Paralleled Geopotential Computing Methods Based on GPU

J. Liu, W. Wang, Y. Gao and L. Shu

Abstract The spherical harmonics model to describe geopotentials relies on recursive algorithms, whose computational efficiency degrades significantly if the order of the expansion into spherical harmonics increases. In order to enhance the computational efficiency for high-fidelity geopotentials, a Global Point Mascon model and a fast algorithm using GPU-based parallel computing are presented in this paper. The GPU-based parallel computing is implemented with compute unified device architecture (CUDA), enabling fast computation of high-fidelity geopotentials. The spherical harmonics and the Global Point Mascon models are evaluated and compared by conducting numerical integration of space trajectories, which show that the latter model with parallel computing leads to one order of magnitude speedup without degradation of numerical integration accuracy.

Keywords Geopotential · Spherical harmonic model · Global point mascon model · GPU · Paralleled computing

1 Introduction

Fast computing of high-fidelity geopotentials is of great significance for space flight concerning satellite on-board real-time navigation, satellite formation maintenance and reconfiguration, space debris orbit prediction, etc. The precise orbits of artificial satellites around the Earth heavily rely on the high-fidelity models for describing geopotentials. In order to achieve high-precision on-board real-time navigation,

J. Liu (✉)
Technology and Engineering Center for Space Utilization,
Chinese Academy of Sciences, University of Chinese Academy of Sciences,
Beijing, China
e-mail: liujiangkai14@csu.ac.cn

W. Wang · Y. Gao · L. Shu
Technology and Engineering Center for Space Utilization,
Chinese Academy of Sciences, Key Laboratory of Space Utilization,
Beijing, China

J. Sun et al. (eds.), *China Satellite Navigation Conference (CSNC) 2016 Proceedings: Volume III*, Lecture Notes in Electrical Engineering 390,
DOI 10.1007/978-981-10-0940-2_8

dynamic filtering with orbit prediction by computing high-fidelity geopotentials is demanded [1], which is also indispensable for implementing the high-precision formation maintenance and reconfiguration and for precisely predicting the orbits of the huge number of space debris. The spherical harmonics (SH) model is currently used for computing geopotentials, which is an analytic model represented by the coefficients of spherical harmonic expansion of gravity anomalies [2]. The spherical harmonic coefficients reflect the distribution, motion, and changing condition of matters inside the Earth, which is crucial to the research and application in geodesy, oceanography, and space science. Because the SH model is represented by a series of expansions, the computation of geopotentials up to a prescribed order of expansion is convenient to accommodate appropriate precision levels of demanding orbital prediction. Therefore, the SH model is widely used and paid much attention by specialists of astrodynamics. The life-off of satellites for gravity survey starts a new epoch for the development of modeling high-fidelity geopotentials, and currently the SH model has been updated to a degree of an order of 200 and a space resolution of 100 km [3]. However, the higher order means heavier computational burden and lower computation efficiency when the SH model is used, thus making it impossible to use the high-order SH model for on-board navigation or space debris tracking. Fast algorithms of computing the high-order SH model need to be developed.

Improving the performance of a single processor is a choice to improve computing efficiency. However, with the gradually invaliding of Moore's law, this way is coming to the end. An alternative way is parallel computing, which means that using more than one processor to solve a same problem could often achieve a speedup based on the number of processors used. Whether the problem could be split into separate parts and solved by multiple processors simultaneously is the key to obtain significant speedup. With the development of models to describe geopotentials, models implemented by parallel computing have been proposed, for example, the global point mascon model and the interpolation model.

