KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Gedas Starkovas

Kompiuterinių matematinių sistemų animacijų tyrimas ir analizė

Magistro darbas

Darbo vadovas

Doc. dr. A. Lenkevičius

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Gedas Starkovas

Kompiuterinių matematinių sistemų animacijų tyrimas ir analizė

Magistro darbas

Recenzentas

Dr. K. Motiejūnas

2011- 05-27

Vadovas

Doc. dr. A. Lenkevičius

2011- 05-27

Atliko

IFM-9/1 gr. stud.
Gedas Starkovas
2011- 05-27

Kaunas, 2011

SUMMARY

Mathematics programs were started to develop as soon as computers, but now most calculations are not possible without a computer. Mathematical computing systems are widely used in modern engineering. All mathematical software systems are written in a high-level language tools (Java, Java +, C + +, etc..) for a wide range of data processing, visualization and others. To develop and explore mathematical models are widely used MATHCAD, MAPLE, MATLAB software package. These packages can draw high-quality graphics and create an animation. The paper has reviewed all of these mathematical systems and analyzed C + + platform. An analysis of schedules and other animation creation was made in this paper. Moreover, the comparison between systems was made also. It was found that all these software packages functions and uses are similar. Particular attention was devoted to coherent and detailed animations of selected mathematical package creation and management of feasibility, since this area has been extensively studied in Lithuania.

SANTRAUKA

Matematikos programos buvo pradėtos kurti vos tik atsiradus kompiuteriams, o dabar dauguma skaičiavimų be kompiuterio tiesiog neįmanomi. Kompiuterinės matematinės sistemos plačiai naudojamos šiuolaikinėje inžinerijoje. Visos šios matematinių programų sistemos yra parašytos aukšto lygio kalbų priemonėmis (JAVA, JAVA+, C++ ir kt.), tinka įvairių duomenų apdorojimui, vizualizacijai ir kt. Matematiniams modeliams kurti ir tyrinėti plačiausiai taikomi MATHCAD, MAPLE, MATLAB programų paketai. Šiuose paketuose galima braižyti aukštos kokybės grafikus ir kurti neprilygstamą animaciją. Darbe buvo apžvelgtos visos šios matematinės sistemos, bei panagrinėta C++ platforma. Buvo išanalizuotas grafikų ir visos kitos animacijos kūrimas, bei palygintos sistemos tarpusavyje. Nustatyta, kad visų šių programinių paketų funkcijos bei paskirtis panašios. Ypatingas dėmesys buvo skirtas nuosekliam bei detaliam pasirinktų matematinių paketų animacijos kūrimo ir valdymo galimybių analizavimui, nes ši sritis Lietuvoje nebuvo plačiai tyrinėta.

TURINYS

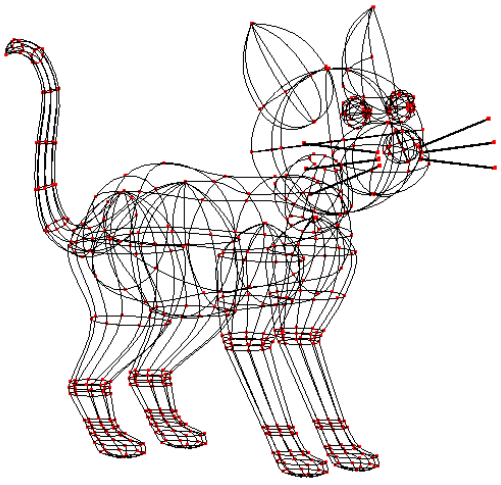
LENTELIŲ SĄRAŠAS

PAVEIKSLĖLIŲ SĄRAŠAS

1. ĮVADAS

Animacija yra greitai besikeičiančių dvimačių ar trimačių piešinių ar iliustracijų vaizdų rodymas, sudarant judėjimo vaizdą. Tai optinė judesio apgaulė, susidaranti dėl vaizdo užtęsimo, ir gali būti kuriama ir vaizduojama daugybe būdų. Pasaulyje labiausiai paplitusios yra trys animacijos technologijos rūšys:

- *Tradicinė animacija*. Kitaip vadinama ranka piešta animacija buvo plačiai naudojama XX a. animaciniuose filmuose. Kiekvieno kadro paveikslėlis piešiamas ant atskiro popieriaus lapo. Lapai vienas nuo kito skiriasi nežymiai taip sukurdami iliuziją, jog yra judesys. Kiekvienas kadro piešinys nuspalvinamas ant permatomos plėvelės – celuloido ir eilės tvarka nufotografuojamas. Šiais laikais ši technologija patobulėjo: nuskanuoti piešiniai spalvinami kompiuteriu. Animaciniai filmai sukurti tradicinės animacijos būdu: „Pinocchio" (JAV, 1940), „Animal Farm" (Didžioji Britanija, 1954), „Akira" (Japonija, 1988). Tradicinė animacija dar skirstoma į pilną, ribotą ir rotoskopinę.
- *Sustabdyto kadro animacija*. Sustabdyto kadro animacijoje paprastai naudojami tikri daiktai. Jie fotografuojami po kadrą truputį keičiant jų padėtį erdvėje. Taip sukuriama judesio iliuzija. Sustabdyto kadro animacijų yra begalės rūšių. Seniausia iš jų – lėlių animacija. Dirbdamas šiuo principu animatorius ne tik kontroliuoja judesį, bet ir pats gamina lėlę, stato aplinką. Žymiausi lėlių animacijos filmai: „Lucanus Cervus" (Lietuva, 1910), „The Tale of the Fox" (Prancūzija, 1937). Taip pat labai populiari sustabdyto kadro animacijos rūšis yra molio arba plastilino animacija. Figūros lipdomos iš molio arba kitos lengvai formuojamos medžiagos ir fotografuojamos taip pat kaip lėlių animacijoje.
- *Kompiuterinė animacija.* XXI a. sparčiai išpopuliarėjo kompiuterinė animacija, kuri leidžia greitai sukurti kokybišką dvimatį arba trimatį personažą ir ji lengvai animuoti. Kompiuterinė animacija pasižymi ypač švariomis linijomis ir tiksliu, apskaičiuotu judesiu. Kompiuterinė animacija skirstoma į dvimatę ir trimatę.



1 pav. Kompiuterinės animacijos piešinys

Kompiuterinė animacija buvo ypatingai reikalinga techninių skaičiavimų pavaizdavime, matematikos, chemijos, gamtos mokslų tyrimuose ir mokyme. Vaizdas – svarbus visos informacijos apjungimui ir vaizdavimui. Akimi žmogus informacijos priima beveik šimtą kartų didesnį kiekį nei ausimis. Vaizdinę informaciją žmogus daug greičiau pamato, suvokia ir įsimena.

Puikus informacijos vaizdavimo būdas yra grafikai. Gaila, bet daugelio programinės įrangos kuriami grafikai nėra dinamiški. Taigi, jei funkcijos kažkuris parametras kinta, tada tampa privalu kurti tiek skirtingų grafikų, kiek yra šio parametro reikšmių. Bet tada tampa sunku įsivaizduoti kurio nors dydžio kitimą laike ir jį išanalizuoti. Taip pat, funkcijose ar jų grafikuose dažnai kintantis parametras yra laikas.

Grafinių brėžinių ir modeliavimo rezultatų vaizdą geriausiai atspindi animacija. Su animacijos pagalba sukuriama seka paveikslėlių, kurie greitai keičiasi ir taip sukuria judesio efektą. Taigi naudojantis animaciją atvaizduoti funkcijos reikšmėms, turime tik vieną grafiką, kuris yra dinamiškas ir keičiasi priklausomai nuo funkcijos savybių, kad ir kiek tų savybių funkcija turėtų.

Lietuvoje kompiuterinių matematinių sistemų animacija beveik netyrinėta. Nors matematinių sistemų animacijos sukūrimo ir pritaikymo privalumai dideli, tik dviejose šaltiniuose, parašytose lietuvių kalba [5, 7], pasakyta, jog su sistema MAPLE galima sukurti animaciją. Šiame magistriniame darbe išbandomos ir tiriamos populiariausios kompiuterinės matematinės sistemos: MAPLE, MATHCAD bei MATLAB, jų animacijos sukūrimo, valdymo, apdorojimo ir pritaikymo galimybės. Tiriamojoje darbo dalyje analizuojamos ir aprašomos kompiuterinių matematinių sistemų animacijų galimybės, algoritmai, atvaizdavimo ypatumai. Darbas baigiamas išvadomis.

2. ANIMACIJOS METODŲ ANALITINĖ DALIS

2.1. MAGISTRINIO DARBO TEMOS TIKSLAS

Analizuoti kompiuterinės animacijos metodų visumą, pasirinkti tyrimo šaką, ištirti šakos metodus, pasiūlyti savo sprendimus.

2.2. MAGISTRINIO DARBO TEMOS APSVARSTYMAS

Animacija yra spartus kintančių dvimačių ar trimačių iliustracijos ar modelių vaizdo epizodų rodymas, sukuriant judėjimo iliuziją. Tai optinė judėjimo iliuzija, sukuriama dėl vaizdo užsitęsimo, ir gali būti kuriama ir vaizduojama daugybe metodų. Kompiuterinės animacijos (*computer animation*) metodų tyrimui buvo pasirinkta siauresnė šaka ir ji bus tyrinėjama iš pagrindų. Pasirinkta kompiuterinių matematinių sistemų animacijų tyrimas ir analizė

2.3. MAGISTRINIO DARBO TEMOS APRAŠAS

Vis greičiau ir greičiau vystantis technologijoms ir kompiuteriniam mokslui, taip pat vystosi ir plečiasi jų panaudojimo galimybės bei taikymai. Vienas iš tokių taikymų yra kompiuterinės matematinės sistemos. Inžinierinių mokslų specialistai, mokslininkai ir studentai dirba, skaičiuoja, braižo grafikus, sprendžia sunkias funkcijas, kuria elektroninius dokumentus su kompiuterinėmis matematinėmis sistemomis, kurios leidžia visa tai atlikti. Taip pat turi begalę kitų svarbių funkcijų: pvz.: ryšį su kitomis programomis. Taip pat lengva viską patalpinti internete. Šių programų dėka daug greičiau atliekami sudėtingi tarpiniai skaičiavimai ir mokymosi, tyrimo, analizavimo procesas tampa efektyvesnis. Labai svarbu išsirinkti tinkamą programinę įrangą, kuri tenkintų vartotojų poreikius.

Populiariausios kompiuterinės skaičiavimo sistemos MAPLE, MATHCAD ir MATLAB yra puikūs matematiniai įrankiai, suteikiantys nemažai duomenų apdorojimo, analizės ir vizualizacijos sprendimų. Minėtose sistemose suteikiama galimybė braižyti aukštos kokybės grafikus ir kurti nepakartojamą animaciją. Funkcijai priklausant nuo kurio nors parametro, kai jo reikšmės yra skirtingos, stebėti jos kitimą leidžia animacija. Šios programų galimybės labai naudingos techninių uždavinių skaičiavimų atvaizdavime, matematikos, chemijos, gamtos ir kitų tiksliųjų mokslų tyrimuose ir mokyme. Magistriniame darbe buvo

apžvelgta ir palyginta kompiuterinių matematinių sistemų MAPLE, MATHCAD ir MATLAB animacijų galimybės ir savybės, pateikti animacijos taikymo matematikos uždavinių sprendimuose pavyzdžiai, taip pat išbandyti animacijos braižymo algoritmai. Didelis dėmesys skirtas nuosekliam bei detaliam minėtų sistemų animacijos kūrimo ir valdymo galimybių aiškinimui.

2.4. MAGISTRINIO DARBO APLINKOS ANALIZĖ

Informacinių technologijų srityje yra sukurta nemažai specialių ir universalių matematikos sistemų (1 lentelė). Žemiau parodyta jų kaita, klasifikacija, struktūra. Aprašomos labiausiai reikalingos priemonės, nagrinėjamos jų galimybės, animacinės galimybės. Dirbant su matematinių sistemų animacija, svarbu taip pasirinkti kelias kompiuterines matematikos sistemas, kad jos teiktų bendrą visų kompiuterinių matematikos sistemų vaizdą, nes yra daug tiek savo apimtimi, tiek turiniu panašių sistemų (pvz., panašios yra MAPLE, MATHCAD, MATHEMATICA) [5, 6].

1 lentelė. Kompiuterinių matematinių sistemų suskirstymas į grupes pagal skaičiavimų pobūdį

Kompiuterinės matematikos sistemos (KMS)		
Skaitmeninių skaičiavimų KMS	Simbolinių skaičiavimų KMS	Universalios KMS
1.**Skaičiuokliai** (Įdiegti asmeniniuose kompiuteriuose) 2.**Skaičiuoklės** (Elektrononės lentelės) (pvz., SuperCalc, OmniCalc, Excel) 3.**Matricinės** (pvz., MATLAB) 4.**Statistinių skaičiavimų** (pvz., Statgraphics, SPSS, Statistica) 5.**Specialiųjų skaičiavimų** (TKSolver, Axum, MathPlot, DataFitNolinear Regresssion ir kt.)	DERIVE MUPAD MAPLE	MATHCAD (nuo 3.0 versijos) MATHEMATICA MAPLE (nuo MAPLE V) MATLAB (nuo 4.0. versijos su paketu Simbolic)

2.4.1. KOMPIUTERINIŲ MATEMATIKOS SISTEMŲ GALIMYBĖS

Magistriniame darbe buvo pasirinkta naudoti trijų skirtingų tipų kompiuterines matematikos sistemas norint ištirti jų animacijų galimybes, t.y. MAPLE, MATHCAD ir MATLAB [4, 5], nes:

- Elementariosios matematikos, trigonometrijos, matematinės analizės ir algebros pagrindams atvaizduoti tinkamiausia nedidelės apimties ir paprasta kompiuterinė matematikos sistema MATHCAD;
- Geometrijos skaičiavimuose geriausias galimybes teikia MAPLE;
- Gilesnėms matematikos studijoms, specialiems kursams, moksliniams tyrimams taikytina viena iš sistemų MAPLE arba MATHCAD, bet patogiausią sąsają turi MAPLE.
- MATLAB yra geriausia matematinių skaičiavimų automatizavimo sistema, naudojama įvairiems taikymams, moksliniams įvairių mokslo sričių tyrimams. Pagrindinis sistemos privalumas – jos praplečiamumas ir galimybė nesudėtingai pritaikyti realių specifinių vartotojo uždavinių sprendimui. Nors programavimo priemonės numatytos visose kompiuterinėse matematinėse sistemose, bet tik sistemą MATLAB galima taip paprastai ir patogiai papildyti vartotojo sukurtomis funkcijomis. Šios vartotojo kuriamos funkcijos paprastai būna trumpos, nes MATLAB turi daugiau kaip 1000 įvairiems taikymams skirtų funkcijų, kurias belieka priderinti prie savo uždavinio.

2.5. ANIMACIJOS METODŲ ANALIZĖ

Matematinėje animacijoje naudojamiems objektams perkelti iš programos, kurioje jie buvo sukurti, į programą, kurioje bus panaudota, naudojami keli rinkmenų formatai.

2 lentelė. Failų formatai

Rinkmenos formatas	Aprašymas
.x	x bylos architektūra yra paremta šablonais. Šablonas aprašo, kaip saugomas objektas. Taip pat vartotojai ir patys gali sukurti šablonus, kurie būna nenumatyti pradinėje bibliotekoje.
.MD2	Formatas dažniausiai naudojamas animacijos grotuvų modeliavimui, nors taip pat gali būti naudojami statiniams modeliavimams.
.MD3	Rinkmenoje animacija saugoma kadrais, kurie pavadinti tam tikrais pavadinimais. Raktinių/pavadintų kadrų naudojimas ganėtinai palengvina animacijos modeliavimą ir valdymą.
.3ds	Grafinio paketo „3D Studio Max“ rinkmenos tipas.

Paskutiniu metu sukurta begalė bylų formatų, kurios gali saugoti paviršiaus trikampių tinklus (skin meshes), bet yra viena iš paprastesnių galimybių - x rinkmenos naudojimas, tuo labiau, kad šią bylą palaiko ir „DirectX“ biblioteka. Segtuvuose gali būti išsaugomi nebūtinai paviršiaus trikampių tinklai (skin meshes), tačiau ir statiniai trikampių tinklai. x byloje

duomenys išsaugomi kaip šablonų kolekcija. Šablonai turi aprašymus, kuriuose yra talpinami duomenys modelyje. Kiekvienam duomenų tipui naudojamas atskiras modelis. Modeliai gali būti sukurti iš kitų modelių, tai suteikia galimybę sudaryti hierarchines scenas.

2.5.1. ANIMACIJOS METODAI

Animacijos algoritmai skirstomi į dvi šakas [9]: atvirąją (explicit) ir užslėptąją (implicit) animaciją. Atviros animacijos metodas išsaugo animuotų viršūnių aibę kiekvienam kadrui. Po išsaugojimo ir užrašymo, animacijos duomenų nuskaitymui būna naudojama įvairi technika, leidžianti animaciją veikti kaip vientisą. Atviroji (explicit) animacija yra lengviau suvokiama už užslėptą. Ją nesunku realizuoti programiškai, taip pat joje naudojama paprasta matematika, reikalaujanti mažų skaičiavimo resursų. Kita vertus, šis metodas viršūnių išsaugojimui eikvoja daug atminties. Norint atvaizduoti tikroviškus judesius, turime saugoti daug vieno veiksmo epizodų.

Užslėpta (implicit) animacija yra saugoma aukštesnio lygio judesio vaizdavimas. Pavyzdžiui, skeletinės animacijos (skeletal animation) modeliai išsaugo informaciją apie kiekvieną sujungimą virtualiajame modelyje. Vėliau esamu laiku, pagal surinktus duomenis suskaičiuojama objekto animacija. Šiame makete panaudota itin sudėtingi trigonometriniai bei matriciniai skaičiavimai, kurie yra labai eikvoja kompiuterio pagrindinį procesorių (CPU). Tačiau atminties panaudojimas yra minimalus, nes reikia saugoti tik mažas duomenų struktūras, norint pavaizduoti judesius.

2.5.2. ATVIROJI ANIMACIJA

Atvira animacija anksčiau buvo dažnai naudojama trimatės grafikos (3D) kompiuteriniuose žaidimuose, nes procesoriai nebuvo tokie galingi. Labiau atviros animacijos algoritmai buvo diegiami programose, kuriose tuo pat metu reikėdavo atvaizduoti grupę objektų. Jeigu norima panaudoti uždaros animacijos metodą, reikia atlikti nemažai matematinių veiksmų. Tačiau uždaros animacijos algoritmai tampa labiau populiaresni, nes šiandien kompiuteriuose naudojamos animacijos virsta vis sudėtingesnėmis, tikroviškesnėmis ir detalesnėmis [9].

Atviros animacijos metodika yra suskirstyta į:

- Kadrų (frame) animacija;
- Raktinių (keyframe) kadrų animacija;

Kadrų (frame) animacija yra viena paprasčiausių animacijos technikų atsiradusių iš tradicinės animacijos metodikos. Seniau animacijoje kadrai buvo piešiami ant celofaninių lakštų, kur kiekviena veikėjo poza buvo nupiešiama ant atskiro lakšto. Kadrų animacija yra labai panašiai veikianti. Veikėjo pozos saugomos atskirai ir vėliau paleidžiamos tam tikru tempu (paprastai 25 kadrai per sekundę).

Kiekviename kadre saugomos viršūnių koordinatės. Išsaugoma tik tai viena tekstūrų koordinačių kopija, nes ji nesikeis visos animacijos veikimo režime. Tas pats taikoma ir viršūnių spalvų, normalių naudojamų apšvietimui skaičiuoti saugojimui. Tokie veiksmai sumažina užimtos atminties kiekį.

Tačiau net ir šitaip patobulinus, kadrų animacija reikalauja labai daug atminties. Šis grubios jėgos (brute-force) metodas yra siūlomas naudoti tik labai specifinėse aplinkose, kai animacijai reikia tik kelių kadrų, kurie neturėtų labai daug trikampių.

Tai ne vienintelis minusas kadrų animacijoje. Pavyzdžiui žaidimuose kadrų skaičius nėra pastovus ir to pasakoje kadrų skaičius priklauso nuo scenos sudėtingumo. Todėl pravartu padaryti, kad atskirų objektų animacijos neišsiskirtų. Pavyzdžiui, žaidimo personažas juda žymiai lėčiau nei kiti objektai naudojami toje animacijoje. Problema yra ta, kad viršūnių tinklai (meshes) yra apskaičiuoti diskrečiu laiko momentu. Jeigu pasirinktume reikšmę, kuri būtų mažiausiai nutolusi nuo reikšmės duotu laiko momentu, tai matytume animacijos trūkčiojimus.

Norint išlaikyti vientisą netrūkčiojančią animaciją, privalome tiksliai suskaičiuoti tarpinius kadrus. Problemos sprendimo būdas – interpoliacija. Interpoliuojant naudojamos matematines funkcijos, kurios leidžia apskaičiuoti norimas reikšmes. Tai atlikę gausime gerą priartėjimą prie pradinės animacijos su norimu tikslumu pavaizdavimą.

Kita animacijos rūšis yra **raktinių kadrų**. Kad geriau įsisavinus šią animaciją, reiktų įsigilinti į paketais, kaip *„3D Studio Max“*, *„Maya“* arba *„XSI“*, kuriama animaciją. Veikėjo animacija dažniausia būna kuriama nurodant raktinius kadrus, kai kiekvienas kadras apibūdina naujo objekto poziciją erdvėje. Taigi šios rūšies animacijoje nereikia saugoti visų kadrų, pakanka išsaugoti tik raktinius kadrus. Tarpinių kadrų suskaičiavimu rūpinasi animacijos varikliukas.

Raktinių kadrų animacija turi kelias schemas. Viena dažniausių schemų yra ta, kad panaudojus tiesinę interpoliaciją galima suskaičiuoti dingusius kadrus. Skaičiuoti tiesinei interpoliacijai gali būti panaudota ši formulė [9]:

Interpolator=(timevalue-lastkeyframe)/ (nextkeyframe-lastkeyframe);

Interpolated_value=lastvalue(1-Interpolator) + nextvalue*Interpolator;*

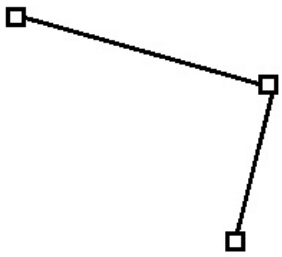
timevalue – laiko reikšmė;
lastkeyframe – paskutinis raktinis kadras;

nextkeyframe – kitas raktinis kadras;
lastvalue – paskutinė reikšmė;
nextvalue – kita reikšmė.

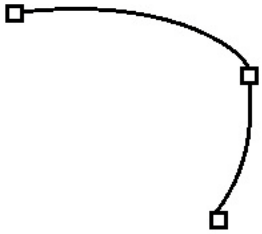
Tiesinę interpoliaciją realizuojame tokiu algoritmu:

```
funkcion Interpolate()
{
float alpha=(currenttime-time1)/(time2-time1);
float alphainv=1-alpha;
point res;
res.x=p1.x*alphainv + p2.x*alpha;
res.y=p1.y*alphainv + p2.y*alpha;
res.z=p1.z*alphainv + p2.z*alpha;
}
```

Tiesinė interpoliacija suteikia animacijai vientisą veiksmą ir nedidelius procesoriaus resursus. Tačiau tiesinė interpoliacija gali neparodyti laukiamų rezultatų. Nemaža dalis animacijos programų pasikliauna galingais interpoliatoriais ir, kai išgaunamas galutinis rezultatas, kai kurie judesiai gali atrodyti ne itin natūralūs.



2 pav. Išgaunamas raktinių kadrų judėjimas animacijoje



3 pav. Norimas judesys animacijoje

Viena iš galimybių pagerinti animacijos kokybę – padaryti taip, kad raktiniai kadrai nebūtų per toli vienas nuo kito ir taip sumažinus kampuotus veiksmus. Matematinė teorema sako, kad bet kokia begalinė kreivė gali būti sutraukta į tiesią liniją, jei intervalas pakankamai mažas.

Labiau patikimesnis metodas, suteikiantis animacijai kokybę, tai automatiškai gauti raktinius kadrus iš didesnės raiškos animacijos sekos, kadangi pasirinkti raktiniai kadrai užtikrina, kad išlaikomas fiksuotas kokybės lygis, be to, minimizuojamas blokinis judėjimas. Programavimo principas labai paprastas: animacija saugoma aukštesniame pavaizdavimo dažnyje ir tada, pasinaudojus analizatoriumi, surandami kadrai, kuriuos būtų galima pašalinti iš animacijos beveik be jokių pašalinių rezultatų galutinei animacijai. Tai galima atlikti iteracijos būdu, kiekvieną kartą pašalinant mažiausiai reikšmingą kadrą, tol kol pasiekiamas užsibrėžtas lygis.

2.5.3. UŽSLĖPTOJI ANIMACIJA

Uždara animacija turi daug gerų savybių nei atviroji animacija [9]. Pirma, šiame animacijos metode tausojama atmintis, nes yra saugoma tik informacija apie sąnario konfigūraciją, o tai yra daug naudingiau nei išsaugoti visą viršūnių tinklą kiekviename raktiniame kadre. Antra, naudodami tokias sistemas galime panaudoti įvairių efektų, tokių kaip fiziškai teisinga animacija, prisitaikymas prie nelygios vietovės bei realaus ryšio tarp aktorių. Taip pat uždaros animacijos sistemos turi savybę keistis animacijos duomenimis tarp skirtingų objektų. Pavyzdžiui, jei animacijoje yra trys skirtingi aktoriai, tai informacija apie sąnarių padėtį ėjimo ciklo metu pakanka išsaugoti tik vieną kartą, kadangi viršūnių padėtis yra suskaičiuojama realiu, esamu laiku.

Didžiausi uždaros animacijos sistemų minusai: realizavimo sudėtingumas ir nemažos skaičiavimo sąnaudos. Suprogramuoti skeletinę animacija yra žymiai sudėtingiau, nei išsaugoti atviros animacijos duomenis. Naudojant fiziškai teisingos animacijos sistemas, susiduriam su lygtimis, kurios nėra labai paprastos, taigi animacijos posistemės taip pat paprašys daugiau procesoriaus resursų.

2.6. ANIMACIJOS METODŲ ANALIZĖS IŠVADOS

1. Kuriant ir pritaikant kompiuterines animacines sistemas konkrečioms programoms ar žaidimams, yra labai svarbūs metodų sprendimai, atsižvelgiantys į programinės ir techninės įrangų specifiką.
2. Kompiuterinėse animacinėse sistemose yra svarbu sukurti mažai energijos eikvojantį variklį, adaptuotą tos srities programoms.
3. Labai svarbus yra kompiuterinės animacijos sistemos lankstumas, t.y. galimybė grafikos variklį ir/ar visą metodą panaudoti efektyviai ir sklandžiai kuo didesniame kompiuterinių programų skaičiuje.
4. Tinkamos animacijos metodo pasirinkimas yra vienas iš svarbiausių kompiuterinės animacijos sistemos efektyvaus veikimo principų, todėl svarbu pasirūpinti daugeliu animacinių sistemų galimybių, trūkumų ir pranašumų vienai kitos atžvilgiu, analize.
5. Kompiuterinės animacinės sistemos dažnai būna kuriamos net kelis metus. Šį laiką labiausiai veikia staigiai besikeičiantys arba nepilnai aprašyti sistemos vartotojų reikalavimai, todėl yra labai svarbu aprašyti tinkamus reikalavimų dokumentus.

