KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Nerijus Žalkauskas

Delninukų energijos suvartojimo apdorojant išretintas matricas saugomas eilutėmis modeliavimas

Magistro darbas

Darbo vadovas

doc. dr. E. Toldinas

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Nerijus Žalkauskas

Delninukų energijos suvartojimo apdorojant išretintas matricas saugomas eilutėmis modeliavimas

Magistro darbas

Recenzentas

doc. A. Venčkauskas

2008-01-14

Vadovas

doc. dr. E. Toldinas

2008-01-14

Atliko

IFM-2/4 gr. stud.

Nerijus Žalkauskas

2008-01-14

Turinys

Įvadas

Didelis energijos suvartojimas yra labai svarbi detalė sistemoms, naudojančioms baterijas: nešiojami kompiuteriai, delninukai, mobilieji telefonai ir t.t. Pradėjus naudoti objektiškai orientuotas sistemas, buvo susirūpinta energijos taupymu. Darbas su tokiomis sistemomis tampa vis labiau sudėtingesnis. Auga vartotojų poreikiai, todėl natūralu, kad imtasi gerinti sistemų darbą, bei nagrinėti jų gyvybingumą. Nevisada įmanoma optimizuoti aparatūrą bei programos kodą, kad baterijos energijos užtektų ilgam laikui. Todėl išsiaiškinus kokią įtaką daro objektiškai orientuotos sistemos baterijos gyvavimo ciklui, galima būtų padėti programuotojams kurti tokias programas, kurios vartoja mažiau energijos.

1. Užduotis

1.1. Darbo vadovas

doc. dr. Eugenijus Toldinas, Kompiuterių katedra, Kauno Technologijos Universitetas.

1.2. Temos pavadinimas

Delninukų energijos suvartojimo apdorojant išretintas matricas saugomas eilutėmis modeliavimas

1.3. Darbo vykdytojai

IFM-2/4 grupės studentas Nerijus Žalkauskas nerzal@gmail.com

1.4. Darbo sritis, objektas

Darbo sritis – mobilios informacinės sistemos. Tyrimo objektas – delniniai kompiuteriai, kurie dirba Windows mobile aplinkoje.

1.5. Darbo tikslas, sprendžiamos problemos

Naudojant mobiliuosius įrenginius viena iš svarbiausių problemų yra baterijos gyvavimo laikas. Įprastos OOP naudoja nemažai kompiuterio resursų, todėl labai greitai išsikrauna mobiliojo įrenginio baterija. Darbo tikslas – naudojant matematinius modelius, ištirti

energijos vartojimą delniniuose kompiuteriuose, bei nustatyti procesoriaus ir atminties įtaką bendram energijos suvartojimui.

1.6. Užduoties formulavimas

- Išanalizuoti sistemos apkrautumo santykį su baterijos gyvavimo ciklu.
- Išanalizuoti kitų mokslinių tyrimų rezultatus ir pateiktą medžiagą apie tyrimą, susijūsį su baterijos energijos taupymu naudojant objektiškai orentuotas sistemas.
- Aprašyti matematinius metodus ir sukurti programą jiems realizuoti bei sekti sistemos parametrus. Paleidus programą, bus stebimi sistemos parametrai: procesoriaus darbo laikas, atminties apkrautumas, baterijos energijos suvartojimas.
- Gautus duomenis išnagrinėti ir pateikti rezultatus grafiniu būdu.

2. Mobilių informacinių sistemų analizė

2.1. Analizės tikslas

Analizės tikslas – išanalizuoti mokslinius straipsnius apie energijos suvartojimą delniniuose kompiuteriuose. Ištirti metodus, bei sprendimus, kurie buvo pasiūlyti, tam, kad sumažintume bendrą energijos suvartojimą ir kurie leistų ilgiau naudoti įrenginį su išorine baterija.

2.2. Mobilių informacinių sistemų vartojimas

Informacinių sistemų vartotojai tampa mobilūs. Šiai tendencijai plisti padeda aukštu technologijų atsiradimas ir vystymasis. Šiandien be mobilių įrenginių gyvenimas yra neįsivaizduojamas. Telefonai, delniniai ir nešiojami kompiuteriai, specialūs įrenginiai naudojami kasdien. Jų vartotojas automatiškai tampa informacinės sistemos naudotoju. Tačiau, kad ir kaip būtume paženge technologijų atžvilgiu, baterijos gyvavimo laikas išliko toks pat. Baterija yra pagrindinis elementas, kuris įgalina mobilias informacines sistemas veikti. Todėl vartotojams svarbu, kad jų sistema kuo ilgiau veiktu be sustojimo.

Iš mobilių įrenginių tampa populiarūs delniniai kompiuteriai. Todėl darbe bus tiriama delninio kompiuterio baterijos gyvavimas.

2.3. Programinės įrangos palyginimas

Veikianti programa gali naudoti daug kompiuterio resursų, jeigu programa parašyta neoptimizuotai, o gali veikti ir naudodama minimalius kompiuterio resursus. Tai turėtų būti svarbu profesionaliems programuotojams, rašantiems programas**, kurios bus naudojamos** įrenginiuose su išorine baterija.

Galutinis vartotojas, kuris naudos sukurtą optimizuotą programą yra vartotojas, besinaudojantis nešiojamu ar delniniu kompiuteriu. Įrodysime, kad tiriamasis darbas taip pat netiesiogiai bus skirtas ir galutiniams vartotojams, kurie naudos programuotojų parašytas programas. Turime dvi vienodas programas, viena parašyta nesilaikant optimizavimo rekomendacijų ir naudoja daug kompiuterio resursų, o kita – parašyta optimizuotai ir minimaliai naudoja resursus. Programas naudoja vartotojai savo nešiojamuose kompiuteriuose su išorine baterija. Vieno vartotojo įrenginio baterija išsikraus daug greičiau (naudojama programa neoptimizuota), negu kito (naudojama programa optimizuota). Aišku, kad labiau patenkintas programa bus antras vartotojas, kurio įrenginio baterija laikė daug ilgiau. Įrenginys veikia ne tik ilgiau, bet ir pailginamas baterijos tarnavimo laikas.

Visi programuotojai turėtų rašyti optimizuotas programas, kurios skirtos nešiojamiems kompiuteriams. Šiuo metu daugelis programuotojų neatsižvelgia į šią problemą, manydami, jog tai nėra svarbu. [1]

Problema	Kaip viskas vyksta dabar
OOP kalbos naudoja daug resusrsų.	Ne visos programos veikia greitai. Kodas neoptimizuotas. Paleidus programas, jos naudoja labai daug sistemos resursų. Tai sekina bateriją.
Programuojama rankiniu būdu.	Rašant kodą ranka, ne visada pavyksta išvengti klaidų, kurios sukelią sistemos veiklos sutrikimus. Programuotojui tenka

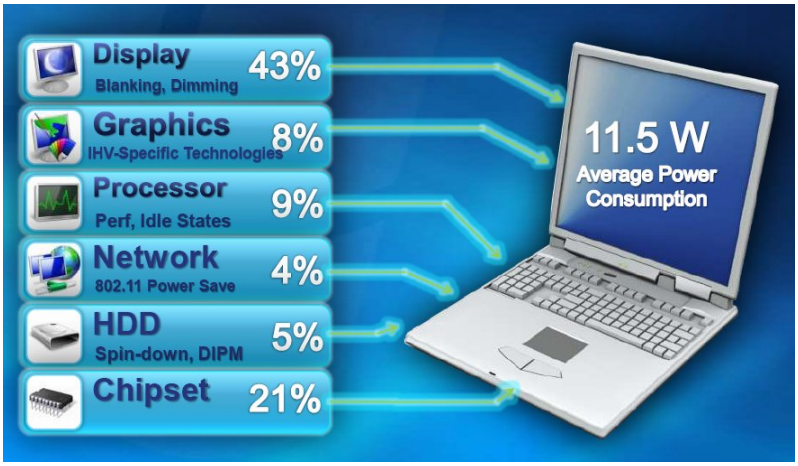
	didžiulis krūvis. Jei programa didelė, pasidalinama į kelias grupes programuotojų, tačiau klaidų vistiek sunku išvengti.
Kuo didesnė programa, tuo daugiau reikia programuotojų.	Kai programuotojų kolektyvas siekia šimtą ir daugiau narių, būtinas komunikavimas tarp programuotojų tampa neįveikiama kliūtimi. Tenka derinti tiek daug kuriamų programų sąveikų, kad nelieka laiko programuoti.
Progaminės įrangos krizės.	Tik penki procentai sukurtos programinės įrangos tampa veikiančiomis sistemomis.

Tyrimas padės programuotojams išspręsti optimizavimo problemas. Žinodami, ką geriau apkrauti, procesorių ar atmintį, parašytos programos bus geresnės, nes naudos mažiau baterijos energijos. Vartotojų delninių kompiuterių baterijos veiks ilgiau ir pailgės baterijos tarnavimo laikas.

2.4. Panašių sistemų, metodų, metodikų, įrankių analizė

2.4.1. Microsoft tyrimas

Microsoft kompanija 2007 metais atliko tyrimą: energijos optimizavimas windows platformoms. Jų tyrimo tikslas yra suteikti sistemų kūrėjams informacijos, kaip galima optimizuoti Windows Vista aplinkoje sunaudojamos energijos kiekį. Prieš pradedant tyrimą, buvo sugalvotas scenarijus. Pagal jį buvo atliekami skaičiavimai kai kompiuteris buvo ramybės būsenoje. Taip pat kai vygdė tam tikras funkcijas: buvo leidžiamas DVD ir panašiai. Atliekant testus, baterija pilnai išsikraudavo ir būdavo renkami duomenys apie suvartotą energiją. Atlikus tyrimą paaiškėjo, kiek kompiuteryje energijos suvartoja atskiros dalys. Rezultatai pateikti pirmame paveiksle.



1 paveikslas. Microsoft tyrimo rezultatai

Iš rezultatų matome, kad didžiausią dalį energijos suvartoja vaizduoklis. Todėl norint sumažinti jo suvartojamą energiją, buvo pasiūlyta kuo greičiau jį išjungti, kai vartotojas nesinaudoja kompiuteriu. Atliktas tyrimas padės sistemų kūrėjams įvertinti kiekvienos kompiuterio dalies energijos suvartojimą ir kuriant sistemas naudoti energijos taupymo politiką.

2.4.2. qSleep (mikro-sleep)

Lawrence S. Brakmo, Deborah A. Wallach, Marc A. Viredaz pasiūlė technologiją, kuri vadinasi qSleep (mikro–sleep). qSleep – tai technologija, kuri sumažina energijos suvartojimą nešiojamuose įrenginiuose [3].

Pagrindinė metodo esmė yra tokia: vietoj to, kad visada procesorius būna savo laukimo būsenos (idle) rėžime trumpais laiko momentais (mažiau nei sekundė), procesorius perjungimas į miegojimo rėžimą, kada tik tai įmanoma padaryti. Miegojimo būsena, kuri naudojama skiriasi nuo paprastos sistemos miegojimo būsenos. Pirmas skirtumas susijęs su įjungtu monitoriumi. Laikyti monitorių įjungtą, kol rodomas paveikslėlis, kuris buvo atidarytas prieš sistemai einant į miegojimo rėžimą. Antras skirtumas susijęs su sistemos pažadinimu prieš kitą operacinės sistemos suplanuotą įvykį, toks kaip branduolio aptarnavimas. Paskutinis skirtumas susijęs su sistemos pažadinimu, kaip pasirodo išorinis įvykis, toks kaip mygtuko paspaudimas ar liečiamo ekrano palietimas. Pagrindinis visų skirtumų tikslas – padaryti taip, kad vartotojas neįtartų, jog procesorius miega. Kiek

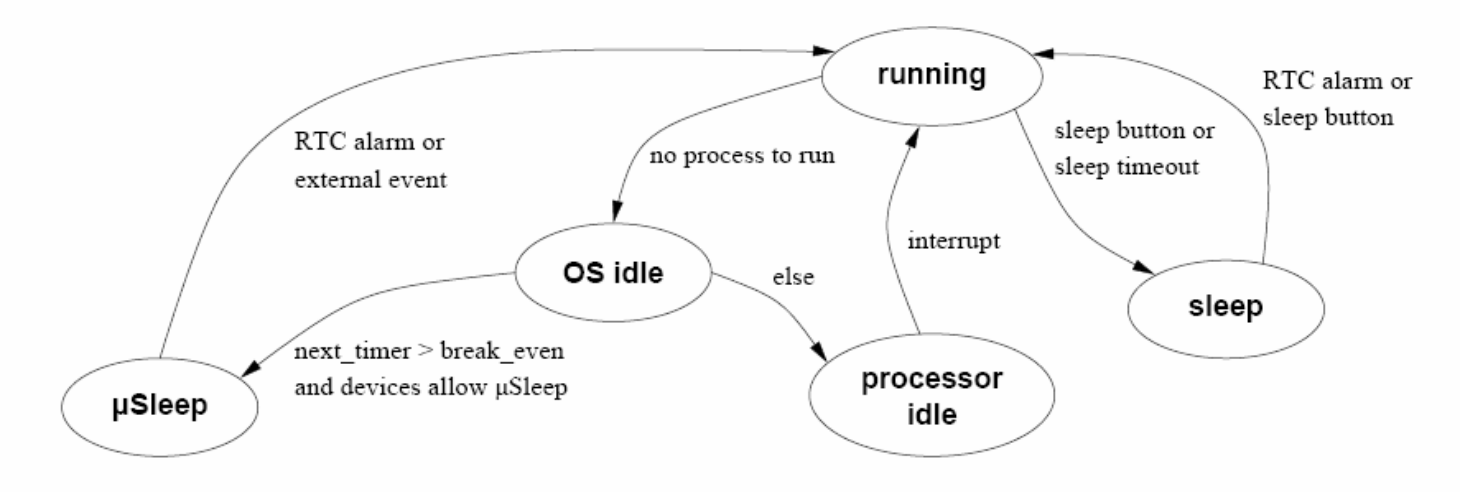
vartotojui derėtų žinoti, kad sistema yra tik laukimo būsenos (idle) rėžime. Ekranas įjungtas ir sistema atsako, ir vartotojo įvykiai yra natūralūs.

Vienintelis pokytis tai, kad prisideda vėlavimas, kai sistema pabunda, nuo išorinio įvykio pasirodymo, toks kaip mygtuko ar ekrano palietimas. Tačiau šis vėlavimas nėra pastebimas vartotojo, kadangi ir blogiausiu atveju, procesoriaus pažadinimo vėlinimas ir paleidimo tęsimas yra mažiau nei 12 ms.

Šis metodas išbandytas praktiškai. Naudojamas Itsy nešiojamas kompiuteris. Įrenginys susideda iš StrongARM SA-1100 procesoriaus, 32 Mb liekinės atminties (flash memory), 32 Mb dinaminės atminties. Naudojat qSleep technologijos prototipą, nešiojamuose kompiuteriuose, energijos sunaudojimas sumažėjo 60%.

Energijos sumažinimo tikslas yra pasiekiamas, draudžiant procesoriui perjungti į miegojimo rėžimą, nebent jis nustatė, kad miegos tiek laiko, kurio pakaks sutaupyti energiją.

Norint įvykdyti qSleep, reikalingi specifiniai aparatūrinės ir programinės dalies reikalavimai. Yra keturi aparatūrinės dalies reikalavimai. Pirma, procesorius turi turėti miegojimo rėžimą. Tai yra bendra moderniems procesoriams, kurie skirti žemos energijos taikymui. Antra, įrenginys turi turėti galimybę parodyti statinį paveikslėlį, kol procesorius miega. Tai gali būti pasiekta daugeliu atvejų; pvz. jeigu procesorius gali laikyti aktyvų integruotą LCD reguliatorių, kol procesorius eina miegoti. Turint LCD reguliatorių išorėje procesoriaus arba naudojant LCD ekraną su sudarytu freimo buferiu, kuris gali parodyti paveikslėlį frame buferyje, kai LCD reguliatorius išjungtas. Trečia, sistema turi turėti galimybę pabusti veikiant išoriniams įvykiams, tokiems kaip mygtuko paspaudimas arba valdomo ekrano palietimas. Galų gale įrenginys turi turėti programuojamą laikmatį, tobulai su geba nuo 1 iki 10 ms, kuris gali pažadinti sistemą. Šitas laikmatis yra naudojamas sistemos pažadinimui prieš kitą suplanuotą operacinės sistemos įvykį.



2 paveikslas. qsleep vykdymo busenų diagrama

Antras paveikslas parodo sistemos būsenas, kai vykdomas qSleep. Šiame vykdymo pavyzdyje, realaus laiko laikrodis (RTC – real time clock) yra naudojamas kaip qSleep pažadinimo laikrodis. Pagrindinė būsena yra tada, kada sistema veikia; veikia procesas arba gija, arba paleidžiamas OS branduolio kodas proceso ar gijos vardu. Sistema gali pereiti į savo miegojimo būseną, kaip rezultatas duoto mygtuko paspaudimas arba kaip rezultatas neveiklaus laikmačio. Į OS laukimo būseną (OS idle) pereinama tada, kai OS laukimo būsenos procesas pradeda veikti, rezultate gauname, kad nė vienas procesas ar gija negali pasileisti. Kai sistema pereina į OS laukimo būseną, ji patikrina ką ji gali padaryti trumpo laiko miegojimui. Patikrinimas susideda iš lemiamo veiksnio ar sistema gali miegoti pakankamai ilgai, tam, kad sutaupyti energiją ir jei visi įrenginiai leidžia procesoriui eiti į miegojimo rėžimą. Jeigu taip, sistema eina į qSleep būseną, kitu atveju sistema eina į procesoriaus laukimo būseną.

Išanalizuotas qSleep metodas turi privalumų ir trūkumų. Privalumai:

- Sutaupo energijos vartojimą iki 60 %
- Vartotojas nepastebi, kad procesorius miega
- Mažas vėlinimo laikas, iki 12ms

Pagrindinis metodo trūkumas – reikia specialios aparatūrinės įrangos. Delninukas turi atitikti jam keliamus reikalavimus, tam, kad galėtų įvykdyti metodą. Todėl senuose nešiojamuose kompiuteriuose qSleep nepavyks paleisti.

2.4.3. Objektiškai orientuoto programavimo strategijos, naudojant C# programavimo kalbą, mobiliuosiuose įrenginiuose

Kayun Chantarasathaporn ir Chanowat Srisa-an [1] atliko tyrimą tema: Objektiškai orientuoto programavimo strategijos, naudojant C# programavimo kalbą, nešiojamuose įrenginiuose. Tyrimas yra pradininkas, kuris bando specifikuoti ir pasiūlyti optimizuotą OOP programavimo strategiją, rašant programas, kurios skirtos įrenginiams su išorine baterija. Metodas paremtas tikrais rezultatais.

Tyrime autoriai bandė tirti energijos suvartojimą lyginant OOP naudojimą. Buvo naudojami OOP programavimo elementai: klasės, metodai, atributai ir t.t. Buvo suprogramuotos paprastos programėlės. Sakykim viena programa suprogramuota naudojat klases, o kita – struktūras. Programos paleidžiamos ir žiūrima per kiek laiko procesorius suskaičiuoja ir kiek atminties užima. Ir daroma išvada, kad programa parašyta naudojant struktūras, resursų naudoja mažiau. Taip atliktas visas eksperimentas su įvairiais programavimo elementais.

Lentelė 1. CPU laiko naudojimas, skirtingose C# OOP programavimo sąlygose

ISSUE	SUB-ISSUE	MAX PROCESSOR TIME (MILLISECONDS)				MEMORY (KILOBYTES)	
		User Time	Privilege Time	Total Time	Difference (%)	Total Memory	Difference (%)
Classes & Structs	Class (data members only)	3,212.62	20.03	3,232.65	*used as base*	3,705.60	3.21
	Struct	1,808.60	18.03	1,826.63	43.49	3,828.40	*used as base*
Prototypes	Abstract Class	986.42	24.03	1,010.45	*used as base*	3,768.80	1.17
	Interface	992.43	16.02	1,008.45	0.20	3,813.60	*used as base*
Fields	Dynamic Field	3,301.75	24.03	3,325.78	*used as base*	3,730.80	1.34
	Static Field	1,811.60	18.03	1,829.63	44.99	3,781.60	*used as base*
Methods	Dynamic Method	3,571.14	16.02	3,587.16	*used as base*	3,771.20	*used as base*
	Static Method	1,620.33	24.03	1,644.36	54.16	3,744.80	0.70
Local Field Accessibility	Private	1,802.59	16.02	1,818.62	43.85	3,770.00	0.22
	Protected	1,804.59	17.02	1,821.62	43.75	3,778.40	*used as base*
	Public	3,219.63	19.03	3,238.66	*used as base*	3,700.00	2.07
Local Method Accessibility	Private	906.30	17.02	923.33	1.39	3,761.20	1.13
	Protected	911.31	22.03	933.34	0.32	3,798.80	0.14
	Public	914.31	22.03	936.35	*used as base*	3,804.00	*used as base*
Field Accessibility from Other Class	Protected	1,805.60	18.03	1,823.62	43.53	3,810.00	*used as base*
	Public	3,214.62	15.02	3,229.64	*used as base*	3,750.40	1.56
Method Accessibility from Other Class	Protected	708.02	17.02	725.04	0.82	3,845.20	*used as base*
	Public	711.02	20.03	731.05	*used as base*	3,821.20	0.62
Anonymous & Named Object	Anonymous Object	4,376.29	105.15	4,481.44	*used as base*	4,001.60	*used as base*
	Named Object	3,510.05	15.02	3,525.07	21.34	3,717.20	7.11

Iš lentelės matosi tyrėjų gauti testų rezultatai. Gautos tyrimo rekomendacijos, gali būti kelrodis, kuris gali padėti programų kūrėjams geriau optimizuoti programas.