The global point mascon model (or Point Masses model) is based on an assumption: a collection of localized mass elements is an approximate model to enable computation of geopotentials. The mass elements are usually fixed on a spherical face buried below the surface [4] to reduce singularity [5]. Many other works have been done to improve the point mascon model [6, 7]. By using GPU in the modeling and computing process, Ryan and Nitin significantly reduced the number of points needed and published their model [8]. Splitting the geopotential computation into thousands of simple two-body problems, the model can be computed densely and suitable to parallelism, though it loses the physical information of the central planet and could not be used in orbit designing and analysis. The interpolation model is essentially first proposed by Junkins [9], which computes and saves geopotential on the grid points surrounding the central planet, and obtains the geopotential at a target point by interpolation. The interpolation model can achieve orders of magnitude in speedup over the SH model while still preserving the accuracy and robustness. However, that is based on one to two orders of magnitude raised in memory though much work has been done [10–12].

Currently, heterogeneous computing architecture (HSA) systems utilize multi-core processors and customized accelerated coprocessors have become the developing direction of high performance computing architecture [13]. With great advantages of GPU's powerful computational capabilities, high performance/price, and high performance/power consumption ratio CPU/GPU heterogeneous architecture has been used extensively and gets impressive acceleration in different areas [14]. Compute unified device architecture (CUDA) and GPU supporting double precision operation released by NVIDIA company makes it easier to program parallelly in scientific computing.

A Global Point Mascon Model and its paralleled algorithm implemented with CUDA and GPU are presented in this paper. The results of space trajectories' numerical integration using gravitional accelerations computed by the model show that the speed up of integration in high-fidelity gravity field could be one order of magnitude relative to traditional SH model.

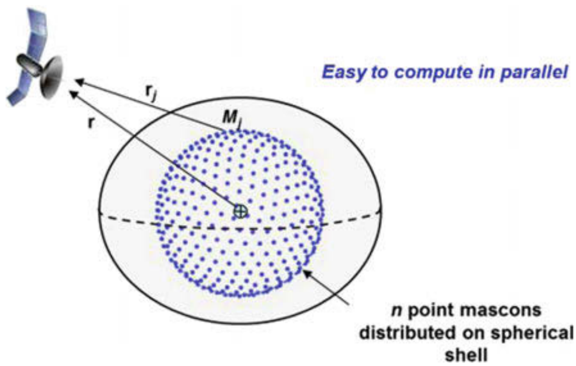
2 The Global Point Mascon Model of the Earth

The global point mascon (PMC) model is depicted in Fig. 1. A total of N point mascons with the mass of M_j distribute uniformly on a sphere with a fixed radius and its central coincides with the mass central of the Earth and is buried below the surface. $\mathbf{r}$ is the position of target point in the body fixed coordinate system, $\mathbf{r}_j$ is the target point position relative to the jth mascon.

The PMC model could be represented as the sum of two-body potential, J_2 potential (an unnormalized oblateness parameter) and the potentials that are caused by n point mascons:

$$U_{\mathrm{PMC}}(\mathbf{r}) = U_{2\mathrm{B}+J_2} + \sum_{j=1}^{n} \mathrm{GM}_j / \left|\mathbf{r} - \boldsymbol{\rho}_j\right| \tag{1}$$

Fig. 1 A sketch of the global point mascon model



where GM_j is the gravitational parameter for the jth mascon, $\boldsymbol{\rho}_j$ is the location of the jth mascon, and U_{2B+J_2} is the potential due to the two-body plus J_2 terms

$$U_{2B+J_2} = \frac{GM_E}{r}\left[1 - J_2\left(\frac{R_E}{r}\right)^2\left(\frac{3z^2}{2r^2} - \frac{1}{2}\right)\right]. \tag{2}$$

where r is the module of $\mathbf{r}$, z is the third Cartesian component of $\mathbf{r}$. GM_E, R_E, and J_2 values come from the GGM02C [15] SH solution and are the gravitational parameter, radius, and unnormalized oblateness parameter for the Earth, respectively.