3. ANIMACIJOS KŪRIMAS MATEMATINĖJE SISTEMOJE MATHCAD

3.1. MATEMATINĖS SISTEMOS MATHCAD APŽVALGA

MATHCAD yra MathSoft programinės įrangos paketas, specializuotas matematinių, techninių ir ekonominių uždavinių sprendimui [*2,* 2]. Iš panašių žinomų kompiuterinių programinės įrangos paketų, kaip MATLAB, MAPLE, MATHEMATICA bei kitų šis paketas išsiskiria tuo, kad jame uždavinių sprendimo komandos rašomos simboliais, praktiškai nesiskiriančiais nuo klasikinės simbolikos. Nors MATHCAD turi kiek mažesnes galimybes nei žinomi galingesni, specializuoti matematinių uždavinių sprendimui paketai, kaip MATHEMATICA, AKSIOM ar MAPLE, tačiau šios galimybės pakankamai tenkina vartotojų, turinčių matematinio raštingumo lygį, pradedant vidurinės mokyklos abiturientu ir baigiant technologijų mokslo krypties mokslo darbuotoju, poreikius. Vaizdžiau vertinant, MATHCAD galima vadinti XXI amžiaus skaitytuvais ar logaritmine liniuote. MATHCAD panaudojimo efektyvumas priklauso nuo naudotojo matematinio bei kompiuterinio raštingumo lygio. Sprendžiant uždavinius MATHCAD pagalba nebūtina žinoti įvairias skaičiuojamąsias procedūras. Tuo pačiu MATHCAD naudotojas turi geriau suprasti sprendžiamų uždavinių fizinę ar geometrinę prasmę. Galima teigti, kad naudojant MATHCAD negali buti lavinami matematinių uždavinių sprendimo skaičiuojamieji įgūdžiai. MATHCAD turi didelės įvairių matematinių priklausomybių bei uždavinių sprendimo rezultatų geometrinio vaizdavimo galimybės.

3.2. SISTEMOS MATHCAD GALIMYBĖS

Palyginus su MATLAB ir MAPLE, MATHCAD pasižymi lengviau išmokstama ir paprasčiau naudojama vartotojo sąsaja, todėl pirmieji žingsniai būna itin lengvi. Tačiau kuriant sudėtingesnius modelius (bent jau biologijoje) gana greitai ši patogi sąsaja nebepadeda: sudėtingus, realius mokslo uždavinius spręsti skirtos sistemos ne paprastesnės kaip MATLAB ar net tradicinės programavimo kalbos. Paketas turi ir programavimo galimybes, tačiau vartotojo programų darbo sparta ribota, nes tai yra interpretuojanti sistema. Funkcijas MATHCAD galima rašyti ir C (DLL bibliotekos forma), tačiau paprastai darbo grupė, ėmusi programuoti C, pastebi, jog paprasčiau šia kalba užprogramuoti visą užduotį, išskyrus nebent rezultatų vizualizaciją [*2,* 3].

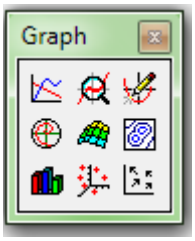
Daug geriau MATHCAD pasiteisina tradicinėmis kalbomis sukurtų modelių rezultatų vizualizavimui, kurį lengva derinti su baigiamąja statistine analize ir kitomis panašiomis operacijomis. Šiuo atveju patys modeliai gali būti vykdomi ir superkompiuteriuose, naudojant lygiagretų programavimą (su MATHCAD tai nėra galima). Draugiška sąsaja šiuo atveju naudojama greitai keisti informacijos pateikimo būdus. Tokiu būdu efektyviai išvengiama daug darbo reikalaujančio grafikos programavimo, kuris biologui ar fizikui didelės fundamentalios mokslinės vertės neturi.

MATHCAD neblogai tinka ir statistinei duomenų analizei, jei ji yra bent kiek sudėtingesnė.

MATHCAD pakete yra patogi informacinė pagalbinė sistema. Šia sistema sudaro iš MATHCAD dialogo lango atidaromas daugelio skyrių žinynas „Resource Center“. Taip pat iš MATHCAD dialogo lango atidarius dialogo langą „Insert Function“ ir jame pažymėjus parinktos funkcijos vardą, pateikiamas trumpas jos aprašymas. Papildomai iš šio dialogo lango gali būti atidarytas informacinės sistemos žinyno puslapis. Šiame puslapyje pateikiama platesnė informacija apie „Insert Function“ lange pažymėta funkciją [8].

3.3. SISTEMOS MATHCAD ANIMACIJOS TYRIMAS PLOKŠTUMOJE (2D)

Beveik visa animaciją MATHCAD matematinėje sistemoje yra braižoma panaudojant komandas, esančias komandų matricoje „*Graph*“ [2]:



4 pav. MATHCAD komandų matrica Graph

 - X-Y Plot (Grafikai Dekarto koordinatėse);

 - Polar Plot (Grafikai polinėse koordinatėse);

 - 3D Bar Plot (Histograma);

 - 3D Scatter Plot (Erdvinės linijos taškai);

 - Surface Plot (paviršiai Dekarto koordinatėse;

 - Zoom, (Mikroskopas – Grafiko dalies išskyrimas);

 - Trace (Pėdsakas);

 - Contour Plot (Lygio linijos Dekarto koordinatėse);

 - Vector Field Plot (Plokščias vektorinis laukas);

Grafikai stačiakampėse koordinatėse braižomi į parinktą ir pažymėtą darbo lapo vietą iš komandų matricos „*Graph*“ perkėlus komanda „*x-y Plot*“. Šiuo veiksmu atidarome grafiko braižymo langą, kuri toliau vadiname *grafiko langu*.

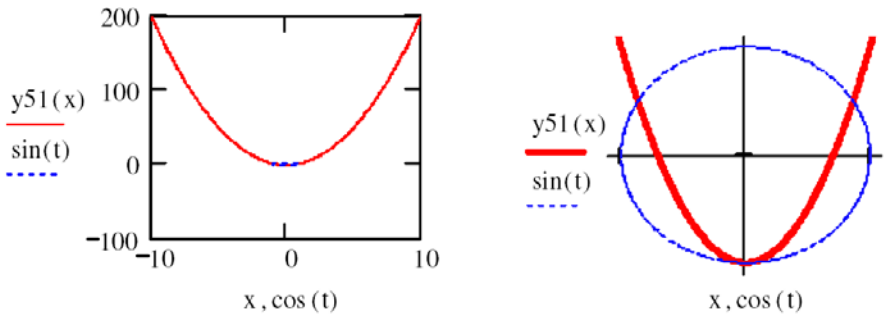
x-y Plot grafiko lange rašomi:

- apačioje esančios žymės vietoje – argumentų vardai. Jei viename brėžinyje braižomi keleto funkcijų grafikai, tai kiekvienos funkcijos argumentų vardai atskiriami **„ , “** ženklu;
- kairėje esančios žymės vietoje rašomi funkcijų vardai arba jų išraiškos.
    - Jei norima rašyti tik funkcijų vardus, tai jų išraiškos turi būti užrašytos aukščiau grafiko lango.
    - Jei viename brėžinyje braižomi keleto funkcijų grafikai, tai jų vardai arba išraiškos atskiriamos **„ , “** ženklu.

Nubraižome funkcijų $y51 = 2x^2 - 1$ ir $\begin{cases} x = \cos t \\ y = \sin t \end{cases}$ grafikus. Atlikę nurodytus veiksmus gauname 5 pav. kairėje esančiame grafiko lange pavaizduotus grafikus. Šiuos grafikus iki pavidalo, pavaizduoto 5 pav. dešinėje esančiame grafiko lange, pertvarkome tokiais veiksmais [9]:

1. |brėžinio langas, < >|;
2. Ištriname argumentų intervalo galų koordinates -10 ir 10 ir vietoje jų įrašome -1.1 ir 1.1;
3. Ištriname funkcijų intervalo galų koordinate -1 ir 199 ir vietoje jų įrašome -1.1 ir 1.1;
4. |brėžinio langas, << >++>|.
5. Atliekame veiksmus atidarytame *“Formating Currently...“* lange:
    5.1. atidarę dalį „*X-Y Axes*“:
    - panaikiname „*X – Axis*“ žymę *„Numbered“;*
    - panaikiname „*Y – Axis*“ žymę *„Numbered“;*
    - žymę *„Axis Style“* perkeliame į poziciją *„Crossed“*.

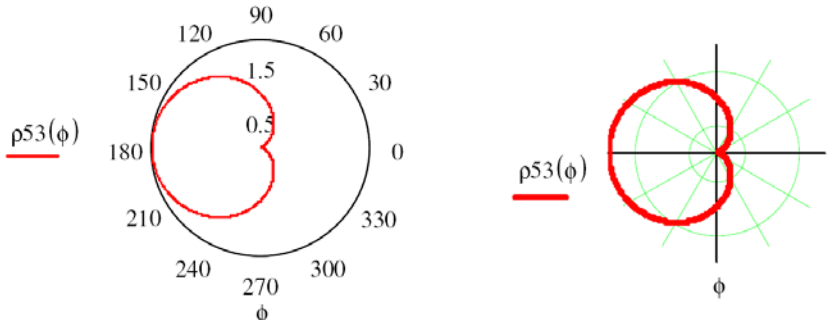
    5.2. atidarome dalį *„Traces“* ir pakeičiame grafikų linijų storį:
    - *trace 1 Weight 3;*
    - *trace 2 Weight 3.*
6. Veiksmus baigiame atidaryto lango komanda **„***OK***“.**



5 pav. MATHCAD funkcijų grafikai OXY koordinatėse

Grafikai polinėse koordinatėse braižomi analogiškai kaip ir stačiakampėse koordinatėse panaudojant komandų matricos *„Graph“* komandą *„Polar plot“*. Analogiškai, kaip ir ankstesniame pavyzdyje veiksmais, panaudoję komandą *„Polar plot“* nubraižome funkcijos $\rho(\varphi) = 1 - \cos\varphi$ grafiką. Tiesiogiai panaudoję komandą *„Polar plot“* grafiką gauname tokį, koks jis pavaizduotas 6 pav. kairėje. Šį grafiką iki pavidalo, pavaizduoto 6 pav. dešinėje pertvarkome taip [9]:

- grafiko lange du kartus spustelime kairįjį pelės klavišą;
- dialogo lange *„Formating Currently...“* atliekame veiksmus:
  - skyrelyje dalyse *„Radial“* ir *„Angular“*, *„Polar Axes“* panaikiname žymes *„Numbered“* ir pažymime *„Grid Lines“;*
  - dalyje *„Axis Style“* pažymime *„Crossed“*.
- atidarę skyrelį *„Traces“* pakeičiame pirmos linijos *„Trace1“* storį *„Weight“* iki reikšmės *„3“*.

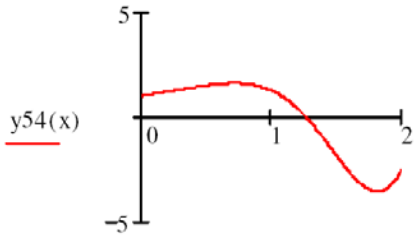


6 pav. MATHCAD grafikai polinėse koordinatėse

Grafinis lygčių $f(x) = 0$ grafinis sprendimas atliekamas panaudojant komandų matricos „*Graph*“ komandas *„x-y Plot“, „Zoom“* ir „Trace“. Grafinio lygties $f(x) = 0$ sprendimo nagrinėjimą atlikome lygties $2x\cos x^2 - \sin x = 1$ pirmos teigiamos šaknies skaičiavimo pavyzdžiu. Lygties $2x\cos x^2 - \sin x - 1 = 0$ pirmą teigiamą šaknį apskaičiavome taip [9]:

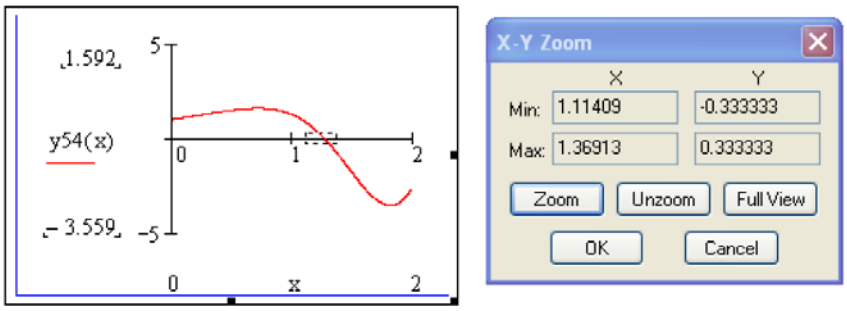
1. Užrašėme funkciją:

$$y54 = 2x\cos(x^2) - \sin(x) + 1$$



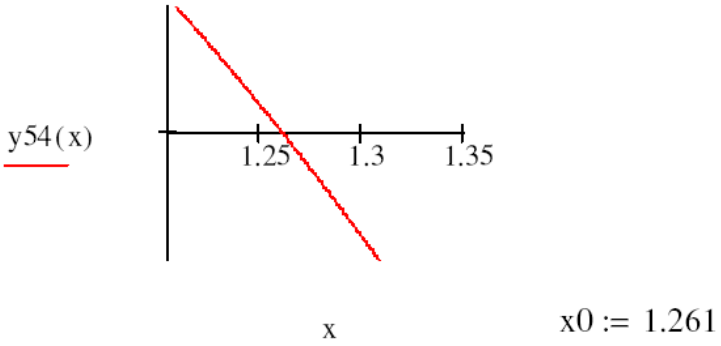
7 pav. Funkcijos $y54 = 2x\cos(x^2) - \sin(x) + 1$ *grafikas*

1. Veiksmais, analogiškais atliktais sprendžiant ankstesnį pavyzdį, nubraižėme užrašytos funkcijos grafiką, pavaizduotą 7 pav.;
2. Spustelėjome kairįjį pelės klavišą grafiko lange;
3. Spustelėję kairįjį pelės klavišą ties komandų matricos *„Graph“* antra piktograma *„Zoom*  *„* atidarome dialogo langą *„x-y Zoom“;*
4. Pelės rodykle grafiko lange perkėlę ties tašku, kuriame funkcijos grafikas kerta argumentų $(0x)$ ašį ir paspaudę kairįjį pelės klavišą apibraukiame nedidelę sritį apie šį tašką (8 pav.);



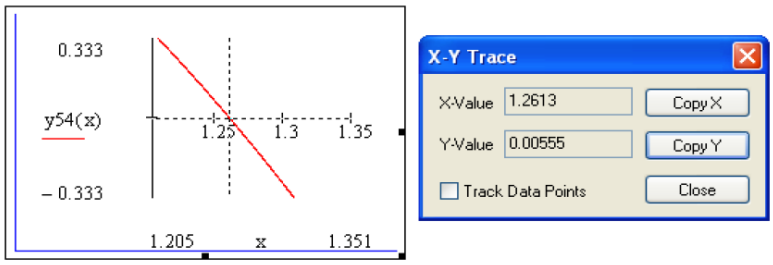
8 pav. Funkcijos $y54 = 2x\cos(x^2) - \sin(x) + 1$ *grafikas*

5. Pelės rodykle perkėlę dialogo lange *„x-y Plot“* ties žyme *„OK“* ir spustelėję kairįjį pelės klavišą grafiko lange gauname 9 pav. pavaizduotą grafiką;



9 pav. Funkcijos y57(x) grafikas ties tašku y54:=0

6. Pakartojame 4 veiksmą ir pelės rodykle perkėlę ties komandų matricos *„Graph“* trečią piktogramą ir spustelėję kairįjį pelės klavišą atidarome dialogo langą *„x-y Trace“;*



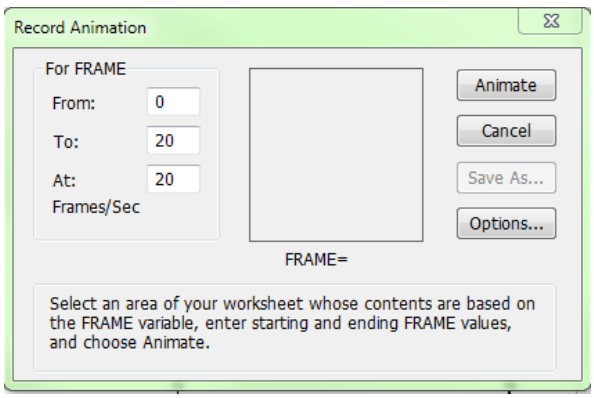
10 pav. Funkcijos $y54 = 2x\cos(x^2) - \sin(x) + 1$ *grafikas*

7. Pelės rodyklę grafiko lange perkėlę į tašką, kuriame funkcijos grafikas kerta argumentų $(0x)$ ašį ir spustelėję kairįjį pelės klavišą dialogo lange *„x-y Trace“* langelyje *„X-Value“* gauname lygties $2x\cos x^2 - \sin x - 1 = 0$ sprendinio reikšmę x =1.2613;
8. Šia reikšmę darbo puslapyje nukopijuojame.

Animation yra MATHCAD funkcija dinamiškam grafikų atvaizdavimui. Tai yra sudėtingesnė ir daugiau resursų reikalaujanti matematinio paketo funkcija. Komanda *Animation* iš meniu srities *Tools* turi du etapus [4]:

1. *Record...* – vienas po kito įrašomi kadrai (frames) grafiko animacijai atvaizduoti;
2. *Playback...* – rodomas iš įrašytų kadrų (frames) sudarytas filmas.

Animation yra skirta kurti animacinei grafikai tiek plokštumoje (2D), tiek ir erdvėje (3D). Įvykdžius komandą *Tools–>Animation–>Record...* MATHCAD programa atveria *Record Animation* dialogo langą (11 pav.).



11 pav. Record Animation dialogo langas

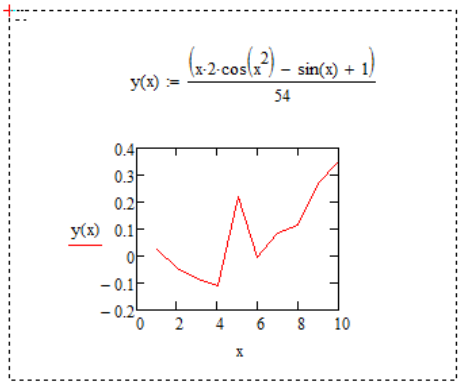
Atsidariusiame lange siūloma nurodyti kintamojo kadro (frame) kitimo sritį, kur *From:* būtų pradinė reikšmė, o *To:* nusakytų galinę reikšmę. Taip pat galima pasirinkti kadrų rodymo kiekį per sekundę - *Frames/Sec*. Dažniausiai pagal nutylėjimą būna, kad *From: 0*, *To:*

9 ir *Frames/Sec=10*, tai reiškia, kad bus dešimt kadrų, kurie bus atvaizduoti per vieną sekundę.

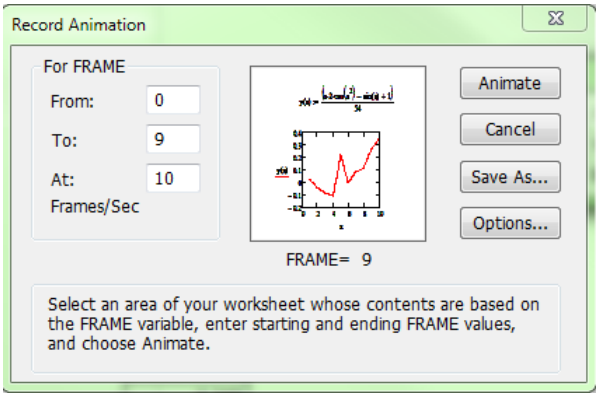
Mygtukas *Animate* įrašo vaizdus (*frame*) ir juos parengia animacijos vykdymui. *Cancel* – nutraukia komandą bei užveria dialogo langą. *Save As* – išsaugo animacijos įrašą (.avi tipo failą), pastarasis mygtukas būna aktyvus tik tada, kai yra ką saugoti. Kadangi animacijos įrašai užima daug atminties, sistema MATHCAD juos išsaugo suspaustus. Mygtukas *Options* leidžia pasirinkti suspaudimo metodą arba visai jo atsisakyti.

Nustačius *frame* kitimo sritį bei dažnį, darbo lape pažymima sritis (brėžinys), kuriai bus vykdoma animacija. Sritis žymima judinant pelytę, prieš tai nuspaudus jos kairįjį klavišą. Į šią sritį gali patekti ne tik grafikas ar jo fragmentas, bet ir papildoma informacija: formulės, duomenys, komentarai ir t.t. (12 pav.). Nurodžius grafiką, kuriam bus taikoma animacija, dialogo lange renkamės mygtuką *Animate*. Ekrane matome kuriamus kadrus ir jų eilės numerius (13 pav.).

Dešimties kadrų sukūrimas užtrunka apie vieną sekundę.

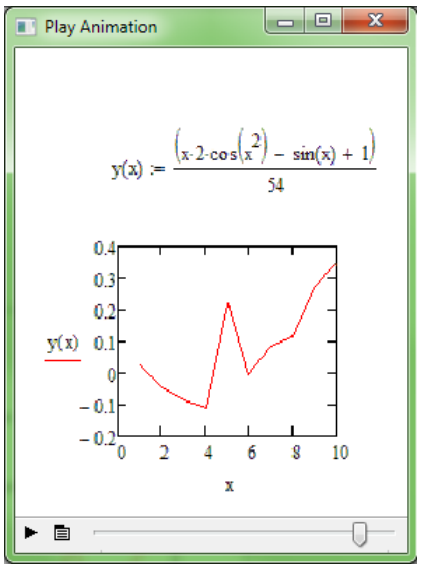


12 pav. Grafiko srities žymėjimas



13 pav. Animacijos kadrų kūrimas

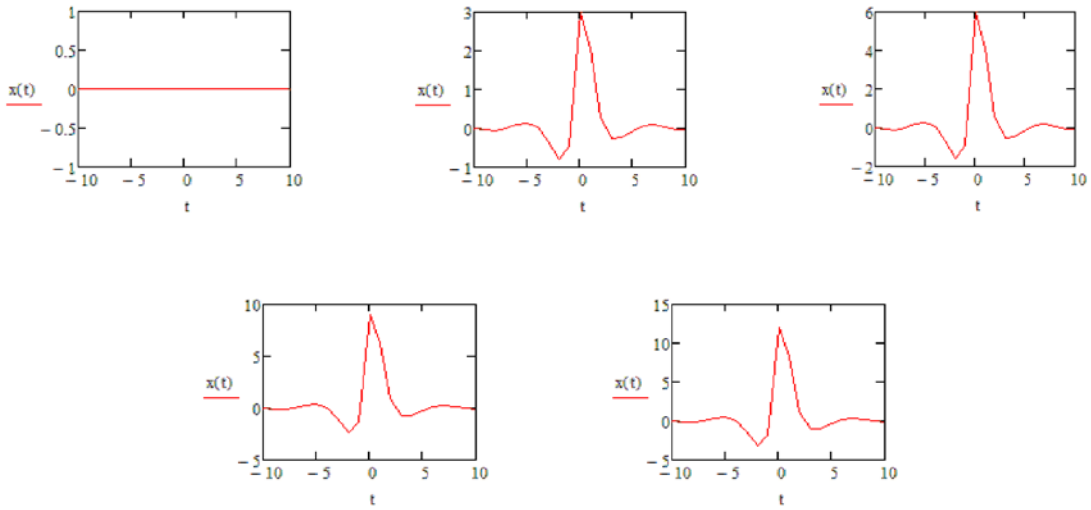
Pabaigus kadrų saugojimą, ekrane iššoka animacijos vaizdavimo dialogo langas „*Play Animation*„(14 pav.). Tame lange parodoma nustatyta animacijos sritis. Animacija kontroliuojama keliais mygtukais. Mygtukas „*Play* ▶“ skirtas animacijai paleisti, ▤ – papildomų komandų menių sąrašui išskleisti, kuriame randama „*View*“ – skirta peržiūros lango dydžio nustatymui, „*Speed*“ – peržiūros greičio keitimui, „*Open*“ – sukurto ir išsaugoto to paties tipo failo atidarymui, „*Close*“ – animacijos peržiūros lango uždarymui, „*Copy*“ – padaryti animuojamo grafiko kopiją [4].



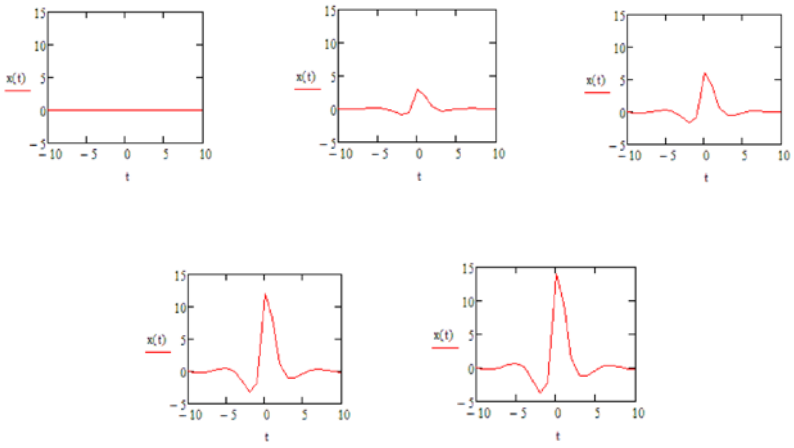
14 pav. Animacijos kadrų kūrimas

Didelį dėmesį reikėtų atkreipti į funkcijos apibrėžimo bei reikšmių sričių nurodymus, nes jos gali keisti automatiškai, o tai pakeistų atvaizdus animacijos veikimo metu. Pagal nutylėjimą grafiko mastelis parenkamas taip, kad funkcija tilptų į grafiką. Animacijos veikimo metu kintant funkcijai, taip pat keičiasi ir mastelis: „didėjant“ funkcijai, „didėja“ mastelis, tada gauname iškreiptą grafiko kitimo atvaizdą (12 pav.). Pavyzdžiui turime formulę [9]:

$$x(t) := (\sin(t) + \cos(t)) \cdot 5 \frac{FRAME}{t^2+1}$$



15 pav. Animaciją sudarantys kadrai, kai funkcijos apibrėžimo sritis kinta automatiškai.

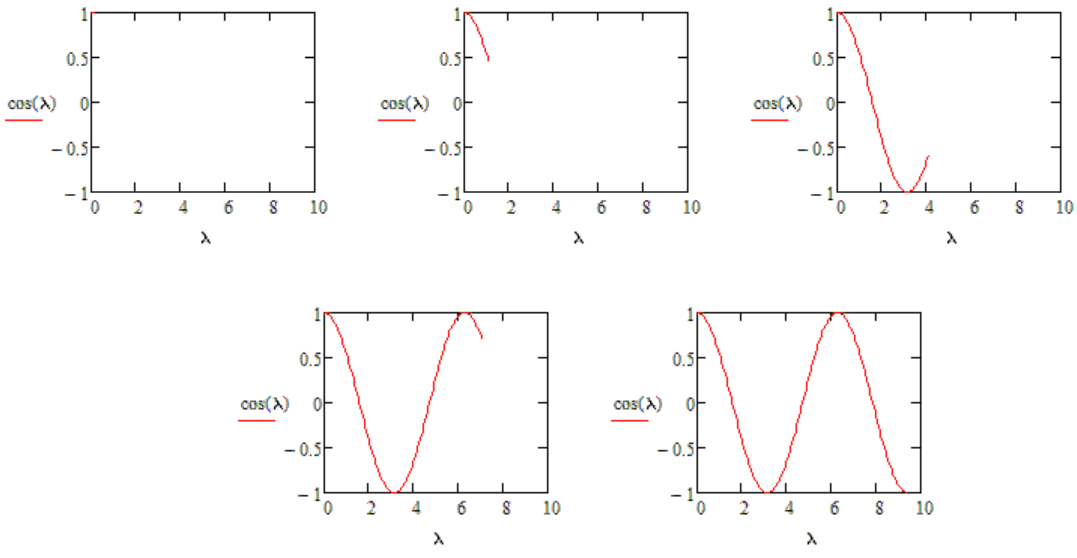


16 pav. Animaciją sudarantys kadrai, kai funkcijos apibrėžimo sritis yra pastovi.

Matematinėje sistemoje MATHCAD galima matyti grafiko braižymo etapų animaciją. Ši animacija vykdoma panaudojant jau minėtą komandą [4] *Tools –> Animation–>Record...*, kurią išbandėme funkcijos kitimo plokštumoje atvaizdavimui. Norint nubrėžti kreivę ekrane, privalome kintamąjį FRAME panaudoti tada kai yra priskiriama funkcijos apibrėžimo sritis. Tada kai FRAME keičiasi nuo 0 iki pasirinktos reikšmės, tai apibrėžimo sritis laipsniškai didėja ir taip gauname funkcijos braižymo efektą. Tarkime turime apibrėžimo sritį:

$$\lambda := 0, \frac{\pi}{60} .. \frac{FRAME + 1}{60} \cdot 3\pi$$

Pasirenkame komandą *Graph –> „X-Y Plot“*. Aprašome braižomą funkciją cos(λ) ir jos reikšmių sritį. Atlikus šiuos veiksmus pradedam kurti animacija. Kurdami animaciją aprašome FRAME kitimo sritį nuo 0 iki 59, tai reiškia, kad animaciją sudarys 60 kadrų, kurie bus atvaizduoti per 6 sekundes po 10 kadrų per sekundę. Sinusoidė yra atvaizduojama intervale nuo 0 iki 3π.