Naudojant šį tyrimo metodą galima šiek tiek optimizuoti programos kodą ir tai jau leistų programuotojams sumažinti energijos suvartojimą. Aišku, nežymiai, nes dar daugelis elementų nėra ištirta.

2.4.4. Grub-pa algoritmas

Dar vienas literatūroje aprašytas metodas: resursų rezervavimo metodas, skirtas energijos prognozės planavimui [4]. Autoriai Claudio Scordino ir Giuseppe Lipari pristato GRUB-PA algoritmą. Naujas planavimo algoritmas skirtas energijos sistemoms. Algoritmas gali efektingai tvarkyti sistemą, kuri susideda iš sunkių ir lengvų realaus laiko užduočių. Užduotys gali būti periodiškos, pavienės arba neperiodinės. Algoritmas gauna išretintą

dažnių juostos plotį, iššauktą periodiškų užduočių, kurios paleidžiamos mažiau nei tikimasi arba pavienės užduotys, kurios atvyksta mažesniu dažniu ir naudoja šią informaciją naudodamos mažesnį procesoriaus dažnį.

GRUB-PA (Greedy Reclamation of Unused Bandwidth-Power Aware) algoritmas. Pirma nusistatome, kad procesoriaus greitis gali būti įvairus, nuo maksimalaus greičio iki minimalaus. Algoritmas paremtas procesoriaus dažnio sumažinimu. Jeigu procesoriaus dažnių juosta nenaudojama ji sumažinama tiek, kad pakaktų įvykdyti užduotis. Procesorius dirba optimaliai, nevartodamas nereikalingos energijos. Tokiu būdu sumažinama sunaudojama energija.

Kadangi tai matematinis algoritmas, todėl jis aprašytas formulėmis, bei teoremomis, kurias galima rasti autorių straipsnyje. [2]

Eksperimento rezultatai rodo, kad naudojant algoritmą, buvo sutaupyta 38.4% bendros sistemos energijos eikvojimo.

2.4.5. Delninuko energijos suvartojimo įvertinimas taikomosios programos lygmenyje multimedia taikymams

KTU informatikos fakulteto, programų inžinerijos katedros habil. dr. Vytautas Štuikys ir doktorantas Jonas Valančius pasiūlė delninuko energijos suvartojimo įvertinimo metodą taikomosios programos lygmenyje multimedia taikymams (vaizdo ir garso failų, užkoduotų MPEG formatu [7] grotuvams), bei atliko tyrimą kaip energijos suvartojimas kinta nuo vartotojo pasirenkamų išorinių taikymo parametrų (garsumo lygio, vaizdo ryškumo ir kitų) [9]. Delninuko energijos suvartojimo įvertinimo modelis taikomosios programos lygmenyje, paremtas juodos dėžės metodu.

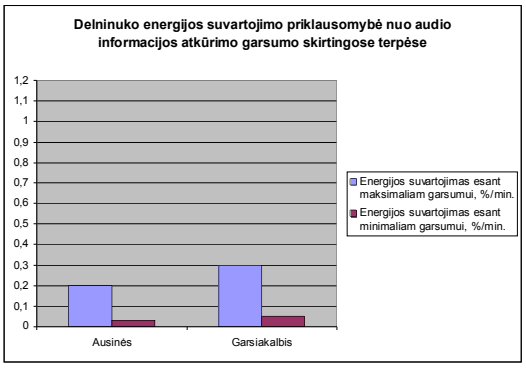
Buvo nustatyti delninuko komponentai, kurie vartoja energiją: garsiakalbis, procesorius, atmintis, ryšio sistema ir kiti komponentai. Pasirinkus tam tikrus komponentus ir keičiant jų parametrus, buvo stebimas baterijos energijos suvartojimas.

2 lentelė. Palm Zire 72 delninuko energiją naudojantys komponentai, jų veiksenos ir parametrai

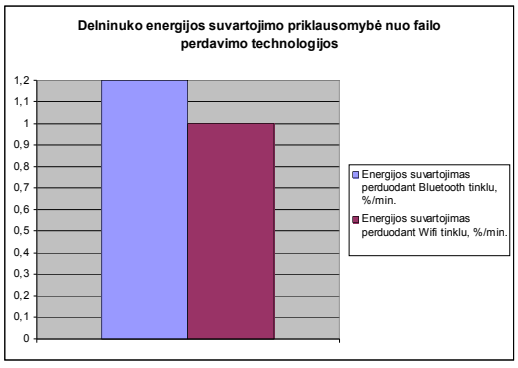
Eil. nr.	G	A	R	E	Galimas taikymas	Garso l.		Vaizdo ryškumas		Perdavimo būdas	
						Min	Max	Min	Max	BT	WIFI
1	0	0	0	1	+	-	-	+	+	-	-
2	0	0	1	0	taikymas negalimas						
3	0	0	1	1	+	-	-	+	+	+	+
4	0	1	0	0	+	+	+	-	-	-	-
5	0	1	0	1	+	+	+	+	+	-	-
6	0	1	1	0	taikymas negalimas						
7	0	1	1	1	+	-	-	+	+	+	+
8	1	0	0	0	+	+	+	-	-	-	-
9	1	0	0	1	+	+	+	+	+	-	-
10	1	0	1	0	taikymas negalimas						
11	1	0	1	1	+	-	-	+	+	+	+
12	1	1	0	0	taikymas negalimas						
13	1	1	0	1	taikymas negalimas						
14	1	1	1	0	taikymas negalimas						
15	1	1	1	1	taikymas negalimas						

Eksperimentas atliktas su delninuku Palm Zire 72 ir leidžia įvertinti šio delninuko energijos suvartojimą. Palm Zire 72 delninuke veikia Palm OS operacinė sistema, pats delninukas pritaikytas spręsti multimedia uždavinius.

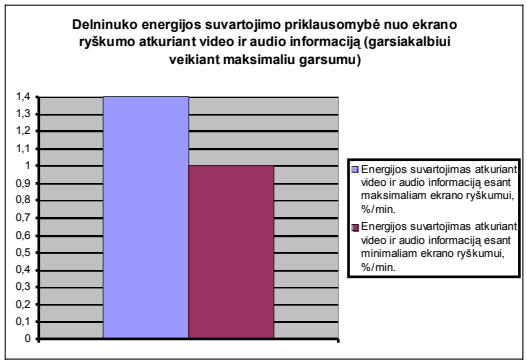
Norint įvertinti delninuko energijos suvartojimą, reikėjo išmatuoti energiją visiems galimiems taikymams. Kiekvienam delninuko taikymui buvo rasta programinė įranga, vartojanti reikiamus resursus ir išmatuotas jos energijos suvartojimas. Eksperimentas parodė, jog energijos suvartojimas delninuke skiriasi skirtingiems taikymams su skirtingais parametrais. Energijos suvartojimo matavimo rezultatai pavaizduoti 3 paveiksle.



a)



b)



c)

3 paveikslas. Vidutinis delninuko taikymų energijos suvartojimas per minutę:
a) kintant garsumui, b) kintant ryšio technologija, c) kintant ekrano ryškumui.

Išmatavę eksperimento rezultatus padarė išvadą, jog delninukas daugiausiai energijos suvartojas veikdamas didžiausiu pajėgumu. Mažiausiai energijos suvartoja taikymai, kai garsas atkuriamas ausinuku. Didžiausias energijos naudotojas – bevielio ryšio sistema Wifi.

3 lentelė. Delninuko Palm Zire 72 energijos suvartojimas

	Ausinės, %	**Garsiakalbis, %**	**Ryšys, %**	**Ekranas, %**	**Video, %**	**Video, %**
Garsumas maksimalus	0,2	0,3	x	x	+	x
Garsumas minimalus	0,03	0,05	x	x	x	+
Ryškumas maksimalus	x	x	x	0,9	1,4	1
Ryškumas minimalus	x	x	x	0,6	1	0,7
Perdavimas BT technologija	x	x	1	x	x	x
Perdavimas WIFI technologija	x	x	1,2	x	x	x

2.5. OOP naudojimas informacinėse sistemose

Informacinėms sistemoms kompiuterių taikymuose skiriamas vis didesnis ir didesnis dėmesys. Tai susiję su tuo, kad kompiuteriai perima žmogaus veiklos rutininių funkcijų. Tačiau šios funkcijos, nors ir rutininės, tampa sudėtingesnės, labiau panašėja į intelektualias procedūras. Didžiausią stimulą informacinių sistemų vystymui ir vystymuisi suteikia jų taikymas firmų ir organizacijų veikloje.

Prieš pora dešimtmečių informacinės sistemos sąvoka apėmė faktiškai tik bibliotekines sistemas ir vadinamąsias dokumentines sistemas, geriausiu atveju įjungiančias savyje nesudėtingas tekstines struktūras.

Per pastaruosius du dešimtmečius kelios kompiuterinių taikymų giminingos kryptys susiliejo į vieną, vadinama informacinių sistemų (IS) vardu. Dabartiniu metu IS paprastai apima:

- žmogaus veikloje generuojamą informaciją;
- jos formalizavimą ir struktūrizavimą;
- informacijos saugojimą ir apdorojimą duomenų bazėse;
- prasminės informacijos atgavimą iš atskirų duomenų bazėse saugomų duomenų ir faktų.

Šiandien kuriant įvairias IS, naudojamos objektiškai orentuotos programavimo kalbos. Objektinis požiūris pirmiausia sutelkia dėmesį probleminės srities objektų identifikavimui, po to realizuoja reikalingas funkcijas per atitinkamų objektų sąveikas. Objektiniai programiniai produktai geriau elgiasi keičiantis reikalavimams, kadangi tokia IS pagrįsta pačios probleminės srities struktūra (objektų rinkiniu, kuris beveik pastovus), o ne veiklos funkcijomis, kurios kinta nuolat.

Microsoft Visual Studio yra pagrindinis Microsoft programinės įrangos kūrimo produktas, skirtas programuotojams. Šiame produkte yra integruota kūrimo aplinka, kuri leidžia programuotojams kurti savarankiškas programas, žiniatinklius, žiniatinklio taikomasias programas, servisus ir veikia ant bet kokių platformų, palaikomų Microsoft .Net Framework. Į palaikomas platformas įeina „Microsoft“ Windows serveriai ir darbo stotys, PocketPC, Smartphone ir žiniatinklio naršyklės.

Kuriant sistemas Windows aplinkai didžiausia problema iki .NET pasirodymo buvo suderinamumas. Visa apimantys nesuderinamumai yra pagrindinis šiuolaikinės programavimo industrijos trūkumas. Skirtingas PĮ kūrimo priemones naudojantys programuotojai praktiškai buvo vienas kito izoliuoti. Įvairioms Win OS parašytos programos naudoja savo specialias API, todėl jas ne tik sudėtinga perkelti į kitas platformas, bet jos netgi neveikia skirtingose Win versijose. Kuriant .NET platformą „Microsoft“ tikslas buvo pašalinti suderinamumo problemas. Bet .NET platformos privalumai iš pirmo žvilgsnio nėra tokie akivaizdūs, kadangi visi skirtumai yra architektūriniame lygyje ir galutiniam vartotojui nėra matomi. .NET platforma atsako už:

- programų kūrimą ir paleidimą,

- programų eigos valdymą
- atminties išskyrimą duomenims ir komandoms, jos atlaisvinimą

Magistriniame darbe bus naudojama objektiškai orentuota programavimo kalba C# sukurta Microsoft kaip dalis .Net iniciatyvos ir vėliau patvirtinta ISO standartu. Programavimo kalba turi procedūrinę objektiškai orentuotą sintaksę, paremtą C++ . Todėl naudojant C#, galime sukurti objektiškai orentuotą programą ir stebėti delninuko parametrus (baterijos energijos sunaudojimą, procesoriaus ir atminties naudojimą). Kadangi šiuolaikiniuose IS naudojamos OOP kalbos, mūsų darbe turime panaudoti OOP kalbą ir sukūrus OO programą, stebėti, kaip jos vygdymas paveikia baterijos energijos suvartojimą. Programoje bus realizuota išretintų matricų daugyba. Pasinaudojus išretintų matricų savybe, kuri leidžia sumažinti matricos formatą, išmetant nulinius elementus, galime stebėti, kaip keičiasi sistemos resursų sunaudojimas, keičiant pradinėje matricoje nulinių elementų skaičių.

2.6. Darbe naudojamų matematinių metodų analizė

Darbe naudosime išretintų matricų metodą. Matrica - stačiakampė mxn skaičių lentelė, susidedanti iš m eilučių ir n stulpelių. Žymima:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \text{ arba } A = (a_{ij}), i = \overline{1,m}, j = \overline{1,n}.$$

Elementai a_{ij} vadinami matricos elementais; pirmasis indeksas i žymi eilutės, kurioje yra elementas, numerį, o antrasis indeksas j - stulpelio numerį. Kai matricoje yra m eilučių ir n stulpelių, ji vadinama $[m \times n]$ formato matrica. Išretintomis matricomis vadinamos matricos, kurios turi santykinai mažą skaičių nenulinių elementų. Išretinta matrica galima saugoti pilnų matricų metodų pagalba. Kai išretinta matrica yra saugoma, visi jos elementai, įskaitant ir nulinius, yra talpinami į masyvus. Darbe bus naudojamas eilučių saugojimo metodas.

Eilučių saugojimo metodas

Matricos A saugojimas naudojant eilučių metodą, remiasi 3 vienos dimencijos masyvais, kurie apibūdina matricą A. Tie masyvai yra AR, IA ir JA. Duota matrica [m x n], kuri turi *ne* ne nulinių elementų. Masyvai sudaromi tokia tvarka:

AR – dydis bent *ne*, susijęs su ne nuliniais matricos A elementais, kurie laikomi gretimai. Matricos A eilutės yra laikomos iš eilės nuo 1 iki m masyve AR. Matricos A kiekvienos eilutės elementas yra laikomas bet kokia tvarka masyve AR.

IA – integer masyvas. Dydis m+1, susijęs su kiekvienos matricos A eilutės pražios pozicija masyve AR; tai yra, kiekvienas elementas IA(i) parodo, kur eilutė i prasideda masyve AR. Jeigu eilutės visi elementai nuliai, tada IA(i)=IA(i+1). Paskutinis elementas IA(m+1) parodo poziciją po paskutinio masyvo AR elemento, kuris yra ne+1

JA – integer masyvas. Dydis bent (ne), susijęs su stulpelio numeriu kiekvieno nenulinio matricos A elemento a_{ij} .

Tarkime turime duotą matrica A:

$$A = \begin{pmatrix} 11 & 0 & 13 & 0 & 0 & 0 \\ 21 & 22 & 0 & 24 & 0 & 0 \\ 0 & 32 & 33 & 0 & 0 & 0 \\ 0 & 0 & 43 & 44 & 0 & 46 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 61 & 62 & 0 & 0 & 0 & 66 \end{pmatrix}$$

Tuomet masyvai AR, IA, JA atrodys taip:

$AR = (11,13,24,22,21,32,33,44,43,46,61,62,66)$
$IA = (1,3,6,8,11,11,14)$
$JA = (1,3,4,2,1,2,3,4,3,6,1,2,6)$

Bendrai šitą saugojimo techniką galima išreikšti:

kiekvienam $a_{ij} <> 0, kai \;\; i = 1,m$ ir $j = 1,n$

egzistuoja k, kur $1 <= k <= ne,$ toks kad $AR(k) = a_{ij}$, $JA(k) = j$ ir kai $i = 1,m,$

$IA(i) = k$, kur a_{ij} yra masyve AR(k) pirmas elementas patalpintas kai eilutės numeris i.

$IA(i) = IA(i+1)$, kur visi $a_{ij} = 0$ kiekvienoje eilutėje i.

$IA(m+1) = ne+1$

2.7. Darbe naudojama programinė įranga

Aprašyti matematiniai modeliai bus realizuojami su MS Visual Studio 2003. Naudojama programavimo kalba C#. Parašytos programos bus vykdomos delniniuose kompiuteriuose, kuriuose yra Windows Mobile 3.0 operacinė sistema, versija 4.20.1081. Delninuko procesorius: Intel(R) PXA255. Delninko atmintis: 126.63 MB. Vykdymo rezultatai bus atvaizduojami per vartotojo sąsają. Pradiniai duomenys ir galutiniai rezultatai nuolat kaupiami ir saugomi faile.

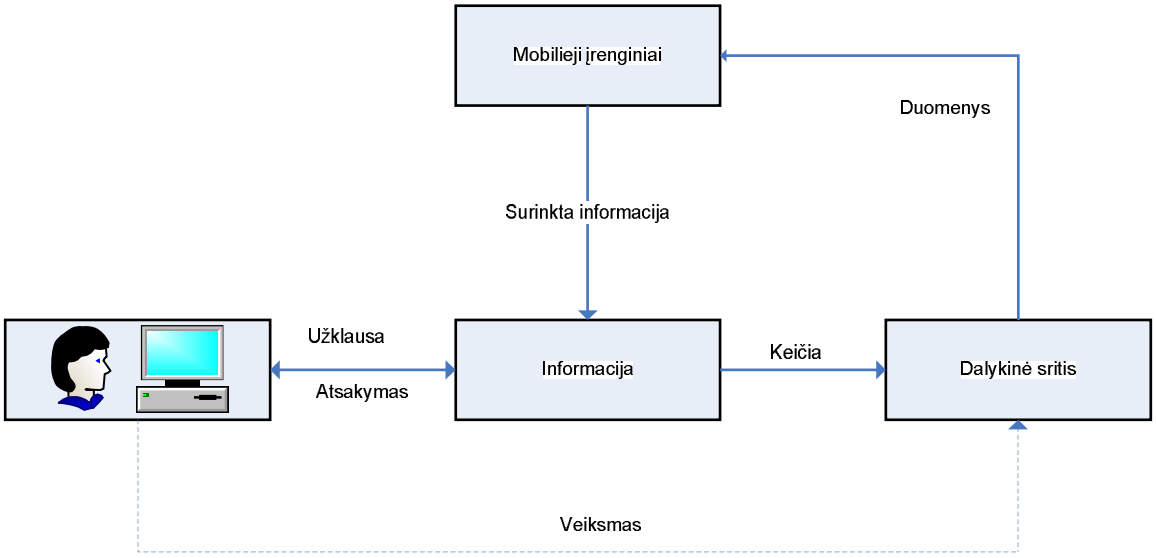
2.8. Analizės išvados

- Svarbi problema yra baterijos ilgaamžiškumas ir energijos taupymas, kuo daugiau energijos bus sutaupyta tuo ilgiau sistema nepetraukiamai veiks.
- Kiekvienas mokslinis tyrimas yra specifinis. Dėmesys skiriamas vienam komponentui, taip stengiamasi sumažinti kompomento įtaką bendram energijos suvartojimui.
- Moksliniuose straipsniuose tyrimai atlikti įvairiausiais metodais ir priemonėmis. Vieni metodai yra teoriniai, o kiti praktiškai įgyvendinti.
- Savo darbe naudosime išretintų matricų metodą. Metodas tinka sistemos resursų stebėjimui, kadangi keičiant įvairius parametrus, keičiasi sistemos resursų panaudojimas. Iš gautų rezultatų galėsime įvertinti, kaip procesorius ir atmintis įtakoja bendrą baterijos energijos suvartojimą. Tyrimą atliksime delniniuose kompiuteriuose, kurie turės Windows Mobile 3.0 operacinę aplinką.
- Naudojant objektiškai orentuotą programavimo kalbą C# galima sukurti programą, kuri ne tik atitinka realybėje naudojamas programas, bet ir išgauna delninuko parametrus, reikalingus atlikti tyrimui.

3. Delninuko energijos suvartojimo modeliavimo taikomosios programos lygmenyje projektas

3.1. Modeliavimo pagrindimas

Pagal ISO WG3 (1982) informacinė sistema apibrėžiama kaip suprojektuota sistema, kuri surenka, saugoja, apdoroja ir paskirsto informaciją apie dalykinės srities būseną. IS atlieka 3 pagrindines funkcijas: atminties funkciją, informavimo funkciją, aktyviąją funkciją. Dažnai informacinėse sistemose naudojami mobilieji įrenginiai, kurie palengvina informacijos surinkimą. Iš 4 paveikslo matome mobiliųjų įrenginių vietą informacinėje sistemoje ir kaip informacinėje sistemoje atliekamos funkcijos. Mobilieji įrenginiai iš dalykinės srities surenka duomenis, apdoroja juos, taip gaudami tam tikra iformaciją ir patalpina ją prie kitos surinktos informacijos.



