Elektrotehniški vestnik 86(4): 219–224, 2019
Izvirni znanstveni članek

Dualna hitra gradientna metoda na grafičnih procesnih enotah

Iztok Ramovš[1], Samo Gerkšič[2], Uroš Lotrič[1]

[1] *Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana, Slovenija*
[2] *Institut Jožef Stefan, Jamova cesta 39, 1000 Ljubljana, Slovenija*
E-pošta: uros.lotric@fri.uni-lj.si

Povzetek. Za magnetno vodenje plazme v reaktorjih tokamak je zelo primerno prediktivno vodenje. Zaradi kratkih časov vzorčenja je ključno učinkovito reševanje sprotnega optimizacijskega problema v obliki kvadratnega programa. Optimizacija z dualno hitro gradientno metodo na centralnih procesnih enotah daje zadovoljive rezultate, s prenosom računanja na grafične procesne enote pa smo jo poskušali še dodatno pohitriti. Pri programiranju z ogrodjem OpenCL sta bili velik izziv iterativna narava dualne hitre gradiente metode in relativna majhnost problema. Izračune smo uspešno prenesli na grafično procesno enoto – pri približno trikratni pohitritvi smo dosegli skoraj enako natančnost izračunov. Poskusi kažejo, da je izvajanje programov na grafičnih procesnih enotah lahko učinkovito tudi v primerih, ko so le-te zaradi narave problema slabo izkoriščene.

Ključne besede: prediktivno vodenje, kvadratno programiranje, dualna hitra gradientna metoda, grafične procesne enote, OpenCL

A Dual Fast Gradient Method for the Graphics Processing Units

A model predictive control scheme gives good results in the plasma shape and current control of the tokamak reactors. Short sampling intervals require an efficient solution to a real-time optimization problem presented in terms of a quadratic program. On a central processing unit, satisfactory results are achieved using a dual fast gradient method. Employing the implemented dual fast gradient method on a graphics processing unit further shortens the computation time. The major challenges in OpenCL programming are the iterative nature of the dual fast gradient method and the relatively small size of the problem. Our experiment shows that the computation on a graphics processing unit is approximately three-times faster with almost the same output accuracy compared to other methods and that a graphics processing unit implementation can be advantageous even when a graphics processing unit cannot be fully exploited.

Keywords: predictive control, quadratic programming, dual fast gradient method, graphics processing units, OpenCL

1 Uvod

Tokamak je naprava toroidne oblike z vakuumsko komoro, v kateri ionizirane delce plazme zadržujemo z magnetnim poljem. Pri dovolj visoki temperaturi, tlaku in zadrževalnem času delcev v plazmi lahko pri trkih delcev dosežemo jedrsko fuzijo oziroma zlivanje jeder, najlaže dosegljivo pri trkih med jedri devterija in tricija, pri čemer se sprošča energija. Cilj razvoja tokamakov je fuzijska elektrarna, ki bi sproščeno toplotno energijo na ekonomičen način prek turbin in generatorjev pretvorila v električno.

Plazmo omejujemo in oblikujemo z velikimi magnetnimi tuljavami. Zaradi velikega števila reguliranih spremenljivk in regulirnih signalov ter hitre dinamike procesa je regulacija, še posebno pri manjših napravah, zelo zahtevna [1]. Eden od pristopov k regulaciji oblike preseka in toka plazme je prediktivno vodenje (angl. Model Predicitive Control) [2], [3]. Prediktivni regulator določa vrednosti regulirnih količin s sprotno optimizacijo cenilne funkcije, pri čemer prihodnje vrednosti procesnih signalov napoveduje z uporabo modela procesa. V nasprotju s klasičnimi regulacijskimi pristopi, kot je sorodni linearni kvadratični regulator, omogoča sistematično upoštevanje omejitev na procesnih signalih. V industrijski praksi so najbolj razširjeni prediktivni regulatorji na podlagi kvadratne cenilne funkcije in linearnih modelov ter omejitev, ki privedejo do problema sprotne optimizacije v obliki kvadratnega programa (angl. quadratic program). Prediktivni regulatorji so se v preteklosti zelo izkazali pri vodenju procesov z množico reguliranih in regulirnih količin ob upoštevanju procesnih omejitev, vendar so bili uporabni le za procese s počasno dinamiko, v zadnjem času pa z razvojem računalniške tehnologije in algoritmov za hitro sprotno optimizacijo prodirajo tudi na podro-

Prejet 28. marec, 2019
Odobren 11. avgust, 2019

čja, kjer so zahtevani krajši časi vzorčenja [4].

