<응용논문> **DOI http://dx.doi.org/10.3795/KSME-A.2012.36.4.465** ISSN 1226-4873

NVIDIA 의 GPGPU 를 이용한 수 많은 구형 접촉 입자가 포함된 다물체 동역학 해석

박지수*· 윤준식**· 최진환*†· 임성수*
* 경희대학교, ** 서울대학교

Co-simulation of MultiBody Dynamics and Plenteous Sphere of Contacted Particles Using NVIDIA GPGPU

Ji Soo Park*, Joon Shik Yoon**, Jin Hwan Choi*† and Sung Soo Rhim*
* Dept. of Mechanical Engineering Kyung Hee Univ.,
** Dept. of Mechanical Engineering Seoul Nat'l Univ.

(Received December 12, 2011; Revised January 8, 2012; Accepted January 20, 2012)

Key Words : MBD(다물체 동역학), Particles(입자), DEM(이산 요소법), GPGPU, Co-simulation(통합 해석)

초록: 본 연구에서는 수 많은 입자가 포함된 다물체 동역학 모델을 시뮬레이션 하여 그 결과를 도출하였다. 수 많은 입자들은 GPU 를 적용한 이산 요소법을 이용해 풀었다. 입자들의 Contact Force 를 계산하기 위해 Fast Algorithm 이 적용되었고 계산 속도 향상을 위해 NVIDIA 사의 CUDA 프로그래밍을 하였다. 입자들간의 계산은 Explicit 적분기가 사용되었으며 다물체 동역학은 순환 공식(Recursive Formulation)을 사용 하고 Implicit 적분기를 사용하였다. 입자들과 다물체 사이의 Contact Force 를 동시에 시뮬레이션 하기 위해서 입자동역학과 다물체 동역학의 통합해석을 할 수 있는 알고리즘을 개발하였다. 수치 실험의 예로서 화물트럭의 입자 영향을 알아 보기 위한 화물트럭 모델과 대부분의 동력 전달 장치에 사용되는 기어 모델을 시뮬레이션 하였다.

Abstract: In this study, a dynamic simulation model that considers many spherical particles and multibody dynamics (MBD) entities is developed. Plenteous spherical particles are solved using the Discrete Element Method (DEM) technique and simulated on a GPU board in a PC. A fast algorithm is used to calculate the Hertzian contact forces between many spherical particles, and NVIDIA CUDA is used to increase the calculation speed. The explicit integration method is applied to solve the many spheres. MBD entities are simulated by recursive formulation. Constraints are reduced by recursive formulation, and the implicit generalized alpha method is applied to solve the dynamic model. A new algorithm is developed to simulate the DEM and MBD models simultaneously. As a numerical example, a truck car model and gear model are developed. The results show that the proposed algorithm using a general-purpose GPU in a PC has many advantages.

- 기호설명 -

Flops : 초당 수행할 수 있는 부동소수점 연산 횟수

1. 서 론

최근 들어, GPU 는 병렬프로그래밍의 알고리즘 속도 및 효율을 증가시키기 위해서 사용할 수 있게 되었다. 이는 중앙처리장치(Central Processing Unit, CPU) 에서 수행했을 때보다 그래픽 연산장치(Graphics Processing Unit, GPU) 에서 병렬 처리 연산을 수행했을 때 가격대비 저렴한 GPU 를 사용함으로써 소프트웨어의 최적화 보다는 하드웨어(GPU) 개수에 따라 속도를 증가시킬 수 있다. 그리고 NVIDIA 회사는 GPU 에서 C 언어 기반의 프로그램을 실행할 수 있게 해주는 Compute United Device Architecture(CUDA)[1]를 지원한다.

일반적으로 GPU 는 모든 프레임에서 수백 수천의 다각형을 빠른 속도로 Rendering 을 해야 하는 비디오 게임을 위해 주로 설계가 되어있다. 하지만 최근에는 GPU 를 컴퓨터 그래픽스를 위한 계

† Corresponding Author, jhchoi@khu.ac.kr

산만 다루는 것이 아닌 CPU 가 전통적으로 취급했던 응용프로그램 들의 계산을 수행하는 기술이 발달했다. 이를 GPGPU(General Purpose Graphic Processing Unit) 라고 한다. GPGPU 를 이용해 비교적 간단하거나 매우 미세한 입자들을 병렬처리 하였을 때 GPU 는 매우 높은 부동 소수점 연산(FLoating-point OPeration, FLOP)을 할 수 있다. 특히 NVIDIA 사의 Tesla C2050 그래픽카드는 1.03 Teraflops 의 단일 정밀도(single precision)와 515 Gigaflops 의 배정밀도(double precision)를 가지고 있다.[2] 상대적으로 인텔의 i7-965 CPU 는 51.2 Gigaflops 의 단일 정밀도(single precision)와 25.6 Gigaflops 의 배정밀도(double precision)를 가지고 있다.[3] 그러므로 GPU 가 CPU 에 비해 반복적인 계산은 매우 빠르게 수행할 수 있다. 따라서 이러한 GPU 의 장점을 이산 요소법에 적용할 수 있다.

이산 요소법(DEM)은 마이크로미터 크기의 매우 많은 입자들의 움직임을 계산하기 위한 수치방법 중에 하나이다. GPU 는 병렬프로그램의 속도를 증가시켜 줄 수 있기 때문에 DEM(Discrete Element Method)[5,6]은 알갱이나 불연속적인 물질의 해석에 도입되기 시작했다.

본 연구에서는 DEM 과 병렬프로그래밍을 이용해서 많은 입자들을 해석하는 것을 입자 동역학이라 하고 입자 동역학을 구속조건과 motion 을 가지고 있는 MBD 모델에 대해서 적용하기로 한다. 대개 동역학 해석은 구속조건과 motion 을 다루고 추가적으로 제어, 최적화 등과 같은 다른 기술들과의 통합해석을 할 수 있다. 만약 입자 동역학이 다물체 동역학에 포함된다면, 시뮬레이션의 시너지 효과는 증가 될 수 있다. MBD 와 입자 동역학 두 종류의 시뮬레이션을 통합하기 위해 domain 알고리즘과 interaction 알고리즘을 나누는 것이 필요하다. 이 두 알고리즘 내에서 MBD 와 입자 동역학을 통합 해석 할 수 있는 알고리즘을 개발 하였다.