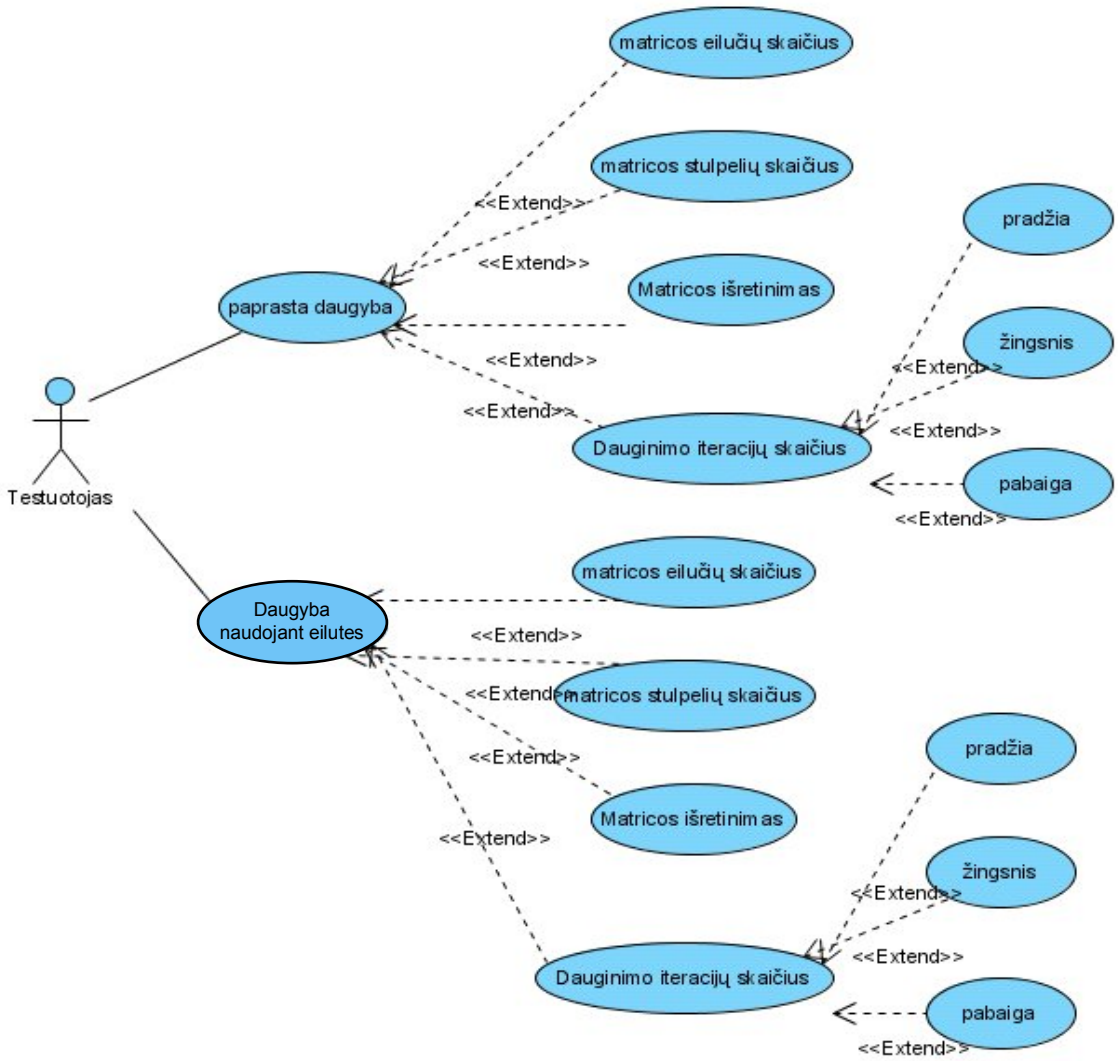
4 paveikslas. Informacinė sistema

Savo darbe aš naudosiu mobilųjį įrenginį – delninuką. Jo funkcionalumo dėka, jis gali būti plačiai naudojamas įvairuose informacinėse sistemose. Magistriniame darbe bus modeliuojamas delninuko energijos suvartojimas taikomosios programos lygmenyje. Delninuko energijos suvartojimas yra labai svarbus faktorius, nes informacinė sistema veiks tol, kol delninukas veiks.

3.2. Programos aprašymas

Norint atlikti modeliavimą, sukuriama programa, kurią vygdant, senka delninuko baterija ir stebimi pasirinkti parametrai (voltai, baterijos nusedimas procentais ir t.t.).

Programoje bus realizuota dviejų matricų daugyba. Dauginant matricas, bus apkraunama delninuko baterija. Dauginsime dvi vienodas matricas, todėl užtenka aprašyti vieną matricą. Matricas užpildome skaičiais nurodę matricos formatą ir naudojantis funkcija *randomize* matrica užsipildys atsitiktiniais skaičiais. Prieš užpildant matricą skaičiais nurodome matricos išretinimą. Skaičių intervalas yra nuo nulio iki nurodyto programos viduje maksimalaus skaičiaus (naudotojas iš išorės negali pakeisti intervalo). Išretinimas nurodomas procentais ir parodo kiek procentų matricoje bus nulinių elementų. Generuojami tik sveiki skaičiai. Delninukui turint ribota atminties kiekį, reikia įvesti dar vieną kintamąjį – iteracijų skaičių. Kadangi didelio formato matricų sudauginti neįmanoma, mes nurodome iteracijų skaičių ir mažesnio formato matricos atliks daugybą tiek kartų, kiek nurodėme iteracijų. Iteracijų skaičių nurodome prieš pradedami daugybą. Matricų daugyba susidės iš dviejų dalių. Pirmoje dauginsime paprastas matricas. Antroje dauginsime išretintas matricas, pasinaudojus saugojimu pagal eilutes metodu.



5 paveikslas. Matricų daugybos panaudojimo atvejų diagrama

Paprastų matricų daugyboje, naudojamos dvi vienodos matricos. Todėl užtenka sugeneruoti vieną matricą ir kitą mes jau turime. Sugeneravus matricas, vygdydami programą, pasirenkame daugybos tipą. Dauginimo rezultatai ir gauti pasirinkti parametrai surašomi į rezultatų failą.

Kaip ir paprastų matricų daugyboje, pradžioje susigeneruojame matricas. Nurodome matricų išretinimą procentais. Išretinę matricas išsaugome jas eilučių saugojimu metodu. Toliau dauginame matricas. Kelių daugybų naudojimas parodys energijos suvartojimo

skirtumą. Išretintos matricos išsaugojimas eilučių metodu panaikina nulinius elementus, taip sutaupoma kreipinių į atmintį, bei skaičiavimo kaštus.

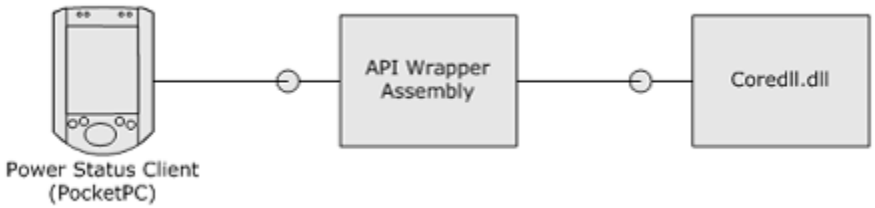
3.2.1. Stebimų parametrų aprašymas

Atliekant daugybą stebime keletą parametrų, kurie padės įvertinti energijos suvartojimą. Vienas iš stebimų parametrų yra baterijos voltažas. Jo kitimas parodys kaip keičiasi baterijos įtampa senkant baterijai. Kitas parametras yra baterijos įkrautumas procentais. Stebėsime kaip kintant procentams keičiasi ir voltažas, ir kokią įtaką tai turi dauginimui. Sekantis parametras yra laikas. Laikas yra pradedamas matuoti pradėjus dauginti ir sustabdomas kada daugyba baigta. Kiti parametrai yra iterecijų pradžia. Iteracijų žingsnis, kuris nurodo, kas kiek iteracijų tarpiniai parametrai rašomi į rezultatų failą. Bei iteracijų skaičius, kuris nurodo kiek reikia kartų dauginti matricas. Rezultatų tikslumui, bus išjungiamas ekranas, nes ekranas suvartoja nemažai energijos ir rezultatai gali būti netikslūs.

3.2.2. Naudojamų klasių aprašymas

Delninuko kai kuriems parametrams gauti, kuriame dalykinį projektą, dauginimui ir rezultatų surašymui skaičiavimo projektą.

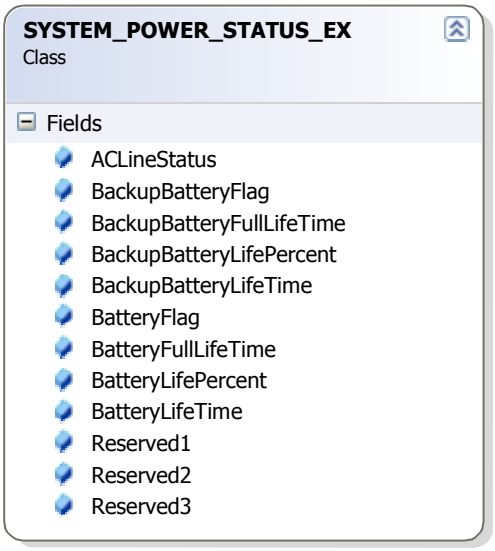
Dalykiniame projekte įdiegiame Windows CE coredll modulį. Coredll modulis yra bazinis operacinės sistemos modulis, kuris leidžia panaudoti branduolio funkcijas kitiems moduliams, taip pat jis leidžia operacinės sistemos konfiguravimą. Norint pasinaudoti šiomis funkcijomis, mes sukuriame nuorodas į coredll. Sukompiliavus projektą, mes gauname power.dll failą, kuriame nurodyta kaip ir iš kur gauti tam tikrus delninuko parametrus, ir kaip biblioteką mes prisegsime prie skaičiavimo projekto. Angliškai toks bibliotekos sukūrimas vadinamas „wrap“- apgaubimu (http://www.likit.lt/en-lt/zod_w.html). Kadangi tiesiai mes i branduolį kreiptis negalime, darome apgaubimą.



6 paveikslas. Coredll panaudojimo pavyzdys

Sukuriame dalykinį projektą „Power“. Visi stebimi prametrai yra surašyti standartinėse klasėse. Naudojamos dvi pagrindinės klasės: `SYSTEM_POWER_STATUS_EX` ir `SYSTEM_POWER_STATUS_EX2`.

Klasėje `SYSTEM_POWER_STATUS_EX` apsirašome kintamuosius.




```
public class SYSTEM_POWER_STATUS_EX
{
      public ACLineStatus ACLineStatus;
      public byte BatteryFlag;
      public byte BatteryLifePercent;
      public byte Reserved1;
      public uint BatteryLifeTime;
      public uint BatteryFullLifeTime;
      public byte Reserved2;
      public byte BackupBatteryFlag;
      public byte BackupBatteryLifePercent;
      public byte Reserved3;
      public uint BackupBatteryLifeTime;
      public uint BackupBatteryFullLifeTime;
}
```

Nevisi kintamieji bus įrašomi į rezultatų failą. Kiti bus panaudoti kaip pagalbinė priemonė programos veikimo tikrinimui.

Kintamasis ACLineStatus rodo ar prijungtas išorinis šaltinis.

Reikšmė	Apibūdinimas
0	Išjungtas pakrovėjas

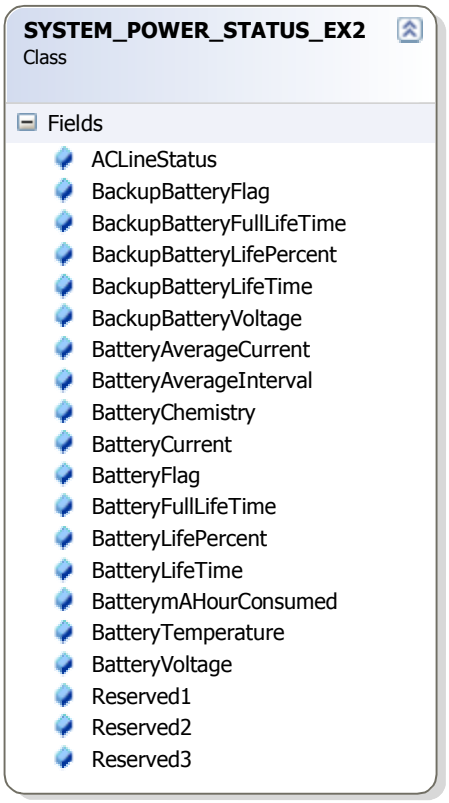
1	Įjungtas pakrovėjas
255	Nežinomi duomenys

BatteryFlag parodo baterijos įsikrovimo lygį.

Reikšmė	Apibūdinimas
1	Aukštas (High)
2	Žemas (Low)
4	Kritinis (Critical)
8	Kraunama baterija
128	Nėra baterijos
255	Nežinoma būsena

Kintamasis BatteryLifePercent – parodo, kiek procentų liko pilnai pakrautos baterijos. Reikšmės kinta nuo 0 iki 100.

Kitoje klasėje System_Power_status_ex2 n audojami tie patys kintamieji kaip ir prieš tai aprašytoje klasėje, bei papildytą naujai.

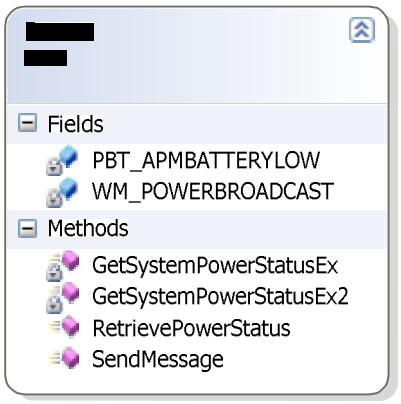



```
public class SYSTEM_POWER_STATUS_EX2
      {
            public ACLineStatus ACLineStatus;
            public byte BatteryFlag;
            public byte BatteryLifePercent;
            public byte Reserved1;
            public int BatteryLifeTime;
            public uint BatteryFullLifeTime;
            public byte Reserved2;
            public byte BackupBatteryFlag;
            public byte BackupBatteryLifePercent;
            public byte Reserved3;
            public uint BackupBatteryLifeTime;
            public uint BackupBatteryFullLifeTime;
            public uint BatteryVoltage;
            public uint BatteryCurrent;
            public uint BatteryAverageCurrent;
            public uint BatteryAverageInterval;
            public Int32 BatterymAHourConsumed;
            public uint BatteryTemperature;
            public uint BackupBatteryVoltage;
            public byte BatteryChemistry;
      }
```

Šioje klasėje bus naudojamas kintamasis `BatteryVoltage`. Jis parodys delninuko baterijos voltažą, kurio reikšmės gali kisti nuo 0 iki 65535. Voltažas matuojamas mili voltais.

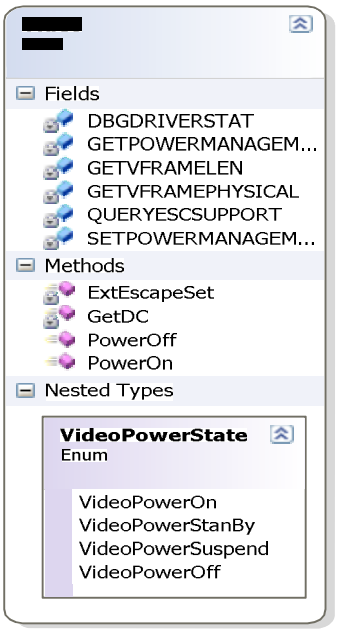
`BatteryChemistry` kintamasis, kuris rodo naudojamos baterijos tipą. Galimi tokie variantai: Alkaline, Nicd, Nimh, Lion, Lipoly.

Abi klases panaudojame pagrindinėje klasėje Power, kurioje bus parašyti metodai, kurių pagalba mes gausime duomenis apie delniniko bateriją.



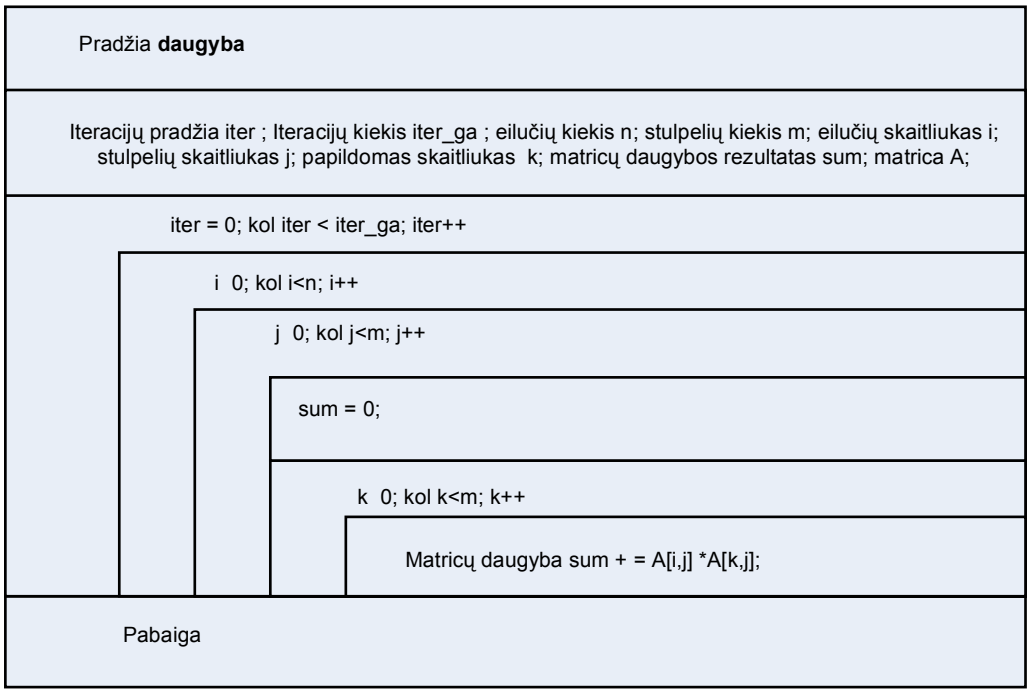
Metodas RetrievePowerStatus gauna įvairius norimus stebėti baterijos parametrus. Metodais GetSystemPowerStatusEx ir Ex2 kreipiamės į išorinę biblioteką, kurioje ir yra anksčiau aprašytos dvi klasės.

Ekrano išjungimas dauginimo metu yra aprašytas klasėje Video.



Metodas PowerOff išjungia monitorių, o PowerOn įjungia. VideoPowerState – parodo monitoriaus būseną. Galimos tokios reikšmės: monitorius įjungtas, išjungtas, ekranas yra miegojimo rėžime.

Skaičiavimo projekte „matricu_daugyba“ yra sukuriama standartine klasė `public class Form1 : System.Windows.Forms.Form.` Šioje klasėje naudojami pagrindiniai metodai, kurie daugina matricas, gauna baterijos parametrus ir surašo juos į rezultatų failą. Metodas *daugyba()* atlieka paprastų matricų daugybą su tam tikru išretinimu. Patį matricų daugybos alrgoritmą pavaizduojame struktūrograma.

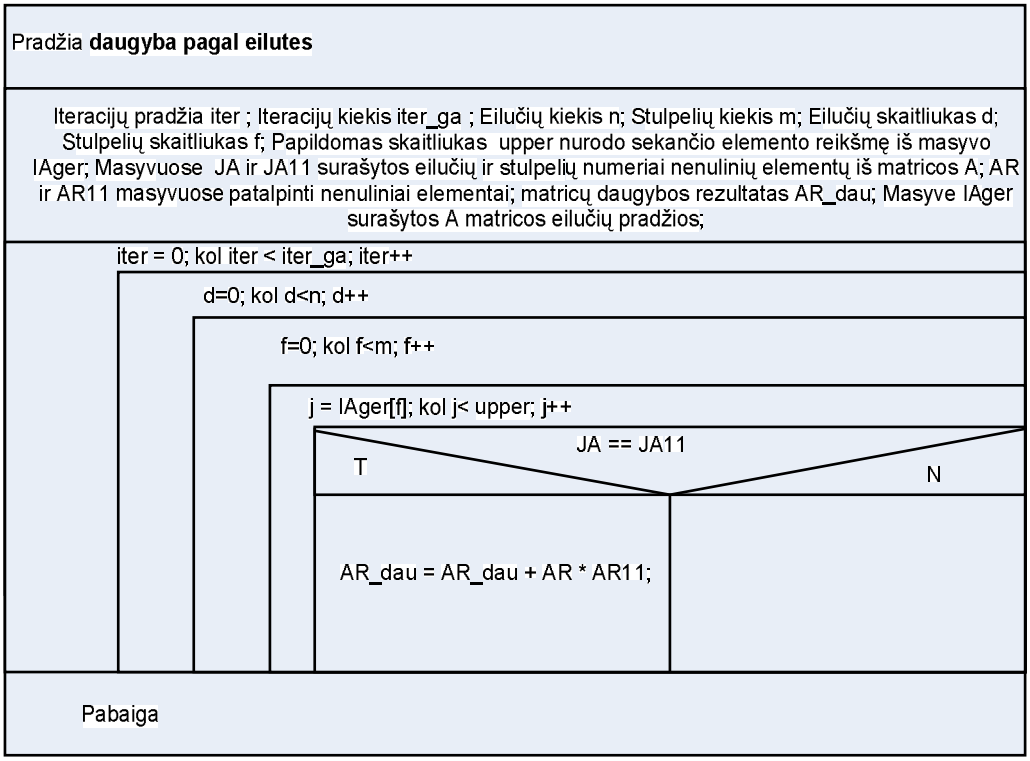


7 paveikslas. Matricų daugybos algoritmo struktūrograma

Struktūrogramoje pavaizduotas programos teksto fragmentas, kuriame sudauginamos dvi matricos. Pateikiame programos fragmentą, parašytą C# kalboje, kuriame atliekama matricų daugyba:

```
for( int iter = 0; iter < iter_ga; iter++)
{
    for (int i = 0; i < n; i++)
    {
       for (int j = 0; j < m; j++)
       {
          sum = 0;
          for (int k = 0; k < m; k++)
          {
               sum += a[i, k] * a[k, j];
          }
       }
    }
}
```

Metodas *daugyba_išretinta()* atlieka suspaustų matricų daugybą, su tam tikru išretinimu. Matricų daugybos saugojant eilutes algoritmo fragmentą, pavaizduosime supaprastinta struktūrograma.



