

SPH 并行计算方案及其在自由表面流动模拟中的应用*

周浩¹, 徐志宏¹, 邹顺², 汤文辉¹

(1. 国防科技大学理学院, 湖南长沙 410073; 2. 国防科技大学计算机学院, 湖南长沙 410073)

摘要:针对光滑粒子动力学主要计算量是近邻粒子搜索这一特点,提出了一种基于粒子分解的光滑粒子动力学并行计算方案。利用该方案可以方便地将任意串行光滑粒子动力学代码并行计算,而且每一个时间步内的信息传递量只和粒子总数有关,而和粒子的分布无关,因而特别适合于自由表面流动等大变形问题的并行数值模拟。对一个粒子总数为40万的三维溃坝问题的模拟结果表明:此方案能达到的最大加速比约为16,这一结果可能比空间分解方案(不考虑动态负载均衡)更优。

关键词:光滑粒子动力学;并行方案;粒子分解;空间分解

中图分类号: TB126 **文献标识码:** A **文章编号:** 1001-2486(2015)04-071-06

SPH parallel schemes and its application in free surface flow simulation

ZHOU Hao¹, XU Zhihong¹, ZOU Shun², TANG Wenhui¹

(1. College of Science, National University of Defense Technology, Changsha 410073, China;

2. College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: For the main computation of smoothed particle hydrodynamics was finite nearest particle search, a novel scheme to parallelize the smoothed particle hydrodynamics method based on the concept of particle decomposition was proposed. Any serial smoothed particle hydrodynamics code could be easily parallelized by using the proposed scheme. The amount of information, which was transformed in each time step, depended only on the total number of particles, but not on the spatial distribution of particles. Therefore, the proposed scheme was particularly useful for the parallel simulation of cases involved violent free surface movements. From the simulation results of a 3D dam break case with a total number of 0.4 million, the proposed scheme can achieve a speedup ratio about 16, which proves that the proposed scheme maybe is better than the domain decomposition scheme (without considering dynamic load balance).

Key words: smoothed particle hydrodynamics; parallel scheme; particle decomposition; domain decomposition

多数数值方法在处理大变形时存在一定困难。光滑粒子动力学 (Smoothed Particle Hydrodynamics, SPH)^[1-2]方法作为一种无网格的拉格朗日方法,在处理自由表面流动等大变形问题时具有很大的优势,近年来其应用已经扩展到很多领域。又因为它的拉格朗日特性,界面附近物质的运动即代表了界面的运动,因此处理多相流也比较自然,且不需要额外的界面追踪算法。但是,SPH方法比有限差分、有限元等网格方法的计算量要大,因此在模拟规模较大的问题时经常需要并行计算。

SPH的计算过程和分子动力学 (Molecular Dynamics, MD)类似,所以其并行也和MD类似。Plimpton^[3]总结了3种不同的MD并行方案,即空

间分解,粒子分解和力分解。目前,文献中绝大多数SPH方法并行采用的都是空间分解方案,因为其信息传递量小,其缺点是编程实现复杂。

空间分解方案又可以分为静态和动态两种。静态空间分解时,每个处理器分配的空间始终固定。当所有粒子始终在计算域近似均匀分布时,这种方法能够得到极高的加速比。但是,在自由表面流动等实际问题中,粒子往往在空间中分布非常不均匀,这会使得负载严重不平衡,导致加速比较小。动态空间分解时,需要根据负载均衡原则每隔一段时间动态调整各个处理器空间的大小和位置,来保证所有处理器的计算量大致相等。调整处理器空间需要额外的计算时间,因此动态调整的频率要适当,而且调整算法也必须高效。

* 收稿日期:2014-07-08

基金项目:国家自然科学基金资助项目(61221491,61303071),广州科信局基金资助项目(134200026)

作者简介:周浩(1986—),男,湖南常德人,博士研究生, E-mail: zhouhao12029014@163.com;