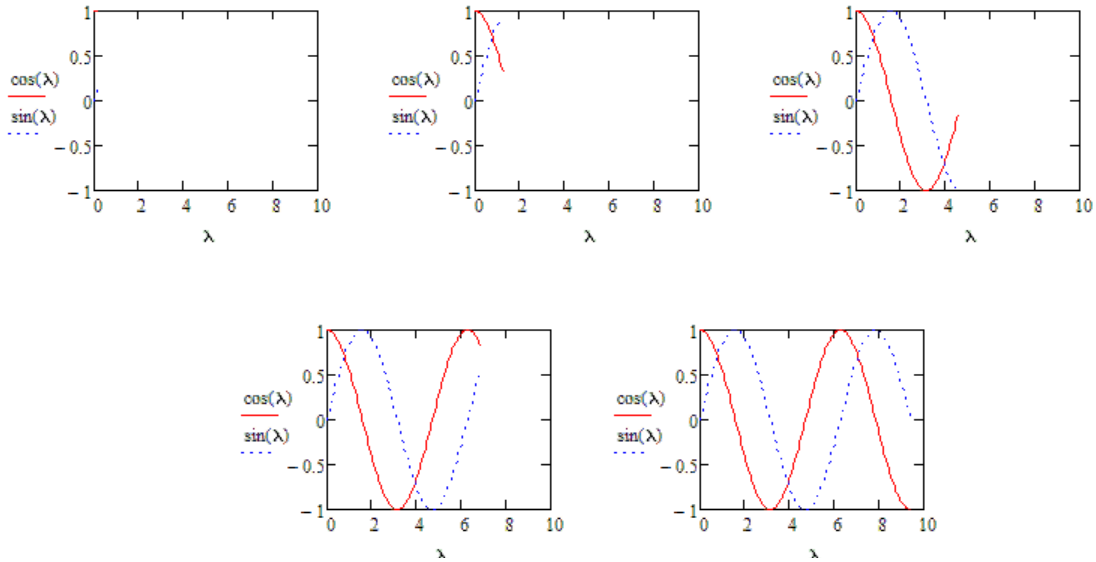
17 pav. Animaciją sudarantys kadrai, kurie parodo sinusoidės brėžimą

Taip pat yra galima braižyti kelias funkcijas vienu metu. Toks atvaizdavimo būdas nepakeičiamas, kai reikia supažinti su funkcijų grafikų transformacijomis, lyginėmis ir nelyginėmis funkcijomis, funkcijos periodiškumu, tolydumu, asimptotėmis ir kt. savybėmis.

Paėmėme prieš tai naudotą funkcijos sritį:

$$\lambda := 0, \frac{\pi}{60} .. \frac{FRAME + 1}{60} \cdot 3\pi$$

Tik dabar šią sritį naudosime dviem funkcijoms apibrėžti: sin(λ) ir cos(λ).



18 pav. Animacijos taikymas dviem funkcijoms

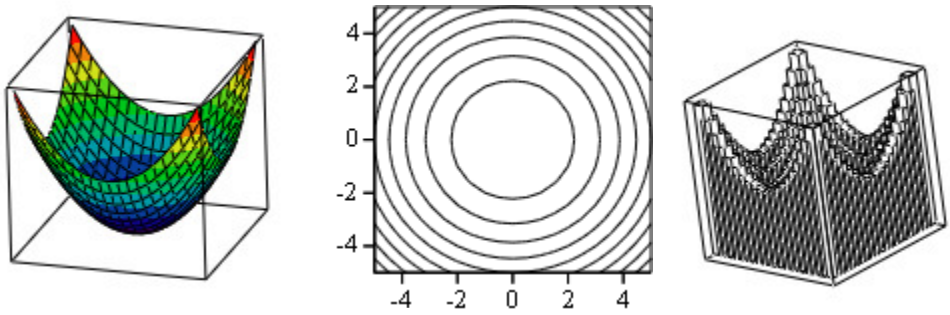
3.4. SISTEMOS MATHCAD ANIMACIJOS TYRIMAS ERDVĖJE (3D)

Animacija erdvėje (3D) kuriama taip pat kaip ir plokštumoje (2D) [3]. Tik reikia pasirinkti tinkamą grafiko tipą. Pavyzdžiui funkciją $z = f(x,y)$ atitinkantis paviršius braižomas komanda *„Surface Plot "*. Funkcijos $z = f(x,y)$ lygio linijos $f(x, y) = C_i$ braižomos komanda *„Contour Plot "*.Funkciją $z = f(x,y)$ atitinkanti histograma braižoma komanda *„3D Bar Plot"*.

Šiomis komandomis funkcijų grafikai braižomi MATHCAD lape užrašius jų vardus ir jų išraiškas. Ne aukščiau šio užrašo iš komandų matricos *„Graph"* atitinkama komanda perkeliama reikalingas grafiko langas. Grafiko lange esančios žymės vietoje užrašomas funkcijos vardas ir taip gaunamas nagrinėjamos funkcijos grafikas.

Pavyzdžiui funkcijos $z = x^2 + y^2$ paviršiaus, lygio linijų ir histogramos grafikai nubraižomi taip [9]:

$$z54(x,y) := x^2 + y^2$$



19 pav. Funkcijos z := $x^2 + y^2$ a – grafikas; b – lygio linijos; c – histograma

Komanda *„3D Scater Plot“* braižomi erdvės taškų grafikai (erdvinių linijų grafikai).

Erdvinė linija $\begin{cases} x = h(t) \\ y = g(t) \\ z = r(t) \end{cases}$ braižoma atliekant tokius veiksmus:

- užrašomas erdvės linijos taškų skaičius *N* ir argumento *t* reikšmių t_i , $i \in \overline{1, N}$ išraiška;
- užrašomos erdvės koordinačių *x*, *y*, *z* vardai ir išraiškos;
- užrašomos erdvės taškų koordinačių reikšmės esant argumento reikšmėms t_i;
- iš komandų matricos „*Graph*“ komanda *„3D Scater Plot“* perkeliamas grafiko langas, kuriame esančios žymės vietoje skliaustuose surašomi erdvės koordinačių vardai.

Pavyzdžiui spiralės $\begin{cases} \mathrm{x} = 2(\exp(-0{,}05\mathrm{t}))\sin \mathrm{t} \\ \mathrm{y} = 2(\exp(-0{,}05\mathrm{t}))\cos \mathrm{t} \\ \mathrm{z} = 2\mathrm{t} \end{cases}$ grafikas braižomas taip:

N := 100 i := 0 ..N $t_i := i \cdot 4 \cdot \frac{\pi}{N}$

x542(t) := 2· (exp(-0.05·t)) ·sin(t)

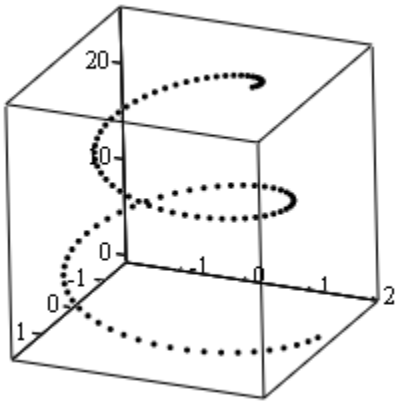
y542(t) := 2· (exp(-0.05·t)) ·cos(t)

z542(t) := 2·t

$x542_i := x542(t_i)$

$y542_i := y542(t_i)$

$z542_i := z542(t_i)$



20 pav. Spiralė

Parametrinė forma $\begin{cases} x = h(u, v) \\ y = g(u, v) \\ z = r(u, v) \end{cases}$ užrašyto paviršiaus grafikas braižomas atliekant tokius veiksmus:

- užrašomi erdvės koordinačių *x*, *y*, *z* vardai ir jų, kaip *u* ir *v* argumentų funkcijų, išraiškos;

- iš komandų matricos *„Graph“* komanda *„Surface Plot“* perkeliamas grafiko langas, kuriame esančios žymės vietoje skliaustuose surašomi erdvės koordinačių vardai.

Pavyzdžiui paviršiaus $\begin{cases} x = (10 + \cos u)\cos v \\ y = (10 + 2\cos u)\sin v \\ z = v \end{cases}$ grafikas braižomas taip:

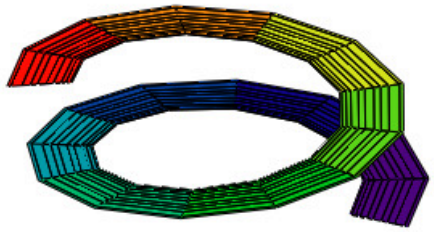
a := 10

b := 2

x543(u,v) := (a + b·cos(u)) ·cos(v)

y543(u,v) := (a + b·cos(u)) ·sin(v)

z543(u,v) := v



21 pav. Paviršiaus x = (10 + cosu)cos v, y = (10 + 2cosu)sin v, z = v grafikas

Komanda *„Graph“ „Vector Field Plot“* braižomas vektorinio lauko $\overrightarrow{F(x,y)} = (A(x,y), B(x,y))$ grafikas. Vektorinio lauko grafikas braižomas atliekant tokius veiksmus:

- užrašomi vektoriaus lauko vardas ir jo dedamųjų išraiškos,
- užrašomos kintamųjų *x* ir *y* reikšmių $_i x$ ir y_i intervale [*A*,*B*] , [*C*,*D*] apskaičiavimo išraiškos,
- užrašomos vektorinio lauko dedamųjų, apskaičiuotų taškuose $M_{ij}(x_i, y_j)$ reikšmės,
- iš komandų matricos *„Graph“* komanda *„Vector Field Plot“* atidaromas grafiko langas, kuriame esančios žymės vietoje skliaustuose įrašomi vektorinio lauko dedamųjų vardai.

Pavyzdžiui funkcijos $z = x + y - x^2 - y^2$ gradientinis $\overrightarrow{gradz} = (z'_x, z'_y)$ laukas braižomas taip:

$z544(x,y) := x + y - x^2 - y^2$

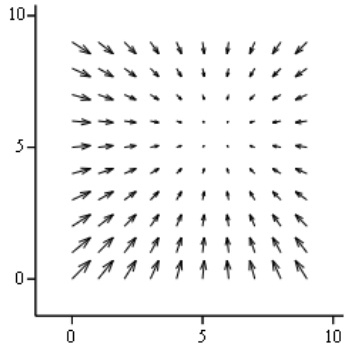
$$gradz544(x,y) := \begin{pmatrix} \frac{d}{dx} z544(x,y) \\ \frac{d}{dy} z544(x,y) \end{pmatrix}$$

A := –5 B := 5 C := –5 D := 5 Nx := 10 Ny := 10

$i := 0 \,..Nx - 1 \qquad x_i = A + \frac{B-A}{Nx} \cdot i$

$j := 0 \ldots Ny - 1 \qquad y_j = C + \frac{D-C}{Ny} \cdot j$

$V544_{i,j} := gradz544(x_i, y_j) \qquad A544_{i,j} := (V544_{i,j})_0 \qquad B544_{i,j} := (V544_{i,j})_1$



22 pav. Funkcijos $z = x + y - x^2 - y^2$ gradientas

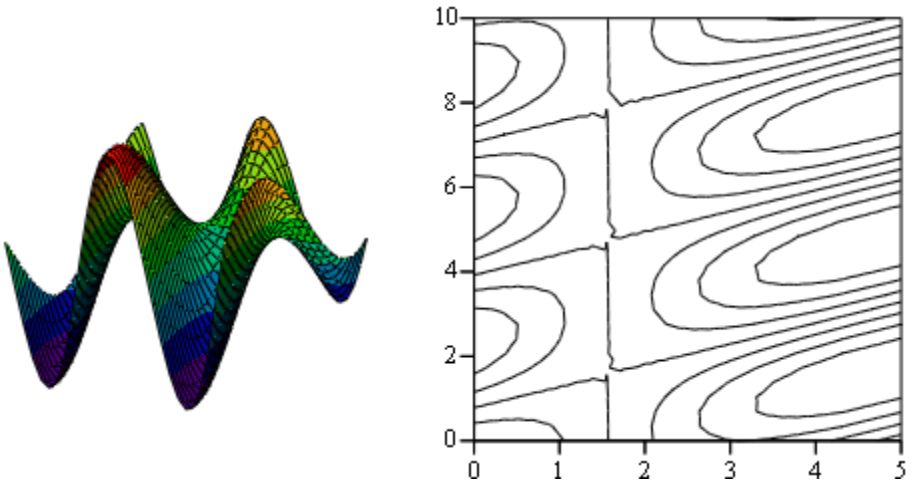
Dviejų kintamųjų funkcijos $z = f(x, y)$ grafikas parinktoje srityje $x \in [a,b]$, $y \in [c,d]$ braižomas naudojant *„Insert Function“* lange skyriuje *„Function Category“* – *„Graph“* esančią funkcij1 *„CreateMesh“*. Funkcija *„CreateMesh“* $z = f(x, y)$ grafikai braižomi tokiais veiksmais:

- užrašomas funkcijos vardas ir išraiška;
- užrašomas $z = f(x, y)$ grafiko braižymo funkcijos vardas ir po priskyrimo simbolio „:=“ iš *„Insert Function“* lango perkeliama funkcija *„CreateMesh“;*
- funkcijos *„CreateMesh“* šablone esančių žymiu vietose įrašoma:
    - pirmos – nagrinėjamos funkcijos vardas;
    - antrosios ir trečiosios – parinktos pirmojo kintamojo intervalo apatinė ir viršutinė reikšmės;
    - ketvirtosios ir penktosios – parinktos antro kintamojo intervalo apatinė ir viršutinė reikšmės;
    - šeštosios – parinkta taškų skaičiaus reikšmė; septintoji ir aštuntoji žymės ištrinamos.

Pavyzdžiui funkcijos $z = cos(y - x) - sin\ y$ intervale $x \in (0;5)$, $y \in (0;10)$ ir jos lygių linijų grafikai braižomi taip:

F545(x,y) := cos(y − x) − sin(y)

P := CreateMesh(F545 , 0, 5, 0, 10, 30)



23 pav. Funkcijos z = cos(y − x)− sin y ir jos lygių linijų grafikai

Funkcijos $\begin{cases} x = F(u,v) \\ y = G(u,v) \\ z = H(u,v) \end{cases}$ grafikai srityje $u \in (A;B)$, $v \in (C;D)$ funkcijos *„CreateMesh“* pagalba braižomi atliekant tokius veiksmus [9]:

- užrašomi erdvės koordinačių x, y, *z* vardai ir jų funkcinės priklausomybės nuo *u* ir *v* išraiškos;
- užrašomas nagrinėjamos funkcijos grafiko braižymo funkcijos vardas ir po priskyrimo simbolio „:=“ iš *„Insert Function“* lango perkeliama funkcija *„CreateMesh“;*
- funkcijos *„CreateMesh“* šablone esančių žymių vietose įrašoma:
  - pirmos, antros ir trečios – erdvės koordinačių funkcijų vardai;
  - ketvirtos ir penktos – pirmo parametro *u* parinkto intervalo apatinė ir viršutinė reikšmės;
  - šeštos ir septintos – antro parametro *v* parinkto intervalo apatinė ir viršutinė reikšmės;
  - aštuntos – parinkta taškų skaičiaus reikšmė.

Pavyzdžiui sukinių, gautų sukant apie 0*x* ir 0*y* ašis kreivė $y = x \cos x$, $x \in [0;1]$, grafikai braižomi taip:

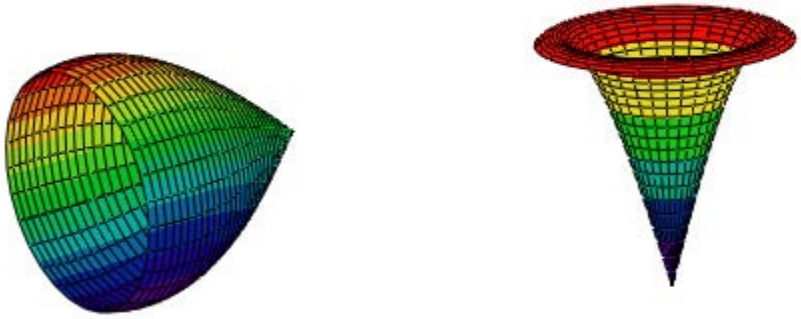
y(x) := x·cos(x) *x*

F5x(u,v) := u G5x(u, v) := y(u) ·cos(v) H5x(u,v) := y(u) ·sin(v)

S5x := CreateMesh(F5x,G5x,H5x,0 ,1 ,−π ,π , 30)

F5y(u,v) := u·sin(v) G5y(u,v) := u·cos(v) H5y(u,v) := y(u)

S5y := CreateMesh(F5y,G5y,H5y,0 ,1 ,−π ,π , 30)



S5x S5y

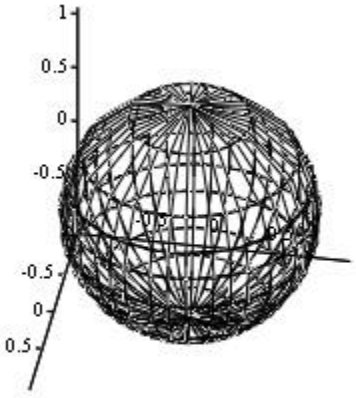
24 pav. Sukinių apie ašis 0x (S5x) ir 0y (S5y) grafikai

Matematinėje sistemoje MATHCAD galima brėžinį erdvėje vartyti visomis krytimis, tereikia užeiti ant brėžinio ir paspausti kairįjį pelytės klavišą. Tokiu būdu paviršių galima apžiūrėti iš įvairiausių kampų. Nupieškime rutulį [4, 9]:

$X(u,v) := \sin(v)\cdot\cos(u)$

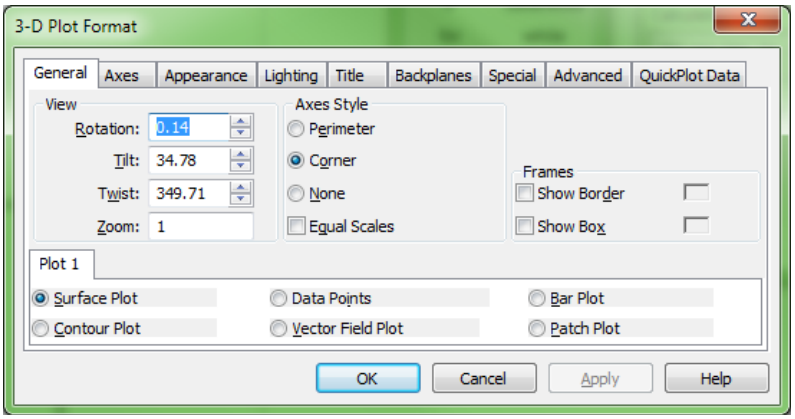
$Y(u,v) := \sin(v)\cdot\sin(u)$

$Z(u,v) := \cos(v)$



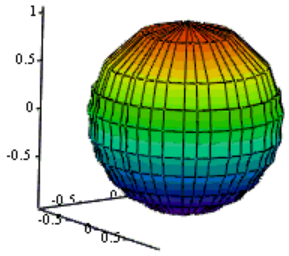
25 pav. Rutulys

Du kartus spustelėjus ant nubrėžto paraboloido 3D grafiko pamatytumėte *3-D Plot Format* nustatymų langą:



26 pav. 3-D Plot Format dialogo langas

Spustelėjus *Appearance* kortelę, joje randame ir pažymime *"Fill surface"* ir *"Colormap"*. Pažymėję brėžinį ir nuspaudę kairį pelės mygtuką galime pastumdyti jį. Pamatome, kaip keičiasi brėžinio vaizdas.



27 pav. Nuspalvintas ir pakreiptas rutulys

Kaip ir plokštumoje, taip ir erdvėje galime kurti animaciją iš kadrų. Kūrimo principas ir metodas yra identiški. Kaip pavyzdį paimkime, kad:

$\alpha := 0..\pi$

$\beta := 0..2\pi$

$X(\alpha,\beta) := \sin(\alpha)\cdot\cos(0)\cdot FRAME$

$Y(\alpha,\beta) := \sin(\alpha)\cdot\sin(0)\cdot FRAME$

$Z(\alpha,\beta) := \cos(\alpha)$

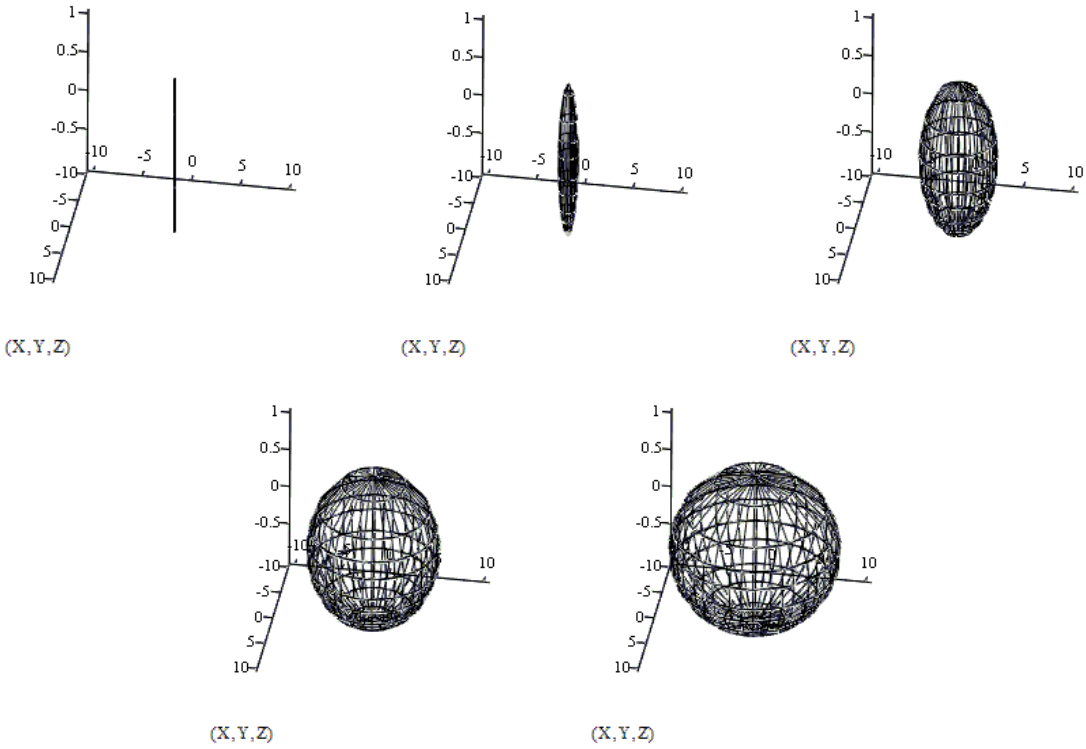
Ant grafiko paspaudus dešinį pelės klavišą, renkamės komandą *Properties.* Kortelėje *Axes* reikia nurodyti grafiko mastelį:

X-Axis – Minimum Value = -10, Maximum Value = -10,

Y-Axis – Minimum Value = -10, Maximum Value = -10,

Z-Axis – Minimum Value = -1, Maximum Value = 1.

FRAME kinta nuo 0 iki 9.



28 pav. Funkcijos animacijos erdvėje kadrų aibė

3.5. SISTEMOS MATHCAD ANIMACIJOS PLOKŠTUMOJE (2D) IR ERDVĖJE (3D) TIKSLUMAS IR PAKLAIDOS

Atliekant skaičiavimus su kuriuo nors matematiniu programiniu paketu, susiduriame su skaičiavimo paklaidomis. Visas paklaidas galima suskirstyti į šias grupes [9]:

- absoliučioji;
- santykinė;
- kvadratinė paklaida;
- vidutinė kvadratinė paklaida;
- vidutinė absoliučioji.
- standartinė;
- skaičiavimo paklaida;
- atsitiktinė paklaida;
- sisteminė paklaida;

Programinis paketas MATHCAD atlieka skaičiavimus 0,001 tikslumu, jei specialiai nėra nurodyta, kokiu tikslumu programa turi skaičiuoti. Tačiau skaičiavimo tikslumą programiniame pakete MATHCAD galima reguliuoti, panaudoję operatorių *TOL* (*Tools* ->

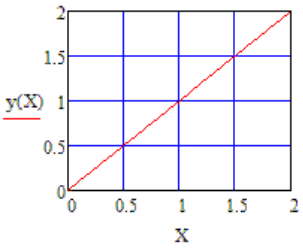
Worksheet Options... -> Built-In Variables): pvz. įvedę *TOL = 0.0001*, visi skaičiavimai užrašyti po šios išraiškos, bus skaičiuojami nurodytu tikslumu.

Naujausiose programinio paketo MATCAD versijose galima atlikti skaičiavimus 10^{-15} tikslumu (t.y, galima apskaičiuoti 15 dešimtainių ženklų). Spragtelėjus du kartus ant ekrane užrašyto skaičiaus, galime pamatyti kiek skaičių po kablelio yra rodoma monitoriuje. Tačiau šio užrašo nereikia painioti su skaičiavimo tikslumu.

Paėmėme pavyzdį:

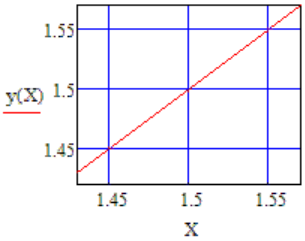
$X := 0, .1 .. 2$

$y(X) := X$



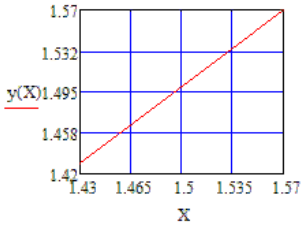
29 pav. Funkcijos y(X):=X grafikas

Dabar prisiartinkime grafiką įvesdami grafiko srities ribas (1,42-1,57 Y kryptimi ir 1,43-1,57 X kryptimi). Atkreipkime dėmesį, kad ašių ribos X ir Y yra ne visai tokios pačios. Atlikus šiuos veiksmus grafikas vis dar atrodo gerai, nes tiesė eina per koordinates 1,5 ir 1,5.



30 pav. Funkcijos y(X):=X grafikas kai jo atvaizdavimo sritis yra 1,42-1,57 Y kryptimi ir 1,43-1,57 X kryptimi

Dabar išjungiame automatinį tinklelį ir nustatom 4x4 narvelių tinklą. Dabar grafikas nerodo teisingai! Kai x = 1,5 vis tiek turėtumėte pamatyti y = 1,5:



31 pav. Funkcijos y(X):=X grafikas kai jo atvaizdavimo sritis yra 1,42-1,57 Y kryptimi ir 1,43-1,57 X kryptimi ir išjungtas AutoGrid

Grafikas atrodo, kad būtų atvaizduotas netiksliai, tačiau šis netikslumas yra dėl to, kad pažymėtas apvalinimas kryptims. Nors MATHCAD automatiškai skirsto kiekvieną ašį į 4 intervalų tinklą, tačiau žymės atrodo nevienodai išdėstytos, nes ribos yra skirtingos tarp dviejų ašių. Antrasis narvelis ant kiekvienos ašies faktiškai prasideda nuo:

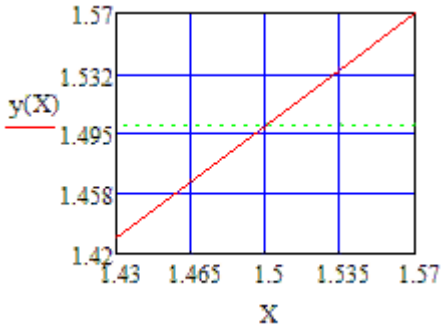
X ašis:

$$(1{,}57 - 1{,}43) \cdot \frac{2}{4} + 1{,}43 = 1{,}5$$

Y ašis:

$$(1{,}57 - 1{,}42) \cdot \frac{2}{4} + 1{,}42 = 1{,}495$$

Deja, šis skirtumas yra toks mažas, kad dingsta, kai MATHCAD apvalina ribas iki dviejų reikšminių skaitmenų. 1,495 suapvalina iki 1,5, dėl ko atsiranda netikslumai grafike. Skaičių formatavimo skirtuke galime sritis padaryti tiksliau. Dukart spustelėję ant grafiko atidarome formatavimo *XY Plot* dialogo langą. Pasirinkame *Number Format*, tada nustatome skaitmenų po kablelio skaičių iki 3. Kai MATHCAD perpiešia grafiką, tinklelio linijos nejuda, tačiau jų vertybės atsispindi tiksliau.



