КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ и МЕХАНИКИ

им. Н.И. ЛОБАЧЕВСКОГО

Ю.Г. ИГНАТЬЕВ

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ФУНДАМЕНТАЛЬНЫХ ОБЪЕКТОВ И ЯВЛЕНИЙ

В СИСТЕМЕ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE

Казанский университет
2013

УДК
530.12+531.51+517.944+519.713+514.774
ББК 22.632
В87

Печатается по рекомендации Учебно-методической комиссии
Института математики и механики им. Н.И. Лобачевского

Игнатьев Ю.Г. Математическое моделирование фундаментальных объектов и явлений в системе компьютерной математики Maple. Лекции для школы по математическому моделированию. – Казань: Казанский университет, 2013, - 298 с. - ISBN 978-5-94990010-9.

Рецензенты: В.П. Дьяконов, доктор техн. наук, профессор;
М.Н. Кирсанов, доктор физ.-мат. наук, профессор.

Книга подготовлена в виде лекций по математическому и компьютерному моделированию. В ней обобщаются результаты работ автора и его учеников в области математического и компьютерного моделирования в системе компьютерной математики Maple применительно к проблемам геометрии, механики, теории поля и задачам высшего физико-математического образования. Книга будет полезна студентам старших курсов, магистрантам и аспирантам, специализирующимся в областях теоретической механики, геометрии, теоретической физики. Особое внимание уделяется визуализации математических моделей, в том числе, и динамической визуализации. Ил. 128. Библиогр. 159 назв.

Yurii G. Ignatyev. Mathematical modelling of fundamental objects and the phenomena in system of computer mathematics Maple. Lectures for school on mathematical modelling. The book is prepared in the form of lectures on mathematical and computer modelling. In it results of works of the author and its pupils in the field of mathematical and computer modelling in system of computer mathematics Maple with reference to problems of geometry, mechanics, the theory of a field and also problems of the higher physical and mathematical education are generalised. The book will be useful to students of older years, undergraduates and to the post-graduate students specialising in areas of the theoretical mechanics, geometry, the theoretical physics, the mathematical modelling. The special attention is given to visualisation of mathematical models, including, and dynamic visualisation.

Оглавление

Введение

Согласно одному из основоположников математического моделирования, академику А.А. Самарскому, (см., например, [1])

«... математическая модель - это эквивалент объекта, отражающий в математической форме важнейшие его свойства - законы, которым он подчиняется, связи, присущие составляющим его частям, и т. д.» [1], причем «... сама постановка задачи о математическом моделировании какого - либо объекта порождает четкий план действий. Его можно условно разбить на три этапа: модель -алгоритм-программа (см. Рис. Рис..1).

Рис..1 *Триада математического моделирования А.А. Самарского (из книги [1]).*

На первом этапе выбирается (или строится) “эквивалент” объекта, отражающий в математической форме важнейшие его свойства - законы, которым он подчиняется, связи присущие составляющим его частям и т.д.. Математическая модель (или ее фрагменты) исследуется теоретическими методами, что позволяет получить важные предварительные знания об объекте.

Второй этап - выбор (или разработка) алгоритма для реализации модели на компьютере. Модель представляется в форме, удобной для применения численных методов, определяется последовательность вычислительных и логических операций, которые нужно.произвести, чтобы найти искомые величины с заданной точностью. Вычислительные алгоритмы должны не искажать основные свойства модели и, следовательно, исходного объекта, быть экономичными и адаптирующимися к особенностям решаемых задач и используемых компьютеров.

На третьем этапе создаются программы, “переводящие” модель и алгоритм на доступный компьютеру язык. К ним также предъявляются требования экономичности и адаптивности. Их можно назвать “электронным” эквивалентом изучаемого объекта, уже пригодным для непосредственного испытания на “экспериментальной установке” - компьютере. Создав *триаду* “модель - алгоритм - программа”, исследователь получает в руки универсальный, гибкий и недорогой инструмент, который вначале отлаживается, тестируется в “пробных” вычислительных экспериментах. После того как адекватность (достаточное соответствие) триады исходному объекту удостоверена, с моделью проводятся разнообразные и подробные “опыты”, дающие все требуемые качественные и количественные свойства и характеристики объекта. Процесс моделирования сопровождается улучшением и уточнением, по мере необходимости, всех звеньев триады...».