Pri regulaciji oblike preseka in toka plazme s prediktivnimi regulatorji so bili doseženi spodbudni rezultati [2], [5]. Poskusi so pokazali, da je mogoče z izbiro ustrezne optimizacijske metode in poenostavitvami regulacijskega problema na enem jedru centralne procesne enote doseči računski čas sprotne optimizacije nekaj milisekund, dovolj kratek za uporabo v velikem reaktorju tokamak Iter. Nov regulacijski pristop pa je treba pred uporabo na veliki napravi preizkusiti na manjših reaktorjih, kjer je dinamika sprememb v plazmi bistveno hitrejša. Za ta namen obstoječi pristop ni dovolj hiter, zaradi iterativne narave in relativne majhnosti problema pa tudi ni mogoče pričakovati hitrejšega izvajanja na večprocesorski centralni procesni enoti s klasičnim večnitnim pristopom. Ena od rešitev je izvedba prediktivnega regulatorja na računsko zelo zmogljivih grafičnih procesnih enotah. Računanje na grafičnih procesnih enotah se je močno razmahnilo predvsem na področjih, za katera je značilna zelo visoka stopnja podatkovnega paralelizma, na primer pri podatkovnem rudarjenju [6], analizi slik [7] in fizikalnih simulacijah [8].

V zadnjem desetletju lahko opazimo znaten napredek na področju hitrih sprotnih optimizacijskih metod kvadratnega programiranja, zlasti pri metodah na podlagi množice aktivnih omejitev (angl. active set) [9], notranje točke (angl. interior point) [10] in proksimalnih metodah prvega reda, kamor se uvršča hitra gradientna metoda [11], [12]. Osredotočamo se na hitro gradientno metodo, ki se sicer ne odlikuje glede reda konvergence v bližini optimuma, vendar se je izkazala za učinkovito pri izvedbi prediktivnih regulatorjev, kjer je proksimalni operator izvedljiv s preprosto projekcijo in je zahtevana točnost rešitve relativno nizka. Poleg tega je izvedbeno relativno preprosta in zato primerna za prenos v manj konvencionalna izvedbena okolja. Ker so v regulacijskem problemu prediktivnega regulatorja prisotne omejitve na izhodnih signalih, ki ne omogočajo preproste projekcije v prostoru primalne optimizacijske spremenljivke v zgoščeni obliki kvadratnega programa, je treba uporabiti dualno obliko metode.

V tem delu opisujemo učinkovito izvedbo in ovrednotenje reševanja kvadratnega programa, ki izhaja iz problema sprotne optimizacije prediktivnega regulatorja oblike preseka in toka plazme za reaktor tokamak Iter, z uporabo dualne hitre gradientne metode na grafični procesni enoti z ogrodjem OpenCL. V naslednjem poglavju bomo predstavili algoritem dualne hitre gradientne metode. V tretjem poglavju bomo opisali ogrodje OpenCL in podrobnosti prilagoditve algoritma za izvajanje na grafičnih procesnih enotah. V četrtem poglavju bomo predlagano izvedbo za grafične procesne enote primerjali z obstoječimi izvedbami za centralne procesne enote v natančnosti in hitrosti izvajanja. V sklepu pa bomo strnili glavne ugotovitve in ideje za nadaljnje delo.

2 Dualna hitra gradientna metoda

Dualna hitra gradientna metoda (angl. dual Fast Gradient Method, dFGM) je ena od učinkovitih metod za iskanje optimalne vrednosti cenilne funkcije pri optimizacijskih problemih kvadratnega programiranja. Gre za iterativni algoritem, ki z gradientnim spustom išče maksimum Lagrangeve dualne funkcije. Ker je optimizacijska funkcija konveksna, je maksimum dualne funkcije enak minimumu osnovne cenilne funkcije.

