ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Remigijus Valčiukas

Informatikos specialybės magistrantūros II kurso dieninio skyriaus studentas

Internetinė matematinio programavimo ir modeliavimo sistema. Sistemos kūrimas ir testavimas Online Mathematical Programming and Simulation System. The Implementation and Testing the System

Magistro darbas

Darbo vadovas:

Doc. K. Žilinskas

Recenzentas:

Doc. R. Laurutis

Šiauliai, 2012

Turinys

Įvadas

Įvairiose žmonių veiklos srityse yra aktualus uždavinių sprendimas. Inžinieriai, projektuojantys naujas technologijas, siekia maksimalaus efektyvumo. Verslininkai ir vadybininkai stengiasi darbą organizuoti taip, kad būtų gautas maksimalus pelnas [14]. Įvairūs gamybos planavimo ir ekonominiai uždaviniai yra susiję su riboto kiekio išteklių (žaliavų, darbo jėgos, energijos, kuro ir kt.) paskirstymu. Turimus išteklius dažnai galima paskirstyti ir panaudoti ne vienu, o keliais būdais. Todėl kyla uždavinys – rasti geriausia išteklių paskirstymo planą (sprendinį), kuris duotų didžiausią ekonominę naudą. Sudėtingais atvejais stengiamasi uždavinį suformuluoti tiksliai ir jį spręsti tam skirtais metodais bei priemonėmis[23, 24]. Tokie uždaviniai programuojami ir skaičiuojami matematinėmis programavimo ir modeliavimo sistemomis, kuriuose yra realizuoti įvairūs matematiniai metodai padedantys juos spręsti. Matematiniams modeliams ir jų realizacijoms sukurti reikia skirti daug resursų: laiko, žmonių darbo ir lėšų. Tačiau dažnai atsitinka taip, kad sukurtais sprendimais pasinaudojama tik trumpą laiką ir jie užmirštami. Visos lėšos skirtos rasti tiems sprendimams neduoda naudos. O galėtų būti ir kitaip, jie galėtų būti panaudojami kitų asmenų arba tobulinami ir pritaikomi kitur. Taip turimus resursus išnaudotume kur kas efektyviau.

Vikinomika arba **vikiekonomika (**angl. *Wikinomics*) - spietinė veiklos forma, greitai arba momentiškai apjungianti žmones saviorganizacijos būdu problemoms spręsti panaudojant turimus išteklius. Vienas pirmųjų vikinomikos sąvoką pradėjo naudoti bei vikinomikos koncepciją pagrindė *Don Tapscott* ir *Anthony D. Williams*, 2006 m. išleidę knygą „Vikinomika - kaip masinis bendradarbiavimas viską keičia“ (angl. *Wikinomics – How Mass Collaboration Changes Everything*)[15, 16]. Ši idėja yra sėkmingai naudojama internetinėje svetainėje „*wikipedia.com“*. Joje vartotojai gali dalintis ir tikslinti informaciją, faktus apie įvairius dalykus.

Šio darbo tikslas – pasinaudojus vikinomikos idėja, sukurti matematinio programavimo ir modeliavimo sistemą (veikiančią internetinės svetainės pagrindu), kurioje akademinės visuomenės atstovai bei kiti suinteresuoti asmenys galėtų aprašyti, redaguoti ir realizuoti (suprogramuoti) matematinius modelius, algoritmus, atlikti jų paiešką ir vykdyti realizuotas programas.

Darbo uždaviniai:

1. Išanalizuoti matematinio programavimo ir modeliavimo sistemas.
2. Suprojektuoti internetinę matematinio programavimo ir modeliavimo sistemą.
3. Suprojektuoti internetinės matematinio programavimo ir modeliavimo sistemos duomenų bazę.
4. Realizuoti internetinę matematinio programavimo ir modeliavimo sistemą.
5. Ištestuoti internetinę matematinio programavimo ir modeliavimo sistemą.
6. Palyginti internetinę matematinio programavimo ir modeliavimo sistemą su panašiomis sistemomis.

1. Analitinė dalis

1.1 Matematinis programavimas ir modeliavimas

Matematinis modeliavimas, anot Leono Valkūno, yra tam tikro reiškinio (ekonominio, fizikinio ar kito) teorijos kūrimas [1]. Kiti šaltiniai teigia, kad matematinis modeliavimas – tai taikomosios matematikos dalis, skirta įvairių sričių uždavinių sprendimui, naudojant eksperimento metodiką. Ji remiasi matematinių modelių sudarymu, jų pirmine analize, skaitinių algoritmų sudarymu ir analize, natūrinių stebėjimų ir eksperimentinių rezultatų apdorojimu bei naujos informacijos apie modeliuojamų procesų, sistemų gavimu ir analize [2]. Įmonės vadovui ar ekonomistui darbe dažnai tenka priimti sprendimus, kaip panaudoti turimus išteklius, kad gautas pelnas būtų didžiausias. Tokie uždaviniai dažnai susiję su išteklių paskirstymu ir turi daug sprendinių. Kiekvienas iš jų turi tenkinti tam tikras sąlygas bei apribojimus. Uždavinys, kuriame tarp daugybės leistinųjų sprendinių reikia rasti optimalų sprendinį (geriausią iš galimų variantų), vadinamas optimizavimo uždaviniu [24].

Tokio tipo uždaviniai sėkmingai sprendžiami tik matematiniais metodais, taikant matematinį programavimą. Norint nustatyti optimalų sprendinį taikant matematinio programavimo metodus, reikia uždavinį aprašyti matematinėmis išraiškomis, t. y. sudaryti optimizavimo uždavinio matematinį modelį. Matematinis modelis – tai sistema matematinių išraiškų (priklausomybių), aprašančių pagrindines modeliuojamo objekto savybes, rodiklius ir ryšius tarp jų. Optimizavimo uždavinio matematiniame modelyje yra tikslo funkcija *f(x)*,

išreiškianti pasirinktą optimalumo kriterijų, taip pat priklausomybės, aprašančios specifines sąlygas. Šias sąlygas turi tenkinti nagrinėjamo uždavinio sprendinys. Paprasčiausiu uždaviniu matematiniai modeliai turi tokią apibendrintą formą:

1. Rasti max *f(x)*, kai

$$\begin{aligned} &g_i(x) = b_i,\ i = 1, 2, \ldots, m_1; \\ &g_i(x) \le b_i,\ i = m_1 + 1, m_1 + 2, \ldots, m\ ; \\ &x \ge 0; \end{aligned} \qquad (1)$$

2. Rasti min f(x), kai

$$\begin{aligned} &g_i(x) = b_i,\ i = 1, 2, \ldots, m_1; \\ &g_i(x) \ge b_i,\ i = m_1 + 1, m_1 + 2, \ldots, m\ ; \\ &x \ge 0; \end{aligned} \qquad (2)$$

Tokio tipo ekstremumo uždaviniai, kai siekiama nustatyti tikslo funkcijos ekstremalią reikšmę, esant apribojimams, vadinami matematinio programavimo uždaviniais. Apribojimų sistema (1) arba (2) aprašo optimizavimo uždavinio leistinųjų sprendinių sritį. Kiekvienas šios srities taškas yra vienas iš leistinųjų sprendinių. Optimizavimo uždavinių esant ribojimams sprendimo teorija ir metodai vadinami matematiniu programavimu [14, 23, 24].

Šiuolaikinio matematinio modeliavimo metodologijos esmė yra tiriamo realaus objekto (proceso, reiškinio, sistemos) pakeitimas – matematiniu modeliu, o vėliau –matematinio modelio kompiuterine realizacija. Darbas ne su pačiu objektu, o su jo modeliu leidžia be didelių išlaidų ir pakankamai greitai atlikti jo savybių ir elgesio tyrimą įvairiausiose įmanomose situacijose. Tuo pat metu skaitiniai eksperimentai su objektų modeliais panaudojant kompiuterius leidžia detaliai ir pakankamai pilnai ištirti objektus, ką ne visada leidžia grynai teoriniai metodai [2, 3, 17]. Galime nagrinėti labai sudėtingus matematinius modelius. Jų sudarymas ir naudojimas yra kertinė viso kompiuterinio modeliavimo dalis. Šiuo metu įvairiose matematinio programavimo ir modeliavimo sistemose jau sukurta daug metodų, leidžiančių aprašyti diferencialinėmis, integralinėmis ir kitokiomis lygtimis daugelį mus dominančių procesų – gamtos, technologijų ar socialinių mokslų [17]. Tačiau daugumos tokių metodų realizacijos yra ne atviro kodo ir jų tobulinimo ar panaudojimo galimybės visiems norintiems yra ne visada prieinamos. Toliau analizuosime, kokias galimybes mums suteikia šiuo metu esančios komercinės ir nemokamos matematinio programavimo ir modeliavimo sistemos.

1.1.1 Programinės įrangos *Mathematica* analizė

Mathematica yra daugiaplatformė komercinė programinė įranga, veikianti *Linux, Apple Macintosh, MS-DOS, NeXT, OS/2, Unix, VMS* ir *Windows* operacinėse sistemose. Ji skirta įvairiems matematiniams uždaviniams programuoti bei skaičiavimams atlikti. Ją sudaro dvi dalys: branduolys ir vartotojo sąsaja. Branduolys yra atsakingas už *Mathematica* programavimo kalbos interpretavimą ir rezultatų grąžinimą, o vartotojo sąsaja – už vizualinį programos apipavidalinimą (teksto formatavimą, grafikų ir lentelių rodymą) bei kodo derinimą. Duomenų perdavimui tarp branduolio ir vartotojo sąsajos naudojamas *MathLink* protokolas. Šis protokolas taip pat leidžia prijungti kitas *Mathematica* sukurtas programas ir jas panaudoti. Toliau panagrinėsime *Mathematica* branduolio funkcionalumą:

1. Gali interpretuoti *Mathematica* programavimo kalbą;
2. Iškviesti kitas programas, kurios turi komandinės eilutės sąsają;
3. Atidaryti failą arba internetinį puslapį per numatytąsias programas;
4. Išsiųsti elektroninį laišką;
5. Įkrauti dinamiškai į branduolį funkcijas iš failo;

Mathematica taip pat turi sąsajas su *.NET(.NET/Link*), *JAVA(JAVA/Link*) ir *C/C++* programavimo kalbomis. Programos parašytos išvardintomis kalbomis gali iškviesti *Mathematica* branduolio metodus. Taip pat galimas ir atvirkščias variantas.

Šiame įrankyje yra įvairiausių priemonių ir matematinių funkcijų bibliotekų, kurios padeda spręsti matematinius uždavinius:

- Elementarių matematinių funkcijų biblioteką;
- Specialių matematinių funkcijų biblioteką;
- Turi matricų ir duomenų manipuliavimo įrankius, įskaitant ir paprastus masyvus;
- Palaikomi kompleksiniai skaičiai, intervalinė aritmetika ir simboliniai palyginimai;
- Dvimatės ir trimatės grafikos įrankiai;
- Lygčių sistemų bei diferencialinių lygčių sprendimai;
- Priemonės sukurti vartotojo sąsajai;
- Priemonės vaizdo ir morfologinio vaizdo apdorojimui bei atpažinimui;
- Priemonės braižyti grafikus;
- Priemonės kombinatorikos uždaviniams;

- Priemonės teksto analizei;
- Įrankiai duomenų gavybos analizei;
- Skaičių teorijos funkcijų biblioteka;
- Įrankiai finansiniams skaičiavimams;
- Bibliotekos garso bangų, paveiksliukų ir duomenų analizei;
- Skaičių integralinės transformacijos;

Mathetmatica branduolys taip pat gali vykdyti skaičiavimus lygiagrečiai, tačiau ši galimybė yra papildomai licencijuojama ir tai vadinama *Grid Mathematica*. Paprastai branduolys visus veiksmus, parašytus jos kalba, atlieka paeiliui, t. y. nuosekliai. Su *Grid Mathematica* kiekvienas lygiagretus procesas yra atliekamas atskiro procesoriaus arba branduolio. Tačiau yra apribojimų. Priklausomai nuo licencijos tipo gali būti naudojama nuo keturių iki šešiolikos branduolių. Tiek branduolių nebūtinai turi turėti vienas kompiuteris. Yra leidžiama panaudoti kitus kompiuterius, kurie yra tinkle, t. y leidžiami paskirstytieji skaičiavimai. Tokiu atveju jau yra reikalingas serveris, kuris kontroliuotų, valdytų procesus, juos statytų į eilę ir dalintųsi virtualia atmintimi. Tinklinis bendravimas vyksta *TCP/IP* protokolu, naudojantis *SSH* arba *RSH* sertifikatais. Tai suteikia saugumą perduodant duomenis.

Mathematica funkcionalumą galima perkelti ir į internetinę svetainę. Toks produktas vadinamas *Web Mathematica.* Ši sistema remiasi *JAVA* technologijomis, tai yra naudojami *Java Applet* ir *Java Servlet* komponentai. *Java Applet* kartu su *JavaScript* programavimo kalbos scenarijais yra klientinė programos dalis, kuri integruojama į internetinį puslapį. Tuo tarpu *Java Servlet* komponentas turi turėti sąsają su *Java Applet* moduliu ir *Mathematica* branduoliu. Ši dalis yra atsakinga už modulio funkcionalumą. Skaičiavimai tokioje sistemoje atliekami asinchroniniu būdų. Tai reiškia, kad visi skaičiavimai atliekami serveryje, o rezultatas gražinamas tik tada, kai skaičiavimai yra baigti. Tačiau vartotojas gali matyti programinių modulių klaidas ir kitus pranešimus. Taip pat administratorius gali matyti informaciją apie vartotojus ir sistemos darbą. Tam, kad galima būtų naudoti šią sistemą, reikalingas *Apache HTTP* arba *Microsoft IIS* serveris. *Java* versija turi būti *5.0* arba aukštesnė [4, 5, 25].

Šis matematinio programavimo įrankis turi daug privalumų: integraciją su kitomis programavimo kalbomis, matematinių funkcijų bibliotekas, skirtas dirbti su įvairias matematiniais uždaviniais, priemones atlikti skaičiavimus lygiagrečiai (nors ir su tam tikrais ribojimais) ir integraciją su internetiniais puslapiais. Tačiau ši sistema nerealizuoja to, ką esame numatę pasiekti šiame darbe. Visų pirma tai komercinis įrankis ir dėl to nėra visiem prieinamas

įrankis. Antra, nors ir yra realizuotas sprendimas *Web Mathematica*, kuriuo galime perkelti *Mathematica* funkcionalumą į internetinius puslapius, tačiau viskas kuriama statiškai. Vartotojas negali internetinėje svetainėje dinamiškai sukurti modulio, jo redaguoti. Tai turi padaryti serverio administratorius, kuriame yra šis įrankis. Taip yra dėl to, jog modulio klientinė ir serverinė dalis reikalauja *Java* programavimo kalbos kodo kompiliavimo, kurį reikia dar susieti su *Mathematica* branduoliu. Taigi, vikinomikos idėjos šiame įrankyje panaudoti negalime.

1.1.2 Programinės įrangos *Scilab Scientific* analizė

Scilab Scientific - nemokamas įrankis skirtas matematiniams uždaviniams programuoti bei skaičiavimams atlikti. Šis įrankis suteikia grafinę ir komandinės eilutės vartotojo sąsajas. Šios vartotojo sąsajos skiriasi tik tuo, jog grafinėje vartotojo sąsajoje per meniu juostą galime išsikviesti pagalbą, peržiūrėti rašytų komandų istoriją, pakeisti teksto spalvą, šriftą ir atlikti kitus veiksmus su tekstu. Reikia pažymėti kad šis įrankis veikia daugelyje operacinių sistemų – *Linux, MacOSX, Windows*. Turi šimtus matematinių funkcijų, aukšto lygio programavimo kalbą, kuria galima suprogramuoti uždavinį. Yra orientuotas į:

- Tiesinės algebros uždavinių sprendimą;
- Matricų sprendimų radimą;
- Polinomų skaičiavimus;
- Diferencialinių lygčių sprendinių ieškojimą;
- Interpoliacijos apskaičiavimą;
- Statistikos uždavinių sprendimą;
- Grafikų braižymą;
- Tinklinių sistemų uždavinių sprendimą.

Šiuo įrankiu galima generuoti dvimatę ir trimatę grafiką. Dvimatėje grafikoje galima atvaizduoti linijas, histogramas bei skritulinias diagramas. Trimatė grafika suteikia galimybę atvaizduoti įvairius paviršius. Visa tai galima išsaugoti šiais kompiuterinių failų formatais: *GIF, BMP, JPEG, SVG, PDF*. Dar viena šio įrankio svarbi savybė – sąsaja su *C/C++, Java* programavimo kalbomis. Vartotojas gali iškviesti vykdyti paprogrames, parašytas *Scilab* programavimo kalba iš *C/C++, Java* programavimo kalba parašytų programų. Skirtingai negu prieš tai nagrinėta matematinio programavimo sistema *Mathematica*, ji neturi sąsajos su internetinio pobūdžio

sistemomis. Taip pat *Scilab* neturi priemonių vykdyti programas lygiagrečiai. Akivaizdu, kad šis įrankis funkcionalumu nusileidžia prieš tai nagrinėtam įrankiui *Mathematica*, tačiau yra nemokamas[6,7,18,27].

1.2 Matematinių funkcijų bibliotekos

Tokios matematinio programavimo sistemos, kaip *Mathematica* ir *Scilab Scientific* turi savo programavimo kalbas, kurios yra labiau orientuotos į matematinių uždavinių programavimą ir skaičiavimą. Tai yra aukšto lygio programavimo kalbos, kurių sintaksė yra truputį supaprastinta, lyginant su *C/C++, Java ar Fortran* programavimo kalbomis. Šios programavimo kalbos turi daug didesnes galimybes negu ankščiau analizuotos, nes nėra orientuotos vien tik į matematinį programavimą. Taip pat operacijos su jomis atliekamos daug greičiau, nes jos nėra interpretuojamos programavimo kalbos. Tačiau jos nėra tokios patogios programuoti matematinio programavimo uždavinius, nes visas matematines funkcijas (pvz. matricų daugybą ir panašias) reikia pasirašyti pačiam. Bet yra sprendimas – naudoti matematinių funkcijų bibliotekas, skirtas toms programavimo kalboms. Jos gali būti atviro kodo arba pateikti tik funkcijų sąsajas. Atviro kodo bibliotekose galima pamatyti kaip yra realizuotos matematinės funkcijos, netgi jas pataisyti ar patobulinti. Toliau panagrinėsime jau sukompiliuotas ir atviro kodo matematinių funkcijų bibliotekas [8, 9, 12, 13].

1.2.1 Matematinių funkcijų biblioteka *ESSL*

ESSL (angl. *The Engineering Scientific Subroutine Library*) yra specialių matematinių funkcijų ir paprogramių biblioteka. Ji gali būti naudojama tiek kuriant naujas programas tiek naudojant ją su esamomis programis[8]. Dideliam skaičiavimų efektyvumui gauti galima naudoti *PESSL* biblioteką. Ji skirta lygiagretiems skaičiavimams atlikti[10]. Biblioteka yra daugiaplatformė, tai yra tinka įvairioms operacinėms sistemoms (*Windows*, *Linux*). Tai suteikia tam tikrą lankstumą programas perkeliant į kitas operacines sistemas. Be to, nereikia perrašinėti visos programos, jeigu norima ką nors joje pakeisti, nes galime iškviest bibliotekos paprogrames, o ne aprašinėti metodus programoje. Taip sutaupoma laiko.

Bibliotekoje galima rasti plataus spektro matematinių funkcijų, skirtų mokslinėms ir gamybinėms programoms:

- Operacijoms su matricomis;
- Tiesinės algebros skaičiavimams;
- Furjė transformacijoms atlikti;
- Duomenų rikiavimo ir paieškos algoritmams;
- Interpoliacijoms;
- Atsitiktinių skaičių generavimui;

Ji apjungia kelias *IBM* sukurtas bibliotekas *LAPACK* (angl. *Linear Algebra Package)*, *BLAS* (angl. *Basic Linear Algebra Subprograms*), bet ne pilnai jas perdengia. Visos jos yra tiesinės algebros matematinių funkcijų bibliotekos, parašytos *Fortran* programavimo kalba[9]. *IBM*, kurdami *ESSL*, laikėsi *Fortran* programavimo kalbos standartų, todėl norint iškviesti bibliotekos paprogrames, reikia jas iškviesti tokioje funkcinėje aplinkoje, kurioje gali būti iškviečiamos įprastos *Fortran* programos. Jeigu paprogramės yra parašytos ne *Fortran* programavimo kalba, o *C/C++*, tai tada, reikia panaudoti tokį *Fortran* mechanizmą, kuris suvienodintų duomenis bei galėtų juos susieti. Pavyzdžiui, norint panaudoti masyvo išrikiavimo algoritmu, iš pradžių jis turi būti suvienodintas pagal *Fortran* programavimo kalbos sintaksę. Tai, žinoma, truputį apsunkina šios bibliotekos panaudojimo galimybes įvairiose programavimo kalbose [8, 9, 10].

1.2.2 Matematinių funkcijų biblioteka *Numerical recipes*

Numerical recipes - tai elektroninė ir popierinė knygos versija apie matematinius uždavinius ir jų sprendimo būdus. Taip pat kiekvienam uždaviniui yra sukurtas programinis kodas, kurį sukompiliavus ir paleidus galime spręsti uždavinius kompiuteriu. Sukompiliuoti kodą autoriai rekomenduoja su *Microsoft visual studio*/*Microsoft visual C++* arba *GNU C++* programavimo aplinkomis. Šios bibliotekos autoriai naudoja *C/C++* programavimo kalbą, o tiksliau *C* programavimo kalbos šeimos sintaksę bei objektus, kurie yra *C++* programavimo kalboje. Jų teigimu, šis produktas nėra idealiausias programavimo pavyzdys (norima pabrėžti, kad funkcijos nėra suprogramuotos labai efektyviai), o ir jame vartotojai randa nemažai klaidų. Tačiau nepaisant to, dauguma pavyzdžių yra veikiantys. Taip pat yra parašytas ir paaiškintas visas uždavinio sprendimo algoritmas, sprendimo būdas ir formulės jam spręsti [11]. Tai nėra

programų biblioteka, kaip *Visual Numerics* ar *NAG*, nėra iškarto paruoštos programavimo sistemos kaip *Mathematica*, *MATLAB* ir *Maple*. Šio projekto tikslas išmokyti spręsti matematinius uždavinius ir juos realizuoti. Tačiau tai nereiškia, kad tų matematinių funkcijų negalima naudoti.

Per ilgą laiką *Numerical recipes* kūrėjai sukaupė labai daug uždavinių su jų sprendimų algoritmais, paaiškinimais ir jų programiniais kodais. Tačiau jų naudojimą riboja licencijos. Yra dvi licencijos: viena riboto naudojimo nemokama licencija, o kita mokama, skirta tik asmeniniam naudojimui. Nemokamoje licencijoje galite naudoti iki 10 kodo gabaliukų asmeninėm reikmėm, o mokamoje licencijoje, negalite kodo perduoti kitam asmeniui [11].

1.2.3 Matematinių funkcijų biblioteka *Netlib Repository LAPACK*

Netlib Repository LAPACK (angl. Linear Algebra Package) yra nemokama specialių matematinių funkcijų biblioteka. Šios bibliotekos metodai naudoja *BLAS* (angl. *Basic Linear Algebra Subprograms*) bibliotekos metodus, todėl norėdami pasinaudoti *LAPACK* biblioteka, turime sukompiliuoti ir *BLAS*. Jos funkcijų pagalba galima spręsti tiesines lygčių sistemas, tikrinių reikšmių uždavinius bei taikyti mažiausių kvadratų metodą. Taip pat joje yra matricų faktorizavimo funkcijos (*LU, QR, Cholesky, Schur* matricų skaidymas). Ši funkcijų biblioteka buvo parašyta *Fortran 77* programavimo kalba, o dabar yra perrašyta su *Fortran 90*. Taigi ji skirta *Fortran* programavimo kalbai, tačiau ją galima naudoti ir kitose programavimo kalbose, pavyzdžiui *C/C++*. Yra du būdai tai padaryti. Pirmas būdas, reikia kviesti *Fortran* kalba parašytas funkcijas su C programavimo kalbai skirta sąsaja. Norint sukompiliuoti ir vykdyti tokias programas, reikės ne vien tik *C* programavimo kalbos kompiliatoriaus, bet ir *Fortran* programavimo kalbos kompiliatoriaus. Kitas būdas panaudoti *LAPACK* bibliotekos funkcijas *C/C++* programavimo kalba parašytose programose, naudoti *CLAPACK* biblioteką, kuri yra *LAPACK* bibliotekos kopija tik perrašyta į *C* programavimo kalbą. Jeigu norime parašyti *C++* programą ir panaudoti šios bibliotekos funkcijas, pirmiausia mums reikia deklaruoti tų funkcijų prototipą[29,30,31]. Ji aprašoma taip:

extern "C" {//CLAPACK funkcijų aprašas};

Visi metodų pavadinimai šioje bibliotekoje sudaryti pagal schemą **XYYZZZ**. Pirmoji raidė **X** šioje schemoje reiškia duomenų tipą:

- S – realieji skaičiai;
- D – dvigubo tikslumo slankaus kablelio skaičiai;

- C – kompleksiniai skaičiai;
- Z – kompleksiniai 32 baitų ilgio skaičiai.
- DS – metodas problemai spręsti naudoja viengubo tikslumo slankaus kablelio skaičius, bet jo duomenų tipas yra dvigubo tikslumo slankaus kablelio skaičius.
- ZC – metodas problemai spręsti naudoja kompleksinius skaičius, bet jo duomenų tipas yra kompleksinis 32 baitų ilgio skaičius.

Kitos dvi raidės **YY** nurodo matricos tipą, o raidės **ZZZ** atliekamus skaičiavimus. Jų paaiškinimus rasite priedų penktame skyrelyje pavadinimu *Netlib Repository LAPACK*.

1.2.4 Matematinių funkcijų biblioteka *IMSL Numerical Libraries*

IMSL Numerical Libraries yra komercinė matematinių ir statistinių funkcijų biblioteka, kuri skirta *Java, C/C++, Fortran, C#, Python* programavimo kalboms. Biblioteka yra palaikoma 32 ir 64 bitų operacinėse sistemose (*Linux, Unix, Windows*). Ją kompiliuoti galima su *Intel, Microsoft, IBM* firmų sukurtais programavimo kalbų kompiliatoriais. Tipinės šios bibliotekos panaudojimo sritys yra :

- Finansinių paslaugų darbo optimizavimas.
- Finansinių paslaugų rizikos valdymo optimizavimas.
- Inventoriaus valdymo ir poreikių prognozavimas.
- Modeliavimo ir didelio našumo skaičiavimas.
- Kompiuterinės biologijos analizavimas ir modeliavimas.
- Programos, kuriuose atliekami matematiniai skaičiavimai.

Funkcijas galime išskaidyti į dvi dalis (žr. Lentelė 1 ir Lentelė 2):

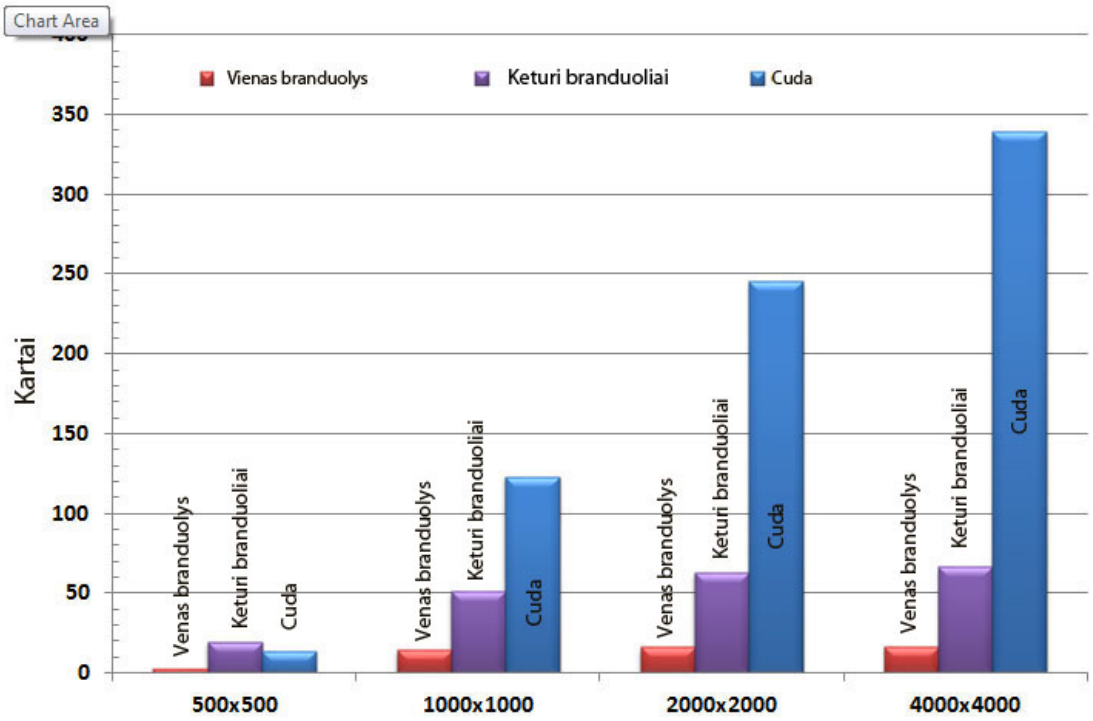
Lentelė 1. Matematinių funkcijų sritys.