개발된 알고리즘의 유효성을 다음 2 가지 경우에 대해서 검증을 하였다. 첫 번째는 정상상태에서 박스의 중력 방향의 아래쪽 면의 반력과 박스 안에 있는 입자들의 총 무게와 같은 것인지 확인하였다. 두 번째는 원심력에 의해 생기는 표면의 곡선을 실제의 이론 값과 비교하여 본 연구에서 사용된 알고리즘의 유효성에 대해 검증을 하였다. 수치 실험의 예로서 화물트럭 모델과 기어 모델을 사용하였다. 첫 번째로 화물트럭 모델은 입자가 화물트럭에 주는 영향을 알 수 있는 모델이다. 이로 인해 화물트럭의 무게 중심 쏠림 현상 등을 알 수 있다. 기어 모델은 대부분의 동력전달장치에 사용되기 때문에 많은 분야에 적용할 수 있을 것이다. 그러면 예상되는 효과로는 기어와 기어 사이의 이물질에 의해 생기는 접촉력이 달라져 기어의 파손 및 변형 등을 예상할 수 있다.

본 논문은 다음과 같이 구성 되어 있다. 서론 후에 본 연구에 사용된 기초 이론이 2 절에 있다. 3 절에는 개발된 알고리즘의 수치 실험에 대한 유효성 및 성능에 대해 서술 하였다. 4 절에는 입자 동역학과 MBD 에 대한 통합 해석을 적용한 사례에 대해서 서술 하였고 마지막 5 절에는 본 연구의 결론이 요약되어 있다.

2. 기초 이론

2.1 입자동역학(DEM)

입자가 기반인 기술들이 많은 응용 분야에서 사용되고 있다. DEM 은 수많은 고체 입자의 거동을 계산하는 수치 방법이다. 이 시뮬레이션은 각각의 세분화 된 입자들의 기구학적인 힘을 결정한다. 각각의 입자에 작용하는 모든 힘들은 모든 time step 에서 계산되고 모델링 된다.[5,6] 대부분의 MBD 솔버에서는 implicit 적분기를 사용하지만 DEM 은 explicit 적분기를 사용한다. 그 이유는 MBD 에서는 상태변수가 저주파의 형태로 나타나기 때문에 Implicit 적분기의 장점인 큰 Time Step 을 사용할 수 있지만 DEM 솔버는 상태변수가 고주파 형태로 나타나기 때문에 Time Stepsize 를 작게 사용하여야 하기 때문에 implicit 의 장점을 이용하기 어렵다. 따라서 일반적으로 정밀도(Accuracy) 관점에서 Explicit 적분기를 사용한다. 입자들의 움직임은 다음의 방정식을 따르는 뉴턴의 운동법칙에 의해 업데이트 된다.

$$\frac{d\mathbf{v}}{dt} = \frac{\sum \mathbf{F}}{m} \tag{1}$$

$$\frac{d\boldsymbol{\omega}}{dt} = \frac{\sum \mathbf{M}}{I} \tag{2}$$

여기서 $\mathbf{v}$ 는 입자 속도, $\mathbf{F}$ 는 입자에 작용하는 힘의 합, m 입자의 질량, $\boldsymbol{\omega}$ 는 각속도, $\mathbf{M}$ 는 moment, I 는 관성 모멘트이다.

두 개의 입자간의 contact 모델은 Hertzian contact 모델과 Voigt 모델에 의해 정의 되어 있다.[7] 접촉력 $\mathbf{F}_n$, 전단력 $\mathbf{F}_t$ 는 다음의 방정식에 의해 계산할 수 있다.

$$\mathbf{F}_{n,ij} = k_n|\delta|^{1.5}\mathbf{n}_{ij} + c_n\mathbf{u}_{n,ij}|\delta|^{0.25} \quad (3)$$

$$\mathbf{F}_{t,ij} = \min\left\{k_t|\delta_t|^{1.5}\mathbf{t}_{ij} + c_t\mathbf{u}_{t,ij}|\delta_t|^{0.25}, \mu_f|\mathbf{F}_n|\mathbf{t}_{ij}\right\} \quad (4)$$

여기서 k 와 c 는 각각 스프링 상수와 감쇠 상수이다. $\mathbf{n}_{ij}$ 와 $\mathbf{t}_{ij}$ 는 i 번째 입자에서 j 번째의 입자방향의 수직 및 접선 방향의 단위 벡터이다. δ 와 $\mathbf{u}$ 는 두 입자간의 contact 과 상대 속도에 의한 변형이다. 강체에 작용하는 힘은 강체와 접촉하고 있는 입자들간의 힘의 합으로 구할 수 있다.

입자들간의 contact search 는 Fast Algorithm 이 적용되었고 Fast Algorithm 을 구현하기 위해 Cell-Linked List[13]를 이용하였다. 입자동역학에서 가장 많은 시간이 소요되는 것이 각각 입자의 contact pair 를 찾는 것이다. 그래서 contact search 를 빨리 하는 것이 전체의 해석 시간을 줄이는 데에 많은 도움이 된다. Contact search 를 빨리 하기 위해 Cell-Linked List 알고리즘을 적용하였다. Cell-Linked List 는 전체 공간에 대해 작은 크기의 Cell 나눈다. 그리고 모든 입자의 위치에 따라 Cell 을 할당 한다. 한 입자에 대해 이웃하고 있는 Cell (2 차원 8 개, 3 차원 26 개) 의 내부에 있는 입자들만 contact search 를 한다. 요약을 하면 모든 입자는 임의의 cell 에 포함되고 이웃하고 있는 Cell 에 대해서만 contact 하는 입자를 찾는다. 아래의 Fig. 1 는 Cell-Liked List 의 알고리즘이 적용이 되지 않은 것(좌)과 적용된(우) 것을 비교해놓은 그림이다. Cell-Linked List 가 적용되지 않으면 Fig. 1 의 왼쪽처럼 1 개의 입자는 23 개의 입자에 대해 contact search 를 해야 하지만 Cell-Linked List 가 적용된 오른쪽 그림은 1 개의 입자에 대해 12 개의 입자만 contact search 를 하면 되기 때문에 contact search 를 빨리 할 수 있다.