汤文辉(通信作者),男,教授,博士,博士生导师, E-mail: wenhuitang@163.com

动态空间分解虽然能够高效地模拟大变形问题,但是实现起来非常复杂。如 Ferrari 等^[4]在采用并行 SPH 方法模拟三维溃坝问题时,采用第三方软件包 METIS 来动态调整处理器空间。在 METIS 中,首先需要在空间布满背景网格,再根据每个小网格中粒子数的多少来给每个小网格一个权重,来实现每个处理器空间大小不等但是计算量近似相等,用以实现负载平衡。Marrone 等^[5]采用并行 SPH 方法模拟行船产生的波浪及其破碎时,只在主要流动方向上动态调整,因为这能极大地简化编程,并且指出存在主要流动方向的特征越明显,并行模拟的加速比越高。并行 SPH 程序 JOSEPHINE^[6]和开源软件 parallelSPHYSICS v2.0 在模拟二维自由表面流动时,也都只在一个主要流动方向上实现了动态调整,因此,当流动有多个主要方向时,加速比会下降。总的来说,动态空间分解的实现复杂度严重限制了其在 SPH 并行模拟中的广泛使用。

Plimpton 描述了 MD 的粒子分解并行方案,其基本思想是每个处理器负责维护固定的某一部分粒子。这种方案的最大优点在于并行很容易在串行代码的基础上实现^[3]。但是,由于 MD 中的分子由若干原子组成,分子具有一定内部结构,需要考虑其旋转等因素,而且与每个分子有相互作用的分子列表可以若干时间步更新一次,所以其主要计算量在于分子间相互作用力的计算。而 SPH 中粒子间的相互作用力相对简单,而且一般每个时间步都需要更新每个粒子的近邻粒子,因此其主要计算量在于近邻粒子搜索^[7]。

针对 SPH 方法的上述特点,提出了一种简单的粒子并行方案。这种方案能够达到可观加速比,有效节约计算时间,而且方案实现非常简单,负载均衡,每个时间步的信息传递量与粒子分布无关,可以十分方便地采取任意奇数个处理器并行计算。但是,这种方案的信息传递量较大,不适合上百个处理器的大规模并行计算。

以二维圆柱绕流和三维溃坝问题为例,比较了这种粒子并行方案和空间分解方案(不考虑动态负载均衡)的计算时间或加速比。在这两个算例中,前者粒子分布始终非常均匀,而后者则是典型的大变形问题,粒子分布严重不均匀。这两个算例的对比可以明显地展示出粒子并行方案只适合于粒子分布严重不均匀问题的并行模拟。SPH 静态空间分解方案在开源软件平台 OpenFOAM 的基础上实现,粒子分解方案采用 Fortran 90 语言加上讯息传递接口(Message Passing Interface,

MPI)消息传递库实现。本文所有模拟都是在广州超算中心的天河二号计算机上完成。

1 控制方程及其 SPH 离散

假设系统中没有温度梯度,黏性流动的控制方程为如下 N-S 方程

$$\frac{d\rho}{dt} = -\rho \nabla \cdot u \quad (1)$$

$$\frac{du}{dt} = \frac{-1}{\rho} \nabla p + \nu \nabla^2 u + f^e \quad (2)$$

其中: ρ, u, p, t 分别为密度、速度、压力和时间; $\nu \nabla^2 u$ 为黏性力; ν 为运动黏性系数; f^e 为外力。控制方程式(1)和式(2)右边各导数项的 SPH 离散形式如下:

$$(-\rho \nabla \cdot u)_i = \sum_{j=1}^N m_j (u_i - u_j) \cdot \Delta_i W_{ij} \quad (3)$$

$$\left(\frac{-1}{\rho} \nabla p\right)_i = \sum_{j=1}^N m_j \left(\frac{p_j}{\rho_j} + \frac{p_i}{\rho_i}\right) \Delta_i W_{ij} \quad (4)$$

$$(\nabla^2 u)_i = \sum_{j=1}^N \frac{m_j (\rho_i + \rho_j) r_{ij} \cdot \Delta_i W_{ij}}{\rho_i \rho_j} (u_i - u_j) \quad (5)$$