Osnovni optimizacijski problem, zapisan v programskem ogrodju Matlab, je bil s pomočjo orodja QPgen [12] predelan v reševanje optimizacijskega problema z metodo dFGM v jeziku C. Prečiščena psevdokoda metode dFGM je predstavljena na sliki 1. Vhod v metodo je s Kalmanovim filtrom

function **u** = dFGM(**x**, **r**)	
lb, **ub** = ref_adjust(**r**)	[99], [99] = ([21])
lb = **e** ∘ **lb**	[99] = [99] ∘ [99]
ub = **e** ∘ **ub**	[99] = [99] ∘ [99]
s = **L2** × **x** + **l1**	[99] = [99×81] × [81] + [99]
se = **s** ∘ **e**	[99] = [99] ∘ [99]
l, **v** = **0**	[99], [99] = [99]
q = **Q2** × **x** + **q1**	[33] = [33×81] × [81] + [33]
w = **R2** × **x**	[33] = [33×81] × [81]
for k = 1: K	
z = **M** × **v** + **q**	[33] = [33×99] × [99] + [33]
h = **C** × **z** + **v**	[99] = [99×33] × [33] + [99]
p = **h** + **se**	[99] = [99] + [99]
p = clip_soft(**p**, **lb**, **ub**)	[99] = ([99], [99], [99])
p = **e** ∘ (**ei** ∘ **p** – **s**)	[99] = [99] ∘ ([99] ∘ [99] – [99])
lp = **l**	[99] = [99]
l = **h** – **p**	[99] = [99] – [99]
p = (**l** – **lp**) × $t[k]$	[99] = ([99] – [99]) × [1]
vp = **v**	[99] = [99]
v = **p** + **l**	[99] = [99] + [99]
l, **v** = restart(**l**, **lp**, **vp**)	[99], [99] = ([99], [99], [99])
u = **R** × **z** + **w**	[33] = [33×33] × [33] + [33]

Slika 1: Osrednji del metode dFGM za regulacijo toka plazme. Na desni ob algoritmu so prikazane velikosti podatkovnih struktur. Navaden produkt je označen s simbolom ×, Hardamandov pa s simbolom ∘.

ocenjeno stanje sistema v vektorju **x** ter vektor referenčnih vrednosti **r**, izhod pa vektor vrednosti regulirnih količin **u**. Vektor **r** vključuje referenčne vrednosti za 21 reguliranih izhodov, ki vključujejo 11 tokov v superprevodnih poloidnih tuljavah, tok plazme in 9 elementov zgoščenega vektorja odmikov roba plazme od stene komore. Regulirne količine v vektorju **u** so napajalne napetosti 11 tuljav, izračunane v treh intervalih na napovednem obzorju.

Matrike in vektorji, izpisani v sivi barvi, opisujejo model procesa in nekatere omejitve in se med izvaja-

njem ne spreminjajo. Potek iskanja optimalne rešitve lahko razdelimo na začetni del, zanko in končni del. Časovno najzahtevnejša je zanka, v kateri iterativno iščemo najboljšo rešitev. Število iteracij K je v našem primeru določeno vnaprej. V vsaki iteraciji se izvajajo operacije množenja vektorjev in matrik ter seštevanje, odštevanje in računanje Hardamandovih produktov nad vektorji. Funkcija ref_adjust preoblikuje vektor **r** v obliko, ki jo funkcija clip_soft uporablja za omejevanje elementov vektorja **p**. Funkcija restart iz skalarnih produktov med vhodnimi vektorji odloči, ali nove vrednosti vektorjev **l** in **v** sprejme ali zavrne.

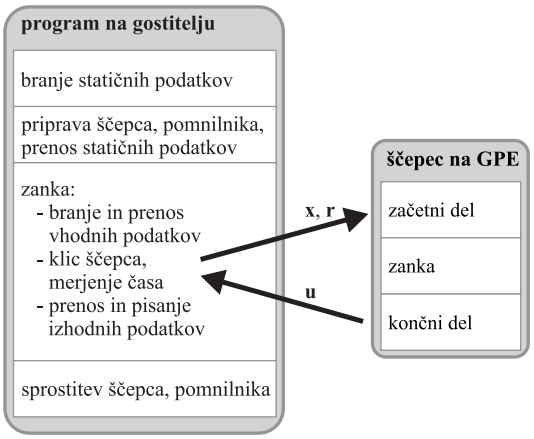
Zaradi iterativne narave algoritma lahko izvajanje pohitrimo samo s paralelizacijo operacij nad matrikami in vektorji znotraj iteracije. Tu pa se zaplete, saj so matrike in vektorji relativno majhni, razporejanje niti pa se na standardnih večjedrnih procesorjih izvaja na nekajmilisekundni časovni skali. Zato s klasičnim večnitnim pristopom ne moremo pričakovati pohitritve naloge, ki v eni niti potrebuje za izvedbo le nekaj milisekund.

