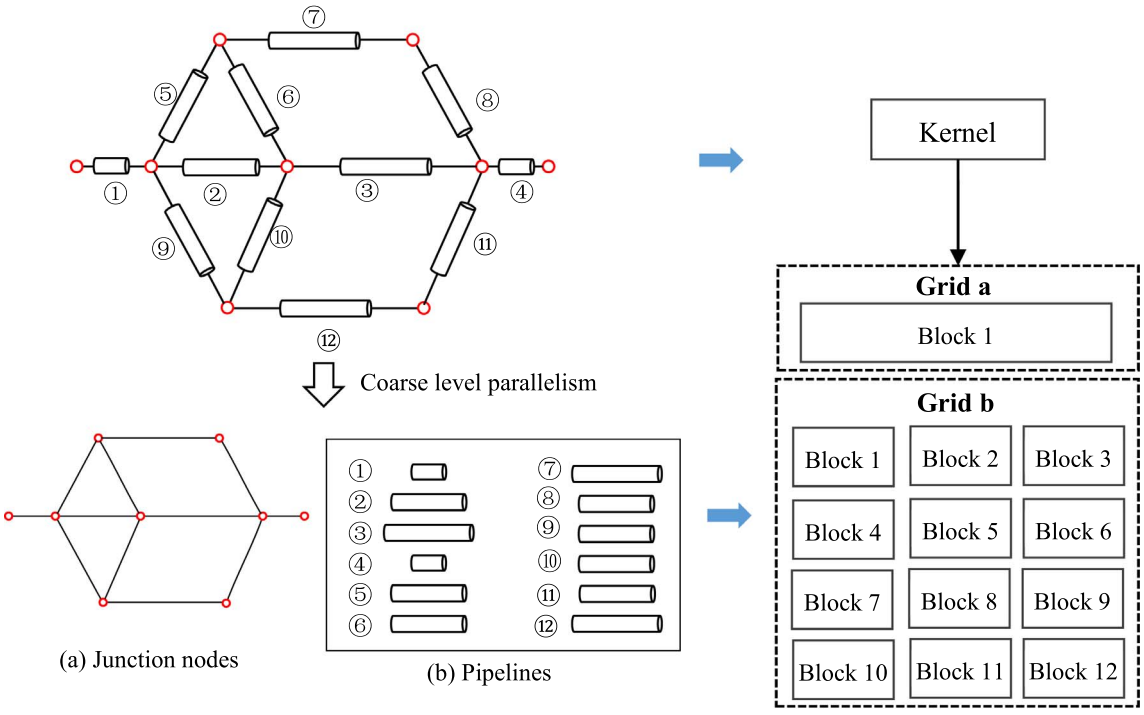

Fig. 8. Thread mapping model of the coarse-level parallelism.

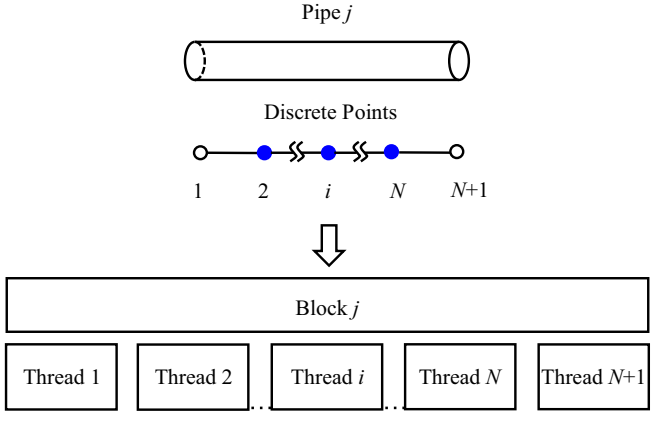

Fig. 9. Thread mapping model of the fine-level parallelism for solving the PDES.