其中,式(5)为 Morris 基于有限差分法和 SPH 离散提出的黏性力计算模型^[8]。本文模拟中时间积分采用蛙跳法。

对不可压缩流体采用弱可压来近似,通过将流体的密度变化控制在 1% 以内来近似实现不可压^[9]。本文中采用如下最常用的物态方程

$$p = \frac{\rho_0 c_0^2}{\gamma} \left[\left(\frac{\rho}{\rho_0}\right)^\gamma - 1 \right] \quad (6)$$

其中, $\gamma = 7, c_0$ 取粒子最大速度的 10 倍。

2 SPH 并行方案

现阶段文献中 SPH 并行绝大多数采用的是空间分解方案。本文实现并比较了空间分解和粒子分解这两种并行方案各自的优缺点。

2.1 空间分解方案

空间分解方案是目前使用最普遍的方案,其基本思想就是首先将整个计算域划分为若干不重叠的子域,然后每个处理器负责一个子域,如图 1 所示。

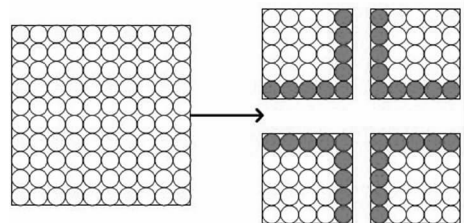


图 1 空间分解示意图

Fig. 1 Domain decomposition

当计算满足局域性(任何一点在下一时刻的值只和包括这点的某局部空间相关)时,只有在子域之间的交界处才需要传递信息,如图1中的实心粒子部分。因此,划分子域的一个原则就是让交界面积尽量小,这代表了信息传递少,即并行带来的额外时间开销小。另一个原则是尽量保持每个子域的计算量相等,即负载均衡,这样才能达到较高加速比。

在欧拉类网格方法中,往往是网格的数量代表了计算量。由于网格是不动的,所以负载均衡比较容易实现。但是 SPH 方法属于拉格朗日方法,粒子数量大致代表了计算量,大变形条件下,粒子占据的空间可能发生很大变化,如模拟溃坝、气体扩散、超高速碰撞等问题。这会导致每个子域的粒子数量严重不平衡,导致并行效率低下。一种做法是采取动态负载均衡,即模拟过程中如果发现负载不均衡,就按照一定策略将空间重新分解,如开源软件 parallelSPHYSICS v2.0 的使用手册中给出的二维溃坝问题的模拟结果,其动态负载均衡如图2所示。

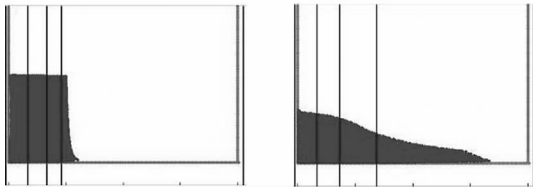


图2 动态负载均衡

Fig. 2 Dynamic load balancing

图2中有3个垂直方向的处理器边界,它们将整个计算域划分为4个子域,每个子域用一个处理器计算。当粒子整体向右移动时,每个处理器边界也适当向右移动,来尽量保证每个子域的粒子数相等。

动态空间分解虽然能够解决负载不均衡问题,但是实现起来复杂,所以本文不考虑动态空间分解。

2.2 粒子分解方案

本文针对 SPH 方法的主要计算量在于近邻粒子搜索这一特点,提出并实现了一种新的粒子分解方案,其基本思想是将找出所有近邻粒子对这样一个大任务划分为 np (处理器数) 个小任务,每个处理器完成一个。

首先将所有粒子依据编号从小到大分为 np 个组 G_m ($m = 1, 2, \dots, np$), 每个组的粒子数近似相等,约为 N/np (其中 N 为粒子总数), 处理器 m 维护组 G_m 。当粒子 i 和粒子 j 是近邻粒子时,可