• Operacijos darbui su matricomis • Tiesinės sistemos • Interpoliacija ir apytiksliai skaičiai • Integralai ir diferencialai • Diferencialinės lygtys	• Transformacijos • Netiesinės lygtys • Optimizacija • Specializuotos funkcijos • Kitos funkcijos

Lentelė 2. Statistinių funkcijų sritys.

• Paprastoji statistika • Regresija • Koleriacija ir kovariacija • Dispersinė analizė ir vizualūs eksperimentai • Kategorizuota ir diskretiška duomenų analizė • Neparametrizuota statistika • Neuroniniai tinklai • Genetiniai algoritmai	• Tinkamumo analizė • Laiko eilučių ir prognozių • Invariantinė analizė • Išgyvenimo analizė • Tikimybių funkcijų paskirstymas ir inversija • Atsitiktinių skaičių generavimas

Šioje bibliotekoje sukaupta daugiau negu tūkstantis algoritmų skirtų spręsti įvairias iškilusias problemas ir uždavinius. Funkcijų pavadinimai sukurti taip, kad vartotojas galėtų intuityviai suprasti, koks algoritmas yra taikomas. Taip pat yra pilna bibliotekos dokumentacija. Viena iš neseniai pristatytų *IMSL* bibliotekos naujovių – grafinių procesorių panaudojimas tiesinėms algebros paprogramėms vykdyti. Vartotojai gali sujunkti *IMSL* esančias funkcijas su *NVIDIA* grafikos procesoriuose naudojama *CUDA BLAS* (angl. *Basic Linear Algebra Subprograms*) biblioteka 4.0. *Cuda* – tai platforma, skirta lygiagretiems skaičiavimams atlikti *NVIDIA* grafikos procesoriuose. Kompiuterio procesorius, dirbdamas su tokiu grafikos procesoriumi, gali ženkliai sumažinti tokių funkcijų vykdymo laiką (žr. Pav.1.).



Pav. 1. Matricų daugyba su dvigubu slankiuoju kableliu

Pirmajame paveikslėlyje esanti stulpelinė diagrama parodo, kiek kartų greičiau yra vykdoma matricų daugyba su dvigubu slankiuoju kableliu, kurių dydis yra nuo 500 iki 4000, palyginant su vieno branduolio procesoriumi. Kaip galime matyti iš grafiko, *C* kalba parašytų paprogramių perkėlimas iš pagrindinio kompiuterio procesoriaus (vieno ir keturių branduolių) į grafikos procesorių, pagreitina jų vykdymą daugiau negu 100 kartų. Galime padaryti prielaidą – kuo daugiau matematinių veiksmų programoje darysime su grafikos procesoriumi, tuo greičiau programa atliks skaičiavimus ir gražins rezultatą [12, 26].

Apibendrinus vidsas nagrinėtas matematinių funkcijų bibliotekas, galima daryti prielaidą, jog matematiniam programavimui puikiai tinka ir kitos programavimo kalbos, kurios nėra skirtos tik matematiniam programavimui. Panaudojus matematinių funkcijų bibliotekas, jomis galima programuoti taip pat efektyviai kaip ir su matematinio programavimo sistemomis.

2. Projektinė dalis

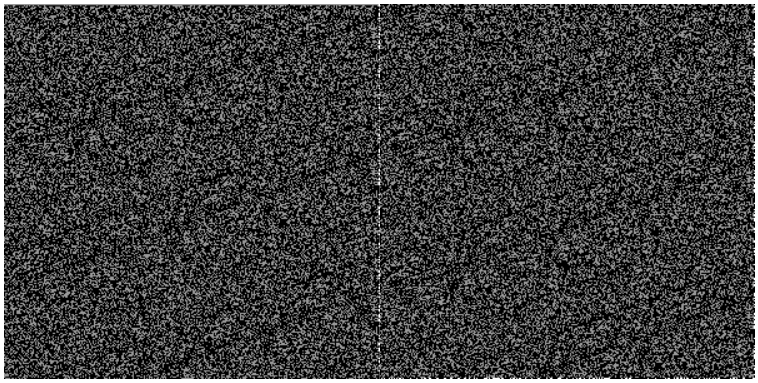
2.1.Įrankių ir priemonių pasirinkimo analizė

Norint sukurti internetinę matematinio programavimo ir modeliavimo sistemą vikinomikos principu, visų pirmą mums reikia pasirinkti programavimo kalbą, kuria vartotojas realizuos skaitinius algoritmus. Šių programų vykdymas gali būti atliekamas tiek serveryje, tiek ir vartotojo kompiuteryje, tačiau daugiau privalumų turi pirmasis variantas. Pirmiausia dėl to, jog vartotojui nebūtina visa laiką būti prisijungusiam prie sistemos ar, kad jo kompiuteris būtų įjungtas. Programa ir jos atliekami skaičiavimai būtų vykdomi serveryje, o gautas rezultatas grąžintas asinchroniniu būdu, kai vartotojas prisijunks prie sistemos. *Web Mathematica* matematinio programavimo ir modeliavimo sistema irgi naudoja tokį rezultato gražinimo būdą. Taip pat, serveryje programos vykdymas gali būti kur kas efektyvesnis, dėl to, kad jo skaičiavimo pajėgumai kur kas didesni, negu vartotojo kompiuterio.

Labai svarbi internetinės matematinio programavimo ir modeliavimo sistemos dalis yra vartotojo sąsaja. Ji turi būti bendra visiems ir savaime atsinaujinanti atsiradus kokiems nors pakeitimams, nes prie šios sistemos ir prie konkretaus matematinio modelio gali dirbti ne vienas vartotojas. Tai realizuoti būtų patogu per internetinę svetainę. Vartotojui tokiu atveju nereiktų instaliuoti jokių papildomų programų, o tik turėti interneto ryšį ir naršyklę.

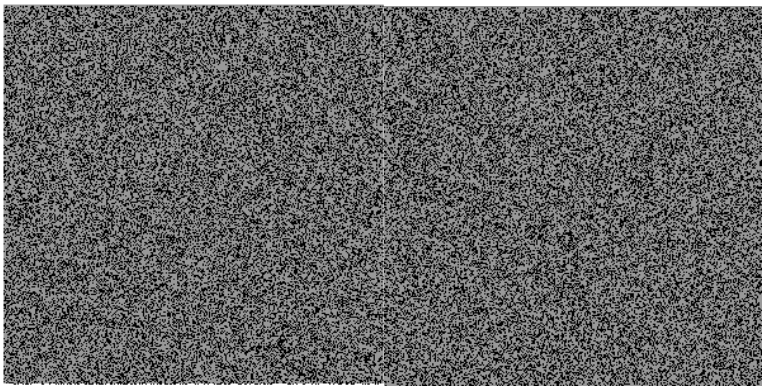
Programavimo kalba, su kuria vartotojas rašys programas, gali būti interpretuojama arba kompiliuojama. Pirmajame skyrelyje mes nagrinėjome matematinio programavimo sistemas, kurios interpretuoja programinį kodą ir kompiliuojamas programavimo kalbas su matematinių funkcijų bibliotekomis. Kadangi mūsų tikslas yra sukurti sistemą, kurioje dinamiškai būtų kuriamos ir vykdomos programos per internetinę svetainę, mums labiau tiktų interpretuojama programavimo kalba. Tokiu atveju vartotojui nereiktų laukti, kol programa bus sukompiliuota, o serveryje nereiktų laikyti programų vykdomųjų failų. Saugumas taip pat būtų daug didesnis, nes interpretuojamos programos orientuotos tik į tam tikrą sritį. Jomis sunkiau realizuoti kenksmingą kodą. Tačiau parašyti interpretavimo kalbą ir tos kalbos interpretatorių būtų labai sunku, o ir vartotojams nebūtų patogu mokytis dar vienos programavimo kalbos. Todėl reiktų ieškoti jau sukurtų programavimo kalbų ir jų interpretatorių. *Web Mathematica* sprendimu pasinaudoti negalime, nes jis neleidžia dinamiškai kurti programos sąsajos internetinėje svetainėje su

Mathematica branduoliu. Be to *Mathematica* įrankis yra komercinis produktas ir kiekvienam jos vartotojui reiktų licencijos. Pačios *Mathematica* programavimo kalbos mes panaudoti negalime, nes interpretuoti šią kalbą gali tik *Mathematica* branduolys. *Scilab* programavimo sistemos panaudoti irgi negalime, nes ji neturi priemonių dirbti su internetinėmis technologijomis [18, 25, 27]. Jei negalime panaudoti šių programavimo kalbų ir įrankių, gal galėtume panaudoti programavimo kalbą, kuri nėra orientuota vien į matematinį programavimą. Viena iš labiausiai paplitusių tokių interpretuojamų programavimo kalbų yra *JavaScript*. Šią kalbą gali interpretuoti *Java, C#, C++* programavimo kalbos prijungus prie jų atitinkamas bibliotekas. Nors *JavaScript* programavimo kalba nėra orientuota į matematinį programavimą, o yra naudojama plačiau, tačiau ją gal būtų galima tam pritaikyti. Mums reikia patikrinti, ar su šia kalba mes galėsime realizuoti matematinius modelius ir skaitinius algoritmus. Vienas iš tokių vertinimo kriterijų, kuris labai svarbus matematiniame programavime, yra atsitiktinio dydžio generavimas. Atsitiktinis dydis generuojamas daugelyje programavimo kalbų, bet ar iš tiesų jis yra atsitiktinis. Tai galime pamatyti atlikę eksperimentą: jeigu apibrėžtoje aplinkoje, ciklišk ai generuosime atsitiktinius dydžius, kurie atitiktų tam tikrą poziciją toje aplinkoje, o tas pozicijas pažymėsime taškais, tai jų visuma sudarys paveikslėlį. Jeigu paveikslėlyje galime pamatyti, kažkokią tendenciją, raštą, tai šis programinis atsitiktinių dydžių generatorius matematiniam programavimui netinka. Žemiau esančiame paveikslėlyje (žr. pav. 2), matome sugeneruotus du paveiksliukus.



Pav 2. PHP *rand()* funkcijos vizualus atvaizdavimas. Pateikti keli bandymai.

Juos sudaro keturiasdešimt du tūkstančiai taškų, kurie buvo sugeneruoti panaudojus *PHP rand()* funkciją. Ši funkcija generuoja atsitiktinį skaičių. Sugeneruotose paveikslėliuose matyti raštas, todėl galime teigti, kad *PHP rand()* funkcija netenkina mūsų kriterijaus. Toks pat bandymas atliktas su *JavaScript* programavimo kalba (žr. pav. 3). Jame matyti jau kitoks vaizdas. Konkretaus rašto įžvelgti paveikslėliuose nebegalime. Todėl galime teigti, kad mūsų parinktas kriterijus yra tenkinamas.



Pav 3. Javascript *rand*() funkcijos vizualus atvaizdavimas. Pateikti keli bandymai

Kadangi pasirinkome *JavaScrip*“ programavimo kalbą, ją turime pritaikyti matematiniam programavimui. Internetinėje svetainėje turime sukurti galimybę vartotojui kurti matematinių funkcijų bibliotekas, kuriuose esančius metodus būtų galima panaudoti kuriamose programose.

Vartotojo sukurtų programų vykdymui atskiruose serveriuose reikia parašyti programą, kuri priimtų duomenis, siunčiamus iš internetinės svetainės ir interpretuotų *JavaScript* programavimo kalbą. Kaip rodo didelių projektų praktika, tokių kaip *google.com, facebook.com*, kurios naudoja internetines svetaines bei serverines programas atliekančias tam tikrus skaičiavimus, kitus darbus, geriausia naudoti *Java* arba *C++* programavimo kalbą[21]. Nors matematinius skaičiavimus atliks interpretuojama kalba, reikia įvertinti, kad *C++* programavimo kalba yra iki keliolikos kartų greitesnė[22]. Todėl darome prielaidą, kad programavimo kalbos interpretavimas irgi turėtų būti greitesnis. Be to galime panaudoti *QT* biblioteką, skirtą *C++* programavimo kalbai, kuri palaikoma įvairiose operacinėse sistemose ir turi *JavaScript* interpretatorių.

Vartotojo sąsajos kūrimui (internetinei svetainei) galime rinktis iš kelių programavimo kalbų: *PHP, ASP.NET, JAVA (Servlet, Applet, JSP). JAVA* ir *PHP* programavimo kalbos turi pranašumą prieš *ASP.NET*. Jos nėra pririštos prie konkrečios duomenų bazės ir operacinės sistemos. Nors *ASP.NET* parašyti internetiniai puslapiai turi truputį didesnį vykdymo greitį, tačiau jai būtina *Windows* operacinė sistema, *IIS* (angl. *Internet Information Services*) servisas ir

Microsoft SQL duomenų bazė. Tuo tarpu *Java* ir *PHP* šioje vietoje yra gerokai lankstesnės. Vis dėlto internetinei svetainei kurti pasirinkome *PHP* programavimo kalbą. Visų pirma dėl to, kad *Java* programų paleidimui (*Java Applet*) reikalinga *Java* virtuali mašina, kurią vartotojui reiktų papildomai įsidiegti. Antra, daugelis projektų, kurie reikalauja daug resursų, yra parašyti panaudojant *PHP, MySQL* ir *C/C++* programavimo kalbas. Kaip pavyzdžius galime pateikti *google.com*, *yahoo.com*, *facebook.com* ir w*ikipedia.com* internetines svetaines [21]. Interaktyvumui padidinti bus naudojama *JavaScript* programavimo kalba ir jos biblioteka *jQuery*. Šios priemonės palengvina manipuliacijas *DOM* (angl. *Document Object Model*) elementais. O interneto svetainėje aprašinėti skaitiniams algoritmams ir matematiniams modeliams naudosime *JavaScript* programavimo kalba parašytą įskiepį *ckeditor.* Tai teksto redaktorius su įvairiomis teksto formatavimo galimybėmis.

Visa sistema, kurią sudaro serverinė (programa interpretuojanti kliento duomenis) ir klientinė (internetinė svetainė) dalis, turi keistis duomenimis su duomenų baze. Čia vėlgi panašiuose sistemose (duomenų saugojimo ir apsikeitimo prasme), dažniausiai naudojama *MySQL* duomenų bazė. Ji yra nemokama ir mūsų pasirinkti įrankiai turi priemones su ja dirbti. Kaip alternatyvą galėtume naudoti *Oracle* duomenų bazę, tačiau nemokama jos versija naudoja ribotą resursų kiekį, kurių poreikiui atsiradus, padidinti negalėtume.

Programavimo įrankių pasirinkimą nulemia mūsų ankščiau priimti sprendimai. Kadangi naudojame *QT* biblioteką, turime naudoti ir pačių bibliotekos kūrėjų sukurtą programavimo aplinką *QT Creator*. Darbui su *PHP*, *HTML*, *JavaScript* programavimo kalbomis galime rinktis tarp dviejų įrankių: *Dreamweaver* ir *Expression Web*. Jos išsiskiria iš kitų programavimo aplinkų skirtų internetinių svetainių kūrimui tuo, kad vienu metu galime matyti ir kodą ir internetinio puslapio vaizdą. Taip pat galime tikrinti, ar parašytas *HTML* programavimo kalba kodas atitinka galiojančius standartus. Abu turi *CSS* ir *JavaScript* kalbų žinyną. Iš šių produktų pasirinkome „*Dreamweaver*“, nes įrankis ne tik vienu metu rodo kodą ir tinklapio vaizdą, bet jis gali veikti ir naršyklės rėžimu. Tai reiškia, kad internetiniame puslapyje galima atlikti įvarius veiksmus ir matyti, kaip realiu laiku keičiasi kodas, kas yra atliekama.

Internetinės matematinio programavimo ir modeliavimo sistemos projektavimui pasirinkome *MagicDraw UML* įrankį. Tai modeliavimo įrankis, kuris leidžia *UML (*angl. *Unified Modeling Language)* kalba aprašyti įvairius modelius. Taip pat ja lengva atvaizduoti duomenų bazės lenteles, jų tarpusavio ryšius, galima matyti pilną jos struktūrinį vaizdą. Šis įrankis turi priemones iš *UML* diagramų generuoti *C#, C++*, *Java* programinį kodą ir atvirkščiai. Alternatyvos šiam įrankiui yra *AgroUML* ir *DIA*. Tačiau *AgroUML* nepalaiko *UML 2* standarto ir

pastebėta, kad turi labai ribotą galimybę atstatyti padarytus pakeitimus, jeigu vartotojas padaro klaidą. O *Dia* turi labai ribotas priemones atvaizduoti duomenų bazės schemas.

2.2.Projekto vykdymo planas

Lentelė 3. Darbų vykdymo planas

Darbo numeris /mėnesiai	1	2	3	4	5	6	7	8	9	10	11	12
2010 rugsėjis												
2010 spalis												
2010 lapkritis												
2010 gruodis												
2011 sausis												
2011 vasaris												
2011 kovas												
2011 balandis												
2011 gegužė												
2011 birželis												
2011 liepa												
2011 rugpjūtis												
2011 rugsėjis												
2011 spalis												
2011 lapkritis												
2011 gruodis												
2012 sausis												
2012 vasaris												
2012 kovas												
2012 balandis												
2012 gegužė												

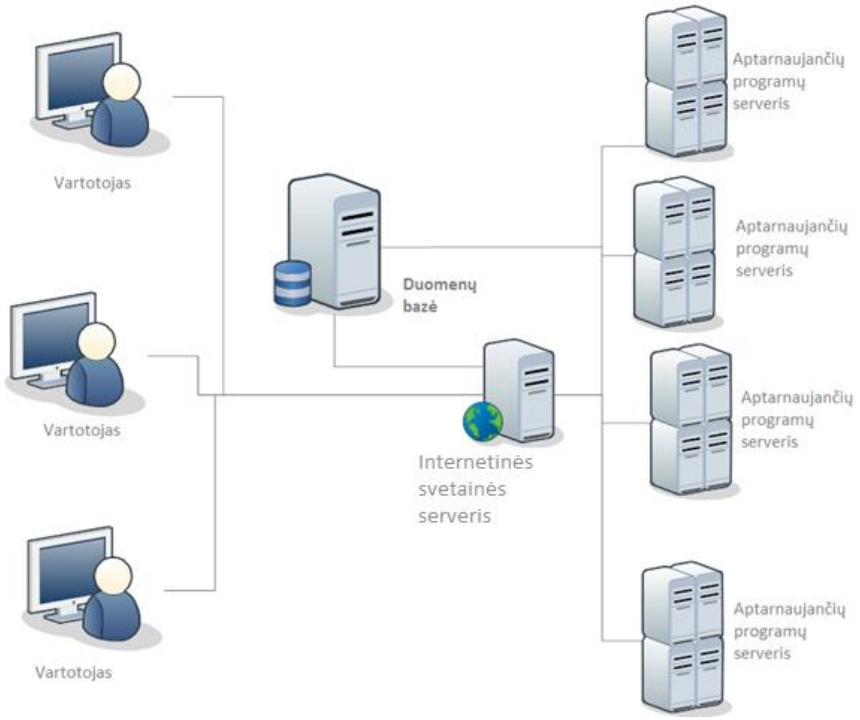
Darbų paaiškinimai:

1. Internetinės matematinio programavimo ir modeliavimo sistemos realizavimui reikalavimų formulavimas.
2. Informacijos paieška.
3. Informacijos šaltinių analizė.
4. Internetinės matematinio programavimo ir modeliavimo sistemos projektui reikalingų *UML* diagramų braižymas.
5. Konsultacijos su darbo vadovu.
6. Darbo aprašymo sudarymas.
7. Duomenų bazės struktūros kūrimas.
8. Internetinės svetainės kūrimas.
9. Programos, skirtos vartotojų programoms apdoroti, kūrimas.
10. Programinio kodo rašymas.

11. Internetinės matematinio programavimo ir modeliavimo sistemos programinio kodo derinimas.
12. Internetinės matematinio programavimo ir modeliavimo sistemos testavimas.

Įvykdyti visus darbe numatytus uždavinius planuojama pagal pateiktų darbų vykdymo planą (žr. Lentelė 3). Darbai 1, 2, 3 yra skirti pirmajam uždaviniui – matematinio programavimo ir modeliavimo sistemų analizei atlikti. Jos metu reikia išsiaiškinti, kokios dabar yra sistemos pritaikytos matematiniam programavimui ir modeliavimui, ir kaip jos veikia. Kad įgyvendinti antrąjį ir trečiąjį uždavinį reikės atlikti 2, 3, 4, 7, 8, 9 darbus. Internetinei matematinio programavimo ir modeliavimo sistemai sukurti reikės atlikti 2, 3, 7, 8, 9, 10 darbus. O norėdami ją ištestuoti ir palyginti – 2, 3, 10, 11, 12 darbus.

2.3.Pradinis projekto aprašymas

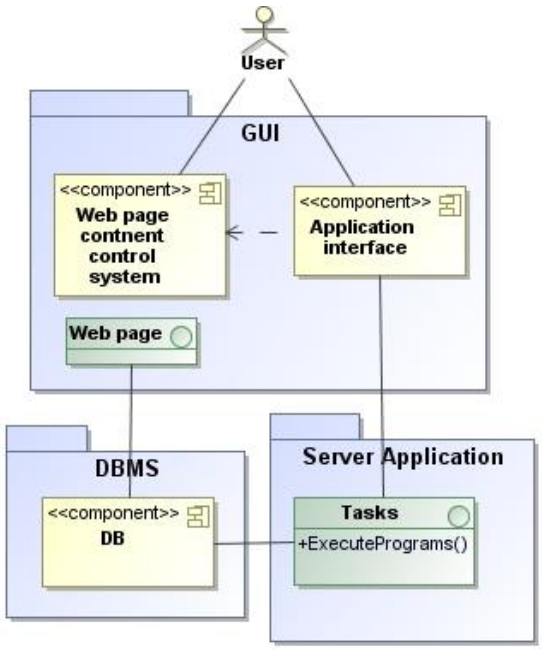


Pav 4. Projekto schema

Pradiniame mūsų projekte (žr. pav. 4) yra numatyta, kad matematinio programavimo ir modeliavimo sistemos vartotojai (kurių kiekis yra neapibrėžtas) turi turėti prieigą prie internetinės svetainės per naršyklę. Toje aplinkoje turėtų turėti galimybę sukurti naują arba redaguoti jau esantį internetinį puslapį, skirtą kokiai nors matematiniai problemai spręsti. Joje būtų galima aprašyti matematinius modelius ir skaitinius algoritmus. Taip pat pagal tuos algoritmus sukurti programas ir aprašyti jų sąsajas. Pagal sąsajos aprašymą *HTML* programavimo

kalba turi būti sugeneruota duomenų forma ir patalpinta šalia matematinio modelio ir skaitinio algoritmo aprašymo. O tai reiškia, kad programas bus galima vykdyti tiesiog iš svetaines, įrašius parametrus į duomenų formoje esančius laukus. Internetinėje svetainėje turi būti realizuota paieška, kuri vartotojui leistų ieškoti norimų programų, matematinių modelių ar skaitinių algoritmų, o taip pat leistų juos keisti ir tobulinti. Kadangi leidžiame tobulinti matematinį modelį, skaitinius algoritmus ir programas visiems, galime tikėtis, kad bus rastas efektyviausias problemos sprendimas. Vartotojai gali remtis kitų patirtimi bei pridėti kažką naujo. Toks yra vikinomikos principas, kada turimus išteklius saviorganizacijos būdu, panaudojame problemai spręsti. Visi pakeitimai internetinėje svetainėje bus išsaugomi. Tai leis vartotojui pasirinkti internetinio puslapio turinio versiją. Tam, kad vartotojams būtų lengviau programuoti mūsų matematinio programavimo sistemoje, mums reikia realizuoti matematinių funkcijų bibliotekos modulį, nes *JavaScript* nėra skirta vien tik matematiniams skaičiavimams, su kuria programuos vartotojai. Joje nėra tokių funkcijų kaip matricų daugyba, sudėtis, atimtis ar kitų sudėtingų matematinių funkcijų. Todėl reikia vartotojams sudaryti galimybę tokias funkcijas rasti sistemoje bei pridėti naujas, jas klasifikuoti, kad galėtų panaudoti savo programose.

Internetinės svetainės aplinkos kalba turėtų būti anglų kalba, nes sistema skirta ne vien tik Lietuvos vartotojams. Pačios informacijos saugojimas turėtų būti galimas bet kokia kalba. Visa informacija (matematinių modelių ir skaitinių algoritmų aprašymai, internetinių puslapių struktūros, programų kodai su vartotojo sąsaja) turi būti saugoma duomenų bazėje. Vartotojo sukurtų programų vykdymas turi būti atliekamas asinchroniniu būdu. Tai yra internetinėje svetainėje esančios programos parametrai iš duomenų formos turi būti perduoti į vieną iš serverių, kuriame veiktų serverinė programa. Ši priimtų duomenis ir sukurtų bei paleistų procesą, kuris apdorotų gautus duomenis bei paleistų *JavaScript* programavimo kalba parašytą vartotojo programą. Procesui pabaigus darbą, gautas rezultatas būtų siunčiamas į duomenų bazę.



Pav 5. Internetinės matematinio programavimo ir modeliavimo sistemos komponentai

Taigi visą internetinio matematinio programavimo ir modeliavimo sistemą sudarys trys dalys (žr. pav. 5):

1. Vartotojo sąsaja (*GUI*). Tai internetinė svetainė su turinio valdymo sistema.
2. Programa, kuri serveriuose priiminės iš internetinės svetainės duomenis (*Server Application*). Ji sukurs *Task* procesą, kuris tuos duomenis interpretuos ir vykdys vartotojo programą.
3. Duomenų bazė (*DBVS*), kurioje bus saugoma visa informacija kurią sukurs vartotojas.

3. Realizacinė dalis

3.1. Darbų eigos grafikas

Galutiniame darbo eigos grafe (žr. Lentelė 4) darbo savaičių skaičius truputį persiskirstė. Daugiau laiko buvo skirta informacijos paieškai ir analizei. Taip pat susidūrus su tam tikromis problemomis (jos aprašytos 3.3 ir 3.4 skyreliuose), turėjome keisti pradinį projektą, o tiksliau vieną jo dalį – programą, kuri turėtų apdoroti vartotojo programas serveryje. Todėl išaugo numatytas darbo savaičių skaičius, skirtas internetinės matematinio programavimo ir modeliavimo sistemos projektavimui ir kūrimui ir sumažėjo laiko testavimui.

Lentelė 4. Darbų eigos grafas

Darbo numeris /mėnesiai	1	2	3	4	5	6	7	8	9	10	11	12
2010 rugsėjis												
2010 spalis												
2010 lapkritis												
2010 gruodis												
2011 sausis												
2011 vasaris												
2011 kovas												
2011 balandis												
2011 gegužė												
2011 birželis												
2011 liepa												
2011 rugpjūtis												
2011 rugsėjis												
2011 spalis												
2011 lapkritis												
2011 gruodis												
2012 sausis												
2012 vasaris												
2012 kovas												
2012 balandis												
2012 gegužė												

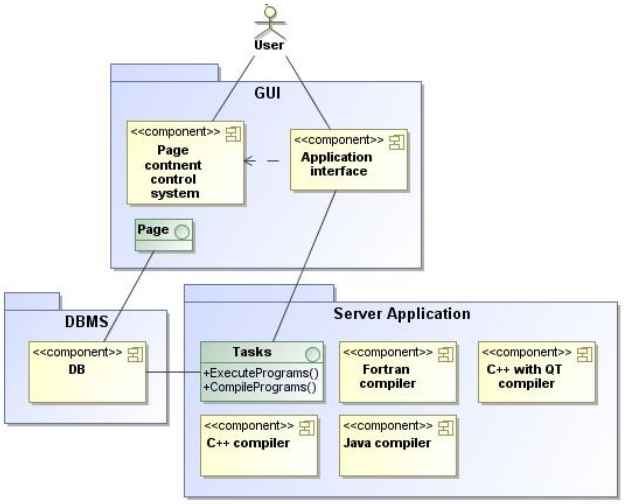
Darbų aprašymas:

1. Internetinės matematinio programavimo ir modeliavimo sistemos realizavimui reikalavimų formulavimas.
2. Informacijos paieška.
3. Informacijos šaltinių analizė.
4. Internetinės matematinio programavimo ir modeliavimo sistemos projektui reikalingų *UML* diagramų braižymas.
5. Konsultacijos su darbo vadovu.
6. Darbo aprašymo sudarymas.
7. Duomenų bazės struktūros kūrimas.
8. Internetinės svetainės kūrimas.
9. Programos, skirtos vartotojų programoms apdoroti, kūrimas.
10. Programinio kodo rašymas.
11. Internetinės matematinio programavimo ir modeliavimo sistemos programinio kodo derinimas.
12. Internetinės matematinio programavimo ir modeliavimo sistemos testavimas.