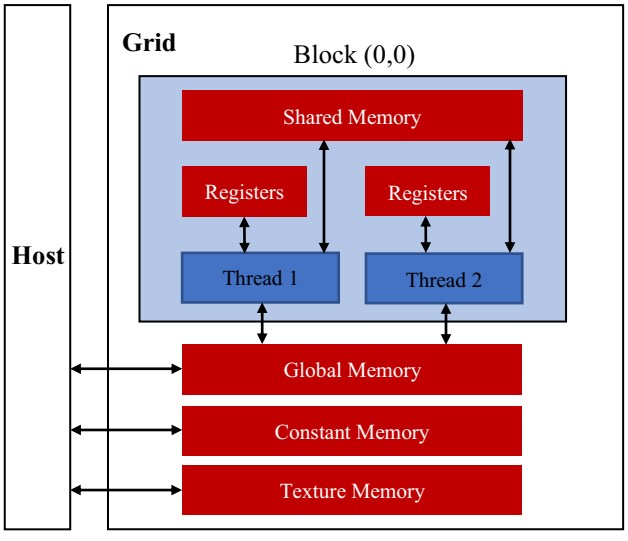

Fig. 10. Sketch of the memory structure on GPU.

improvement in computing performance using high-speed memory and coalesced memory accessing. All the numerical experiments conducted in this study are based on the hardware configuration introduced in Section 5. The data in Figure 11 show that in these four cases, the speedup ratio for using both optimization methods are 5.67x, 4.00x, 3.83x and 3.57x, respectively.

5 Case study

In this section, experiments are provided to test the performance of the proposed GPU simulation of a large-scale gas pipeline network, short as GPU method. To illustrate the efficiency of GPU acceleration, the computing speed increase is compared with that of the DIMENS method which implemented on CPU by sequential FORTRAN program and with the widely used commercial software Stoner Pipeline Simulator (SPS) in these experiments. In this paper, a PC with E5-2640 V3 CPU and GTX 980 GPU is used for the experiments and the CUDA version is 7.5.

Table 1. Access performance of different types of memory.

Memory type	Registers	Shared memory	Global memory	Constant memory	Texture memory
Band width (TB/S)	≈8	≈1.5	≈0.2	≈0.2	≈0.2
Access latency (clock cycle)	1	1~32	400 ~ 600	400 ~ 600	400 ~ 600

Table 2. Memory organization for solving PDES on GPU.

Memory type	Registers	Shared memory	Global memory
Data stored	Intermediate variables	$\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{C}_i$, $\mathbf{D}_i$	Result variables

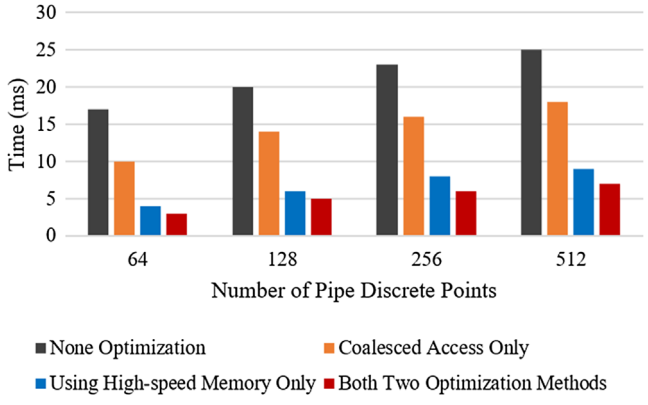

Fig. 11. Comparison of the computing performance solving PDES with and without memory access optimization.

5.1 Small-sized nature gas pipeline network case

5.1.1 Description of pipeline network

First, to evaluate the accuracy of the proposed GPU method, a small-sized gas pipeline network is tested. Figure 12 shows the topological structure of a gas gathering pipeline network in a particular oil field [34]. The pipeline network comprises 73 pipes, 70 junction nodes and 52 boundary nodes. The structure data and parameters of pipes are presented in the Appendix Table B1. The boundary conditions and the components of the natural gas are listed in the Appendix Tables B2 and B3, respectively.

5.1.2 Comparison of numerical accuracy

The steady-state simulation results computed by SPS software, DIMENS and the proposed method are shown in Tables 3 and 4. The total simulation time is 120 h. The time step is $\Delta t = 6$ s and the spatial step $\Delta x = 600$ m, based on which the grid independent numerical solution can be obtained. It should be noted that: (1) the flowrate of the junction nodes represents the net value; (2) the max error represents the maximum value of the error between SPS software and GPU method and that between DIMENS and GPU method.

It can be seen that the maximum relative error of flowrate obtained by the three methods is 4.15% at the junction node 32, as shown in the Table 3; the maximum relative error of pressure obtained by the three methods is 2.88% at the end node of pipe 1, as shown in the Table 4. The maximum relative error is acceptable in numerical simulation and engineering. This comparison indicates good precision of the proposed GPU accelerated method.