以用 (i, j) 来表示这一近邻粒子对,为了避免重复,因此要求 $i < j$ 。 (G_m, G_n) 表示所有近邻粒子对的某个子集,定义为 $(G_m, G_n) = \{(i, j) | i \in G_m, j \in G_n\}$, 由 $i < j$ 可知 $m \leq n$ 。显然,当 $np = 1$ 时, (G_1, G_1) 就是所有近邻粒子对的集合。一个子集就代表了部分近邻粒子对,需要分配于不同的处理器。

由于 $m \leq n$, 可知子集 (G_m, G_n) 有 $np(np + 1)/2$ 个,且相互不重叠。每个处理器负责 $(np + 1)/2$ 个。显然,当 np 为奇数时, $(np + 1)/2$ 为整数,此时,每个处理器的负载非常均衡。下面以 $np = 5$ 为例来说明这些子集在处理器上的分配策略,如表1所示。

表1 近邻粒子对在不同处理器之间分配策略

Table 1 Distribution of neighbor particle pairs on different processors

处理器	P_1	P_2	P_3	P_4	P_5
任务	(G_1, G_1)	(G_2, G_2)	(G_3, G_3)	(G_4, G_4)	(G_5, G_5)
	(G_1, G_2)	(G_2, G_3)	(G_3, G_4)	(G_4, G_5)	(G_1, G_5)
	(G_1, G_3)	(G_2, G_4)	(G_3, G_5)	(G_1, G_4)	(G_2, G_5)

由表1可以看到,处理器 P_1 负责维护 G_1 这组粒子。在每个时间步长内, P_1 首先需要将 G_1 组粒子的所有信息发送给 P_4 和 P_5 , 因为 P_4 需要找出近邻粒子对子集 (G_1, G_4) , P_5 需要找出 (G_1, G_5) ; 然后 P_1 搜索被分配的近邻粒子对(本文采用树搜索算法)并同时计算相互作用力;最后 P_4 和 P_5 将所计算的 G_1 组粒子受到的部分作用力发送回 P_1 , 由 P_1 更新 G_1 这组粒子的状态。

处理器 P_2 的计算过程类似,即 P_2 首先将 G_2 组粒子的所有信息发送给 P_1 和 P_5 ; 然后寻找被分配的近邻粒子对并同时计算相互作用力;最后 P_1 和 P_5 将所计算的 G_2 组粒子受到的部分作用力发送回 P_2 , 由 P_2 更新 G_2 这组粒子的状态。其他处理器的计算过程也类似。

于是,每个时间步可以分为如下4步:粒子信息发送、近邻粒子搜索并同时计算粒子间相互作用力、相互作用力信息交换和更新粒子状态。分别统计每步的计算时间,可以得到:第1步和第3步为信息传递,时间随着处理器的增加而缓慢增加;第2步是主要计算量,随着处理器的增加而单调减小;第4步也是计算,时间几乎可以忽略不计^[10]。

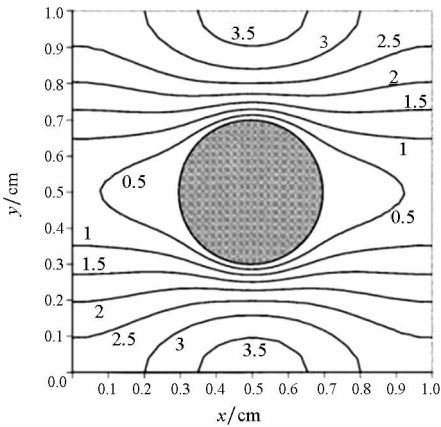
可以看到,本粒子分解方案有如下特点:①—

个时间步长内的所有通信为每个处理器调用一次广播函数和一次求和归约函数,实现非常简单;②每个处理器的计算量和通信量都大致相等,负载非常均衡;③每步的信息传递量为 $O(N)$,与粒子的分布无关;④可以十分方便地采取任意奇数个处理器并行计算。