32 pav. Funkcijos y(X):=X grafikas

Tokios pat problemos išlieka ir grafikams trimatėje erdvėje (3D) [9]. Taip pat erdvėje esantis brėžinys gali būti neteisingai suprastas ir įvertintas, kadangi tai priklauso nuo objekto matymo kampo. Stebėjimo kampas gali tik vizualiai iškreipti rezultatus, tačiau tikrųjų rezultatų tikslumas priklauso tik nuo grafiko sričių nustatymo ribų.

3.6. SISTEMOS MATHCAD ANIMACIJOS PLOKŠTUMOJE (2D) IR EDVĖJE (3D) IŠVADOS

1. Matematinis paketas MATHCAD yra daugybės inžinierių, mokslininkų ir studentų darbo ar mokslo įrankis. Šiame pakete yra galimybė kurti iki septynių tipų grafikus (plokštumoje, erdvėje, polinėje koordinačių sistemoje, kontūrinius, stulpelinius ir kt.) ir jiems visiems pritaikyti animaciją. MATHCAD pakete visiems grafikams skirta viena komanda animacijai kurti.
2. MATHCAD programinės įrangos simbolika nesiskiria nuo simbolikos įprastos matematikoje ir gali būti efektyviai panaudota moksliniams ar kasdieniniams

skaičiavimams. Lengvai suvokiami grafikų apsirašymai bei jų valdymas. Brėžinius labai nesunku sukurti ir modifikuoti.

3. MATHCAD paketo animaciją lengva valdyti: galima rodyti po vieną kadrą atskirai, kontroliuoti vaizdavimo greitį, bet kada sustabdyti. Animacinius brėžinius yra galimybė išsaugoti kaip AVI tipo failą. Vėliau šiuos failus įmanoma įkelti į MATHCAD darbo lapą ar kitas aplinkas.

4. MATHCAD animavimo galimybė leidžia stebėti grafiko pokytį realiu laiku, kai keičiasi funkcijos kintamojo reikšmės.

4. ANIMACIJOS KŪRIMAS MATEMATINĖJE SISTEMOJE MAPLE

4.1. MATEMATINĖS SISTEMOS MAPLE APŽVALGA

MAPLE – daugiaplatformis Waterloo MAPLE Software programinis produktas matematiniams skaičiavimams. Pirmoji versija išleista Ontarijuje Kanadoje 1981 m. Nuo 1988 m. programą komerciniais tikslais kūrė Waterloo MAPLE Inc. (dar žinoma MAPLEsoft pavadinimu) [*1,* 5]. Paskutinė versija yra MAPLE 14.

MAPLE turi labai gerą (vieną geriausių) sistemą įvairioms paprastoms bei diferencialinėms lygtims, jų sistemoms analitiškai spręsti, bei atlikti veiksmus su matricomis. MAPLE taip pat turi galimybes funkcijų grafikams bei kitai grafinei informacijai pateikti. Aritmetiniai apskaičiavimai gali būti vykdomi bet kokiu norimu tikslumu (gali būti šimtai ar tūkstančiai ženklų po kablelio). Tačiau tiek didelio, tiek ir mažesnio tikslumo aritmetiniai skaičiavimai MAPLE vykdomi sąlyginai lėčiau ir tai nėra stiprioji šio paketo pusė.

Skirtingai nuo MATHCAD, kuris riboja sukurtų algoritmų perkėlimo galimybes, MAPLE galima užrašyti bei perskaityti lygtis ir „gryno teksto“ forma, kuri panaši į FORTRAN [5, 7].

Iš gautų sprendinių prireikus generuojamas C ar FORTRAN kodas, kurį lengva pritaikyti ir C++ bei Java programose [*1,* 5]. Skirtingai nuo MATLAB ir MATHCAD, kurie patys siūlo skaitmeninio sprendimo galimybes, MAPLE prireikus yra geras įrankis kuriant tradicine programavimo kalba parašytą skaitmeninio užduoties sprendimo programą. Tokie sugeneruoti fragmentai naudingi ir tada, jei kuriama programa bus vykdoma vienu metu daugeliu procesorių superkompiuteryje, arba jei sprendimo algoritmas, jo duomenų struktūros yra tokios, jog ir naudojant „palengvinto programavimo“ paketus programa neužrašoma

pastebimai paprasčiau. Šis metodas efektyvus dirbant su matricomis, nes šiuo atveju glaustai užrašomas matricas turintis reiškinys „išskleidžiamas“ į matricų bibliotekos nenaudojantį kodą bei neretai taip pat randami analitiniai matricų inversijos ir kiti sprendiniai. MAPLE sugeneruotas kodas naudojamas ir taikomosiose, galutiniam vartotojui skirtose programose.

4.2. SISTEMOS MAPLE GALIMYBĖS

MAPLE 7 versijoje kaip priedas, o jau MAPLE 8 versijoje įdiegta nauja technologija *MAPLEts*, kuri leidžia bet kuriai sistemos MAPLE programai kurti patogią vartotojo grafinę sąsają [*1*]. Technologija *MAPLEts* leidžia naudoti mygtukus, įrankių juostas, meniu ir kitus elementus. Sistemos MAPLE darbiniame lange nereikia įvesti procedūrų vardų ir parametrų reikšmių, sąsajoje sukurti duomenų įvesties laukeliai ir mygtukai iškviečia reikiamas procedūras įvestoms reikšmėms. *MAPLEts* paketas turi viršutinę funkciją MAPLE, procedūrą *Display* ir tris vidinius paketus: *Elements*, *Tools* ir *Examples*. Procedūra *Display* atvaizduoja ekrane vartotojo nurodytą grafinę sąsają. Pakete *Elements* saugomi šablonai komponentų grafinei vartotojo sąsajai kurti. Jie klasifikuojami į septynias grupes: komandų, dialogo, išdėstymo, meniu, įrankių juostos, lango korpuso ir kitų elementų. Pakete *Examples* saugomi vartotojo grafinės sąsajos pavyzdžiai, kuriuos galima vartoti vienus arba su kitomis sąsajomis. Pakete *Tools* saugomi šablonai, kurie skirti manipuliuoti ir bendrauti su vartotojo sąsajomis ir vartotojo sąsajų aprašais. Tiek sistemos MAPLE programos, tiek *MAPLEt* programos rašomos komandų eilutėje. Sistemos MAPLE kodas vykdomas komandų eilutėje, o *MAPLEt* programa vykdoma komanda *MAPLEts[Display]*. Vartotojo grafinė sąsaja kuriama įtraukiant elementus į sąrašą, kurio aukščiausio lygio elementas yra *MAPLEt()* elementas. Yra keli pasirinkimai vartotojo grafinės sąsajos išdėstymui struktūrinti. Galima sukurti vartotojo sąsają be formatavimo. Tam MAPLE sąraše apibrėžiami elementai laužtiniuose skliaustuose.

4.3 SISTEMOS MAPLE ANIMACIJOS TYRIMAS PLOKŠTUMOJE (2D)

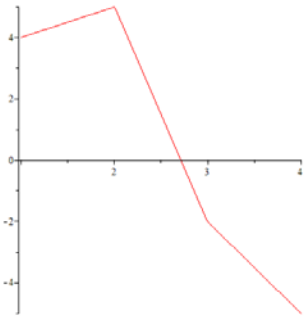
Pagrindinės priemonės grafikams plokštumoje (2D) kurti yra [10]:

- *plot();*
- *plots();*
- *display();*
- *implicitplot();*
- *inequal();*

- *odeplot();*
- *phaseportrait().*

Priemonė *plot()* skirta dvimačių grafikų, laužtinės linijos, taškų, parametrinės kreivės braižymams ir kelių kreivių viename grafike atvaizdavimui. Funkcijos panaudojimo metodai (sintaksės):

- *plot(x^2/(x-1),x=-5..5,y=-5..10)* - dvimačių grafikų braižymas;
- *plot([[1,4],[2,5],[3,-2],[4,-5]])* - laužtinės linijos braižymas;
- *plot([[1,4],[2,5],[3,2]],style=point)* – taškų braižymas;
- *plot([sin(t),cos(3*t),t=-Pi..Pi])* - parametrinės kreivės braižymas;
- *plot([x^2+2*x,exp(x)],x=1..2)* - kelių kreivių viename grafike atvaizdavimas.



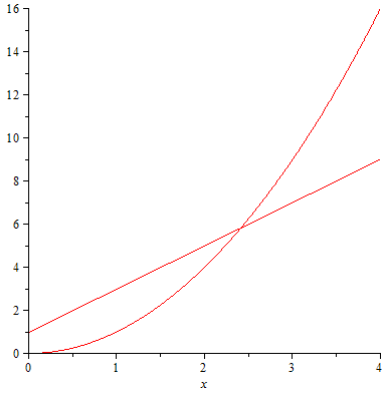
33 pav. MAPLE funkcijos plot () pavyzdys

Priemonė *plots()* skirta grafikos paketo iškvietimui. Funkcijos panaudojimo metodai (sintaksės):

- *with(plots)* - grafikos paketo iškvietimas.

Priemonė *display()* skirta kelių brėžinių parodymui viename. Funkcijos panaudojimo metodai (sintaksės):

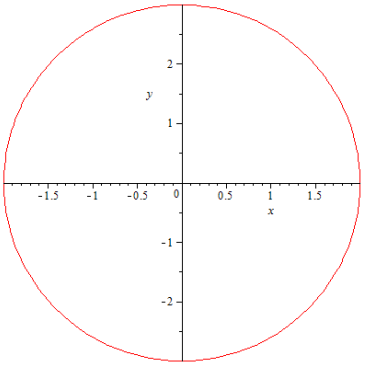
- *br1:=plot(x^2,x=0..4): br2:=plot([[0,1],[2,5],[4,9]],style=point): display(br1,br2);* - kelių brėžinių parodymas viename.



34 pav. MAPLE funkcijos display () pavyzdys

Priemonė *implicitplot()* skirta neišreikštinės funkcijos grafiko brėžimui. Funkcijos panaudojimo metodai (sintaksės):

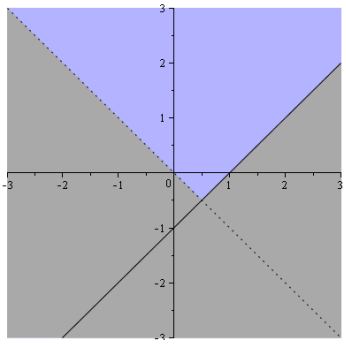
- *plots[implicitplot]((9*x^2+4*y^2)=36,x=-3..3, y= -4..4);* - neišreikštinės funkcijos grafiko brėžimas.



35 pav. MAPLE funkcijos implicitplot pavyzdys

Priemonė *inequal()* skirta tiesinių nelygybių sistemos sprendinių aibės pavaizdavimui. Funkcijos panaudojimo metodai (sintaksės):

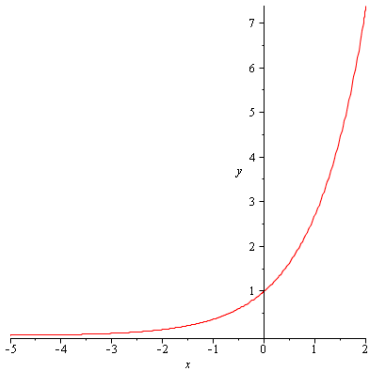
- *plots[inequal]({x+y>0,x-y<=1},x=-3..3,y= -3..3)* - neišreikštinės funkcijos grafiko brėžimas.



36 pav. MAPLE funkcijos inequal pavyzdys

Priemonė *odeplot()* skirta skaitmeniškai sprendžiamų diferencialinių lygčių sprendinių brėžimui. Funkcijos panaudojimo metodai (sintaksės):

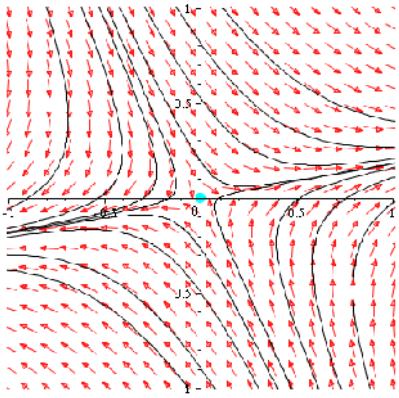
- *plots[odeplot](spr,[[t,y(t)],[t,x(t)]],0..30);* - skaitmeniškai sprendžiamų diferencialinių lygčių sprendinių brėžimas.



37 pav. MAPLE funkcijos odeplot pavyzdys

Priemonė *phaseportrait()* skirta diferencialinių lygčių sistemų sprendinių ir fazinio portreto brėžimui. Funkcijos panaudojimo metodai (sintaksės):

- *DEtools[phaseportrait]([L],[x(t),y(t)],t=0..30,[[ps]]);* - diferencialinių lygčių sistemų sprendinių ir fazinio portreto brėžimas.



38 pav. MAPLE funkcijos phaseportrait pavyzdys

Nereiktų pamiršti, kad visas aprašytas grafikų braižymo priemones galima animuoti. Animaciniams grafikams dvimatėje erdvėje kurti yra pateikta pora sintaksių:

- *animate(F, x, t);* - *F* – braižoma funkcija, *x* – horizontalių koordinačių nustatymas, *t* – kadro parametrų intervalas;
- *animate(F, x, t,...);* - *F* – braižoma funkcija, *x* – horizontalių koordinačių nustatymas, *t* – kadro parametrų intervalas.

  Priemonė *animate* dažniausia aprašoma šitaip:

animate(F(x,t),x=m..n,y=j..i),

čia *F* yra funkcija su kintamaisiais *x* ir *y*, o *m..n* nusako realiųjų skaičių sritį, kurioje ir yra funkcija *F*. Intervalas *j..i* nusako, kaip kinta eiliškų kadrų koordinatės. Be abejo galima apibrėžti ir vertikalių koordinačių sritį naudojant vaizdo parametrus - *view*.

Kadrai aprašomi parametru *frames*, kur pagal nutylėjimą numatytasis *frames* yra 16. Animacijos efektą sukuria kintamojo *y* kitimas. Kai yra įvykdoma komanda, tai pamatomas tik funkcijos grafikas, bet jeigu jį pažymėsime, atsiras animacijos įrankių juosta. Animacijai veikimui reikia suaktyvinti su šiomis funkcijomis nubrėžtus grafikus ir paspausti atitinkamus animacijos vykdymo mygtukus. Jų sąrašas yra toks [7]:

3 lentelė. MAPLE animacijos vykdymo mygtukai

	buvusio kadro rodymas
	animacijos sustabdymas
	animacijos vykdymas

	kito iš eilės einančio kadro rodymas
	animacijos režimas pirmyn ir atgal
	animacijos režimas atgal
	animacijos režimas į priekį
	animacija cikliškai atliekama be perstojo (iki sustabdymo)
	animacija atliekama vieną kartą

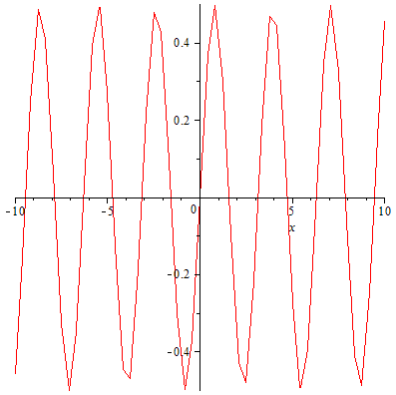
Dabar panagrinėkime detaliau. Tarkime turime funkciją:

$$f(x,y)=\frac{\sin(x\times y)}{\cos(x\times y)};$$

Ją aprašome MAPLE matematinėje sistemoje apibrėždami sritis. Funkcija apibrėžta MAPLE priemonėje *plots*, todėl prieš funkcijos panaudojimą privaloma iškviesti šią sistemos priemonę:

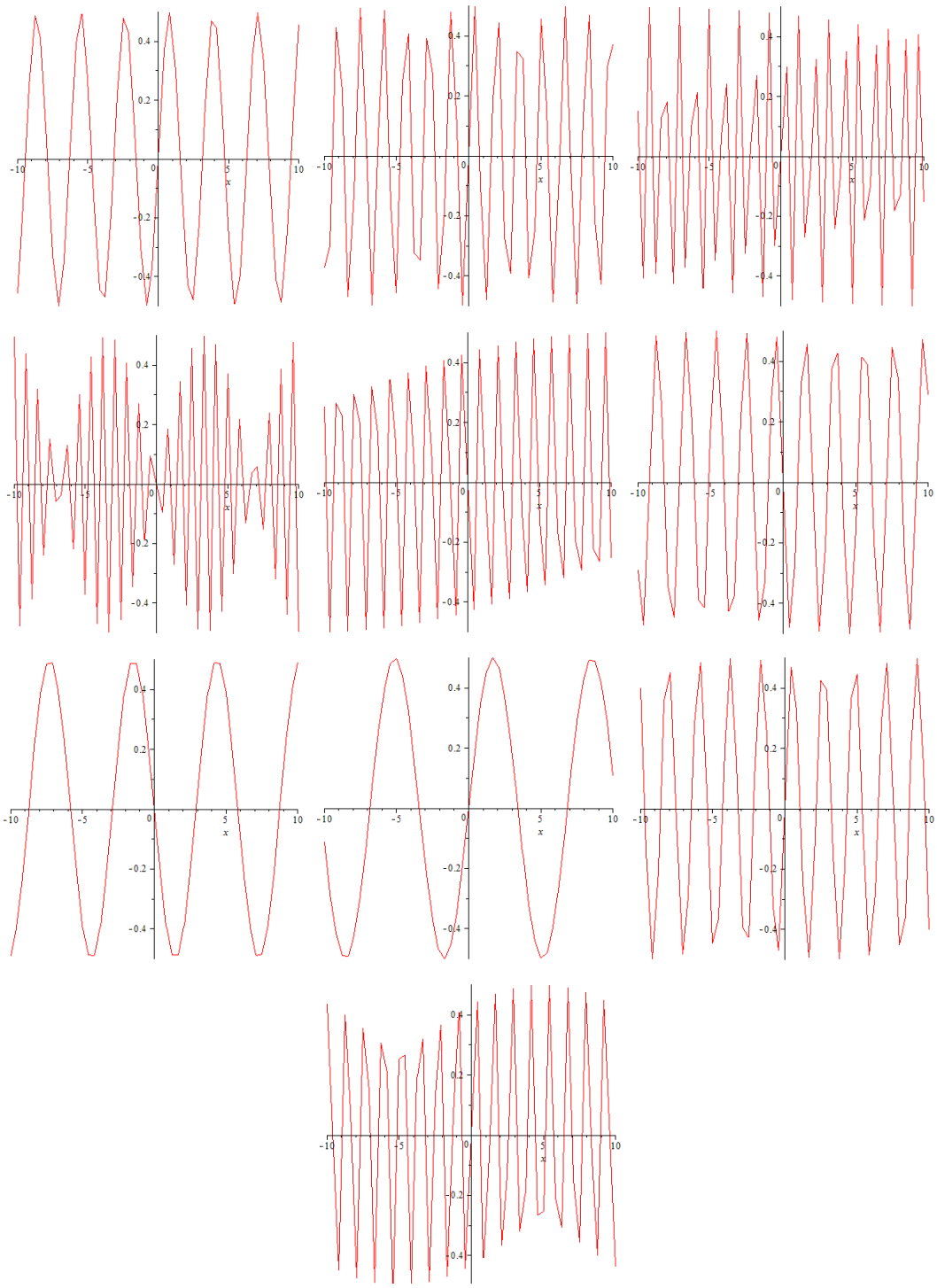
```
with(plots):
animate(sin(x*y)*cos(x*y),x=-10..10,y=1..10,frames=10,color=red);
```

Užrašius programos tekstą ir paleidus matematinę sistemą generuoti animaciją pirmiausia išvystame pirmą animacijos kadrą pavaizduotą 39 pav.



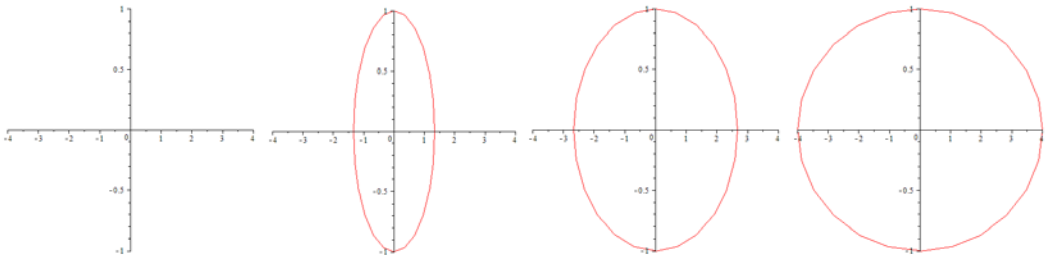
39 pav. MAPLE animacijos (animate()) pirmas kadras

Be abejo ši priemonė nupiešia kadrų aibę. Suaktyvinus grafiką ir paspaudus animacijos paleidimo mygtuką  , pamatome funkcijos keitimąsi nuo pirmojo kadro iki paskutinio. Išsamiau ir lėtai peržiūrėti visus animacijos kadrus (40 pav.) galima naudojantis animacijos mygtuku  .

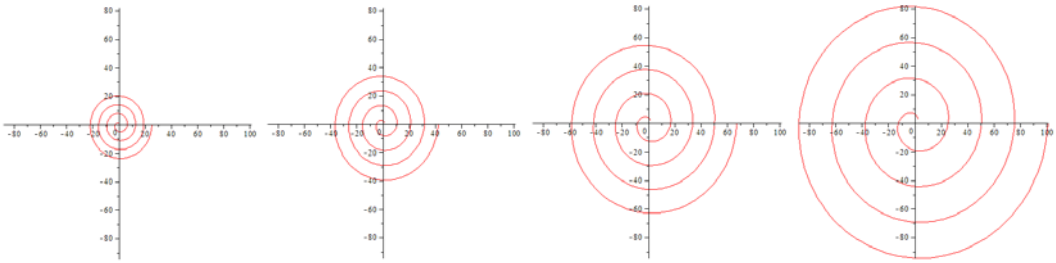


40 pav. MAPLE animacijos (animate())kadrų aibė

Animacijos priemonę taip pat įmanoma taikyti parametrinėms kreivėms ir funkcijoms (41 pav.), kurios yra brėžiamos ne Dekarto, o polinėje koordinačių sistemoje (42 pav.) [7].



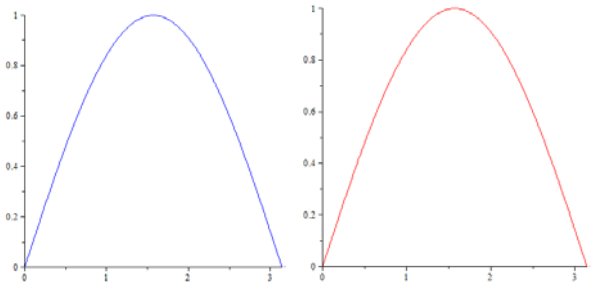
41 pav. MAPLE animacijos panaudojimas parametrinei kreivei



42 pav. MAPLE animacijos panaudojimas polinėje koordinačių sistemoje

Taip pat nereiktų pamiršti, kad matematinė sistema MAPLE suteikia galimybę animuoti ir brėžinių spalvas (43 pav.) ir brėžiniuose esančius pagalbinius tekstus (44 pav.). Spalvų keitimui pavaizduoti naudojant animaciją pasitelkėme Aladjevo funkcijos grafiką [10]:

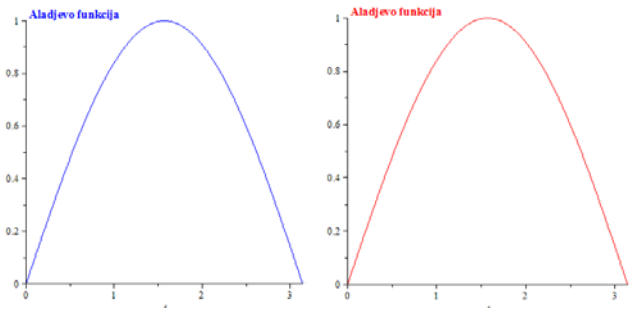
```
T1:=plot(sin(t),t=0..Pi,colour=red):
T2:=plot(sin(t),t=0..Pi,colour=blue):
display({T1,T2},insequence=true);
```



43 pav. MAPLE animacijos panaudojimas grafiko spalvų keitime

Tame pačiame grafike uždėkime kelis skirtingus užrašus, kuriuos aprašome taip:

```
T1 := plot(sin(t), t = 0 .. Pi, colour = red):
T2 := plot(sin(t), t = 0 .. Pi, colour = blue):
T3 := textplot([0, 1, "Aladjevo funkcija"], 'align = {above, right}',
colour = red, font = [TIMES, BOLD, 12]):
T4 := textplot([0, 1, "Aladjevo funkcija"], 'align = {above, right}',
colour = blue, font = [TIMES, BOLD, 12]):
display({display(T1, T3), display(T2, T4)}, insequence = true);
```



44 pav. MAPLE animacijos panaudojimas grafiko spalvų keitime ir teksto rašyme

Kaip matyti iš grafiko užrašymo, kad grafiko spalva neįtakoja teksto spalvos ir atvirkščiai.

4.4 SISTEMOS MAPLE ANIMACIJOS TYRIMAS ERDVĖJE (3D)

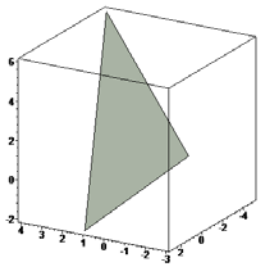
Pagrindinės priemonės grafikams erdvėje (3D) kurti yra [10]:

- *display();*
- *implicitplot();*
- *inequal();*
- *odeplot();*
- *phaseportrait();*
- *plot3d().*

Priemonės *display(), implicitplot(), inequal(), odeplot()* ir *phaseportrait()* buvo aprašytos kalbant apie sistemos MAPLE grafikų piešimą plokštumoje (2D). Šių priemonių panaudojimas ir funkcijos nesikeičia.

Priemonė *plot3d()* skirta trimačių grafikų ir kelių paviršių viename grafike atvaizdavimui. Funkcijos panaudojimo metodai (sintaksės):

- *plot3d(sin(x+y),x=-1..1,y=-1..1)* - trimačių grafikų braižymas;
- *plot3d({x*sin(y^2),1 -y*cos(x^2)}, x= -1..1, y= -1..1)* - kelių paviršių viename grafike atvaizdavimas.