Таким образом, академик А.А. Самарский дал четкое, ставшее классическим, определение объекта математического моделирования и основных задач математического моделирования.

Уникальные графические возможности системы компьютерной математики (СКМ) Maple, в частности, возможности создания трехмерных анимационных моделей, хорошо проработанные программные процедуры численного интегрирования систем обыкновенных дифференциальных уравнений (ОДУ), сплайновой и B - сплайновой интерполяции функций позволяют рассматривать СКМ Maple в качестве мощного современного инструмента математического моделирования объектов механики и родственных им [9] [7]. В настоящее время методы математического моделирования в СКМ нача-

ли эффективно применяться в исследованиях математических моделей как фундаментальных физических явлений, так и прикладных задач. В частности, монографии Д.П. Голоскокова [4, 5] целиком посвящены проблемам математического моделирования в СКМ Maple объектов математической физики – физических полей, гидродинамических процессов, процессов теплопереноса и диффузии; в фундаментальной монографии В.П. Дьяконова [6] обширная глава посвящена применению СКМ Maple в математическом моделировании, в частности, моделировании электронных схем и измерительных систем на основе эффекта Допплера; монографии М.Н. Кирсанова [7] содержит материалы по применению графов в математическом моделировании и математическому моделированию сложных механических систем со связями и визуализации математических моделей этих систем [8]. В работах Ю.Г. Игнатьева с соавторами [19] - [27] методы математического моделирования с помощью СКМ Maple успешно применяются для решения весьма сложных задач релятивистской кинетики, теории гравитации и космологии ранней Вселенной. В последнее время СКМ Maple, особенно ее приложение Maplet, стала применяться для компьютерного моделирования процесса обучения, в частности, для создания системы аналитического тестирования знаний [28, 29, 30]. Важным преимуществом СКМ Maple является и возможность интеграции этой системы с СКМ MatLab, которая приспособлена к моделированию электронных систем и технологических процессов.

Глава I

Система компьютерной математики Maple

I.1 Системы компьютерной математики и их возможности для математического моделирования

Первые пакеты программ аналитических вычислений появились в 60-х – 70-х годах XX - го столетия. Первоначально такие пакеты выполняли узкопрофессиональные задачи и были предназначены для реализации на больших ЭВМ. В бывшем СССР большой вклад в развитие систем символьной математики внесла школа академика Глушкова В.М. В конце 70-х годов были созданы малые инженерные ЭВМ класса «Мир», способные выполнять аналитические вычисления даже на аппаратном уровне. Был разработан и успешно применялся язык символьных вычислений «Аналитик» [38]. Эти работы отчасти предвосхитили развитие систем символьной математики.

Но первые системы символьной математики, пригодные для работы на ЭВМ и рассчитанные на широкого пользователя, появились в 80-х годах. За прошедшие 20-25 лет эти системы были значительно развиты, среди них появились признанные лидеры, развилась профессиональная терминология. В конце 80-х годов такие системы назывались *системами компьютерной алгебры* (см., например, [39], [40],[41]). В настоящее время такие системы принято называть *системами компьютерной математики* (СКМ) (см., например, [42], [43]). В настоящее время можно выделить несколько основных лидеров СКМ (см. таблицу на 2.) – это `Derive, MathCAD, Maple, Mathematica, MatLAB`.

Наиболее популярной программой считается `MathCAD`, но она приспособлена, скорее всего, для инженерных расчетов, а также для задач образования. Среди, действительно, мощных систем, предназначенных для профес-

сиональных, научных целей можно выделить 3: `Maple`, `Mathematica`, `MatLAB`. Система `MatLAB` имеет мощные встроенные методы численного интегрирования для моделирования конкретных систем, но ее недостатком являются ограниченные возможности символьных вычислений и высокая стоимость пакета (порядка 80 тысяч рублей). Системы `Mathematica` и `Maple`, в принципе сопоставимы по своим возможностям, однако, пакет `Mathematica` до сих пор имеет значительно меньшее по сравнению с системой `Maple` встроенных функций. Примером может явиться весьма необходимая нам функция построения сплайнов, которая в системе `Maple` существует с начала 90-х годов, а в системе `Mathematica` она появилась только в последней версии. Динамическая графика, особенно трехмерная, столь необходимая для целей компьютерного моделирования, гораздо раньше появилась в системе `Maple` и лучше в нем прописана.