3 两种并行方案的比较

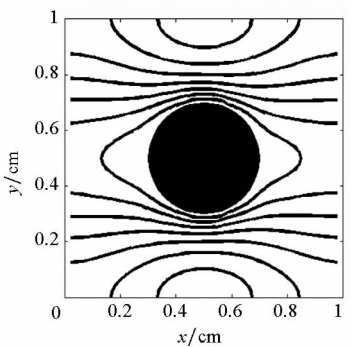
3.1 二维圆柱绕流

模型参数与文献[8]中的圆柱绕流算例一致,雷诺数为 0.03。初始时, X 方向和 Y 方向各均匀分布 1000 个粒子,总粒子数为 100 万。 X 方向为周期性边界条件,对应的实际物理问题为无限长管道中均匀分布无数个圆柱。图 3 给出了 SPH 模拟结果和有限元模拟结果的对比,其中图 3(a)为文献[8]中的有限元模拟结果,图 3(b)为本文模拟结果。



(a) 有限单元法

(a) Finite Element Modeling



(b) 光滑粒子动力学

(b) Smoothed particle hydrodynamics

图 3 SPH 和有限元模拟的速度大小等值线对比

Fig. 3 Contour lines of velocity magnitude using FEM and SPH for $Re = 0.03$

分别采取两种并行方案,测得的加速比如图 4 所示。

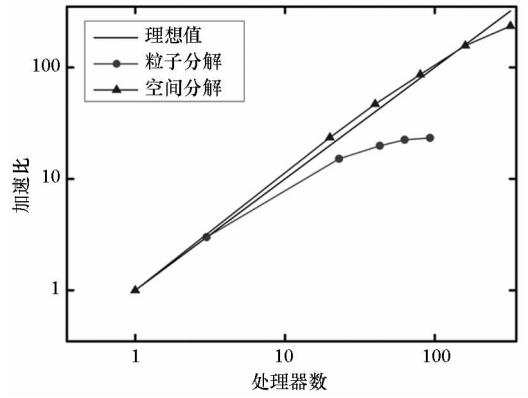


图 4 空间分解和粒子分解加速比随处理器的变化
Fig. 4 Speed-up ratio of domain decomposition and particle decomposition with the change of processors

从图 4 可以看到,空间分解时,当处理器数目在小于 100 时,加速比反常地略高于理想值,其原因可能和缓存等硬件有关,文献[4-5]以及开源软件 parallelSPHYSICS v2.0 使用手册中都报告了类似的情况。当处理器数目为 320 时,加速比约为 234.69,并行效率为 73.34%。如果继续增加处理器数至 640,加速比迅速降低为 100.13,对应的并行效率为 15.65%,因此结果没有在图 4 中给出。

粒子分解时,最多采用了 93 个处理器,此时的加速比为 23.33,并行效率为 25.09%。如果继续增加处理器数量,加速比则缓慢下降。

因此,对于粒子分布均匀的情况,空间分解的加速比远大于粒子分解。而且可以发现,粒子分解不适合上百个处理器的大规模并行计算。

3.2 三维溃坝问题

模型参数与文献[11]中的溃坝算例一致,初始水柱高 0.292m,宽 0.146m,容器宽度为 0.584m。初始粒子总数大约为 40 万,时间步长固定为 $20\mu s$,共计算了 50 000 个时间步长。图 5 为文献[11]给出的二维流体体积法 (Volume Of Fluid, VOF) 模拟结果,从左到右的时刻依次为 0.2s, 0.4s, 0.6s 和 0.8s。



图 5 溃坝问题的二维模拟结果^[10]