8 paveikslas. Matricų daugybos algoritmo struktūrograma

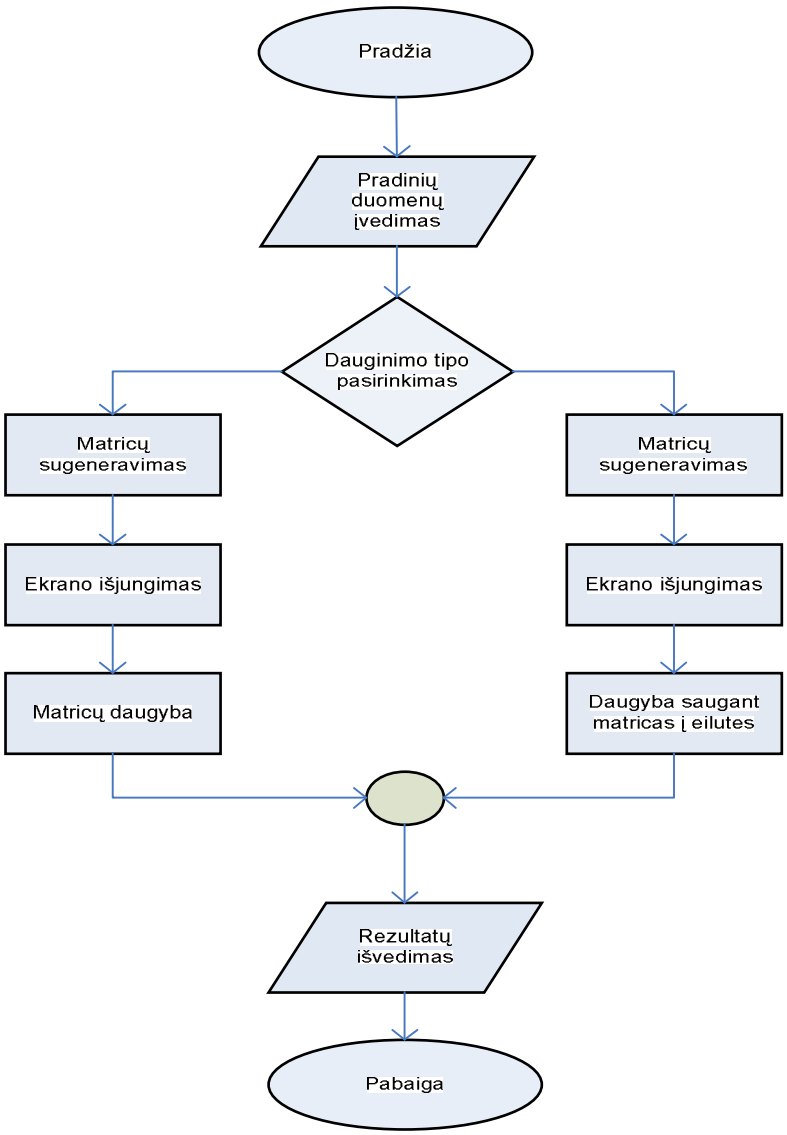
Programos fragmentas, parašytas C# kalboje yra toks:

```
for( int iter = 0; iter < iter_ga; iter++)
{
   for (int d = 0; d < n; d++)
   {
     eil = eil_kiek[d];
     for (int f = 0; f < m; f++)
     {
        upper = IAger[f+1];
        if (d == 0)
        {
           indx = 1;
        }
        else
        {
           indx = eil_kiek[d-1];
           indx++;
        }
        for ( int j = IAger[f]; j< upper; j++)
        {
          if (indx <= eil)
            if (JA[j-1] == JA11[indx-1])
            {
               AR_dau[f] = AR_dau[f] + AR[j-1] * AR11[indx-1];
               indx++;
             }
             else
             {
                 for (int t = indx; t <= eil; t++)
                 {
                   if (JA[j-1] == JA11[t-1])
                   {
                      AR_dau[f] = AR_dau[f] + AR[j-1] * AR11[t-1];

                   }
                   else
                   {
                        AR_dau[f]= AR_dau[f] + 0 ;
                   }
                 }

              }

         }

      }
   }
}
```

Kiekvienas metodas skaičiavimo metu ir jiems pasibaigus rezultatus surašo į rezultatų failą. Rezultatų failo pavyzdys:

```
AC power status: Offline
Battery Charged: 63%
Battery VOLTAGE: 3,952  V
pradinis iteraciju skaicius: 1
iteraciju zingsnis: 10
galutinis iteraciju skaicius: 1000
Ivykdytu iteraciju skaicius: 1
ivykdytu iteraciju dauginimo laikas00:00:00

AC power status: Offline
Battery Charged: 63%
Battery VOLTAGE: 3,952  V
Ivykdytu iteraciju skaicius: 11
ivykdytu iteraciju dauginimo laikas00:00:00
.....................................
AC power status: offline
Battery Charged: 58%
Battery VOLTAGE: 3,925 V
Sudaugintos tokios matricos:
Matricos eiluciu skaicius: 100
Matricos stulpeliu skaicius: 100
Matricu isretinimas procentais: 91
Dauginimo iteraciju skaicius: 1000
---LAIKAS---------
00:32:53
```

Apibendrintą programos veikimą, galima būtų pavaizduoti algoritmo blokine schema:



9 paveikslas. Apibendrinta daugybos algoritmo blokinė schema

4. Eksperimentinis delninuko energijos suvartojimo tyrimas

4.1.1. Pradinių duomenų nustatymas

Sukūrus programą, reikia nusistatyti su kokais pradiniais duomenimis atliksime tyrimą. Kadangi eksperimentas yra atliekamas pirmą kartą, tik pabandę su įvairiais duomenimis eksperimento būdu mes išrenkame kokios matricos dimensijas mes naudosime, bei kiek iteracijų galime atlikti. Tam tikslui susikuriame lentelę, kur bandymo metu suvesime gautus duomenis.

4 lentelė. Pavyzdinė lentelė, kuri bus užpildyta bandymo metu gautais duomenimis

Matricų formatas 40x40	Iteracijų pradžia:			
	Iteracijų žingsnis:			
	Iteracijų pabaiga:			
Baterijos įkrautumas žodžiais		max	average	min
Baterijos įkrautumo intervalas %		100-67	66-33	33-0
Matricos išretinimas				
Baterijos voltažas mV				
Baterijos įkrautumas %				
Įvygdytos iteracijos				

Dėl delninuko vidinės atminties ribotumo, mes negalime sudauginti didelio formato matricų. Bandymo metu buvo nustatyta, kad didžiausias formatas, kūri sudaugina yra 500 eilučių ir 500 stulpelių. Taip pat paprastų matricų daugyboje mes nepraleidžiame daugindami nulius, todėl matricų išretinimas neturi didelės reikšmės ir bandymą atliksime tik su paprastų matricų dauginimo būdu. Pasirenkame ir kelis matricų formato variantus. Kadangi daugiausiai galime sudauginti 500x500 formato matricas, pasirenkame 250x250 ir 100x100. Tokiu būdu gauti rezultatai bus tikslesni, bei parodys kaip priklauso baterijos energijos suvartojimas nuo atminties apkrovimo. Taip pat abiems dauginimo būdams eksperimento metu naudosime tokius pat pradinius duomenis gautus bandymo metu, tik tokiu būdu galėsime palyginti abiejų dauginimo tipų gautus rezultatus. Taigi pradžioje sudauginkime matricas, kurių formatas 500x500, pradinis iteracijų skaičius 1, žingsnis 20, itercijų skaičius 200. Atlikę bandymą gauname rezultatus:

5 lentelė. Matricos formato 500x500 dauginimo paprastų matricų metodu duomenys

Matricų formatas 500x500	Iteracijų pradžia:1							
	Iteracijų žingsnis:20							
	Iteracijų pabaiga:200							
Baterijos įkrautumas žodžiais		max			average		min	
Baterijos įkrautumo intervalas %		100-67			66-33		33-0	
Matricos išretinimas		97			97		97	
Baterijos voltažas mV		4,088	4,039	3,984	3,893	3,803	3,654	3,541
Baterijos įkrautumas %		100	85	70	54	37	20	3
Įvygdytos iteracijos		1	20	40	60	80	100	120

Iš dauginimo rezultatų matome, kad visų iteracijų nespėta atlikti, nes baterija išsikrovė visiškai. Todėl iš gautų rezultatų matome, kad iteracijų skaičių reikia sumažinti. Kad

spėtų atlikti visas iteracijas, iteracijų pabaiga pasirenkame 100, o procentais išreikštas baterijos įkrautumas neturi būti mažesnis nei 90 %. Kad gautume kuo tikslesnius grafikus, sumažiname ir iteracijų žingsnį iki 10.

Atliekame bandymą su 250x250 formato matricomis. Pasirenkame iteracijų pradžią 1, žingsnis 10, itercijų skaičius 500. Atlikę bandymą gauname tokius rezultatus:

6 lentelė. Matricos formato 250x250 dauginimo paprastų matricų metodu duomenys

Matricų formatas 250x250	Iteracijų pradžia:1 Iteracijų žingsnis:10 Iteracijų pabaiga:500							
Baterijos įkrautumas žodžiais		max				average		min
Baterijos įkrautumo intervalas %		100-67				66-33		33-0
Matricos išretinimas		97				97		97
Baterijos voltažas mV		4,088	4,051	4,028	3,992	3,952	3,925	-
Baterijos įkrautumas %		98	88	80	72	66	59	-
Įvygdytos iteracijos		1	100	200	300	400	500	-

Iš gautų rezultatų matome, kad iteracijų skaičių galime padidinti iki 1000. Tada turėsime rezultatus iškrovę beveik visą bateriją, palikę tik rezervinę dalį. 6 lentelėje duomenys pavaizduoti kas 100 iteracijų, nes dėl duomenų gausos neįmanoma juos čia pavaizduoti kas iteracijų. Tačiau brėžiant grafikus toks tikslumas pravers. Kuo daugiau duomenų, tuo tikslesnį ir detalesnį grafiką mes gausim.

Liko atlikti paskutinį bandymą su 100x100 formato matricomis. Iteracijų pradžią pasirenkame 1, žingsnį 100, iteracijų kiekį 5000. Atlikę bandymą gauname:

7 lentelė. Matricos formato 100x100 dauginimo paprastų matricų metodu duomenys

Matricų formatas 100x100	Iteracijų pradžia:1 Iteracijų žingsnis:100 Iteracijų pabaiga:5000								
Baterijos įkrautumas žodžiais		max						average	min
Baterijos įkrautumo intervalas %		100-67						66-33	33-0
Matricos išretinimas		97						97	97
Baterijos voltažas mV		4,056	4,053	4,051	4,028	4,015	3,992	-	-
Baterijos įkrautumas %		95	91	86	81	76	71	-	-
Įvygdytos iteracijos		1	1000	2000	3000	4000	5000	-	-

Iš gautų duomenų matome, kad iteracijų kiekis nebuvo pakankamas, kad užpildytų visus baterijos įkrautumo intervalus. Dėl duomenų gausos lentelėje duomenys rodomi kas 1000 iteracijų. Kad užpildytų baterijos įkrautumo intervalus, reikia iteracijų kiekį padidinti iki 15000. Taigi bandymo būdu nusistatėme visų trijų formatų matricų pradinius duomenis ir galime atlikti ekperimentą.

Ekperimento metu taip pat stebėsime ir laiką, kuris rodys kiek programa užtruko daugindama matricas. Taip pat ekperimento metu bus svarbus ir matricos išretinimas. Bandymų metu nustatėme, kad pastebimas skirtumas išretinimo tarp paprastų matricų daugybos ir išsaugotų matricų daugybos yra kai išretinimas yra apie 94%. Taip pat norėdami pamatyti kaip keičiasi dauginimo laikas nuo išretinimo pasirenkame kelis išretinimus, artimus 94%. Sudarome lentelę, kurioje bus surašyta kokius pradinius duomenis naudosime eksperimente ir kokių formatų matricas dauginsime.

8 lentelė. Eksperimente naudojami pradiniai duomenys

Dauginimo tipas	Matricų formatas	Išretinimas	Iteracijos		
			Iteracijų pradžia	Iteracijų žingsnis	Iteracijų kiekis
Paprastų matricų daugyba	100x100	97	1	100	15000
	100x100	95	1	100	15000
	100x100	91	1	100	15000
	250x250	97	1	10	1000
	250x250	95	1	10	1000
	250x250	91	1	10	1000
	500x500	97	1	10	100
	500x500	95	1	10	100
	500x500	91	1	10	100
Suspaustų matricų daugyba	100x100	97	1	100	15000
	100x100	95	1	100	15000
	100x100	91	1	100	15000
	250x250	97	1	10	1000
	250x250	95	1	10	1000
	250x250	91	1	10	1000
	500x500	97	1	10	100
	500x500	95	1	10	100
	500x500	91	1	10	100

Iš sudarytos lentelės matome, kad turime gauti 18 eksperimento rezultatų.

4.1.2. Eksperimento atlikimas

Pirmiausia daugybą atliksime su viena iteracija ir stebėsime laiką, per kūrį buvo sudaugintos matricos. Gauti rezultatai tokie:

9 lentelė. Rezultatai gauti atlikus vieną iteraciją

Dauginimo tipas	Matricų formatas	Išretinimas	Atlikta iteracijų	Laikas
Paprastų matricų daugyba	100x100	97	1	00:00:02
	100x100	95	1	00:00:02
	100x100	91	1	00:00:02
	250x250	97	1	00:00:37
	250x250	95	1	00:00:37
	250x250	91	1	00:00:37
	500x500	97	1	00:05:58
	500x500	95	1	00:05:58
	500x500	91	1	00:05:58
Suspaustų matricų daugyba	100x100	97	1	00:00:00.6
	100x100	95	1	00:00:00.7
	100x100	91	1	00:00:00.8
	250x250	97	1	00:00:21
	250x250	95	1	00:00:24
	250x250	91	1	00:00:27
	500x500	97	1	00:04:25
	500x500	95	1	00:04:59
	500x500	91	1	00:05:31

Atlikę vieną iteraciją, mes matome, kad daugybą pagal eilutes atliekama greičiau. To ir reikėjo tikėtis, nes daudyboje pagal eilutes yra praleidžiami nuliai. Tarkime turime matricą 100x100 formato su išretinimu 97%:

10 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 26 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 0 0 0 0 0 5 0 0 28 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 10 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 46 0 0 0 0 0 0 0 0 0 0 0 33 0 0 0 0 0 0 0 0 0 0 19 0 3 0 0 0 0 0 0 43 0 0 8 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 14 0 36 22 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0

0 0 0 0 0 0 19 0 0 0 0 0 0 0 33 0 0 0 0 36 0 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 3 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 0 0 43 0 0 0 0 9 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0 0 0 0 14 0 26 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0

0 14 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 42 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 46 42 0 0 0 24 0 48 0 0 0 0 0 0 0 36 0 0 0 0 23 0 0 0 0 0 7 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 0 0 17 0 0 20 0 27 0 0 0 0 0 0

0 37 0 0 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 35 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 13 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 48 0 0 0 0 0 0 0 0 0 0 0 0 0 0 36 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 40 0 0 0 0 0 0 0 0 49
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 48 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 31 0 0 0 0 0 47 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 0 0 0 0 0 26 0 0 34 0 0 36 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 39 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 11 40 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 21 0 34 0 0 0 0 0 0 0 0 0 0 0 0 19
0 0 11 43 0 16 0 0 0 0 0 0 0 0 0 0 11 29 0 0 14 0 0 0 0 0 0 2 0 0 0 1 0 0 20 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 10 0 0 0 0 0 11 6 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 6 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 0 0 0 0 33 0 0 0 0 17 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 39 37 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 25 0 46 0 0 0 0 26 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 37 21 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 20 0 0 0 0 0 0 0 0 0 0 35 0 46 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 9 0 0 0 0 16 0 0 0 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0 33 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 15 22 0 0 0 0 0 0 0 0 0 34 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 42 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 10 0 11 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 22 0 0 0 0 0 0 28 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 17 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0 0 29 0 45 0 0 0 0 0 0 0 0 28 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0
4 0 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 17 0 0 0 0 43 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 47 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 48 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 14
0 0 0 0 0 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 39 3 0 0 0 0 14 0 0 31 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0
0 0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 2 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 0 0 27 3 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 43 0 0 0
0 40 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 5 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 21 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 10 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 29 0 0 0 0 0 0 0 46 0 0 0 4 0 0 0 45 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0
0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 42 29 0 0 0 0 48 0 0 0 38 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 5 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 49 0 0 16 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 16 0 49 33 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 25 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0 0 14 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 0 0 19 0 0 0 0 0 0 0 0 0 0 13 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 36 0 0 0 0 0 0 6 5 0 44 0 11 0 0 0 0 0 32 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 0 0 0 0 0 19
0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 9 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 14 0 0 0 0 42 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 27 29 0 0 7 0 31 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 0 0 16 0 0 0 0 0 0 0 0 0 40 0 0 0 0 25 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0
0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 34 0 0 0 35 0 6 0 21 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 3 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 31 0 16 0 0 0 0 0 0 0 0 0 0 46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 44 0 14 0 0 0 0 0 0 0 0 26 0 16 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 19 9 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 18 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 33 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 41 0 0 0 41 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 20 0 8 0 0 0 0 0 45 0 0 0 0 0 44 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0
0 47 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0 0 0 33 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 0 0 13 0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 36 0 0 0 0 0 39 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 1 0 23 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 25 0 0 0

0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 0 0 0

44 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 35 0 0 44 0 0 0 0 0 0 0 0 0 0 13 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 16 0 46 0 0 0 0 12 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 26 7 0 0 0 44 0 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 40 28 0 28 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 0 0 0 0 0 0 0 0

0 46 0 0 0 0 0 15 4 0 0 0 0 0 0 0 0 0 0 48 0 0 0 0 29 0 0 33 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 36 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 43 0 0 0 36 0 0 0 0 0 0 0 0 27 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 0 0 44 1 0 0 8

0 0 0 0 0 0 0 0 12 15 0 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 15 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 33 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 49

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 25 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 1 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 48 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 0 0 0 0 40 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 27 0 19 0 0 0 0 0 0 0 0 13 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 42 16 0 0 19 45 0 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 14 0 0 0 6 0 43 0

0 0 0 0 17 0 15 0 0 0 0 21 0 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 0 0 0 40 0 0 0 0 0

31 0 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 43 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 0 0 45 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 43 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 0 0 0 21 0 0 11 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 42 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 19 0 0 0 0 0 0 0 0 0 22 0 0 29 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 7 0 0 17 0 12 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 39 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0 46 0 0 0 0 0 0 0 0 38 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 13 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21 0 35 0 0 29 0 0 0 0

0 0 0 35 5 0 0 0 0 1 11 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 43 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 11 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 33 0 25 0 0 4 40 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 12 0 0 36 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 0 0 0 0 0 0 39 0 0

0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 40 45 0 0 0 0 0 0 0

0 0 0 0 0 0 35 0 0 48 0 0 0 0 0 0 0 0 0 0 0 39 0 0 0 1 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 23 0 42 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 27 0 0 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 0 0 0 0 0 37 0 0

0 29 0 0 0 0 0 0 0 0 39 40 0 0 0 0 0 0 0 0 0 0 0 44 0 0 0 42 0 0 25 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 29 0 0 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 19 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23

0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 21 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 39 0 0 0

0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 33 0 7 0 0 0 0 0 0 0 19 49 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0

16 44 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27

0 0 0 0 0 0 0 0 0 0 0 0 0 35 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 0 0 0 0 0 0 0 0 0 0 0 47 0 0 0 28 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 46 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 48 9 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 29 0 0 0 3 0 0 0 0 0 26 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 43 0 0 18 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 35 0 0 0 0 11 0 0 0 38 0 0 0 0 0 0 0 0 38 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 34 0 0 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 0 0 28 0 0 0 0 0 0 0 0 0 0

0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18 0 26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 49 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 41 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 0 49 0 0 0 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 35 0 0 0 21 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Kaip matome iš pateikto pavyzdžio 97% matricos užima nuliai. Dauginant matricas standartiškai visi elementai yra dauginami ir sumuojami. Tuo tarpu dauginant matricas patalpinus į eilutes, nuliai yra praleidžiami ir dauginami tik sveiki nenuliniai skaičiai. Todėl ir daugyba antru atveju yra įvygdoma greičiau.

Turėdami vienos iteracijos laikus, mes galime maždaug apskaičiuoti kiek truks visas ekperimentas. Turėdami pradinius duomenis, bei patikrinę ar tikrai metodų efektyvus skiriasi vienos iteracijos metu, mes atliekame eksperimentą su visomis iteracijomiss ir stebime visus užsiduotus parametrus.