В отечественной научной и учебной литературе эти основные математические пакеты достаточно хорошо описаны. В частности, наиболее подробно описан пакет `MatCAD` в работах В.П. Дьяконова [43] – [48], Л.В. Ефремова [49, 50], В.А. Охорзина [51]-[53], Р. Ивановского [54], В.В. Фриска [55], Д. Гурского [56], Д.В. Кирьянова [57], [58], Д.С. Поршнева и И.С. Беленковой [59] и др.. Система `Mathematica` описана в трудах В.П. Дьяконова [60, 61], [64] - [66], Т.В. Капустиной [62], А.М. Половко [67], Я.К. Шмидского [68] и др. Система `MathLAB` описана в трудах В.П. Дьяконова [69]-[72], система `Derive` – в трудах В.П. Дьяконова [73, 74, 75, 76]. Наконец, из наиболее ранних отечественных книг по СКМ `Maple` можно выделить книги В.Н. Говорухина В.Н. и В.Г. Цибулина [77], В.П. Дьяконова [78], посвященные пакету `Maple V`, затем цитированные выше монографии А.В. Матросова [9], М.Н. Кирсанова [7], В.П. Дьяконова [79] - [82]. М.Н.

Перейдем теперь к работам по математическому моделированию в системах компьютерной математики. СКМ с самого начала обратили внимание исследователей как мощное средство компьютерного моделирования объектов, свойств и, особенно, процессов. В настоящее время трудно представить развитие современной теоретической физики, физики высоких энергий, фундаментальных полей, астрофизики и космологии без систем компьютерной математики. Необходимо отметить, что СКМ как раз и были созданы в лабораториях физики высоких энергий. Как ни странно, но одними из первых монографий, посвященных систематическому применению СКМ в математическом моделировании были монографии Альфреда Грэя [83], [84], посвященные применению пакета `Mathematica` к проблемам дифференциальной геометрии и теории обыкновенных дифференциальных уравнений.

Далее следует отметить монографию В.П. Дьяконова, посвященную моделированию научно - технических задач с помощью пакета `Mathematica 4` [85], монографию Ч.Г. Эдвардса и Д.Э. Пенни [86], посвященную моделированию краевых задач в пакете `Mathematica`, цитированные выше монографию А.В. Матросова [9], посвященную математическому моделированию задач механики, монографии Д.П. Голоскокова [4, 5], посвященные математическому моделированию задач математической физике в пакете `Maple`, и другие [87]-[96]. Следует заметить, что пакет компьютерной математики Maple, начиная уже с самых ранних версий имеет несомненные преимущества в области 3D-графики, особенно, интерактивной и динамической, по сравнению с пакетом «Mathematica», и позволяет программными средствами решить указанные задачи[1]. Эта книга посвящена разработке программных процедур математического моделирования именно в пакете «Maple».

I.2 Пакет Maple и принципы программирования в нем

Обратим внимание на некоторые особенности пакета `Maple`, которые делаю этот пакет оптимальным для наших задач. Во-первых, – это непревзойденные возможности интерактивной графики, в частности, возможности покадрового создания 3-х мерной динамической интерактивной графики. Во-вторых, это весьма удобный для пользователя Maple-язык программирования, имеющий четкую логику, доступную не профессиональному программисту. В-третьих, – это возможность простыми способами создавать пакеты прикладных программ и встраивать их в систему `Maple`. Кратко опишем эти особенности.

I.2.1 Возможности динамической интерактивной 3d-графики пакета Maple

Графические возможности последних версий пакета `Maple` достаточно подробно описаны в фундаментальной монографии В.П. Дьяконова [6]. В этой монографии возможностям и технике визуализации вычислений в пакете `Maple` посвящена обширная глава. Помимо стандартных графических программных процедур ядра пакета `Maple`, ответственных за 2-мерную и 3-мерную графику, `plot()` и `plot3d` и имеющих около 50-ти необязательных опций, которыми можно регулировать изображением, пакет `Maple` имеет несколько специализированных графических библиотек, – `plots`,

[1] О сравнительных характеристиках систем «Mathematica» и «Maple» см., например, [13, 15].