Fig. 5 2D VOF simulation of dam break

本文采用静态空间分解和粒子分解两种不同的并行方案对同样的问题进行了三维数值模拟。模拟的粒子分布如图 6 所示。

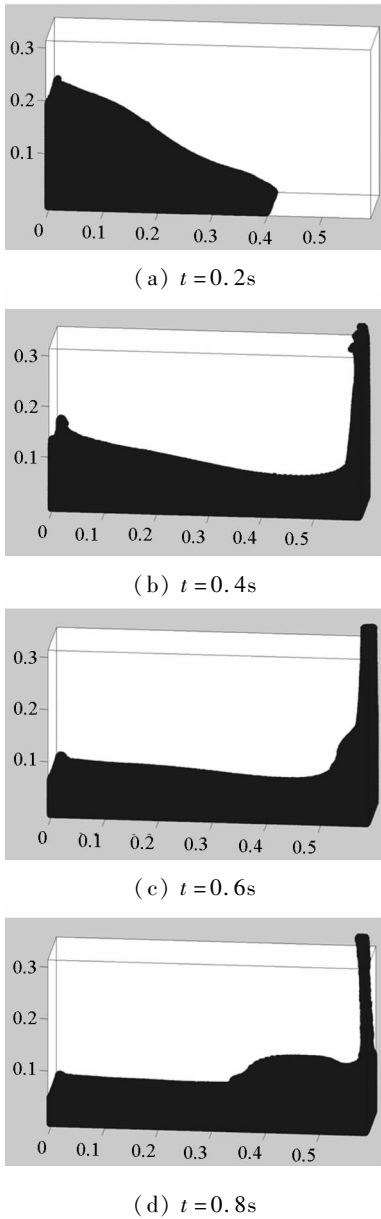


图6 三维溃坝问题的 SPH 模拟
Fig. 6 3D SPH simulation of dam break

可以看到,三维模拟结果和文献[11]中的实验及 VOF 模拟结果基本一致。为了和二维模拟结果对比,图7给出了0.8s时刻粒子分布的一个二维视图。

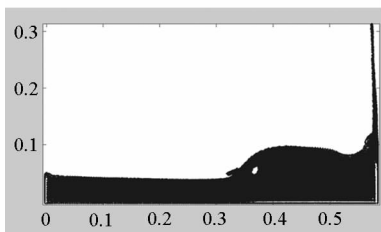


图7 0.8s时刻的二维视图
Fig. 7 2D view point at $t = 0.8s$

从图7可以看到,流体在0.8s时刻形成了一

个空腔,这和 VOF 计算结果吻合。

因为容器在 X, Y, Z 方向上的尺寸约为 1:4:2,因此在空间分解时, X, Y, Z 方向上的处理器个数取为 $n, 4n, 2n$,从而让每个处理器分配的空间大小近似相等,总处理器个数为 $8n^3$ 。但是由于粒子在整个容器中的分布极不均匀,每个处理器上分配的粒子数实际上极不均匀,甚至多数处理器上没有粒子。本算例中,空间分解时处理器个数分别取为 8,64,216,分别对应于 $n = 1, 2, 3$ 。

两种并行方案测得的计算时间如图8所示。

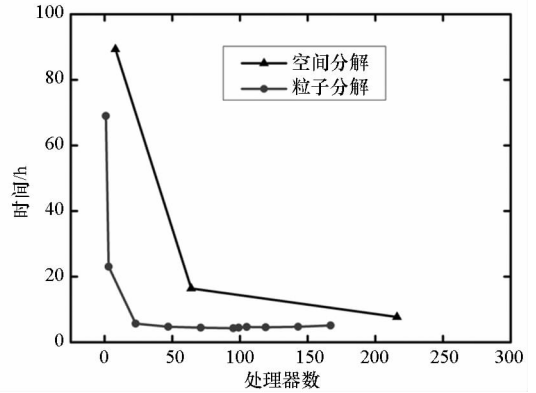


图8 空间分解和粒子分解计算时间随处理器的变化

Fig. 8 Total computing time of domain decomposition and particle decomposition with the change of processors