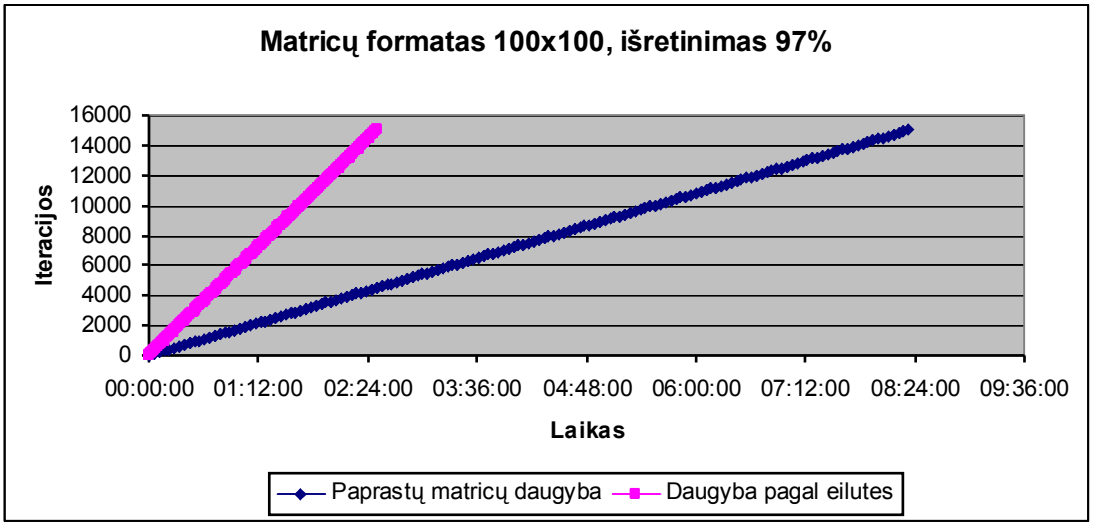
4.1.3. Eksperimento rezultatai

Po kiekvieno bandymo baterija buvo išnaujo įkraunama. Eksperimento tikslumui delninuko ekranas dauginimo metu išjungiamas. Rezultatai buvo surašyti į rezultatų failą. Visi gauti rezultatai sukelti į excel failą, kuris bus įdėtas prie priedų. Atlikę visą ekperimentą gauname rezultatus:

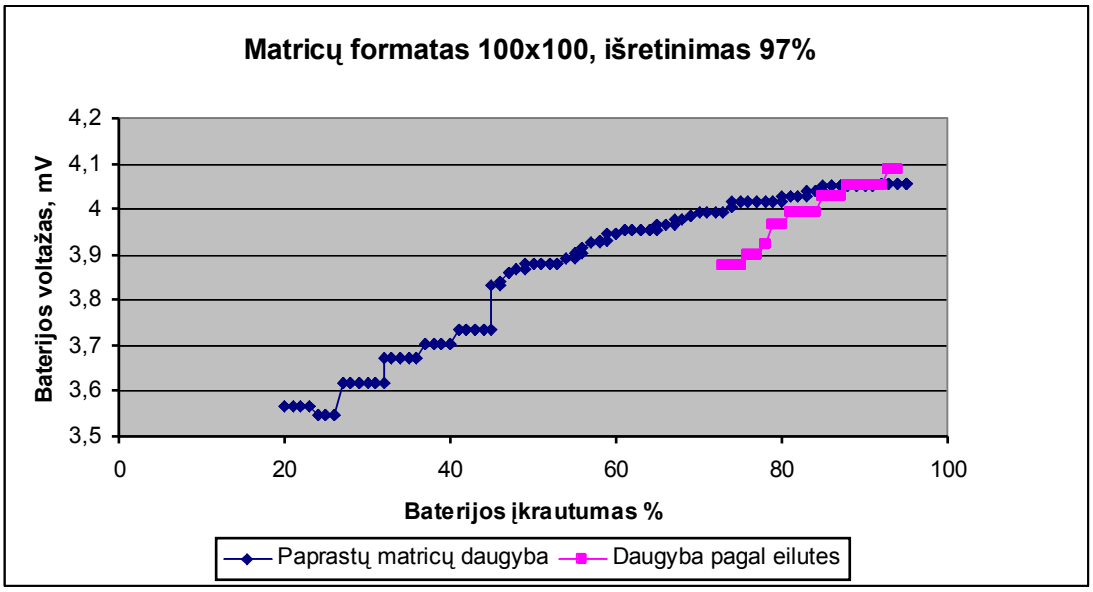
10 lentelė. Eksperimento rezultatai

Dauginimo tipas	Matricų formatas	Išretinimas	Atlikta iteracijų	Daugybos laikas	Baterijos įkrautumas % dauginimo pradžioje	Baterijos įkrautumas % dauginimo pabaigoje	Baterijos voltai dauginimo pradžioje, mV	Baterijos voltai dauginimo pabaigoje, mV
Paprastų matricų daugyba	100x100	97	15000	08:20:00	95	20	4,056	3,566
	100x100	95	15000	08:45:00	98	22	4,088	3,591
	100x100	91	15000	07:55:00	96	20	4,084	3,591
	250x250	97	1000	10:30:00	96	21	4,056	3,654
	250x250	95	1000	10:33:20	87	18	3,992	3,545
	250x250	91	1000	10:31:40	79	9	3,921	3,452
	500x500	97	100	09:46:30	98	20	4,088	3,654
	500x500	95	100	09:47:00	99	21	4,091	3,671
	500x500	91	100	09:46:20	97	20	4,083	3,654
Suspaustų matricų daugyba	100x100	97	15000	02:30:00	96	73	4,088	3,876
	100x100	95	15000	02:54:56	91	69	4,053	3,709
	100x100	91	15000	03:20:08	88	61	4,051	3,721
	250x250	97	1000	06:05:00	91	42	4,053	3,833
	250x250	95	1000	06:43:20	87	36	3,992	3,571
	250x250	91	1000	07:25:00	90	39	4,053	3,833
	500x500	97	100	07:21:40	73	21	3,992	3,794
	500x500	95	100	08:18:20	92	32	4,053	3,819
	500x500	91	100	09:11:40	88	19	4,051	3,789

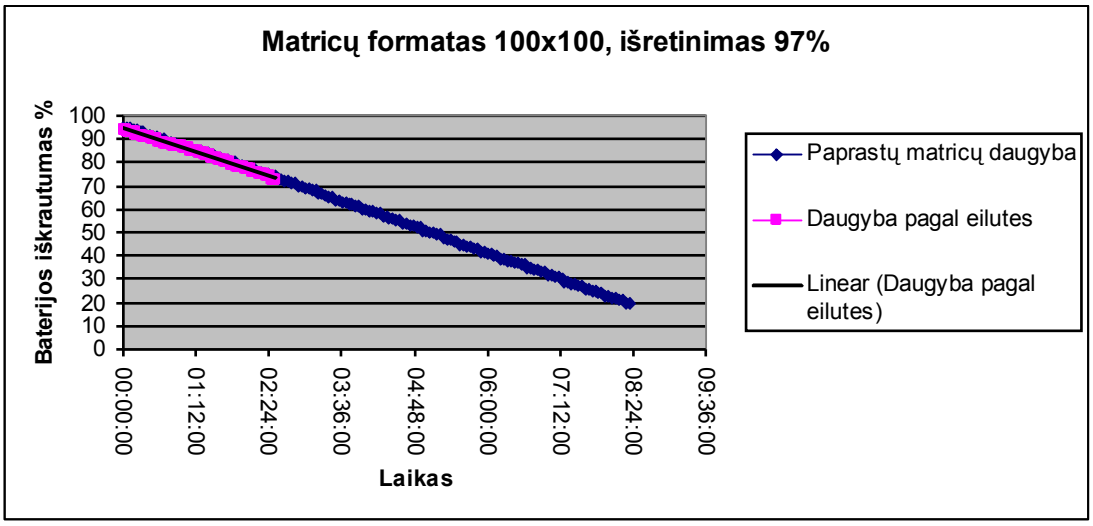
Iš gautų rezultatų matome, kad paprasta daugyba su matricomis užtrunka ilgiau nei dauginant eilučių saugojimo metodu. Taip pat pastebime, kad paprastoje matricų daugyboje, dauginimo greičiui neturi reikšmės išretinimas. Šiek tiek kai kurie laikai skiriasi, nes dauginant atsiranda paklaidos dėl kitų programų veikimo, tačiau didelių skirtumų nėra. Baterijos voltai taip pat ne visada vienodi prie tam tikro baterijos įkrautumo procentais. Įtampa atliekant dauginimą šiek tiek keičiasi, tačiau laikui dauginimo laikui didėjant ir baterijai senkant, mažėja baterijos voltai. Baterijos įkrautumo procentų skirtumas pradėjus daugybą ir baigus, išlieka vienodas su tam tikra paklaida, kai laikai dauginimo panašūs. Tai yra jei dauginama buvo 10 valandų, tai bet kokiu dauginimo būdu daugintumėm su bet kokiu matricų formatu, mes gauname vienodą procentų skirtumą, todėl baterijos iškrautumas priklauso nuo laiko. Kad galėtume palyginti naudotus dauginimo tipus, iš gautų rezultatų brėžiame grafikus.



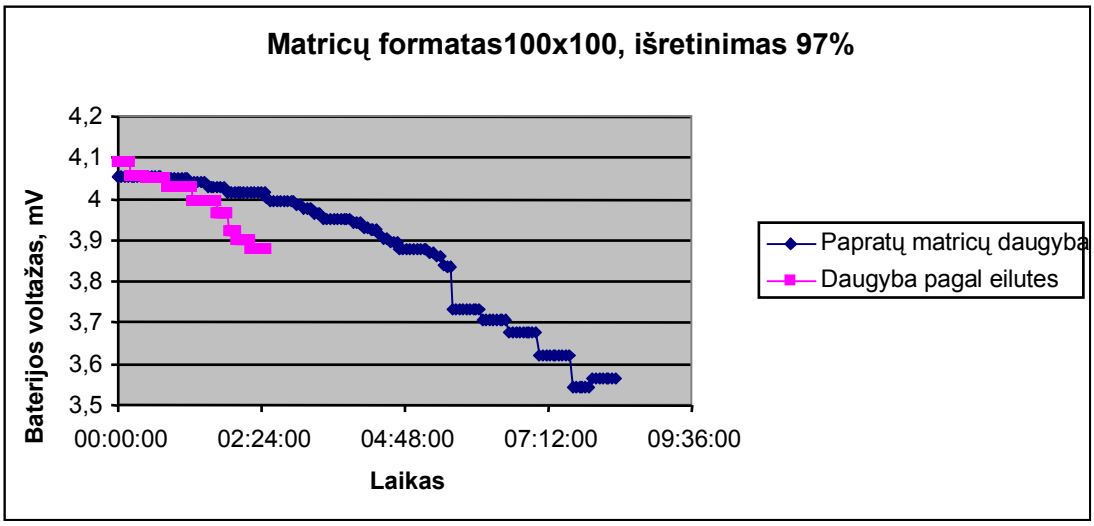
a)


Matricų formatas 100x100, išretinimas 97%
Baterijos voltažas, mV
4,2
4,1
4
3,9
3,8
3,7
3,6
3,5
0
20
40
60
80
100
Baterijos įkrautumas %
Paprastų matricų daugyba
Daugyba pagal eilutes

b)


Matricų formatas 100x100, išretinimas 97%
Baterijos iškrautumas %
100
90
80
70
60
50
40
30
20
10
0
00:00:00
01:12:00
02:24:00
03:36:00
04:48:00
06:00:00
07:12:00
08:24:00
09:36:00
Laikas
Paprastų matricų daugyba
Daugyba pagal eilutes
Linear (Daugyba pagal eilutes)

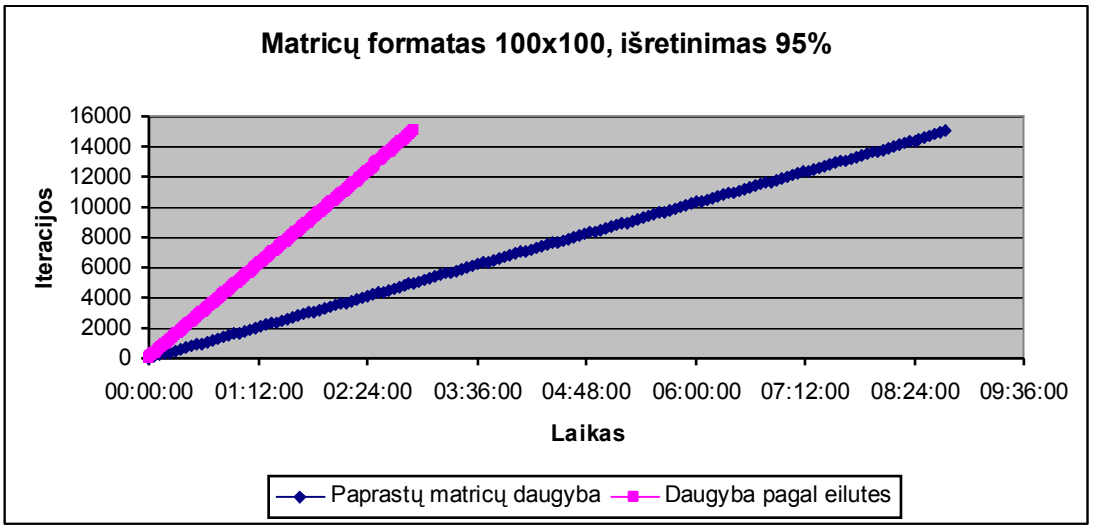
c)



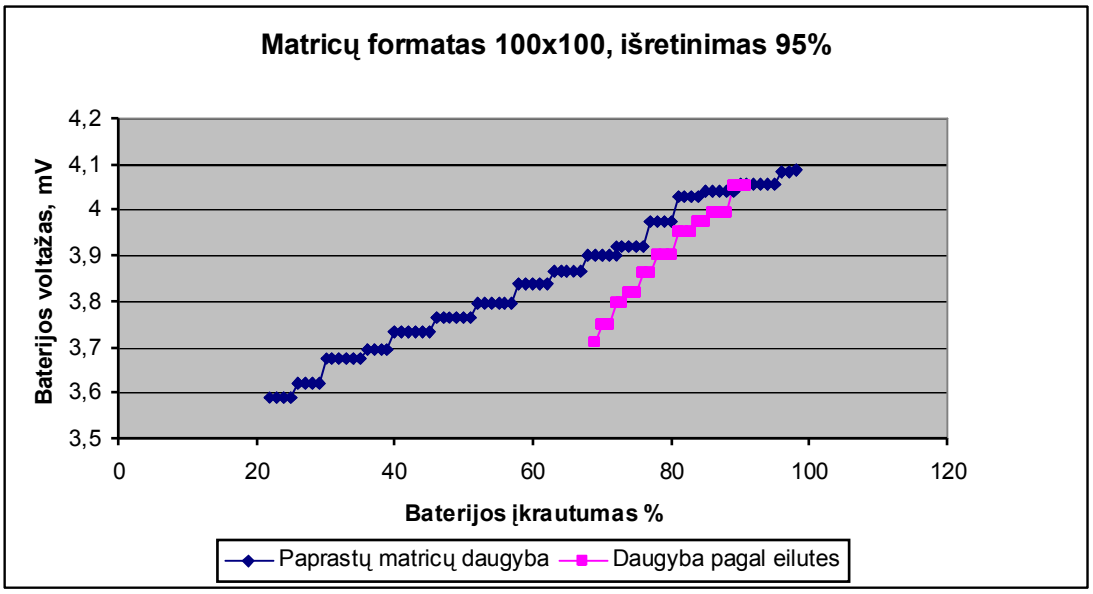
d)

10 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 100x100 ir išretinimas 97%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

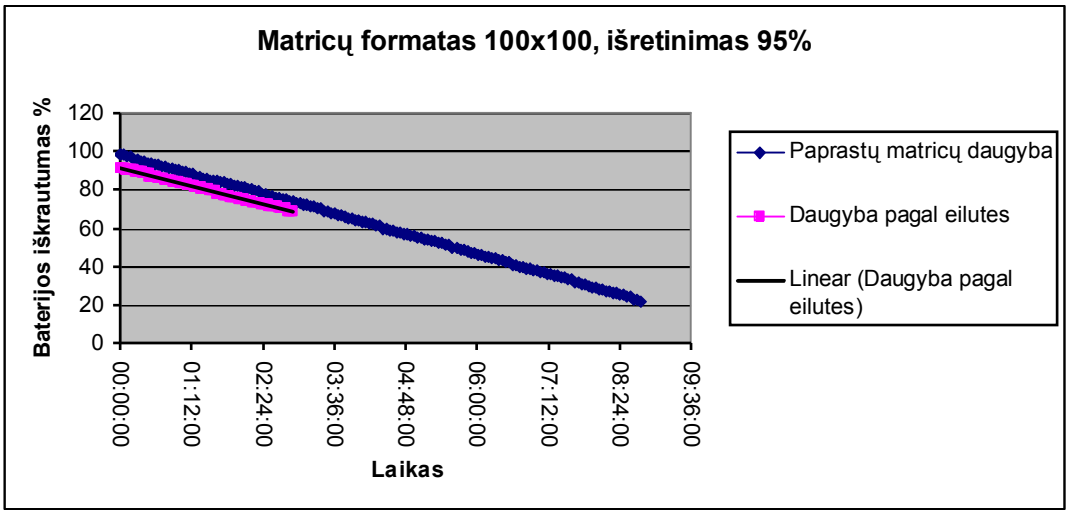
Iš a) grafiko matome, kad daugybos pagal eilutės tipą yra greitesnis nei paprasta matricų daugyba. Atliekant tiek pat iteracijų dauginant pagal eilutes užtrunkame maždaug keturis kartus trumpiau. Iš to galime spręsti, kad naudojant nedidelio formato matricas, daugyba pagal eilutes yra efektyvi. Todėl matome kad dauginant trumpesnį laiką, baterijos voltų bei įkrautumo procentų sumažėjo nedaug, tai yra baterija nusėdo mažiau nei dauginant matricas standartiškai. Pagal baterijos įkrautumo procentus ir laiką matome, kad dauginant matricas abiem daugyboms tipais, baterija sėdo vienodai. Tai įrodo faktą, kad baterijos sėdimas priklausė tik nuo baterijos sodinimo laiko. Toliau brėžiame grafikus, kai matricų išretinimas 95%.


Matricų formatas 100x100, išretinimas 95%
Iteracijos
16000
14000
12000
10000
8000
6000
4000
2000
0
00:00:00
01:12:00
02:24:00
03:36:00
04:48:00
06:00:00
07:12:00
08:24:00
09:36:00
Laikas
Paprastų matricų daugyba
Daugyba pagal eilutes

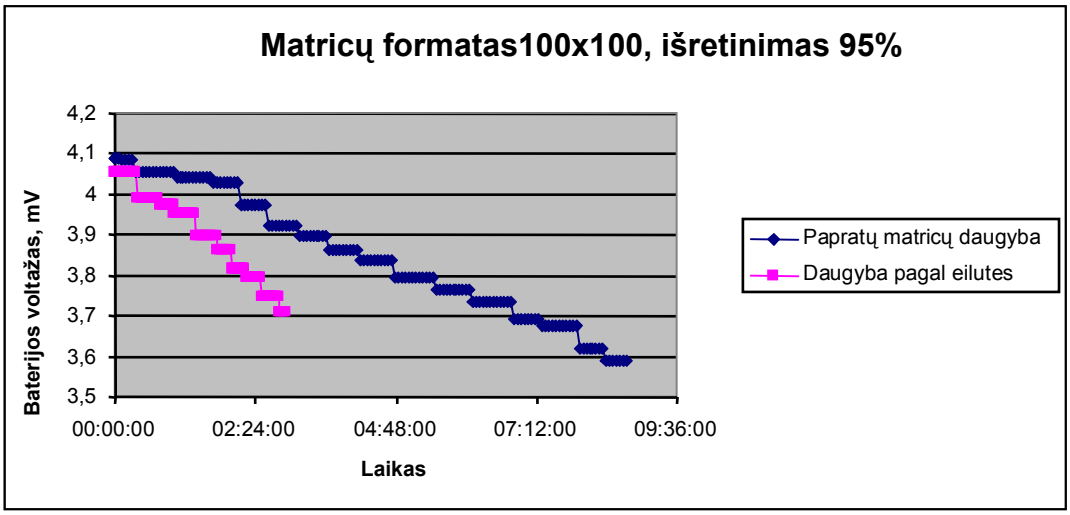
a)


Matricų formatas 100x100, išretinimas 95%
Baterijos voltažas, mV
4,2
4,1
4
3,9
3,8
3,7
3,6
3,5
0
20
40
60
80
100
120
Baterijos įkrautumas %
Paprastų matricų daugyba
Daugyba pagal eilutes

b)



c)



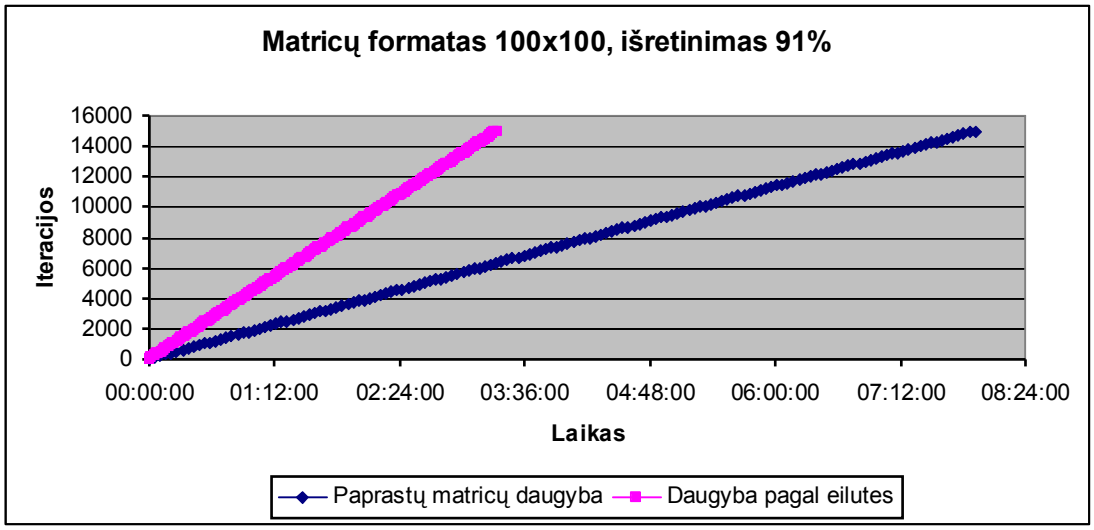
d)

11 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 100x100 ir išretinimas 95%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

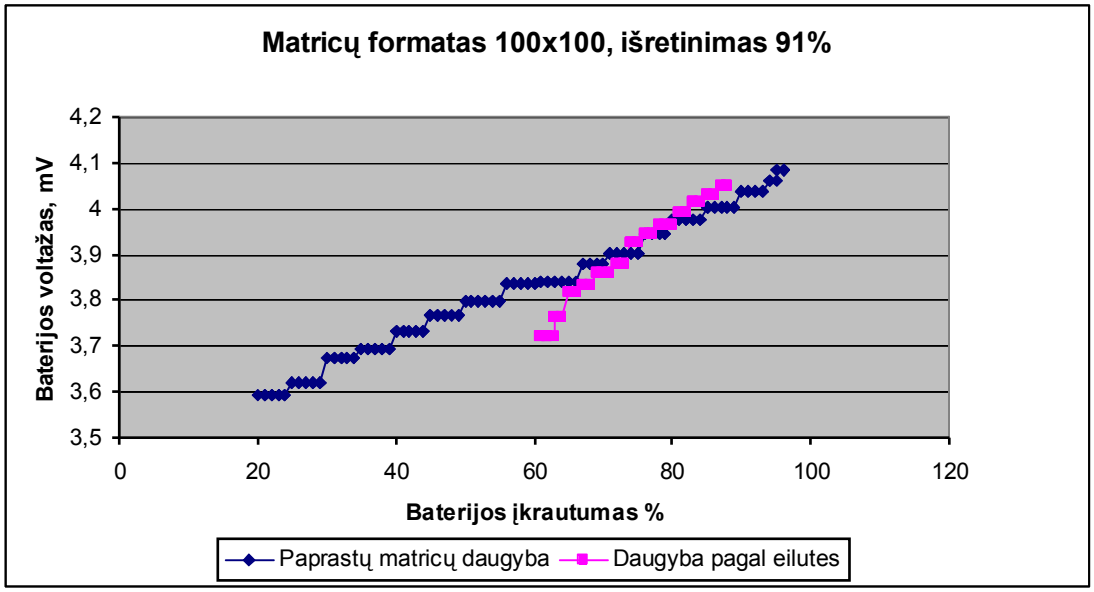
Kaip ir su 97% išretinimu, dauginat pagal eilutes matricos buvo sudaugintos greičiau nei paprasto daugybos metu. Tačiau išretinimui esant 91% daugyba užtruko ilgiau nei su 97% išretinimu. Tai siejama su tuo, kad matricoje yra mažiau nulinių elementų ir programa turi sudauginti daugiau matricos elementų, kas prailgina daugybos laiką. Skiriasi šiek tiek įtampa. Dauginant antruoju metodu (pagal eilutes) įtampos kritimas yra didesnis ir spartesnis, tačiau baterijos įkrautumas išlieka toks pat kaip ir dauginant

pirmuoju tipu. Šie skirtumai gali atsirasti dėl pilnai neįkrautos baterijos ir ekperimento metu paimtų duomenų tikslumo. Įrašymo metu gali būti padarytos paklaidos, kai baterija naudoja kitas programas.