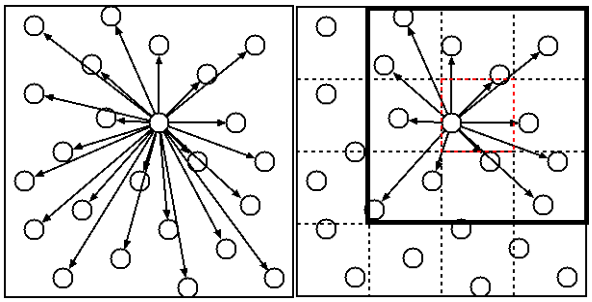

Fig. 1 Comparison of contact search of cell-linked list

2.2 GPU 의 가속화 기술

현재까지 CPU 를 이용해 많은 입자가 포함된 모델을 시뮬레이션 하기 위해서는 높은 컴퓨터의 사양이 필요한 것이 가장 큰 단점이었다. 하지만 입자 시스템은 상대적으로 MBD 모델에 비해 병렬 프로그래밍하는 것이 어렵지 않다. 그래서 GPU 기술은 효과적으로 많은 입자를 다루고 결과 값을 얻는 데 매우 효과적인 방법 일 것이다. 그러므로 GPU 기술을 DEM 솔버에 적용하였다.

GPU 는 병렬 프로그래밍을 함으로써 엄청나게 많은 멀티 쓰레드를 사용할 수 있게 해준다. GPU 는 많은 멀티프로세서를 가지고 있고 또 그 멀티프로세서는 여러 개의 코어들로 이루어져 있다. 본 연구에서 사용된 NVIDIA Tesla C2050 은 14 개의 멀티프로세서와 448 개의 코어로 이루어져 있다. 다음 Fig. 2 는 본 연구에 사용된 PC 이다. 그림에서 볼 수 있듯이 GPU 를 사용하기 위해서는 평소에 사용하던 PC 에서 사용하던 일반적인 그래픽 카드 대신에 CUDA 가 지원되는 NVIDIA 그래픽카드로 교체만 하면 된다. 단순히 그래픽 카드만을 교체하면 되기 때문에 GPU 를 사용하기 위해 복잡한 과정이 필요 없어 이 또한 계산용 GPU 를 사용하는 데에 있어 큰 장점이다.

GPU 에서 실행되는 코드는 단일 구조에서 멀티 쓰레드를 이용할 수 있는 코드로 되어 있다. 즉 각각의 쓰레드에서 같은 코드는 실행이 되지만 다른 코드는 실행이 되지 않는다. 다음 Table 1 은 CPU 와 GPU 병렬 프로그래밍을 비교한 예제 코드이다.

Fig. 2 NVIDIA Tesla C2050

Table 1 Comparison of CPU and GPU code

CPU	**void division(float *a, float b, int n)** **{** **for(int i=0; i<n; i++)** **a[i] = a[i] / b;** **}** **void main()** **{** **float *cpu;** **cpu = (float *)malloc(sizeof(float));** **free(cpu);** **delete x_cpu;** **if(b != 0){** **division (a,b,16);** **}** **else{** **b = 0.5;** **division (a,b,16);** **}** **}**
GPU	**__global__ void division(float *a, float b, int n)** **{** **int i= blockIdx.x*blockDim.x+threadIdx.x;** **a[i] = a[i] / b;** **}** **void main()** **{** **float *gpu;** **cudaMalloc((void**)&gpu, sizeof(folat));** **cudafree(gpu);** **if(b != 0){** **division <<<8,2>>>(a,b,16);** **}** **else{** **b = 0.5;** **division <<<8,2>>>(a,b,16);** **}** **}**

위의 CPU 와 GPU 의 비교 코드는 각각 메모리 할당과 a 의 값을 b 의 값으로 16 번 나누는 코드이다. CPU 의 경우에 메모리 할당은 일반적으로 C 언어에서 제공하는 malloc 이라는 라이브러리 함수를 사용하면 된다. 하지만 GPU 의 경우는 CUDA 에서 제공되는 cudaMalloc 이라는 라이브러리 함수를 사용하면 된다. 2 번째인 a 의 값을 b 의 값으로 16 번 나누는 코드는 CPU 와 GPU 에서의 가장 큰 차이점은 함수 호출 방법이 다르다. 우선 b 의 값

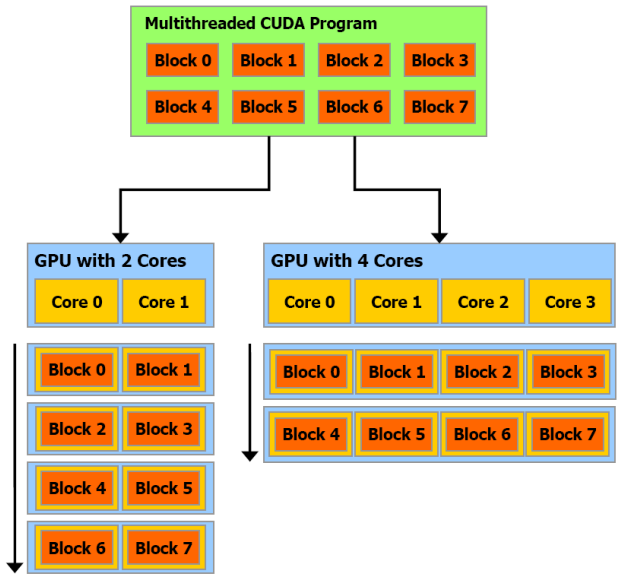

Fig. 3 Comparison of the number of core of GPU

값은 0 이 되면 안되기 때문에 0 일 경우에는 b 의 값에 0.5 를 넣어서 계산하게 되어있다. 그러면 b 의 값이 0 이 아닐 경우에 CPU 가 1 개만의 코어만을 이용한다면 총 16 번의 계산을 CPU 에서 순차적으로 하게 된다. 하지만 GPU 는 CPU 와 같은 방법으로 계산을 하지 않는다. GPU 의 경우는 함수 정의 및 호출부터 다르다. GPU 의 함수 정의는 함수 이름 앞에 __global__이란 예약어가 필요하다. 그리고 반복문에 사용되는 index 값인 i 의 정의는 내가 사용할 block 과 thread 의 index 값을 사용한다. 함수 호출은 division<<<8,2>>>(a,b,16)의 방법으로 호출을 한다. "<<< >>>"는 GPU 에서 함수를 호출하는 약속된 문법이다. 그리고 "<<< >>>"안에 있는 8,2 는 8 개의 block 에 각각 2 개의 thread 를 사용한다는 뜻이다. 그러면 GPU 코드를 실행을 하게 되면 Fig. 3 과 같이 block 과 thread 가 할당된다. 만약 2 개의 코어를 가진 GPU 를 사용한다면 2 개의 코어에 각각 4 개의 block 이 할당되고 다시 거기에 2 개의 thread 가 할당됨으로써 4 번의 루프를 실행하면 된다. 만약에 4 개의 코어를 가진 GPU 를 사용한다면 두 번의 루프 실행을 통해 모든 계산을 할 수 있다. 본 연구에 사용된 GPU 는 448 개의 코어를 가지고 있기 때문에 448 번의 계산을 한 번에 할 수 있다. 이 점이 한 번에 한번씩 계산할 밖에 없는 CPU 와 다른 점이고 이를 이용하면 CPU 에 비해 GPU 를 이용할 경우 매우 빠른 계산 속도를 이용할 수 있다.