3.2.Galutinis projekto aprašymas

Suprojektuota ir realizuota internetinė matematinio programavimo ir modeliavimo sistema, kurią sudaro trys dalys (žr: pav. 6).

1. Vartotojo sąsaja (*GUI*). Tai internetinė svetainė su turinio valdymo sistema.
2. Programa, kuri serveriuose priima iš internetinės svetainės duomenis (*Server Application*).
3. Duomenų bazė (*DBVS*), kurioje saugoma visa informacija kurią sukuria vartotojas.



Pav 6. Internetinės matematinio programavimo ir modeliavimo sistemos komponentai

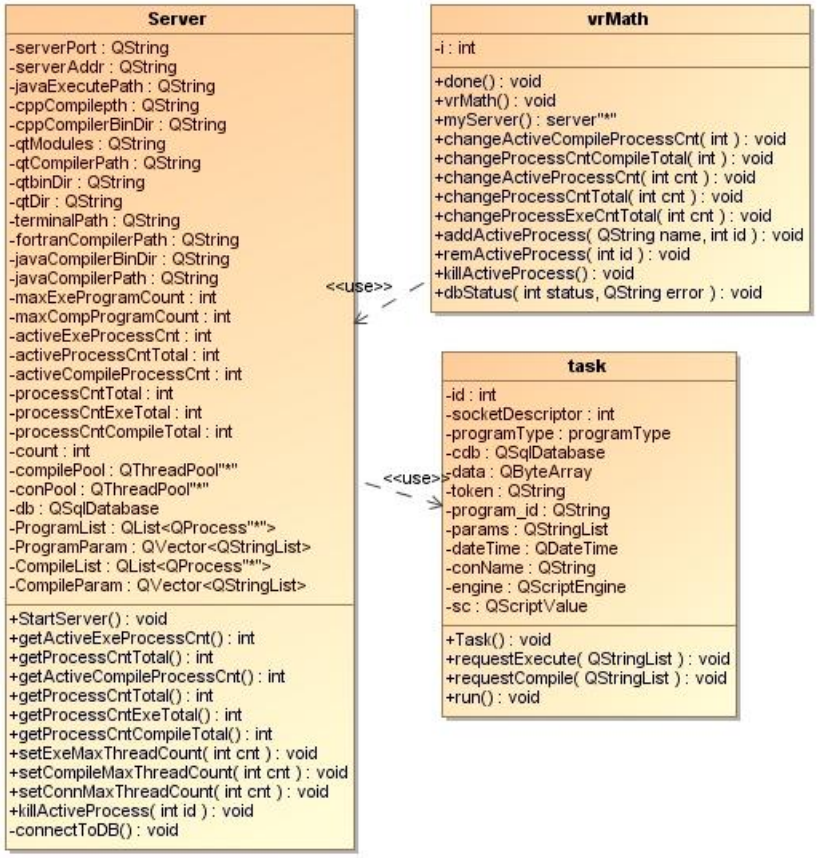
Matematinio programavimo ir modeliavimo sistemos internetinė svetainė sukurta panaudojant *MVC (*angl. *Model–View–Controller***)** technologiją. Ši technologija leidžia visą sistemą išskirstyti į tris dalis: modelį, vartotojo sąsają ir kontroliuojančią dalį. Pastaroji apjungia modelį ir vartotojo sąsają. Naudojant tokią technologiją supaprastėja pačios sistemos kūrimas ir jos testavimas, nes kiekviena dalis turi savo vaidmenį. Internetinėje svetainėje realizuoti šie moduliai:

- *Internetinio puslapio kūrimas pagal šabloną*. Šis modulis leidžia vartotojui kurti internetinio puslapio turinį pagal šabloninę struktūrą bei automatiškai generuoja internetinio puslapio turinio aprašymą. Taip pat jame realizuotas viso internetinio puslapio turinio saugojimas ir jo versijavimas.

- *WYSIWYG* (angl. *What You See Is What You Get*) *teksto redaktorius*. Šio modulio pagalba, galima aprašyti matematinius modelius ir skaitinius algoritmus internetiniame puslapyje. Jis suteikia galimybę naudotis dauguma teksto formatavimo funkcijų bei apsaugo nuo kenksmingo kodo įskiepiu į internetinę svetainę.
- *Programų rašymo modulis*. Jis atsakingas už programų bei jų vartotojo sąsajos kūrimą ir redagavimą.
- *Matematinių funkcijų bibliotekos modulis*. Šis modulis suteikia galimybę vartotojui kurti matematinių funkcijų bibliotekas. Bibliotekas galima klasifikuoti, kurti naujas, redaguoti jose esančias arba pridėti naujas funkcijas, bei atlikti tų funkcijų paiešką.

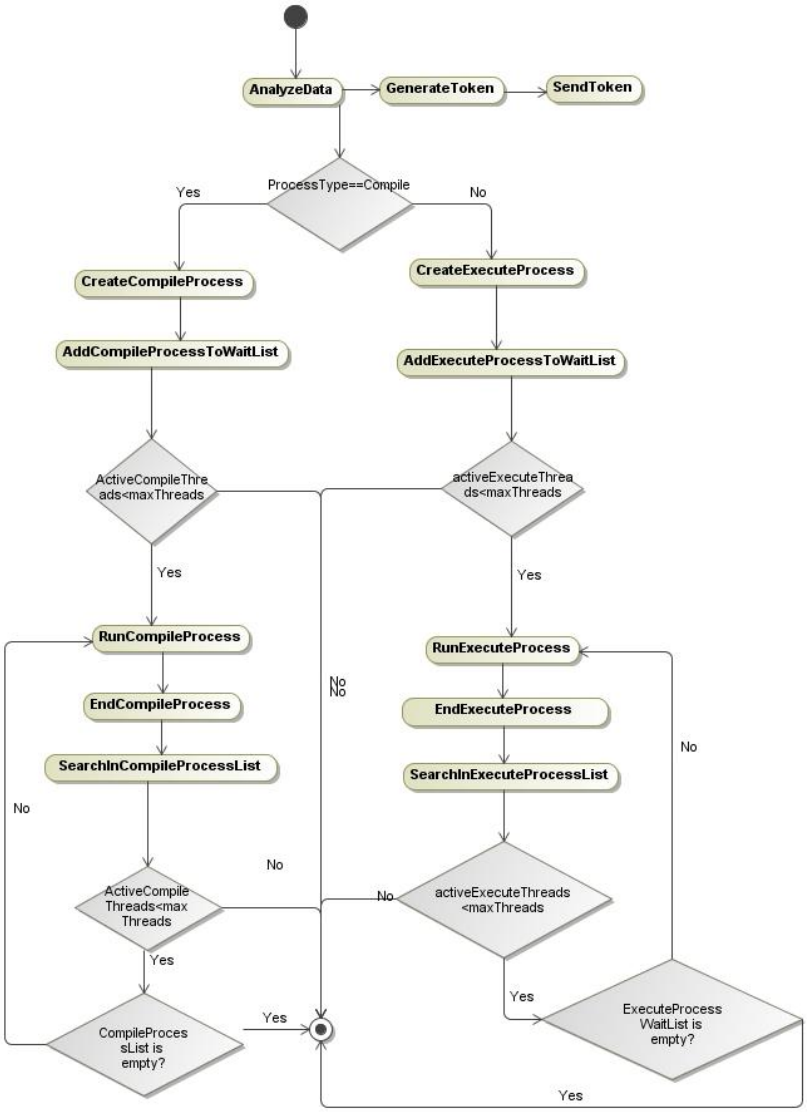
Kaip naudotis internetine matematinio programavimo ir modeliavimo sistema aprašyta prieduose ketvirtame skyrelyje. Viso projekto kodą rasite *DVD* laikmenoje, kurio turinys aprašytas priedų skyrelyje *DVD turinys*.

Vartotojo programas kompiliuoja ir paleidžia serveryje esanti programa. Ši programa turi tris klases *Server*, *Task*, *vrMath* (žr. pav. 7). Kai ši serverinė programa paleidžiama, susikuria *vrMath* objektas, kuris sukuria vartotojo sąsają bei sukuria dar vieną objektą *Server*.



Pav 7. Serveryje esančios programos klasių schema

Programos vartotojo sąsajoje galima matyti, kiek šiuo metu yra iš viso priimtų užduočių (kompiliuojamų ir vykdomų vartotojų programų) ir kurios šiuo metu yra vykdomos. Taip pat realizuota galimybė administratoriui nutraukti konkretų vykdomą procesą. Kaip vyksta procesas, kai vartotojo duomenys jau yra perduoti serverinei programai, galime matyti aštuntajame paveikslėlyje (žr. pav. 8). Priėmęs duomenų masyvą iš internetinės svetainės, *Server* objektas tų duomenų apdorojimui sukuria *Task* procesą, kuriame duomenys yra išanalizuojami ir vartotojui išsiunčia žymę (angl. *Token*).



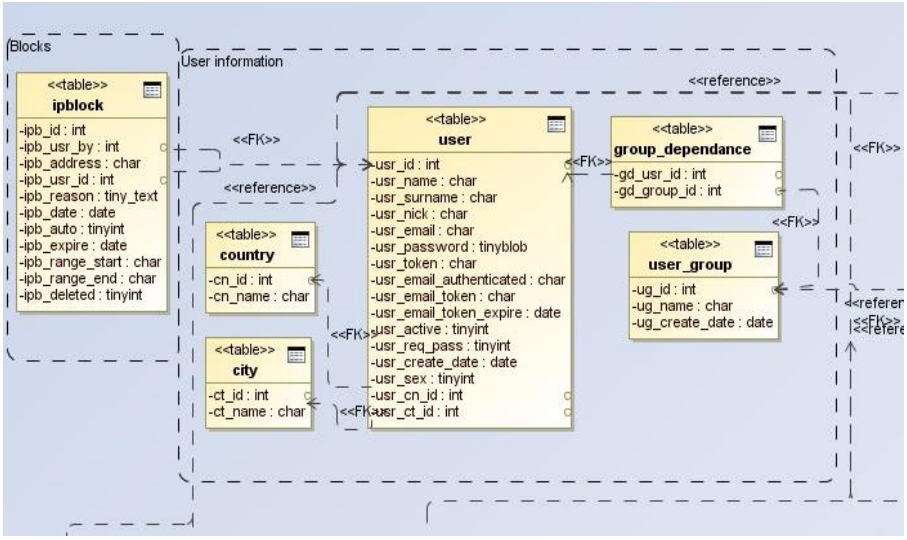
Pav 8. Duomenų analizė ir procesų kūrimas

Pagal šią žymę vartotojas galės rasti rezultatą, kurį pateikia baigusi darbą programa arba kompiliatorius. Baigęs darbą *Task* procesas, apdorotus duomenis pateikia *Server* objektui, pagal kuriuos yra sukuriamas arba vykdomosios programos arba kompiliavimo procesas. Visi procesai yra dedami į eilę. Jeigu šiuo metu neveikia daugiau procesų negu yra nustatyta didžiausia

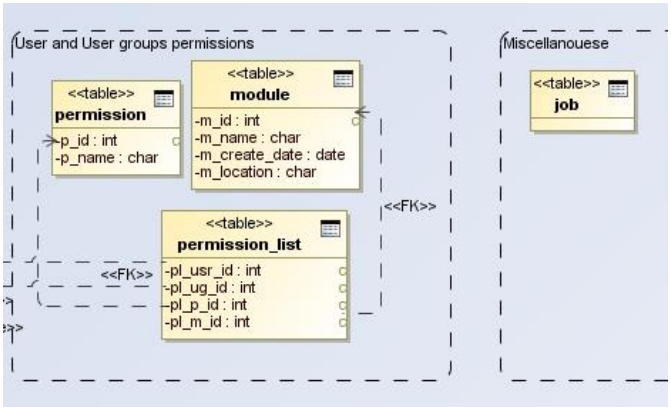
leidžiama riba, tada procesas yra paleidžiamas. O jeigu vykdomų procesų skaičius lygus maksimaliai jų reikšmei, tada laukiama, kol bent vienas procesas baigs darbą. Baigęs darbą, procesas visą gautą rezultatą grąžina į duomenų bazę. Šiuo metu ši serverinė programa, gali sukompiliuoti ir vykdyti *C/C++*, *C/C++* su *QT* biblioteka, *Java* ir *Fortran 90* programavimo kalbomis parašytas programas. *C/C++* ir *Fortran 90* parašytos programos yra kompiliuojamos su *Netlib Repository LAPACK* biblioteka. Dėl to galite naudotis šios bibliotekos teikiamais metodais. Plačiau apie biblioteką skaitykite 1.2.3 skyrelyje. Kaip naudotis serverine programa, rasite ketvirtame priedų skyrelyje.

Šios sistemos duomenų bazę sudaro trisdešimt keturios lentelės. Pagal joje saugomą informaciją, ją galima išskaidyti į modulius:

1. Modulis vartotojo duomenims saugoti. Jame numatyti sąryšiai tarp vartotojų bei jų grupių ir teisių(žr. pav. 9, 10).

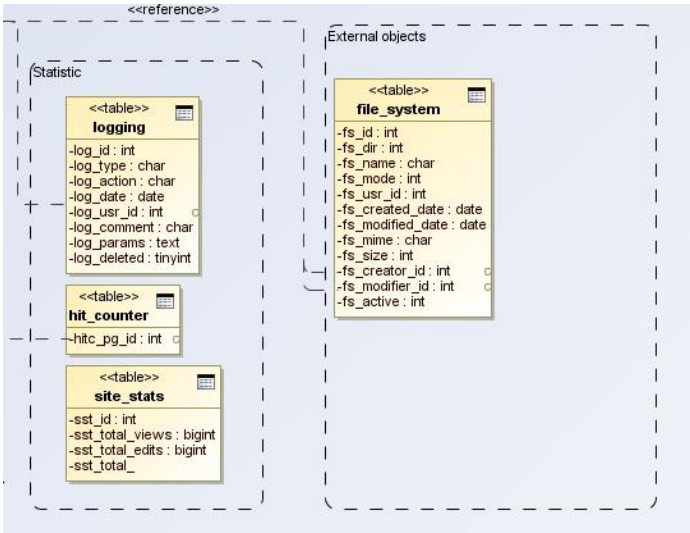


Pav 9. Lentelės vartotojo duomenims saugoti



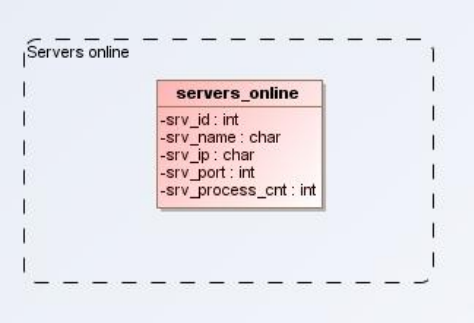
Pav 10. Lentelės vartotojo teisėms saugoti

2. Modulis įvairiai statistikai kaupti apie vartotojus ir internetinėje svetainėje esančius puslapius (žr. pav. 11).



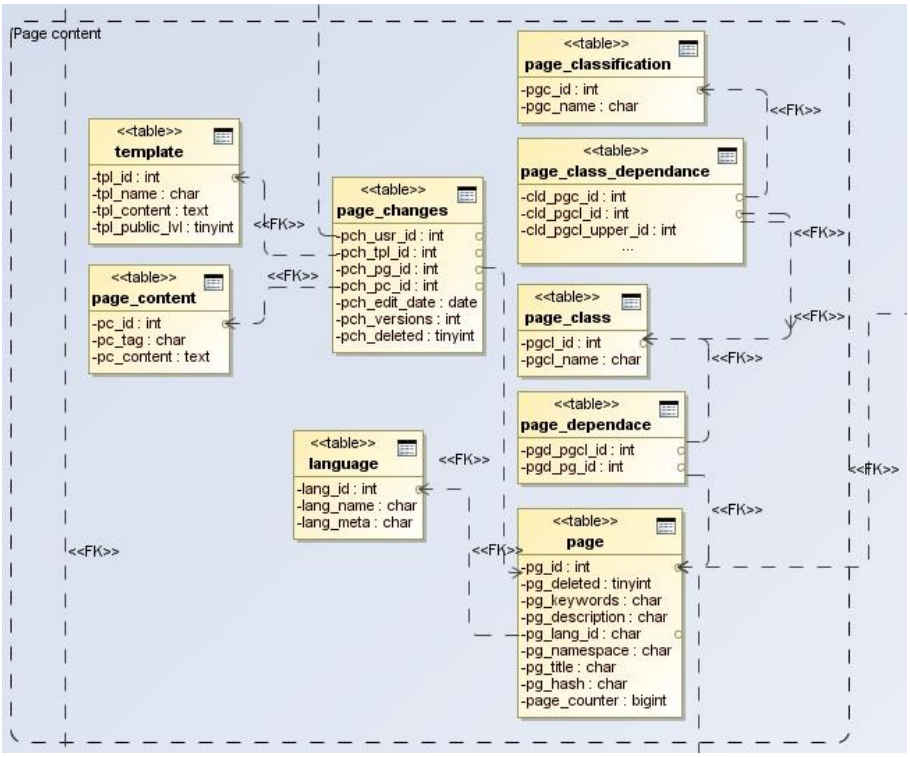
Pav 11. Lentelės vartotojo veiksmams ir internetinių puslapių statistikai saugoti

3. Modulis saugoti informaciją apie serveriuose esančias programas, jų būsenas ir užimtumą, kurios atsakingos už programų kompiliavimą ir paleidimą (žr. pav. 12).



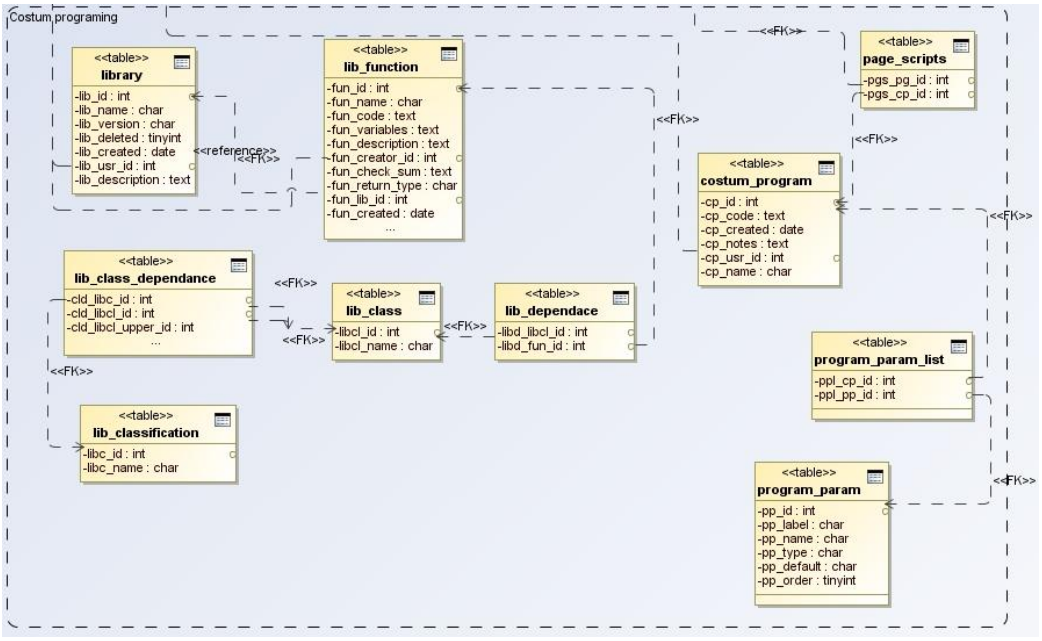
Pav 12. Lentelė saugoti duomenis apie veikiančias programas serveriuose

4. Modulis saugoti visą internetinės svetainės turinį, pakeitimus bei klasifikaciją (žr. pav. 13).



Pav 13. Lentelės saugoti internetinių puslapių turinį bei klasifikaciją.

5. Modulis saugoti programų kodą, jų parametrus. Taip pat matematinių funkcijų bibliotekas (žr. pav. 14) .



Pav 14. Lentelės skirtos saugoti matematines programas ir matematinių funkcijų bibliotekas

3.3.Problemos ir jų sprendimų būdai

1. Pirmoji problema, su kuria susidūrėme kurdami projektą, tai internetinės svetainės puslapių struktūra, kurioje vartotojai ne tik turi aprašyti matematinius modelius, skaitinius algoritmus, bet ir pridėti tų modelių realizaciją ir vartotojo sąsają. Šią problemą išspręsti iš dalies padėjo *wikipedia.com* svetainėje naudojama struktūra. Šioje svetainėje vartotojai patys aprašinėja puslapio struktūrą, pagal tam tikrus reikalavimus, o tada pildo turinį. Mes paėmėme šio tinklapio puslapių struktūros idėją ir ją patobulinome. Padarėme taip, kad vartotojui nereiktų pačiam aprašinėti paragrafų ir sukūrėme automatizuotą jų kūrimą. Kadangi visi puslapio paragrafai turi vienodą struktūrą, todėl buvo galima toje struktūroje atvaizduoti ir vartotojo aprašytas programų sąsajas.

2. Išsprendę pirmąją problemą susidūrėme su kita, kuri sekė iš pirmosios. Viską, ką sukūrė vartotojas reikėjo išsaugoti taip, kad būtų galima grįžti į ankstesnę internetinio puslapio versiją, išlaikant buvusią jo struktūrą ir funkcionalumą. O tai reiškia, kad turėtume sekti vartotojo veiksmus, lyginti, ką jis pakeitė tekste, puslapio struktūroje ir pakeitimus saugoti. Tai ne tik sudėtinga užduotis bet ir reikalaujanti daug kompiuterio resursų. Bet vėlgi, pasinaudojome *wikipedia.com* internetinės svetainės idėja, kaip saugoti puslapio pakeitimus į duomenų bazę. O idėja yra tokia, kad visa puslapio struktūra su visu turiniu nėra skaidoma į atskiras dalis (lenteles), o saugoma kaip vientisas objektas. Kadangi *wikipedia.com* internetinės svetainės duomenų bazės realizacija yra viešai prieinama, todėl kai kurias jos dalis mes panaudojame savo projekte.

3. Kita problema su kuria susidūrėme, tai duomenų apsikeitimas tarp internetinės svetainės ir serveryje esančių programų. Šią problemą išsprendėme panaudodami *TCP* (angl. *Transmission Control Protocol*) protokolą ir *JSON* (angl. *JavaScript Object Notation*) duomenų formatą. *TCP* protokolą duomenų perdavimui naudoja *Grid Mathematica*. Idėja naudoti *JSON* formatą duomenims perduoti, kilo dėl to, jog *PHP* turi metodus duomenų masyvus konvertuoti į šį formatą, o *QT* biblioteka su *JavaScript* interpretatoriumi, gali atkurti duomenis iš šio formato. Todėl nebereikėjo interpretuoti ir skaidyti gautų duomenų.

4. Dar viena problema, kuri iškilo kuriant sistemą, tai įvairių programų, parašytų skirtingomis programavimo kalbomis kompiliavimas. Tam, kad nekurti savo kompiliatorių *Java*, *C/C++*, *Fortran 90* programavimo kalboms, mes projekte panaudojome jau sukurtus kompiliatorius, kurie turi komandinės eilutės sąsaja. Tačiau, norint kompiliuoti programas, dar turėjome sugalvoti ir metodiką, kaip panaudoti kompiliatorius. Taigi, prieš kompiliuojant

programas yra sukuriamas katalogas, kurio pavadinimas – unikalus skaičius. Šiame kataloge yra įrašomas ir failas, kuriame yra vartotojo sukurtos programos kodas. Failo pavadinimas yra toks pat kaip katalogo. Tada, priklausomai nuo programavimo kalbos, yra paleidžiamas procesas su tam tikru kompiliatoriumi, kuris atlieka kompiliavimą. *C/C++*, *Java* ir *Fortran 90* programavimo kalbų kodo kompiliavimas yra panašus. Pirmiausia prie proceso sisteminės aplinkos dar pridedamas vienas parametras – kelias iki kompiliatoriaus katalogo. Tada procesui nurodoma *string* tipo eilutė, kurioje nurodomas visas kelias iki kompiliatoriaus vykdomojo failo, failo pavadinimas, kurį kompiliuosime su standartiniais kompiliavimo parametrais, bei programos pavadinimas. Programos pavadinimą visada nurodome tokį, koks yra ir katalogo pavadinimas kuriame yra programa. Tačiau yra keletas išimčių: *Java* programavimo kalba parašytai programai sukompiliuoti, procesui paduodamai eilutei nereikia nurodyti programos pavadinimo. O kompiliuojant *Fortran 90* programavimo kalba parašytas programas, proceso sisteminei aplinkai nereikia pridėti katalogo kelio, kuriame yra kompiliatorius. Sudėtingesnis variantas yra norint sukompiliuoti *C/C++* programavimo kalba parašytą programą, kur naudojama *QT* biblioteka. Tokiu atveju kataloge, kuriame yra programos kodas, turi būti sukurtas ir projektinis failiukas. Tada, kaip procesas, yra kviečiamas ne pats kompiliatorius, o operacinės sistemos komandinės eilutės aplinka. Komandinės eilutės aplinkoje reikia pridėti *QT* kompiliatoriaus vykdomojo failo, ir katalogo, kuriame jis yra, kelią. Tai atlikus, komandinės eilutės aplinkoje kviečiama komanda *qmake*. Ši komanda sukuria projektą iš projektinio failiuko. Po projekto sukūrimo kviečiamas *QT* kompiliatorius. Jam parametrų perduoti nereikia, nes jis juos pasiima iš sugeneruoto projekto.

5. Sukompiliavus programas, jos yra paruoštos vykdymui. Tačiau tų programų vartotojo sąsaja, kurioje įrašomi parametrai yra internetinėje svetainėje ir tai yra paprasta *HTML* programavimo kalba sugeneruota duomenų forma. Susidūrėme su kliūtimi, kaip tuos parametrus iš duomenų formos perduoti sukompiliuotai programai ir paruošti ją vykdymui. Šiai problemai išspręsti buvo priimtas toks sprendimas, kad visos vartotojo programos turi būti rašomos taip, kad turėtų komandinės eilutės sąsają. Pavyzdžiui kaip *C* programavimo kalboje programos pagrindinė dalis aprašoma eilute „*int main(int argc, char *argv[]);“*. Tokiu atveju duomenų formoje programos parametrai turi būti aprašomi tokia eilės tvarka, kokia juos priima programos komandinės eilutės sąsaja. Čia labai svarbu paminėti, kad pirmiausia duomenų formoje turi būti aprašomi paprasti parametrai, o duomenų failai turėtų būti aprašyti formos pabaigoje (irgi eilės tvarka). Taip yra dėl to, kad siunčiant formos duomenis su paprastais parametrais ir failais gaunami du masyvai ir nėra galimybes juos sujunkti į vieną taip, kad būtų žinoma jų tiksli pozicija masyve.

6. Testavimo metu buvo pastebėta, kad serveryje esanti programa, padidėjus užklausų kiekiui, nebeatsako į jas ir tampa neveiksni. Šią problemą lėmė ne vienas veiksnys. Visų pirma, padidėjus užklausų kiekiui, programa nebesugeba jų visų priimti ir apdoroti, nes išnaudojami visi serverio resursai tik tam darbui atlikti. Šiai problemai išspręsti panaudojome *QT* bibliotekos priemonę *QThreadPool*. Tai yra klasė, kuri aprašo gijų masyvą. Ji leidžia apsirašyti kiek gijų dirbs su tam tikrais objektais. Mūsų programoje su *Task* objektais. Jeigu pasiekiamas maksimalus gijų skaičius, o objektų daugėja, jie dedami į eilę. Baigusi darbą su vienu objektu, gija pasiima kitą iš objektų masyvo. Tokiu būdu suvaldomi serverio resursai vienam darbui atlikti. Kitas veiksnys, kuris stabdė serverio ir programos darbą buvo programų vykdymas ir kompiliavimas. Šioje vietoje padarėme taip, kad visi procesai būtų kaupiami virtualiuose konteineriuose, dedami į eilę ir vienu metu nebūtų vykdoma daugiau negu yra nustatyta. Tam, kad kompiliavimo darbams netrukdytų vykdomosios programos (kurių vykdymo laikas gali užsitęsti kur kas ilgiau), tokiems procesams sukurtas atskiras virtualus konteineris. Todėl vartotojo programų kompiliavimo ir vykdymo darbai vykdomi lygiagrečiai.

