Chapter 1

EredmÃ©nyek, kÃ¶vetkeztetÃ©sek

1.1 BeÃ©pÃtett szoftverek esetÃ©n

A **??**. fejezetben bemutattam kÃ©t beÃ©pÃtett, elÅre megvalÃ³sÃtott szoftvert, melyek a Matlab ode45 programja Ã©s a Boost - Odeint kÃ¶nyvtÃ¡r. Az elÃ©rt eredmÃ©nyek Ã©s tesztek azt mutatjÃ¡k, hogy a Matlab ode45 differenciÃ¡legyenlet megoldÃ³ja sokkal gyorsabb Ã©s hatÃ©konyabb, mint az Odeint. Ha a tesztesetekben mÃ©rt Ã¡tlagidÅket Ã¶sszehasonlÃtjuk lÃ¡thatjuk (**??**. alfejezet), hogy az ode45 $20-30$ -szor gyorsabb a OdeintnÃ©l. Ez talÃ¡n annak is kÃ¶szÃ¶nhetÅ, hogy a Matlab egy nagyon komoly szoftver, amelynek programcsomagjain mÃ©rnÃ¶kÃ¶k Ã©s programozÃ³k szÃ¡zai (vagy akÃ¡r ezrei) dolgoznak, Ãgy termÃ©szetes, hogy az algoritmusok jobban optimizÃ¡ltak Ã©s hatÃ©konyabbak az ingyenes szoftvereknÃ©l. A tovÃ¡bbiakban Ã¶sszegezzÃ¼k, hogy a kÃ©t technolÃ³giÃ¡nak milyen elÅnyei Ã©s hÃ¡trÃ¡nyai vannak vagy Ã©ppen miÃ©rt Ã©rdemes/nem Ã©rdemes hasznÃ¡lni Åket, lásd [**?**]

Matlab ode45 elÅnyei:

- nagyon egyszerÅ± a hasznÃ¡lata, nem igÃ©nyel komoly programozÃ¡si ismereteket
- kÃ¶nnyÅ± beÃ©pÃteni Ã©s Ã¶sszekÃ¶tni mÃ¡s Matlab programokkal
- a kapott eredmÃ©nyeket mÃ¡trix vagy vektor tÃpusokban tÃ©rÃti vissza, ami kÃ¶nnyÅ±vÃ© teszi az eredmÃ©nyek tovÃ¡bbi kezelÃ©sÃ©t
- az eredmÃ©nyeket grafikus felÃ¼leten azonnal meg tudjuk jelenÃteni

- jobb eredmÃ©nyeket produkÃ¡lt, mint az Odeint
- jÃ³l dokumentÃ¡lt, sok pÃ©lda van a hasznÃ¡latÃ¡ra

[main node] (1) 1; [main node] (2) [below left = 2.3cm and 1.5cm of 1] 2; [main node] (3) [below right = 2.3cm and 1.5cm of 1] 3; [draw,thick] (1) edge node (2) (2) edge node (3) (3) edge node (1); [xshift=4cm] [main node] (1) 1; [main node] (2) [right = 2cm of 1] 2; [main node] (3) [below = 2cm of 1] 3; [main node] (4) [right = 2cm of 3] 4; [draw,thick] (1) edge node (2) (1) edge node (4) (3) edge node (2) (3) edge node (4) ;

Figure 1.1: Egyszerű gráf TIKZ segítségével

Matlab ode45 hÃ¡trÃ¡nyai:

- komoly hÃ¡trÃ¡nya az Odeinttel szemben, hogy **fizetni kell a hasznÃ¡latÃ¡Â©rt**
- nagyon sok memÃ³riÃ¡t hasznÃ¡l, komolyan igÃ©nybeveszi a szÃ¡mÃ-tÃ³gÃ©p erÅforrÃ¡sait

Odeint elÅnyei:

- ingyenes Ã©s nyÃlt forrÃ¡skÃ³dÃº, hasznÃ¡lhatÃ³ szemÃ©lyi Ã©s kereskedelmi cÃ©lokra egyarÃ¡nt
- nagyon rugalmas, absztrak, Ãgy kÃ¶nnyedÃ©n vÃ¡ltoztathatÃ³ a bemeneti adatok tÃpusa vagy struktÃºrÃ¡ja
- C++ nyelven Ãrodott, tÃ¡mogatva a modern programozÃ¡si technolÃ³giÃ¡kat (Generikus programozÃ¡s, Template Metaprogramming)

Odeint hÃ¡trÃ¡nyai:

- hasznÃ¡lata nehezebb, mint a Matlab ode45 programÃ©, szÃ¼ksÃ©ges a C++ programozÃ¡si nyelv ismerete
- a kapott eredmÃ©nyekkel nem olyan kÃ¶nnyÅ± bÃ¡nni, mint a Matlab esetÃ©ben
- absztraktsÃ¡ga miatt nehÃ©z a felmerÃ¼lÅ problÃ©mÃ¡kat megoldani
- dokumentÃ¡ltsÃ¡ga jÃ³val szegÃ©nyesebb, mint a MatlabÃ©