The distribution of mascon is uniform, so we can fix the mascon locations $\boldsymbol{\rho}_j$ beforehand. The model fitting method estimates only the associated mascon gravitational parameters. We use the geopotential, which is computed by SH model GGM02C, as the pseudo-observations. The observations are taken at surface locations distributed uniformly across the Earth. Once the locations of mascons are fixed and the number of observations exceeds the number of mascons, the optimal estimation problem can reduce to the weighted linear least-squares problem with performance index J:

$$J = \frac{1}{2}\sum_{i=1}^{m} w_i \varepsilon_i^2 = \frac{1}{2}\sum_{i=1}^{m} w_i\left[U_{\mathrm{PMC}}(\boldsymbol{\eta}_i) - U_{\mathrm{SH}}(\boldsymbol{\eta}_i)\right]^2. \tag{3}$$

where m is the number of observations; $\boldsymbol{\eta}_i$ is the location of the ith observations; U_{SH} is the observations and is the potential computed by spherical harmonics for $\boldsymbol{\eta}_i$; U_{PMC} is the observation model and is the potential computed by PMC model; ε_i and $\boldsymbol{\eta}_i$ are the residual and location of the ith observation, respectively.

It is well known that two-body and J_2 have tremendous impact on the whole geopotential. On the other hand, the computational burden of two-body plus J_2 terms is not large. So the Earth PMC model fits only the geopotential terms except the two-body plus J_2 contribution. After the estimation process, the gravitational parameters are determined for different degree and order of spherical harmonics model.

According to the PMC model of Ryan [8] proposed, the models with different degree and order are listed in Table 1. As the degree increases, the number of mascons increases nonlinearly. For a 156 degree model, the mascons number exceeds 30 thousands.

Table 1 Point mascon models and their corresponding fields

Model descriptor	Size of SH fitting field ($d \times d$)	No. of mascons (n)
PMC11n240	11×11	240
PMC33n1920	33×33	1920
PMC71n7680	71×71	7680
PMC156n32720	156×156	30,720

3 GPU Parallel Computing

3.1 Introduction of CUDA Programming

The CUDA programming model uses CPU as the host, and GPU as device. CPU is responsible for logical tasks while GPU undertakes the works of highly parallelism. The fundamental unit for GPU programming is thread [16]. As shown in Fig. 2, a number of threads form the block and lots of blocks become one grid which is used to accomplish a task. The fundamental unit for work arrangement is kernel function, all threads belong to one grid execute one same kernel function.

The kernel function is defined as *kernelFunction* ⋘*dimGrid, dimBlock* ⋙ (*Para*), where *dimGrid* is the number of blocks and *dimBlock* is the number of threads per block.

The CUDA device has several memory spaces include global, local, shared, texture, and registers [17], which have different characteristics as shown in Table 2. Of these different memory spaces, global memory is the most plentiful (typically more than 2 GB) and can be accessed by any thread, while shared memory is relatively less (typically 48 kB per block) and can only be accessed by the threads pertaining to the same block. By accessing the values of registers like *blockDim,*

Fig. 2 CUDA's heterogeneous programming

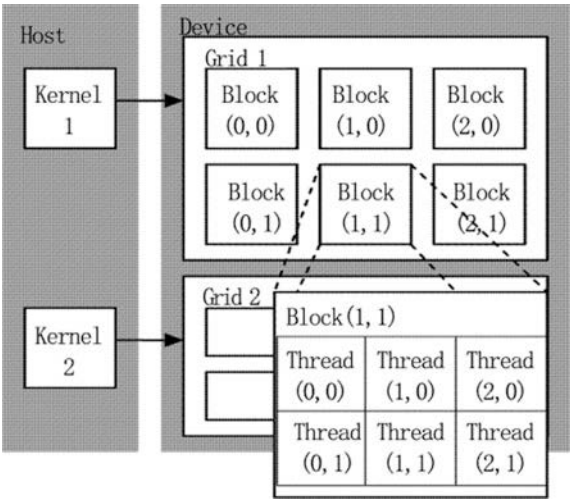


Table 2 Salient features of device memory

Memory	Location on/off chip	Access	Scope	Lifetime
Register	On	R/W	1 thread	Thread
Local	Off	R/W	1 thread	Thread
Shared	On	R/W	All threads in block	Block
Global	Off	R/W	All threads + host	Host allocation
Constant	Off	R	All threads + host	Host allocation

blockIdx and *threadIdx,* a thread gets it's global index in a grid, and executes individual task according to the index.