3.4.Darbo rezultatų analizė

Kuriant internetinio matematinio programavimo ir modeliavimo sistemą paaiškėjo, jog pradiniame projekte numatytos *JavaScript* programavimo kalbos mums nepakaks. Nors ir buvo galima *JavaScript* programavimo kalba realizuoti matematinius modelius, programos nebuvo efektyvios, negalėjome jų išlygiagretinti. Be to buvo sudėtinga tokioms programoms perduoti parametrus iš duomenų formų, esančių internetinėje svetainėje. Dėl šių priežasčių buvo nuspręsta panaudoti kitas programavimo kalbas, kuriomis parašytas programas būtų galima lygiagretinti, taip padidinant jų efektyvumą. Taigi, vietoj *JavaScript* interpretuojamos programavimo kalbos, buvo pasirinktos kompiliuojamos *C/C++*, *Java*, *Frotran 90* ir *C/C++* su *QT* biblioteka programavimo kalbos. Kad įgyvendintume šį sumanymą, sugalvojome metodiką, kaip panaudoti esamus jau sukurtus kompiliatorius ir prijunkti juos prie mūsų sistemos. Teoriškai prie sistemos galime prijungti visas programavimo kalbas, kurių kompiliatorius ar interpretatorius turi komandinės eilutės sąsajas. Tas pats yra ir su vartotojo sukurtomis programomis. Internetinė matematinio programavimo ir modeliavimo sistema gali vykdyti tik tokias programas, kurios turi komandinės eilutės sąsają. Taip pat prie sistemos prijungėme *Netlib Repository LAPACK* biblioteką, kurios metodus galima naudoti *C++* ir *Fortran 90* programavimo kalbose. Joje jau yra realizuoti algoritmai, kurie padeda spręsti įvarius matematinio programavimo uždavinius. Be

to internetinėje matematinio programavimo ir modeliavimo sistemoje sukurtos priemonės vartotojams kurti ir tobulinti matematinių funkcijų bibliotekas įvairioms programavimo kalboms, kuriomis vėliau galima bus pasinaudoti. Kadangi vartotojų visi matematinių modelių aprašymai, sukurti algoritmai, programos, matematinių funkcijų bibliotekos bus saugomi duomenų bazėje, todėl ši sistema nuolat plėsis ir tobulės. O tai leis vartotojams dar greičiau ir efektyviau išspręsti iškilusias matematines problemas ir uždavinius.

3.4.1 Internetinės matematinio programavimo ir modeliavimo sistemos testavimas

Internetinė matematinio programavimo ir modeliavimo sistema ištestuota, rašant programas su *Fortran 90*, *Java*, *C/C++* programavimo kalbomis bei *C/C++*, panaudojant *QT* ir *Netlib Repository LAPACK* bibliotekas. Buvo testuojamas programų kompiliavimas, jų vykdymas, duomenų perdavimas vykdomoms programoms ir rezultato išvedimas vartotojui. Taip pat buvo ištestuota Monte Karlo metodą realizuojanti lygiagreti ir nuosekli programos bei programa, kuri sprendė tiesinių lygčių sistemą panaudojant *Netlib Repository LAPACK* bibliotekos metodus. Šio testavimo metu visos testinės programos parašytos minėtomis programavimo kalbomis buvo sukompiliuotos ir įvykdytos. Vartotojui gražintas programų rezultatas buvo toks kokio ir tikėtasi (žr. priedų 2 skyrių). Internetinės svetainės testai parodė, kad atliekami vartotojo veiksmai svetainėje, daro tai ko ir tikimasi. Tačiau pastebėta, jog sukūrus programos vartotojo sąsają su vienodais programos parametrų pavadinimais, jos vykdymo metu programai perduodamas tik paskutinis parametras. Dėl šios priežasties parametrų pavadinimai turi būti su skirtingais pavadinimais, nors sukurtai programai tie parametrų pavadinimai nėra svarbūs. Taip pat buvo atliktas sistemos apkrovos testas. Buvo bandoma vienu metu įvykdyti 500 vartotojo užklausų, kad paleistų lygiagrečią programą su Monte Karlo metodu. Atlikti keturi testai, su skirtingais serverinės programos nustatymais. Tai yra serverinė programa galėjo tuo pat metu paleisti nuo vieno iki keturių programų (procesų):

Lentelė 5. Sistemos apkrovos testo rezultatai

Procesų skaičius	Laiko tarpas per kurį buvo aptarnauti visi vartotojai	Laiko tarpas per kurį buvo įvykdytos visos programos
1 procesas	12 s	14 min. 21 s
2 procesai	16 s	9 min. 41 s
3 procesai	22 s	9 min. 51 s

4 procesai	-	-

Iš sistemos apkrovos testų rezultatų (žr. Lentelė 5) galima matyti, kad greičiausiai vartotojai aptarnaujami kai serverinė programa vartotojo programas paleidžia po vieną. Tačiau programų vykdymo laikas apytiksliai 48% ilgesnis lyginant su eksperimentu, kai serverinė programa paleidžia dvi programas vienu metu. Bet vartotojai antruoju atveju aptarnaujami 33% lėčiau. Toliau didinant serverinėje programoje paleidžiamų programų kiekį vienu metu vartotojų aptarnavimo ir programų vykdymo laikas tik didėja arba sutrikdo serverio darbą. Taigi sistemos stabilumas ir greitis priklauso nuo serverinės programos nustatymų bei žinoma nuo serverio skaičiavimo galios. Kadangi testai buvo atliekami su keturių branduolių procesoriumi, galime daryti prielaidą, jog serverinės programos maksimalus procesų skaičius negali viršyti ½ procesoriaus branduolių skaičiaus, nes optimalus paleidžiamų programų kiekis vienu metu testuojamame serveryje yra dvi programos.

3.4.2 Internetinės matematinio programavimo ir modeliavimo sistemos palyginimas

Palyginsime, mūsų sukurtos internetinės matematinio programavimo ir modeliavimo sistemos funkcionalumą su kitomis panašiomis matematinio programavimo sistemomis, kurias nagrinėjome analitinėje darbo dalyje. Visų pirma, kad palygintume šias sistemas mums reikia pasirinkti kriterijus bei įsivesti svorinius koeficientus, kurie parodytų pasirinktų kriterijų svarbumą.

Lentelė 6. Matematinių programavimo sistemų palyginimo kriterijai ir svoriniai koeficientai

Kriterijus	**Svoris**
Matematinių funkcijų, modelių, programų panaudojimo galimybės	0.3
Sistemoje esančių matematinių funkcijų, modelių, programų papildymas ir tobulinimas	0.2
Vartotojo sąsajos galimybės	0.2
Naudojamos programavimo kalbos	0.1
Sukurtų programų vykdymas	0.1
Matematinio programavimo sistemų integravimas į kitas programas	0.1

Svarbiausias matematinio programavimo ir modeliavimo sistemos kriterijus, tai matematinių funkcijų, modelių ir programų panaudojimo galimybės programuojant ir sprendžiant uždavinius su sistema. Tai nulemia, kaip greitai ir efektyviai galime parašyti programą sprendžiamam uždaviniui. Dar vienas svarbus kriterijus yra sistemoje esančių matematinių funkcijų, programų papildymas ir tobulinimas. Jis nusako matematinės programavimo sistemos funkcionalumo praplėtimo ir tobulinimo galimybes. Taip pat svarbi ir vartotojo sąsaja, kurioje vartotojas kuria ir vykdo programas bei kaip tos sukurtos programos yra vykdomos ir kokiomis programavimo kalbomis jos parašytos. Turime atkreipti dėmesį į tai, ar matematinio programavimo sistema gali būti naudojama rašant programas su kitomis priemonėmis.

Lentelė 7. Matematinių programavimo sistemų vertinimo lentelė

Matematinio programavimo sistemų funkcionalumas/matematinio programavimo ir modeliavimo sistemos	**Mathematica**	**Scilab**	**Internetinė matematinio programavimo ir modeliavimo sistema**
1. Matematinių funkcijų, modelių, programų panaudojimo galimybės	10	7	8
2. Sistemoje esančių matematinių funkcijų, modelių, programų papildymas ir tobulinimas	3	3	9
3. Vartotojo sąsajos galimybės	9	5	7
4. Naudojamos programavimo kalbos	6	6	8
5. Sukurtų programų vykdymas	9	5	7
6. Matematinio programavimo sistemų integravimas į kitas programas	10	6	2
Suma padauginus iš koeficientų	7.9	5.4	7.3

Visos trys matematinio programavimo sistemos (žr. Lentelė 7) *Mathematica*, *Scilab* ir mūsų sukurta sistema turi matematinių funkcijų bibliotekas. *Mathematica* branduolys turi daugiau matematinių funkcijų, bei realizuotų algoritmų negu *Scilab* ar mūsų sukurta matematinio programavimo sistema *(*1.1.1, 1.1.2, 1.2.3 skyreliai*)*, kurias galima naudoti sprendžiant matematinio programavimo uždavinius. Mūsų sukurta sistema naudoja *Netlib Repository LAPACK* biblioteką *C++* ir *Fortran* programavimo kalba rašomoms programoms. Ši biblioteka turi gerokai daugiau metodų ir realizuotų algoritmų negu *Scilab*, todėl mūsų internetinė matematinio programavimo ir modeliavimo sistema šiuo požiūriu yra pranašesnė. *Mathematica* ir *Scilab* gali iškviesti iš failų funkcijas bei paprogrames, kurios prašytos jų kalba ir jas panaudoti

programose. Mūsų sukurta tokios galimybės neturi, nes sistema yra internetinė ir sukurta programa yra serveryje. Internetinėje matematinio programavimo ir modeliavimo sistemoje reikalingos paprogramės, pati programa, jų vartotojo sąsajos gali būti atvaizduotos viename internetiniame puslapyje, o funkcijos pridėtos prie matematinių funkcijų bibliotekų. Tačiau programiškai visas dalis susieti galima tik visą jų programinį kodą sujungus į vieną programą arba jeigu programos koncepcija leidžia atskiras paprogrames vykdyti po vieną ir gautus duomenis per vartotojo sąsają perduoti pagrindiniai programai. Šioje sistemoje teoriškai leidžiamas tik internetinių servisų iškvietimas. Taigi *Scilab* įvertinome 7 balais pagal pirmąjį kriterijų, nes matematinių funkcijų gausa nusileidžia internetinei matematinio programavimo ir modeliavimo sistemai, kurią įvertinome 8 balais.

Internetinė matematinio programavimo ir modeliavimo sistema iš nagrinėtų sistemų labiausiai išsiskiria pagal antrąjį kriterijų. Ją mes įvertinome 9 balais. Šioje sistemoje yra duomenų bazė, kurioje galima saugoti sukurtų programų, matematinių funkcijų programinį kodą, skaitinių algoritmų, matematinių modelių aprašymus. Vartotojai gali ieškoti ir naudotis šiais ištekliais, kurti, pildyti, tobulinti programas, matematinių funkcijų bibliotekas, algoritmus, modelius. Tai yra pati sistema ir jos galimybės yra plečiamos pačių vartotojų. Kitose matematinio programavimo sistemose to padaryti negalima, nes jų koncepcija skiriasi. Jos yra orientuotos į vieną vartotoją. Tai yra kiekvienas vartotojas turi nepriklausomą programavimo aplinką, o jų praplėtimo galimybės yra lokalios ir jos netampa sistemos dalimi. *Mathematica* ir *Scilab* gali tik iškviesti iš failų funkcijas bei paprogrames, kurios yra parašytos jų programavimo kalba. Visos jų funkcijos yra branduolyje ir jų patobulinti arba pakeisti ar peržiūrėti programinį kodą neįmanoma. Dėl to jas įvertinome 3 balais.

Nagrinėjamų matematinių programavimo ir modeliavimo sistemų vartotojo sąsajos įvairios. Mūsų sukurtos sistemos vartotojo sąsaja yra internetinė svetainė. *Mathematica* siūlo grafinę vartotojo sąsają bei atskirus modulius, skirtus perkelti *Mathematica* branduolio funkcionalumą į internetinius puslapius. *Scilab* turi grafinę bei komandinės eilutės vartotojo sąsajas. *Scilab* grafinė vartotojo sąsaja turi labai nedaug funkcijų. Galima minimaliai formatuoti programinį kodą ir valdyti programos vykdymo procesą. Rezultatus galima matyti grafiniu pavidalu dvimatėje ir trimatėje erdvėje. Tuo tarpu *Mathematica* grafinė vartotojo sąsajos galimybės yra kur kas didesnės. Ja galima atlikti programinio kodo derinimo darbus. Kurti parašytų programų grafinę vartotojo sąsają su dvimate ir trimate grafika bei animacija. *Web Mathematica* modulių pagalba, galima vykdyti programas internetiniame puslapyje bei gautus rezultatus atvaizduoti jame teksto ar grafiniu pavidalu. Mūsų sukurtoje sistemoje trūksta grafinio

duomenų atvaizdavimo. Tačiau galima kurti internetinius puslapius pagal šabloną (struktūrą), aprašyti algoritmus, matematinius modelius, matematinių funkcijų bibliotekas, vykdyti jų paiešką. Taip pat galima kurti programas ir jų grafinę sąsają (*HTML* programavimo kalbos duomenų įvesties forma) bei jas vykdyti. Visa informacija yra automatiškai versijuojama. Šią programavimo sistemą pagal trečiąjį kriterijų įvertinome 7 balais, nes trūksta duomenų vizualizacijos modulio. Tačiau didelis jos privalumas, kad visa informacija talpinama vienoje vietoje (matematinių modelių aprašymai, algoritmai, programos ir jų programinis kodas). Taip pat parodomos kompiliavimo klaidos bei programos vykdomos tiesiogiai iš internetinės svetainės. Todėl vartotojui nereikia nieko papildomai instaliuoti, kad jas paleisti. Dar vienas sistemos privalumas toks, kad bet kuris vartotojas gali naudotis esamais ištekliais. *Scilab* vartotojo sąsają įvertinome 5 balais, nes tai yra tik patobulinta komandinės eilutės sąsaja ir vienintelis jos privalumas tai duomenų atvaizdavimas dvimatėje ir trimatėje grafikoje. *Mathematica* vartotojo sąsajos galimybes įvertinome 9 balais, nes *Web Mathematica* modulius ir internetinius puslapius reikia kurti atskirai, norint turėti sukurtų programų vartotojo sąsają internetinėje svetainėje.

Dar vienos mūsų sukurtos sistemos išskirtinumas yra įvairių programavimų kalbų palaikymas. Sistemoje yra galimybė programuoti su *C/C++, Java, QT, Fortran* programavimo kalbomis. To pasekoje vartotojams lengviau rašyti programinį kodą, nes nereikia mokytis specifinės programavimo kalbos. Be to, šios programavimo kalbos yra kompiliuojamos, dėl to programų vykdymas yra spartesnis negu interpretuojamų programų. Teoriškai prie sistemos galime prijungti daug daugiau programavimo kalbų (žr. 3.3, 3.4 skyreliuose). Tuo tarpu kitos mūsų nagrinėtos matematinio programavimo sistemos turi savo programavimo kalbas. Jų sintaksė yra supaprastinta ir galimai lengvesnė, lengviau įsimenama vartotojui. Tiesa, *Scilab* įrankis gali versti programinį kodą iš *MATLAB* programavimo kalbos į savo. Taip yra dėl to, jog *Scilab* naudojama programavimo kalba yra labai panaši į *MATLAB* programavimo kalbą. Kadangi mūsų sistemoje naudojamos įvairios programavimo kalbos, todėl šią sistemą įvertinome 8 balais, pagal ketvirtąjį kriterijų. Kitas sistemas įvertinome 6 balais, nes turi tik savo interpretuojamas programavimo kalbas.

Pagal penktąjį kriterijų daugiausiai balų surinko *Mathematica.* Ją įvertinome 9 balais. Ši sistema gali vykdyti parašytas programas nuosekliai ir lygiagrečiai (gali naudoti iki 16 branduolių). Taip pat galima naudoti ir paskirstytos atminties sistemas vykdant skaičiavimus tinkle. Tuo tarpu internetinė matematinio programavimo ir modeliavimo sistema programas gali vykdyti tik nuosekliai arba lygiagrečiai. Norėdamas vykdyti programą mūsų sistemoje vartotojui

nereikia įdiegti specialios programinės įrangos, užtenka tik interneto naršyklės. Tuo tarpu *Scilab* ir *Mathematica* reikalauja kompiuteryje turėti specialią programinę įrangą. Su *Web Mathematica* dar galima iškviesti programos vykdymą iš internetinio puslapio, bet tai nėra lygiavertis dalykas mūsų sukurtai sistemai. *Web Mathematica* nėra internetinės svetainė ar turinio valdymo sistema. Tai atskirai kuriami moduliai, kuriuos galima įdiegti į internetinį puslapį. Scilab įvertinome silpniausiai, 5 balais, nes ji programas vykdo nuosekliai ir neturi tokių galimybių kokias siūlo kitos nagrinėtos sistemos. Internetinę matematinio programavimo ir modeliavimo sistemą 7 balais, nes negali vykdyti programų paskirstytos atminties sistemose.

Matematinio programavimo sistemos *Mathematica* ir *Scilab* turi integracines priemones pritaikytas *Java, C/C++, C#* programavimo kalboms. *Mathematica* dar turi *MathLink* protokolą per kurį gali kitos sistemos kviesti jos branduolio metodus. Mūsų sistemą irgi įmanoma integruoti, tačiau ji nėra tam pritaikyta. Norint panaudoti internetinę matematinę programavimo ir modeliavimo sistemą su kitomis programomis, sistemomis reikia į internetinės svetainės serveryje esantį *execute.php* scenarijų, perduoti duomenų masyvus *POST* metodu, o rezultato ieškoti *download.php* scenarijuje. Todėl pagal integracijos su kitomis sistemomis kriterijų mūsų sistemą įvertinome silpniausiai 2 balais. Truputi didesnes galimybes turi *Scilab*, dėl to ją įvertinome 6 balais, o *Mathematica* 10 balų.

Palyginę sistemas pagal kriterijus ir gautą bendrą balų sumą, galime teigti, kad *Mathematica* yra geresnė matematinio programavimo sistema už internetinę matematinio programavimo ir modeliavimo sistemą, o pastaroji už *Scilab*. Tačiau internetinė matematinio programavimo ir modeliavimo sistema pagal išvardintus kriterijus yra labai panaši *Mathematica*. Be to ji išsiskiria iš kitų matematinio programavimo sistemų, nes turi vieną bendrą duomenų bazę, kurioje galima ieškoti realizuotų programų, matematinių funkcijų, matematinių modelių ir skaitinių algoritmų aprašymų. Viskas yra paremta atviro kodo principų ir vartotojai gali plėsti, tobulinti šios sistemos galimybes, jos turinį. Taip pat leidžia vykdyti programas, nenaudojant vartotojo kompiuterio resursų ir nediegiant specialių programų į jį.

3.5. Patarimai, pastebėjimai, rekomendacijos

Šiuo metu sistema veikia tik su vienu serveriu, skirtu programų vykdymui. Duomenų bazėje ir projekte buvo numatyta, kad sistema gali dirbti ir su daugiau serverių. Šiuo metu kiekvienas atskiras serveris sukompiliuotą kodą laiko pas save ir jį vykdyti gali tik pas save. Norint, kad ir kiti serveriai galėtų vykdyti kito serverio sukompiliuotas programas, prieš tai jas

turi atsisiųsti. Tam reikia sukurti papildomą servisą, kuris palaikytų tą patį turinį visuose serveriuose, kurie atsakingi už kodo kompiliavimą ir vykdymą.

Taip pat reiktų įdiegti daugiau saugumo priemonių. Šiuo metu internetinėje svetainėje yra pritaikytos kelios saugumo priemonės, kad vartotojai negalėtų įdiegtį kenksmingų scenarijų. Tai yra, kai kurie formų laukai yra šifruojami, o tekste, kuris yra saugomas į duomenų bazę, neleidžiama naudoti *JavaScript* įskiepiu. Tačiau vartotojas gali realizuoti bet kokią programą, kuri yra kompiliuojami serveryje. Šioje vietoje reiktų padaryti taip, jog vykdoma programa negalėtų įrašinėti į serverį jokių failų ir skaityti tik savo katalogo turinį.

Dar pastebėta, kad prie internetinės matematinio programavimo ir modeliavimo sistemos galima prijungti daugiau programavimo kalbų, su kuriomis galėtų vartotojas programuoti (žr. 3.3 ir 3.4 skyreliuose).

Kai kurios matematinio programavimo sistemos turi galimybę apdoroti duomenis grafiškai. Internetinėje matematinio programavimo ir modeliavimo sistemoje to trūksta.

Išvados ir rezultatai

1. Remiantis galutinio projekto darbo rezultatais, galime teigti, jog visi iškelti uždaviniai yra įvykdyti ir darbo tikslas – pasinaudojant vikinomikos idėją, sukurti internetinę matematinio programavimo ir modeliavimo sistemą – yra pasiektas.
2. Remiantis galutinio projekto darbo bei testavimo rezultatais, galime teigti, kad sukurtoje sistemoje galima programuoti su *C/C++*, *Java*, *Fortran 90* programavimo kalbomis bei naudoti *QT* ir *Netlib Repository LAPACK* bibliotekas.
3. Internetinės matematinio programavimo ir modeliavimo sistemos testavimo rezultatai parodė, kad sukurta sistema veikia stabiliai.
4. Iš atlikto internetinės matematinio programavimo modeliavimo sistemos palyginimo su *Mathematica* ir *Scilab* sistemomis, galime teigti, kad internetinės matematinio programavimo sistemos funkcionalumas yra didesnis negu *Scilab* bei panašus į *Mathematica*.

Literatūra ir informaciniai šaltiniai

1. Julijonas Jonas Kaladė, Leonas Valkūnas. *Matematinis modeliavimas ir sinergetikos pagrindai*. Vilnius, 2008.
2. *Matematinis modeliavimas* [interaktyvus] [žiūrėta 2010m. gruodžio 27d.]. Prieiga per internetą: <http://lt.wikipedia.org/wiki/Matematinis_modeliavimas>.
3. *Mathematical model* [interaktyvus] [žiūrėta 2010m. gruodžio 27d.]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Mathematical_model>.
4. Stefan Steinhaus. *Comparison of mathematical promgrams for data analysis*. Munich/Vokietija, 2008.
5. *Mathematica* [interaktyvus] [žiūrėta 2011m. sausio 17d.]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Mathematica>.
6. Michael Baudin. *Introduction to Scilab.* The Scilab Consiortum, 2010.
7. C Bunks, J.-P. Chancelier, F. Delebecque. C. Gomez, M. Goursat, R. Nikoukhah. and S. Steer. *Engineering and Scientific Computing With Scilab.* Birkhauser Boston, 1999.
8. *ESSL and Parallel ESSL library* [interaktyvus] [žiūrėta 2012m. vasario 27d.]. Prieiga per internetą: <http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp>.
9. *IBM Engineering and Scientific Software Library* [interaktyvus] [žiūrėta 2011m. balandžio 14d.]. Prieiga per internetą: <http://people.sc.fsu.edu/~jburkardt/f_src/essl/essl.html>.
10. *IBM Parallel Engineering and Scientific Software Library* [interaktyvus] [žiūrėta 2010m. gegužės 3d.]. Prieiga per internetą: <http://people.sc.fsu.edu/~jburkardt/f_src/pessl/pessl.html >.
11. William T. Vetterling, Polaroid Corporation, Brian P. Flannery, Saul A. Teukolsky, William H. Press. *Numerical Recipes in C. The Art of Scientific Computing*,. Cambridge university press, 2002.
12. *IMSL Numerical Libraries* [interaktyvus] [žiūrėta 2011m. sausio 11d.]. Prieiga per internetą: < http://www.vni.com/products/imsl/ >.
13. *NAG Numerical* [interaktyvus] [žiūrėta 2011m. vasario 3d.]. Prieiga per internetą: <http://www.nag.co.uk/numeric/numerical_libraries.asp>.

14. K. Žilinskas. *Stochastinio tiesinio programavimo Monte Karlo metodu tyrimas.* Daktaro disertacija Vilnius 2007.
15. Don Tapscott, Anthony D. Williams. *Wikinomics – How Mass Collaboration Changes Everything.* 2006 m.
16. *Vikinomika* [interaktyvus] [žiūrėta 2011m. sausio 22d.]. Prieiga per internetą: <http://lt.wikipedia.org/wiki/Vikinomika>.
17. *Matematinis modeliavimas* [interaktyvus] [žiūrėta 2010m. gruodžio 27d.]. Prieiga per internetą: <http://www.techmat.vgtu.lt/model_pag.html>.
18. *Scilab* [interaktyvus] [žiūrėta 2012m. vasario 27d.]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Scilab>.
19. *Different execution modes of Scilab* [interaktyvus] [žiūrėta 2012m. kovo 14d.]. Prieiga per internetą: <http://wiki.scilab.org/Different%20execution%20modes%20of%20Scilab>.
20. *ASP versus PHP* [interaktyvus] [žiūrėta 2011m. sausio 29d.]. Prieiga per internetą: <http://www.aspvsphp.com/>.
21. *PHP vs ASP.net Comparison* [interaktyvus] [žiūrėta 2011m. kovo 15d.]. Prieiga per internetą: <http://www.comentum.com/php-vs-asp.net-comparison.html>.
22. Dhananjay Nene, *Performance Comparison - C++ / Java / Python / Ruby/ Jython / JRuby / Groovy* [interaktyvus] [žiūrėta 2011m. kovo 16d.]. Prieiga per internetą: <http://blog.dhananjaynene.com/2008/07/performance-comparison-c-java-python-ruby-jython-jruby-groovy>.
23. Žilinskas, K. *Matematinis programavimas. I dalis. Tiesinis programavimas.* VŠĮ Šiaulių universiteto leidykla, 2007.
24. Žilinskas, K. *Optimizavimo metodai ir algoritmai* [interaktyvus] [žiūrėta 2011m. kovo 16d.]. Prieiga per internetą: < http://ik.su.lt/~kestzil/OMA/>.
25. *Mathematica Documentation Center* [interaktyvus] [žiūrėta 2010m. spalio 21d.]. Prieiga per internetą: <http://reference.wolfram.com/mathematica/guide/Mathematica.html>.
26. *IMSL® C Numerical Library V 8.0* [interaktyvus] [žiūrėta 2010m. spalio 21d.]. Prieiga per internetą: <http://www.roguewave.com/products/imsl-numerical-libraries/c-library/overview/whats-new.aspx>.
27. *Scilab* [interaktyvus] [žiūrėta 2010m. spalio 21d.]. Prieiga per internetą: <http://help.scilab.org/docs/5.3.3/en_US/section_46d43fa43588512dc7dc2c3d190b775b.html>.

28. *Database layout* [interaktyvus] [žiūrėta 2010m. spalio 21d.]. Prieiga per internetą: < http://www.mediawiki.org/wiki/Manual:Database_layout>.
29. *LAPACK - Linear Algebra PACKage* [interaktyvus] [žiūrėta 2012m. gegužės 21d.]. Prieiga per internetą: < http://www.netlib.org/lapack/>.
30. *Computational Routines* [interaktyvus] [žiūrėta 2012m. gegužės 23d.]. Prieiga per internetą: < http://www.netlib.org/lapack/lug/node37.html>.
31. *Usingclapack* [interaktyvus] [žiūrėta 2012m. gegužės 24d.]. Prieiga per internetą: < http://www.cs.rochester.edu/~bh/cs400/using_lapack.html>.

Anotacija

Valčiukas R. Internetinė matematinio programavimo ir modeliavimo sistema. Sistemos kūrimas ir testavimas. Doc. K. Žilinskas. – Šiaulių universitetas, Matematikos ir informatikos fakultetas, 2012. – 89 p.

Šio darbo pagrindinis tikslas yra suprojektuoti ir sukurti internetinę matematinio programavimo ir modeliavimo sistemą. Šiam tikslui pasiekti buvo nagrinėjama matematinio programavimo samprata. Atlikta panašių matematinių programavimo sistemų bei matematinių funkcijų bibliotekų, skirtų įvairioms programavimo kalboms, analizė. Identifikuojamos ir nagrinėjamos problemos, kurios iškilo kuriant internetinę matematinio programavimo ir modeliavimo sistemą. Taip pat šiai sistemai parašytos testinės programos su *C++, Java, Fortran* programavimo kalbomis bei *Netlib Repository LAPACK ir QT* bibliotekomis. Sukurta sistema palyginta su *Mathematica* ir *Scilab* matematinio programavimo sistemomis.

Pagrindiniai žodžiai: vikinomika, atviras kodas, matematinis programavimas, matematinis modeliavimas, internetinė matematinio programavimo sistema.

Summary

Valčiukas R. Online Mathematical Programming and Simulation System. The Implementation and Testing the System. Tutor: Doc. K. Žilinskas. – Šiauliai University, Faculty of Mathematics and Informatics, 2012. – 89 p.

The aim of this work is to design and develop web-based mathematical programming and simulation system. For this purpose were analyzed the concept of mathematical programming and performed analysis of similar mathematical programming systems and libraries of mathematical functions for various programming languages. Identified and analyzed the problems that arose in the development of an online mathematical programming and simulation system. Also, written test programs for created system in *C++*, *Java, Fortran* programming languages and *Netlib Repository LAPACK*, *QT* libraries. The developed system was compared with *Mathematica* and *Scilab* mathematical programming systems.