5.1.3 Comparison of computing speed

The computing acceleration of the GPU method is compared with that of the SPS software and the DIMENS method with different spatial steps Δx. The results are shown in Table 5.

It can be seen that the SPS software is most time-consuming. The computing time of the DIMENS method is about half of that of SPS software. For example, when the spatial step $\Delta x = 400$ m, the computing time of SPS software and DIMENS method are 5848.29 s and 2735.21 s, respectively. It means that the computational speed of DIMENS method is about 2 times faster than the SPS software, which is consistent with the conclusion of our previous study [11]. Additionally, the computing time of GPU method is much less than those of DIMENS method and SPS software. The computing time of GPU method is about 1/10 and 1/5 of those of the SPS software and DIMENS method. For example, when the spatial step $\Delta x = 400$ m, the computing time of the GPU method is 516.62 s, which is about 1/10 and 1/5 of 5848.29 s and 2735.21 s. It indicates that the computational efficiency of GPU method is about 10 and 5 times than those of SPS software and DIMENS method, respectively. What is more, the speedup ratio of GPU method *vs.* SPS increases from 5.73 to 12.63 as spatial step decreases from 600 m to 200 m, which means the speedup ratio increases as the computation scale increases. It indicates that the GPU method has strong adaptability to the simulation of the pipeline network.

5.2 Large scale natural gas pipeline network cases

In Section 5.1, the calculation accuracy and computing speed of the GPU method have been verified in the small

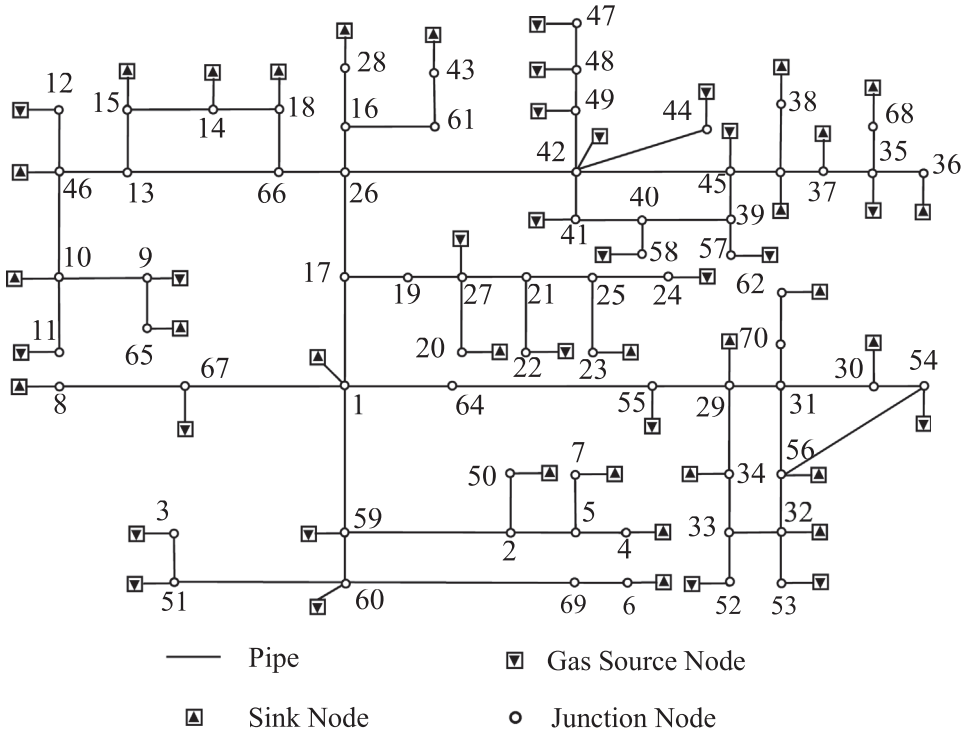

Fig. 12. Topological structure of an actual gas pipeline network.

Table 3. Flowrate (WNm^3/d) obtained by the three methods and their comparison.

Location	SPS	DIMENS	GPU method	Max error (%)
Boundary node 31	241.93	247.81	247.81	2.43
Junction node 32	210.49	222.01	219.22	4.15
Junction node 42	397.5	411.36	405.11	1.91
End node of pipe 1	322.5	331.76	327.14	1.44
Start node of pipe 30	292.56	305.17	300.11	2.58
End node of pipe 43	397.5	411.48	407.28	2.46
End node of pipe 55	320.5	331.36	327.1	2.06

scale pipeline networks. In this subsection, the calculation speed is further tested with large gas pipeline networks.