3 Prilagoditev metode dFGM za izvajanje na GPE

Moderne grafične procesne enote so zelo primerne za izvajanje računsko intenzivnih operacij, saj je delež procesnih elementov enot na čipu veliko večji kot pri navadnih procesorjih. Povečanje deleža gre predvsem na račun kontrolnih elementov, zato so grafične procesne enote izvrstne pri izračunih, za katere sta značilna masivni podatkovni paralelizem in majhno število vejitev v programu. Programer vidi grafično procesno enoto kot soprocesor, ki ga lahko uporabi za pohitritev izvajanja računsko intenzivnih delov algoritma. Eno od ogrodij za programiranje grafičnih procesnih enot je OpenCL [13], ki razvijalcem med drugim ponuja standardiziran in učinkovit dostop do grafičnih procesnih enot različnih proizvajalcev.

Izvajanje programa OpenCL poteka delno na gostitelju in delno na grafični procesni enoti (GPE). Gostitelj med drugim poskrbi za vzpostavitev komunikacije z grafično procesno enoto, prenos podatkov na grafično procesno enoto in iz nje ter za zagon ščepca (angl. kernel). Na grafični procesni enoti je izračun razdeljen na množico niti, ki izvajajo ščepec. Pri pisanju ščepcev moramo biti pozorni na omejitve, ki izhajajo iz hierarhične arhitekture grafičnih procesnih enot. Procesni elementi so razdeljeni med več računskih enot, ki imajo tudi svoj lokalni pomnilnik. Vse niti iz iste delovne skupine se izvajajo na isti računski enoti in se lahko sinhronizirajo prek lokalnega pomnilnika. Ogrodje OpenCL nam ne zagotavlja, da se bo ščepec vedno izvajal na istih računskih enotah. Gostitelj lahko piše in bere samo v globalni pomnilnik grafične procesne enote.

Pri prediktivnem regulatorju toka in oblike preseka plazme so podatkovne strukture dokaj majhne. Zato moramo za učinkovito izvajanje na grafičnih procesnih enotah kar se da zmanjšati prenose podatkov med gostiteljem in grafično procesno enoto in število klicev ščepca. Iz gostitelja zato vse statične strukture (matrike, vektorje, tabelo vrednosti $t[k]$), skupaj približno 165 kB, prenesemo na grafično procesno enoto samo enkrat, ravno tako samo enkrat vzpostavimo ščepec (slika 2). Zaradi narave

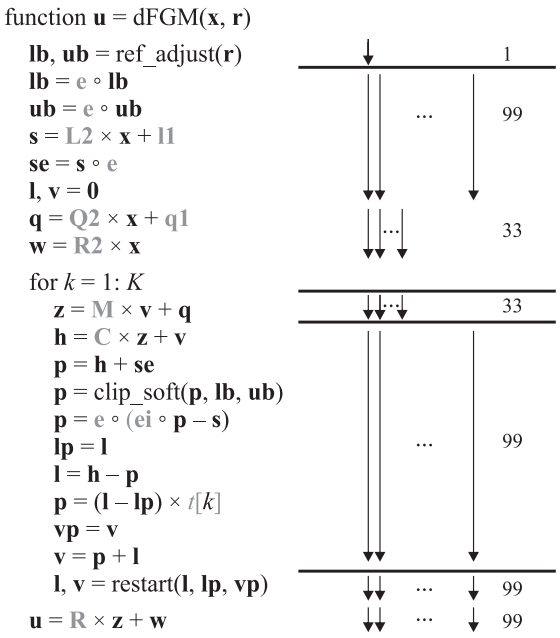


Slika 2: Izvajanje metode dFGM v OpenCL

problema pa moramo v vsakem koraku regulatorja na grafično procesno enoto prenesti vektorja **x** in **r** (408 B), izvesti ščepec in prenesti vektor **u** (132 B) nazaj na gostitelja.

Dostop do lokalnega pomnilnika je nekaj velikostnih redov hitrejši od dostopa do večjega globalnega pomnilnika grafične procesne enote, zato v začetnem delu ščepca vse ključne statične strukture prenesemo iz globalnega v lokalni pomnilnik procesne enote.