1.2 SajÃ¡t szoftverek esetÃ©n

SajÃ¡t szoftverek esetÃ©ben sikerÃ¼lt nÃ©gy kÃ¼lÃ¶nbÃ¶zÅ technolÃ³gia segÃtsÃ©gÃ©vel megvalÃ³sÃtani a Dorman-Prince differenciÃ¡legyenlet megoldÃ³ algoritmust (lÃ¡sd **??**. alfejezet). A felhasznÃ¡lt technolÃ³giÃ¡k: Matlab, Java, C++ Ã©s Android voltak.

Ezen technolÃ³giÃ¡k kÃ¶zÃ¼l az Androidos szoftverrel kapcsolatban mÃ¡r elÅzetes fÃ©lelmeink voltak, mivel azt feltÃ©teleztÃ¼k, hogy bÃ¡rmennyire is fejlettek napjainkban a mobileszkÃ¶zÃ¶k, mÃ©gis hardveresen nem lesznek elegek ahhoz, hogy versenybe tudjanak szÃ¡llni a szÃ¡mÃtÃ³gÃ©pekkel. NÃ©hÃ¡ny teszt utÃ¡n feltÃ©telezÃ©seink beigazolÃ³dtak, lÃ¡thatjuk a **??**. alfejezet teszteseteiben is, hogy az Android szoftver mennyire gyengÃ©n teljesÃtett (mind a Motorola Moto E2, mind a Samsung Galaxy Core Prime esetÃ©ben). ÃsszehasonlÃtva a tÃ¶bbi algoritmussal az Ã¡tlagidÅket nÃ©zve $150 - 160$ - szor lassabb a JavanÃ¡l Ã©s $300 - 350$ - szer a C++ - nÃ¡l! TehÃ¡t azt a kÃ¶vetkeztetÃ©st vonhatjuk le, hogy nem Ã©rdemes mobileszkÃ¶zÃ¶kÃ¶n differenciÃ¡legyenleteket oldani, mivel tÃºlsÃ¡gosan nagy a hardver igÃ©nye Ã©s egyelÅre ezen a tÃ©ren nem kÃ©pesek tartani a lÃ©pÃ©st a szÃ¡mÃtÃ³gÃ©pekkel.

A tovÃ¡bbi hÃ¡rom technolÃ³gia kÃ¶zÃ¼l (Matlab, Java, C++) meglepÅ mÃ³don itt is a Matlab teljesÃtett a legjobban, igaz, hogy ebben az esetben mÃ¡r nem hasznÃ¡ltuk az ode45 programot, hanem megÃrtam Ã©n a sajÃ¡t fÃ¼ggvÃ©nyemet. Ez a teljesÃtmÃ©nyen is meglÃ¡tszott, mert az Ã¡ltalam Ãrt fÃ¼ggvÃ©ny nem tudott jobban teljesÃteni a tesztek alatt, mint az ode45. Mindezek ellenÃ©re $35 - 40$ - szer gyorsabb volt a JavanÃ¡l Ã©s megkÃ¶zelÃtÅleg $15 - 20$ - szor gyorsabb a C++ - nÃ¡l.

A Java Ã©s C++ szoftvereket Ã¶sszehasonlÃtva elmondhatom, hogy az esetek tÃ¶bbsÃ©gÃ©ben a C++ kÃ¶rÃ¼lbelÃ¼l 2 - szer volt gyorsabb a JavanÃ¡l, ami megfelel az elÅzetes elvÃ¡rÃ¡soknak.

Amit nagyon fontosnak tartok kihangsÃºlyozni, hogy az Ã¡ltalam megvalÃ³sÃtott C++ szoftver a tesztek sorÃ¡n nagyon jÃ³l teljesÃtett, felvette a versenyt az Odeint kÃ¶nyvtÃ¡rral Ã©s az elÃ©rt Ã¡tlagok is csak nagyon kicsivel maradnak el az Odeint Ã¡ltal produkÃ¡lt eredmÃ©nyektÅl (**??**. alfejezet).

