Domácí úkol PA081 – kometa

1 Scénář

Do blízkosti planety Jupiter vletí kometa, při explozi poblíž oběžné dráhy měsíce Callisto se rozletí na větší množství úlomků, z nichž každý pokračuje po své dráze.

Simulujeme trajektorie úlomků v gravitačním poli planety a jejich čtyř největších měsíců, včetně detekce dopadu úlomků na povrch planety nebo některého z měsíců, resp. opuštění gravitačního pole planety.

Výsledkem simulace jsou polohy a rychlosti úlomků po uplynutí zadaného času (resp. místa kolize s planetou nebo měsíci, došlo-li k ní).

2 Implementace

Implementace počítá v souřadné soustavě s planetou Jupiter v počátku, vše je redukováno pouze do dvou dimenzí, všechny čtyři měsíce se pohybují rovnoměrně po kružnicích.

Gravitační působení Slunce zanedbáváme, neřešíme kolize ani vzájemné působení úlomků.

Pro výpočet gravitačních sil uvažujeme úlomky, planetu i měsíce pouze jako hmotné body.

Na začátku (funkce `init_state()`) program vygeneruje zadaný počet úlomků, všechny vycházejí z jednoho bodu, distribuce jejich vektorů rychlosti je deterministická.

Výpočet pohybové rovnice pro jeden úlomek, tj. zejména výpočet gravitačního působení planety a měsíců je ve funkci `eval_f()`. Funkce `check_active()` kontroluje podmínky ukončení simulace pro jeden úlomek, tj. dopad na planetu nebo některý z měsíců, případně opuštění gravitačního pole.

Během výpočtu vypisuje program na standardní výstup informace o ukončení simulace jednotlivých úlomků, na závěr (po dosažení zadaného času) vypíše aktuální polohu a rychlost všech aktivních úlomků, pro neaktivní úlomky jejich polohu a rychlost v okamžiku ukončení simulace.

Vzorová implementace obsahuje pouze dopřednou Eulerovu integrační metodu. Ta je pro tyto účely nepřesná a prakticky nepoužitelná.

Parametry programu jsou následující:

-s délka časového kroku integrace (default 1 s)

-n počet úlomků (default 1)

-l délka simulace (default 10^6 s).

-m volba integrační metody

-t průběžný výpis polohy všech simulovaných těles

K vyhodnocení přesnosti spočteného výsledku je určen skript `diff.pl`. Jeho argumenty jsou dva soubory – přesměrované standardní výstupy dvou běhů simulačního programu, z nich spočte průměr a maximum vzdáleností finálních poloh odpovídajících si úlomků v obou bězích.

3 Zadání

3.1 Implementace stabilní integrační metody

Do programu doimplementujte vhodnou stabilní integrační metodu (semiimplicitní, leapfrog, Runge-Kutta, ...), tak aby při vhodné volbě délky integračního kroku (parametr `-s`) se výsledky pro 100 úlomků a délku simulace 10^6 s (`-n 100 -l 1e6`) nelišily od referenčního výpočtu (soubor `sim-rk-1e6-0.1-100.dat`) ve smyslu výstupu skriptu `diff.pl` o více než 50 000 v průměrné odchylce a zároveň o více než 1 000 000 v maximální odchylce.

Odevzdejte rozšířenou implementaci programu a specifikujte délku integračního kroku.

Hodnocení: 1 bod za funkční implementaci vyhovující zadaným podmínkám.

3.2 Optimalizovaná sekvenční implementace

Výpočet podle předchozího bodu modifikujte tak, aby byl s využitím jen jednoho jádra CPU co nejrychlejší, při dodržení uvedených kritérií na přesnost. Možné optimalizace jsou:

- umožněte kompilátoru vektorizovat
- použijte AVX instrukce (volby kompilátoru `-mavx`)
- dbejte na využití cache
- nepoužívejte dvojnásobnout přesnost (`double`) tam, kde to není třeba
- vyhněte se zbytečným výpočtům náročných funkcí (sqrt, sin, cos, ...)
- ...

Odevzdejte modifikovaný kód s podrobnými komentáři o provedených optimalizačních krocích. Opět specifikujte délku kroku a použité volby kompilátoru.

Hodnocení: 0,5 bodu za rozvahu co, jak a proč optimalizovat a funkční implementaci. Za úspěšně optimalizovanou považujeme implementaci, která využije alespoň 10 % teoretického výkonu CPU.

3.3 Paralelní implementace

Stejně jako předchozí bod s tím, že lze libovolným způsobem (ptrheads, OpenMP, MPI, ...) využít jádra CPU na jednom výpočetním uzlu (využití více uzlů není nezbytné).

Hodnocení: 0,5 bodu za návrh a funkční implementaci. Za přijatelnou je považována paralelní implementace, která na 16 jádrech zrychlí alespoň 11×.