Key words: wikinomics, open source, mathematical programming, mathematical modeling, online mathematical programming system.

Priedai

1. DVD turinys

- Šakniniame kataloge yra elektroninės magistro darbo aprašymo versijos (rinkmenos *aprasymas.doc* ir *aprasymas.pdf*).
- Kataloge Internetine programavimo sistema rasite internetinės svetainės failus (katalogas Internetine svetaine), duomenų bazės failiuką (*duomenu_baze.sql*) bei serverinę programą (katalogas *Programa*). Taip pat rasite serverines programos projekto failus kataloge *Projektas*. Projektą galima atidaryti su *QT Creator* programa.
- Kataloge *Programine iranga* įdėta programinė įranga reikalinga internetinės matematinio programavimo ir modeliavimo sistemos veikimui bei programinio kodo peržiūrai.
- Kataloge *Diagramos* įdėtos visos su projektu susijusios diagramos.
- Kataloge *Testines programos* įdėtos testinių programų kodas.

2. Internetinės matematinio programavimo ir modeliavimo sistemos testavimas

Testavimo programas galite rasti internetinėje matematinio programavimo ir modeliavimo sistemoje adresais http://78.157.93.221/Stochastinis_programavimas ir http://78.157.93.221/Testai.

2.1. C++ programavimo kalba parašytos programos kompiliavimas

2.1.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai iš internetinės matematinio programavimo ir modeliavimo sistemos bei jos rezultato gražinimą vartotojui.
Testas: kompiliuojama *C++* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Cpp_programa.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos.
Žinomas rezultatas: programos vykdomasis bei programinio kodo failai turi susikurti serveryje, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: programos vykdomasis ir programinio kodo failas sukurtas serveryje bei vartotojui gražinta *null* reikšmė, kas reiškia, kad kompiliavimas įvyko sėkmingai. Jeigu programa dar kompiliuojama arba laukia eilėje, o vartotojas ieško tos programos kompiliavimo rezultatų, pranešama jog kompiliavimas nėra baigtas ir nurodoma kada jis buvo pradėtas.

2.1.2. Programa su *Netlib Repository Lapack* biblioteka

Aprašymas: programa skirta išbandyti *Netlib Repository Lapack* bibliotekos metodų panaudojimą kuriamose programose. Šioje programoje išbandysime *DGESV* metodą, kuriuo skaičiuojamas tiesinių lygčių sistemos.

Testas: kompiliuojama *C++* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Cpp_programa_lapack.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos.

Žinomas rezultatas: programos vykdomasis bei programinio kodo failai turi susikurti serveryje, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: programos vykdomasis ir programinio kodo failas sukurtas serveryje bei vartotojui gražinta *null* reikšmė, kas reiškia, kad kompiliavimas įvyko sėkmingai. Jeigu programa dar kompiliuojama arba laukia eilėje, o vartotojas ieško tos programos kompiliavimo rezultatų, pranešama jog kompiliavimas nėra baigtas ir nurodoma kada buvo pradėtas.

2.1.3. Programa su sintaksės klaidomis

Aprašymas: programos kode yra padarytos klaidos, tuom siekiama išsiaiškinti kaip internetinė matematinio programavimo ir modeliavimo sistema reaguos į tokios programos kompiliavimą.

Testas: kompiliuojama *C++* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Cpp_programa_error.cpp)*, kurioje padarytos trys sintaksės klaidos.

Žinomas rezultatas: kompiliatorius parodo kompiliavimo klaidas.

Gautas rezultatas: vartotojui išvedamos klaidos ir nurodomos eilutės kur yra klaidos:

*/480/480.cpp: In function "int main(int, char**)":*

/480/480.cpp:11: error: invalid operands of types "const char [8]" and "int" to binary "operator<<"

/480/480.cpp:13: error: "ifstreamc" was not declared in this scope

/480/480.cpp:13: error: expected ";" before "myReadFile"

/480/480.cpp:25: error: "myReadFile" was not declared in this scope

/480/480.cpp:49: error: expected "}" at end of input

2.2. C++ programavimo kalba parašytos programos vykdymas

2.2.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai bei jos rezultato gražinimą vartotojui.

Testas: vykdomai programai (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Cpp_programa.cpp*) perduodami parametrai: skaičius 5, paprastas tekstas „testinė programa“, didelės apimties tekstas bei tekstinis failas (esantis *DVD* diske kataloge *Testines programos* failas *text.txt*)

Žinomas rezultatas: išvedamos ta pačia eilės tvarka išvardintų parametrų reikšmės.

Gautas rezultatas: vartotojui gražintos parametrų reikšmės ta pačia eilės tvarka, kokia buvo perduoti parametrai. Pastebėta, kad atsakymas gražintas be lietuviškų raidžių.

2.2.2. Programa su *Netlib Repository Lapack* biblioteka

Aprašymas: programa skirta išbandyti *Netlib Repository Lapack* bibliotekos metodų panaudojimą kuriamose programose. Šioje programoje išbandysime *DGESV* metodą, kuriuo skaičiuojamos tiesinių lygčių sistemos Ax=b. Metode *dgesv_(const int *N, const int *nrhs, double *A, const int *lda, int *ipiv, double *b, const int *ldb, int *info)* naudojami parametrai:

N	A matricos stulpelių skaičius
nrhs	b matricos stuleplių skaičius
lda	A matricos eilučių skaičius
ipiv	Sukimosi indeksai
ldb	b matricos eilučių skaičius

Testas: vykdoma *C++* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Cpp_programa_lapack.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos, kur A 3x3 matrica lygi:

76 25 11

27 89 51

18 60 32,

o b =10, 7, 43.

Žinomas rezultatas: Turime gauti skaičius -0.661082 9.456125 -16.014625.

Gautas rezultatas: programa įvykdyta sėkmingai ir gautas rezultatas yra -0.661082 9.456125 -16.014625.

2.2.3. Programa su atminties klaida

Aprašymas: programoje netaisiklingai modifikuojamas atminties turinys, kuris priverčia programą baigti darbą neatlikus visų skaičiavimų iki galo.

Testas: vygdoma programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Cpp_programa_error2.cpp).*

Žinomas rezultatas: programa nustoja veikti operacinėje sistemoje, tačiau išvedą į ekraną vieną žodį *start* per komandinę eilutę.

Gautas rezultatas: programa nustoja veikusi toje vietoje, kurioje yra klaida. Vartotojui gražitnas vienas žodis *start*.

2.3. C++ programavimo kalba su QT biblioteka parašytos programos kompiliavimas

2.3.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai iš internetinės matematinio programavimo ir modeliavimo sistemos bei jos rezultato gražinimą vartotojui.

Testas: kompiliuojama *C++* programa panaudojant *QT* biblioteką (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Qt_programa.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos.

Žinomas rezultatas: serveryje turi susikurti du katalogai *debug* ir *release* bei projektinis ir programinio kodo failai. *Release* kataloge turi susikurti vykdomasis programos failas, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: serveryje sukurti *debug* ir *release* katalogai, projektinis, programos vykdomasis ir programinio kodo failai bei vartotojui gražinta *null* reikšmė, kas reiškia kad kompiliavimas įvyko sėkmingai. Jeigu programa dar kompiliuojama arba laukia eilėje, o

vartotojas ieško tos programos kompiliavimo rezultatų, pranešama jog kompiliavimas nėra baigtas ir nurodoma kada buvo pradėtas.

2.3.2. Programa su sintaksės klaidomis

Aprašymas: programos kode yra padarytos klaidos, tuom siekiama išsiaiškinti kaip internetinė matematinio programavimo ir modeliavimo sistema reaguos į tokios programos kompiliavimą.

Testas: kompiliuojama *C++* programa panaudojant *QT* biblioteką (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Qt_programa_error.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos. Programoje yra padarytos trys sintaksės klaidos.

Žinomas rezultatas: kompiliatorius parodo kompiliavimo klaidas.

Gautas rezultatas: vartotojui išvedamos klaidos ir nurodomos eilutės kur jos yra:

*489.cpp: In function "int main(int, char**)":*
489.cpp:21: error: expected "," or ";" before "i"
489.cpp:21: error: expected ";" before ")" token
489.cpp:35: error: expected ";" before "QTextStream"
489.cpp:37: error: "in" was not declared in this scope
489.cpp:49: error: expected "}" at end of input
*mingw32-make[1]: *** [release/489.o] Error 1*
*mingw32-make: *** [release] Error 2*

2.3.3. Nuosekli ir lygiagreti programa realizuojanti Monte Karlo metodą

Aprašymas: Monte Karlo metodą realizuojančios nuoseklioji ir lygiagrečioji programos.

Testas: kompiliuojama nuosekli *C++* programa su *QT* biblioteka (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos/monte_karlo_metodas* faile *nuosekli.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos.

Žinomas rezultatas: serveryje turi susikurti du katalogai *debug* ir *release* bei projektinis ir programinio kodo failai. *Release* kataloge turi susikurti vykdomasis programos failas, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: serveryje sukurti debug ir release katalogai, projektinis, programos vykdomasis ir programinio kodo failai bei vartotojui gražintos kompiliavimo pastabos:

490.cpp: In function "double Chi(double, double, int)":
490.cpp:977: warning: suggest explicit braces to avoid ambiguous "else"

490.cpp:957: warning: suggest explicit braces to avoid ambiguous "else"

490.cpp: In function "double Fish(int, int, double)":

490.cpp:1121: warning: suggest parentheses around "&&" within "||"

490.cpp: In function "void Projector(QVector<QVector<double> >, QVector<QVector<double> >&, QVector<int>, QVector<double>, QVector<double>&, QVector<double>, int, int, unsigned char&)":

490.cpp:2959: warning: zero-length ms_printf format string

490.cpp: In function "double Fish(int, int, double)":

490.cpp:1067: warning: "fish_r" may be used uninitialized in this function

490.cpp: In function "double Statist(QVector<QVector<double> >, QVector<QVector<double> >&, QVector<double>, QVector<int>, int&, int, int, unsigned char&)":

490.cpp:1553: warning: "ass" may be used uninitialized in this function

490.cpp: In function "void simplex(int, int, QVector<QVector<double> >&, QVector<QVector<double> >&, QVector<double>&, QVector<double>&, double&, bool&)":

490.cpp:1867: warning: "seil" may be used uninitialized in this function

Testas: kompiliuojama lygiagreti *C++* programa su *QT* biblioteka (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos/monte_karlo_metodas* faile *lygegreti.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos.

Žinomas rezultatas: serveryje turi susikurti du katalogai *debug* ir *release* bei projektinis ir programinio kodo failai. *Release* kataloge turi susikurti vykdomasis programos failas, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: serveryje sukurti *debug* ir *release* katalogai, projektinis, programos vykdomasis ir programinio kodo failai bei vartotojui gražintos kompiliavimo pastabos:

*491.cpp: In function "int main(int, char**)":*

491.cpp:303: warning: unused variable "ij"

491.cpp: In function "double Chi(double, double, int)":

491.cpp:1145: warning: suggest explicit braces to avoid ambiguous "else"

491.cpp:1125: warning: suggest explicit braces to avoid ambiguous "else"

491.cpp: In function "double Fish(int, int, double)":

491.cpp:1289: warning: suggest parentheses around "&&" within "||"

491.cpp: In function "void Projector(QVector<QVector<double> >, QVector<QVector<double> >&, QVector<int>, QVector<double>, QVector<double>&, QVector<double>, int, int, unsigned char&)":
491.cpp:3843: warning: zero-length ms_printf format string
491.cpp: In function "double Fish(int, int, double)":
491.cpp:1235: warning: "fish_r" may be used uninitialized in this function
491.cpp: In function "double Statist(QVector<QVector<double> >, QVector<QVector<double> >&, QVector<double>, QVector<int>, int&, int, int, unsigned char&)":
491.cpp:1721: warning: "ass" may be used uninitialized in this function
491.cpp: In function "void simplex(int, int, QVector<QVector<double> >&, QVector<QVector<double> >&, QVector<double>&, QVector<double>&, double&, bool&)":
491.cpp:2035: warning: "seil" may be used uninitialized in this function
491.cpp: In function "grazinama lygiagreciai(const _Padavimas&)":
491.cpp:2731: warning: "seil" may be used uninitialized in this function

2.4. C++ programavimo kalba su QT biblioteka parašytos programos vykdymas

2.4.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai bei jos rezultato gražinimą vartotojui.

Testas: vykdomai programai (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Qt_programa.cpp*) perduodami parametrai: skaičius 5, paprastas tekstas „testinė programa“, didelies apimties tekstas bei tekstinis failas (esantis *DVD* diske kataloge *Testines programos* failas *text.txt*)

Žinomas rezultatas: išvedamos ta pačia eilės tvarka išvardintų parametrų reikšmės.

Gautas rezultatas: vartotojui gražintos parametrų reikšmės ta pačia eilės tvarka, kokia buvo perduoti parametrai. Pastebėta, kad atsakymas gražintas be lietuviškų raidžių.

2.4.2. Programa su atminties klaida

Aprašymas: programoje netaisiklingai modifikuojamas atminties turinys, kuris priverčia programą baigti darbą neatlikus visų skaičiavimų iki galo.

Testas: bandoma vykdyti programa su klaida (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Qt_programa_error2.cpp).*

Žinomas rezultatas: programa nustoja veikti operacinėje sistemoje, tačiau išvedą į ekraną vieną žodį *start* per komandinės eilutės aplinką.

Gautas rezultatas: programa nustoja veikusi toje vietoje, kurioje yra klaida. Programa vartotojui gražino vieną žodį *start*.

2.4.3. Nuosekli ir lygiagreti programa realizuojanti Monte Karlo metodą

Aprašymas: Monte Karlo metodą realizuojančios nuoseklioji ir lygiagrečioji programos.

Testas 1: kompiliuojama nuosekli C++ programa su *QT* biblioteka (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos/monte_karlo_metodas* faile *nuosekli.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos. Parametrai:

Iteracijų skaičius kurie skaičiuojami lygiagrečiai (turis)=100,

1 etapo kintamųjų skaičius (mm) = 20,

2 etapo kintamųjų skaičius (nm) = 30,

1 etapo ribojimų skaičius (nt) = 10,

Didžiausias iteracijų skaičius (iter) = 1000,

Pradinis iteracijų skaičius (nn) = 200,

Mažiausias imties tūris (Lmin) = 200,

Didžiausias imties tūris (Lmax) = 1000000,

Norimo tikslumo skaičius (eps) = 2 (norimas tikslumas).

Failas su testiniais duomenimis (jį galite rasti *DVD* diske kataloge *Testines programos/monte_karlo_metodas* faile *testiniai_duomenys.txt).* Testas atliekamas kompiuteryje, kuris turi *Intel i5* keturių branduolių, 3.2 GHZ spartos procesorių.

Žinomas rezultatas: Turi būti rastas optimalus sprendinys.

Gautas rezultatas: su duotais ribojimais rastas optimalus sprendinys:

---

point:

0.0000 0.0000 0.4400 0.1062 0.0271 0.0000 0.0000 0.2997 0.0279 0.1035 0.0000 0.1798 0.0000 0.7306 0.5624 0.2862 0.3098 0.0000 0.1261 0.3532

---

gradient:

19.8769 57.3716 -20.2576 41.6075 69.9335 45.2892 -5.8955 -36.6334 26.9153 81.1171 14.2747 25.6756 26.4356 10.2409 -9.6425 -9.7744 -5.3548 6.2582 -0.8265 0.5901

---

projection of gradient:

0.0000 0.0000 3.3160 -0.7863 1.2113 0.0000 0.0000 3.3835 -1.0260 1.9287 0.0000 1.6648 0.0000 -2.1280 -3.6547 -0.1651 -1.8011 0.0000 -1.8466 1.7538

Nt=24 FF= 181.990+- 1.998 NN=413 Lsum=5504 TT= 13.33 as= 0.646

FINISH ! (13.33) ,Sugaistas laikas: 2,253 sec.

Testas 2: kompiliuojama lygiagreti *C++* programa su *QT* biblioteka (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos/monte_karlo_metodas* faile *lygegreti.cpp)* iš internetinės matematinio programavimo ir modeliavimo sistemos. Parametrai:

Iteracijų skaičius kurie skaičiuojami lygiagrečiai (turis)=1000,

1 etapo kintamųjų skaičius (mm) = 20,

2 etapo kintamųjų skaičius (nm) = 30,

1 etapo ribojimų skaičius (nt) = 10,

Didžiausias iteracijų skaičius (iter) = 10000,

Pradinis iteracijų skaičius (nn) = 200,

Mažiausias imties tūris (Lmin) = 200,

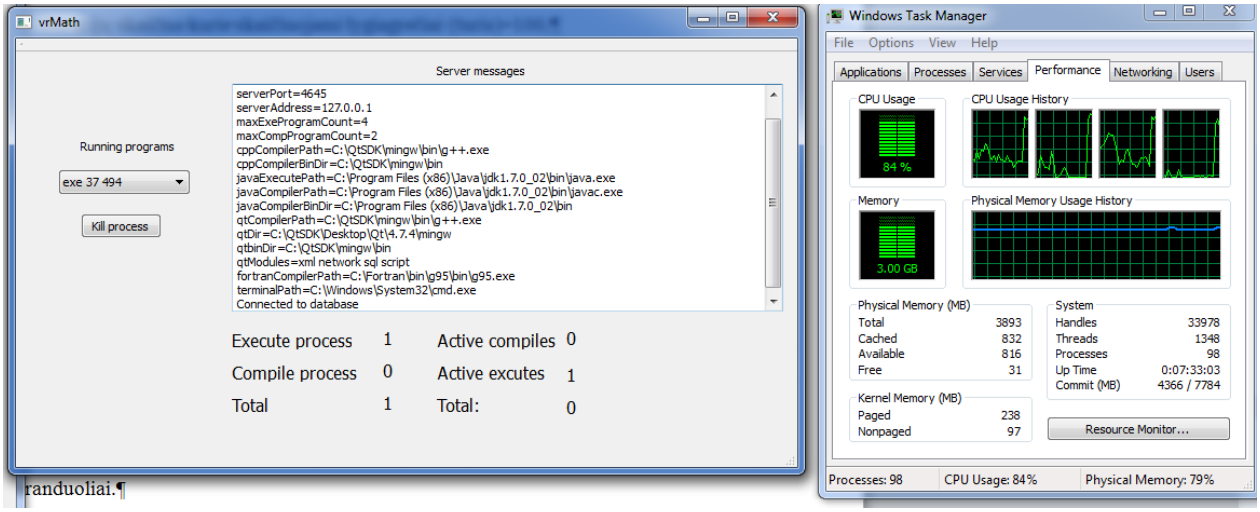
Didžiausias imties tūris (Lmax) = 1000000,

Norimo tikslumo skaičius (eps) = 2 (norimas tikslumas).

Failas su testiniais duomenimis. (jį galite rasti *DVD* diske kataloge *Testines programos/monte_karlo_metodas* faile *testiniai_duomenys.txt).* Testas atliekamas kompiuteryje, kuris turi *Intel i5* keturių branduolių, 3.2 GHZ spartos procesorių.

Žinomas rezultatas: Turi būti rastas optimalus sprendinys ir šiam darbui turi būti išnaudojami visi branduoliai.

Gautas rezultatas:



randuoliai.¶

Skaičiavimai buvo atlikti su keturiais branduoliais, tai matyti paveikslėlyje, bei su duotais ribojimais rastas optimalus sprendinys:

---

point:

0.0000 0.0000 0.4376 0.1066 0.0263 0.0000 0.0000 0.2971 0.0288 0.1024 0.0000 0.1777 0.0000 0.7327 0.5675 0.2857 0.3118 0.0000 0.1277 0.3517

---

gradient:

19.9094 57.2775 -20.7059 41.3926 61.2961 48.7636 -7.4542 -36.7200 27.6559 71.8236 14.7125 25.1733 23.5172 12.2419 -5.6129 -9.6094 -2.5322 6.1185 -1.2813 2.1482

---

projection of gradient:

0.0000 0.0000 0.0135 -0.1160 0.0879 0.0000 0.0000 0.2818 0.0204 -0.0465 0.0000 -0.0227 0.0000 0.1632 -1.2065 -0.4502 -0.4370 0.0000 -0.4256 0.4801

Nt=32 FF= 180.842+- 1.263 NN=1000 Lsum=32000 TT= 32.00 as= 0.985

FINISH ! (32.00) Sugaistas laikas: 7,847 sec.

2.5. Java programavimo kalba parašytos programos kompiliavimas

2.5.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai iš internetinės matematinio programavimo ir modeliavimo sistemos bei jos rezultato gražinimą vartotojui.
Testas: kompiliuojama *Java* programa (jos programinį kodą galite rasti DVD diske kataloge Testines programos faile *java_programa.java*) iš internetinės matematinio programavimo ir modeliavimo sistemos.
Žinomas rezultatas: programos vykdomasis bei programinio kodo failai turi susikurti serveryje, o vartotojui pranešta jog kompiliavimo klaidų nėra.
Gautas rezultatas: programos vykdomasis ir programinio kodo failas sukurtas serveryje bei vartotojui gražinta *null* reikšmė, kas reiškia, kad kompiliavimas įvyko sėkmingai. Jeigu programa dar kompiliuojama arba laukia eilėje, o vartotojas ieško tos programos kompiliavimo rezultatų, pranešama, jog kompiliavimas nėra baigtas ir nurodoma kada jis buvo pradėtas.

2.5.2. Programa su sintaksės klaidomis

Aprašymas: programos kode yra padarytos klaidos, tuom siekiama išsiaiškinti kaip internetinė matematinio programavimo ir modeliavimo sistema reaguos į tokios programos kompiliavimą.
Testas: kompiliuojama *Java* programa (jos programinį kodą galite rasti DVD diske kataloge *Testines programos* faile *Java_programa_error.java*), kurioje padarytos trys sintaksės klaidos.
Žinomas rezultatas: kompiliatorius parodo kompiliavimo klaidas.
Gautas rezultatas: vartotojui išvedamos klaidos ir nurodomos eilutės kur jos yra:

```
.java:9: error: "{" expected
    try
      ^

.java:61: error: ";" expected
      bufRead.close()
                   ^

.java:65: error: "catch" without "try"
    }catch (ArrayIndexOutOfBoundsException e){
```

^

.java:65: error: ")" expected

}catch (ArrayIndexOutOfBoundsException e){

^

.java:65: error: not a statement

}catch (ArrayIndexOutOfBoundsException e){

^

.java:65: error: ";" expected

}catch (ArrayIndexOutOfBoundsException e){

^

.java:77: error: "catch" without "try"

}catch (IOException e){

^

.java:77: error: ")" expected

}catch (IOException e){

^

.java:77: error: not a statement

}catch (IOException e){

^

.java:77: error: ";" expected

}catch (IOException e){

^

.java:9: error: "try" without "catch", "finally" or resource declarations

try

^

.java:89: error: reached end of file while parsing

}

^

12 errors

2.6. Java programavimo kalba parašytos programos vykdymas

2.6.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai bei jos rezultato gražinimą vartotojui.

Testas: vykdomai programai (jos programinį kodą galite rasti DVD diske kataloge *Testines programos* faile *Java_programa.java*) perduodami parametrai: skaičius 5, paprastas tekstas „testinė programa“, didelies apimties tekstas bei tekstinis failas (esantis DVD diske kataloge *Testines programos* failas *text.txt*).

Žinomas rezultatas: išvedamos ta pačia eilės tvarka išvardintų parametrų reikšmės.

Gautas rezultatas: vartotojui gražintos parametrų reikšmės ta pačia eilės tvarka, kokia buvo perduoti parametrai. Pastebėta, kad atsakymas gražintas be lietuviškų raidžių.

2.7. Fortran programavimo kalba parašytos programos kompiliavimas

2.7.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai iš internetinės matematinio programavimo ir modeliavimo sistemos bei jos rezultato gražinimą vartotojui.

Testas: kompiliuojama *Fortran* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *fortran95_programa.f*) iš internetinės matematinio programavimo ir modeliavimo sistemos.

Žinomas rezultatas: programos vykdomasis bei programinio kodo failai turi susikurti serveryje, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: programos vykdomasis ir programinio kodo failas sukurtas serveryje bei vartotojui gražinta *null* reikšmė, kas reiškia, kad kompiliavimas įvyko sėkmingai. Jeigu programa dar kompiliuojama arba laukia eilėje, o vartotojas ieško tos programos kompiliavimo rezultatų, pranešama jog kompiliavimas nėra baigtas ir nurodoma kada jis buvo pradėtas.

2.7.2. Programa su Netlib Repository Lapack biblioteka

Aprašymas: programa skirta išbandyti *Netlib Repository Lapack* bibliotekos metodų panaudojimą programose. Šioje programoje išbandysime *DGESV* metodą, kuriuo skaičiuojamas tiesinių lygčių sistemos.

Testas: kompiliuojama *Fortran* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Fortran_programa_lapack.f*) iš internetinės matematinio programavimo ir modeliavimo sistemos.

Žinomas rezultatas: programos vykdomasis bei programinio kodo failai turi susikurti serveryje, o vartotojui pranešta jog kompiliavimo klaidų nėra.

Gautas rezultatas: programos vykdomasis ir programinio kodo failas sukurtas serveryje bei vartotojui gražinta null reikšmė, kas reiškia, kad kompiliavimas įvyko sėkmingai. Jeigu programa dar kompiliuojama arba laukia eilėje, o vartotojas ieško tos programos kompiliavimo rezultatų, pranešama jog kompiliavimas nėra baigtas ir nurodoma kada buvo pradėtas.

2.7.3. Programa su sintaksės klaidomis

Aprašymas: programos kode yra padarytos klaidos, tuom siekiama išsiaiškinti kaip internetinė matematinio programavimo ir modeliavimo sistema reaguos į tokios programos kompiliavimą.

Testas: kompiliuojama *Fortran* programa (jos programinį kodą galite rasti *DVD* diske kataloge *Testines programos* faile *Fortran_programa_error.f*), kurioje padarytos trys sintaksės klaidos.

Žinomas rezultatas: kompiliatorius parodo kompiliavimo klaidas.

Gautas rezultatas: vartotojui išvedamos klaidos ir nurodomos eilutės kur yra klaidos:

In file /488/488.f:8

END D
1

Error: Expecting END DO statement at (1)
In file /488/488.f:9

END PROGRAMs
1

Error: Expecting END DO statement at (1)
Error: Unexpected end of file in "/488/488.f"

2.8. Fortran programavimo kalba parašytos programos vykdymas

2.8.1. Duomenų įvedimo ir jų išvedimo programa

Aprašymas: programa skirta ištestuoti duomenų perdavimą pačiai programai bei jos rezultato gražinimą vartotojui.

Testas: vykdomai programai (jos programinį kodą galite rasti DVD diske kataloge Testines programos faile *Fortran_programa.f*) perduodami parametrai: skaičius 5, paprastas tekstas „testinė programa“, didelės apimties tekstas bei tekstinis failas (esantis DVD diske kataloge Testines programos failas text.txt)

Žinomas rezultatas: išvedamos ta pačia eilės tvarka išvardintų parametrų reikšmės.

Gautas rezultatas: vartotojui gražintos parametrų reikšmės ta pačia eilės tvarka, kokia buvo perduoti parametrai. Pastebėta, kad atsakymas gražintas be lietuviškų raidžių.

2.8.2. Programa su Netlib Repository Lapack biblioteka

Aprašymas: programa skirta išbandyti *Netlib Repository Lapack* bibliotekos metodų panaudojimą programose. Šioje programoje išbandysime *DGESV* metodą, kuriuo skaičiuojamas tiesinių lygčių sistemos Ax=b.

Testas: vykdoma *Fortran* programa (jos programinį kodą galite rasti *DVD* diske kataloge Testines programos faile *Fortran_programa_lapack.f*) iš internetinės matematinio programavimo ir modeliavimo sistemos. Programos aprašymą rasite adresu http://faculty.washington.edu/rjl/uwamath583s11/sphinx/notes/html/lapack_examples.html

Gautas rezultatas: programa įvykdyta sėkmingai ir gautas rezultatas yra :

x computed rel. error

0.699951D+00 0.699951D+00 0.000000D+00

3. Programos vartotojo sąsajos kūrimas

Kuriant programos vartotojo sąsają programai pastebėta, jog programos parametrų pavadinimai turi būti skirtingi. Kitu atveju programai bus perduodamas paskutinis parametras tuo pačiu pavadinimu.