With the pipeline network shown in Figure 12 as the basic pipeline network unit, we have five pipeline networks, named No. 1 to No. 5, which contain different number of network units and therefore have various sizes. The topological graphs of these five gas pipeline networks are too large to demonstrate in this paper, thus only the component data are shown in Table 6. The main purpose of designing these five pipeline networks is to verify the applicability of the proposed method to the large-scale pipe network. These five networks cases are all transient simulations cases. The simulation time is 1 day, the mesh spatial size and time step are $\Delta x = 500$ m and $\Delta t = 30$ s, respectively.

The computing time of the three methods in the five large pipeline networks is shown in Figure 13. It can be seen from Figure 13 that the computing time of SPS software is the largest one, while that of GPU method is the smallest one. It suggests that the simulation efficiency of the GPU method is high. Additionally, the computing time of the SPS software and DIMENS method linearly depends on the total discrete points, while the computing time of GPU method is almost stable. In other words, when the total discrete points increase, the computing time of the SPS software and DIMENS method linearly increases, but that of the GPU method almost keeps constant. Thus, the GPU method has high computational speed and strong adaptability to the large pipeline networks.

The comparisons of computing time of the three methods in the five large pipeline networks are shown in Table 7. The speedup ratio of DIMENS *vs.* SPS is about 2 in those five large networks, which is consistent with the data in Table 5. It is to say that the computational speed of DIMENS method is also about 2 times faster than the SPS software for the large pipeline network. This proves that DIMENS method is very suitable for large pipe

Table 4. Pressure (MPa) obtained by the three methods and their comparison.

Location	SPS	DIMENS	GPU method	Max error (%)
Boundary node 1	3.74	3.79	3.69	2.51
Boundary node 3	3.89	3.88	3.82	1.93
Boundary node 4	3.81	3.72	3.72	2.57
Junction node 32	3.74	3.75	3.69	1.57
Junction node 42	3.74	3.72	3.69	1.36
Junction node 58	3.94	3.97	3.86	2.80
End node of pipe 1	3.24	3.26	3.17	2.88
Start node of pipe 30	3.63	3.57	3.54	2.53
End node of pipe 43	3.23	3.19	3.15	2.57
End node of pipe 55	3.53	3.59	3.47	3.02

Table 5. Computing time of the three methods and their comparison (small-sized nature gas pipeline network).

Δx (m)	Total discrete points	Computing time (s)			Speedup ratio		
		SPS	DIMENS	GPU method	DIMENS *vs.* SPS	GPU method *vs.* SPS	GPU method *vs.* DIMENS
600	2717	2048	1435.01	357.49	1.43	5.73	4.01
400	4076	5848	2735.21	516.62	2.14	11.32	5.29
200	8151	9479	4124.49	749.48	2.30	12.65	5.50

Table 6. Component data of the five large gas pipeline networks.

Network No.	Number of pipes	Number of junction nodes	Number of boundary nodes	Total discrete points
1	252	88	149	11 974
2	774	229	398	23 566
3	1347	535	715	58 864
4	2531	1288	1186	127 662
5	4128	2143	1865	247 894

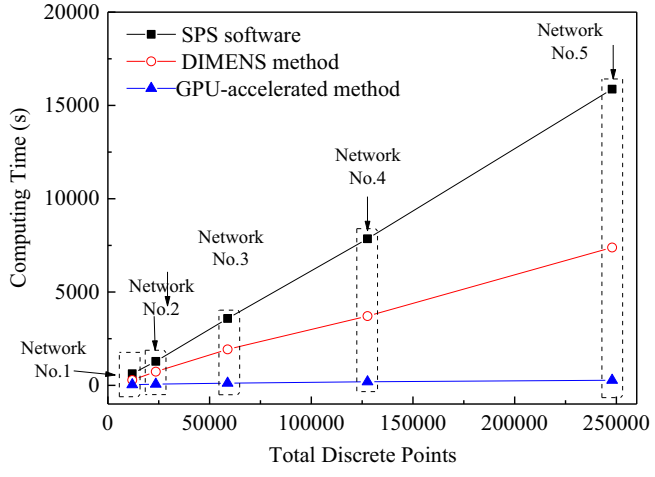

Fig. 13. Computing time of the three methods in the five large pipeline networks.

network, which is consistent with the conclusion of our previous study [17]. In contrast, the speedup ratio of GPU is notable. The minimum speedup ratio of GPU method *vs.* DIMENS and GPU method *vs.* SPS are respectively 7.93 and 17.24, which are much larger than 2. In the case of GPU method *vs.* SPS, the maximum speedup ratio is even up to 57.57. It further confirms that the computational speed of the GPU method is fast.