`PlottingGuide`, `plottools` и др., значительно расширяющих графические возможности пакета. Одна только библиотека `plots` содержит 60 графических программных процедур.

Для наших целей особенно важны возможности покадрового строительства динамической графики с помощью процедуры `display` библиотеки `plots`, позволяющей создавать графические последовательности из любых объектов, в том числе, использовать в качестве объектов строковые переменные. Последнее обстоятельство позволяет создавать цифровое оснащение динамической графике.

I.2.2 Принципы программирования в пакете `Maple`

Последние версии СКМ `Maple` содержат около 4000 функций, – в этом обилии функций одно из главных достоинств `Maple`. Однако, программные возможности `Maple` позволяют значительно расширить и этот богатый ассортимент. Для нас будут важны два формата задания функции пользователя. В первом, наиболее близком к языку `TurboPascal` функция пользователя записывается в виде процедуры:

```
>name:=proc(x1,x2,...,xn)
local y1,y2,...,ym:
y1:=f1;y2:=f2:
......
end proc:
```

где `name` – имя процедуры; `x1,x2,...` – параметры процедуры, - внешние переменные; `y1,y2,...` – локальные переменные. Второй, упрощенный, формат функции пользователя имеет вид:

```
>name:=(x1,x2,...,xn)->f(x1,x2,...,xn):
```

Язык программирования `Maple` имеет стандартные управляющие структуры в виде условных выражений и циклов. В частности, условные выражения имеют структуру:[2]

```
if <условие сравнения> then <элементы>:
elif <условие сравнения>
then <элементы>: else <элементы>: end if:
```

Наиболее подробно основные принципы программирования в СКМ `Maple` описаны в цитированных выше книгах А.В. Матросова [9] и В.П. Дьяконова [6].

[2] В более ранних версиях Maple условное выражение оканчивалось знаком `fi` вместо `end if`.

I.3 Создание пользовательских библиотек в пакете `Maple`

Наиболее простым способом создания пользовательской библиотеки программных процедур, при котором она не встраивается в систему `Maple`, но всегда может быть вызвана, является следующая [6].

- Файл, содержащий библиотеку `Name`, следует начать с образования пустой таблицы:

```
>restart:
 Name:=table():
```

- Создаваемые процедуры `Com_i` библиотеки должны иметь следующий формат:

```
>Name[Com_i]:=proc(x,y,..) ... end proc:
```

- Библиотека сохраняется в виде файла с именем `File.m` с помощью процедуры:

```
>save(Name,‘Path/File.m‘):
```

После запуска программы она генерирует `File.m` по адресу `Path`. Указанный файл не открывается в текстовой моде, и поэтому содержимое его не доступно для просмотра. Вызов библиотеки `Name` производится процедурой:

```
>restart:
 read "YuDifEquat.m";
```

Отметим следующий немаловажный замеченный факт: обращение к программной процедуре в библиотеке зачастую выполняется быстрее, чем непосредственное обращение к этой же программной процедуре в файле. Этот эффект, по - видимому, вызван тем обстоятельством, что топология системы `Maple` устроена таким образом, что обращение к внутренним процедурам пакета производится быстрее, чем внешнее обращение.

I.4 Принципы создания оснащенной динамической визуализации математических моделей в системе компьютерной математики «Maple»

Системы компьютерной математики (СКМ) обладают богатыми и до сих пор еще мало оцененными графическими возможностями, позволяющими создавать многопараметрические графические модели как геометрических, так и

физических объектов. Особенно богатыми возможностями трехмерной графики обладает пакет Maple[3], предоставляющий пользователю возможность интерактивного взаимодействия с трехмерной графической средой, что открывает уникальные возможности изучения свойств трехмерных графических моделей. При этом следует отметить, что прямое использование стандартных графических процедур компьютерной математики не позволяет создавать достаточно сложные графические модели, - для создания их необходимо создавать собственные специализированные библиотеки программных графических процедур. Для создания этих процедур в нашей работе применяются расширенные опции процедур ядра Maple и библиотеки plots. Ниже формулируются основные принципы создания программных графических процедур в СКМ.