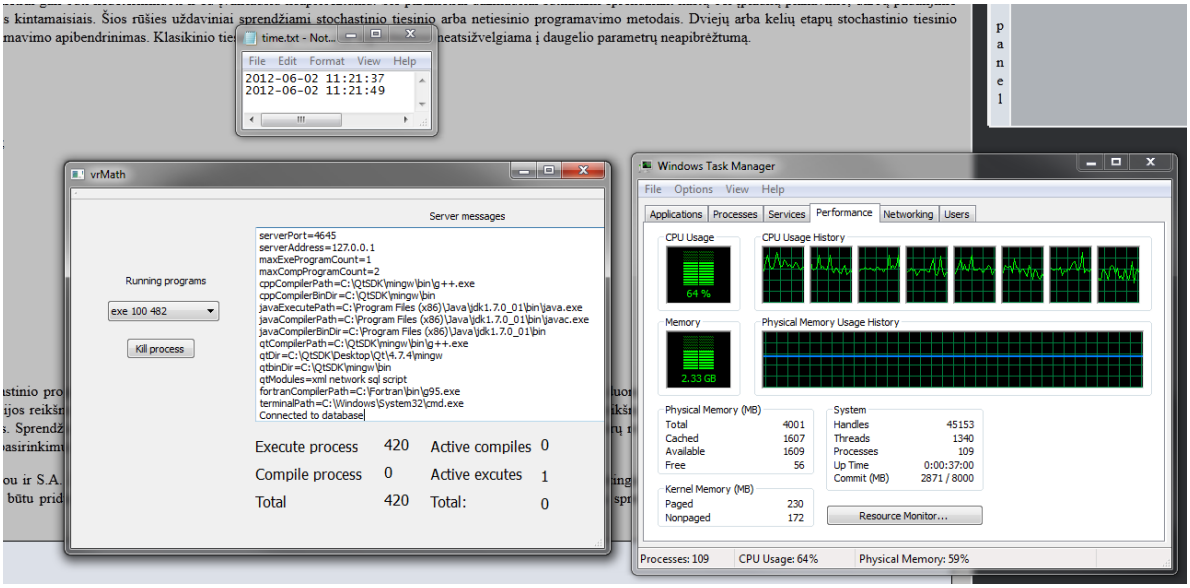
4. Sistemos apkrovos testas

Sistemos apkrovos testą atkliksime su C++ programavimo kalba ir QT biblioteka parašyta programa, kuri realizuoja Monte Karlo metodą. Naudosime lygiagrečią programą, nes ji yra reiklesnė kompiuterio resursams. Eksperimentus atliksime su kompiuteriu, kuriame yra keturių

branduolių Intel i7 procesorius, kurio taktinis dažnis 2,2 GHZ. Taip pat jis turi 4GB operatyvinės atminties. Eksperimento metu bandysime nustatyti, kiek daugiausiai vartotojų serverinė programa gali aptarnauti. Eksperimento metu imituosime 500 vartotojų, bandančius paleisti programą su Monte Karlo metodu. Eksperimento laiko trukmę nustatysime pagal scenarijaus paleidimo pradžią ir jo vykdymo pabaigą.

Testas 1: serverinė programa nustatyta taip, kad ji galėtų paleisti tik vieną programą vienu metu.

Rezultatas:

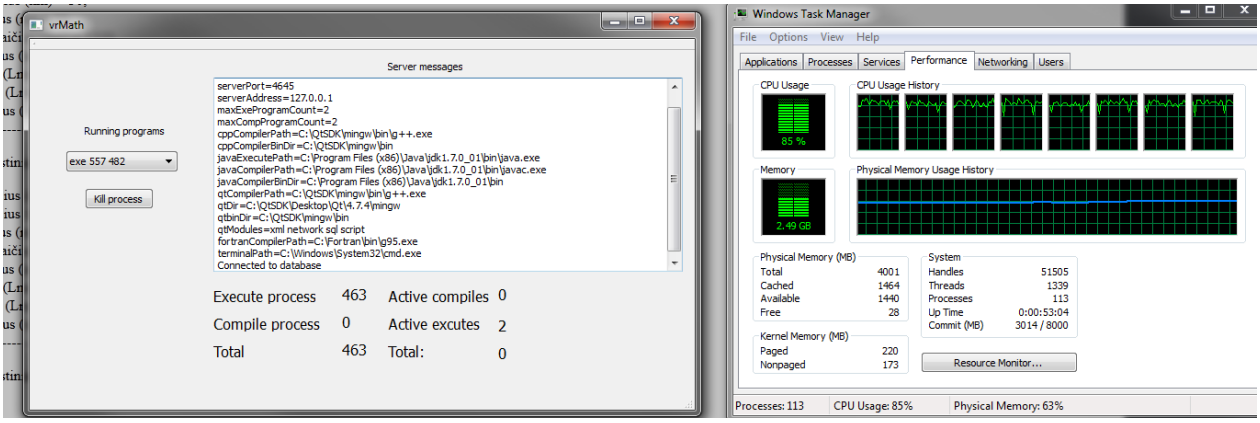


Pav 15. Serverinės programos darbas.

Paleidus scenarijų, kuris imituoja 500 vartotojų užklausų vienu metu į serverį, kuriame serverinė programa (žr. pav. 15) apdoroja duomenis ir paleidžia programas. Joje matome, kad užduočių skaičius (*Execute process*) yra didelis. bet serverinė programa jas vykdo po vieną (*Active executes*). Eksperimento metu visiems imituojamiems vartotojams žymės buvo išsiųstos per 12 sekundžių, nes tiek laiko vykdėsi scenarijus. O visas 500 užduočių jis įvykdė per 14 minučių ir 21 sekundę. Tai reiškia, kad paskutinis vartotojas turėjo laukti 12 sekundžių, kol gavo žymę pagal kurią galės rasti atsakymą. Nesunku paskaičiuoti, kad programa per vieną sekundę tokiame serveryje gali aptarnauti apytiksliai 46 vartotojus. Penkioliktame paveikslėlyje taip pat matome, kad serveris išnaudoja tik 64% procesoriaus galios. Tai reiškia, kad jį apkrauti galime ir daugiau.

Testas 2: serverinė programa nustatyta taip, kad ji galėtų paleisti dvi programas vienu metu.

Rezultatas:

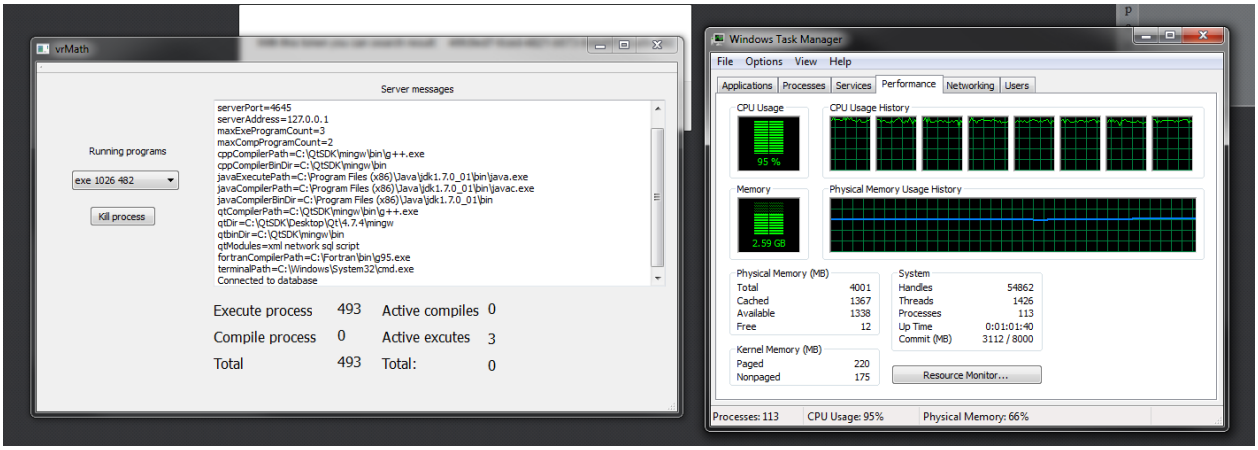


Pav 16. Serverinės programos darbas.

Šio eksperimento metu visos užduotys buvo įvykdytos per 9 minutes ir 41 sekundę, o vartotojai aptarnauti per 16 sekundžių. Serverio procesoriaus galia išnaudota 85% (žr. pav. 16).

Testas 3: serverinė programa nustatyta taip, kad ji galėtų paleisti tris programas vienu metu.

Rezultatas:

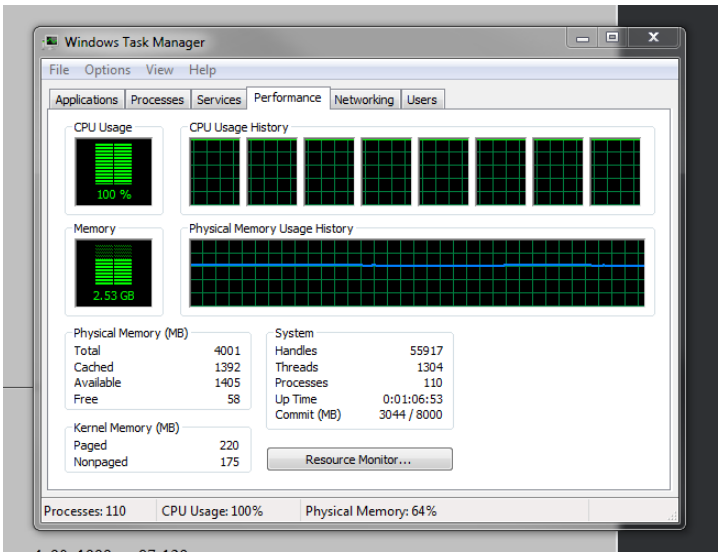


Pav 17. Serverinės programos darbas.

Šio eksperimento metu visos užduotys buvo įvykdytos per 9 minutes ir 51 sekundę, o vartotojai aptarnauti per 22 sekundžių. Serverio procesoriaus galia išnaudota 95% (žr. Pav. 17).

Testas 4: serverinė programa nustatyta taip, kad ji galėtų paleisti keturias programas vienu metu.

Rezultatas:

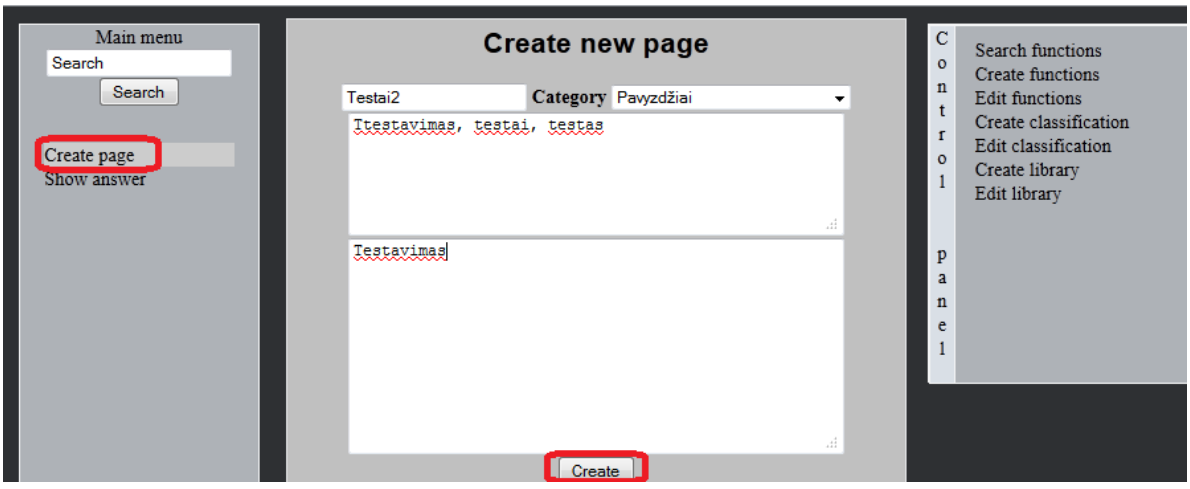


Pav 18. Serverinės programos darbas.

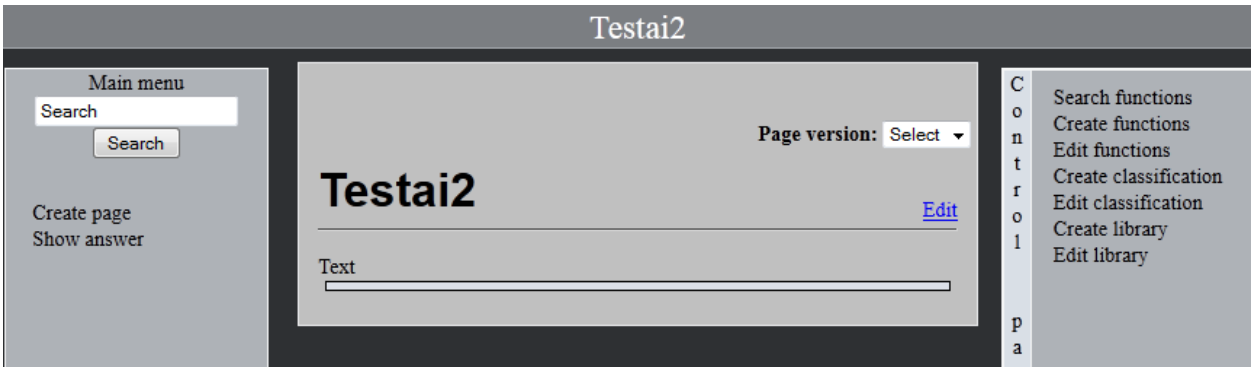
Šio eksperimento metu sistemos darbas buvo sutrikdytas, nes serverio procesoriaus apkrova pasiekė 100%. Dėl to labai sulėtėjo operacinės sistemos darbas ir eksperimentas buvo nutrauktas, nors serverinė programa vis tiek dar atsakė į užklausas tačiau labai lėtai.

5. Internetinio puslapio sukūrimas matematinio programavimo uždaviniui spręsti.

Paspaudus meniu punktą *Create page* pagrindiniame internetinės svetainės meniu pasirodo įvedimo formą, o ją užpildžius ir įvedus duomenis, paspaudus mygtuką *Create*, sukuriamas tuščias internetinis puslapis su nurodyta tema.

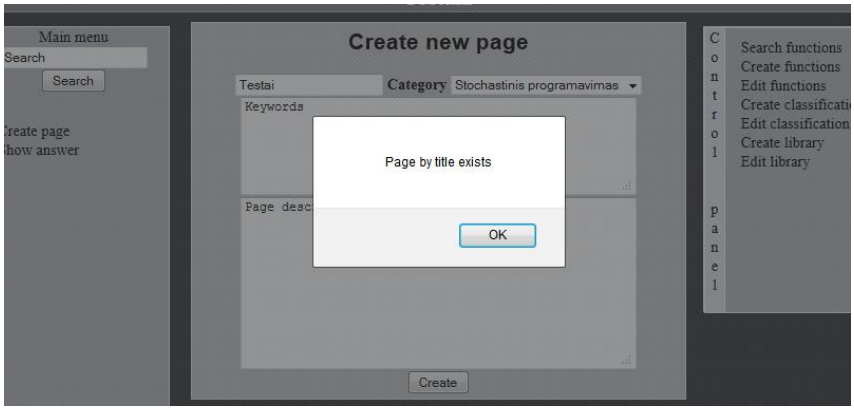


Pav 19. Puslapio kūrimas



Pav 20. Sukurtas puslapis

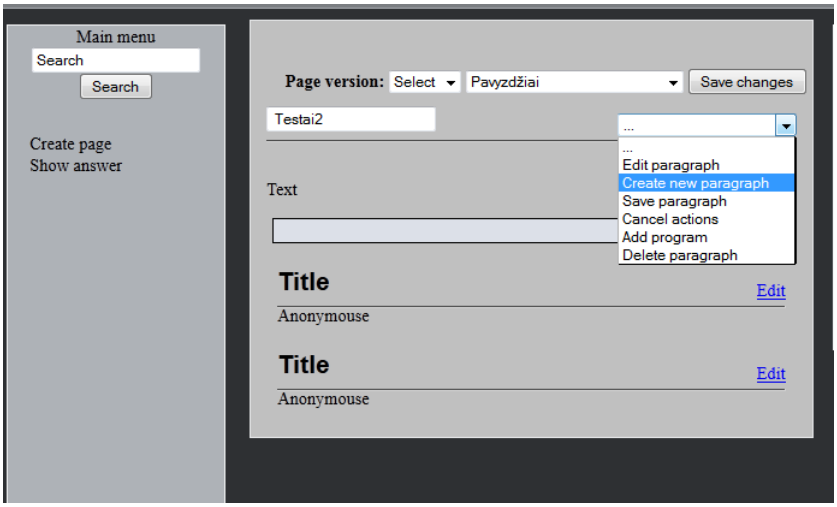
Jeigu įvedame jau esančią temą internetinėje svetainėje ir bandome sukurti iš naujo gauname pranešimą, kad toks internetinis puslapis su tokia tema jau egzistuoja.



Pav 21. Klaidos pranešimas

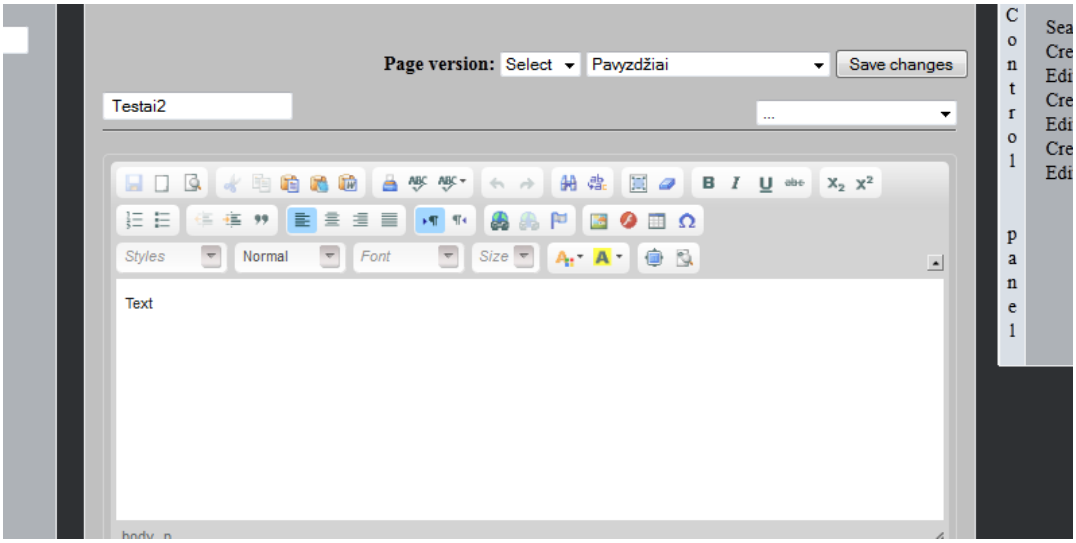
6. Internetinio puslapio turinio kūrimas ir išsaugojimas.

Pasirinkus iš veiksmų sąrašo punktą *Create new paragraph* sukuriamas naujas paragrafas su antrašte *Title*.



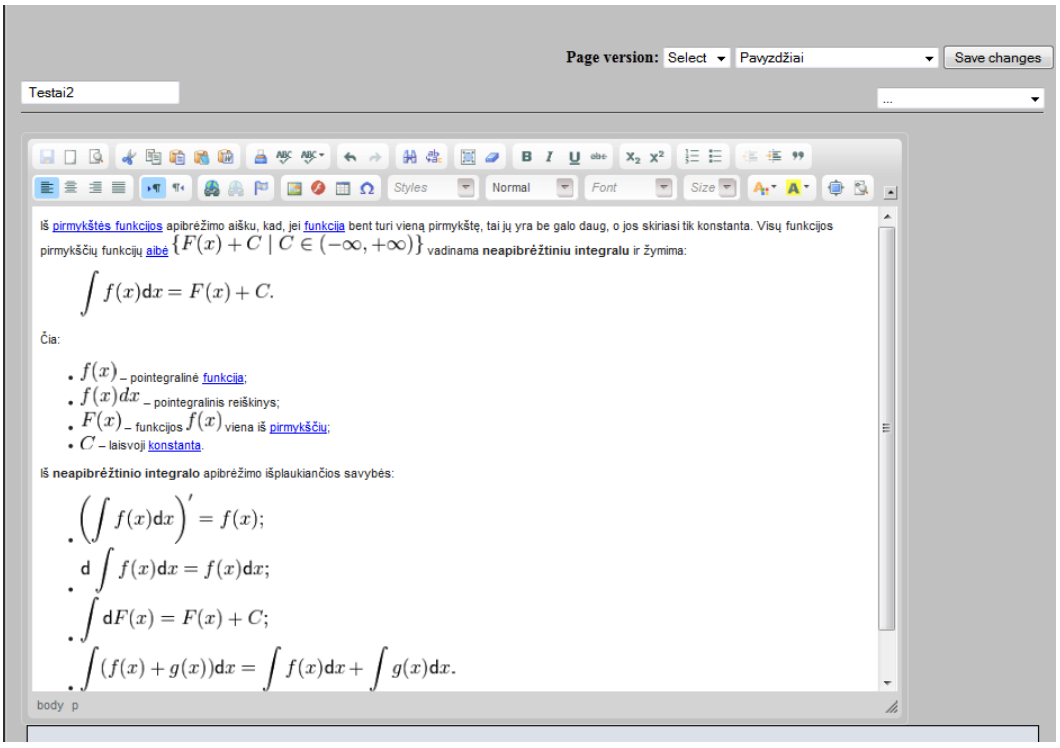
Pav 22. Kuriamas naujas paragrafas

Pasirinkus veiksmą *Edit paragraph* iškviečiamas teksto redaktorius su buvusiu tekstu paragrafe.



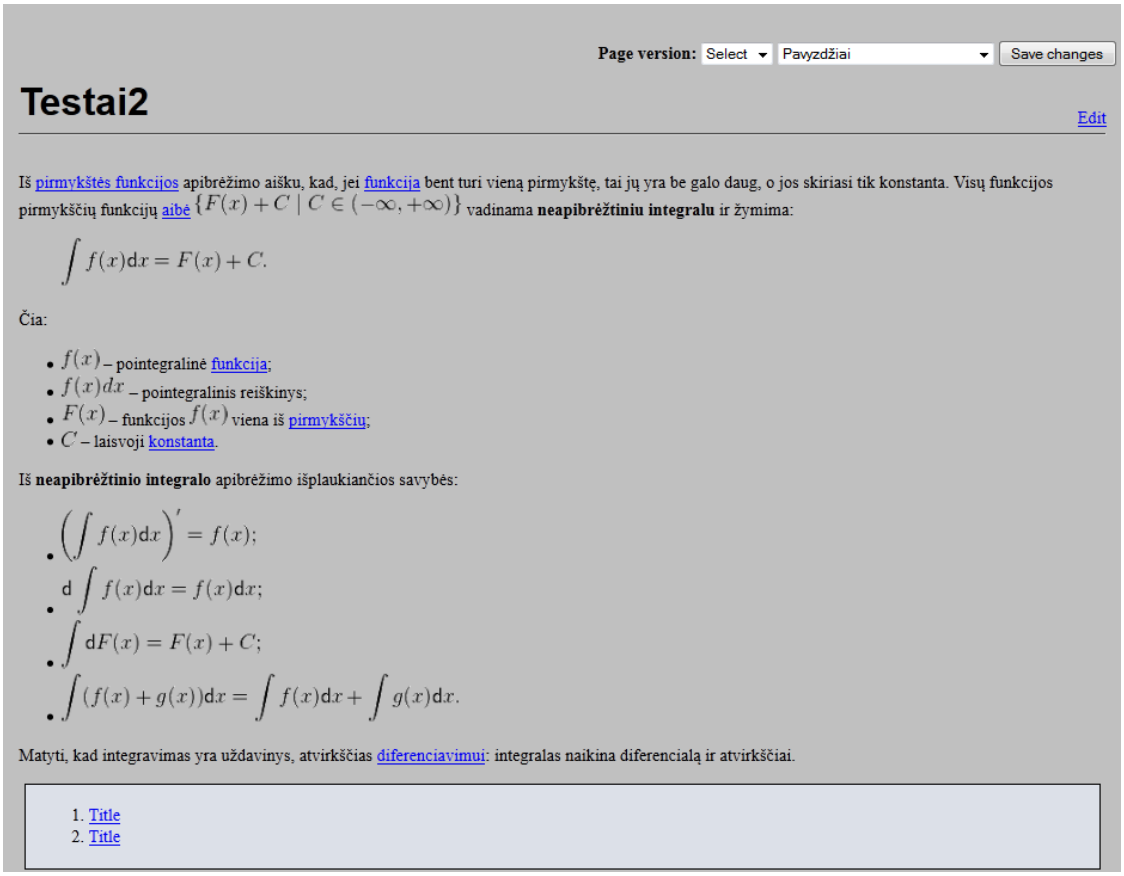
Pav 23. Iškviečiamas tekstinis redaktorius

Įrašius jame matematines formules, paveiksliukus, paprastą ir formatuotą tekstą bei



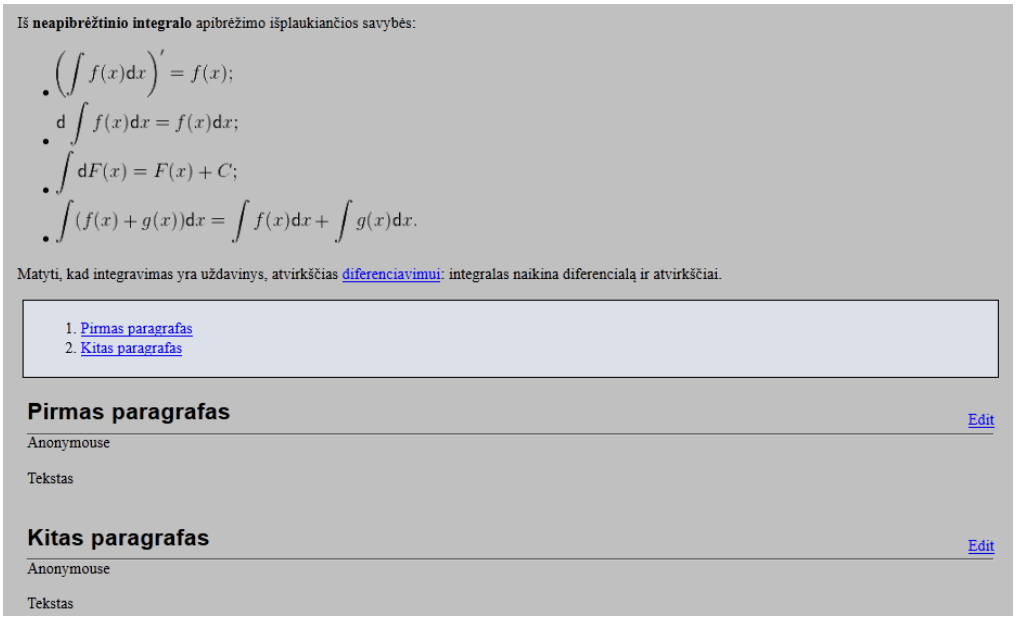
Pav 24. Redaguojamas paragrafas

pasirinkus veiksmą *Save paragraph* jis buvo atvaizduotas jau internetinio puslapio paragrafe.



Pav 25. Paragrafas išsaugotas

Taip pat automatiškai susiformavo internetinio puslapio turinys, kuris matomas pagrindinio aprašymo apačioje.



Pav 26. Automatinis turinio formavimas

Išsaugojus pakeitimus, po to pridėjus dar vieną paragrafą ir išsaugojus viską vėl atsirado, dvi puslapio versijos. Pasirinkus vieną iš jų gauname tokį patį puslapį kokį buvome išsaugoję.

Pav 27. Internetinio puslapio versijavimas

Pav 28. Internetinio puslapio versijavimas

Delete paragraph sėkmingai panaikina paragrafą, o *Cancel actions* atstato pakeitimus.

7. Temų paieška

Buvos sukurtos trys temos:

1. Testai. Raktažodžiai: testavimas, C++, Qt, C, Fortran 95. Aprašymas: Testavimo darbai įvairiomis programavimo kalbomis.
2. Testai2. Raktažodžiai: Testavimas, testai, testas. Aprašymas: Testavimas.
3. Stochastinis programavimas. Raktažodžiai: Stochastinis programavimas. Aprašymas: Stochastinio programavimo temos puslapis.

Įvedus į paieškos laukelį žodį darbai, buvo gražinta pirmoji tema. Įvedus žodį testavimas, buvo gražintos pirma ir antra temos. Įvedus žodį stochastinis – trečioji tema.

8. Matematinių funkcijų sukūrimas ir redagavimas.

Formose *Create functions ir Edit functions* užpildžius laukelius duomenimis ir paspaudus atitinkamai mygtukus *Save* ir *Save changes* duomenys buvo sėkmingai išsaugoti ir pakeisti.

9. Matematinių funkcijų paieška.

Formoje *Search* functions buvo sėkmingai surastos funkcijos pagal jų pavadinimą ir aprašymą.