Different memory type has different data bandwidth. Shared memory has a bandwidth of 1.5 TB/s while global memory only has a bandwidth no more than 190 GB/s. Thus a speedup ratio up to 7 could be achieved if the shared memory is adequately used [18]. In a general case, data that are frequently used in a same block should be loaded in the shared memory to avoid the frequently accessing to global memory and to take advantage of the 'Coalesced Access' mechanism, both of which are essential to improve performance of the programs.

3.2 Strategies to Parallel with GPU

The most time-consuming part in the PMC model is the computation of the gravity forces at the target point. Since the computation is not very large for one point mascon and any point mascon has no relation to other point mascons, it is very suitable to parallel this part of the work with GPU.

On the GPU device, threads could be considered to execute the kernel function simultaneously (actually not). First, every block loads the mass points information (including the position and mass) from the global memory and saves them in its shared memory. According to its global index, every thread in the block uses the corresponding point's information to compute the gravitational acceleration at the target point generated by the single point, and saves it to the shared memory. Second, when every thread in a block finishes its work, a synchronization is triggered, and the reduction method is applied to sum all the results in a block up into a single value and save it to the global memory. Third, a new kernel function is launched to sum the results of every block up into the final results, since it is not supported to synchronize between different blocks. The process in GPU is illustrated by Fig. 3.

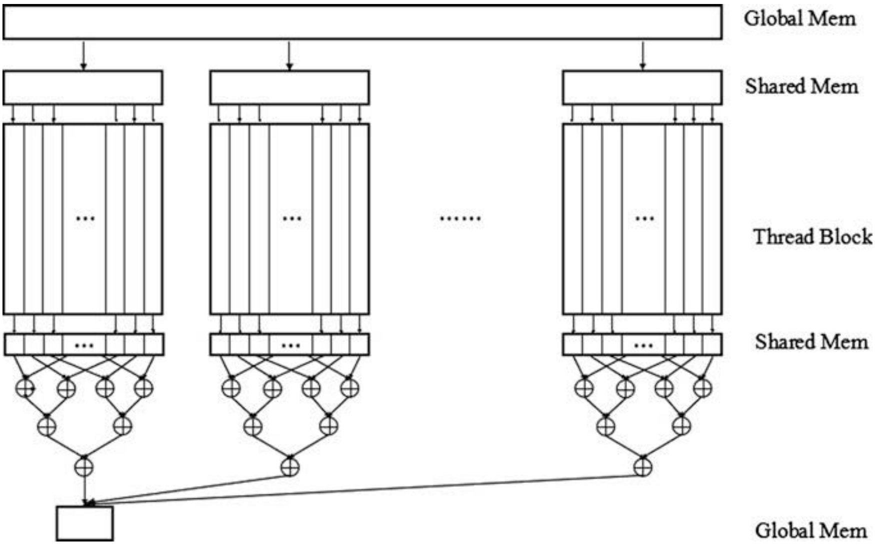


Fig. 3 The PMC model computing process on GPU

4 The Results and Analysis

A simulated scenario is built in order to verify the calculation speed and accuracy of the PMC model. Table 3 summarizes the system configuration of this scenario, in which the spacecraft orbit dynamics model is assumed to consist of the Earth's gravitational only, and the orbit is numerical integrated with the Runge-Kutta 4(5) Dormand–Prince method. The integral results are then compared with the results of corresponding SH model to perform the precision analysis.