2.3 다물체 동역학(MBD)

강체 동역학은 다양한 공식을 사용해 모델링 할

수 있다.[8] 본 연구의 다물체 동역학 모델의 솔버는 순환 공식(Recursive Formulation)이 사용되었고 이 번 절에서 순환 공식에 대해 설명한다. 3 차원 공간에 접촉하고 있는 두 강체 사이의 좌표 시스템은 Fig. 4 에서 볼 수 있다. 두 강체는 조인트에 의해 연결되어 있고 j 강체에 외력 F 가 작용한다. X-Y-Z 는 절대좌표계이고 x'-y'-z'은 절대좌표계에서의 강체좌표계이다. 절대좌표계에서 강체좌표계의 원점에 대한 속도와 가상 변위는 각각 다음과 같이 정의된다.

$$\begin{bmatrix}\dot{\mathbf{r}} & \boldsymbol{\omega}\end{bmatrix}^T \tag{5}$$

$$\begin{bmatrix}\delta\mathbf{r} & \delta\boldsymbol{\pi}\end{bmatrix}^T \tag{6}$$

강체좌표계에서 크기는 각각 다음과 같이 정의된다.

$$\mathbf{Y}=\begin{bmatrix}\dot{\mathbf{r}}'\\ \boldsymbol{\omega}'\end{bmatrix}\equiv\begin{bmatrix}\mathbf{A}^{\mathrm{T}}\dot{\mathbf{r}}\\ \mathbf{A}^{\mathrm{T}}\boldsymbol{\omega}\end{bmatrix} \tag{7}$$

$$\delta\mathbf{Z}=\begin{bmatrix}\delta\mathbf{r}'\\ \delta\boldsymbol{\pi}'\end{bmatrix}\equiv\begin{bmatrix}\mathbf{A}^{\mathrm{T}}\delta\mathbf{r}\\ \mathbf{A}^{\mathrm{T}}\delta\boldsymbol{\pi}\end{bmatrix} \tag{8}$$

여기서 $\mathbf{A}$ 는 강체에 부착된 강체 좌표계로의 좌표 변환행렬이다.

접해 있는 강체 사이의 recursive 속도 방정식은 다음과 같이 정의 된다.

$$\mathbf{Y}_j=\mathbf{B}_{ij}^{1}\mathbf{Y}_i+\mathbf{B}_{ij}^{2}\dot{\mathbf{q}}_{ij} \tag{9}$$

여기서 $\mathbf{Y}$ 는 식 (7)에서 정의된 병진과 회전이 결합된 속도이고 $\mathbf{B}_{ij}^{1}$ 와 $\mathbf{B}_{ij}^{2}$ 는 다음과 같다.

$$\mathbf{B}_{ij}^{1}=\begin{bmatrix}\mathbf{A}_{ij}^{\mathrm{T}} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_{ij}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\mathbf{I} & -\left(\tilde{\mathbf{s}}'_{ij}+\tilde{\mathbf{d}}'_{ij}-\mathbf{A}_{ij}\tilde{\mathbf{s}}'_{ji}\mathbf{A}_{ij}^{\mathrm{T}}\right)\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{10}$$

$$\mathbf{B}_{ij}^{2}=\begin{bmatrix}\mathbf{A}_{ij}^{\mathrm{T}} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_{ij}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \left(\mathbf{d}'_{ij}\right)_{\mathbf{q}_{ij}}+\mathbf{A}_{ij}\tilde{\mathbf{s}}'_{ji}\mathbf{A}_{ij}^{\mathrm{T}}\mathbf{H}'_{ij}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{11}$$

행렬 $\mathbf{B}_{ij}^{1}$ 와 $\mathbf{B}_{ij}^{2}$ 는 단지 $\mathbf{q}_{ij}$ 의 함수라는 것은 중요한 사실이다. 비슷하게 recursive 가상 변위 관계도 다음과 같이 얻을 수 있다.

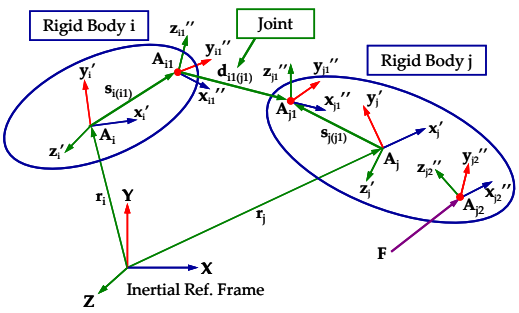

Fig. 4 Two contiguous rigid bodies

$$\delta\mathbf{Z}_j=\mathbf{B}_{ij}^{1}\delta\mathbf{Z}_i+\mathbf{B}_{ij}^{2}\delta\mathbf{q}_{ij} \tag{12}$$

만약 식 (5)의 순환 공식이 모든 조인트에 적용이 된다면 직교좌표계와 상대 좌표 사이의 속도 관계는 다음과 같이 정의된다.

$$\mathbf{Y}=\mathbf{B}\dot{\mathbf{q}} \tag{13}$$

여기서 $\mathbf{B}$ 는 $\dot{\mathbf{q}}_{ij}$ 의 계수들의 집합이다.

$$\mathbf{Y}=\left[\mathbf{Y}_0^{\mathrm{T}},\mathbf{Y}_1^{\mathrm{T}},\mathbf{Y}_2^{\mathrm{T}},\ldots,\mathbf{Y}_n^{\mathrm{T}}\right]^{\mathrm{T}}_{\mathrm{nc}\times 1} \tag{14}$$

$$\dot{\mathbf{q}}=\left[\mathbf{Y}_0^{\mathrm{T}},\dot{\mathbf{q}}_{01}^{\mathrm{T}},\dot{\mathbf{q}}_{12}^{\mathrm{T}},\ldots,\dot{\mathbf{q}}_{(n-1)n}^{\mathrm{T}}\right]^{\mathrm{T}}_{\mathrm{nr}\times 1} \tag{15}$$