Med izvajanjem ščepca moramo poskrbeti, da hkrati dela čim več niti, seveda pa moramo na ključnih delih kode postaviti sinhronizacijske ovire, ki poskrbijo, da vse niti zaključijo eno fazo, preden nadaljujejo z drugo. Ovire so na sliki 3 označene z vodoravnimi črtami. Delo z ovirami je mogoče samo med nitmi na isti računski enoti, zato ščepec inicializiramo za delo z eno samo delovno skupino niti. Matrike in vektorji imajo v eni dimenziji največ 99 elementov, zato število niti v delovni skupini omejimo na 128, večkratnik najmanjšega števila hkrati aktivnih niti. Pri računanju je vsaka nit zadolžena za izračun dela rezultata – pri produktu matrike in vektorja sekvenčno izračuna skalarni produkt med vrstico matrike in vektorjem, pri računanju vsote, razlike in Hardamandovega produkta nad vektorji pa izvede operacijo nad istoležnimi elementi. Računanje skalarnega produkta v funkciji restart izvede ena nit. Pri kodiranju smo se v želji po čim hitrejšem izvajanju ščepca izognili klicem pomožnih funkcij.



Slika 3: Ilustracija sinhronizacijskih ovir in števila niti pri izvedbi metode dFGM na grafični procesni enoti

Dodatno pri izračunih na grafičnih procesnih enotah vse podatke predstavimo v plavajoči vejici z enojno natančnostjo, saj so operacije v plavajoči vejici nad operandi v enojni natančnosti veliko hitrejše kot nad operandi v dvojni natančnosti.

4 Rezultati

Regulator, prilagojen za delo z grafičnimi procesnimi enotami, smo primerjali z regulatorji za centralno procesno enoto glede natančnosti izračunov in hitrosti izvajanja na več testnih primerih.

Testne primere smo dobili iz obstoječega simulacijskega okolja, ki je sestavljeno iz simulatorja za tokamak reaktor, pripravljenega v okolju Matlab Simulink, in prediktivnega regulatorja, pripravljenega za centralno procesno enoto [14]. Testni primeri zajemajo štiri tipe motenj (MD, ELM, H-L/L-H i VDE) v treh različnih ravnotežnih modelih (t80, t90 in t520). Za vsak testni primer smo posneli 300 zaporednih korakov, pri čemer smo v vsakem koraku shranili vektorje **x**, **r** in **u**.

Vse meritve smo opravili na računalniku s procesorjem Intel Core i7-6700K in grafično procesno enoto nVidia GeForce GTX 1070 s frekvenco ure 1683 MHz.

4.1 Natančnost izračunov

Na grafični procesni enoti (GPE) smo operande predstavili v plavajoči vejici z enojno natančnostjo (EN), na centralni procesni enoti (CPE) pa v enojni natančnosti ali dvojni natančnosti (DN). Za mero natančnosti izračuna regulacijskega koraka smo vzeli mero

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{u_i - u_i^{\mathrm{ref}}}{a_i}\right)^2} \quad , \tag{1}$$

kjer je n število elementov u_i vektorja **u**, elementi a_i pa pomenijo razpon območja, v katerem se lahko nahaja element izhodnega vektorja. Za referenčne vrednosti u_i^{ref} smo vzeli izračune nad operandi v dvojni natančnosti po $K = 100.000$ korakih optimizacijske zanke na sliki 1.

V tabeli 1 so zbrane povprečne vrednosti E_{avg} in maksimalne vrednosti E_{max} napak E, izračunanih na vseh 12×300 vzorcih. Na napako najbolj vpliva

Tabela 1: Natančnost izračunov glede na procesno enoto, predstavitev operandov in število korakov K.

izvedba	k	E_{avg} $[10^{-5}]$	E_{max} $[10^{-3}]$
CPE+DN	10.000	0,90	0,14
	1.000	8,72	2,97
	250	10,43	6,73
CPE+EN	10.000	0,97	0,21
	1.000	8,74	2,98
	250	10,44	6,73
GPE+EN	10.000	0,97	0,23
	1.000	8,74	2,98
	250	10,44	6,73

število korakov K v optimizacijski zanki, veliko manj pa predstavitev operandov. Napake pri izračunih z operandi v enojni natančnosti na centralni procesni enoti in grafični procesni enoti so konsistentne. Razliko v E_{max} pri 10.000 korakih optimizacijske zanke lahko pripišemo numerični napaki zaradi različnega vrstnega reda izvajanja izračunov, ki je pri paralelni izvedbi na grafični procesni enoti lahko precej drugačen kot pri zaporedni izvedbi na centralni procesni enoti.