4.1 Testing Process

Assuming that the Earth rotates at a constant speed around its spin axis, the dynamic model for the spacecraft in a simplified rotation frame can be denoted as:

Table 3 System configuration of the simulated scenario

Component type	Component
CPU	Intel Xeon E5-1620@3.7 GHz
RAM	8 GB
GPU	Tesla K20C
Operating system	Windows7 64 bit
Development environment	VS2012, CUDA7.5

$$
\begin{aligned}
&\ddot{x} = 2\Omega\dot{y} + \partial W/\partial x \ddot{y} = -2\Omega\dot{x} + \partial W/\partial y \\
&\ddot{z} = \partial W/\partial z \quad W = (\Omega^2/2)(x^2 + y^2) - U
\end{aligned} \tag{4}
$$

where x, y, z represents the position coordinates in the ECEF frame, Ω is the rotation velocity and takes the value of 7.292115×10^{-5} for Earth. U is the Earth gravity potential, and the gravity acceleration is corresponding to Eq. 5, where p represents x, y, z.

$$
\frac{\partial U}{\partial p} = \frac{\partial U_{2B+J_2}}{\partial p} - \sum_{j=1}^{n} \frac{\mathrm{GM}_j(p - p_i)}{(\boldsymbol{r} - \boldsymbol{\rho}_j)^3} \tag{5}
$$

Parallel computing process is described as Fig. 4. During the preparation stage, CPU reads the mass points' information, including position coordinates and mass, from the data file and sends them to GPU's global memory. Then the integrator on the CPU is initialed and starts working. Integrator is responsible for error controlling, step-size adjusting and the vehicle state updating. The vehicle position coordinates is sent into GPU whenever the gravity acceleration is needed to be calculated. CPU will proceed to the next step once the GPU returns the calculation result.

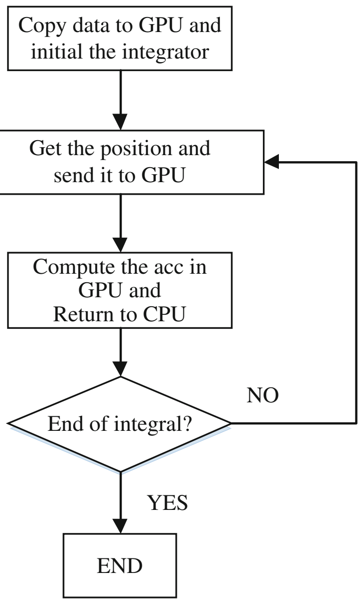


Fig. 4 Flow chart of the testing process on CPU

4.2 Testing Results

4.2.1 Calculation Speed

Three pairs of PMC models and SH models of different order (see Table 4) were chosen to investigate the calculation speed.

A MEO orbit is taken into consider, and the orbit parameters are listed in Table 5. The gravity field is calculated using two kinds of models mentioned above. The numerical integral results are shown in Figs. 5 and 6, for integration duration of 1 and 3 days, respectively.

Results show that, the computation speed-up ratio between the PMC model and SH model increases gradually as the gravitational order increases. A speed-up ratio of an order of 10 can be achieved for high order gravity field. While the length of the integral time has no significant effect to the speed-up ratio.

4.2.2 Calculation Accuracy

High efficiency of orbital numerical integration is the advantage of the PMC model, but the premise is that its calculation precision should be compatible with the same order SH model. Figure 7 shows the position offset of two orbits integrated using those two different 156 order gravity field models as show in Table 4. The position offset in *X* axis, as shown in the red line, increases linearly with time increased. The blue and green line represents the offset in *Y* and *Z* axis, respectively. The offset in these two axes increases volatility over time, but not exceeding a maximum of 2×10^{-3} m after 3 days of integration.