여기서 nc 와 nr 은 각각 절대좌표계와 상대좌표계에서 정의된 일반 좌표의 개수이다. $\dot{\mathbf{q}}\in\mathbf{R}^{\mathrm{nr}}$ 의 주어진 $\mathbf{Y}\in\mathbf{R}^{\mathrm{nc}}$ 는 식 (9) 또는 식 (13)을 이용해서 구할 수 있다. 또한 $\mathbf{R}^{\mathrm{nc}}$ 에 속해 있는 $\mathbf{G}$ 벡터는 $\mathbf{R}^{\mathrm{nr}}$ 의 새로운 벡터 $\mathbf{g}=\mathbf{B}^{\mathrm{T}}\mathbf{G}$ 로 변환이 자주 필요하게 되었다. 이런 변환은 직교좌표계에서 정의된 힘들을 조인트 부위에서 얻을 수 있는 일반화된 힘으로 계산 할 수 있게 한다. 직교 좌표계에서 정의된 힘 $\mathbf{Q}\in\mathbf{R}^{\mathrm{nc}}$ 이 행한 가상일은 다음과 같이 정의된다.

$$\delta\mathbf{W}=\delta\mathbf{Z}^{\mathrm{T}}\mathbf{Q} \tag{16}$$

여기서, $\delta\mathbf{Z}$ 는 시스템의 모든 조인트에 대해서 기구학적으로 정의 될 수 있어야 한다. $\delta\mathbf{Z}=\mathbf{B}\delta\mathbf{q}$ 로 변환하여 식 (16)에 대입하면 다음과 같은 식으로 유도 된다.

$$\delta\mathbf{W}=\delta\mathbf{q}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{Q}=\delta\mathbf{q}^{\mathrm{T}}\mathbf{Q}^{*} \tag{17}$$

여기서, $\mathbf{Q}^{*}\equiv\mathbf{B}^{\mathrm{T}}\mathbf{Q}$ 이다.

구속조건이 주어진 시스템의 운동방정식은 다음과 같다.[9,10]

$$\mathbf{F}=\mathbf{B}^{\mathrm{T}}\left(\mathbf{M}\dot{\mathbf{Y}}+\boldsymbol{\Phi}_{\mathbf{Z}}^{\mathrm{T}}\boldsymbol{\lambda}-\mathbf{Q}\right)=\mathbf{0} \tag{18}$$

여기서, $\boldsymbol{\lambda}$ 는 Lagrange multiplier 이며, $\boldsymbol{\Phi}$ 는 위치 수준 구속 벡터이다. $\mathbf{M}$ 과 $\mathbf{Q}$ 는 각각 직교좌표계에서 질량행렬과 접촉력이 포함된 힘 벡터이다.

2.4 입자 동역학과 다물체 동역학의 통합

다물체 시스템에서 수 많은 입자들이 포함된 시뮬레이션은 MBD 솔버만을 이용해서는 시뮬레이션 하

는 것이 매우 어렵다. 따라서 GPU 를 이용한 입자동역학을 도입하여 MBD 모델과의 통합 해석을 할 수 있는 새로운 알고리즘을 개발하였다. Fig. 5 는 다물체 동역학과 입자 동역학의 통합해석을 위한 개념을 나타낸다. 예를 들어 스프링, 박스, 수 많은 입자들로 이루어진 시스템이 있다고 하면 Fig. 5 의 모델은 각각 MBD 부분의 모델과 Particle 부분의 모델로 나눈다. 2 부분으로 나눈 모델의 MBD 부분인 스프링과 박스는 MBD 솔버로 풀고 박스 안의 수 많은 입자들은 DEM 솔버로 각각 풀 수 있다. 각각의 솔버가 자신들의 모델을 풀기 위해서 필요한 정보는 MBD 모델은 박스 6 면에서 생성되는 힘의 크기이다. 그리고 Particle 솔버에서 필요한 정보는 박스의 위치와 속도가 필요하다. 하지만 MBD 모델과 Particle 솔버는 사용하는 적분기가 다르기 때문에 매 순간 순간 같은 step 에서 통합 해석을 할 수 없다. Fig. 6 는 다른 적분기를 사용하는 솔버 간의 통신을 할 수 있게 하는 알고리즘을 그림으로 나타낸 것이다. Fig. 6 와 같이 t(i) 스텝에서 입자들과 박스 사이에서 생기는 힘을 구한다. 그렇게 구해진 힘을 MBD 솔버에 넘겨주면 MBD 솔버는 그 힘의 정보를 가지고 t(i+1)의 스텝에서의 박스의 위치와 속도를 구한 후에 그 결과 값을 Particle 솔버에 넘겨준다. 그러면 Particle 솔버는 다시 전달 받은 박스의 위치와 속도를 가지고 t(i+1) step 에서 생성되는 박스에 가해지는 힘을 계산하고 그 결과를 다시 MBD 솔버에 넘겨준다. 이러한 과정을 반복하면 다른 적분기를 사용하는 MBD 와 Particle 솔버 간에 통합 해석을 할 수 있다.

3. 통합해석의 유효성 및 성능

입자동역학의 유효성을 검증하기 위해 두 개의 모델을 시뮬레이션 했다. 첫 번째 모델은 Fig. 7 처럼 모델링 한 후 시뮬레이션을 하여 얻은 값과 이론 값을 비교해 보았다. 공의 지름과 공의 개수에 따른 오차율을 알아보기 위해 각각 시뮬레이션 해보았고 그 결과들을 Table 2와 Table 3에 정리하였다. 공의 밀도는 7800kg/m^3, 중력은 9.807m/s^{2}을 적용하였다. 2가지의 경우의 오차율을 보았을 때 입자동역학과 다물체 동역학 솔버간의 통합해석의 유효성이 검증되었다고 할 수 있다.