10. Klasifikacijos sukūrimas ir redagavimas.

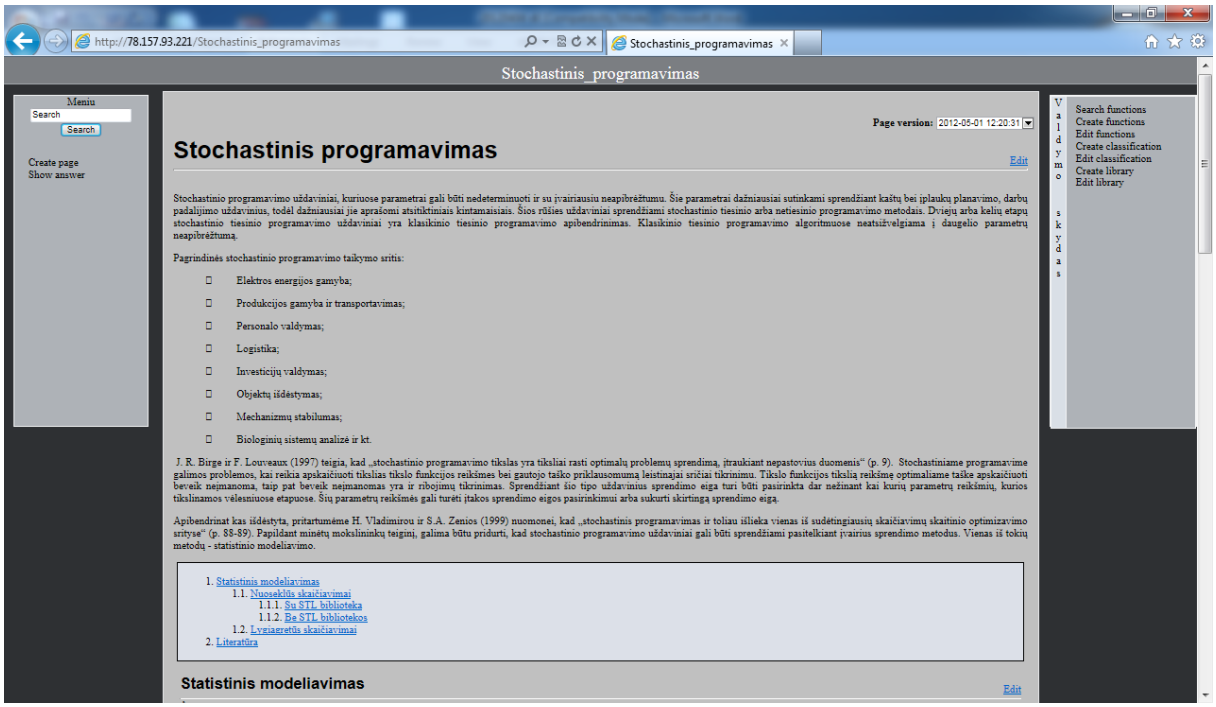
Formose *Create classification ir Edit classification* užpildžius laukelius duomenimis ir paspaudus atitinkamai mygtukus Create classification ir *Create class* buvo sėkmingai išsaugoti, o paspaudus *Edit classification* ir *Edit class*, pakeisti.

11. Matematinių funkcijų bibliotekos sukūrimas

Formose *Create library ir Edit library* užpildžius laukelius duomenimis ir paspaudus atitinkamai mygtukus Create ir Edit duomenys buvo sėkmingai išsaugoti ir pakeisti.

12. Monte Karlo metodas

Monte Karlo metodą realizuojančios programos t.y. nuoseklioji programa be STL bibliotekos, nuoseklioji programa su STL biblioteka bei lygiagrečioji programa patalpintos internetinėje matematinio programavimo ir modeliavimo sistemoje. Ją rasite šiuo adresu: http://78.157.93.221/Stochastinis_programavimas. Joje galima rasti šių programų išeities kodus ir atlikti jų tobulinimo darbus. Šioje svetainėje galima sukompiliuoti programinį kodą ir atlikti skaičiavimus. Internetinė matematinio programavimo ir modeliavimo sistema sukurta vikinomikos modeliu, todėl galima sukurti temas ir potemes konkrečiais klausimais, kurias papildyti gali visi informacinės visuomenes vartotojai. Šioje internetinėje svetainėje buvo sukurta puslapis su tema: Stochastinis programavimas (žr. pav. 29) , kurioje pateikti ir šios temos potemės, o taip pat ir programinės realizacijos.



Pav 29. Stochastinio programavimo puslapis internetinėje matematinio programavimo ir modeliavimo sistemoje

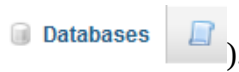
Su aukščiau minėta sistema buvo atliekami bandomieji kompiliavimo ir programų vykdymo veiksmai, kurie buvo sėkmingai atlikti.

3. Internetinės matematinio programavimo ir modeliavimo sistemos diegimas

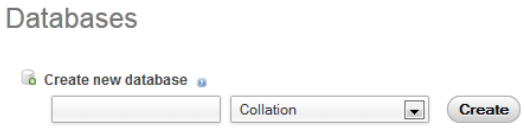
- ***Wamp* serverio diegimas**

Norint į kompiuterį įdiegti internetinę svetainę reikalingas *Wamp* serverio diegimas. Diegimo failą *wampserver2.2d-x32.exe* galite rasti *DVD* laikmenoje, kataloge *Programine iranga*. Instrukciją, kaip įdiegti šią programą, rasite internetiniu adresu: http://www.mysql.lt/wiki/WAMP_diegimas. Atkreipkite dėmesį į tai, kad ši programa reikalauja 80 prievado (angl. *Port*). Jis turi būti laisvas. Pavyzdžiui jį naudoja *skype* programa ir kartu programos veikti negalės, jeigu nepakeisite nustatymų.

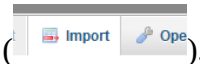
- **Duomenų bazės įkėlimas**

Duomenų bazės failiuką *duomenu_baze.sql* rasite kataloge *Internetine programavimo sistema*. Įdiegę ir paleidę programą *Wamp* programą, internetinėje naršyklėje įveskite adresą http://localhost/phpmyadmin/. Meniu juostoje pasirinkite punktą *Database* (

). Tada įrašykite duomenų bazės pavadinimą *math* ir spauskite mygtuką *create* (žr. pav.30).



Pav 30. Duomenų bazės kūrimo forma

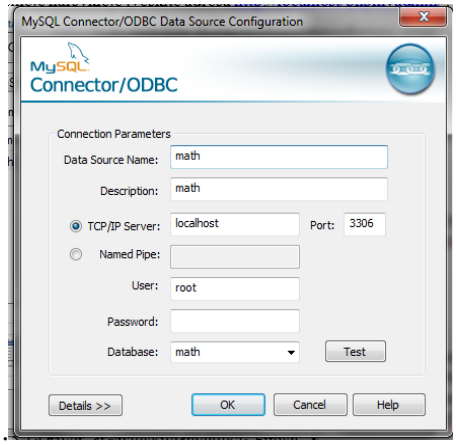
Atlikę veiksmus, kairėje pusėje pasirinkite duomenų bazę *math*. Tada meniu juostoje pasirinkite *import* mygtuką (

). Pagrindiniame puslapyje suraskite mygtuką *Browse*, pasirinkite failiuką *duomenu_baze.sql* ir spauskite mygtuką *Go*. Duomenų bazė bus importuota.

- ***ODBC* konfigūravimas**

Tam, kad serverinė programa galėtų prisijunkti prie duomenų bazės turite sukonfigūruoti *Windows* operacinėje sistemoje *ODBC* (angl. *Open Database Connectivity*). Pirmiausia įrašykite *MySql* tvarkyklę. Ją rasite kataloge *Programine iranga*, failas *mysql-connector-odbc-5.1.10-win32.msi*. Ją įdiegę nueikite į katalogą *Windows/system32* (64 bitų operacinėje sistemoje

Windows/SysWOW64). Paleiskite failiuką *odbcad32.exe*. Tenai pasirinkite kortelę *System DNS*. Tada spauskite *Add* mygtuką. Susiraskite tvarkyklę *MySQL ODBC 5.1 Driver* ir spauskite mygtuką *Finish*. Nustatykite nustatymus tokius kaip pavaizduota 31 paveikslėlyje. Jeigu keitėte vartotojo vardą (*User*), slaptažodį (*Password*) įrašydami *Wamp* programą, tada juos pakeiskite. Spauskite mygtuką *Test* norėdami patikrinti ar yra ryšys su duomenų baze.



Pav 31. ODBC konfigūravimas

- **Internetinės svetainės diegimas ir konfigūravimas**

Internetinės svetainės katalogus ir failus, kurie yra kataloge *Internetine programavimo sistema/Internetine svetaine*, nukopijuokite i *Wamp* programos diegimo kataloge esantį *www* katalogą. Jeigu keitėte duomenų bazės vartotojo vardą ir slaptažodį, tada pakoreguokite internetinės svetainės failą *conf.php* esantį kataloge *config*. Funkcijoje „*function conn($host='localhost',$user='root',$pass='',$db='math')*“ galite pakeisti duomenų bazės serverio adresą, vartotojo vardą, slaptažodį ir duomenų bazės pavadinimą. Visą tai padarius diegimas baigtas ir naršyklėje įvedus http://localhost/ adresą, turėtumėte įeiti į matematinio programavimo ir modeliavimo internetinę svetainę. Taip pat, kad veiktų programų kompiliavimas ir vykdymas jums reikia sukonfigūruoti ir paleisti serverinę programą.

- **Serverinės programos ir jos komponentų diegimas, konfigūravimas**

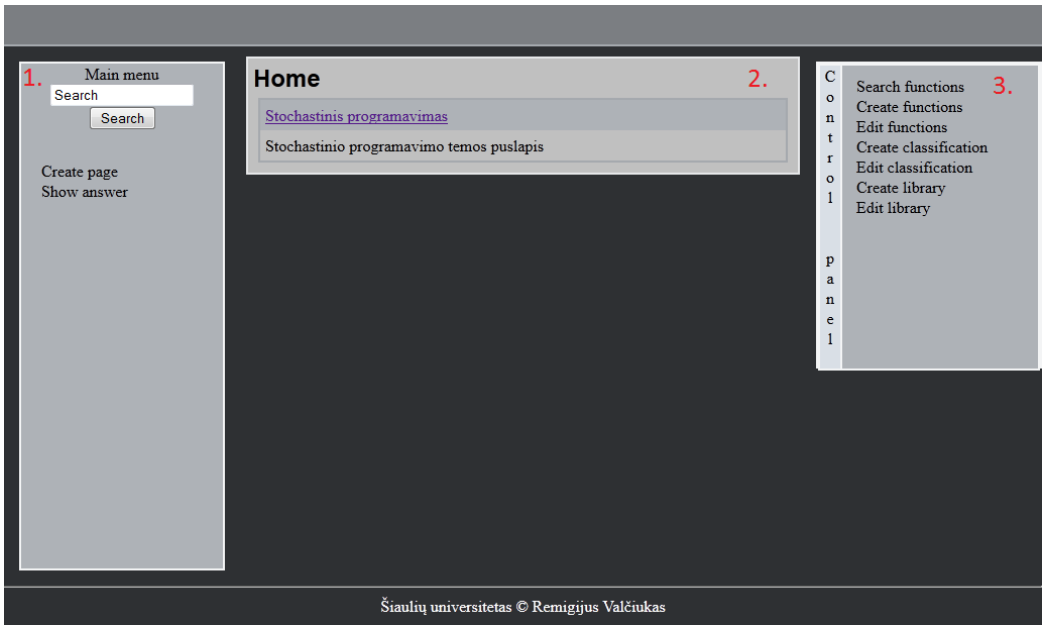
Visų pirma tam, kad galėtume kompiliuoti ir vykdyti sukurtas programas turime įrašyti kompiliatorius tame kompiuteryje/serveryje kur bus paleidžiama serverinė programa (Internetinę svetainę ir serverinę programą galite paleisti viename įrenginyje). Kataloge *Programine iranga* yra programų diegimo *g95-MinGW.exe*, *jdk-6u31-nb-7_1_1-windows-ml.exe*, *Qt_SDK_Win_offline_v1_1_4_en.exe*. Tai yra *Fortran*, *C/C++*, *Qt*, *Java* kompiliatoriai. Juos

jums reikia įdiegti. Kataloge *Internetine programavimo sistema/Programa* rasite serverinę programą (*vrMathServer.exe*). Jos instaliuoti nereikia, ją galima paleisti iš katalogo, bet prieš tai katalogą nukopijuokite į kompiuterį. Programą paleidus turite pamatyti išvestus visus parametrus ir pranešimą *Connected to database*, kuris reiškia, kad yra ryšys su duomenų baze. Faile *config.xml* galite nustatyti programos parametrus:

- <serverPort> – prievadas per kurį siaučiami duomenys iš internetinės svetainės;
- <serverAddress> – serverio adresas;
- <maxExeProgramCount> – maksimalus leidžiamas vykdomų programų kiekis vienu metu;
- <maxCompProgramCount> – maksimalus leidžiamas kompiliuojamų programų kiekis vienu metu;
- <cppCompilerPath> – kelias iki *C/C++* kompiliatoriaus vykdomojo failo;
- <cppCompilerBinDir> – kelias iki *C/C++* kompiliatoriaus katalogo;
- <javaExecutePath> – kelias iki *Java* programų vykdymo failo;
- <javaCompilerPath> – kelias iki *Java* kompiliatoriaus vykdomojo failo;
- <javaCompilerBinDir> – kelias iki *Java* vykdomųjų failų katalogo;
- <qtCompilerPath> – kelias iki *QT* kompiliatoriaus vykdomojo failo;
- <qtDir> – kelias iki *QT* bibliotekos katalogo;
- <qtbinDir> – kelias iki *QT* bibliotekos vykdomųjų failų;
- <qtModules> – leidžiami *QT* bibliotekos moduliai.
- <fortranCompilerPath> – kelias iki *Fortran* kompiliatoriaus vykdomojo failo;
- <terminalPath> – kelias iki operacinėje sistemoje esančio komandinės eilutės interpretatoriaus vykdomojo failo.

4. Internetinės matematinio programavimo ir modeliavimo sistemos vartotojo vadovas

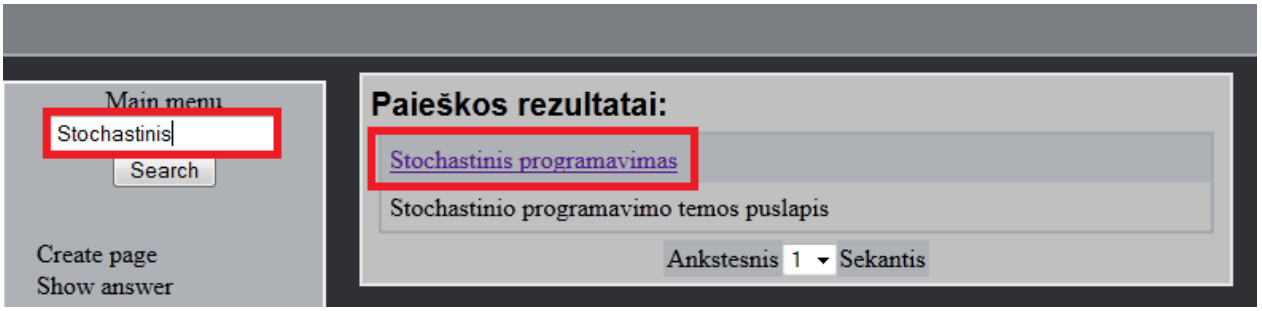
1. Internetinės svetainė



Pav 32. Internetinės matematinio programavimo ir modeliavimo sistemos vartotojo sąsaja

Trisdešimt antrame paveikslėlyje matome internetinės svetainės pagrindinį puslapį. Jame raudonais skaičiais pažymėtos dalys: 1 – pagrindinis meniu, 2 – turinio rodymo modulis, 3 – valdymo skydas. Pagrindiniame meniu matome meniu punktus, kurių pagalba galime iškviesti puslapio kūrimo *Create page* bei rezultato paieškos (*Show answer*) modulius. Taip pat yra paieškos laukelis (*Search*), kuriame įvedus raktinius žodžius galime ieškoti internetinių puslapių, kuriuose aprašyti skaitiniai algoritmai, kita informacija bei realizuotos programos matematinio programavimo uždaviniams spręsti. Turinio rodymo modulyje dinamiškai keičiami internetiniai puslapiai. Pradiniame svetainės puslapyje yra rodomos naujai sukurtos temos. Valdymo skydo (*Control panel*) modulyje yra meniu punktai, kurie iškviečia modulius skirtus kurti ir redaguoti funkcijų bibliotekoms.

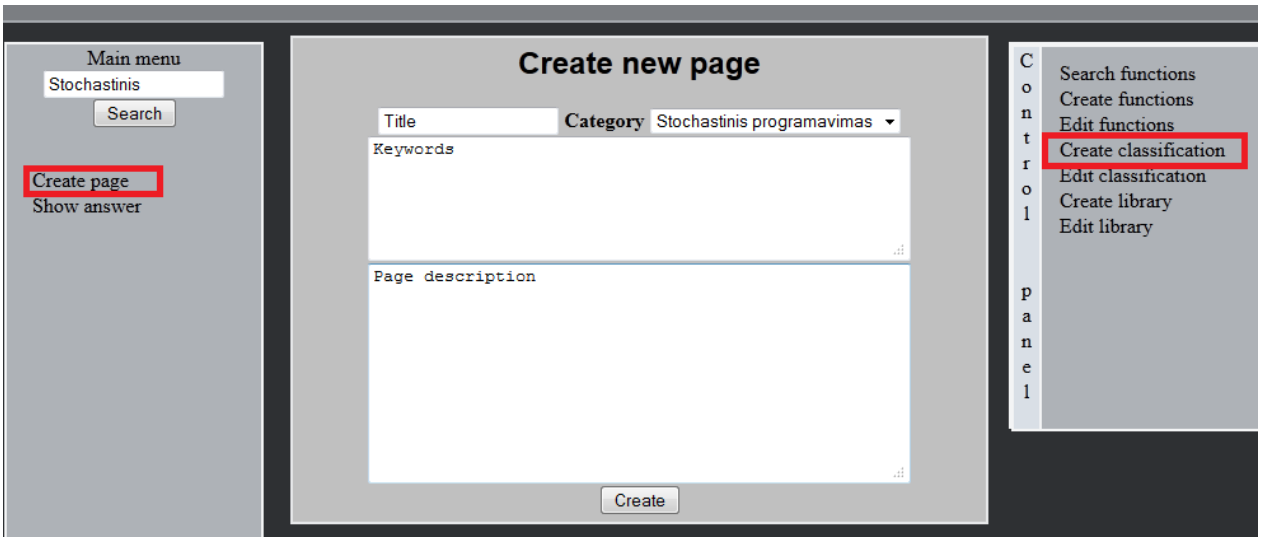
2. Turinio paieška



Pav 33. Temų paieška

Norint surasti puslapį kokia nors tema, į paieškos laukelį įveskite norimą frazę (žr. pav. 33.), tada spauskite *Search* mygtuką. Kiekviena rasta tema atvaizduojama su jos pavadinimu ir trumpu aprašymų. Paspaudę ant temos pavadinimo būsite nukreipti į puslapį ta tema.

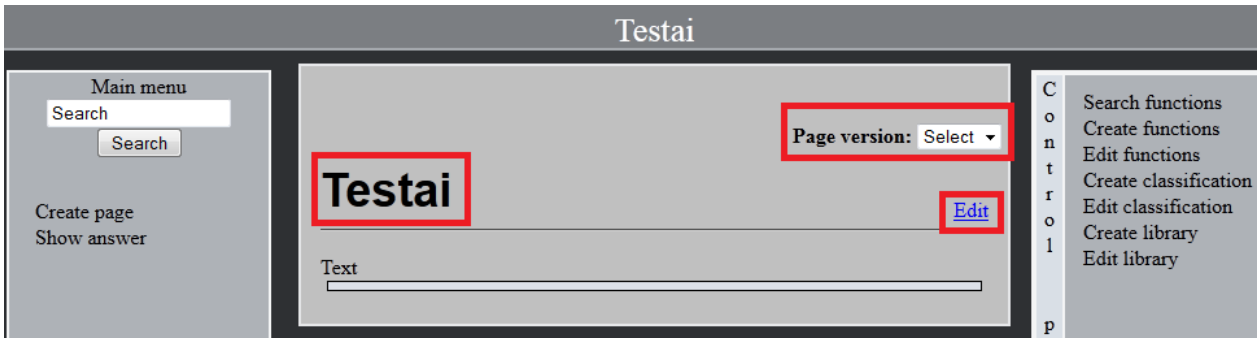
3. Puslapio ir temos kūrimas



Pav 34. Puslapio kūrimas

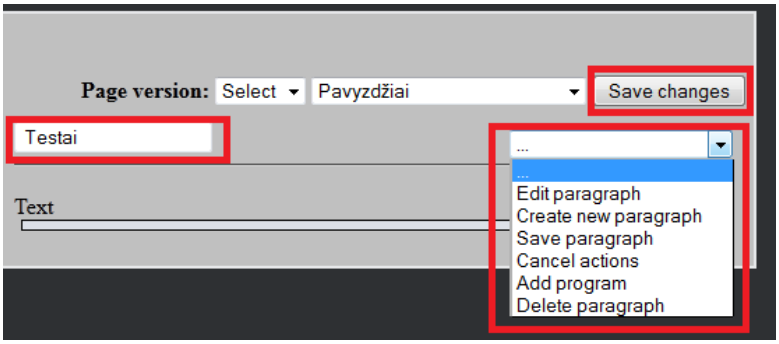
Norėdami sukurti puslapį kokia nors tema, paspauskite meniu punktą *Create page*. Tada atsiradusioje formoje (žr. pav. 34), įrašykite temos pavadinimą (*Title*), raktažodžius (*keywords*) ir trumpą aprašymą (*Page description*). Pagal visus šiuos žodžius bus atliekama paieška. Taip pat puslapiui priskirkite kategoriją. Ją galima pasirinkti punkte *Category*. Jeigu nėra tokios kategorijos kokios norėtumėte, tada ją susikurkite. Tam reikia iškviesti klasifikacijos kūrimo modulį (spauskite meniu punktą *Create classification*). Puslapio kūrimo formoje paspaudus mygtuką *Create* sukuriamas puslapis.

4. Puslapio redagavimas



Pav 35. Internetinis puslapis su tema

Sukurtame puslapyje (žr. pav. 35.) matome puslapio turinio versijos pasirinkimo galimybę (*Page versijon*), puslapio temos pavadinimą, bei prie kiekvienos pastraipos, punktą *Edit*. Jį paspaudus, turime galimybę esamame puslapyje atlikti daugiau veiksmų (žr. pav. 36).



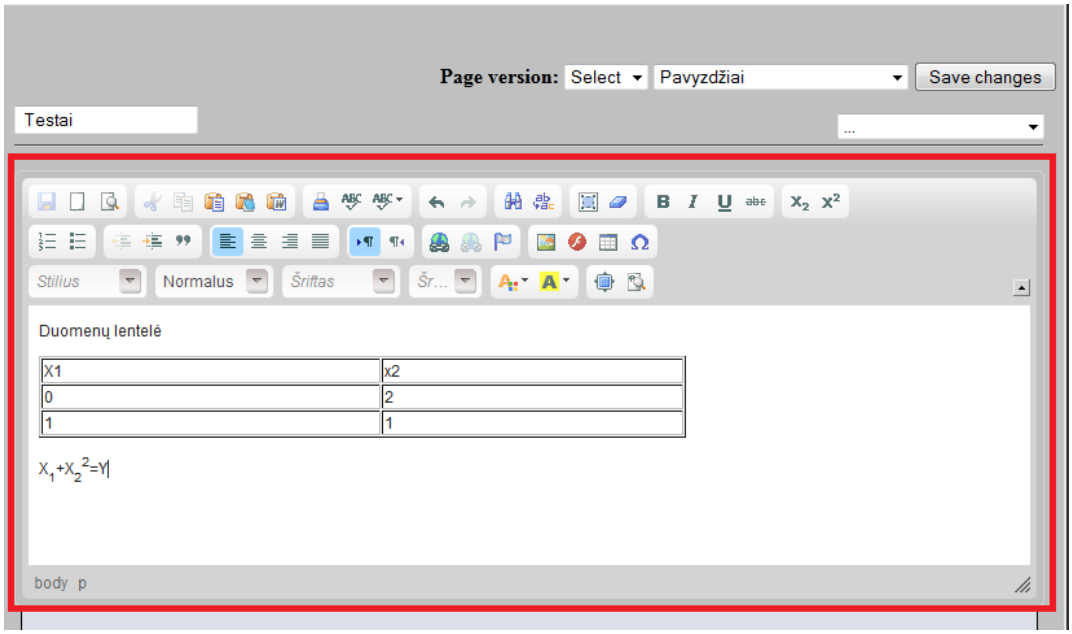
Pav 36. Internetinio puslapio redagavimo galimybės

Pirmiausia mes galime koreguoti paragrafo pavadinimą. Antra išskleisti veiksmų pasirinkimą (žr. pav. 36):

a. *Edit paragraph* – iškviečia teksto redaktorių konkrečiame paragrafe, kuriame galite aprašyti matematinius modelius, skaitinius algoritmus bei pateikti kitą informaciją. **Prisiminkite, kad vienu metu galite redaguoti tik vieną paragrafą!**
b. *Create new paragraph* – internetiniame puslapyje prideda naują paragrafą.
c. *Save paragraph* – kviečiamas tada, kai norime išsaugoti informaciją paragrafe.
d. *Cancel actions* – kviečiamas tada, kai norime atšaukti daromus pakeitimus.
e. *Add program* – paragrafe prideda modulį programos rašyti.
f. *Delete paragraph* – ištrina visą paragrafą.

Visi pakeitimai padaryti puslapyje bus išsaugoti tik tada, kai paspausite mygtuką *Save changes* (žr. pav. 36). Taip pat turite būti išsitikinę, kad nei vienas paragrafas nėra šiuo metu redaguojamas, kitaip nebus galima išsaugoti informacijos.

Priemonė paragrafui redaguoti pavaizduota 37 paveikslėlyje. Daugiau apie įrankio galimybes galite pasiskaityti internetiniame puslapyje: http://docs.cksource.com/CKEditor_3.x/Users_Guide.



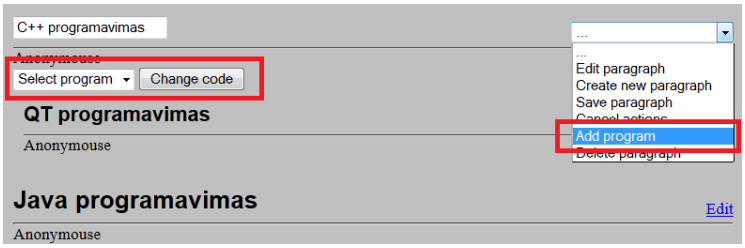
Pav 37. Teksto redaktorius

Pridedant paragrafus ir juos saugant yra automatiškai generuojamas turinys (žr. pav. 28.)

Pav 38. Automatinis puslapio turinys

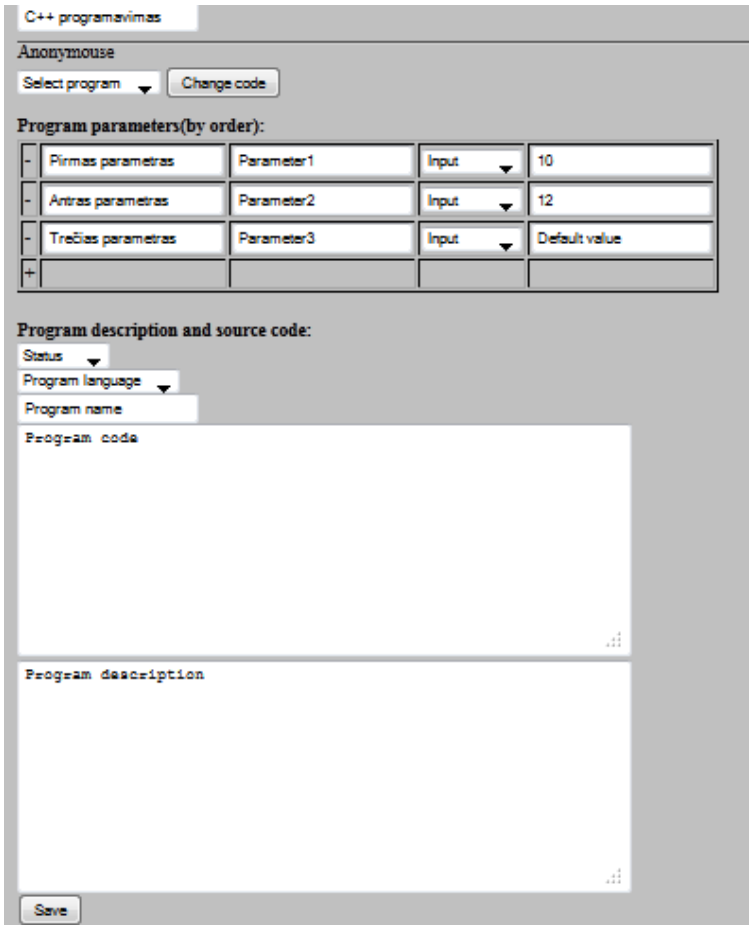
5. Programų rašymas, kompiliavimas, vykdymas, rezultatų paieška

Norint parašyti programą pirmiausia paragrafe pridėkite programų rašymo modulį (žr. pav. 39). Pasirinkite veiksmą *Add program*. Tada atsiradusiame modulyje spaudžiame *Change code*.