We further compared the integration accuracy for three different types of orbit, including LEO, GEO, and highly elliptical orbit (Molniya orbit). The orbit parameters are listed in Table 5 and the compared results are summarized in Table 6. As can be

Table 4 PMC model and SH model used in the test

Order	PMC model	SH model
33	PMC331920	GGM02C 33 × 33
71	PMC71n7680	GGM02C 71 × 71
156	PMC156n32720	GGM02C 156 × 156

Table 5 Parameters for different orbit types

Orbit type	Perigee altitude (km)	Apogee altitude (km)	Inclination (°)
LEO	400	400	97.1
MEO	1350	1350	80
GEO	35,788	35,788	40
Molniya	500	39,850	63.3

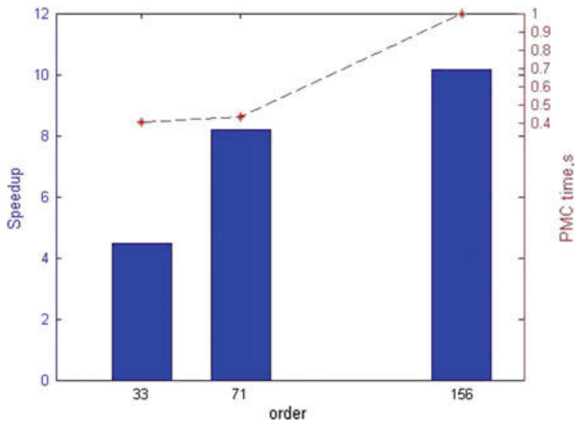


Fig. 5 PMC integral time and speed-up ratio for one day

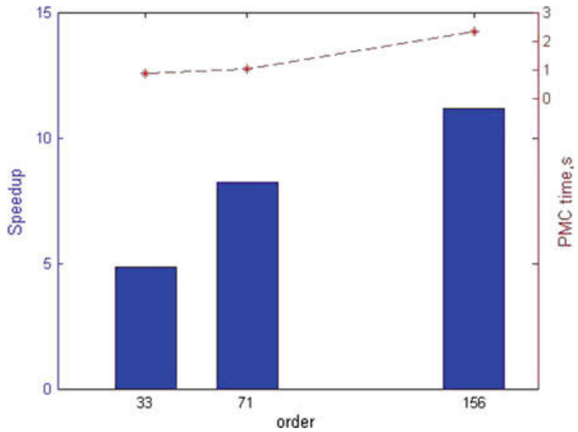


Fig. 6 PMC integral time and speed-up ratio for three days

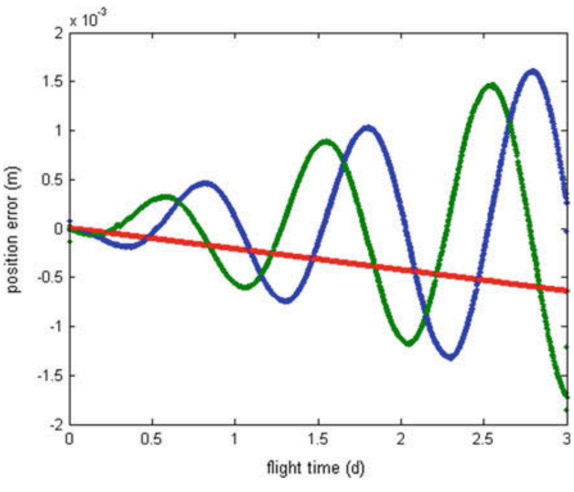


Fig. 7 Orbit position error between 2 model integrating 3 day

Table 6 Error and speed-up ratio on different orbit types

Orbit type	Max error (m)	Speed-up ratio
LEO	0.0167	15
MEO	0.0020	11
GEO	0.00065	14
Molniya	0.0020	15

seen from the results, the calculation accuracy of the PMC model meets the demand of various types of orbit, and the speed-up ratio for high order gravity field are all more than an order of 10.