The speedup ratio of the GPU method in the five large pipeline networks is shown in Figure 14. It can be seen that, the speedup ratio of GPU method *vs.* SPS and GPU method *vs.* DIMENS increase as the total discrete points of pipeline network increase. The speedup ratio of the GPU method approximately linearly depends on the total discrete points. It implies that the larger the pipeline network is, the larger speedup ratio of the GPU method is. It further confirms that the GPU method has strong adaptability to the simulation of the pipeline network, especially for large pipeline network.

Table 7. Comparison of computing time of the three methods in five large nature gas pipeline networks.

No.	Total discrete points	Computing time (s)			Speedup ratio		
		SPS	DIMENS	GPU method	DIMENS *vs.* SPS	GPU method *vs.* SPS	GPU method *vs.* DIMENS
1	11 974	624	287.21	36.21	2.14	17.24	7.93
2	23 566	1286	734.54	64.49	1.75	19.94	11.39
3	58 864	3582	1927.88	118.57	1.86	30.21	16.26
4	127 662	7854	3713.98	191.73	2.11	40.97	19.37
5	247 894	15 873	7386.32	274.89	2.14	57.57	26.87

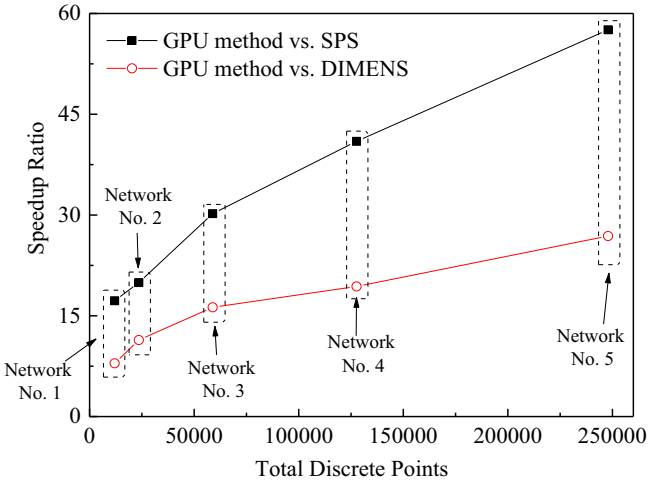

Fig. 14. Speedup ratio of the GPU method in the five large pipeline networks.

6 Conclusion

To further improve the simulation efficiency of natural gas pipeline networks, a novel GPU-accelerated hydraulic simulation was study in this paper. First, based on the DIMENS method in our previous study [16, 17], a two-level pattern for division of the computing tasks and the corresponding parallel numerical method were proposed. Then the thread mapping, memory organization and memory access of GPU were carefully designed, and the two-level parallel process was implemented on the GPU. Last, one small and five large nature gas pipeline networks were presented to verify the accuracy and efficiency of the proposed GPU method. The following conclusions can be drawn:

1. The proposed GPU method has high accuracy. The accuracy of the result obtained by the GPU method is almost the same with the DIMENS method and the SPS software. In the small nature gas pipeline network case, the maximum relative error of results obtained by the above three methods is only 4.15%.
2. The proposed GPU method has notable speedup. In the five large-scale pipe networks cases, compared with the well-known commercial simulation software SPS, the speedup ratio of the proposed method is up to 57.57 with comparable calculation accuracy.
3. The proposed method has strong adaptability to the large pipeline networks. In the five large-scale pipe networks cases, the larger the pipeline network is, the larger speedup ratio of the proposed method is. The speedup ratio of the GPU method approximately linearly depends on the total discrete points of the network.

Acknowledgments. The study is supported by the *National Natural Science Foundation of China* (No. 51806018), the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (No. IDHT20170507) and the Program of Great Wall Scholar (No. CIT&TCD20180313).