Pav 39. Programų rašymo modulis

Po šio veiksmo apačioje pamatysime formą, kuria galėsite parašyti programą. *Program parameters* lentelėje (žr. pav. 40) aprašome kuriamos programos vartotojo sąsają.

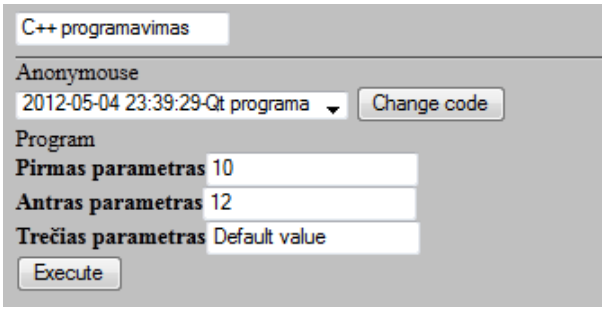


Pav 40. Programos kūrimo forma

Toje lentelėje paspaudę minusą, parametrą ištrinsite, o paspaudus pliusą – pridėsite naują. Antrajame stulpelyje aprašomas parametro pavadinimas, kurį matys vartotojai. Sekantis stulpelis skirtas taip pat parametro pavadinimui, tačiau jis reikalingas tik programiniu požiūriu. **Šiame stulpelyje parametrų pavadinimai turi būti skirtingi!** Priešingu atveju, kai kurie parametrai gali būti neperduoti vykdomai programai. Ketvirtajame lentelės stulpelyje, formos laukelio tipas:

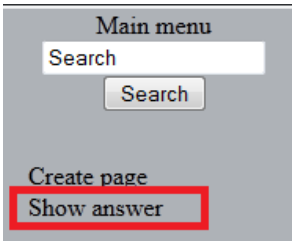
a. *Input* – tekstinis laukas.
b. *Text* – tekstinis laukas, kuriame galima įvesti didelį kiekį duomenų.
c. *Browse* – failo pridėjimo laukas.

Paskutiniajame laukelyje yra aprašoma numatytoji parametro reikšmė pagal nutylėjimą. Aprašę programos vartotojo sąsają, reikia parašyti pačią programą. *Program description and source code* formos skiltyje, turime pasirinkti programos statusą bei kalbą. Tada parašome programos pavadinimą, programos kodą ir jos aprašymą (žr. pav. 40). Viską atlikę spaudžiame mygtuką *Save*. Duomenys yra išsiunčiami į serverį ir programa yra kompiliuojama, o internetinėje svetainėje sugeneruojama programos vartotojo sąsaja – duomenų įvedimo forma (žr. pav. 41). **Nepamirškite išsaugoti pakeitimų, puslapyje paspauskite *Save changes*.**



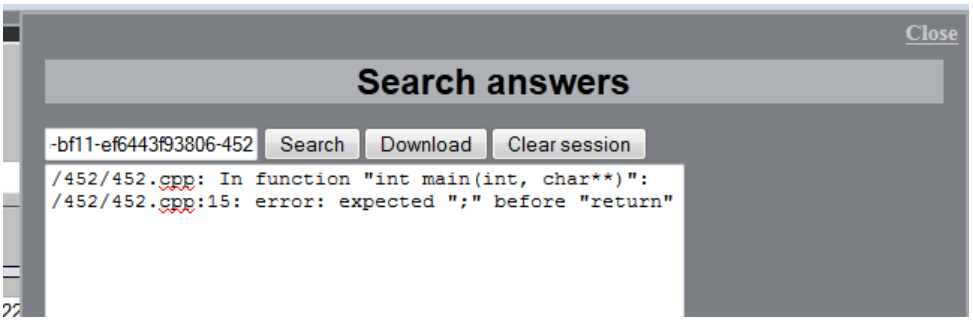
Pav 41. Duomenų forma skirta programos parametram suvesti

Pranešimus apie kompiliavimo klaidas rasite paspaudę meniu punktą *Show answers* (žr. pav. 42).



Pav 42. Kviečiamas rezultatų paieškos modulis

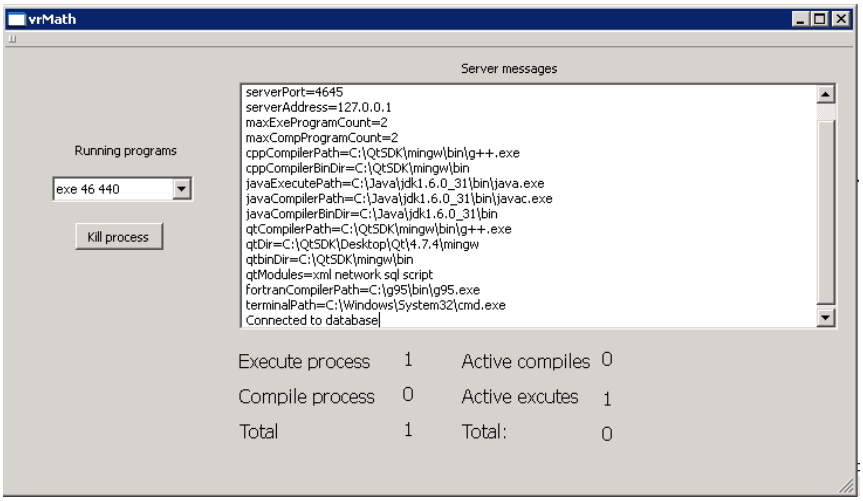
Papildome modaliniame lange jūs pamatysite rezultatų paieškos formą (žr. pav. 43).



Pav 43. Rezultatų paieška modaliniame lange

Joje bus matyti laukelis su žyme. Paspaudus mygtuką *Search* pagal šią žymę yra atliekama paieška. Ši žymė automatiškai įrašoma į laukelį, kai vartotojas atlieka programos kompiliavimą arba vykdymą. Svarbu, kad tas žymes įsisaugotumėte ir baigus darbą paspaustumėte mygtuką *Clear session*. Taip yra išvaloma sesija ir visi laukai. Su išsaugota žyme jūs bet kada galėsite rasti rezultatus. Paspaudę *Download* mygtuką, jūs galėsite juos parsisiųsti. Vykdomų programų rezultatų ieškokite tokiu pat būdu.

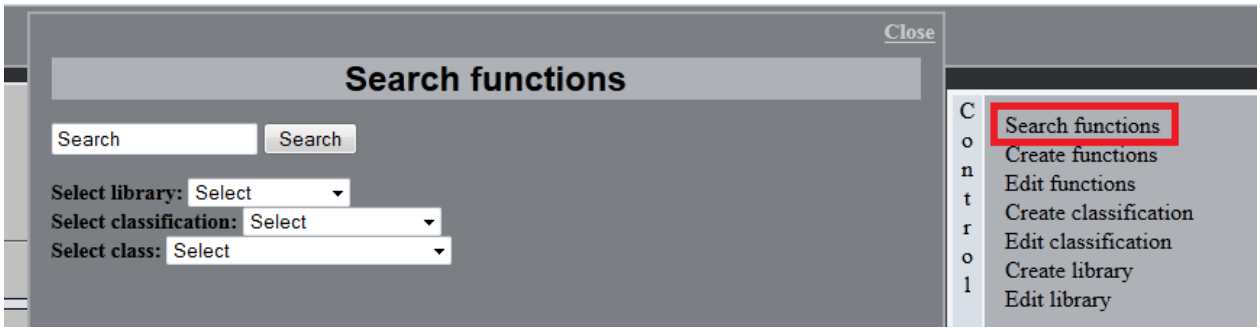
6. Serverinė programa



Pav 44. Serverinės programos vartotojo sąsaja

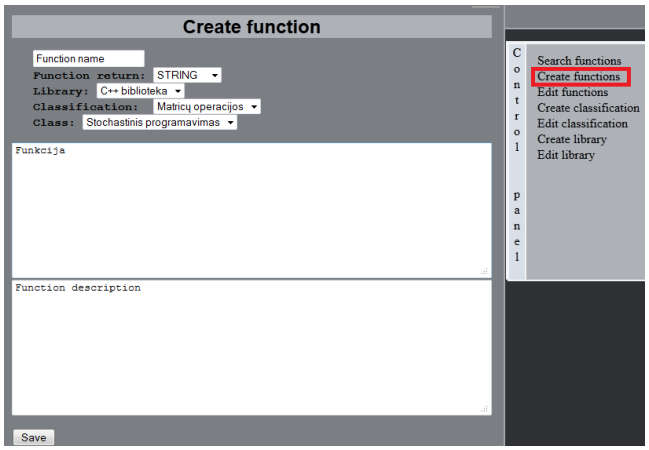
Serverinėje programoje administratorius gali matyti kiek yra išviso aktyvių ir neaktyvių procesų, kurie turi vykdyti programų kompiliavimo ir vykdymo darbus (žr. pav. 44). Taip pat jam suteikiama galimybė nutraukti konkretų vykdomą procesą, paspaudus mygtuką *Kill process*.

7. Bibliotekų ir funkcijų kūrimas, redagavimas, paieška, klasifikavimas

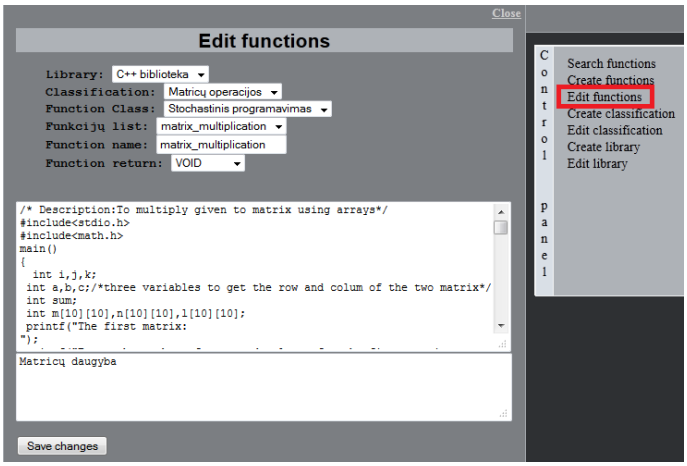


Pav 45. Funkcijų paieška

Valdymo skyde (*Control panel*) (žr. pav. 45) paspauskite meniu punktą *Search functions*. Atsiradusiame modaliniame lange *Search functions* galite atlikti funkcijų paiešką pagal funkcijų bibliotekos, klasifikacijos ir jų klases pavadinimą. Sekantis meniu punktai yra *Create functions* ir *Edit functions* (žr. pav. 46, 47).

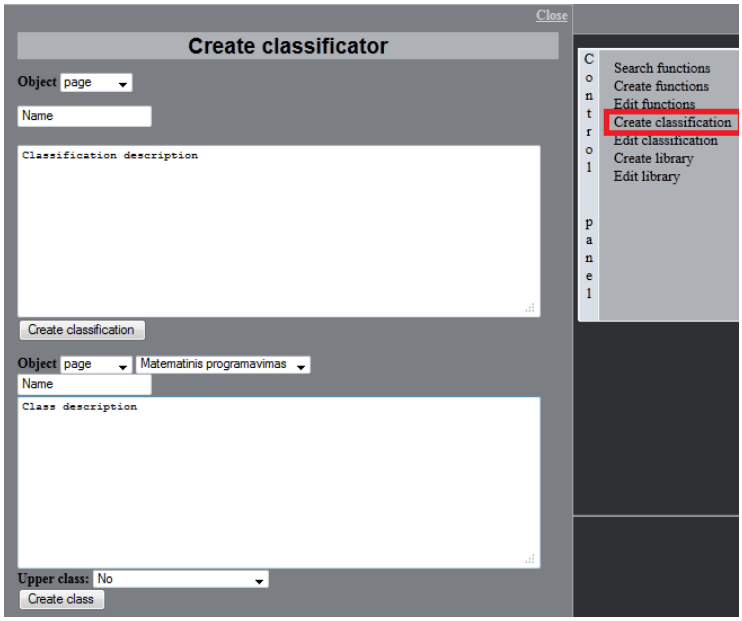


Pav 46. Funkcijų kūrimas

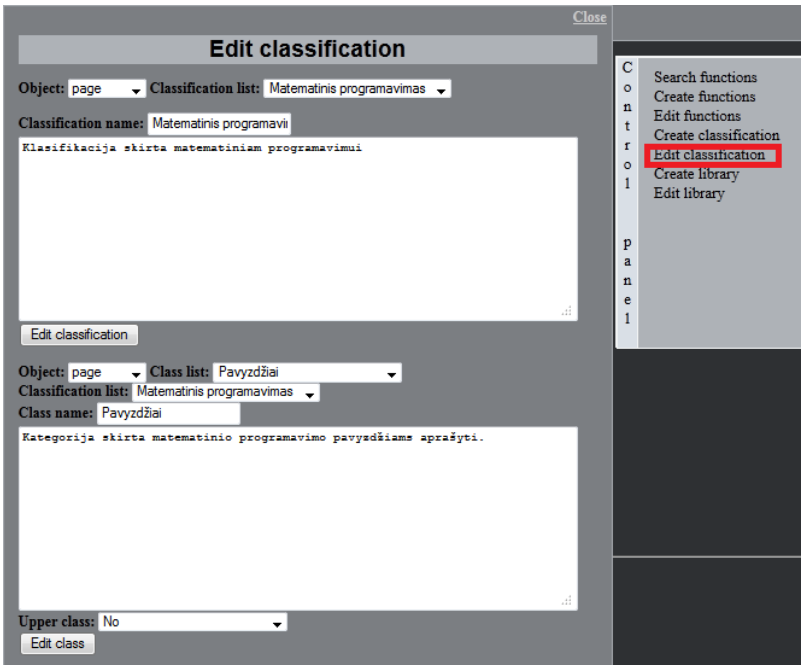


Pav 47. Funkcijų redagavimas

Šiose modaliniuose languose jūs galite kurti ir redaguoti matematines funkcijas skirtas įvairioms programavimo kalboms. Norint sukurti funkciją reikia parašyti jos pavadinimą, nustatyti jos tipą, priskirti jai konkrečia matematinių funkcijų biblioteką, klasifikaciją ir klasę. Tada įrašome pilną funkcijos kodą ir jos aprašymą.



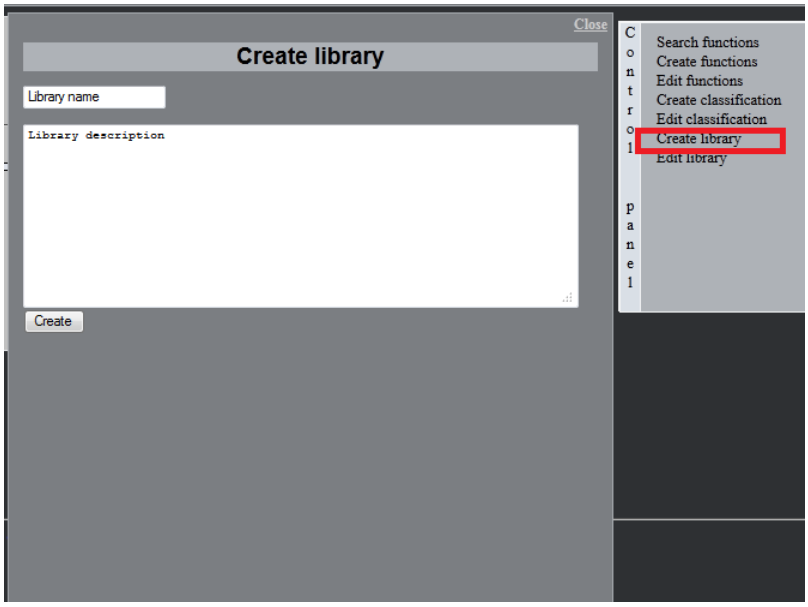
Pav 48. Klasifikatoriaus kūrimas



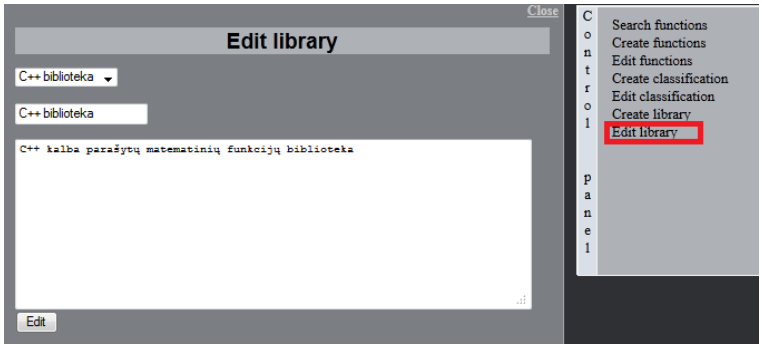
Pav 49. Klasifikatoriaus redagavimas

Taip pat galite sukurti ir redaguoti internetinių puslapių ir matematinių funkcijų klasifikacijas (žr. pav. 48, 49). Pirmiausia, kuriant ar redaguojant klasifikaciją reikia pasirinkti objektą (*Object*).

Page reiškia puslapio klasifikaciją, o *Program* funkcijų klasifikaciją. Klasifikacija apibrėžia funkcijų ar internetinių puslapių grupes, o klasė (*class*) konkrečią funkcijų ar internetinių puslapių grupę. *Upper class* punktas nurodo klasių sąryšį.



Pav 50. Funkcijų bibliotekos kūrimas



Pav 51. Funkcijų bibliotekos redagavimas

Paskutiniai du punktai *Create library* ir *Edit library* iškviečia modalinius langus (žr. Pav. 50, 51), kuriuose galite kurti ir redaguoti funkcijų bibliotekas. Jose turite nurodyti bibliotekos pavadinimą ir trumpą aprašymą.

5. *Netlib Repository LAPACK*

5.1. *Netlib Repository LAPACK* bibliotekoje naudojami matricų tipai ir jų trumpiniai

BD – bidiagonal

DI – diagonal

GB – general band

GE – general (i.e., unsymmetric, in some cases rectangular)

GG – general matrices, generalized problem (i.e., a pair of general matrices)

GT – general tridiagonal

HB – (complex) Hermitian band

HE – (complex) Hermitian

HG – upper Hessenberg matrix, generalized problem (i.e a Hessenberg and a triangular matrix)

HP – (complex) Hermitian, packed storage

HS – upper Hessenberg

OP – (real) orthogonal, packed storage

OR – (real) orthogonal

PB – symmetric or Hermitian positive definite band

PO – symmetric or Hermitian positive definite

PP – symmetric or Hermitian positive definite, packed storage

PT – symmetric or Hermitian positive definite tridiagonal

SB – (real) symmetric band

SP – symmetric, packed storage

ST – (real) symmetric tridiagonal

SY – symmetric

TB – triangular band

TG – triangular matrices, generalized problem (i.e., a pair of triangular matrices)

TP – triangular, packed storage

TR – triangular (or in some cases quasi-triangular)

TZ – trapezoidal

UN – (complex) unitary

UP – (complex) unitary, packed storage

5.2. *Netlib Repository LAPACK* bibliotekoje naudojamos paprogramės

Apie paprogrames naudojamas *Netlib Repository Lapack* bibliotekoje plačiau galite rasti adresu http://www.netlib.org/lapack/lug/node37.html [30]. Lentelės su paprogramių pavadinimais yra paimtos iš šio internetinio puslapio.

Computational routines for linear equations (1)

Type of matrix	Operation	Single precision		Double precision	
and storage scheme		real	complex	real	complex
general	factorize	SGETRF	CGETRF	DGETRF	ZGETRF
	solve using factorization	SGETRS	CGETRS	DGETRS	ZGETRS
	estimate condition number	SGECON	CGECON	DGECON	ZGECON
	error bounds for solution	SGERFS	CGERFS	DGERFS	ZGERFS
	invert using factorization	SGETRI	CGETRI	DGETRI	ZGETRI
	equilibrate	SGEEQU	CGEEQU	DGEEQU	ZGEEQU
general	factorize	SGBTRF	CGBTRF	DGBTRF	ZGBTRF
band	solve using factorization	SGBTRS	CGBTRS	DGBTRS	ZGBTRS
	estimate condition number	SGBCON	CGBCON	DGBCON	ZGBCON
	error bounds for solution	SGBRFS	CGBRFS	DGBRFS	ZGBRFS
	equilibrate	SGBEQU	CGBEQU	DGBEQU	ZGBEQU
general	factorize	SGTTRF	CGTTRF	DGTTRF	ZGTTRF
tridiagonal	solve using factorization	SGTTRS	CGTTRS	DGTTRS	ZGTTRS
	estimate condition number	SGTCON	CGTCON	DGTCON	ZGTCON
	error bounds for solution	SGTRFS	CGTRFS	DGTRFS	ZGTRFS
symmetric/Hermitian	factorize	SPOTRF	CPOTRF	DPOTRF	ZPOTRF
positive definite	solve using factorization	SPOTRS	CPOTRS	DPOTRS	ZPOTRS
	estimate condition number	SPOCON	CPOCON	DPOCON	ZPOCON
	error bounds for solution	SPORFS	CPORFS	DPORFS	ZPORFS
	invert using factorization	SPOTRI	CPOTRI	DPOTRI	ZPOTRI
	equilibrate	SPOEQU	CPOEQU	DPOEQU	ZPOEQU
symmetric/Hermitian	factorize	SPPTRF	CPPTRF	DPPTRF	ZPPTRF
positive definite	solve using factorization	SPPTRS	CPPTRS	DPPTRS	ZPPTRS
(packed storage)	estimate condition number	SPPCON	CPPCON	DPPCON	ZPPCON
	error bounds for solution	SPPRFS	CPPRFS	DPPRFS	ZPPRFS

	invert using factorization	SPPTRI	CPPTRI	DPPTRI	ZPPTRI
	equilibrate	SPPEQU	CPPEQU	DPPEQU	ZPPEQU
symmetric/Hermitian	factorize	SPBTRF	CPBTRF	DPBTRF	ZPBTRF
positive definite	solve using factorization	SPBTRS	CPBTRS	DPBTRS	ZPBTRS
band	estimate condition number	SPBCON	CPBCON	DPBCON	ZPBCON
	error bounds for solution	SPBRFS	CPBRFS	DPBRFS	ZPBRFS
	equilibrate	SPBEQU	CPBEQU	DPBEQU	ZPBEQU
symmetric/Hermitian	factorize	SPTTRF	CPTTRF	DPTTRF	ZPTTRF
positive definite	solve using factorization	SPTTRS	CPTTRS	DPTTRS	ZPTTRS
tridiagonal	estimate condition number	SPTCON	CPTCON	DPTCON	ZPTCON
	error bounds for solution	SPTRFS	CPTRFS	DPTRFS	ZPTRFS

Computational routines for linear equations (2)

Type of matrix	Operation	Single precision		Double precision	
and storage scheme		real	complex	real	complex
symmetric/Hermitian	factorize	SSYTRF	CHETRF	DSYTRF	ZHETRF
indefinite	solve using factorization	SSYTRS	CHETRS	DSYTRS	ZHETRS
	estimate condition number	SSYCON	CHECON	DSYCON	ZHECON
	error bounds for solution	SSYRFS	CHERFS	DSYRFS	ZHERFS
	invert using factorization	SSYTRI	CHETRI	DSYTRI	ZHETRI
complex symmetric	factorize		CSYTRF		ZSYTRF
	solve using factorization		CSYTRS		ZSYTRS
	estimate condition number		CSYCON		ZSYCON
	error bounds for solution		CSYRFS		ZSYRFS
	invert using factorization		CSYTRI		ZSYTRI
symmetric/Hermitian	factorize	SSPTRF	CHPTRF	DSPTRF	ZHPTRF
indefinite	solve using factorization	SSPTRS	CHPTRS	DSPTRS	ZHPTRS
(packed storage)	estimate condition number	SSPCON	CHPCON	DSPCON	ZHPCON
	error bounds for solution	SSPRFS	CHPRFS	DSPRFS	ZHPRFS
	invert using factorization	SSPTRI	CHPTRI	DSPTRI	ZHPTRI
complex symmetric	factorize		CSPTRF		ZSPTRF
(packed storage)	solve using factorization		CSPTRS		ZSPTRS
	estimate condition number		CSPCON		ZSPCON
	error bounds for solution		CSPRFS		ZSPRFS

	invert using factorization		CSPTRI		ZSPTRI
triangular	solve	STRTRS	CTRTRS	DTRTRS	ZTRTRS
	estimate condition number	STRCON	CTRCON	DTRCON	ZTRCON
	error bounds for solution	STRRFS	CTRRFS	DTRRFS	ZTRRFS
	invert	STRTRI	CTRTRI	DTRTRI	ZTRTRI
triangular	solve	STPTRS	CTPTRS	DTPTRS	ZTPTRS
(packed storage)	estimate condition number	STPCON	CTPCON	DTPCON	ZTPCON
	error bounds for solution	STPRFS	CTPRFS	DTPRFS	ZTPRFS
	invert	STPTRI	CTPTRI	DTPTRI	ZTPTRI
triangular	solve	STBTRS	CTBTRS	DTBTRS	ZTBTRS
band	estimate condition number	STBCON	CTBCON	DTBCON	ZTBCON
	error bounds for solution	STBRFS	CTBRFS	DTBRFS	ZTBRFS

Computational routines for the symmetric eigenproblem

Type of matrix	Operation	Single precision		Double precision	
and storage scheme		real	complex	real	complex
dense symmetric	tridiagonal reduction	SSYTRD	CHETRD	DSYTRD	ZHETRD
(or Hermitian)					
packed symmetric	tridiagonal reduction	SSPTRD	CHPTRD	DSPTRD	ZHPTRD
(or Hermitian)					
band symmetric	tridiagonal reduction	SSBTRD	CHBTRD	DSBTRD	ZHBTRD
(or Hermitian)					
orthogonal/unitary	generate matrix after	SORGTR	CUNGTR	DORGTR	ZUNGTR
	reduction by xSYTRD				
	multiply matrix after	SORMTR	CUNMTR	DORMTR	ZUNMTR
	reduction by xSYTRD				
orthogonal/unitary	generate matrix after	SOPGTR	CUPGTR	DOPGTR	ZUPGTR
(packed storage)	reduction by xSPTRD				
	multiply matrix after	SOPMTR	CUPMTR	DOPMTR	ZUPMTR
	reduction by xSPTRD				
symmetric	eigenvalues/	SSTEQR	CSTEQR	DSTEQR	ZSTEQR
tridiagonal	eigenvectors via QR				
	eigenvalues only	SSTERF		DSTERF	

	via root-free QR				
	eigenvalues/	SSTEDC	CSTEDC	DSTEDC	ZSTEDC
	eigenvectors via				
	divide and conquer				
	eigenvalues/	SSTEGR	CSTEGR	DSTEGR	ZSTEGR
	eigenvectors via				
	RRR				
	eigenvalues only	SSTEBZ		DSTEBZ	
	via bisection				
	eigenvectors by	SSTEIN	CSTEIN	DSTEIN	ZSTEIN
	inverse iteration				
symmetric	eigenvalues/	SPTEQR	CPTEQR	DPTEQR	ZPTEQR
tridiagonal	eigenvectors				
positive definite					

Computational routines for the singular value decomposition

Type of matrix	Operation	Single precision		Double precision	
and storage scheme		real	complex	real	complex
general	bidiagonal reduction	SGEBRD	CGEBRD	DGEBRD	ZGEBRD
general band	bidiagonal reduction	SGBBRD	CGBBRD	DGBBRD	ZGBBRD
orthogonal/unitary	generate matrix after	SORGBR	CUNGBR	DORGBR	ZUNGBR
	bidiagonal reduction				
	multiply matrix after	SORMBR	CUNMBR	DORMBR	ZUNMBR
	bidiagonal reduction				
bidiagonal	SVD using	SBDSQR	CBDSQR	DBDSQR	ZBDSQR
	QR or dqds				
	SVD using	SBDSDC		DBDSDC	
	divide-and-conquer				

Computational routines for the generalized symmetric definite eigenproblem

Type of matrix	Operation	Single precision		Double precision	
and storage scheme		real	complex	real	complex
symmetric/Hermitian	reduction	SSYGST	CHEGST	DSYGST	ZHEGST
symmetric/Hermitian	reduction	SSPGST	CHPGST	DSPGST	ZHPGST
(packed storage)					
symmetric/Hermitian	split Cholesky	SPBSTF	CPBSTF	DPBSTF	ZPBSTF
banded	factorization				
	reduction	SSBGST	DSBGST	CHBGST	ZHBGST

Computational routines for the generalized singular value decomposition

Operation	Single precision		Double precision	
	real	complex	real	complex
triangular reduction of ***A*** and ***B***	SGGSVP	CGGSVP	DGGSVP	ZGGSVP
GSVD of a pair of triangular matrices	STGSJA	CTGSJA	DTGSJA	ZTGSJA