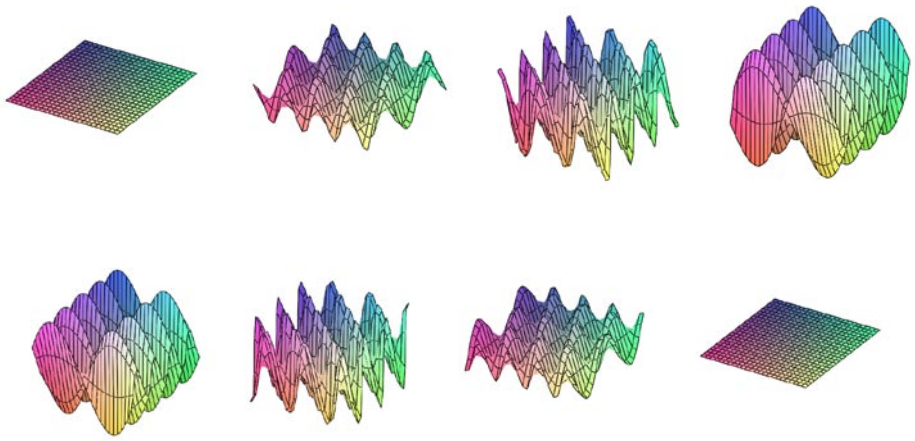
45 pav. MAPLE plot3d() priemonės panaudojimas

Kaip ir plokštumoje (2D), taip ir erdvėje (3D) MAPLE sistema gali pasiūlyti animaciją grafikams vaizduoti. Priemonė animaciniams trimačiams grafikams piešti yra *animate3d(F, x, y, t)*, kur *F* – piešiama funkcija, *x* – horizontalių koordinačių nustatymas, *y* – vertikalių koordinačių nustatymas, *t* – kadro parametrų intervalas. Dažniausiai priemonė *animate3d* rašoma *animate3d(F(x, y, t), x=m..n, y=i..j, t=p..q*, kur *F* yra funkcija su kintamaisiais *x* ir *y* bei *t***.** Intervalai *m..n* ir *i..j* nusako realiųjų skaičių sritį, kurioje atvaizduojama funkcija *F*. Intervalas *p..q* nusako, kaip animacijos veikimo metu keičiasi

kadrų koordinatės. Likusieji argumentai tiesiog nusako lygybe *parametras = reikšmė*. Pavyzdžiui, taip nurodomas kadrų kiekis, kur numatytoji reikšmė yra *frames=8*. Tarkime turime tokį *animate3d* aprašymą MAPLE sistemoje:

```
with(plots)
A:=-4*Pi..4*Pi:
animate3d(sin(p)*sin(y*cos(p))*cos(x*sin(p)),x=A,y=A,p=A,frames=8);
```

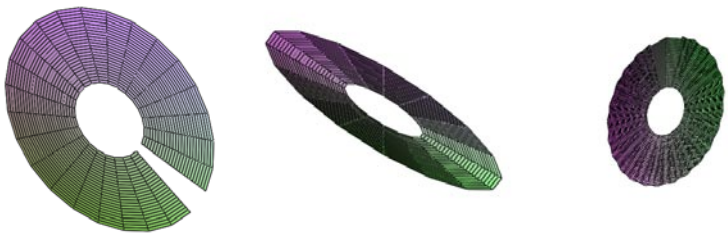
Žemiau pateikiami gauti rezultatai (46 pav.):



46 pav. MAPLE animate3d() priemonės panaudojimo pavyzdys

Suaktyvinus brėžinį taip pat kaip ir plokštumoje atsiranda animacijos įrankių juosta, kurios priemonės yra tokios pat ir veikia taip pat, kaip ir animacijos plokštumoje. Skiriasi tik tuo, kad, piešiant animaciją trimatėje erdvėje, pažymėjus grafiką ir laikant pelę ant jo pasikeičia pelės kursorius į lenktą rodyklę, kuria galima reguliuoti horizontalios ir vertikalios plokštumų posūkio kampus. Erdvinės animacijos komandą galima taikyti ir parametrinėmis koordinatėmis nurodytam paviršiui [8]:

```
with(plots):
A:=-4*Pi..4*Pi:
animate3d([x*u,u*t,x*cos(t*u)],x=1..3,t=1..4,u=2..4,coords=cylindrical);
```



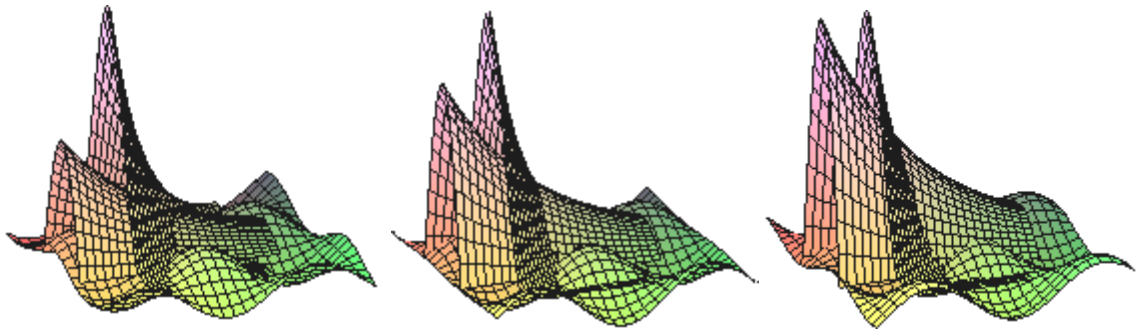
47 pav. MAPLE animate3d() priemonės panaudojimo pavyzdys parametrinei funkcijai

Norėdami vienu metu atvaizduoti kelias funkcijas erdvėje, naudojama priemonė *display3d*, kuri yra rašoma taip:

- *display3d(L)* - kur *L* – norimų atvaizduoti funkcijų sąrašas ar masyvas, numatytasis *insequence=false*, todėl visos funkcijos vaizduojamos vienu metu.

Pavyzdžiui [*6*]:

```
a := animate3d(cos(t+x)*sin(t+y),x=-Pi..2*Pi, y=-Pi..2*Pi,t=1..2,frames=3):
b := animate3d((1.3)^x*sin(y)*t,x=-1..2*Pi,y=0..Pi,t=1..2,frames=3):
c := plot3d(binomial,0..5,0..5):
     display([a,b,c]);
```



48 pav. Funkcijos display3d() sukurti kadrai

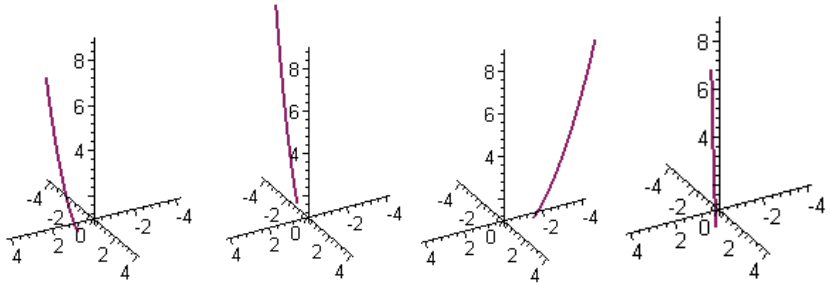
Taip matematinė sistema MAPLE leidžia kreivę atvaizduoti erdvėje naudojant priemonę *spacecurve*. Funkcijos panaudojimo metodas (sintaksė):

- *spacecurve(L, options)- L* – braižomų kreivių rinkinys.

Priemonė *spacecurve* trimatėje erdvėje nupiešia vieną ar kelias kreives. Pirmas argumentas yra taškų aibė arba kreivė, turinti tris ar daugiau parametrų. Pirmosios komponentės laikomos *x, y* ir *z* koordinačių vaizdavimu. Papildomi argumentai - kreivės braižymo sritį apibūdinantis intervalas *t=m..n* bei kreivės brėžimui naudojamų taškų skaičius *numpoints=n,* kurios pagal nutylėjimą numatytoji reikšmė yra 50. Pirmas argumentas taip pat gali būti erdvėje piešiamų kreivių sąrašas [*1*].

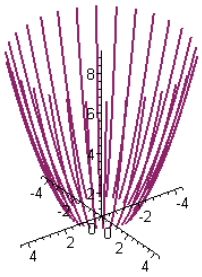
Panaudojant priemonę *display*, tokiu būdu sukonstruotas kreives galima paeiliui atvaizduoti ekrane, taip sukuriant animaciją.

```
p:=spacecurve([t,t,t^2], t=1..3,color=maroon,thickness=2):
b:=array(0..30):
for i from 0 to 30 do b[i]:=rotate(p,0,0,i*Pi/15) od:
display( seq(b[i], i=1..30),insequence=true);
```



49 pav. Funkcijų spacecurve ir display sukurti kadrai

Jei prieš tai buvusiame pavyzdyje parametro *insequence* reikšmė būtų *false*, visos kreivės, sukurtos priemone *spacecurve*, būtų paišomos iš karto po komandos vykdymo ir grafikas neturėtų animacijos savybių [*6*].



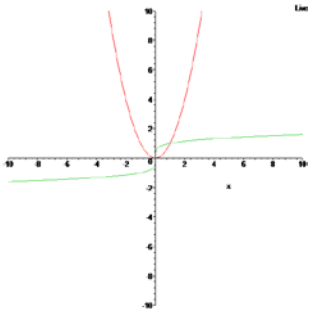
50 pav. Funkcijų spacecurve ir display sukurti grafikas, kai prarametras insequence=false

Matematinis paketas MAPLE suteikia galimybę įrašyti brėžinius, kuriems atvaizduoti buvo taikomos animacijos kūrimo priemonės, grafiniuose GIF tipo failuose. Šiuos failus paskui galima panaudoti animacijos vaizdavimui interneto puslapiuose. MAPLE sukuria animuotą GIF-failą. Norėdami įrašyti animacinį brėžinį, pirma reikia iškviesti komandą *plotsetup()* ir nurodyti failo, kuriame bus saugoma animacija, vardą:

```
plotsetup(gif,plotoutput=`d:\plot.gif`);
animate(cos(x+phi),x=0..2*Pi,phi=0..2*Pi,color=black,thickness=2);
```

4.5 SISTEMOS MAPLE ANIMACIJOS PLOKŠTUMOJE (2D) IR ERDVĖJE (3D) TIKSLUMAS IR PAKLAIDOS

Su lyg MAPLE 6 išleidimu šioje matematinėje sistemoje pasirodė ir daugiau klaidų grafikoje. Keletą grafikų variantų patyrinėjome. Algebrinę išraišką `smartplot(x^2,y^3);` pavaizdavome grafiškai [*6*]:

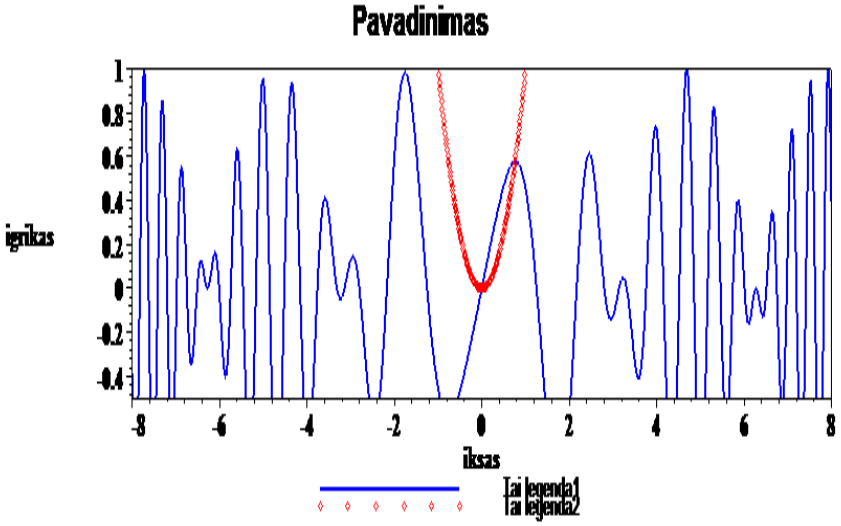


51 pav. Algebrinės išraiškos smartplot(x^2,y^3); grafikas

Kaip matyti iš grafiko ir aprašytos išraiškos MAPLE kalba, kad pagal nutylėjimą numatytosios koordinačių ribos yra [-10; 10]. Dėl neaprašyto grafiko valdymo prarandame dalį svarbios informacijos. Grafikas yra netinkamai išnaudojamas atvaizdavimo lange:

nematome tikslių susikirtimo taškų, parabolių kitimo greičio. Tačiau tai nėra didelė MAPLE bėda. Ji sprendžiama labai paprastai: aprašius papildomus parametrus grafiko braižymui.

Kita matematinės sistemos MAPLE piešiamų grafikų klaida – tai teksto atvaizdavimas. Ypatingai matosi senesnėse MAPLE versijose (MAPLE 5, MAPLE 6, MAPLE 7). Didžiausią žalą daro teksto paryškinimas (bold) - pradeda simboliai lipti vienas ant kito taip uždengdami vienas kitą ar tiesiog uždengdami patys save (52 pav.) [6].



52 pav. MAPLE problema – teksto pateikimas

Tačiau didesni nuostoliai yra tada, kai tokių pačių bėdų turi ir esminės grafiko detalės – linijos, taškai, kreivės, plokštumos ir kt.. Tokie iškreipimai iškreipia ir vaizdavimo rezultatus. Todėl svarbu pasirinkti tinkamą taško, linijos ar kito objekto piešimo storį. Be abejo svarbu ir be reikalo neapkrauti brėžinio nereikalingais objektais.

Grafikos kokybė priklauso ir nuo monitoriaus specifikų – rezoliucijos ir raiškos. Mažesnė raiška – prastesnė grafiko kokybė, didesni brėžinio iškraipymai.

4.6. SISTEMOS MAPLE ANIMACIJOS PLOKŠTUMOJE (2D) IR EDVĖJE (3D) IŠVADOS

1. Kompiuterinė matematinė sistema MAPLE suteikia galimybę piešti paprastus ir sudėtingus grafikus dvimatėje (2D) ir trimatėje (3D) erdvėse, redaguoti vaizdavimo kampus horizontalios ir vertikalios plokštumos atžvilgiu, lengvai ir greitai kurti ir valdyti animaciją.
2. MAPLE sistemoje yra speciali įrankių juosta leidžianti valdyti animaciją. Juostoje esančios priemonės suteikia galimybę animaciją stebėti nuo pirmo iki paskutinio kadro

ir atvirkščiai. Taip pat yra galimybė bet kuriuo metu animaciją nutraukti ir parodyti po vieną kadrą.

3. Yra galimybė MAPLE sistemoje animaciją pritaikyti polinėms kreivėms bei funkcijoms, išreikštoms parametrinėmis lygtimis, vienu metu vaizduoti kelių funkcijų kitimą.
4. Animaciją leidžiama įrašyti, o paskui ir peržiūrėti grafiniuose .gif tipo failuose. Failus galima įterpti į interneto puslapius kaip pavyzdinius grafikus parodant jų kitimo eigą.
5. Matematinės sistemos MAPLE brėžiniuose svarbu pasirinkti tinkamą piešiamų objektų storumą (bold) ir tinkamas koordinačių sritis, kitaip matysime netikslų vaizdą.

5. ANIMACIJOS KŪRIMAS MATEMATINĖJE SISTEMOJE MATLAB

5.1. MATEMATINĖS SISTEMOS MATLAB APŽVALGA

MATLAB (iš žodžių MATrix LABoratory) yra daugiaplatformė MathWorks programinė įranga, skirta įvairių mokslo šakų problemoms spręsti, ypač matematinėms. Kaip galima spręsti iš pavadinimo, turi puikias galimybes manipuliacijoms su matricomis – būtent toks buvo pirminis šios programos tikslas. Dabar tai didžiulis galingas paketas, turintis savitą lengvai perprantamą programavimo kalbą [*3,* 8].

Nors simbolinėms matematinėms manipuliacijoms geriau tinka MAPLE ir Mathematica produktai, su MATHCAD pradedantis vartotojas gali greičiau gauti pirmuosius rezultatus, MATLAB kaip programavimo kalba turi didesnį lankstumą, kuris naudingas įvairiems fizikos, biologijos bei kitiems matematiniams modeliams kurti ir tirti. Svarbus MATLAB trūkumas (palyginus su tradicinėmis programavimo kalbomis) yra reikalavimas visiems naudojantiems turėti įkeltas mokamas MATLAB bei visų reikalingų jo papildomų modulių kopijas [*8,* 5]. Tradicinės programavimo kalbos šiuo metu turi atviro kodo ar nemokamus kompiliatorius bei pasiekia didesnį vykdymo greitį, tačiau programavimas jomis ilgiau užtrunka. Bioinformatikiniuose tyrimuos MATLAB neretai naudojamas kaip „žvalgybinis“ paketas: pasiteisinę matematiniai modeliai vėliau perrašomi C (neretai generuojant žymias kodo dalis su MAPLE) lygiagrečiam vykdymui superkompiuteriuose [*3,* 1].

5.2. SISTEMOS MATLAB GALIMYBĖS

MATLAB 5 versijoje grafinė vartotojo sąsaja buvo kuriama *m*-rinkmenoje iš specialių komandų ir funkcijų (*uitools*) [2]. MATLAB R2010b versijoje jau palaikomas vaizdinis programavimas (GUIDE). Atidaromas specialus langas, kuriame yra priemonės programoms kurti, testuoti ir derinti. Vartotojo grafinė sąsaja kuriama komponentus iš komponentų paletės pele įkeliant į darbalaukį, kuris įforminamas kaip FIG-rinkmena (panašiai dirbama *C++Builder* sistemoje) [5]. Dešiniojo pelytės klavišo paspaudimu ant komponento iškviečiamas savybių langas, kuriame nustatomos reikalingos komponento savybės: *enable*, *extent*, *fontsize* ir kt. Į programą galima įtraukti daug tiek to paties, tiek skirtingų tipų komponentų. Su kiekvienu objektu galima atlikti tam tikrus veiksmus (metodus). Jie aprašomi programos kodo lange (*m*-rinkmenoje), kuriame jau yra tam tikros struktūros metodo aprašo šablonas. Kompiuterinės matematinės sistemos vartotojo grafinės sąsajos kūrimo elementai sukurti *Java* kalbos pagrindu ir gana paprastai naudojami. Vartotojui *Java* kalbos žinių nereikia, užtenka pagrindinių MATLAB programavimo kalbų žinių. Sistemų MAPLE 9 ir MATHEMATICA 5.0 savarankiška vartotojo sąsajos kūrimo programa – tai komandų aprašas, kuris prasideda nuo labiausiai nutolusio grafinės vartotojo sąsajos valdymo elemento. Kiekvienas kompiuterinės matematinės sistemos elementas apibūdinamas savybėmis, metodais, įvykiais. Pavyzdžiui, elementas *Label* sistemoje MAPLE turi 13 savybių (*background*, *caption*, *enabled*, *visible* ir kt.), sistemoje *C++Builder* – 36 (*caption*, *enabled*, *visible*, *width* ir kt.). Kompiuterinės matematinės sistemos leidžia tiesiogiai kurti vartotojo grafinę sąsają jų pagrindiniuose languose. Taip palengvinamas interaktyvių grafikų pristatymo ar atvaizdavimo generavimas [5].

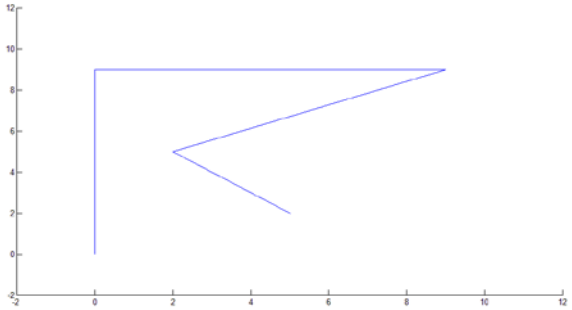
5.3. SISTEMOS MATLAB ANIMACIJOS TYRIMAS PLOKŠTUMOJE (2D)

Vienos populiaresnių priemonių animaciniams grafikams ir kitiems objektams plokštumoje kurti [1]:

- line();
- plot();
- ezplot();
- fill();

Priemonė *LINE ()* skirta linijų braižymui. Funkcijos panaudojimo metodai (sintaksės):

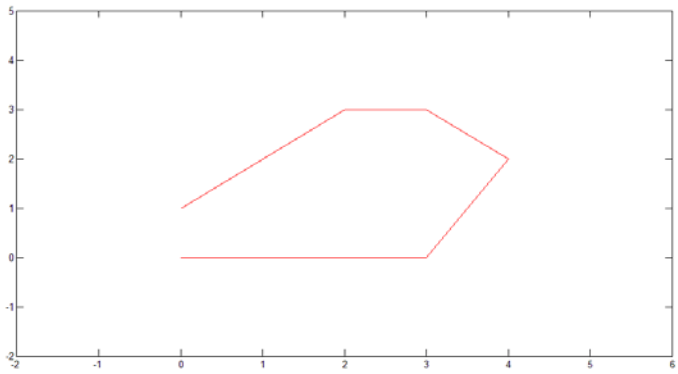
- *line* – pagal nutylėjimą braižo liniją, kurios pradinių taškų koordinatės yra [0 0] ir [1 1].
- *line(X,Y)* – braižo liniją, kurios taškai yra aprašyti masyvuose X ir Y. Pirmoji (X) reikšmė nusako horizontalios (abscisių) ašies reikšmes, o antroji (Y) - vertikalios (ordinačių) ašies reikšmes.
- *line(X,Y,Z)* - braižo trimatę liniją.
- *line(X,Y,Z,'PropertyName',propertyvalue,...)* - sukuria linijas, nustatytomis naudojamų savybėmis ir jų reikšmėmis. *PropertyName* – naudojama savybė, *propertyvalue* – savybės reikšmė.
- *line('XData',x,'YData',y,'ZData',z,...)* - sukuria linijas, kai konkrečiai koordinačių ašiai priskiriamos reikšmės. *Xdata, Ydata, Zdata* - koordinačių ašys, *x, y, z,* - vienmačiai masyvai su grafikų reikšmėmis.



53 pav. MATLAB funkcijos line() pavyzdys

Priemonė *PLOT()* – išveda grafiką. Funkcijos panaudojimo metodai (sintaksės):

- *plot(Y)* - grafike ant x ašies atidedami taškų eilės numeriai;
- *plot(X1,Y1,...,Xn,Yn)* - išveda kelių funkcijų grafikus viename lange. Dydžiai Xn, Yn –n - osios funkcijos argumentų bei funkcijų reikšmių masyvai;
- *plot(X1,Y1,LineSpec,...,Xn,Yn,LineSpec)* - išveda funkcijos grafiką lange. Dydžiai X1, Y1 funkcijos argumentų bei funkcijų reikšmių masyvai, *LineSpec* – simbolinis kintamasis (nebūtinas), kuriame gali būti iki 3 specialių simbolių, nurodančių kreivės charakteristikas: linijos tipą, mazgo (taško) tipą bei linijos spalvą;
- *plot(X1,Y1,LineSpec,'PropertyName',PropertyValue)* - išveda funkcijos grafiką lange. Dydžiai X1, Y1 funkcijos argumentų bei funkcijų reikšmių masyvai, *LineSpec* – simbolinis kintamasis (nebūtinas), kuriame gali būti iki 3 specialių simbolių, nurodančių kreivės charakteristikas: linijos tipą, mazgo (taško) tipą bei linijos spalvą. *PropertyName* – naudojama savybė, *propertyvalue* – savybės reikšmė;

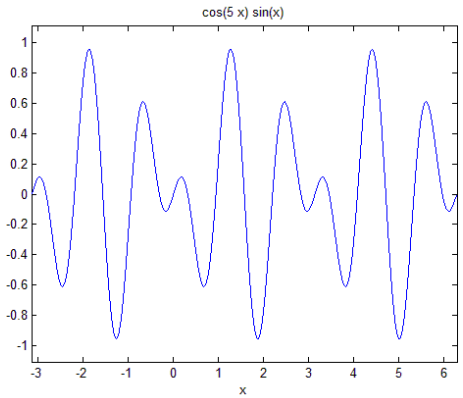


54 pav. MATLAB funkcijos plot() pavyzdys

Priemonės LINE() ir PLOT() yra panašaus principo. Tiek naudojant LINE(), tiek PLOT() galima nubrėžti liniją ar tiesę. Taip pat įmanoma su LINE() nubrėžti grafiką.

Priemonė EZPLOT *()* skirta supaprastintam grafiko braižymui. Funkcijos panaudojimo metodai (sintaksės):

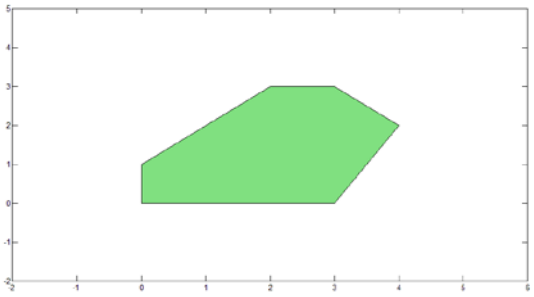
- *ezplot(fun)* - braižo išraišką *fun(x),* kur pagal nutylėjimą *x* yra $-2\pi < x < 2\pi$. *fun* yra viena iš numatytų MATLAB funkcijų, pvz.: *sin, cos* ir t.t.
- *ezplot(fun,[min,max])* - braižo išraišką *fun(x),* kur *x* yra $min < x < max$. *fun* yra viena iš numatytų MATLAB funkcijų, pvz.: *sin, cos* ir t.t.
- *ezplot(fun2)* - braižo išraišką *fun2(x,y)=0,* kur pagal nutylėjimą *x* yra $-2\pi < x < 2\pi$ *ir y* yra $-2\pi < y < 2\pi$.
- *ezplot(fun2,[xmin,xmax,ymin,ymax]))* - braižo išraišką *fun2(x,y)=0,* kur *x* yra x$min < x <$ *xmax ir y* yra y$min < y < ymax$.
- *ezplot(fun2,[min,max])* - braižo išraišką *fun2(x,y)=0,* kur *x* yra $min < x < max$ *ir y* yra $min < y < max$.
- *ezplot(funx,funy)* - braižo išraiškas *funx(t)* ir *funy(t),* kur pagal nutylėjimą *t* yra $0 < t < 2\pi$.
- *ezplot(funx,funy,[tmin,tmax])* - braižo išraiškas *funx(t)* ir *funy(t),* kur *t* yra $tmin < t <$ *tmax.*



55 pav. MATLAB funkcijos ezplot() pavyzdys

Priemonė *FILL ()* skirta spalvinti dvimačiams daugiakampiams. Funkcijos panaudojimo metodai (sintaksės):

- *fill(X,Y,C)* – čia *X* ir *Y* daugiakampio viršūnių koordinačių masyvai, *C* daugiakampio spalva, kuri yra aprašoma spalvų kodais.
- *fill(X,Y,'ColorSpec')* - čia *X* ir *Y* daugiakampio viršūnių koordinačių masyvai, *ColorSpec* daugiakampio spalva, kuri gali būti aprašoma viena raide ar pilnu žodžiu.
- *fill(X1,Y1,C1,X2,Y2,C2,...)* – kelių dvimačių daugiakampių spalvinimas.
- *fill(...,'PropertyName',PropertyValue* - *PropertyName* – naudojama savybė, *propertyvalue* – savybės reikšmė.



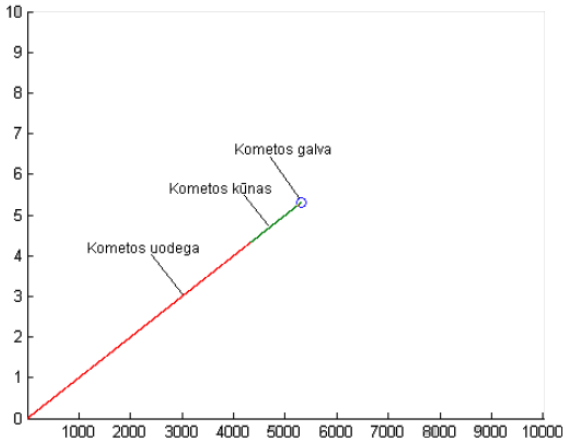
56 pav. MATLAB funkcijos fill() pavyzdys

Visos aukščiau išvardintos MATLAB priemonės yra pritaikytos statiškam atvaizdavimui. Tačiau reikiant atvaizduoti dinamiškus vaizdus naudojamas kitos funkcijos arba modifikuojamos esamos. Viena iš dinamiškų priemonių yra *comet()*. Funkcija *comet()* atvaizduoja slenkantį tašką plokštumoje. Šis objekto piešimas plokštumoje asociojuojasi su kometos judėjimu. Funkcijos panaudojimo metodai (sintaksės):

- *comet(y)* – vaizduoja kometos judėjimą, vektoriaus y trajektorija;
- *comet(x,y)* – vaizduoja kometos judėjimą, kai kreivė apibrėžta abiejų ašių atžvilgiu.

Kaip pavyzdį imkime, kad x kinta nuo 0 iki 10, žingsniu 0,001. Kuo mažesnis x kitimo žingsnis, tuo lėčiau ir tiksliau brėžiama funkcija. Numatytasis žingsnis 1. Vadinasi, jei norime, kad kometa judėtų lėtai, turime parinkti mažesnį kitimo žingsnį [2].