Table 2 The error according to the diameter of sphere

공의 지름	공의 개수	이론값 (N)	수치해석값 (N)	오차율 (%)
0.030m	1000	1081.4169	1081.4001	1.6e-3
0.025m	1000	625.81998	625.78877	5.0e-3
0.020m	1000	320.41983	320.41694	9.0e-4
0.015m	1000	135.17712	135.17630	6.1e-4
0.010m	1000	40.05248	40.05032	5.4e-3

Table 3 The error according to the number of sphere

공의 개수	공의 지름	이론값 (N)	수치해석값 (N)	오차율 (%)
1000	0.025m	625.8199	625.7887	5.0e-3
2744	0.025m	1717.2500	1717.2778	1.6e-3
4840	0.025m	3028.9687	3028.8804	2.9e-3
10648	0.025m	6663.7312	6663.6976	5.0e-4
20700	0.025m	12954.474	12954.161	2.4e-3
54872	0.025m	34339.994	34339.499	1.4e-3

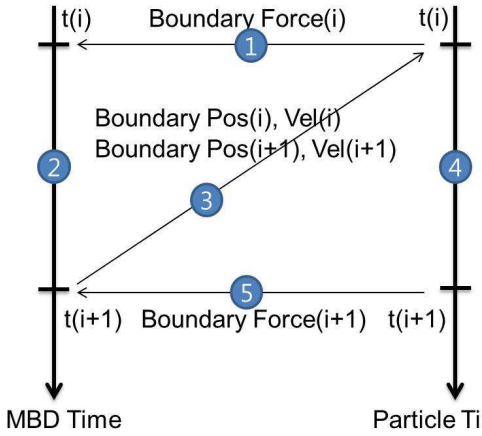

Fig. 6 Co-simulation procedure of MBD and particle pynamics

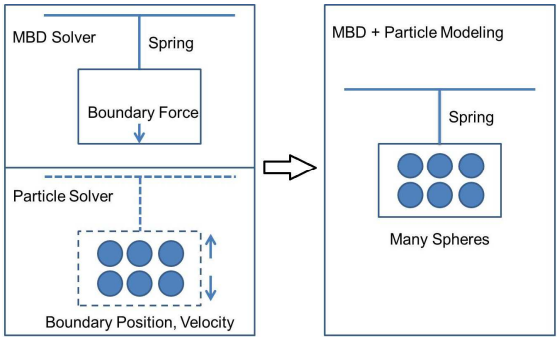

Fig. 5 Modeling of MBD and particle dynamics

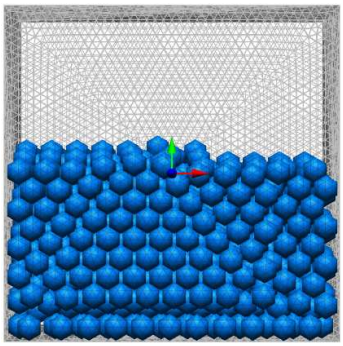

Fig. 7 Total sphere weight in a box model

두 번째 모델은 원심력을 테스트하는 모델이다. 한 실린더에 25,100 개의 공을 Fig. 8 과 같이 모델링 하였다. 모델의 처음 공들의 모여 있는 표면은 평평한 상태였다. 그리고 실린더를 10rad/s 로 회전 시켰다. 실린더 내부가 안정화 된 상태가 되면 그 때의 위쪽 표면의 곡선을 수학적으로 표현할 수 있다.

식 (19)는 표면에서는 곡선의 방정식이다.[11] 시뮬레이션 결과는 Fig. 9 에 있고 샘플링 된 공의 위치는 이론적인 방정식과 같은 모양을 하고 있다.

$$z = h_0 - \frac{(wR)^2}{2g}\left[\frac{1}{2} - \left(\frac{r}{R}\right)^2\right] \tag{19}$$

여기서, R 은 실린더의 반지름, r 은 회전 중심축으로부터의 거리, w 은 실린더의 각속도, g 는 중력가속도($9.807m/s^2$) 이고 h_0 는 실린더의 초기 높이 이다.

입자동역학의 성능을 테스트하기 위해 2 가지 경우에 대해서 테스트를 하였다. 첫 번째는 입자의 개수에 따라 걸리는 해석 시간이고 두 번째는 입자간의 contact 의 수가 증가함에 따라 contact search 하는 시간을 테스트 하였다.

Fig. 8 25,100 contacted spheres in a cylinder model

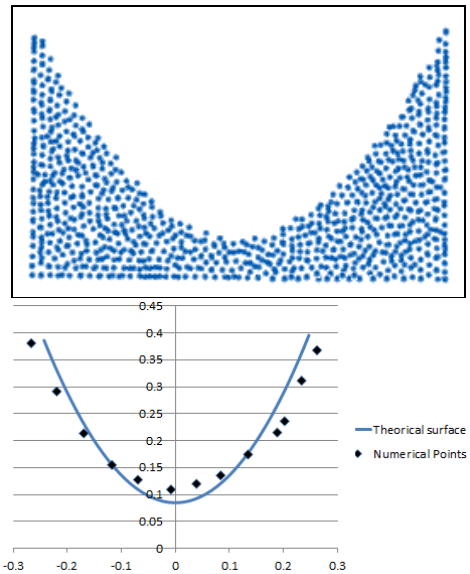

Fig. 9 Surface shape of a centrifugal force in a cylinder model

첫 번째 테스트 모델은 Fig. 7 과 같이 박스 안에 공의 개수를 바꾸어 채워가며 시뮬레이션을 해보았다. 모든 모델에서의 해석은 NVIDIA Tesla C2050 을 사용하였다. Stepsize 는 8.75e-7 이고 시뮬레이션 시간은 1 초이다. Fig. 10 은 입자의 개수가 증가함에 따라 해석 시간을 나타내는 그래프이다. 입자의 개수가 10000 이하의 경우에는 1 시간 정도의 시간이 걸렸으며 10000 개의 이상의 경우에는 시뮬레이션 하는데 걸리는 시간은 10000 개의 1 시간을 기준으로 선형적으로 증가하는 것을 알 수 있었다. 10000 개 이하의 경우에 1 시간 정도로 비슷한 시간이 걸리는 이유는 CPU 와 GPU 간에 정보를 주고 받는 일정한 시간이 필요로 하기 때문이다. 두 번째 모델은 입자의 수가 증가함에 따라 contact search 를 하는 데 걸리는 시간을 측정하였다. 다음 Fig. 11 은 입자의 수가 증가함에 따라 contact search 에 걸리는 시간의 CPU 와 GPU 의 비를 나타낸 그래프이다. Contact search 에 사용되는 시간이 입자의 개수가 2 만개 이하 일 때는 CPU 의 계산 속도가 빨랐지만 그 이상의 개수가 되었을 경우에는 GPU 의 계산 속도가 더 빨랐다. 특히 100 만개의 입자의 contact search 는 GPU 가 CPU 에 비해 약 10 배 정도의 계산 속도 향상이 있었다.