Системы компьютерной математики (СКМ) Maple, обладают богатыми и до сих пор еще мало оцененными графическими возможностями, позволяющими создавать многопараметрические графические модели как геометрических, так и физических объектов. Особенно богатыми возможностями трехмерной графики обладает пакет Maple, предоставляющий пользователю возможность интерактивного взаимодействия с трехмерной графической средой, что открывает уникальные возможности изучения свойств трехмерных графических моделей. При этом следует отметить, что прямое использование стандартных графических процедур компьютерной математики не позволяет создавать достаточно сложные графические модели, - для создания их необходимо создавать собственные специализированные библиотеки программных графических процедур. Для создания этих процедур в нашей работе применяются расширенные опции процедур ядра Maple и библиотеки plots. Ниже формулируются основные принципы создания программных графических процедур в СКМ.

I.4.1 Методы создания анимационного кадра и управление параметрами анимации

Динамическую графику в системе `Maple` можно создать двумя способами. Первый из них заключается в использовании прямых команд анимации библиотеки `plots`. Их всего 4:

- `animate(f(x,t),x=a..b,t=t0..t1,options)` - анимация плоских кривых, формат использования процедуры практически повторяет формат процедуры двумерной графики `plot`;

[3] см., например, [9, 10, 11, 13].

- `animate3d(f(x,t),x=a..b,y=c..d,t=t0..t1,options)` - анимация поверхностей, формат использования процедуры практически повторяет формат процедуры трехмерной графики `plot3d`;
- `animatecurve(f(x,t),x=a..b,t=a..b,options)` - анимация пространственных кривых, формат использования процедуры практически повторяет формат процедуры трехмерной кривой `spacecurve`;

– в этих процедурах параметр t играет роль времени. Добавляя в эти процедуры необязательный параметр `frames=N`, мы получаем возможность регулировать число кадров анимации. Возможности указанных программных процедур ограничены. Для создания сложных анимационных моделей необходимо применять второй способ анимации на основе программной процедуры библиотеки `plots display` с необязательной опцией `insequence = true`. Применение этой процедуры без указанной опции создает комбинацию графических объектов, указанных в теле команды `display`. Поэтому сложные анимационные структуры можно создавать из отдельных графических объектов, собирая их с помощью процедуры в различные структуры, которые снова можно интегрировать функцией `display`. При этом надо заметить, что отдельными графическими объектами, собираемыми процедурой `display` могут быть и анимационные объекты, создаваемые указанными выше способами. Ниже показан пример строительства такого графического объекта:

```
>rr:=plots[animate]([r*cos(t),r*sin(t),r=0..1],t=0..2*Pi):rr;
 rb:=plots[animate]([r*cos(t),-r*sin(t),r=0..1],t=0..2*Pi,color=BLUE):
 plots[display](rr,rb,insequence=true);
```

Общая идеологию построения анимационных моделей представлена на схеме Рис.V.56 и подробно разрабатывалась в работах [147], [148]. Следует заметить, что пакет компьютерной математики Maple, начиная уже с самых ранних версий имеет несомненные преимущества в области 3D-графики, особенно, интерактивной и динамической, по сравнению с пакетом «Mathematica», и позволяет программными средствами решить указанные задачи[4]. Эта статья посвящена разработке программных процедур в пакете «Maple», позволяющих осуществлять управляемую, оснащенную динамическую визуализацию основных задач дифференциальной геометрии кривых.

Под управляемой, оснащенной динамической визуализацией здесь и далее мы понимаем визуализацию многопараметрической математической модели, изменение свойств которой можно наблюдать во временной последовательности в графической форме, сопровождаемой изменяющейся со временем числовой или графической информацией, с возможностью изменения пользователем параметров модели.

[4] О сравнительных характеристиках систем «Mathematica» и «Maple» см., например, [13, 15].

Методы оснащенной динамической визуализации математических моделей разрабатываются в группе Ю.Г. Игнатьева с 2004 года [16]. Отметим, что профессором М.Н. Кирсановым (Московский энергетический университет) разрабатываются методы динамической визуализации сложных, лагранжевых механических систем [18]. В работах [97, 98, 99, 100] конструируются математические и компьютерные модели *нелинейных обобщенно-механических систем*, описываемых системой нелинейных обыкновенных дифференциальных уравнений произвольного порядка с помощью методов сплайновой интерполяции численных решений дифференциальных уравнений.