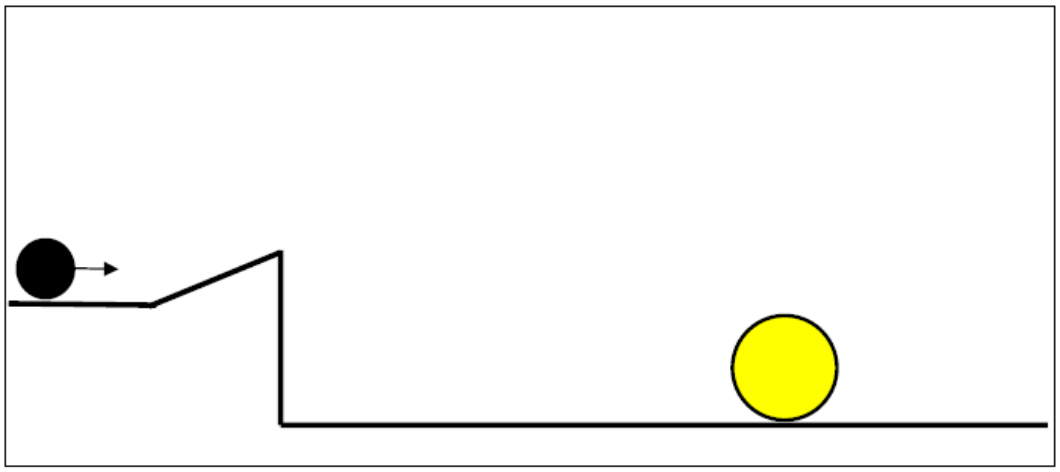
```
x=0:0.001:10;
comet(x)
```



57 pav. MATLAB funkcijos comet(x) pavyzdys

Komandos *comet()* kuriamame brėžinyje mažas apskritimas (kometos galva) vieną po kito vaizduoja visus funkcijos taškus. Kometos kūnas yra paskui apskritimą (kometos galvą) besidriekianti atkarpa, paskui kurią tęsiasi kometos uodega [5]. Kometos uodega ištisine linija vaizduoja funkcijos kreivę (8 pav.).

Kometos veikimo pagrindu MATLAB sistemoje galime atvaizduoti įvairius reiškinius ir jų pasekmes. Galime sumodeliuoti įvykių seką. Tarkime juodas kamuoliukas (9 pav.) su dideliu pradiniu greičiu į pakyla į įkalnę, laisvai krenta žemyn ir atsimušęs į grindis šokinėja (gal atsitrenkia į kitą kamuolį). Modeliuojama, kol sistema nusistovės. Riedant kamuoliams, įvertinama trintis su paviršiumi ir tai, kad smūginis kontaktas su sienele dalinai plastiškas.



58 pav. Kamuoliuko judėjimo pavyzdys

Baigtinių elementų metodu suformuojamas diskretusis tiriamos struktūros modelis standartiniu pavidalu. Jis užrašomas matricinėmis algebrinėmis arba paprastosiomis diferencialinėmis lygtimis. Ne mažiau svarbu parinkti racionalų ir efektyvų tolesnės modelio analizės būdą.

Be abejo, pirmiausia norint modeliuoti juodo kamuoliuko veiksmus reikia atsižvelgti į visus fizikinius dėsnius, kurie gali lemti kamuoliuko ir jį supančios aplinkos tolimesnius veiksmus ir pasekmes. Aprašytų objektų elgseną šioje sistemoje iš esmės lemia trys esminės sąveikos:

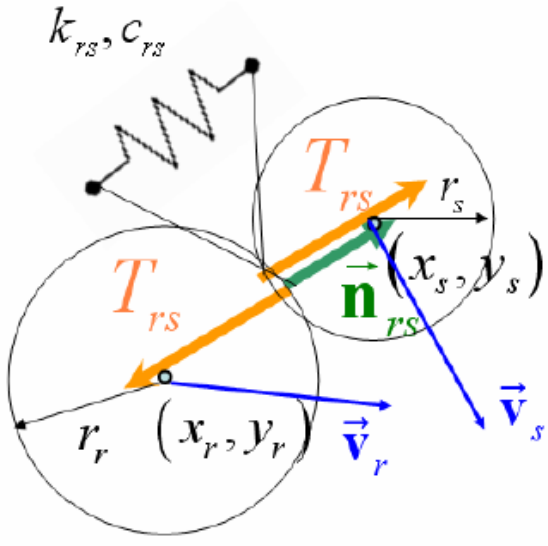
- Žemės traukos sąveika;
- Objekto sąveika su sienomis;
- Objekto sąveika su kampais.

Objektų tarpusavio sąveiką aprašoma *baudos funkcijų metodu.* Ši schema veikia baudos elemento įdėjimu tarp sąveikaujančių objektų (10 pav.). Kai tik objektai priartėja vienas prie kito, tarp jų sukuriamas tamprusis elementas, kurio veikimo kryptis yra išilgai objektų centrų. Šio elemento charakteristikos nebūtinai yra realios, bet dažniausiai parenkamos atsižvelgiant į sąveikos procesą. Kad kūnų sąveika atrodytų realistiška, baudos elemento klampumo bei tamprumo koeficientai parenkami atsižvelgiant į sąveikaujančių objektų masę. Literatūroje [*10*] pateikiamos formulės:

$$c \geq (1.6 \div 2) \times \sqrt{mk} \quad (1)$$

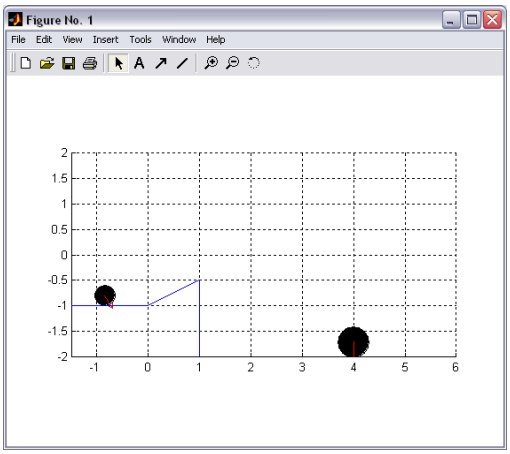
$$c \geq (1.6 \div 2) \times \sqrt{\frac{m_r m_s k_{rs}}{m_r + m_s}} \quad (2)$$

Pirmoji (1) formulė skirta apskaičiuoti orientaciniam klampumo koeficientui tarp objekto ir kraštinių sienų (dugno). Joje įrašoma objekto, sąveikaujančio su kraštine sieną masė. Antroji formulė (2) skirta apskaičiuoti orientaciniam dvejų objektų sąveikos klampumo koeficientui.

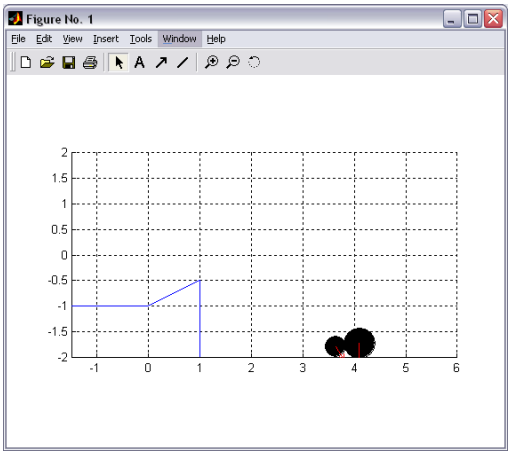


59 pav. Objektų tarpusavio sąveiką

Realizavus modelį MATLAB programiniu paketu žemiau pateiktose iliustracijose (11 pav.) ir (12 pav.) matome programos eigos vykdymo momentus.



60 pav. Programos pradžios iliustracija



61 pav. Programos pabaigos iliustracija

Šis grafinis atvaizdavimo metodas padeda realiai įsivaizduoti numatomo veikimo eigą. Tačiau dėl dažno ir didelio elementų perpaišymo skaičiaus metodas reikalauja didelių kompiuterinių resursų. Sumodeliuotas vaizdų metodas atvaizduojamas kadrų sistema – kadras seka kadrą, o kiekviename kadre perpaišomas naujas vaizdas su naujomis objektų koordinatėmis plokštumoje. Iš principo, labai nedideles iš vieno ar keleto elementų sudarytų konstrukcijų lygtis galima būtų suformuoti ir išspręsti analitiškai bei gauti analitines sprendinių išraiškas. Tačiau tokios moduliacijos yra suvokiamos kaip skaitiniai metodai, t. y. jais apskaičiuojame konkrečiu žinomu būdu veikiamos sistemos atsako skaitines reikšmes, o ne to atsako išraiškos analitinį pavidalą.

Baigtinių elementų metodu taip pat formuojamos ir moduliacijos, kurios paremtos spalvinių diagramų metodu. Tarkime, reikia moduliuoti keleto plonos plokštės elementų konstrukcijos stacionarių temperatūrų apskaičiavimą esant užduotiems šilumos šaltiniams, temperatūroms ir konvekcijos per paviršių sąlygoms. Pirmiausia įvertiname teorinius niuansus, kurie įtakoja galutinius rezultatus. Pagrindinės formulės, naudojamos pateiktoje MATLAB programoje yra pateikiamos žemiau:

$$\left[\mathbf{C}^e\right] = \mu \iiint_V \left[\mathbf{N}\right]^T \left[\mathbf{N}\right] dV \qquad (1)$$

Matrica C apskaičiuojama formule (1). Tai yra kūno šiluminio talpumo matrica, nurodanti kūno vidinę savybę kaupti šilumą. Tai yra nuo elemento tūrio priklausanti savybė, kurios pagrindinę reikšmę sudaro koeficientas μ - kūno savitąją šiluminę talpą apibūdinantis koeficientas.

$$\left[\mathbf{K}^e\right] = \iiint_V \left[\mathbf{B}\right]^T \left[\mathbf{D}\right]\left[\mathbf{B}\right] dV + \alpha \int_{S_q} \left[\mathbf{N}\right]^T \left[\mathbf{N}\right]\, dS + \xi \iiint_V \left[\mathbf{N}\right]^T \left[\mathbf{N}\right] dV \qquad (2)$$

Pagrindinę kūno dinamiką parodanti matrica K, apskaičiuojama (2) formule. Tai yra konstrukcijos laidumo matrica. Šios lygties pagalba sistemoje atsižvelgiama į konstrukcijoje esančių dalių laidumą (matrica D – laidumo koeficientų matrica). Taip pat parametras α parodo medžiagos savybę, kaip greitai ši sugeba išspinduliuoti šilumą į aplinką. Tuo tarpu parametras ξ demonstruoja kūno savybę absorbuoti šilumą iš aplinkos (absorbcijos koeficientas).

$$\left\{\mathbf{P}^e\right\} = \iiint_V \left[\mathbf{N}\right]^T b\, dV \tag{3}$$

Matrica P – poveikiai baigtinio elemento mazgams, dėl žinomų srities viduje veikiančių šilumos šaltinių. Vienintelis parametras b, nurodantis šio baigtinio elemento spinduliuojamą šilumos kiekį.

$$\left\{\hat{\mathbf{S}}^e\right\} = -\int_{S_q} \left[\mathbf{N}\right]^T q_0 \tag{4}$$

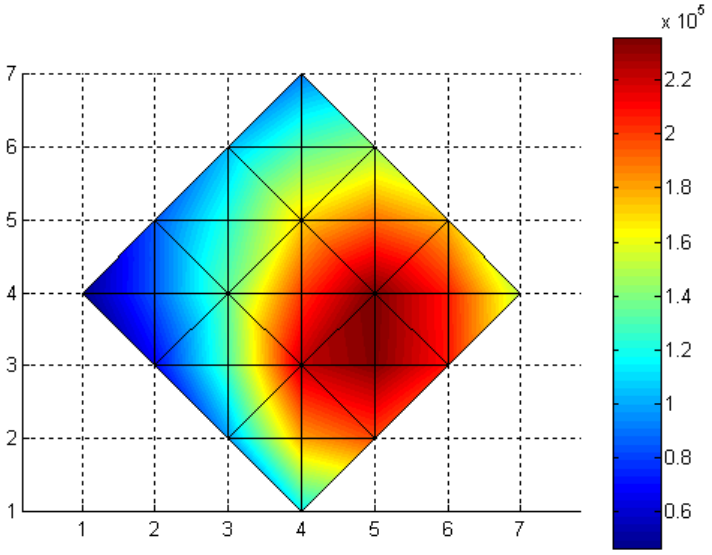
$$\left\{\mathbf{S}_\infty^e\right\} = \alpha\Phi_\infty \int_{S_q} \left[\mathbf{N}\right]^T dS \tag{5}$$

S – poveikiai baigtinio elemento mazgams dėl Koši kraštinėje sąlygoje pateiktų šilumos srauto tankių (poveikiai iš elemento vidaus). S_∞ - aplinkos poveikis elemento mazgams per jų paviršių.

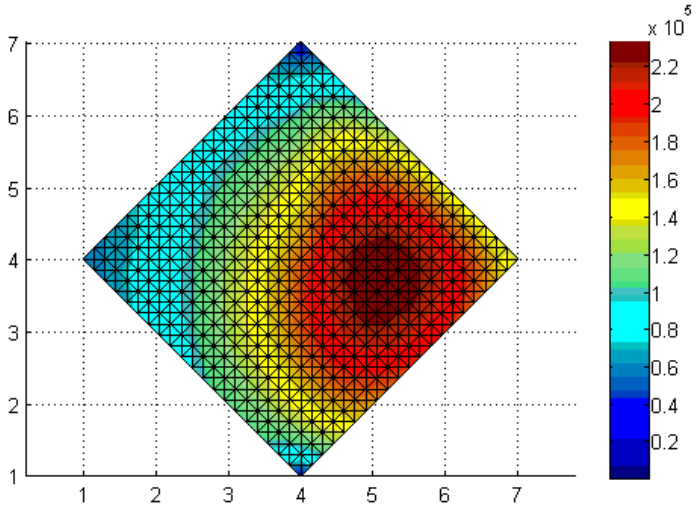
Visiems pagrindiniams šiluminio laidumo procesams užtenka naudoti tik (1) – (5) lygtimis išvestas matricas. Visas šias matricas sudėjus į vieną išraišką (6) galimas šios konstrukcijos būsenos apskaičiavimas pasibaigus pereinamiesiems procesams. Nežinomas šilumos reikšmes sukėlus į kairiąją pusę, o visus žinomas dalis palikus tik dešiniojoje, turėsime paprastą matricinę lygtį.

$$\begin{bmatrix} \mathbf{M}^e & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{\Phi}}^e \\ 0 \end{Bmatrix} + \begin{bmatrix} \mathbf{C}^e & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{\Phi}}^e \\ 0 \end{Bmatrix} + \begin{bmatrix} \mathbf{K}^e & -\mathbf{H}^{eT} \\ \mathbf{H}^e & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{\Phi}^e \\ \mathbf{\Lambda}^e \end{Bmatrix} = \begin{Bmatrix} \mathbf{P}^e + \hat{\mathbf{S}}^e + \mathbf{S}_\infty^e \\ \mathbf{\Phi}_0^e \end{Bmatrix} \tag{6}$$

Atlikus programos modeliavimą MATLAB aplinkoje, gaunami žemiau paveikslėlyje (13 pav.) pateikiami rezultatai.



62 pav. Programos moduliacijos rezultatai



63 pav. Programos moduliacijos rezultatai

Aiškiai matome, kad naudojant mažą kiekį elementų (13 pav.), gaunami iškreipti rezultatai – sistema vaizduoja, kad elementuose šiluma išspinduliuojama tiesiškai.

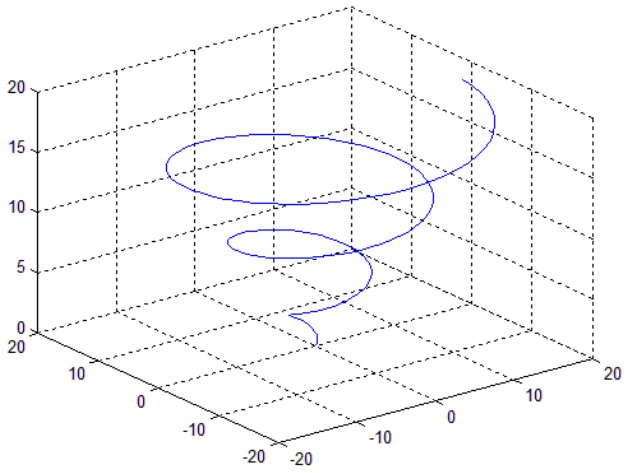
5.4. SISTEMOS MATLAB ANIMACIJOS TYRIMAS ERDVĖJE (3D)

Vienos populiaresnių priemonių animaciniams grafikams ir kitiems objektams erdvėje kurti [1]:

- plot3();
- comet3();
- surf();
- mesh();

Priemonė PLOT3*()* skirta kreivių braižymui erdvėje. Funkcijos panaudojimo metodai (sintaksės):

- *plot3(X1,Y1,Z1,...)*- 3D grafikų išvedimas. *X1,Y1,Z1*- vektoriai, kurių elementai, tai taškų koordinatės, per kurias brėžiama linija.
- *plot3(X1,Y1,Z1,LineSpec,...)* - 3D grafikų išvedimas. *X1,Y1,Z1*- vektoriai, kurių elementai, tai taškų koordinatės, per kurias brėžiama linija. *LineSpec* – simbolinis kintamasis (nebūtinas), kuriame gali būti iki 3 specialių simbolių, nurodančių kreivės charakteristikas: linijos tipą, mazgo (taško) tipą bei linijos spalvą.
- *plot3(...,'PropertyName',PropertyValue,...)*- *PropertyName* – naudojama savybė, *propertyvalue* – savybės reikšmė;



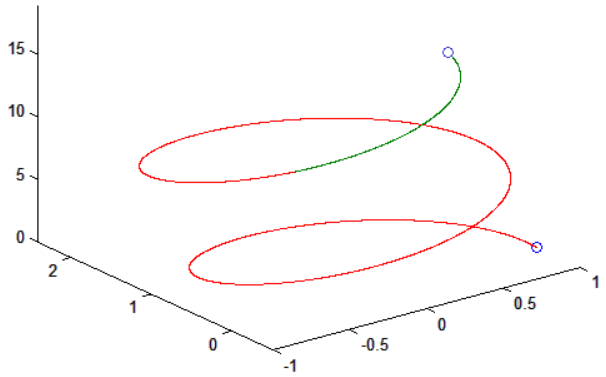
64 pav. MATLAB funkcijos plot3 pavyzdys

Priemonė *comet3()* skirta vaizduoti erdvėje judantį tašką. Funkcijos panaudojimo metodai (sintaksės):

- *comet3(z)* – erdvėje vaizduoja judantį tašką su spalvota „uodega“, kuris juda kreive, apibrėžta masyvu z.
- *comet3(x,y,z)* – erdvėje vaizduoja kometą, kuri juda kreive, einančia per taškus *[x(i),y(i),z(i)]*.
- *comet3(x,y,z,p)* – leidžia tiksliai apibrėžti kometos kūno ilgį *p*length(y)*. Numatytasis *p =0,1*.

Pvz.,

```
w=0:pi/1000:6*pi;
comet3(cos(w),sin(w)+w/10,w);
```

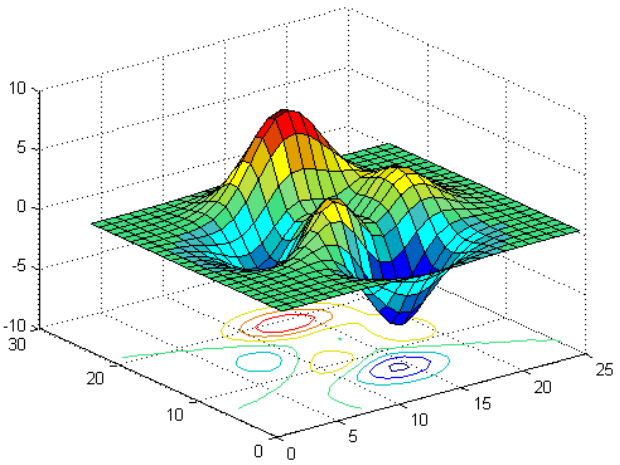


65 pav. MATLAB funkcijos comet3() pavyzdys

Grafikas, taip pat kaip ir plokštumoje, yra atvaizduojamas trinant ir nupiešiant vienu žingsniu į priekį pasislinkusį veiksmą. Taško judesio kelias kreive dvimatėje ir trimatėje erdvėje yra vienas elementariausių animacijos pavyzdžių MATLAB matematinėje sistemoje. Tačiau tokia paįvairinta animacija gali išspręsti daug grafinio modeliavimo uždavinių. Sistemoje MATLAB įmanoma modeliuoti sudėtingesnę animaciją trimatėje erdvėje, pagrįstą kadrų piešimu ir jų atvaizdavimu ekrane remiantis eiliškumo principu [5]:.

Priemonė SURF() skirta grafiškai pavaizduoti dviejų kintamųjų funkcijos paviršių. Funkcijos panaudojimo metodai (sintaksės):

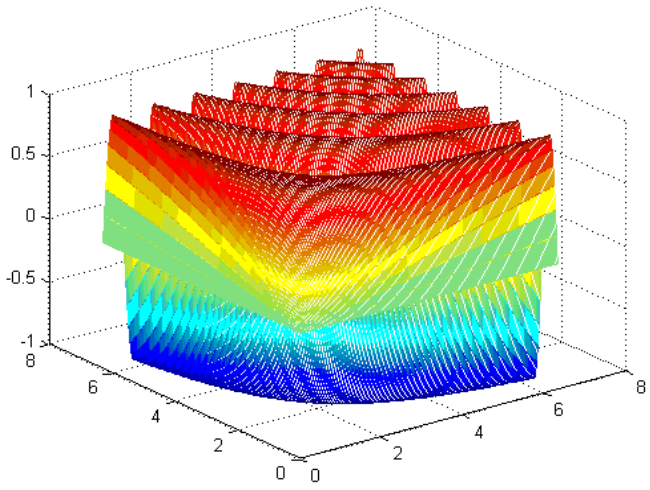
- *surf(Z)* - sukuria trimatį paviršių atsižvelgiant z komponentų matricą Z, pagal nutylėjimą nusakant, kad x=1:n ir y=1:m, kur [m, n] = size(Z). Aukštis, Z, yra vienareikšmiškai funkcija, apibrėžta geometrine stačiakampe spalvų gardele. Z nurodo spalvos duomenis, taip pat paviršiaus aukštį, todėl spalvos yra proporcingas paviršiaus aukščiui.
- surf(Z,C) - matrica C, proporcinga matricos Z dydžiui, yra naudojama paviršiui nuspalvinti.
- *surf(X,Y,Z)* - X ir Y – funkcijos argumentų reikšmių matricos, Z – funkcijos reikšmių matrica. Spalvų mastelis parinktas pagal funkcijos reikšmę.
- *surf(X,Y,Z,C)* - X ir Y – funkcijos argumentų reikšmių matricos, Z – funkcijos reikšmių matrica. Spalvų mastelis parinktas pagal matricą C.
- *surf(...,'PropertyName',PropertyValue)* - *'PropertyName',PropertyValue* nurodo paviršiaus savybės atsižvelgiant į duomenis.
- *surfc(...)* – piešiami paviršiaus kontūrai ant plokštumos po pačiu paviršiumi.



66 pav. MATLAB funkcijos surfc() pavyzdys

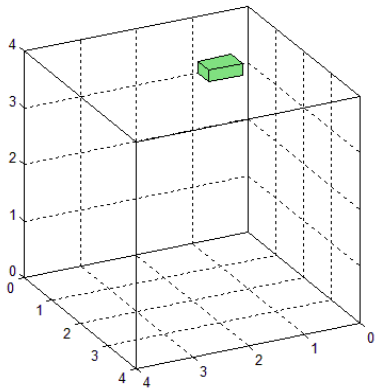
Priemonė MESH() skirta grafiškai pavaizduoti dviejų kintamųjų funkcijos paviršių tinkleliu. Funkcijos panaudojimo metodai (sintaksės):

- *mesh(X,Y,Z)* - X ir Y – funkcijos argumentų reikšmių matricos, Z – funkcijos reikšmių matrica. Spalvų mastelis parinktas pagal funkcijos reikšmę.
- *mesh(Z)* - sukuria trimatį paviršių atsižvelgiant z komponentų matricą Z, pagal nutylėjimą nusakant, kad x=1:n ir y=1:m, kur [m, n] = size(Z). Aukštis, Z, yra vienareikšmiškai funkcija, apibrėžta geometrine stačiakampe spalvų gardele. Z nurodo spalvos duomenis, taip pat paviršiaus aukštį, todėl spalvos yra proporcingas paviršiaus aukščiui.
- *mesh(...,C)* - naudojamas spalvos apibrėžimui.
- *mesh(...,'PropertyName',PropertyValue,...)* - nurodo paviršiaus savybės atsižvelgiant į duomenis.
- meshc(...) - piešiami tinklelio kontūrai ant plokštumos po pačiu tinkleliu.
- meshz(...) - piešiami tinklelio uždangos ant aplink patį tinklelį.



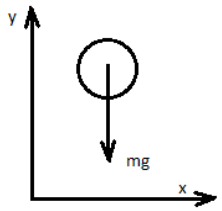
67 pav. MATLAB funkcijos mesh() pavyzdys

Remiantis ankščiau minėtos MATLAB funkcijos comet3() principu vaizdavimo (kadrų perpiešimu) ir plokštumoje atliktu tyrimu atliekamas tyrimas trimatėje erdvėje. Modeliuojama veiksmų seka: plytos (stačiakampio gretasienio) kritimas iš pasirinkto aukščio (12 pav.).



68 pav. Sumodeliuoto modelio pradžios vaizdas

Kaip ir plokštumoje, taip ir erdvėje įvertiname išorinius ir vidinius fizikos faktorius. Kadangi pateiktame modelyje nėra sąveikų tarp objektų, tai vienintelė jėga pastoviai veikianti objektą (stačiakampį gretasienį) yra tik žemės trauka (13 pav.). Ši jėga yra išorinė, ir veikia tik statmenai $\vec{x}$ vektoriui.



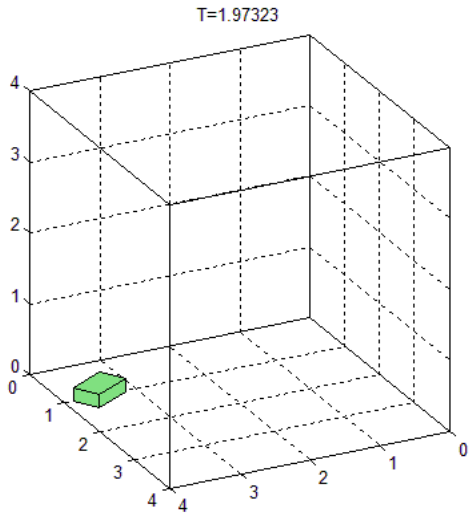
69 pav. Žemės traukos vaizdas

Kad kūno sąveika atrodytų realistiška, baudos elemento klampumo bei tamprumo koeficientai parenkami atsižvelgiant į sąveikaujančių objektų masę. Literatūroje [*11*] pateikiama formulė:

$$c \geq (1.6 \div 2) \times \sqrt{mk} \quad (1)$$

Formulė (1) skirta apskaičiuoti orientaciniam klampumo koeficientui tarp objekto ir kraštinių sienų (dugno). Joje įrašoma objekto, sąveikaujančio su kraštine sieną masė.

Atlikus moduliaciją MATLAB sistemoje iki galo gauname rezultatus pateiktus žemiau (pav.14):



70 pav. Plytos kritimo moduliacijos rezultatai erdvėje

Kaip buvo minėta anksčiau, ši sudėtingesnė matematinės sistemos MATLAB animacija yra paremta principu, kad funkcijos grafiką su pradinėmis kintamųjų reikšmėmis iš karto vaizduoti ekrane, jį trinti, perpiešti, keičiant kintamųjų reikšmes, vėl trinti, iš naujo perpiešti su kitomis reikšmėmis ir t.t. (animacijos kadrai iš anksto nekuriami!). Kitas būdas yra sukurti seką kadrų, kuriuos vėliau rodyti vieną paskui kitą.