4.3 Results Analysis

The paralleled PMC model is quite outstanding in accelerating the calculation of gravitational acceleration. Compared with the classical SH model, the speed-up ratio for low order (30) PMC model is about 5; this value increases to 8 when the model order increases to 70. For high-order PMC model, the speed-up ratio can achieve an order of 10. For low order gravity model, the accelerating effect of PMC model is not so prominent as low gravity field computation is relatively inadequate and, the effect of GPU acceleration is concealed by the extra time of the device initialization and data copies. Inversely, a rapid increase in calculation intensity will cause the GPU computing acceleration effect reaching more than 10 times for the high-order gravity field.

For precise orbit determination applications, the higher order integrator, such as Romberg 7(8) method, is generally used to propagate the orbit, which will cause more computation in gravitational acceleration. In this case, paralleled PMC model will result in significant speed up effect. For further work, the fast integration method in high-fidelity gravitational field would be investigated.

Besides the computational efficiency, the PMC model can ensure the accuracy of numerical integration in compatible with the same order of the SH model. For 3 days integration time, the maximum position error is less than 2 cm for a low-Earth-orbit (LEO) satellite, and less than 2 mm for a high-Earth-orbit satellite. This favorable match supports that one can use the PMC model instead of the SH model in the precise orbit determination applications.

5 Conclusions

The PMC model and GPU parallel computing technology are analyzed to realize the orbit integration of the same accuracy comparing with classical spherical model. With the emergence of space-borne GPU technology, on-board autonomous dynamic navigation and formation control would be significantly promoted. The effectiveness of PMC model in other areas, such as the gravity gradient computation based on PMC model, is still needed to be validated and will be researched later.

References

1. Wang F (2006) Theory and software development on autonomous orbit determination using space borne GPS measurements. Wuhan University
2. Ning JS (1994) The earth geopotential models and applications. Eng Surv Mapp (2):1–8
3. Zhang CD (2005) Three satellites measuring gravity field–CHAMP, GRACE, GOCE. Geophys Geochem Explor 29(5):377–382
4. Weightman JA (1967) Gravity, geodesy and artificial satellites. A unified analytical approach. Use Artif Satell Geod 2:467
5. Chao DB (1992) Global virtual particle model. Geomat Inf Sci Wuhan Univ 02
6. Wu X (2009) A point mascon model based on satellite gravity gradient data, pp 414–417
7. Li ZW (2004) Point mass model tacking account of spectrum. The PLA Information Engineering University
8. Russell RP, Arora N (2012) Global point Mascon models for simple, accurate, and parallel geopotential computation. J Guid Control Dyn 35(5):1568–1581
9. Junkins JL (2012) Investigation of finite-element representations of the geopotential. AIAA J 14(6):803–808
10. Junkins JL, Miller GW, Jancaitis JR (1973) A weighting function approach to modeling of irregular surfaces. J Geophys Res 78(11):1794–1803
11. Engels RC, Junkins JL (2012) Local representation of the geopotential by weighted orthonormal polynomials. J Guid Control Dyn 3
12. Arora N, Russell RP (2015) Efficient interpolation of high-fidelity geopotentials. J Guid Control Dyn 1–16
13. Lu FS (2011) Survey of CPU/GPU synergetic parallel computing. Comput Sci 38(3):5–9
14. http://www.nvidia.cn/object/machine-learning-cn.html. Accessed Sept 2015
15. Tapley B, Ries J, Bettadpur S, Chambers D, Cheng M, Condi F, Gunter B, Kang Z, Nagel P, Pastor R, Pekker T, Poole S, Wang F (2005) GGM02—an improved Earth gravity field model from GRACE. J Geodesy. doi:10.1007/s00190-005-0480-z
16. Nvidia C (2011) Nvidia cuda C programming guide. NVIDIA Corp 120:18
17. Cuda C (2012) Best practices guide. Nvidia Corp
18. Wilt N (2013) The CUDA handbook: a comprehensive guide to GPU programming