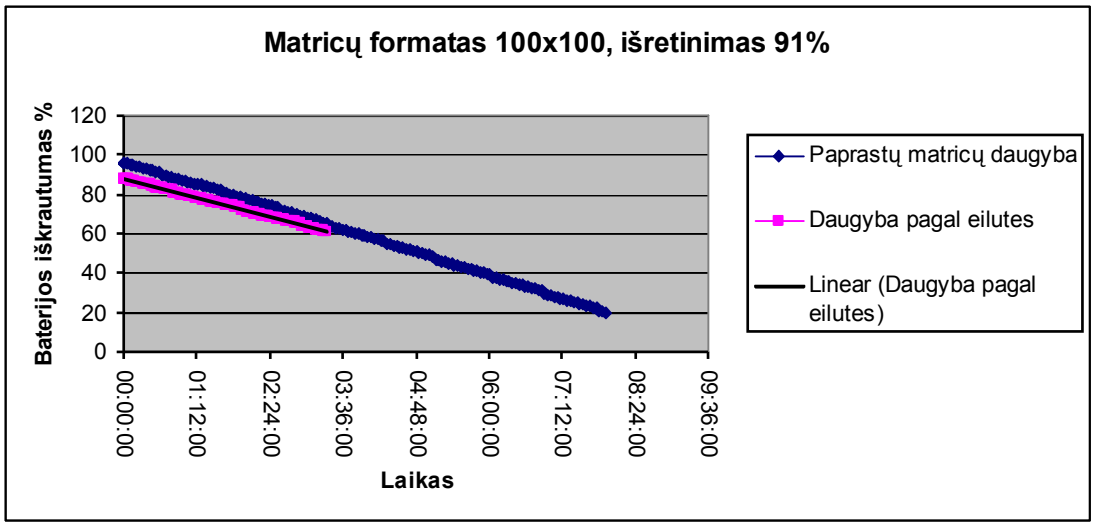
Toliau brėžiame grafikus kai išretinimas yra 91%.



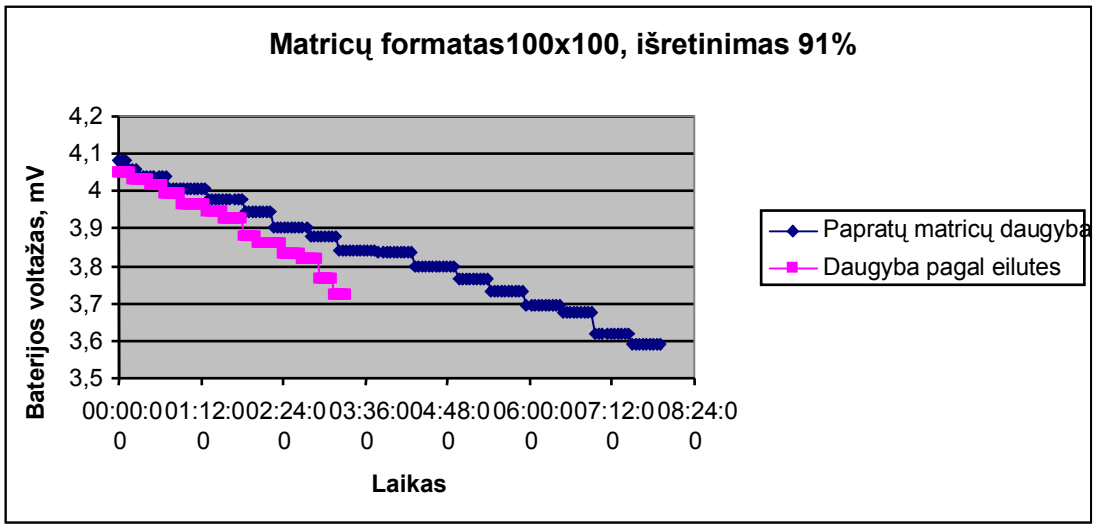
a)



b)



c)

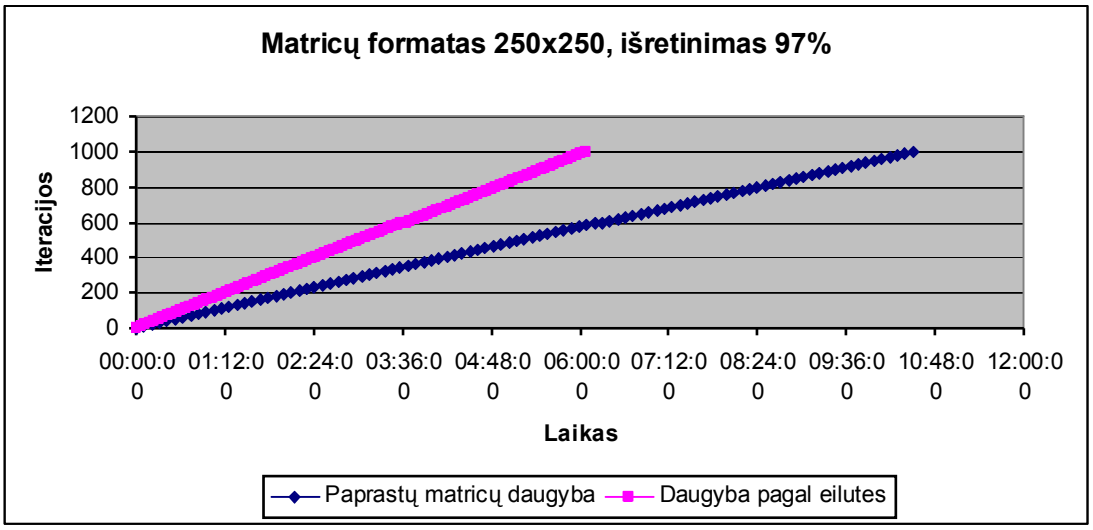


d)

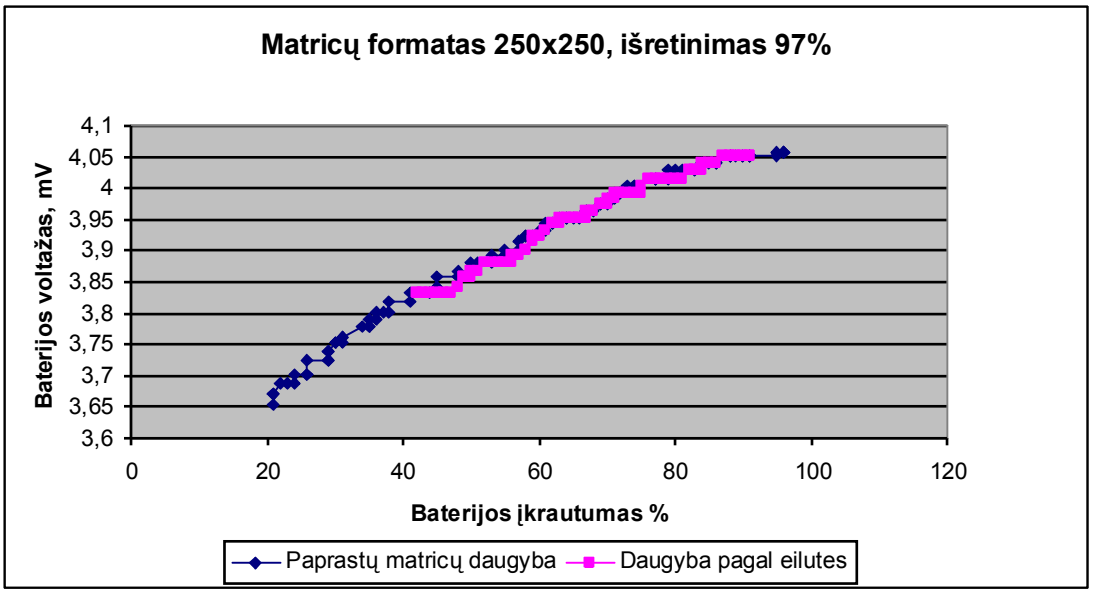
12 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 100x100 ir išretinimas 91%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Iš gautų duomenų, galime matyti, kad sumažėjus išretinimui, dauginimo laikas pailgėjo naudojant daugybos pagal eilutes būdą. Tuo tarpu matricas dauginant paprastu būdu išretinimas neturėjo didelės įtakos (daugybos laiko skirtumai nėra dideli). Baterijos sodinimas taip pat kito vienodai, kaip ir su kitais išretinimo procentais ir priklausė nuo daugybos trukmės.

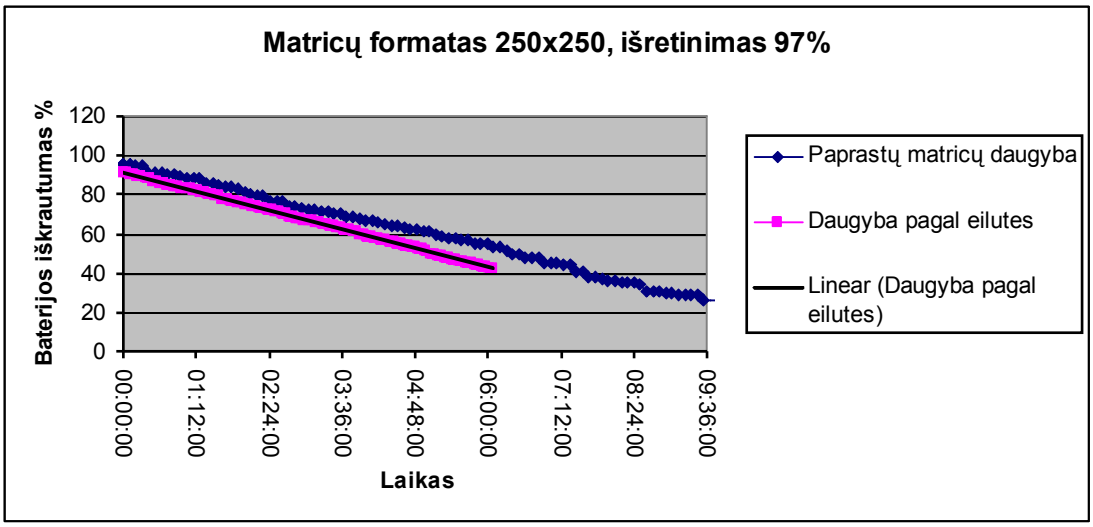
Sekantys grafikai bus su 250x250 formato matricomis.



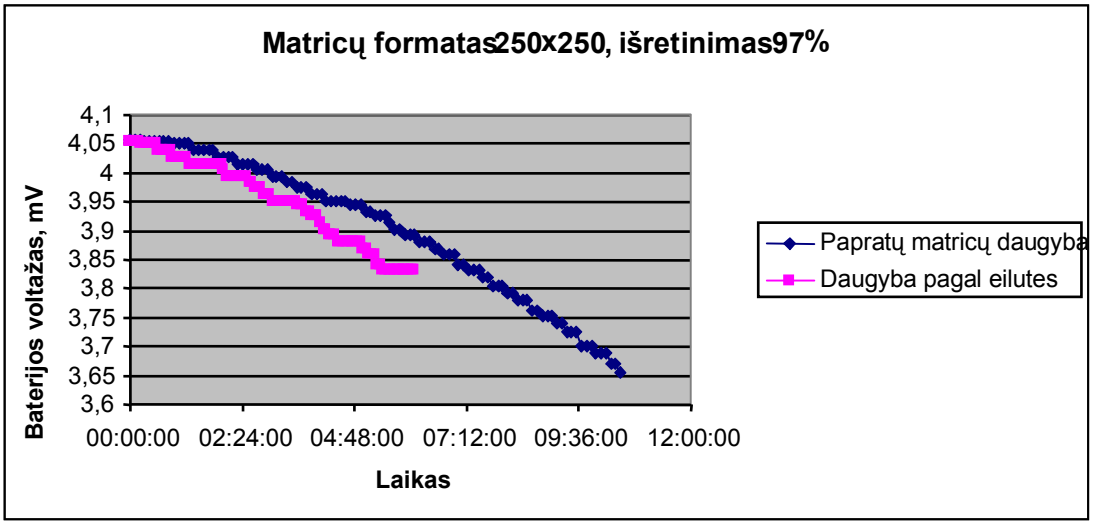
a)



b)



c)

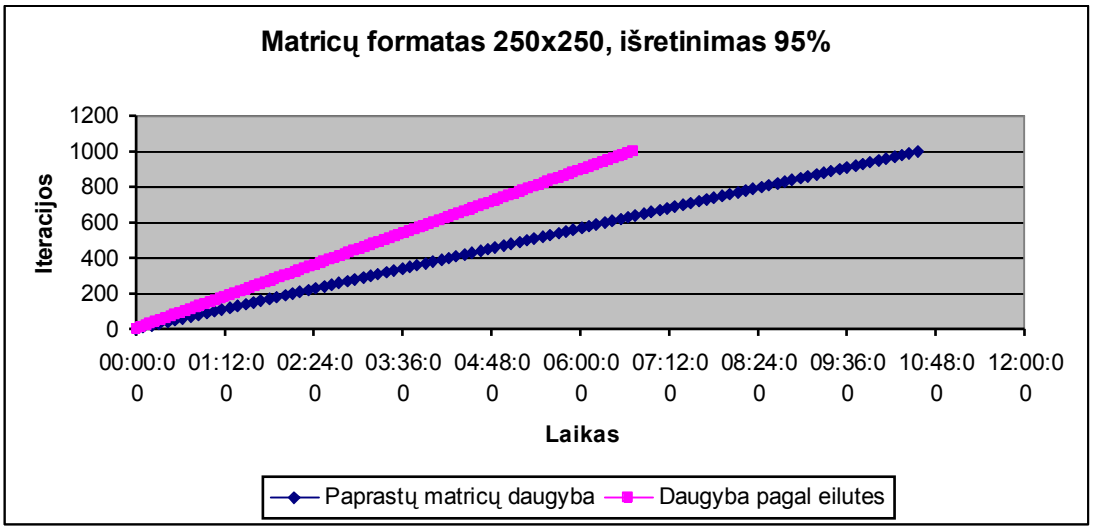


d)

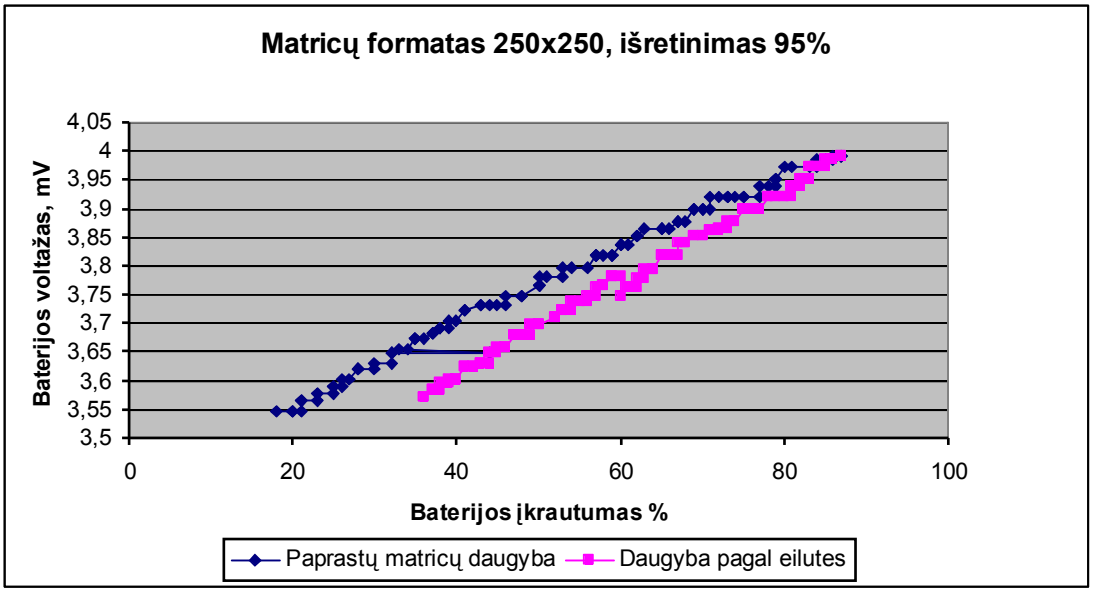
13 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 250x250 ir išretinimas 97%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Naudojant didesnio formato matricas, iteracijų skaičių sumažiname iki 1000. Kaip matome iš a) grafiko, laiko skirtumas tarp dauginimo būdų sumažėjo. Taip pat padidėjo ir baterijos nusodinimas.

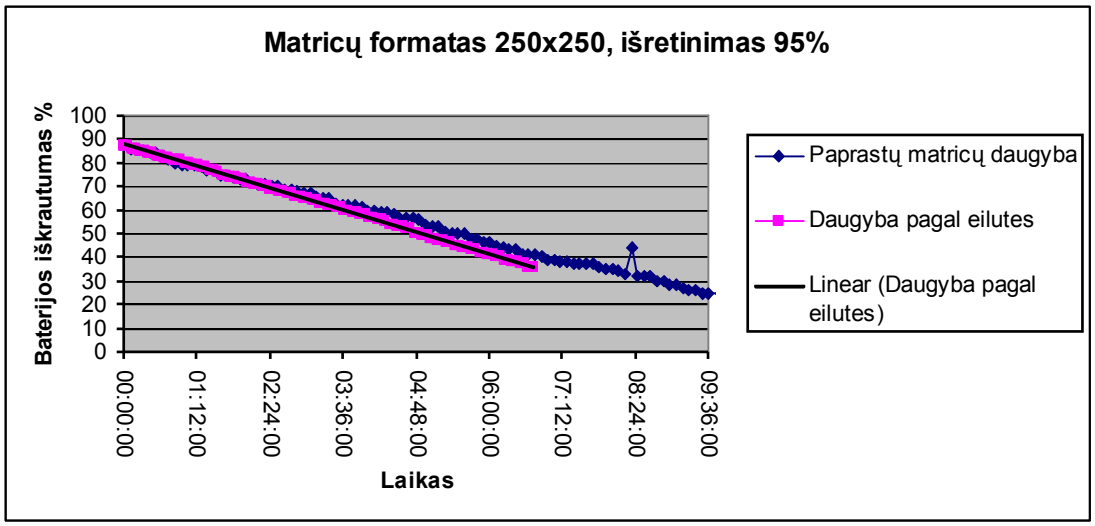
Brėžiame sekančius grafikus, kai išretinimas yra 95%.