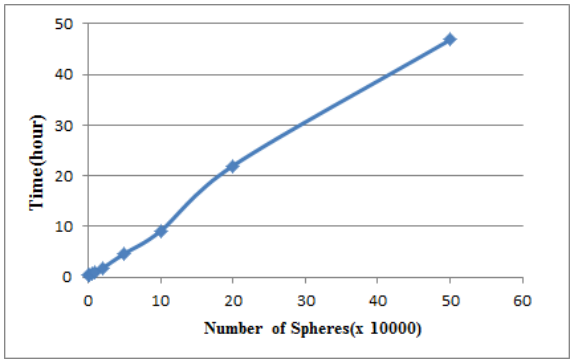

Fig. 10 Performance of the number of particles

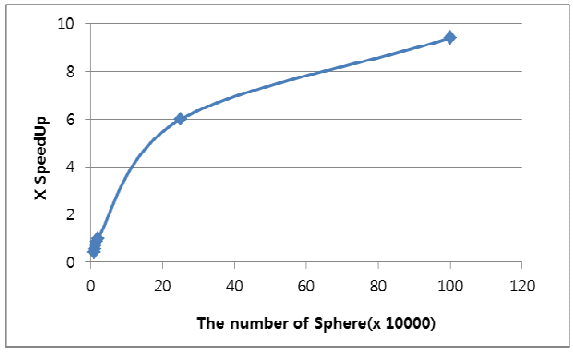

Fig. 11 The ratio of CPU and GPU according to the number of sphere

위의 성능 테스트의 결과들 보았을 때 GPU 는 매우 많은 수의 입자들을 빠르게 계산하는 매우 효과적인 방법이다.

4. 수치 실험의 통합 해석

이번 장에서는 두 가지 모델의 MBD 와 입자 동역학의 통합해석을 하였다.

첫 번째 모델은 화물 트럭 모델이다. 모델링은 Fig. 12 처럼 하였고 통합 해석을 위해 사용된 입자의 개수는 53000 개이다. 해석하는데 4 시간 24 분이 걸렸다. 이 모델을 통하여 화물 트럭에 입자가 있었을 때와 없었을 때, 화물 트럭이 갑작스러운 출발을 하였을 경우 생기는 입자가 한쪽방향으로 쏠리는 현상으로 인하여 화물 트럭의 가속도의 변화를 측정하였다. 이 결과로 화물 트럭의 무게중심 변화를 예측할 수 있고 이 결과로 화물 트럭에 걸리는 부하에 대해 해석을 할 수 있다.

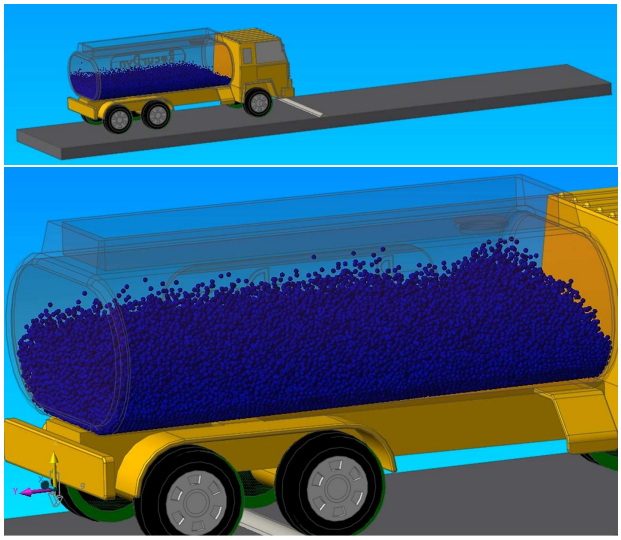

Fig. 12 Simulation results of truck car

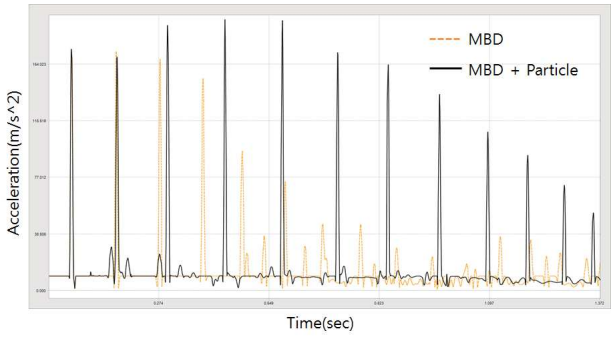

Fig. 13 Acceleration of MBD and MBD + particle

위의 Fig. 13 은 시간에 따른 화물 트럭의 가속의 변화를 나타낸 것이다. 그래프에서 알 수 있듯이 MBD 에 Particle 이 포함된 모델이 입자의 영향으로 가속도의 크기가 MBD 모델만의 가속도의 크기보다 컸다. 가속도의 크기가 크다는 것은 화물 트럭의 무게중심에 영향을 주었다고 할 수 있고 무게중심의 변화 때문에 전반적으로 화물 트럭에 부하를 줄 수 있다. 두 번째 모델은 대부분의 동력 전달 장치에 사용되는 기어 모델이다. Fig. 14 는 기어 모델을 모델링하고 통합해석을 한 그림이다. 총 입자의 개수는 10296 개이고 해석하는데 20 시간이 걸렸다. 이 모델은 이물질(입자) 등이 기어 사이에 끼였을 경우 기어의 구동 문제 및 부하에 대해 해석을 할 수 있다. Fig. 15 는 입자가 있었을 때와 없었을 때의 기어의 반력을 비교한 그래프이다. 그래프에서 볼 수 있듯이 입자의 영향으로 기어의 반력이 입자가 없었을 때보다 10 배 정도 크게 나왔음을 알 수 있다. 반력이 크다는 것은 기어의 동력 전달 효율에 악영향을 줄 뿐만 아니라 기어 이 표면의 파손의 한 요인으로 작용할 수 있다.

예제로 사용한 2 개의 모델에 대한 상세한 내용은 다음 Table 4 와 같다.