4.2 Časi izvajanja

Pri izvajanju metode dFGM na grafičnih procesnih enotah moramo za vsak korak regulacije prenesti vektorja **x** in **r** na grafično procesno enoto, izračunati vektor **u** in ga prenesti nazaj na gostitelja. Za vse prenose porabimo manj kot 2 μs.

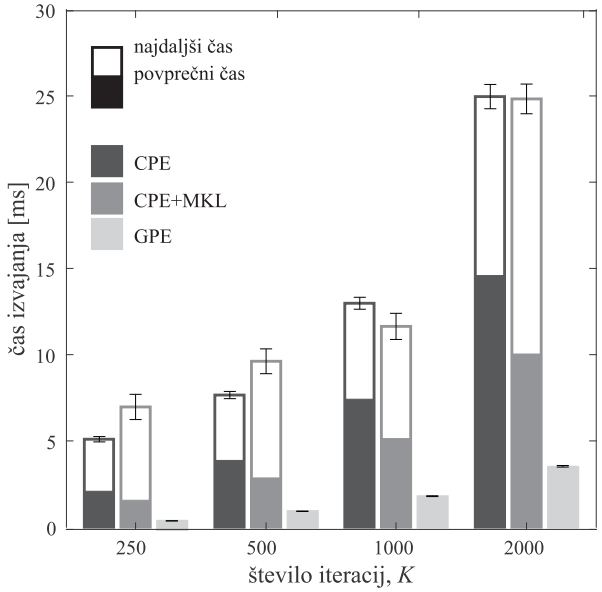
Med izvajanjem metode dFGM računske enote na grafični procesni enoti veliko uporabljajo statične podatkovne strukture, ki jih ob vzpostavitvi shranimo v glavni pomnilnik na grafični procesni enoti. V tabeli 2 vidimo, da je pri računanju na grafičnih procesnih enotah dobro uporabljati deljeni pomnilnik. Kratki dostopni časi do deljenega pomnilnika več kot odtehtajo kopiranje podatkov iz glavnega v deljeni pomnilnik ob vsakem zagonu ščepca.

Čase izvajanja metode dFGM na grafični procesni enoti smo primerjali z izvajanjem na centralni

Tabela 2: Časi izvajanja metode dFGM na grafičnih procesnih enotah glede na uporabljeni pomnilnik. Predstavljeni so meritve pri 1.000 korakih optimizacijske zanke. Napaka meritev 0,03 ms.

pomnilnik GPE	t_{avg} [ms]	t_{max} [ms]
globalni	6,01	6,19
deljeni	1,76	1,78

procesni enoti in na centralni procesni enoti s klicanjem funkcij iz knjižnice Intel MKL. Knjižnica Intel MKL [15] vključuje dobro optimizirane funkcije za matematične operacije, ki lahko izkoriščajo vektorske enote in več jeder v procesorjih. Na sliki 4 so predstavljeni povprečni časi izvajanja in najdaljši časi izvajanja za zadnjih 75 % testnih primerov, ko se delovanje procesnih enot po začetnem prehodnem pojavu stabilizira. Po pričakovanju so časi izvajanja



Slika 4: Povprečni in maksimalni časi izvajanja metode dFGM z napako meritve

na centralni procesni enoti z uporabo knjižnice MKL boljši kot brez nje. V obeh primerih so maksimalni časi izvajanja precej daljši in za obe izvedbi zelo podobni. Časi izvajanja so verjetno daljši tedaj, ko operacijski sistem izvaja druga opravila. Grafična procesna enota za izračun porabi le približno tretjino časa, potrebnega za izračun na centralni procesni enoti s knjižnico MKL.

5 Sklep

V delu smo se osredotočili na pospešitev računske izvedbe sprotne optimizacije prediktivnega regulatorja toka in oblike preseka plazme v reaktorju tokamak z uporabo grafičnih procesnih enot. Omejili smo se na obstoječi prediktivni regulator, ki za reševanje optimizacijskega problema uporablja dualno hitro gradientno metodo. Pri tem sta bili izziv predvsem iterativna narava algoritma in majhnost problema, ki sta precej zmanjšali možnosti izrabe grafične procesne enote. Kljub slabi izkoriščenosti grafične procesne enote, za izračun smo porabili manj kot 1 % razpoložljivih računskih virov, smo dobili boljše rezultate kot na centralni procesni enoti: izvajanje metode je približno trikrat hitrejše, maksimalni časi izvajanja pa ne odstopajo veliko od povprečnih. Grafične procesne enote so tako zelo zanimiva alternativa obstoječim rešitvam.