可以看到,空间分解的计算时间比粒子分解要长,而且计算时间随处理器数的增加缓慢下降。当处理器数为 216 时,空间分解需要约 7.7h,加速比为 9.0。此处加速比定义为 $speedup = T_1/T_n$,其中 T_n 为采用 n 个处理器并行计算的时间, T_1 为串行计算时间。由于空间分解程序的计算时间较长,没有测试其串行计算时间,所以其串行计算时间采用粒子分解程序的串行时间。计算中还发现,当处理器数为 216 时,绝大多数处理器中没有粒子,这是由溃坝问题中粒子分布不均匀的特点以及没有动态负载均衡造成的。

粒子分解时,当处理器数为 23 时,计算时间为 5.7h,几乎已经降到最低值,且计算时间比空间分解方案的最优值少。当处理器数为 95 时,需要的时间最少,为 4.3h,最大加速比为 16.1。如果继续增加处理器数,计算时间缓慢增加。

4 结论

本文提出了一种非常简单的基于粒子分解的 SPH 并行方案,并以二维圆柱绕流和三维溃坝问题的 SPH 并行模拟为例,比较了粒子分解和空间

分解的计算时间或加速比,得到了以下主要结论:

1)当粒子分布均匀时,SPH 并行应该采用空间分解方案。此时,只有处理器边界处有信息传递,信息传递量小,能够达到非常大的加速比。而粒子分解的加速比相对较小。

2)当粒子分布非常不均匀时,SPH 并行可以采用粒子分解方案。此时,静态空间分解的负载严重不平衡,导致加速比较小。而粒子分解采用较少处理器就能达到可观加速比。

3)从文中两个算例可以看出,当问题规模在 40 万~100 万左右时,粒子分解方案的加速比固定在 16~23 左右。在粒子分布非常不均匀的情况下,这一结果可能比静态空间分解更优。因而,本文提出的粒子分解方案在自由表面流动模拟中具有一定优势。

参考文献 (References)

[1] Lucy L B. A numerical approach to the testing of the fission hypothesis[J]. *Astronomical Journal*, 1977, 82: 1013 - 1024.

[2] Monaghan J J, Gingold R A. Shock simulation by the particle method SPH [J]. *Journal of Computational Physics*, 1983, 52(2): 374 - 389.

[3] Plimton S. Fast parallel algorithms for short-range molecular dynamics [J]. *Journal of Computational Physics*, 1995,

117(1): 1 - 19.

[4] Ferrari A, Dumbser M, Toro E F, et al. A new 3D parallel SPH scheme for free surface flows [J]. *Computers & Fluids*, 2009, 38(6): 1203 - 1217.

[5] Marrone S, Bouscasse B, Colagrossi A, et al. Study of ship wave breaking patterns using 3D parallel SPH simulations [J]. *Computers & Fluids*, 2012, 69: 54 - 66.

[6] Cherfils J M, Pinon G, Rivoalen E. JOSEPHINE: a parallel SPH code for free-surface flows [J]. *Computer Physics Communication*, 2012, 183(7): 1468 - 1480.

[7] Moulinec C, Issa R, Latino D, et al. High-performance computing and smoothed particle hydrodynamics [C]// *Proceedings of Parallel Computational Fluid Dynamics*, Lyon, 2008.

[8] Morris J P, Fox P J, Zhu Y. Modeling low reynolds number incompressible flows using SPH[J]. *Journal of Computational Physics*, 1997, 136(1): 214 - 226.

[9] Monahan J J. Simulating free surface flows with SPH[J]. *Journal of Computational Physics*, 1994, 110(2): 399 - 406.

[10] 周浩,汤文辉,冉宪文,等. 光滑粒子流体动力学的一种并行数值计算方案[J]. *航天器环境工程*, 2012, 29(1): 23 - 26.

ZHOU Hao, TANG Wenhui, RAN Xianwen, et al. Spacecraft environment engineering[J]. *Spacecraft Environment Engineering*, 2012, 29(1): 23 - 26. (in Chinese)

[11] Doring M, Andrillon A, Alessandrini B, et al. SPH free surface flow simulation [C]//*Proceedings of the 18th International Workshop on Water Waves and Floating Bodies*, Le Croisic, 2003.