Matricų formatas 250x250, išretinimas 95%
Iteracijos
1200
1000
800
600
400
200
0
00:00:00
01:12:00
02:24:00
03:36:00
04:48:00
06:00:00
07:12:00
08:24:00
09:36:00
10:48:00
12:00:00
Laikas
Paprastų matricų daugyba
Daugyba pagal eilutes

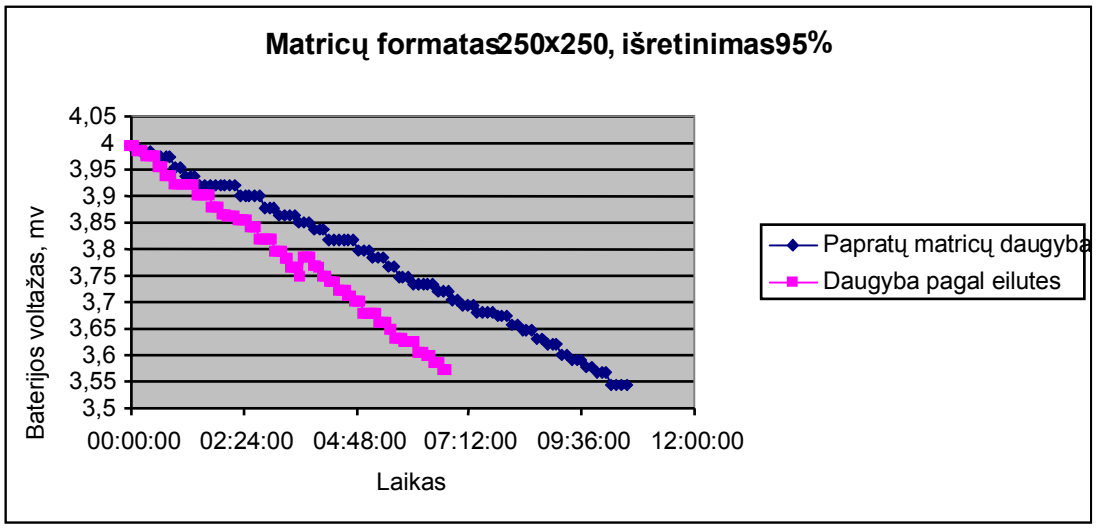
a)


Matricų formatas 250x250, išretinimas 95%
Baterijos voltažas, mV
4,05
4
3,95
3,9
3,85
3,8
3,75
3,7
3,65
3,6
3,55
3,5
0
20
40
60
80
100
Baterijos įkrautumas %
Paprastų matricų daugyba
Daugyba pagal eilutes

b)



c)

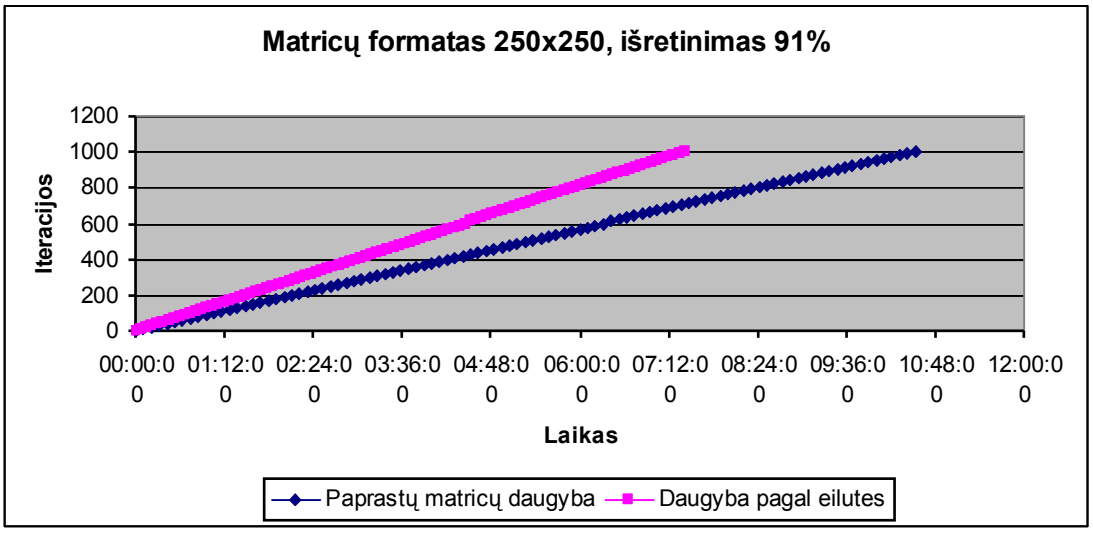


d)

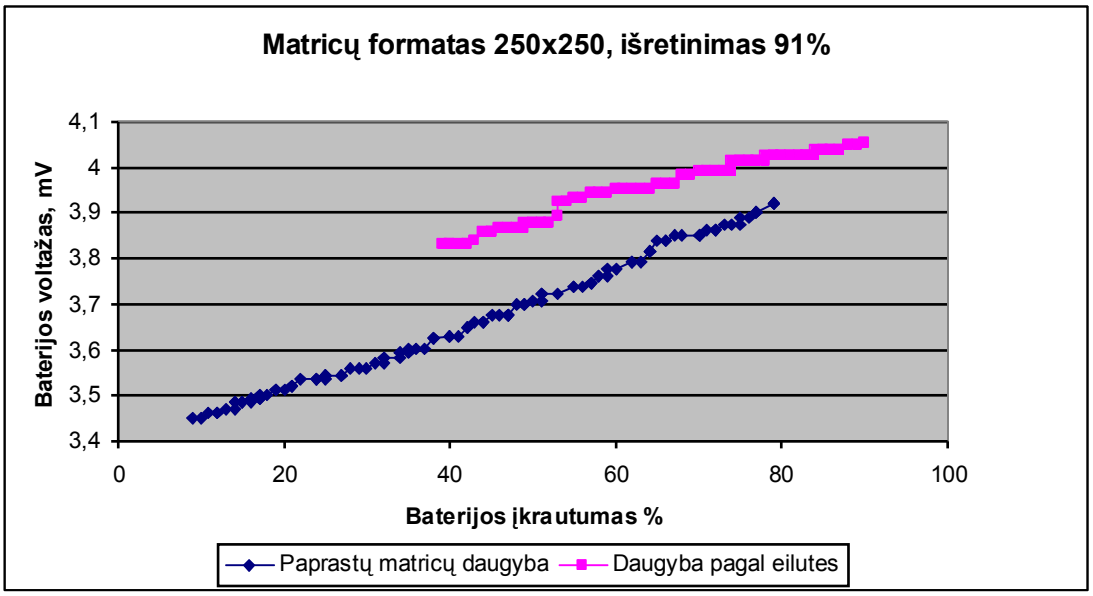
14 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 250x250 ir išretinimas 95%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Sumažinus išretinimą nuo 97% iki 95%, dauginimo laikas dauginant pagal eilutes dar pailgėjo. Taip pat padidėjo ir baterijos nusodinimas. Tuo tarpu naudojant dauginant paprastas matricas, dauginimo laikai bei nusodinimas beveik nepakito.

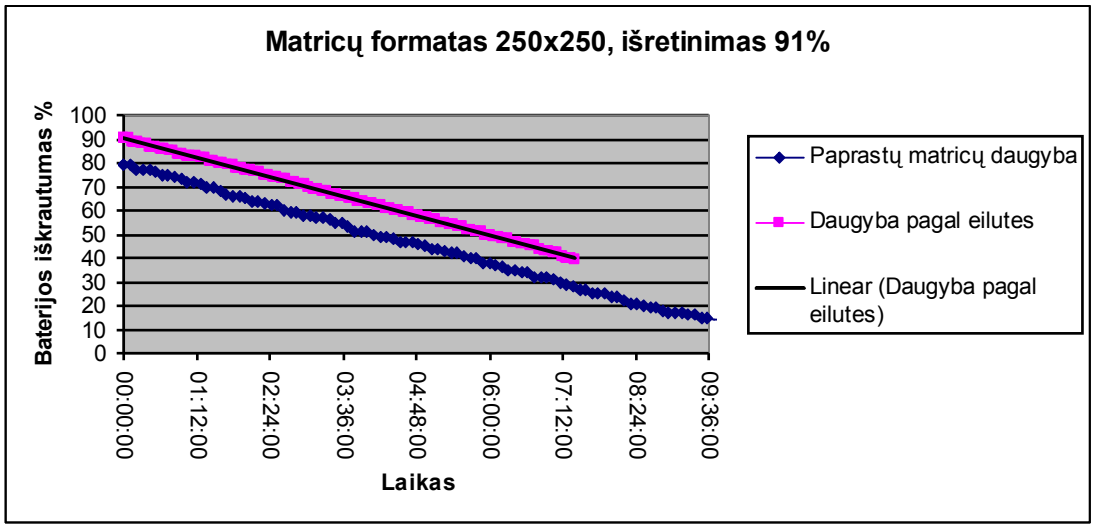
Sudarome grafikus su 91% išretinimu.


Matricų formatas 250x250, išretinimas 91%
Iteracijos
1200
1000
800
600
400
200
0
00:00:00
01:12:00
02:24:00
03:36:00
04:48:00
06:00:00
07:12:00
08:24:00
09:36:00
10:48:00
12:00:00
Laikas
Paprastų matricų daugyba
Daugyba pagal eilutes

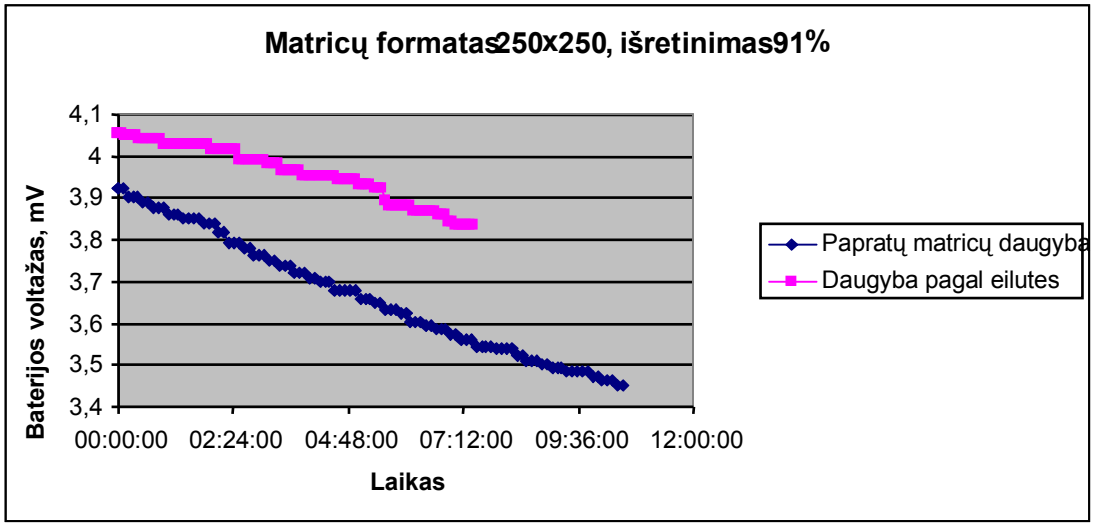
a)


Matricų formatas 250x250, išretinimas 91%
Baterijos voltažas, mV
4,1
4
3,9
3,8
3,7
3,6
3,5
3,4
0
20
40
60
80
100
Baterijos įkrautumas %
Paprastų matricų daugyba
Daugyba pagal eilutes

b)



c)

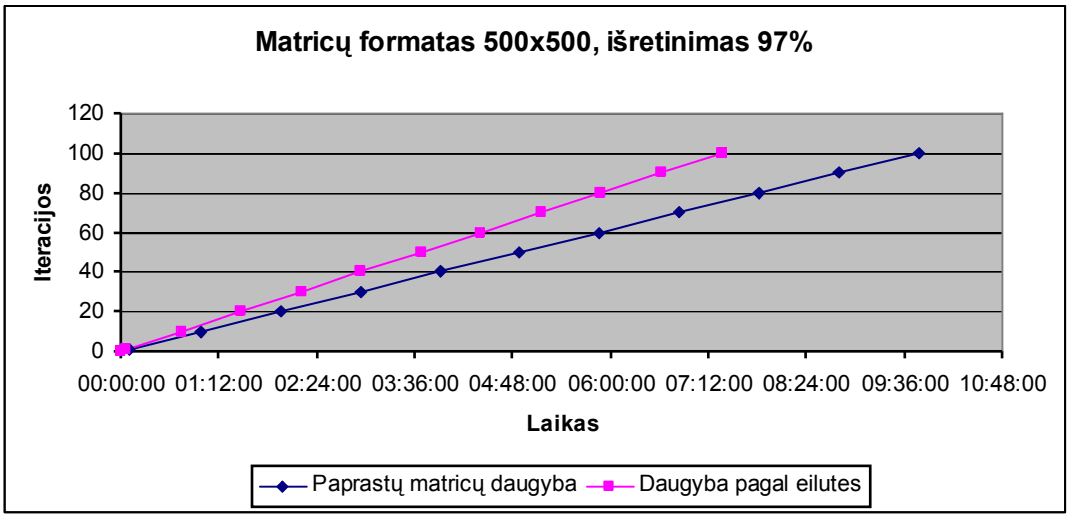


d)

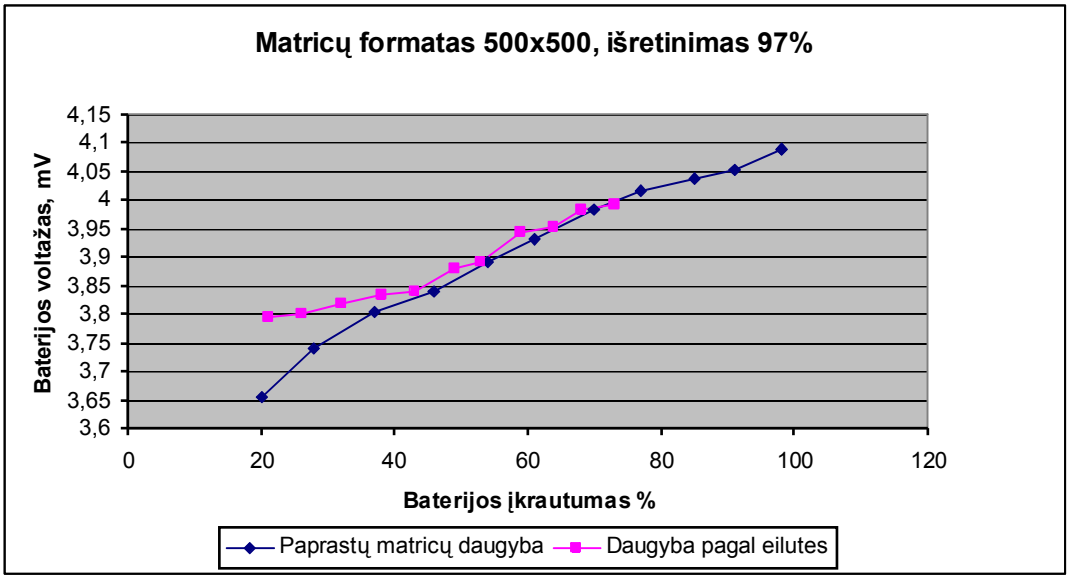
15 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 250x250 ir išretinimas 91`%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Lyginat dauginimo laikus, kai dauginamos 250x250 formato matricos daugybos būdu pagal eilutes, ilgiausiai truko daugyba prie 91% išretinimo. Toliau mažėjant išretinimui, dauginimo laikas artėja prie paprastos daugybos trukmės, taip pat didėja ir baterijos energijos suvartojimas. Tuo tarpu baterijos nusėdimas ir voltai mažėja vienodais tempais. Grafikuose matosi linijų lygegretumas, kuris atsirado dėl pradinių reikšmių skirtumo.

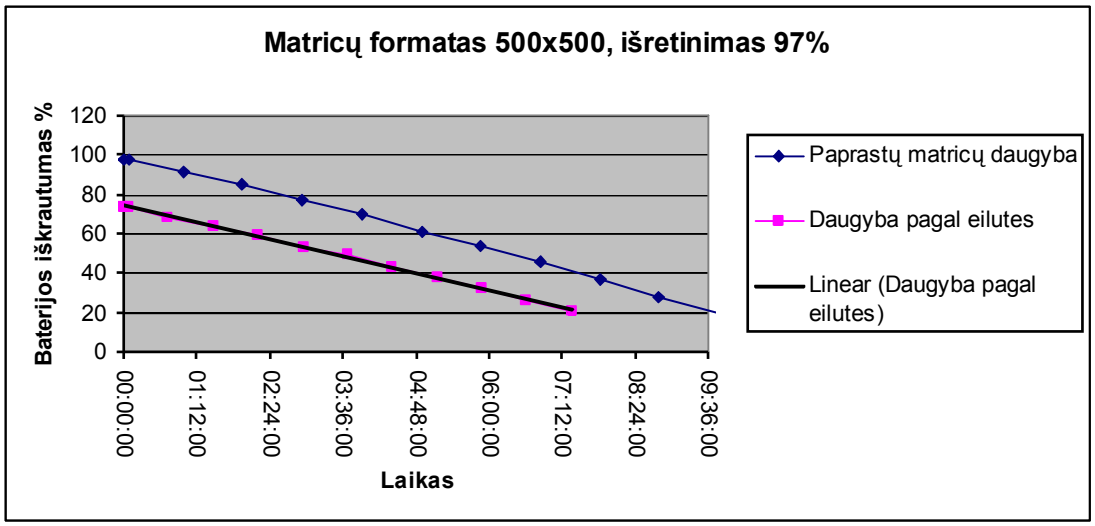
Toliau sudarome grafikus kai matricų formatas yra 500x500 ir išretinimas 97%.



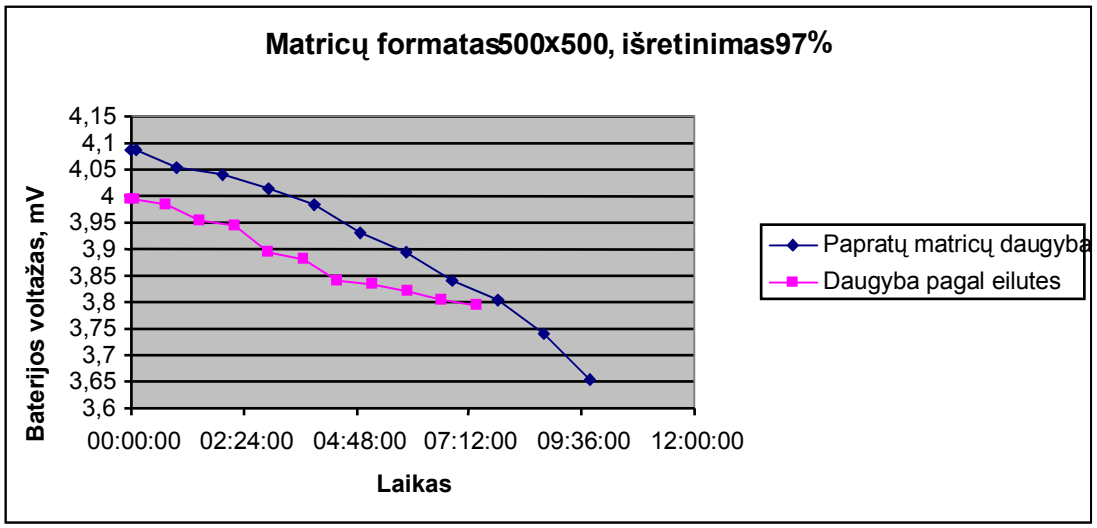
a)



b)



c)

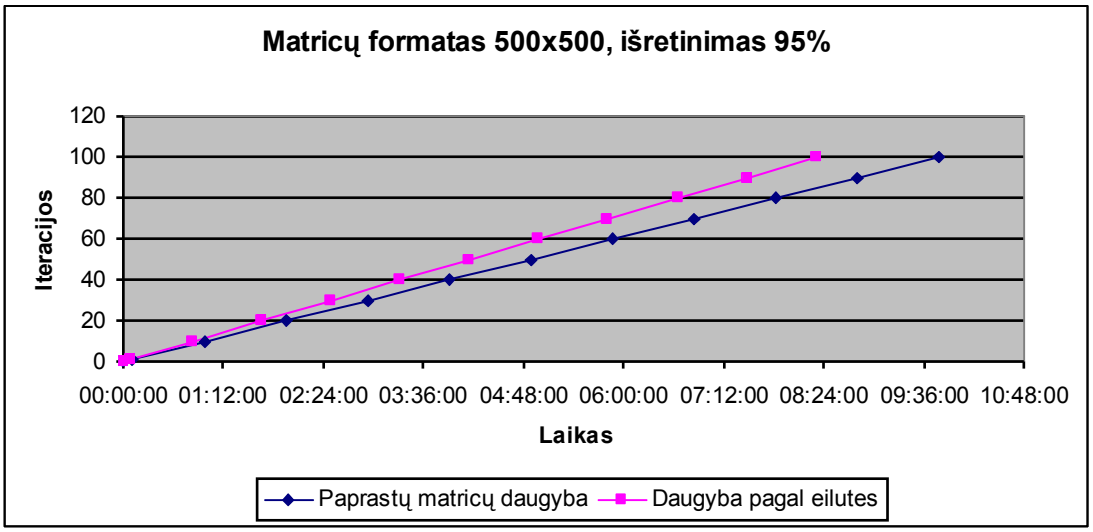


d)

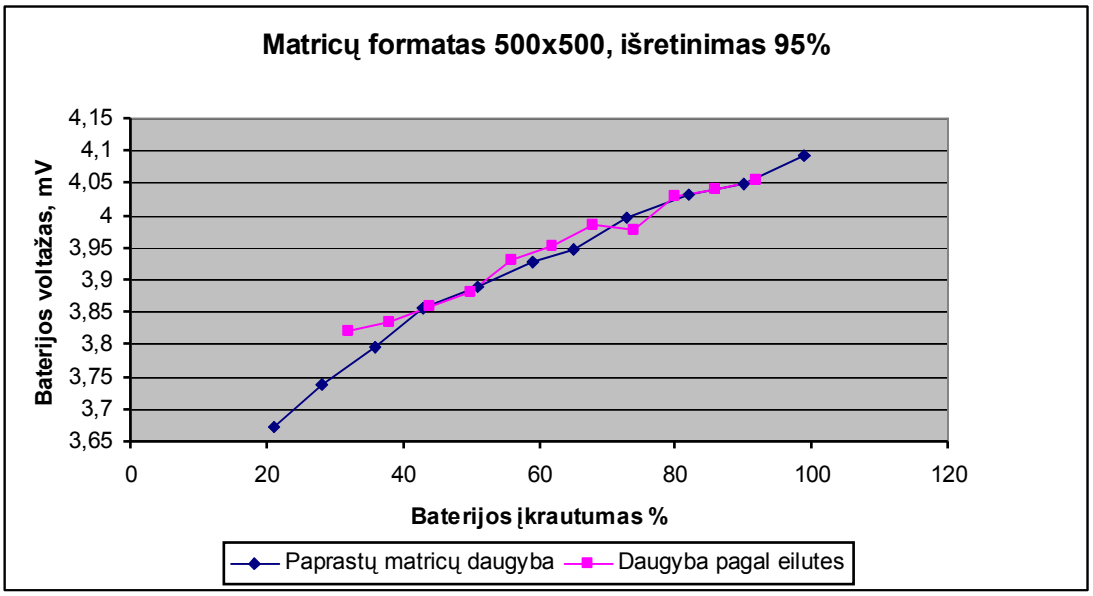
16 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 500x500 ir išretinimas 97`%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Eksperimentu metu naudojant 500x500 matricų formatus, iteracijų skaičių mažiname iki 100, nes dauginant daugiau kartų, baterija pilnai išsikrauna ir mes vistiek neužfiksuojame rezultatų. Kaip ir kitų formatų matricų daugyboje, 500x500 matricas sudaugina greičiau naudojant daugybos pagal eilutes daugybos tipą. Didėjant formatui ir didėjant skaičaivimams, laiko skirtumai tarp dauginimo būdų mažėja.