Za še hitrejše izvajanje bi lahko poskusili izboljšati upravljanje pomnilnika na grafični procesni enoti. Zaradi množice neizkoriščenih virov bi bilo verjetno mogoče hitro reševanje tudi pri kompleksnejših regulacijskih problemih. Pri tako majhnem problemu bi še boljše rezultate predvidoma dosegli, če bi namesto na grafičnih procesnih enotah ščepec izvajali na vezjih FPGA, ki omogočajo še višjo stopnjo paralelizacije izvajanja.

Ponavadi grafične procesne enote zaradi velikih računskih zmogljivosti uporabljamo za reševanje problemov, pri katerih obdelujemo velike količine podatkov. Predstavljeni primer pa je pokazal, da so grafične procesne enote lahko uporabne tudi za reševanje problemov, pri katerih so le-te precej slabo izkoriščene.

Zahvala

Raziskavo je omogočila Javna agencija za raziskovalno dejavnost Republike Slovenije v okviru programov P2-0241 - Sinergetika kompleksnih sistemov in procesov in P2-0001 - Sistemi in vodenje.

Literatura

[1] G. De Tommasi, "Plasma magnetic control in tokamak devices," *Journal of Fusion Energy*, May 2018. [Online]. Available: https://doi.org/10.1007/s10894-018-0162-5.

[2] S. Gerkšič and G. De Tommasi, "Iter plasma current and shape control using MPC," in *Proceedings of 2016 IEEE Conference on Control Applications (CCA)*, 09 2016, pp. 599—-604.

[3] B. Maljaars, F. Felici, M. de Baar, and M. Steinbuch, "Model predictive control of the current density distribution and stored energy in tokamak fusion experiments using trajectory linearizations," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 314–321, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896315025859.

[4] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066102001867.

[5] S. Gerkšič, B. Pregelj, M. Perne, M. Ariola, G. De Tommasi, and A. Pironti, "Model predictive control of Iter plasma current and shape using singular-value decomposition," *Fusion Engineering and Design*, vol. 18, pp. 158–163, 2018.

[6] D. Sluga, T. Curk, B. Zupan, and U. Lotrič, "Heterogeneous computing architecture for fast detection of snp-snp interactions," *BMC bioinformatics*, pp. 1–16, 2014.
[7] R. Češnovar, V. Risojević, Z. Babić, T. Dobravec, and P. Bulić, "A GPU implementation of a structural-similarity-based aerial-image classification," *The journal of supercomputing*, pp. 978–996, 2013.
[8] Y. Liang, X. Xing, and Y. Li, "A gpu-based large-scale monte carlo simulation method for systems with long-range interactions," *Journal of Computational Physics*, vol. 338, pp. 252–268, 2017.
[9] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: a parametric active-set algorithm forÂ quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, Dec 2014. [Online]. Available: https://doi.org/10.1007/s12532-014-0071-1.
[10] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. C. Kerrigan, and G. A. Constantinides, "Predictive control using an fpga with application to aircraft control," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1006–1017, May 2014.
[11] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time mpc with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, June 2012.
[12] P. Giselsson, "Improved fast dual gradient methods for embedded model predictive control," in *IFAC Proceedings Volumes*, vol. 47, no. 3, 2014, pp. 2302–2309.
[13] D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, *Heterogeneous Computing with OpenCL 2.0*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.
[14] I. Ramovš, "Dualna hitra gradientna metoda na grafičnih procesnih enotah," Bachelor's Thesis, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2018.
[15] *Intel Math Kernel Library. Reference Manual.* Intel Corporation, 2009.

Iztok Ramovš je študent magistrskega programa na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Med študijem se je osredotočil na področje razvoja programske opreme in sistemskih programov. Sodeloval je pri projektih podjetja Adacta in Inštituta Jožef Stefan, Odseka za sisteme in vodenje.

Samo Gerkšič je raziskovalec na Odseku za sisteme in vodenje Instituta Jožef Stefan. Raziskovalno dela na področju aplikacij naprednih metod vodenja procesov.

Uroš Lotrič je izredni profesor na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Raziskovalno dela na področju informacijske teorije in visoko zmogljivega računanja. Omenjeni področji pokriva tudi v pedagoškem procesu.