References

1 Brentner K.S., Farassat F. (1998) Analytical comparison of the acoustic analogy and Kirchhoff formulation for moving surfaces, *AIAA J.* **360**, 8, 1379–1386. https://doi.org/10.2514/2.558.
2 Bai H. (2020) Mechanism analysis, anti-corrosion techniques and numerical modeling of corrosion in energy industry, *Oil Gas Sci. Technol. - Rev IFP Energies nouvelles* **75**, 42.
3 Shi J., Al-Durra A., Matraji I., Al-Wahedi K., Abou-Khousa M. (2019) Application of Particle Swarm Optimization (PSO) algorithm for Black Powder (BP) source identification in gas pipeline network based on 1-D model, *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles* **74**, 47.
4 Su H., Zio E., Zhang J., Yang Z., Li X., Zhang Z. (2018) A systematic hybrid method for real-time prediction of system conditions in natural gas pipeline networks, *J. Nat. Gas Sci. Eng.* **57**, 31–44.
5 Sanaye S., Mahmoudimehr J. (2012) Technical assessment of isothermal and non-isothermal modelings of natural gas pipeline operational conditions, *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles* **67**, 3, 435–449.
6 Qiao W., Huang K., Azimi M., Han S. (2019) A novel hybrid prediction model for hourly gas consumption in supply side based on improved whale optimization algorithm and relevance vector machine, *IEEE Access* **7**, 88218–88230.
7 Wang P., Yu B., Deng Y., Zhao Y. (2015) Comparison study on the accuracy and efficiency of the four forms of hydraulic equation of a natural gas pipeline based on linearized solution, *J. Nat. Gas Sci. Eng.* **22**, 235–244.
8 Bagajewicz M., Valtinson G. (2014) Computation of natural gas pipeline hydraulics, *Ind. Eng. Chem. Res.* **53**, 26, 10707–10720.

9 Streeter V.L., Wylie E.B. (1970) Natural gas pipeline transients, *Soc. Pet. Eng. J.* **10**, 04, 357–364.
10 Osiadacz A.J. (1996) Different transient flow models-limitations, advantages, and disadvantages, in: *PSIG Annual Meeting*, PSIG-9606, San Francisco, California.
11 Yow W. (1971) *Analysis and control of transient flow in natural gas piping systems*. http://hdl.handle.net/2027.42/8473.
12 Osiadacz A. (1984) Simulation of transient gas flows in networks, *Int. J. Numer. Meth. Fl.* **4**, 1, 13–24.
13 Wylie E.B., Stoner M.A., Streeter V.L. (1971) Network: System transient calculations by implicit method, *Soc. Pet. Eng. J.* **11**, 04, 356–362.
14 Luongo C.A. (1986) An efficient program for transient flow simulation in natural gas pipelines, in: *PSIG Annual Meeting*, PSIG-8605, New Orleans, Louisiana.
15 Behbahani-Nejad M., Shekari Y. (2008) Reduced order modeling of natural gas transient flow in pipelines, *Int. J. Eng. Appl. Sci.* **5**, 7, 148–152.
16 Wang P., Yu B., Han D., Li J., Sun D., Xiang Y., Wang L. (2018) Adaptive implicit finite difference method for natural gas pipeline transient flow, *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles* **73**, 21.
17 Wang P., Yu B., Han D., Sun D., Xiang Y. (2018) Fast method for the hydraulic simulation of natural gas pipeline networks based on the divide-and-conquer approach, *J. Nat. Gas Sci. Eng.* **50**, 55–63.
18 Wang P., Ao S., Yu B., Han D. (2019) An efficiently decoupled implicit method for complex natural gas pipeline network simulation, *Energies* **12**, 8, 1516.
19 Thibault J.C. (2009) *Implementation of a Cartesian grid incompressible Navier-Stokes solver on multi-GPU desktop platforms using CUDA*, Boise State University, Boise, Idaho.
20 Lacasta A., Morales-Hernández M., Murillo J., García-Navarro P. (2014) An optimized GPU implementation of a 2D free surface simulation model on unstructured meshes, *Adv. Eng. Softw.* **78**, 1–15.
21 Tölke J., Krafczyk M. (2008) TeraFLOP computing on a desktop PC with GPUs for 3D CFD, *Int. J. Comput. Fluid. D.* **22**, 7, 443–456.
22 Michalakes J., Vachharajani M. (2008) GPU acceleration of numerical weather prediction, *Parallel Process. Lett.* **18**, 04, 531–548.
23 Acuña M., Aoki T. (2009) Real-time tsunami simulation on multi-node GPU cluster, in: *ACM/IEEE Conference on Supercomputing*.
24 Lu X., Han B., Hori M., Xiong C., Xu Z. (2014) A coarse-grained parallel approach for seismic damage simulations of urban areas based on refined models and GPU/CPU cooperative computing, *Adv. Eng. Softw.* **70**, 90–103.
25 Cai Y., Li G., Wang H., Zheng G., Lin S. (2012) Development of parallel explicit finite element sheet forming simulation system based on GPU architecture, *Adv. Eng. Softw.* **45**, 1, 370–379.
26 Zheng J., Song F., Chen G. (2011) Development of RealPipeGas simulation software for gas pipeline network, *Oil Gas Storage Transp.* **30**, 9, 659–662. (in Chinese).
27 Luskin M. (1979) An approximation procedure for nonsymmetric, nonlinear hyperbolic systems with integral boundary conditions, *Siam J. Numer. Anal.* **16**, 1, 145–164.
28 Kiuchi T. (1994) An implicit method for transient gas flows in pipe networks, *Int. J. Heat. Fluid Fl.* **15**, 5, 378–383.
29 Abbaspour M., Chapman K.S. (2008) Nonisothermal transient flow in natural gas pipeline, *J. Appl. Fluid Mech.* **75**, 3, 519–525.
30 Zhang T., Li Y., Li Y., Sun S., Gao X. (2020) A self-adaptive deep learning algorithm for accelerating multi-component flash calculation, *Comput. Method. Appl. M.* **369**, 113207.
31 Li R., Saad Y. (2013) GPU-accelerated preconditioned iterative linear solvers, *J. Supercomput.* **63**, 2, 443–466.
32 Hockney R.W., Jesshope C.R. (2019) *Parallel computers 2: Architecture, programming and algorithms*, CRC Press, Boca Raton, FL.
33 Harris M. (2013) *How to access global memory efficiently in CUDA C/C++ kernels*. http://devblogs.nvidia.com/parallelforall/how-access-global-memory-efficientlycuda-c-kernels.
34 Yu B., Wang P., Wang L., Xiang Y. (2017) A simulation method for natural gas pipeline networks based on the divide-and-conquer concept, *Oil Gas Storage Transp.* **36**, 1, 75–84. (in Chinese).