Table 4 Result of co-simulation models

Model	Truck Car	Gear
Number of Sphere	53,000	10,296
Radius of Sphere	0.0225m	0.00015m
Stepsize of Particle	8.54e-6	5.82e-6
Simulation Endtime	5sec	1sec
Stepsize of MBD	1.0e-3	1.0e-3
CPU Time	4hr 24min	20hr

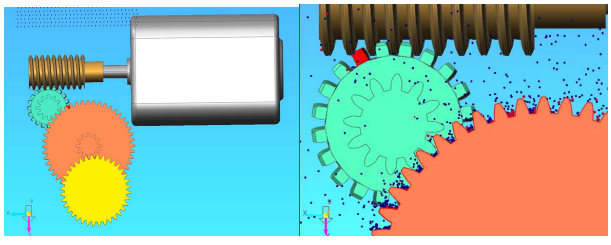

Fig. 14 Simulation results of gear model

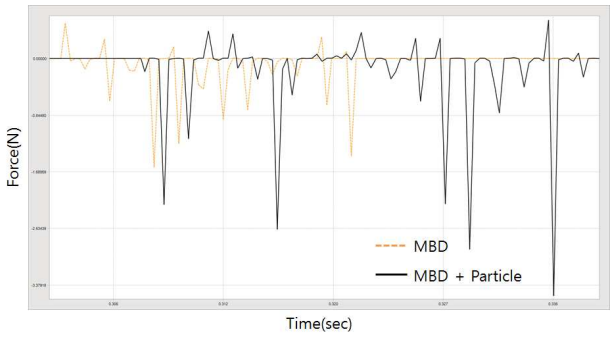

Fig. 15 Comparison of reaction force of gear of MBD and MBD + particle

위의 두 모델들은 수 많은 입자들을 계산하기 위해 계산용 GPU 인 NVIDIA TESLA C2050 을 사용하여 시뮬레이션 했다. 위의 두 예제들은 CPU 만으로는 해석이 굉장히 어려운 모델이다. 따라서 GPU 기술을 적용한 DEM 솔버를 적용함으로써 불가능 했던 시뮬레이션을 빠른 시간에 할 수 있었다. GPU 기술을 적용하여 화물 트럭에 입자들이 주는 영향을 통하여 화물 트럭의 무게 중심에 영향을 준 다는 것을 알 수 있었다. 기어 모델의 경우 이물질(입자)이 기어에 주는 영향을 알아 보고 싶어도 CPU 만의 기술로는 시뮬레이션 통하여 알 수가 없었다. 하지만 이 역시 GPU 기술을 적용하여 이물질(입자)이 기어 표면이나 기어 이 사이에 끼어서 생기는 반력을 통하여 기어에 주는 부하를 알 수 있었다.

5. 결 론

본 연구에서 GPU 를 사용한 입자 동역학은 우수한 성능을 낼 수 있다고 소개되었다. 입자 동역학의 유효성을 검증를 하기 위해 공들의 총 무게와 원심력을 테스트하는 모델을 가지고 유효성을 검증하였다. 그 결과 충분히 신뢰성이 있다고 판단되었다. 또한 입자 동역학의 성능 테스트는 contact 의 수를 검색하는 시간과 박스 안에 입자들의 개수를 변화를 줌으로써 성능을 테스트 하였다. 그 결과 해석 시간도 CPU 에 비해 많은 시간을 단축할 수 있다. MBD 와 입자 동역학과의 통합해석을 위해 화물 트럭 모델과, 기어 모델로 시뮬레이션을 해보고 결과를 얻었다. 그 결과로 화물트럭 모델은 갑작스러운 출발을 할 경우에 화물트럭 탱크 안에 입자들의 반력 및 입자들의 움직임을 통해 화물트럭의 무게중심 쏠림 현상 등을 알 수 있었다. 기어 모델은 이물질(입자)등이 기어 이 사이에 끼게 되면 기어 이 표면에 생기는 반력이 변하는 것을 알 수 있었다. 그 결과를 가지고 기어에 걸리는 부하 및 기어 이 표면 파손에 대한 예측을 할 수 있을 것이다.

지금까지 MBD 에서는 많은 입자가 있는 MBD 모델은 해석을 하기가 매우 어려웠다. 하지만 GPU 기술의 발전으로 MBD 모델에 적용할 수 있었고 이를 통해 수많은 입자가 포함된 MBD 모델을 저렴한 비용으로 빠른 시간 안에 해석을 할 수 있고 입자들이 MBD 모델에 주는 영향을 직접 실험을 통하지 않고 시뮬레이션을 통하여 알 수 있었다. 향후에는 DEM 이 아닌 또 다른 수치해석 방법인 Smoothed Particle Hydrodynamics (SPH) 를 이용해 수 많은 입자가 포함된 다물체 동역학 모델에 대해 연구할 것이다. SPH 는 유체를 입자들의 한 집합체라 생각하고 입자들에 대해 유체 방정식을 적용한 방법이다.

참고문헌

(1) NVIDIA Corporation, 2008, NVIDIA CUDA: Compute Unified Device Architecture, Programming Guide, in, NVIDIA Corporation, Santa Clara.

(2) NVIDIA Corporation., 2010, Tesla C2050 and Tesla C2070 Computing Processor Board; Available From: http://www.nvidia.com/docs/IO/43395/BD-04983-001_v03.pdf.

(3) Kapre, N. and DeHon, A., 2009, Performance Comparison of Single-Precision SPICE Model-Evaluation on FPGA, GPU, Cell, and multi-core Processors, in: International Conference on Field Programmable Logic and Applications, pp. 65~72.

(4) FunctionBay, Inc, 2010, RecurDyn User Manual, http://eng.functionbay.co.kr/.

(5) Cundall, P. A. and Strack, O. D. L., 1979, A Discrete Numerical Model for Granular Assemblies, Geotechnique, 29, 47~65.

(6) Hockney, R. W. and Eastwood, J. W., 1981, *Computer Simulation Using Particles*, McGraw-Hill, New York.

(7) Mindlin, R.D. and Deresiewicz, H., 1953, Elastic Spheres in Contact Under Varying Oblique Forces, *Trans. ASME, J. Appl. Mech*., 20, 327~344.

(8) Jalón, J.G. and Bayo, E., 1994, *Kinematic and Dynamic Simulation of Multibody Systems*, Springer-Verlag New-York.

(9) García de Jalón D. J., Unda J., and Avello A., 1986, "Natural Coordinates for the Computer Analysis of Multibody Systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 56, pp.309~327.

(10) Wittenburg, J., 1977, Dynamics of Systems of Rigid

Bodies, BF Teubner, Stuttgart.
(11) Fox, R. W. and McDonald, A. T., 1994, Introduction to Fluid Mechanics the fourth edition, John Wiley & Sons, Inc.
(12) Metariver, 2011, http://www.metariver.kr/.
(13) Mattson, W. and Rice, B. M., 1999, Neighbor Calculations Using a Modified Cell-Linked List Method, *Computer Physics Communication,* Vol.119 pp.135~148.
(14) Yoon, J. S., Park, J. S., Ahn, C. O and Choi, J. H., 2011, Cosimulation of MBD(Multi Body Dynamics) and DEM of Many Spheres Using GPU Technology, Particles 2011.
(15) Yoon, J. S., Choi, J. H., Rhim, S. and Koo, J. C., 2011, Particle Dynamics Integration to MultiBody Dynamics Using GPU, ICETI 2011.