Šis animacijos kūrimo būdas vadinamas filmo (*movie*) kūrimu [3]. Toks metodas labiau naudojamas tada, kai animacijos kadruose atvaizduojami labai sudėtingi ir nelabai greitai perpiešiami objektai ir jų savybės. Tada kiekvieną animacijos kadrą rekomenduojama piešti iš anksto. Animacinės moduliacijos metu bus rodoma sukurtų kadrų seka.

Filmo kūrimas susideda iš dviejų etapų:

1. Komanda *getframe* - animacijos kadrų kūrimas.

2. Komandą *movie* – animacijos vykdymas.

Kiekvieno kadro sukūrimas atliekamas užfiksuojant vaizdą, grafiką ar kitą elementą su vis kitais kintamųjų parametrais. Komanda *getframe* veikia cikle *for*, kadangi reikia sukurti animacinį kadrų masyvą. Tarkime, kad ciklo *for* kintamasis *j* kinta nuo 1 iki 10, žingsniu 0,1 (*j = 1:.0.1:10*), tai bus sukurtas 101 kadro animacija.

Pvz.,

```
for j = 1:n
vaizduojamos funkcijos apibrėžimas
A(j) = getframe;
end
movie(A)
```

Komanda *movie()* įvykdo animaciją, kuri išreiškiama matrica. Matricos stulpeliai - animacijos kadrai. Jei MATLAB funkcijos *movie()* programoje neįtraukiame, tai animacijos

veikimo metu matome eilės tvarka funkcijos *getframe()* kuriamus kadrus, tačiau animacinis filmas (movie) nėra rodomas.

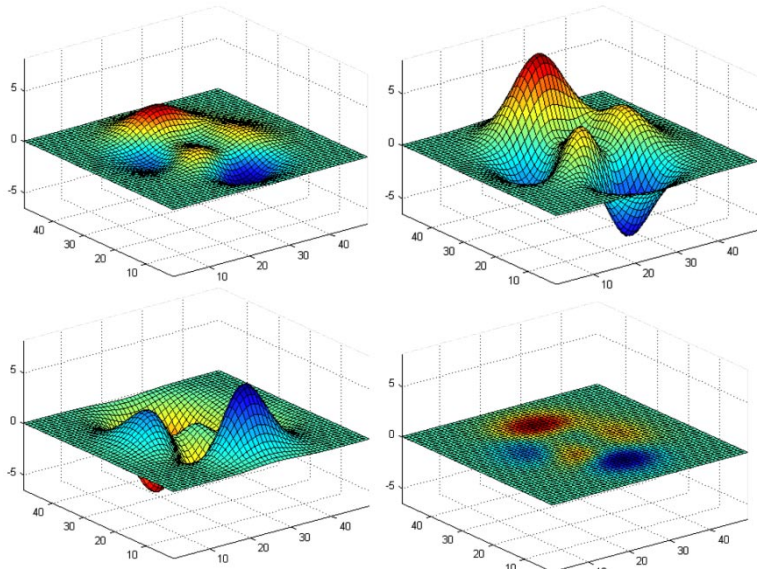
- *movie(M)* – masyvo M vaizdavimas vieną kartą.
- *movie(M,n)* – animacijos vykdymas n kartų. Jei n – neigiamas skaičius, animacija kiekvieną kartą (n kartų) yra rodoma į priekį (nuo pirmo kadro iki paskutinio) ir priešinga kryptimi. Jei n yra vektorius, pirmasis skaičius reiškia animacijos vykdymo kartojimų skaičių, likusieji elementai – animacijos kadrų, kurie sudarys filmą, sąrašą.

Pavyzdžiui, jei M sudaro keturi kadrai ir n = [10 4 4 2 1], animacija bus vykdoma dešimt kartų. Du kartus vaizduojamas 4 kadras, tada – 2 ir 1 kadrai. Šiuo būdu animaciją galima kurti ir plokštumoje, ir erdvėje.

Dažnai naudojama dviejų kintamųjų komanda *peaks*, pritaikyta sudėtingesniems brėžiniams kūrti. Komanda generuoja skaičių matricą, gautą įvairiomis transformacijomis iš normaliųjų skaičių skirstinių.

Pvz.,

```
A = peaks; surf(A); % surf() - vaizduoja spalvotus paviršius
axis tight; %Nustatoma grafiko sritis (tikslios ašių ribos)
%Nustatome, kad vykdymo metu išliktų tas pats mastelis ir grafiko
%spalvos
set(gca,'nextplot','replacechildren');
%Cikle sukursime 20 kadrų masyvą
for j = 1:20
      surf(sin(2*pi*j/20)*S,S)
      A(j) = getframe;
end
movie(A,2) % animaciją vykdysime 2 kartus
```

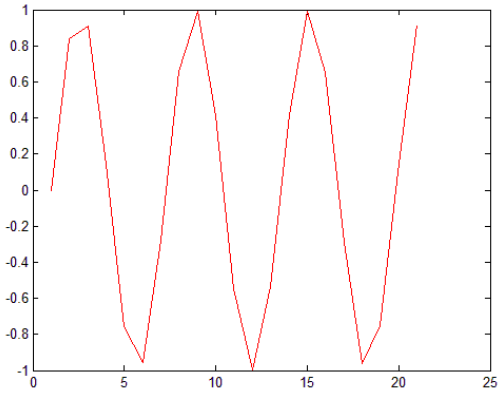


71 pav. MATLAB sistemos komandos movie() pavyzdys

Anksčiau tirtas animacijos kūrimo būdas (plytos kritimo moduliacija) – trynimo metodas (*Erase Mode*) – naudoja MATLAB braižymo komandas. Šis animacijos kūrimas yra greitesnis, tačiau nukenčia grafiko vaizdavimo tikslumas. Todėl reikia apgalvoti, kurį metodą tikslingiau pasirinkti.

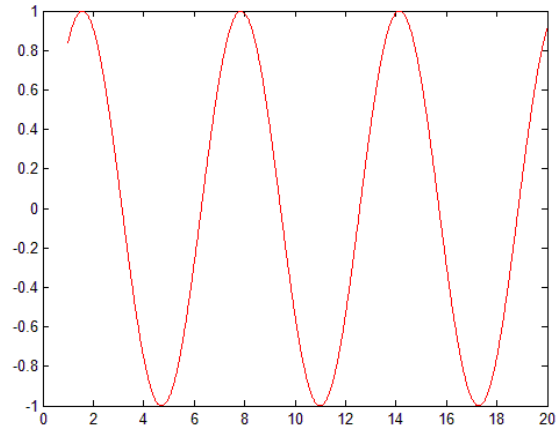
5.5. SISTEMOS MATLAB ANIMACIJOS PLOKŠTUMOJE (2D) IR ERDVĖJE (3D) TIKSLUMAS IR PAKLAIDOS

Priemonės *PLOT()* tikslumo nustatymui naudojama keletas skirtingai aprašytų parametrų būdų. Pirmas būdas yra vienas paprasčiausių: *plot(sin(0:20),'r')*. Brėžiamas sinuso grafikas (0:20) ribose. Linijos spalva pasirenkama raudona.



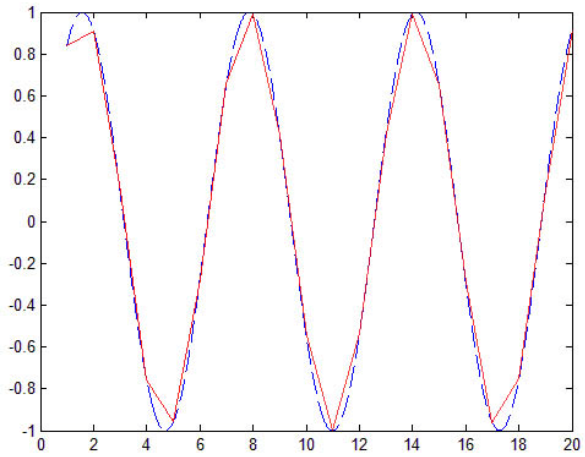
72 pav. MATLAB funkcijos plot(sin(0:20),'r')pavyzdys

Tačiau ne visada pats paprasčiausias būdas būna veiksmingiausias. Matome, kad grafikas lūžio taškuose yra kampuotas, todėl kreivė praranda tikslumą. Viso to priežastis yra kreivės ribos imamų taškų ribos. Norint išgauti suapvalintus lūžio kampus, to pasakoje ir tikslesniu rezultatus, reikia įterpti papildomus parametrus: nustatyti imamos ribos taškų smulkesnį tankį. Nustačius ribas (0:20), pagal nutylėjimą sistema MATLAB naudoja sveikų skaičių masyvą tose ribose, pavyzdžiui šitame atvejyje: 0, 1, 2, ..., 18, 19, 20. Kad gautume tikslesnius rezultatus, sistemai reikia aprašyti pageidaujamus parametrus smulkiau. Tarkime, kad grafikas būtų brėžiamas imant smulkesnį masyvą 0,01 tikslumu: 0.1, 0.2, 0.3, ..., 19.97, 19.98, 19.99, 20.00. Tai aprašoma labai nesudėtingai: *sin(1:0.01:20)*. Svarbu nepamiršti prieš aprašomą funkciją nurodyti ir x ašies tikslumą tokį pat kaip ir funkcijos. Kaip matome (9 pav.) grafiko kreivė tapo tikslesnė.



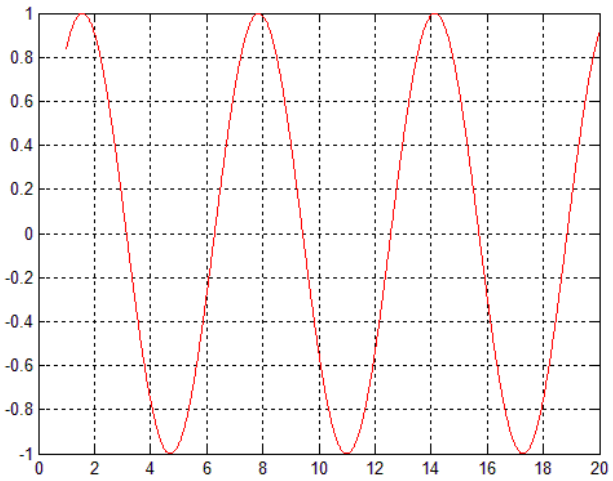
73 pav. MATLAB funkcijos plot(1:0.01:20, sin(0:20),'r') pavyzdys

Tai parodo ir kitas paveikslėlis (10 pav.), kur grafikų kreivės yra uždėtos viena ant kitos. Taigi galime daryti išvada, kad kuo smulkesnis intervalas imamų reikšmių, tuoj gauname tikslesnius rezultatus, tačiau eikvoja ir daugiau kompiuterio resursų.



74 pav. MATLAB funkcijos plot() su skirtingai parametrais palyginimo pavyzdys

Braižant grafikus būna svarbu nustatyti objekto koordinates plokštumoje. Čia labai gerai pasitarnauja MATLAB funkcija *grid.* Jos paskirtis yra uždėti plokštumai „tinklą“, kurio pagalbą galima geriau orientuotis aplinkoje. „Tinklas“ įjungiamas *grid on*, o išjungiamas *grid off*.



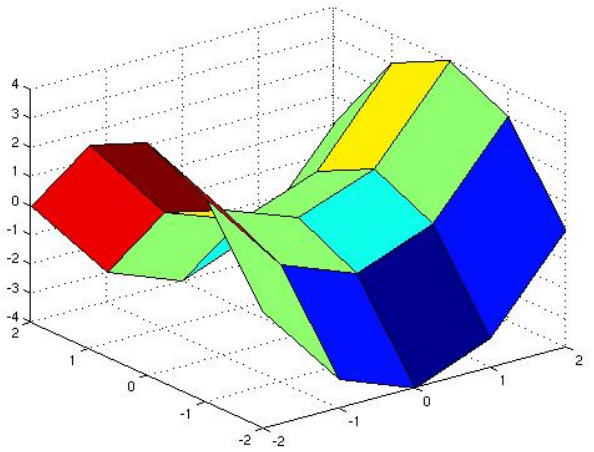
75 pav. MATLAB funkcijos grid on pavyzdys

Trimatis grafikos kūrimas yra vienas iš MATLAB privalumų. MATLAB vadovas turi vien tik grafikos dalies aprašymą apie 400 puslapių. Todėl kaip tikslumo tyrimui panaudosim vieną komandą turinti vieną iš geriausių savybių trimatės grafikos komandose – *surf()*.

Ši funkcija yra pagrindinis metodas naudojamas MATLAB sistemoje, kuri yra naudojama kartu su komanda *meshgrid*. Ši komanda sukuria matricą (x, y), kuri nusako paviršiaus vietą kur turi būti grafikas. Pavyzdžiui:

```
[x,y]=meshgrid(-2:1:2,-2:1:2);
surf(x,y,x.^2-y.^2)
```

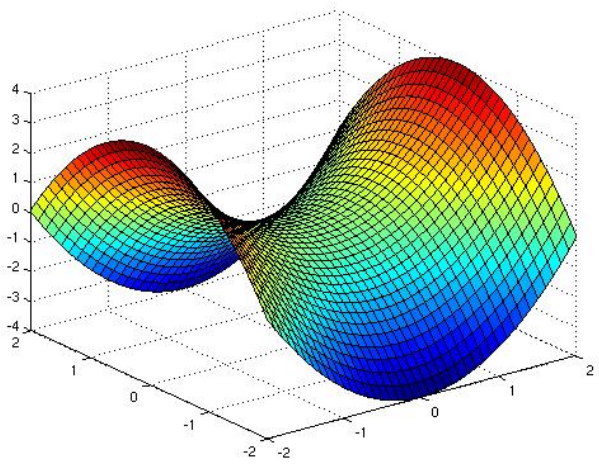
sukuriamos dvi matricos *x* ir *y*, kurios savyje saugo 25 taškų koordinates.



76 pav. MATLAB funkcijos surf pavyzdys

Kaip matome iš rezultatų (12 pav.) animacija gavosi kampuota, grafikas neturi tikslumo. Kad išlyginti paviršių, reikia pridėti daugiau taškų į *meshgrid* komandą:

```
[x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
surf(x,y,x.^2-y.^2)
```



77 pav. MATLAB funkcijos surf pavyzdys

5.6. SISTEMOS MATLAB ANIMACIJOS PLOKŠTUMOJE (2D) IR EDVĖJE (3D) IŠVADOS

1. Kuriant ir pritaikant 2D ir 3D kompiuterinę animaciją MATLAB sistemoje konkrečioms užduotims pastebime, kad yra labai svarbūs sprendimų būdai, atsižvelgiant į programinės ir techninės įrangų specifiką.
2. Tinkamos animacijos metodo pasirinkimas yra vienas iš svarbiausių uždavinio sprendimo akcentų naudojant kompiuterines animacijas kaip rezultatą.
3. Modeliuojant situacijas MATLAB aplinkoje yra svarbu pasirinktas duomenų, skaičiavimų ir rezultatų dažnio intensyvumas. Daugiau atliekama skaičiavimų – didesnė tikimybė gauti tikslesnius rezultatus.

6. ANIMACIJOS KŪRIMAS C++ PLATFORMOJE

6.1. C++ PLATFORMOS APŽVALGA

C++ – bendros paskirties programavimo kalba, viena populiariausių programavimo kalbų pasaulyje.

Pirmosios programavimo kalbos C++ versijos buvo sukurtos tos pačios “Bell Labs” kompanijos darbuotojo Brajeno Struastrupo kaip C kalbos išplėtimą, apibrėžiantį (*include*

failų – bibliotekų) rinkinį bei specialų *pre* - procesorių, suteikiantį C kalbai objektinio programavimo galimybes. Tai įvyko 1986 m. Į naują kalbą buvo įtraukta objektinio programavimo galimybė bei buvo ištaisytos ankstesnės versijos klaidos. Pirmieji šios kalbos vartotojai buvo "Bell Labs" kompanijos darbuotojai, o pirmasis komercinis transliatorius buvo parašytas 1993 m. Pirmuoju transliatoriumi tapo preprocesorius Croft, transliuojantis C++ kodą į alternatyvų jam C kodą. Kaip tik nuo tada atsirado knygų apie C++ ir jos greitai išpopuliarėjo. Dabar ši kalba skaitoma kaip viena svarbiausių kuriant didelius ir sudėtingus projektus. Žinoma, C++ kaip ir visa kita, turi ir trūkumų. C++ kalbos standartas patvirtintas 1998 metais [11].

C++ pasižymi labai dideliu lakoniškumu, sintaksinių struktūrų lankstumu ir universalumu, todėl šią kalbą dažniausiai pradedama mokytis jau turint programavimo kitomis kalbomis patyrimą.

6.2. C++ PLATFORMOS GALIMYBĖS

Programavimo kalba C++ yra vienos populiariausių iš visų modernių programavimo kalbų. Tuo pačiu ji nėra iš pačių paprasčiausių. Norint ją perprasti, reikia įdėti labai daug pastangų. Kita vertus ji yra labai funkcionali ir universali programavimo kalba. Turint atitinkamą kiekį žinių galima kurti tiek konsolines programas, tiek grafinę sąsają turinčias programas taip pat net 2D ar 3D žaidimus [11].

C++ geriau nei C, nes:

- Išsamesnis klaidu tikrinamas, lengvesnė klaidu paieška;
- Nuorodų (*reference*) panaudojimas funkcijų argumentuose ir grąžinamose reikšmėse yra patogesnis nei rodyklių (*pointer*).
- Funkcijų perkrovimas (*overloading*) leidžia naudoti tuos pačius funkcijų pavadinimus skirtingoms funkcijoms.
- Vardu erdvės (*namespace*) leidžia geriau kontroliuoti vardų naudojimą;
- ~10 % didesnis programų našumas nei C;
- Programos (didelės) lengviau taisomos bei modifikuojamos;
- Dėl klasių pritaikymo lengviau naudojamos bibliotekos;
- Šablonai (*template*) automatiškai modifikuoja programinį kodą, taip palengvinami bibliotekų naudojimą;
- Programinis kodas gali viršyti 50.000 eilučių;

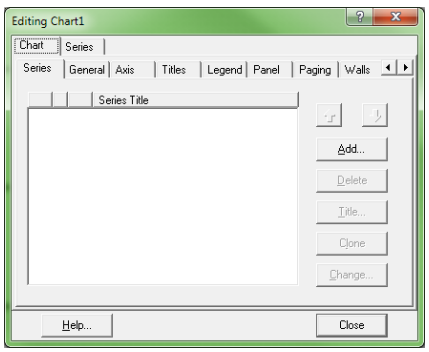
Pradedant programuoti su C++ kalba, siūloma atkreipti dėmesį į tokias šios kalbos ypatybes:

- yra tik viena programos struktūrizavimo priemonė – funkcija;
- programos sąvoką atitinka pagrindinė funkcija, kuri žymima vardu *main*;
- identifikatoriuose didžiosios ir mažosios raidės nesutapatinamos;
- programos objektų (struktūrų, kintamųjų, funkcijų) aprašai gali būti bet kurioje programos vietoje – svarbu tik tai, kad objektas būtų apibrėžtas prieš jį naudojant;
- aprašuose plačiai vartojami funkcijų prototipai;
- C++ kalboje nedidelis standartizuotų operatorių skaičius, todėl naudojamos įvairios bibliotekų sistemos;
- kreipiantis į kintamuosius ir struktūras, plačiai vartojamos rodyklės;
- daugelį duomenų tipų galima interpretuoti keliais įvairiais būdais.

6.3. C++ PLATFORMOS ANIMACIJOS TYRIMAS

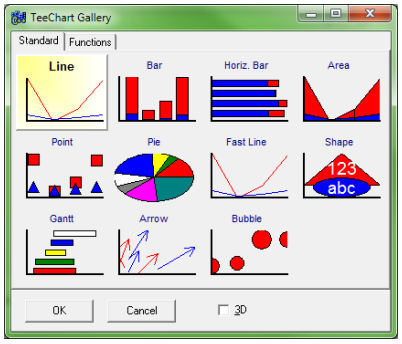
Atlikti animacijos piešimo tyrimą ir analizę C++ platformoje buvo pasirinktas Borland C++ Builder 6 kompiliatorius. Pirmiausia buvo pasirinkta siūlomas grafikų komponentas [12] – *Chart*  , kuris randasi *Additional*  paletėje. Planuojant programoje panaudoti diagramas, reikia žinoti, kaip parinkti diagramų formas ir nustatyti jų savybes, mokėti perduoti duomenis diagramų objektui. Vienoje diagramoje galima pateikti kelių duomenų sekų grafinius vaizdus.

Diagramos komponentą *Chart* įkėlus į programos formą, spragtelėjus dešiniuoju pelės klavišu, išskleidžiamas konteksto meniu. Pasirenkamas meniu elementas *EditChart...*. Ekrane atveriamas diagramų tvarkyklės langas (78 pav.). Jame yra dvi pagrindinės kortelės: *Chart* – tai bendrų diagramų komponento savybių sustatymai, ir *Series* – atskirų duomenų sekų savybių nustatymai. Kiekviena iš šių pagrindinių kortelių turi visą grupę savų kortelių.



78 pav. diagramų tvarkyklės langas

Mygtukas *Add* leidžia įtraukti naują seką. Spragtelėjus šį mygtuką, atveriamas diagramų formų langas (79 pav.). Pele pasirenkama diagramos forma bei jos vaizdavimo būdas – dvimatis arba trimatis (jei 3D žymimasis su varnele). Tvarkyklės lange pasirodo pasirinkta seka. Mygtukas *Title...* leidžia sekai suteikti savo pavadinimą. Sekas naudinga pavadinti, jeigu naudojamos kelios sekos (*Series1, Series2* ir t.t.). Tai sumažina painiavą tarp sekų pavadinimų.



79 pav. Diagramų formos

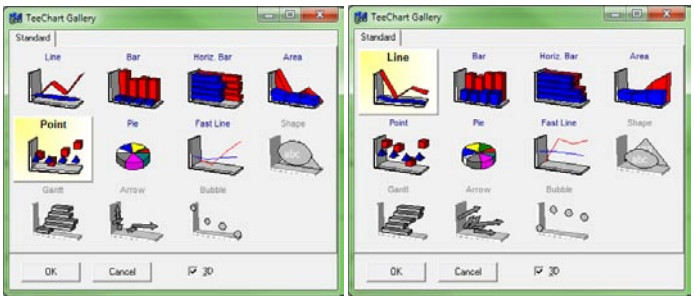
Pamėginkime nubrėžti dviejų funkcijų grafikus: $y = sin(x)$ ir $y = cos(x)$. Grafikus vaizduosime argumento *x* kitimo intervale nuo 0 iki 2**Pi*, kur *Pi* = 3,1415. Tiek ši konstanta (*M_PI*), tiek ir funkcijos *sin(x)* bei *cos(x)* aprašytos bibliotekoje *math.h*, todėl šią biblioteką reikia įtraukti į programą: *#include <math.h>*.

Parenkame vienodus dvimačius sekos tipus *Line* (langas *TeeChart Gallery*, kortelė *Standart*) abiejų grafikų duomenims saugoti. Pirmajai sekai *Series1* suteikiame pavadinimą *Sinusas*, antrajai – *Kosinusas*. Sekai reikalingus duomenis – argumento reikšmę, funkcijos reikšmę, atitinkamą užrašą bei spalvą suteiksime atitinkamos sekos (*Series1, Series2*) metodo *AddXY()* pagalba.

Abiem duomenų sekoms nustatome linijos storį 3 taip: pasirenkame *Series* kortelę, nustatome sekos tipą (*Series1* arba *Series2*) ir kortelė *Format* spragtelime mygtuką *Border* (*Line* dalyje). Toliau langelyje *Withd* užrašome linijos storį. Nustatę linijos storį antrajai sekai, į programą grįžtame uždarydami *Chart* tvarkyklę. Programos darbo rezultatas atvaizduotas 80 pav..

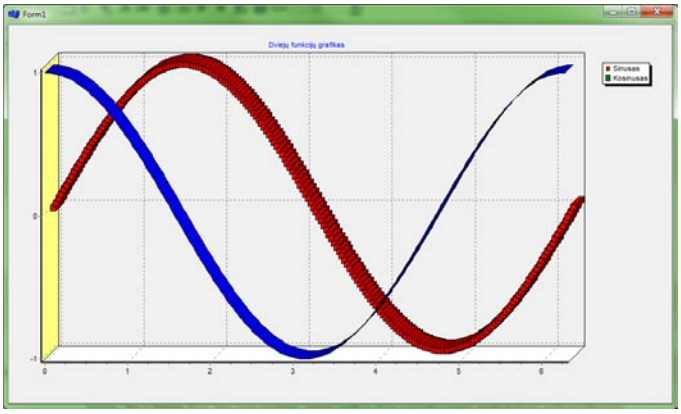
80 pav. Funkcijų sin(x) ir cos(x) grafikai

Pasirinktas kompiliatorius suteikę galimybę grafikus matyti ir trimačiam (3D) pavidale. Tereikia uždėti varnelę ant laukelio 3D *TeeChart Gallery* lange. Šį kartą parenkame skirtingus trimačius sekos tipus: sekai *Sinusas* parenkame *Point*, o *Kosinusas* sekai parenkame *Line* grafikus duomenims saugoti.



81 pav. Funkcijų sin(x) ir cos(x) trimčio atvaizdavimo pasirinkimas

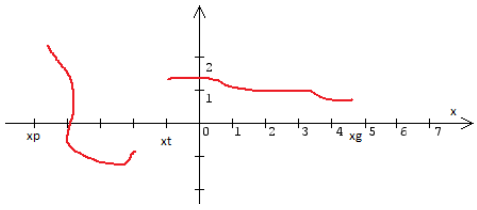
Tačiau reiktų atkreipti dėmesį į gautus rezultatus. Gavome grafikus tokius pat kaip ir dvimačiam atvaizdavime, todėl galima teigti, kad 3D vaizdavimas yra tik vizualus. Grafikas yra braižomas dviejų ašių atžvilgiu, bet piešiama trimačio vaizdo iliuzija (82 pav.).



82 pav. Funkcijų sin(x) ir cos(x) trimčio grafiko vaizdas

Kompiliatoriaus Borland C++ Builder siūlomas komponentas *Chart* turi begalę redagavimo įrankių, todėl vartotojas-programuotojas gali ganėtinai lengvai nusistatyti jam reikalingus brėžinio nustatymus. Tačiau šis komponentas ne visuomet yra patogus, ypač kai reikia grafiškai stebėti funkcijas, turinčias trūkio taškus, analizuoti, kaip kinta vaizdas, priklausomai nuo funkcijos parametrų.

Funkcijų grafikus galima piešti komponentų *Canvas* lauke. Vienas paprasčiausių būdų – koordinačių plokštumą sutapatinti su *Canvas* lauku ir jame piešti funkcijos taškus. Kuo daugiau ir tankiau taškai bus dedami lauke, tuo funkcijos vaizdas bus panašesnis į ištisinę liniją. Jis į liniją bus dar panašesnis, jeigu taškai bus vaizduojami nedideliais pilnaviduriais apskritimais.



83 pav. Funkcija su trūkio tašku

Kitas labai populiarus būdas piešti grafiką – taškų sujungimas tiesės atkarpomis. Kuo tankiau dedami sujungiami taškai, tuo laužtė panašesnė į ištisą liniją. Jeigu funkcija $y = F(x)$, kai $xp \le x \le xg$, hx (čia $xp - x$ paskutinė reikšmė, $hx - x$ keitimo dydis – žingsnis), turi trūkio taškus, tuomet juos stebėti, kai grafikas nupieštas tik iš taškų, yra sunku. Kai grafikas piešiamas iš atkarpų, grafiko linija nutrūksta intervale nuo $y1 = F(xt - hx)$ iki $y1 = F(xt + hx)$, jeigu funkcijos reikšmė taške $y = F(xt)$ neegzistuoja. Jeigu yra kelios iš eilės einančios x reikšmės, kurioms funkcija neegzistuoja, tuomet linija nepiešiama didesniam intervalui. Žinoma, vaizdas yra apytikslis, nes pirmiausia jis nuo hx reikšmės. Kuo ji didesnė, tuo labiau didėja tikimybė, kad trūkio taškas bus „nepastebėtas“. Vaizdo tikslumas priklauso nuo vaizduojamo grafiko mastelio, kuris suvedamas į kiek ekrano taškų atitinka funkcijos vieną reikšmę. Vaizdingumui funkcijos intervalo, kuriame reikšmės neegzistuoja, taškus galima sujungti kitos spalvos linija (pvz., taškas *y1* ir *y2*). Tai naudinga tuo atveju, kai funkcijos kreivė sudėtinga, o trūkio intervalų ne vienas.