TovÃ¡bbÃ¡ megvalÃ³sÃtottam az Euler Ã©s Runge-Kutta mÃ³dszerek pÃ¡rhuzamosÃtott vÃ¡ltozatait is CUDA technolÃ³gia segÃtsÃ©gÃ©vel. Ebben az esetben a tesztek azt mutattÃ¡k, hogy az Euler mÃ³dszer esetÃ©ben tÃ¶bbe kerÃ¼l a sok CPU Ã©s GPU memÃ³ria kÃ¶zÃ¶tti mÃ¡solÃ¡s mÅ±velete, mint amennyit nyerÃ¼nk a szÃ¡mÃtÃ¡sok elvÃ©gzÃ©se sorÃ¡n. TehÃ¡t ebben az esetben ez a fajta pÃ¡rhuzamosÃtÃ¡si megkÃ¶zelÃtÃ©s

nem Ã©ri meg. Ezzel ellentÃ©tben a Runge-Kutta mÃ³dszer esetÃ©ben a megkÃ¶zelÃtÃ©s eredmÃ©nyesnek bizonyult abban az esetben, ha az egyenletek szÃ¡ma nagy Ã©s a lÃ©pÃ©skÃ¶z kicsi. A **??** alfejezetben lÃ¡thattuk, hogy abban az esetben ha az egyenletek szÃ¡ma $n = 10$ Ã©s a lÃ©pÃ©skÃ¶z $h = 0.001$, a GPU-n megkÃ¶zelÃtÅleg 5 Ã©s fÃ©l perccel hamarabb lefutott az algoritmus, mint a CPU-n. Ezzel a pÃ¡rhuzamosÃtÃ¡si mÃ³dszerrel nem tudtuk kihasznÃ¡lni a videÃ³kÃ¡rtya Ã¡ltal nyÃºjtott maximÃ¡lis szÃ¡mÃtÃ¡si kapacitÃ¡st, de Ãgy is jelentÅs kÃ¼lÃ¶nbsÃ©get sikerÃ¼lt elÃ©rni a futÃ¡si idÅket nÃ©zve, lásd [**?**].

1.3 ÃsszessÃ©gÃ©ben

A fentieket Ã¶sszegezve elmondhatom, hogy megÃ©ri elÅre megÃrt szoftvereket vagy kÃ¶nyvtÃ¡rakat hasznÃ¡lni differenciÃ¡legyenletek megoldÃ¡sÃ¡ra. Nagyon megkÃ¶nnyÃthetik az eltÃ¼nket egyszerÅ±sÃ©gÃ¼kkel Ã©s nagy teljesÃtmÃ©nyÃ¼kkel. Viszont fontos elmondani, hogy hasznÃ¡latuk problÃ©mÃ¡kkal is jÃ¡rhat, pÃ©ldÃ¡ul fizetni kell Ã©rtÃ¼k vagy nem lehet belenyÃºlni az algoritmusokba kedvÃ¼nk szerint, esetleges fellÃ©pÅ hibÃ¡k esetÃ©n nagyon nehÃ©z lenyomozni a hiba forrÃ¡sÃ¡t (vagy szinten lehetetlen).

SajÃ¡t algoritmusok terÃ©n bÃ¡tran elmondhatom, hogy megÃ©ri a C++ technolÃ³giÃ¡t vÃ¡lasztani Ã©s ezen a vonalon tovÃ¡bbhaladni egy esetleges sajÃ¡t kÃ¶nyvtÃ¡r megÃrÃ¡sa, megvalÃ³sÃtÃ¡sa felÃ©. LÃ¡thattuk, hogy az Ã¡ltalam megÃrt C++ szoftver is felvette a versenyt az Odeint kÃ¶nyvtÃ¡rral, ami szintÃ©n C++ technolÃ³giÃ¡t alkalmaz, lásd [**?**].

Egy mÃ¡sik vonal, amit Ã©rdemes sokkal jobban felderÃteni az a CUDA technolÃ³giÃ¡val Ã©s grafikus kÃ¡rtyÃ¡val tÃ¶rtÃ©nÅ differenciÃ¡legyenlet megoldÃ¡sa. LÃ¡thattuk, hogy egy kis pÃ¡rhuzamosÃtÃ¡s is jelentÅs idÅbeli kÃ¼lÃ¶nbsÃ©get jelenthet bizonyos algoritmusok esetÃ©ben. Annak tudatÃ¡ban is, hogy a differenciÃ¡legyenletek megoldÃ¡sa nem a legjobban adatpÃ¡rhuzamosÃthatÃ³ feladatok kÃ¶zÃ© sorolhatÃ³ azt mondom, hogy megÃ©ri ezzel a technolÃ³giÃ¡val foglalkozni. Ennek kapcsÃ¡n a legnagyobb motivÃ¡ciÃ³ szÃ¡momra a jÃ¶vÅre nÃ©zve a parciÃ¡lis differenciÃ¡legyenletek pÃ¡rhuzamosÃtÃ¡sÃ¡nak megvalÃ³sÃtÃ¡sa Ã©s tanulmÃ¡nyozÃ¡sa, mivel ezeknÃ©l az egyenleteknÃ©l jobban ki lehet hasznÃ¡lni a videÃ³kÃ¡rtya rÃ¡csos szerkezetÃ©t.