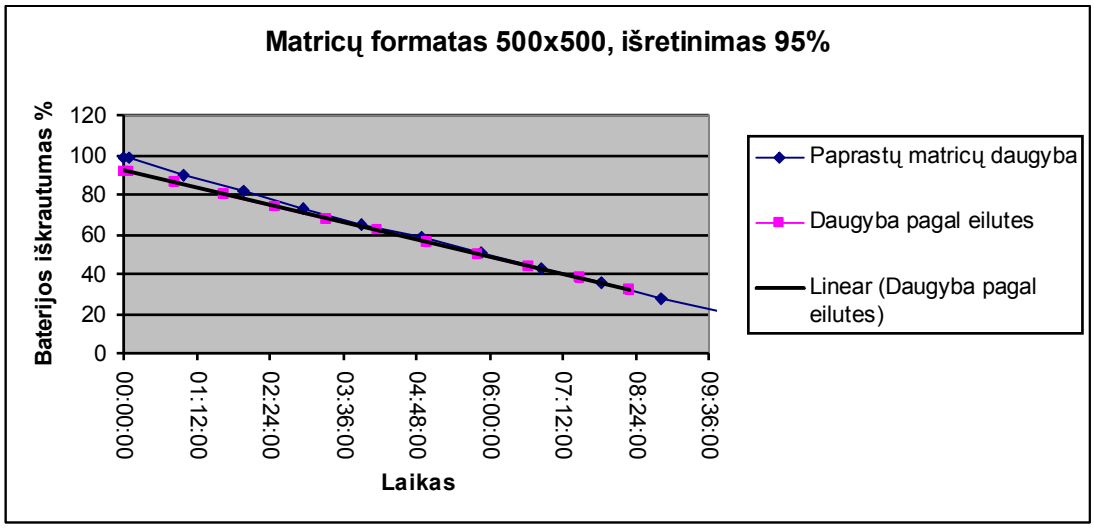
Brėžiame grafikus kai išretinimas 95%.



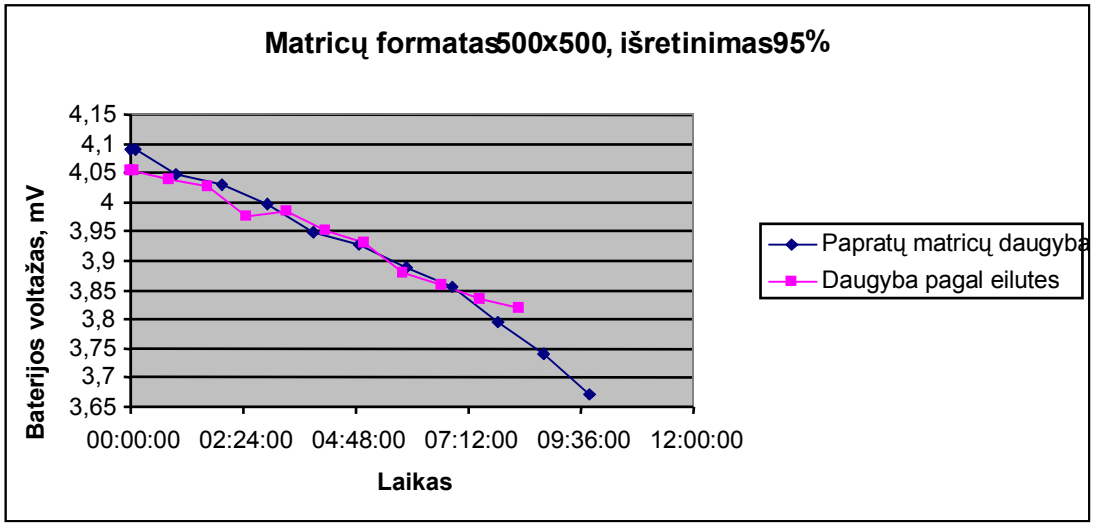
a)



b)



c)

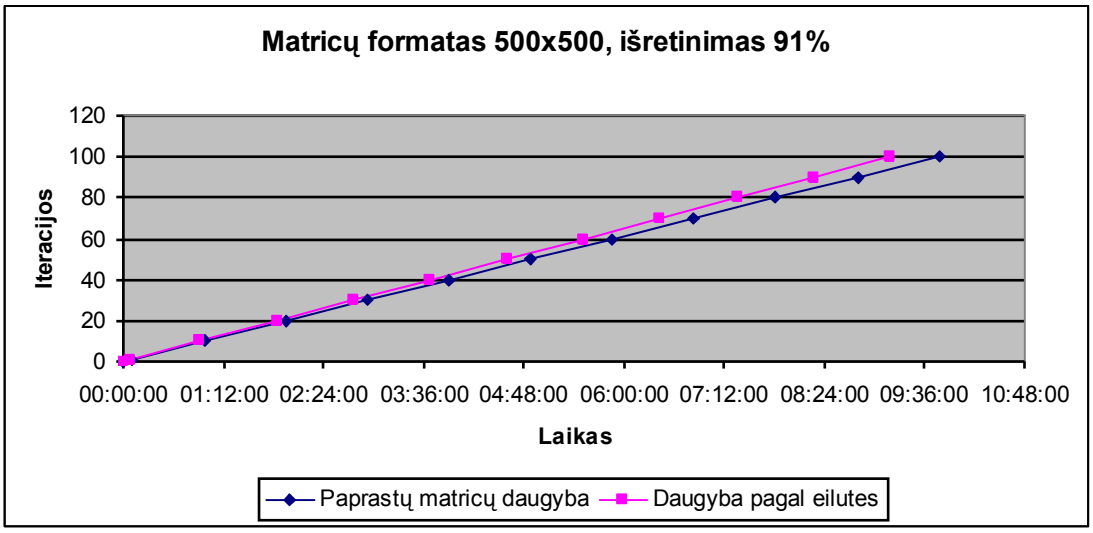


d)

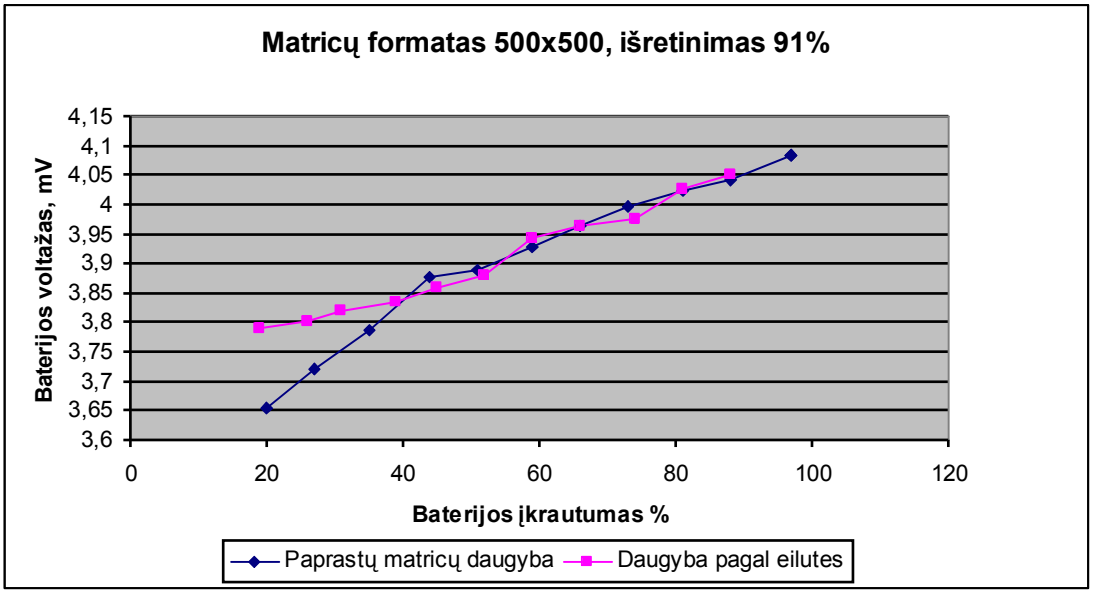
17 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 500x500 ir išretinimas 95`%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Sumažėjus išretinimui iki 95%, pailgėjo dauginimo laikas naudojant suspaustų matricų dauginimo tipą, lyginant su 97% matricų išretinimu. Ilgėjant dauginimo laikui, taip pat didėja ir baterijos energijos suvartojimas.

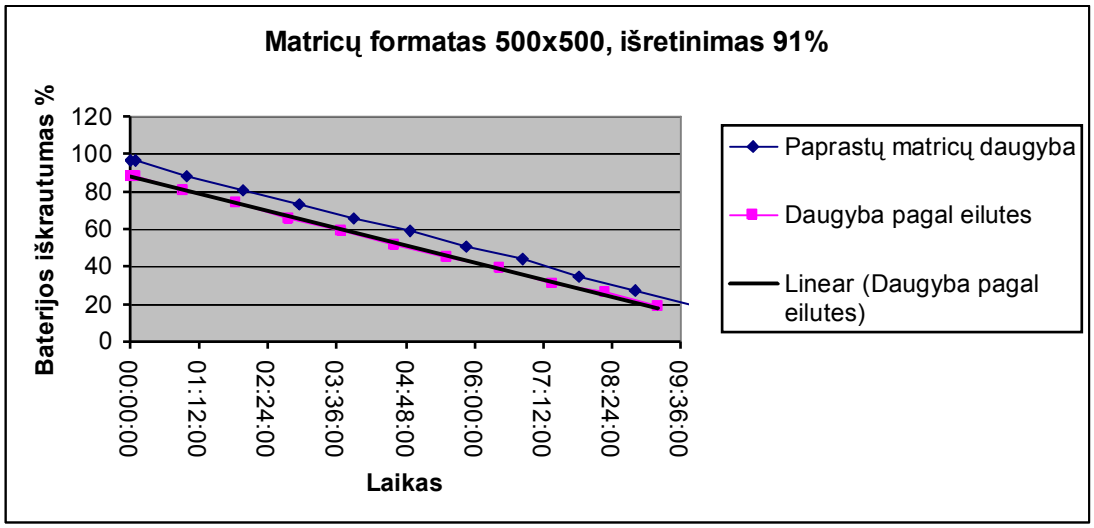
Brėžiame grafikus su 91% išretinimu.


Matricų formatas 500x500, išretinimas 91%
Iteracijos
120
100
80
60
40
20
0
00:00:00
01:12:00
02:24:00
03:36:00
04:48:00
06:00:00
07:12:00
08:24:00
09:36:00
10:48:00
Laikas
Paprastų matricų daugyba
Daugyba pagal eilutes

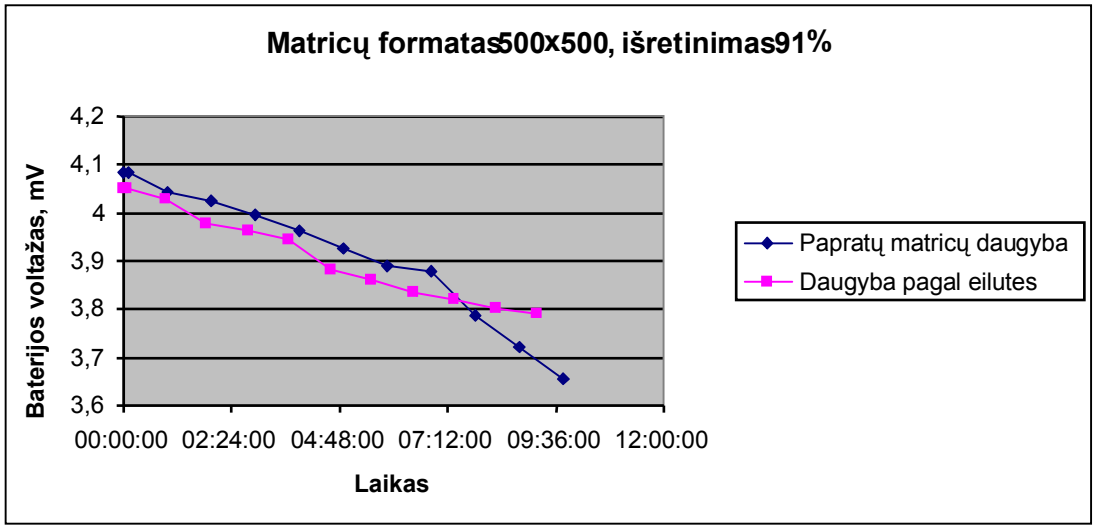
a)


Matricų formatas 500x500, išretinimas 91%
Baterijos voltažas, mV
4,15
4,1
4,05
4
3,95
3,9
3,85
3,8
3,75
3,7
3,65
3,6
0
20
40
60
80
100
120
Baterijos įkrautumas %
Paprastų matricų daugyba
Daugyba pagal eilutes

b)



c)



d)

18 paveikslas. Rezultatų palyginimas, dauginant matricas skirtingais dauginimo tipais, kai matricų formatas yra 500x500 ir išretinimas 91`%: a) Iteracijų ir laiko b) Baterijos voltažo ir baterijos įkrautumo % c) Baterijos įkrautumo % ir laiko d) Baterijos voltažo ir laiko

Kaip matome iš 15 paveikslo grafikų, naudojant 91% išretinimą daugybos laikai nedaug skiriasi tarp dauginimo tipų. Iš visų gautų grafikų mes matome, kad daugybos algoritmas dauginat matricas pagal eilutes yra efektyvus lyginant su paprastų matricų daugyba. Tačiau didėjant matricų formatams ir mažėjant matricų išretinimui skirtumas tarp metodų mažėja: dauginimo laikai, baterijos energijos suvartojimas tampa vienodi. Jei daugybos

pagal eilutes algoritme padidintume sąlyginių sakinių, įdėtume papildomų skaitliukų, toks dauginimo būdas būtų neefektyvus paprastų matricų daugybos atžvilgiu. Dėl atmienties ir procesoriaus apkrautumo pailgėja dauginimo laikas ir baterija iškraunama labiau, norint pasiekti tą patį rezultatą. Taip pat matome, kad baterijos energijos suvartojimas priklauso nuo daugybos trukmės. Kuo ilgiau dauginama, tuo daugiau energijos suvartojama. Skirtingų dauginimo tipų naudojimas ir įteracijų skaičius neįtakojo baterijos energijos suvartojimo greičiui. Tai yra baterija su visais dauginimo būdais ir įvairaus formato bei išretinimo matricomis sėdo vienodai. Todėl mes galime daryti prielaidą, kad atmintis ir procesorius dirbo pilnai apkrauti su visomis matricomis. Neturėdami tiesioginių duomenų apie procesoriaus ir atminties darbą, bei dėl kitų programų (pvz. Operacinė sistema) naudojimosi procesoriumi ir atmintimi sukeliamų paklaidų, mes negalime nustatyti, kokią įtaką baterijos energijos suvartojimui turėjo mūsų ekperimento metu apkrautas procesoriu ir atmintis.

5. Išvados

- Atlikus eksperimentą ir gavus rezultatus, galime daryti išvada kad jis pavyko. Matematinių metodų pagalba, įrodyta optimizuoto algorimto efektyvumas.
- Nepavyko gauti procesoriaus ir atminties tiesioginio poveikio įrodymo energijos suvartojimui, naudojant ekperimentui sukurtą progamą, tačiau pagal gautus rezultatus galime daryti prielaidą, kad jie buvo pilnai apkrauti ir gavome bendrą energijos suvartojimą, įskaitant kitų programų naudojimasi procesoriumi ir atmintimi.
- Optimizuoto kodo programos vygdymas sumažina baterijos energijos vartojimą. Tai turėtų atkreipti dėmesį programų kūrėjai, sistemų inžinieriai. Daugelis kuriamų programų nėra pakankamai optimizuotos. Greitų sistemų kūrimo priemonės CASE programuotojai taip pat turėtų stengtis, kad jų sukurtas produktas ne tik išgautų vartotojo norimą sitemos funkcionalumą, bet ir kad sistema veiktų kuo greičiau ir stabiliau.
- Didėjant mobilių įrenginių paklausai, vis daugiau atsiranda mobilių informacinių sistemų. Todėl gerėja ir techninės galimybės mobilių įrenginių. Techniškai

tobulesnis mobilusis įrenginys skaičiavimus atliks greičiau, nei morališkai pasenęs. Tačiau energijos suvartojimo proporcijos išliks tokios pat. Be to techninės naujovės yra brangesnės, todėl reikia atkreipti dėmesį į pačios sistemos kaštus. Gal brangesnė įranga neduos tokios kokybės, kuri atsvertu sistemos sukūrimo kaštus. Todėl naudojima senesnė įranga su optimizuotu programos kodu, gali būti pranašesnė, nei brangesnė įranga.

6. Literatūra

1. Kayun Chantarasathaporn and Chonawat Srisa-an. Object-Oriented Programming Strategies in C# for Power Conscious System // International journal of computer science. ISSN 1306-4428. 2006, Volume 1, Number 1, p. 54
2. Sparse Matrix. Engineering and scientific subroutine library (ESSL) documentation. ESSL version 3, release 1.2 Guide and Reference. Prieiga per internetą: http://www.navo.hpc.mil/usersupport/IBM/ESSL/essl148.html
3. Lawrence S. Brakmo, Deborah A. Wallach, Marc A. Viredaz. qSleep: A Technique for Reducing Energy Consumption in Handheld Devices. // International Conference On Mobile Systems Applications And Services, Proceedings of the 2nd international conference on Mobile systems, applications, and services. [2004].
4. Claudio Scordino, Giuseppe Lipari. Using resource reservation techniques for power-aware scheduling. // International Conference On Embedded Software; Proceedings of the 4th ACM international conference on Embedded software. [2004]
5. Tajana Simunic, Luca Benin, Giovanni De Micheli. Energy-efficient design of battery-powered embedded systems. // International Symposium on Low Power Electronics and Design; Proceedings of the 1999 international symposium on Low power electronics and design. [1999]
6. Alvin R. Lebeck, Xiaobo Fan, Heng Zeng, Carla Ellis. Power aware page allocation. // ACM SIGOPS Operating Systems Review , ACM SIGARCH Computer Architecture News , Proceedings of the ninth international conference

on Architectural support for programming languages and operating systems ASPLOS-IX. [November 2000]

7. C. Brandolese, W. Fornaciari, L. Pomante, F. Salice, D. Sciuto. High-level power estimation: A multi-level strategy for software power estimation // Proceedings of the 13th international symposium on System synthesis. [September 2000]
8. Jason Flinn, M. Satyanarayanan. Managing battery lifetime with energy-aware adaption. // ACM transactions on computer systems (TOCS). ISNN 0734-2071. 2004, Volume 22, Issue 2, p. 137
9. J.Valančius, V. Štuikys. Delninuko energijos suvartojimo įvertinimas taikomosios programos lygmenyje. // *KTU IT konferencija 2007.*

Pocket PC Energy Consumption Using Sparse Matrix Storage by Rows Modeling

Summary

Low power consumption is a major constraint for battery-powered system like computer notebook or pocket PC, mobile phone. In the past, specialists usually designed both specific optimized equipments and codes to relief this concern. Doing like this could work for quite a long time, however, in this era, there is another significant restraint, the time to market. To be able to serve along the power constraint while can launch products in shorter production period, objectoriented programming (OOP) has stepped in to this field.

In work we create program, whose multiply sparse matrix. Multiplication are two types: one we use standart matrix multiplication, other use compresed matrix storage by rows multiplication. When execute program, we can track, how battery power are consumpt. When we use compresed matrix storage by rows multiplication, we eliminate zero elements and multiplication execute faster, then standart matrix multiplication. So baterry power comsumption are lower.

If your system are very important battery life time, then you must use optimized programm code. Optimized programm code use less battery power, then not optimized. Then your system can work much longer.

Priedai

Visi priedai yra sudėti į kompaktą. Kompaktas prisegtas prie ataskaitos galo.

1. Priedas. Vartotojo instrukcija

Vartotojo instrukcijoje nurodoma ko reikia norint pasileisti programą, kaip ją paruošti paleidimui.

2. Priedas. Eksperimento rezultatai

Visi eksperimento rezultatai yra surašyti į „Microsoft Excel failą“. Jame taip pat yra ir rezultatų diagramos

3. Priedas. Programa

Šiame priede yra įdėta programa, kurią, naudojant istrukciją galima pasileisti. Taip pat galima pažiūrėti ir pogramos kodą.