Piešiant funkcijos grafiką yra sprendžiami du pagrindiniai uždaviniai:

- piešimo lauko paruošimas, t.y. koordinačių plokštumos, kurioje skaičiuojamos funkcijos reikšmės, suderinamas su *Canvas* lauko dydžiu;
- funkcijos grafiko piešimas.

Piešimo lauko paruošimas prasideda nuo lauko valymo. Tai būtinas veiksmas, nes perpiešiant funkciją su kitomis parametrų reikšmėmis, senasis vaizdas automatiškai nenaikinamas. Laukas valomas jame brėžiant stačiakampį, kurio dydis sutampa su lauko dydžių, užpildant norima spalva ir raštu.

Tada yra atliekama koordinačių plokštumos centro vietos paieška. Paprasčiausia koordinačių pradžios tašką (0, 0) padėti *Canvas* lauko viduryje: *Canvas->Height/2, Canvas->Withd/2*. Tačiau, pavyzdžiui, žinant, kad visos funkcijos reikšmės bus tik pirmajame ketvirtyje, koordinačių pradžios tašką tikslinga sutapatinti su *Canvas* lauko apatiniu kairiuoju kampu *(0, Canvas->Height)*. Tam reikia surasti didžiausią *dy* ir mažiausią *my* funkcijos $y = F(x)$ reikšmę duotame argumento x kitimo intervale $xp \le x \le xg$, hx. Po to galima skaičiuoti mastelių horizontalią *xm* ir vertikalią *ym* reikšmes taškais:

```
xm = Canvas->Withd / (abs(xp-xg));
ym = Canvas->Height / (abs(dy-my));
```

Gautos reikšmės turi būti koreguojamos iki sveikojo skaičiaus, nes jos rodo, kiek taškų tenka vienam koordinačių plokštumos vienetui. Mažiausia galima reikšmė yra vienetas. Taigi egzistuoja galimybė, kad visas funkcijos grafiko paveikslas netilps *Canvas* lauke. Tuomet reikia numatyti galimybę vaizduoti norimą dalį grafiko arba pakeisti argumentų dydžius. Koordinačių pradžios taškas (*xk, yk*) ekrane bus randamas:

*xk = abs(xp)*xm;*

*yk = abs(dy)*ym;*

Kitas žingsnis yra koordinačių ašių piešimas. Per surastą tašką (*xk, yk*) *Canvas* lauke brėžiamos dvi linijos:

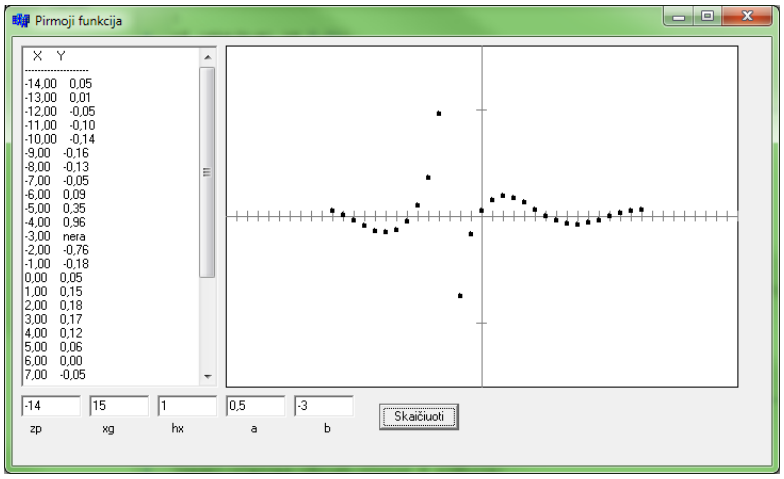
Canvas->MoveTo(0, yk); Canvas->LineTo(Canvas->Withd, yk);

Canvas->MoveTo(xk, 0); Canvas->LineTo(xk, Canvas->Height);

Po to linijų galuose galima nupiešti rodykles ir užrašyti pavadinimus. Būtina įvertinti tai, kad gali būti ne visi keturi ketvirčiai, taigi ir rodyklių bei pavadinimų linijoms gali prireikti mažiau. Rašant realias programas reikėtų *Canvas* lauko dydžius imti keliais taškais mažesnius, nes vaizdas bus ne koks, kai ašinė linija sutaps su lauko riba.

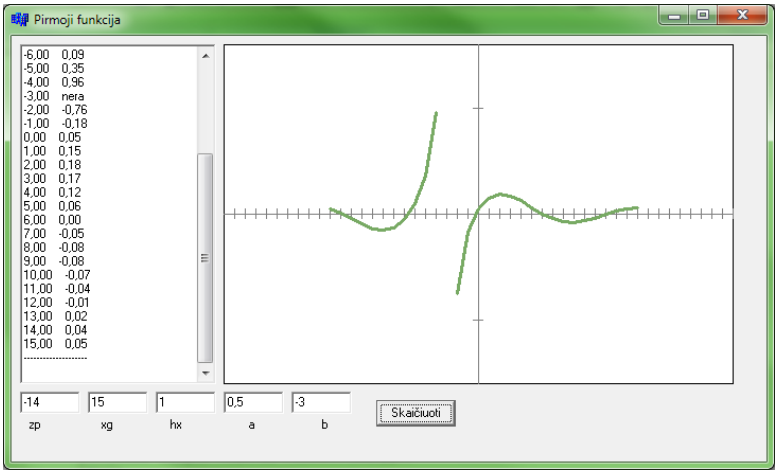
Dabar uždėkime žymas ant ašių. Jų kiekis ir intervalas priklauso nuo pasirinkto mastelio ir funkcijos argumento kitimo žingsnio. Žymos vaizduojamos kaip trumpos tiesės atkarpos, statmenos ašinėms linijoms. Reikšmės prie žymų surašomos įvairiai. Horizontalios ašies reikšmė dažniausiai rašoma žemiau ašies, o vertikalios – dešiniau. Tačiau jeigu horizontali ašis sutaps su apatine *Canvas* lauko riba, žymų reikšmes teks rašyti virš ašies. Žymų reikšmių nebūtina rašyti prie kiekvienos žymos, jeigu jos yra labai arti viena kitos. Panašiai elgiamasi ir vertikalios ašies atžvilgiu. *Canvas* lauko paruošimas yra sudėtingiausia dalis.

Kai piešimo laukas yra paruoštas, galima brėžti grafiką. Funkcijos grafiko reikšmės skaičiuojamos ir koreguojamos pagal *Canvas* lauke vaizduojamą koordinačių plokštumą (centro koordinates ir mastelius). Po to *Canvas* lauke dedamas taškas arba brėžiama tiesės atkarpa į tą tašką, priklausomai nuo pasirinkto grafiko piešimo būdo. Žinoma, reikia įvertinti tai, ar funkcijos reikšmė egzistuoja. Kaip pavyzdį nubrėšime funkcijos $y = \frac{\sin(ax+b)}{b-x}$ grafiką, kai duotos *a* ir *b* reikšmės, o *x* reikšmė kinta nuo *xp* iki *xg* žingsniu *hx*. Funkciją reikia piešti komponento Image Canvas lauke, atvaizduojant taškais. Gauname grafiką:



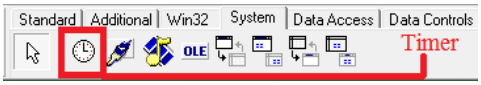
84 pav. Funkcija piešiama taškais

Atlikus kelias modifikacijas programiniame tekste, grafiko piešimo metode, galime nubrėžti ir liniją, kuri sujungtu taškus į kreivę:



85 pav. Funkcija piešiama atkarpomis

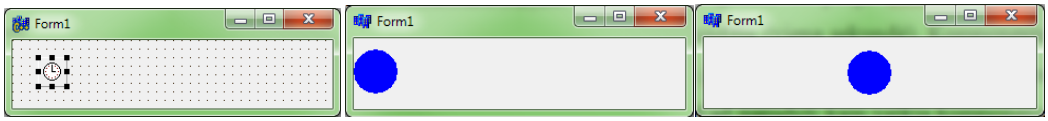
Tobulesnius grafikus galima piešti pasitelkiant judesį – animaciją. Animacija paprastai suprantama kaip judantis ir ekrane besikeičiantis paveikslas. Paprastesnis atvejis – kai atvaizdas tik juda arba tik keičiasi. Paprasčiausią atvaizdą galima nupiešti iš geometrinių figūrų. Priversti jį judėti ekrane nesudėtinga. Tereikia jį perpiešti kitoje ekrano vietoje. Buvusią paveikslėlio vietą reikia išvalyti. Judesio natūralumas priklauso nuo perpiešimo greičio ir postūmio nuo senosios vietos dydžio. Taip pat priklauso ir nuo atvaizdo dydžio. Laikui kontroliuoti skirtas *Timer* komponentas. Jis yra *System* paletėje (84 pav.).



86 pav. Komponentas Timer paletėje System

Komponentas turi vieną įvykio *OnTimer* metodą. Jis kviečiamas darbui intervalu, kuris saugomas savybėje *Interval*. Pagal numatytąjį nustatymą jo reikšmė yra 1000 milisekundžių (viena sekundė). Komponento darbas stabdomas savybei *Enabled* suteikiant reikšmę *False*. Komponentas darbo metu ekrane nematomas.

Kad parodyti kaip veikia komponentas, sumodeliuojame situaciją. Judantis skritulys pasirodo kairėje programos lango pusėje ir perėjęs per visą langą dingsta dešinėje. Ir taip jis juda cikliškai. Cikliškai jis juda tiek kartų, kiek yra nurodyta programoje. Pasibaigus ciklui, jis sustoja ekrano centre.



87 pav. Programos forma ir darbo langai

Nuoseklus skritulio pasirodymas iš už lango ribų efektas išgaunamas, kai pradinė skritulio koordinatė yra už lango ribų. Taip pat elgiamasi ir skritulio išėjimo iš lango atveju.

Trimatė grafika ir animacija C++ platformoje reikalauja didelių kompiuterio resursų ir didelio kiekio papildomų priedų (DirectX, OpenGL ir kt. paketų). DirectX tinka tik žaidimų

kūrimui ir tik Windows OS. Visiems kitiems darbams rekomenduojama OpenGL, nes ji yra ne tik paprastesnė, bet dažnai ir efektyvesne. Norint programuoti OpenGL, reikės kokios nors vartotojo sąsają galinčios vaizduoti bibliotekos. Keletas variantų yra:

1. Win32 (WinApi) - veikia tik Windows OS, programuojama C kalba, tačiau programinis kodas gana sudėtingas.
2. SDL arba SFML - šios bibliotekos labiau tinkamos OpenGL nei Win32. Pirmojoje programuojama C, antrojoje - C++ kalba.
3. Qt - šiuo metu labai populiarėjanti programavimo aplinka ir biblioteka. Didžiausias privalumas - labai didelis palaikymas ir nesudėtingas vartotojo sąsajos kūrimas.

Yra ir daugiau pasirinkimų, bet šitie yra vieni iš populiariausių.

6.4. C++ PLATFORMOS ANIMACIJOS TIKSLUMAS IR PAKLAIDOS

Komponentas *Chart* ir įvairios jo panaudojimo galimybės atvaizduojant programų rezultatus grafikais ar diagramomis turi daugybę nustatymų ir platų valdymo asortimentą. Tai šiam komponentui suteikia gana galingos priemonės statusą, tenkinantį ir įnoringą programos užsakovą. Tačiau kuo daugiau nustatymų, tuo didesnė tikimybė blogai pasirinkti tinkamus parametrus, dėl kurių priklauso grafiko tikslumas.

Dėl tokių pat priežasčių klaidos ir netikslumai gali atsirasti ir piešiant *Canvas* priemonėje. Naudojant šitą komponentą reikia nesuklysti braižant koordinačių ašis ir dedant ant jų atžymos taškus. Kitas svarbus aspektas, kad grafikas su visom koordinačių ašimis būtų piešiami nustatant vieną atskaitos tašką.

Taip pat yra svarbu pasirinktos linijos ar taško storis. Per didelis objekto storis gali užgožti kitą svarbią brėžinio detalę.

6.5. C++ PLATFORMOS ANIMACIJOS IŠVADOS

1. C++ platforma padedant kompiliatoriui suteikia galimybę piešti paprastus ir sudėtingus grafikus dvimatėje (2D) ir trimatėje (3D) erdvėse. Tačiau viskas paprasta yra tik dvimatėje erdvėje. Norint kurti grafiką, animaciją trimatėje erdvėje reikalingi papildomi įskiepai operacinėje sistemoje ir papildomų failų, bibliotekų kompiliatoriui.
2. Borland C++ Builder kompiliatorius turi didelio pasirinkimo komponentų įrankių juostą. Buvo nagrinėti *Chart* ir *Canvas* komponentai.

3. *Chart* komponentas programuotojui padeda negalvoti apie grafiko elementų kūrimą. Šis komponentas pateikia daugybę įvairaus pobūdžio grafikų.
4. Komponentas *Canvas* programuotojui suteikia didesnę modeliavimo laisvę. Programuotojas su šiuo komponentu gali ne tik brėžti grafikus, bet ir išgauti judesio efektą.
5. Labai nepatogu kurti trimačius vaizdus. Kompiliatoriai reikalauja daugybės papildomų komponentų ir įskiepų.
6. Brėžiniuose svarbu pasirinkti tinkamą piešiamų objektų storumą (bold) ir tinkamas koordinačių sritis, kitaip matysime netikslų vaizdą.

7. MATEMATINIŲ SISTEMŲ IR C++ PLATFORMOS ANIMACIJŲ SAVYBIŲ PALYGINIMAS

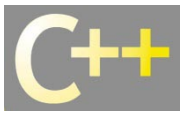
Žinomiausios pasaulyje kompiuterinės matematinės sistemos MATLAB, MATHCAD ir MAPLE yra praktiški ir universalūs paketai spręsti sudėtingus matematinius uždavinius bei kurti aukštos kokybės grafikus. Šios matematinės sistemos turi įvairiapusišką animaciją ir galimybę per keletą sekundžių suvokti apibrėžtos funkcijos kitimą ir kaip dinamiškai vaizduojamas paviršius ar figūra elgiasi. Atlikę tyrimus animacijos kūrimo MATLAB, MATHCAD ir MAPLE kompiuterinėse matematinėse sistemose, išanalizavę galimybes ir patyrinėję grafikų kūrimą C++ platformoje susidarėme aiškų vaizdą apie šių sistemų pranašumus ir trūkumus. Išanalizavome šias sistemų savybes:

- Animacija plokštumoje ir erdvėje;
- Galimybė animaciją taikyti kelioms funkcijoms vienu metu;
- Animacijai kurti skirti įrankiai;
- Animacijos valdymas;
- Animacijos sudėtingumas;
- Animacijos įrašymas.

7.1. ANIMACIJA PLOKŠTUMOJE IR EDVĖJE

4 lentelė. Animacijos plokštumoje ir erdvėje palyginimas

MATLAB 7 (R2010b)	Visi brėžiniai, grafikai ir kiti vaizdiniai objektai bei animacija yra programuojami komandų ciklais. Aprašoma eilė funkcijų, iškviečiamos

	specialios piešimo priemonės.
MATHCAD 14 (M035) 	Animacijos ir kitų brėžinių kūrimas atliekamas mažiausiai penkiais etapas. Apsirašoma funkcija su jos kintamaisiais. Pasirenkamas grafiko tipas ir sumaketuojamas grafikas. Polinėje koordinačių sistemoje reikia nurodyti šią sistemą. Pasirenkamas animacijos įrašymo įrankis. Nusistatomas kadrų kiekis ir jų atvaizdavimo dažnis per sekundę. Viskas įrašoma.
MAPLE 14 	Programuojamas funkcijos grafikas, kuriam tuo pat metu nustatomi parametrai. Pati animacija atliekama paprastai – vienu žingsniu. Pažymimas grafikas. Tai atlikus atsiranda animacijos valdymo įrankių panelė.
C++ 	Dvimatę animaciją kurti nėra sudėtinga, tam yra skirta net palengvinimo priemonės (priklauso nuo kompiliatoriaus). Trimatei animacijai reikalingi papildomi paketai, klasės, bibliotekos.

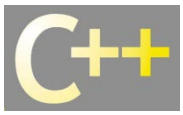
7.2. GALIMYBĖ ANIMACIJĄ TAIKYTI KELIOMS FUNKCIJOMS VIENU METU

5 lentelė. Galimybė animaciją taikyti kelioms funkcijoms vienu metu

MATLAB 7 (R2010b) 	Ši sistema neturi tokios galimybės. Kiekvieną naują animaciją reikia vaizduoti skirtinguose languose (*figure*).
MATHCAD 14 (M035) 	Tokią galimybę kompiuterinė matematinė sistema turi. Visos funkcijos ir kiti elementai į animaciją patenka, jeigu tik yra apibrėžiamos pelės kursoriumi, kai yra parenkamas langas animacijai.
MAPLE 14 	Tokią galimybę kompiuterinė matematinė sistema taip pat turi. Visos funkcijos ir kiti elementai į animaciją patenka, kai yra aprašomi bendrai, pavyzdžiui tai daro priemonė *display()*.
C++ 	Tokia galimybė yra ir ji yra nesudėtinga.

7.3. ANIMACIJAI KURTI SKIRTI ĮRANKIAI

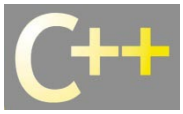
6 lentelė. Animacijai kurti skirtų įrankių palyginimas

MATLAB 7 (R2010b) 	MATLAB matematinė sistema turi 3 pagrindines funkcijas kurti animacijai: *comet()* (taip pat erdvėje *comet3()*), *getframe()* ir *movie()* ir perpiešimo galimybę.
MATHCAD 14 (M035) 	Ši sistema turi vieną įrankį*: Animation* (kartu su funkcija FRAME). Aprašomi funkcijos kintamieji ir į funkciją ar į kintamųjų aprašymą įtraukiame parametrą FRAME. Animacija prasideda nuo pradinio kadro – FRAME=0. Tada pasirenkami animacijos parametrai specialiame lange. Pasirenkamas plotas, kuris bus įtraukas į animaciją. Tai atlikus galima peržiūrėti animaciją.
MAPLE 14 	MAPLE turi daugiausiai funkcijų skirtų animacijai kurti – net 6. Vienu iš būdu animacijai kurti yra naudojama funkcija *animate()* arba *animate3d()*. Pvz.: *animate*(sin($x*t$), $x=-4..4$, $t=0..4$); *animate3d*(cos($t*x$) $*$ sin($t*y$), $x=-Pi..Pi$,$y=-Pi..Pi$, $t=1..2$). Taip pat yra galimybė kurti animaciją iš eilės tvarka sudėtų grafikų rodymo pasinaudojant priemone *display()* ir jos parametru i*nsequence()*.
C++ 	Daugybė priemonių. Priemonės pasirinkimas ir panaudojimas priklauso nuo animacijos tipo ir tai ką norime, kad animacija atvaizduotų.

7.4.ANIMACIJOS VALDYMAS

7 lentelė. Animacijos valdymo palyginimas

MATLAB 7 (R2010b) 	Nėra jokių animacijos valdymo įrankių.
MATHCAD 14 (M035) 	MATHCAD turi specialiai animacijos valdymui skirtus įrankius.
MAPLE 14	MAPLE taip pat turi labai patrauklų sąrašą priemonių valdyti animaciją.

	
C++ 	Valdymo priemonių kiekis priklauso nuo vartotojo suprogramuoto įrankių kiekio.

7.5.ANIMACIJOS SUDĖTINGUMAS

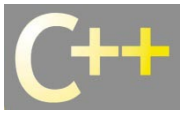
8 lentelė. Animacijos sudėtingumo palyginimas

MATLAB 7 (R2010b) 	Norint sukurti animaciją šioje matematinėje sistemoje privaloma turėti programavimo pagrindų ir patirties šioje sistemoje. Sistema nėra „draugiška“ paprastam vartotojui.
MATHCAD 14 (M035) 	Reikalingas supratimas apie šį matematinį paketą. Mokėjimas užrašyti reikalingas funkcijas, parinkti tinkamus parametrus – palengvina animacijos kūrimą.
MAPLE 14 	MAPLE naujojoje (14) versijoje reikalingas didesnis įgūdžių bagažas negu senesnėse, tačiau tai vis tiek yra patogi sistema.
C++ 	Reikalingi dideli programavimo įgūdžiai. Animacijos pagrindų išmanymas. Mokėjimas įterpti reikalingus paketus. Susieti kompiliatorių su reikalingais kitais paketais.

7.6. ANIMACIJOS ĮRAŠYMAS

9 lentelė. Animacijos įrašymo galimybių palyginimas

MATLAB 7 (R2010b) 	MATLAB suteikia galimybę animaciją įrašyti į .mpeg tipo failą. Yra galimybė failą peržiūrėti bet kokiame video grotuve kompiuteryje. Po failo konvertavimo yra galimybė talpinti failą internetiniame puslapyje.
MATHCAD 14 (M035) 	MATHCAD suteikia galimybę animaciją įrašyti į .avi tipo failą. Yra galimybė failą peržiūrėti bet kokiame video grotuve kompiuteryje. Po failo konvertavimo yra galimybė talpinti failą internetiniame puslapyje.

MAPLE 14 	MAPLE suteikia galimybę animaciją įrašyti į .gif tipo failą. Yra galimybė failą peržiūrėti kompiuteryje. Be jokio konvertavimo yra galimybė talpinti failą internetiniame puslapyje.
C++ 	Animacijos įrašymo failo tipą pasirenka programuotojas pats. Nuo pasirinkto failo tipo priklauso programavimo eiga ir panaudoti vaizdo apdorojimo paketai.

8. IŠVADOS

Atlikus tyrimą ir padarius analizę apie trijų žinomiausių kompiuterinių matematinių sistemų MAPLE, MATHCAD ir MATLAB grafikos ir animacijos kūrimo bei valdymo galimybes padarytos tokios išvadas:

1. Absoliučiai visose tyrinėtose matematinėse sistemose yra galimybė braižyti paprastus ir sudėtingus grafikus tiek dvimatėje , tiek trimatėje erdvėje, pakeisti brėžinių dydžius, žiūrėjimo kampą, kurti animaciją.
2. Minėtas sistemas galima pritaikyti funkcijoms vaizduojamoms Dekarto koordinačių sistemoje, polinėms kreivėms ir funkcijoms, kurios yra išreikštos parametrinėmis lygtimis.
3. Vienu metu įmanoma vaizduoti kelių funkcijų eigą MAPLE ir MATHCAD sistemose. Matematiniame pakete MATLAB tokios funkcijos įvykdyti negalima.
4. Matematinis paketas MAPLE turi ryškiausią priemonių rinkinį animacijai kurti ir plėtoti. Visos priemonės yra lengvai suvokiamos ir lengvai pritaikomos praktikoje. Sistemoje MATHCAD animacijos kūrimas yra paprastesnis, čia ji kuriama pasitelkiant tik vieną funkciją. Bet animacijos kūrimas reikalauja penkių žingsnių. MATLAB pakete animacijai kurti reikia programinio ciklo, tai reikalauja įgūdžių ir programavimo pagrindų.
5. Matematiniai paketai MAPLE ir MATHCAD turi animacijos kontroliavimo priemones, kurių patogus valdymas leidžia reguliuoti animacijos spartą, kryptį, bet kada nutraukti ir vėl pradėti animacijos vykdymą. MAPLE sistema turi galimybę animaciją vykdyti ciklišk ai, kai animacija veikia be perstojimo iki kol nėra sustabdoma rankiniu būdu. MATHCAD sistema tokios galimybės nesuteikia. MATLAB visiškai neturi animacijos valdymo įrankių paketo. Šiame matematiniame pakete animacijos veikimo eigoje jos negalima sustabdyti. Taip pat nėra galimybės

rodyti po vieną kadrą ir kartoti ciklišKai visus kadrus. Kad visą tai galima būti išgauti, reikia tai numatyti rašant animacijos kūrimo programą.

6. Be išimties visi matematiniai paketai turi galimybę animacinius grafikus įrašyti. Paskui tuos įrašus galima stebėti nenaudojant kompiuterinių matematinių sistemų. Sukurtus animacijų failus galima naudoti interneto svetainėse ir kitose elektroniniuose dokumentuose.
7. Paprasčiausia, lengvai suprantama ir patogiausia valdyti animacinė sistema yra MAPLE. Norint dirbti tiek su MATHCAD, tiek su MATLAB sistema reikia turėti daugiau įgūdžių ir programavimo patirties, kad išgautume panašius rezultatus kaip su MAPLE sistema. Siūloma kuriant matematines sistemas remtis šios sistemos savybėmis, algoritmais bei patrauklumu vartotojui.
8. Nesudėtingiems skaičiavimams ir jų grafiniam pavaizdavimui rekomenduojama senoji MAPLE versija (visos versijos iki MAPLE 9).
9. Nerekomenduojama kurti animacijos C++ platformoje, jei vartotojo poreikius pilnai patenkina MAPLE, MATHCAD, MATLAB matematinės sistemos ar dar kitos mokomos ir nemokomos tokio pobūdžio sistemos.

9. LITERATŪRA

1. Gekeler, Eckart. *Mathematical methods for mechanics :a handbook with MATLAB experiments.* Berlin *:* Springer, 2008.

2. Adomavičius J., Dulinskienė T. *Informatika 2, užduočių paruošimas sprendimui MATLAB ir MATHCAD aplinkose. Mokomoji knyga.* Kaunas: Technologija, 2009.

3. Hörhager M., Partoll H. *MATHCAD, Einführung – Anwendung – Referenz.* Oxford: Blackwell, 1998.

4. Lapinskas A. *Matematikos praktikumas su MATHCAD*. Kaunas: Akademija, 2006.

5. Lipeikienė J. *Matematika su kompiuteriu. Kompiuterinės matematinės sistemos DERIVE, MAPLE, MATLAB*. Vilnius: Matematikos ir informatikos institutas, 2002.

6. Lupeikis Z., Šinkūnas J., Urbonas A. *Matematinė analizė I.* Vilnius: VPU leidykla, 1998.

7. Turskienė S. *Darbo su sistema MAPLE V pagrindai*. Šiauliai: ŠU leidykla, 2001.

8. Turskienė S. *Informacijos mokslai*. Šiauliai: ŠU leidykla, 2005.

9. Eberhardt B., Etzmuß O., Hauth M. *Implicit-Explicit Schemes for Fast Animation with Particle System.* Tubingen, Germany: Wilhelm-Schickard-Institut, Universit at Tubingen, 2008.

10. Kleiza J. *Matematinis paketas MAPLE*. Vilnius: leidykla " Technika " , 2003

11. Blonskis J., Bukšnaitis V., Končienė J., Rubliauskas D. *C++ Praktikumas.* Kaunas: KTU, 2001.

12. Blonskis J., Bukšnaitis V., Misevičius A., Rubliauskas D. *C++ Builder grafika.* Kaunas: leidykla „Smaltijos", 2004.

ŠALTINIAI INTERNETE

1. http://www.MAPLEsoft.com

2. http://www.MATHCAD.com

3. http://support.mathsoft.com

4. http://www.mathworks.com

5. http://dgs.dtiltas.lt/MATHCAD

6. http://193.219.157.231/Vilkas/Matematiniai_paketai/MAPLE.htm

7. http://193.219.157.231/Vilkas/Matematiniai_paketai/Mathematica.htm

8. http://193.219.157.231/Vilkas/Matematiniai_paketai/MATLAB.htm

9. http://193.219.157.231/Vilkas/Matematiniai_paketai/MATHCAD.htm

10. http://oras.if.ktu.lt/moduliai/t210m009/Informatikos fak/Vadoveliai/KD_2.pdf
11.http://oras.if.ktu.lt/moduliai/t210m009/Informatikos fak/Vadoveliai/DLDI_COMSOL-1.pdf