Appendix A

The detailed expressions of **G** and **S** in equation (2) are as presented follows,

$$\mathbf{G} = \begin{bmatrix} \frac{1}{A}\frac{\partial}{\partial p}\left(\frac{\partial p}{\partial \rho}\right)_T \frac{\partial m}{\partial x} & 0 \\ \left\{\begin{array}{c} \frac{m^2}{A}\left[\frac{2}{\rho^3}\left(\frac{\partial \rho}{\partial p}\right)_T^2 - \frac{1}{\rho^2}\frac{\partial}{\partial p}\left(\frac{\partial \rho}{\partial p}\right)_T\right]\frac{\partial p}{\partial x} \\ -\frac{2m}{A\rho^2}\left(\frac{\partial \rho}{\partial p}\right)_T \frac{\partial m}{\partial x} \end{array}\right\} & -\frac{2m}{A\rho^2}\left(\frac{\partial \rho}{\partial p}\right)_T \frac{\partial p}{\partial x} + \frac{2}{A\rho}\frac{\partial m}{\partial x} \end{bmatrix}, \tag{A1}$$

$$S = \begin{bmatrix} \frac{\partial T}{\partial t}\frac{\partial}{\partial p}\left(\frac{\partial p}{\partial T}\right)_\rho & 0 \\ \left\{\begin{array}{c} \frac{\lambda}{2}\frac{m|m|}{dA\rho^2}\left(\frac{\partial \rho}{\partial p}\right)_T - Ag\sin\theta\left(\frac{\partial \rho}{\partial p}\right)_T \\ +\frac{m^2}{A}\frac{\partial T}{\partial x}\frac{\partial}{\partial p}\left(\frac{1}{\rho^2}\left(\frac{\partial \rho}{\partial T}\right)_p\right) \end{array}\right\} & -\lambda\frac{|m|}{dA\rho} + \frac{2m}{A\rho^2}\left(\frac{\partial \rho}{\partial T}\right)_p \frac{\partial T}{\partial x} \end{bmatrix}. \tag{A2}$$

Appendix B

Table B1. Structure data and parameters of pipes.

Pipe	1	2	3	4	5	6	7	8	9	10	11
Start node	64	3	5	51	5	69	5	67	38	9	46
End node	1	2	2	3	4	6	7	8	63	10	10
Length	22	8	16	3	3.5	22	9	18	5	10	5
Diameter	610	325	273	273	273	273	273	508	325	273	325
Thickness	7.1	6	7	5.6	7	5.6	5.6	8	7	7	7
Pipe	12	13	14	15	16	17	18	19	20	21	22
Start node	11	13	66	13	14	1	17	66	18	19	27
End node	10	46	13	15	15	17	26	18	14	27	20
Length	9	7	2	20	3	20	16	3	5	41	9.6
Diameter	426	325	325	325	273	610	610	325	426	325	325
Thickness	12	7	7	7	5	7.1	7.1	6	7	7.1	6
Pipe	23	24	25	26	27	28	29	30	31	32	33
Start node	27	22	25	25	24	26	55	31	31	32	32
End node	21	21	21	23	25	28	64	29	30	56	33
Length	23	3	18	3	25	135	22	46	14.5	50	30
Diameter	273	168	273	273	273	508	610	610	457	610	610
Thickness	7	4.5	7	7	7	7.1	7.1	7.1	6.3	7.1	7.1
Pipe	34	35	36	37	38	39	40	41	42	43	44
Start node	33	52	35	37	38	45	45	39	40	42	16
End node	34	33	36	35	37	38	39	40	41	26	61
Length	7	10	2	29.1	44.7	69.2	44.5	17.8	7.6	115.7	2
Diameter	219	610	610	610	610	610	610	610	610	610	273
Thickness	4	7.1	8	7.1	7.1	7.1	7.1	7.1	7.1	7.1	5
Pipe	45	46	47	48	49	50	51	52	53	54	55
Start node	44	47	48	49	2	41	45	12	53	54	29
End node	42	48	49	42	50	42	42	46	32	30	55
Length	56.6	19.5	23.5	24	23.5	72	142	3	8.7	14.5	52
Diameter	406.4	355	355.6	273	219	610	273	426	813	457	610
Thickness	6.3	6.5	6.5	5.6	6	7.1	5	7	9.5	6.3	7.1
Pipe	56	57	58	59	60	61	62	63	64	65	66
Start node	31	57	58	59	60	60	35	69	54	31	66
End node	56	39	40	1	59	51	68	51	32	70	26
Length	50	10	8.8	13	6	14	36	15.9	42	2	1
Diameter	610	219	508	273	273	273	355.6	273	610	610	273
Thickness	8	7	7	7	7	5.6	7.1	5.6	7.1	7.1	6
Pipe	67	68	69	70	71	72	73				
Start node	26	70	17	26	1	9	34				
End node	65	62	19	16	67	65	29				
Length	1	1	1	5	1	5	2				
Diameter	273	273	273	325	273	273	273				
Thickness	6	6	6	7	6	6	6				

Note: The unit of length, diameter and thickness are km, mm and WNm^3/d, respectively.

Table B2. Structure data and boundary conditions of boundary nodes.

Boundary node	1	2	3	4	5	6	7	8	9	10	11	12	13
Connected node	44	53	45	47	48	49	51	3	35	52	42	11	12
Flowrate	7	20	25	22	6	8	4	5	38	230	55	68	32
Boundary node	14	15	16	17	18	19	20	21	22	23	24	25	26
Connected node	54	55	22	24	9	57	58	41	59	60	27	67	6
Flowrate	220	2	2.5	0.8	4	2	300	4	21	5.6	12	5.2	−60
Boundary node	27	28	29	30	31	32	33	34	35	36	37	38	39
Connected node	7	8	23	46	28	20	50	4	63	29	1	43	30
Flowrate	−34	−22	−13	−57	–	−23.7	−14	−68	−3.5	−22	−33	−44	−21
Boundary node	40	41	42	43	44	45	46	47	48	49	50	51	52
Connected node	34	36	63	10	15	18	14	62	68	32	56	37	38
Flowrate	−25	−17	−26	−56	−29.4	−63	−58.6	−66	−37	−15	−0.5	−9	−3

Note: The boundary node No. 31 is a pressure controlled node, so the working condition indicates its pressure value with a unit of MPa. The rest nodes are flowrate-controlled, and the unit of their value of working condition is WNm^3/d.

Table B3. Main components of natural gas.

Components	CH_4	C_2H_6	C_3H_8	N_2	CO_2
Volume fraction (%)	99.79	0.07	0.02	0.07	0.05