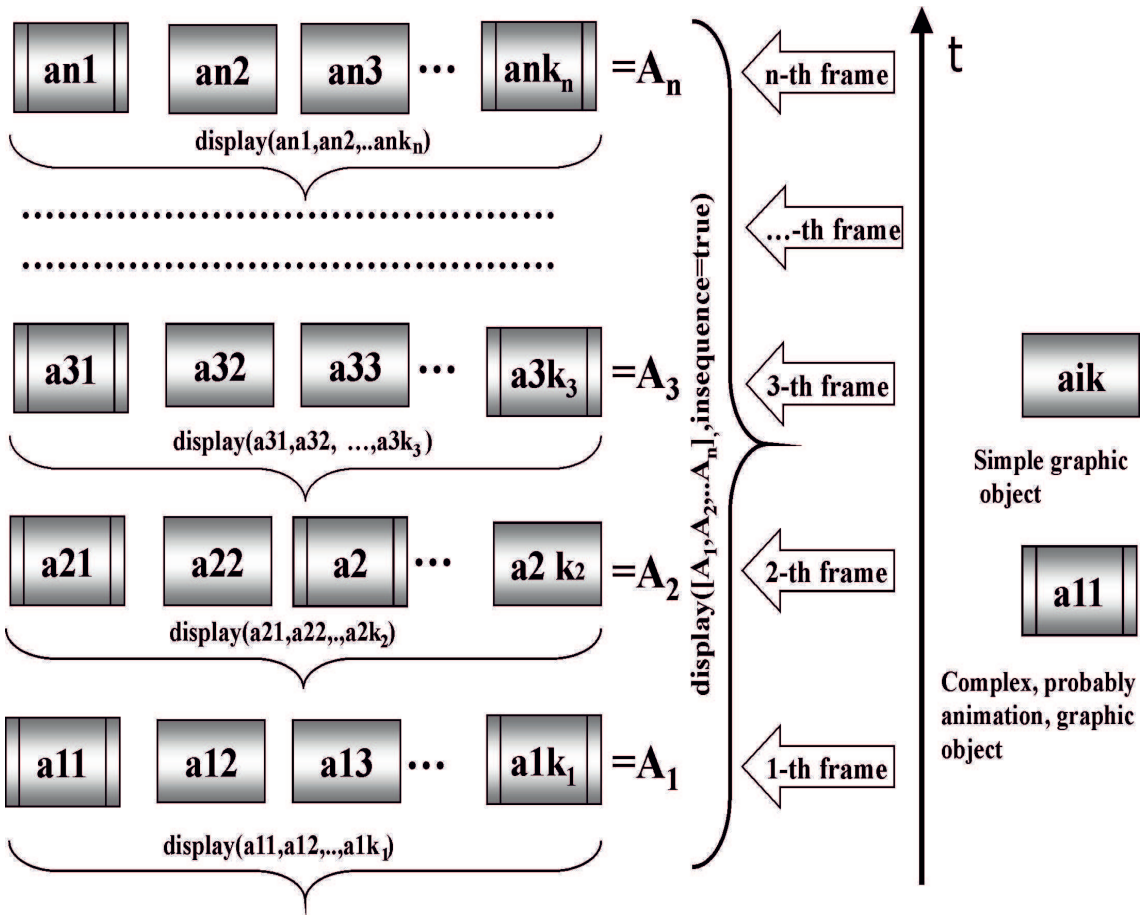


Рис.I.2 *Схема создания динамической графики. Светлые квадраты - простые графические объекты, темные - сложные, включающие в себя несколько простых, и, возможно, анимированные структуры. Цифры в правой части рисунка - номера кадров анимации. Неизменяющиеся слева квадраты - графическая структура фона – эти элементы должны присутствовать в каждом кадре.*

Технология создания оснащенной управляемой динамической модели в СКМ Maple заключается в следующем [97, 101] (см. Рис.V.56). Сначала с помощью стандартных процедур Maple или программирования в пакете создаются графические объекты `a_ik`. Эти объекты могут быть как статическими графическими объектами, так и сложными, анимационными, созданными с помощью встроенных в Maple процедур анимации. Простые графические объекты представлены на Рис.V.56 прямоугольниками с горизонтально-градиентной заливкой, сложные - прямоугольниками с вертикальными секциями и вертикальной градиентной заливкой. Полученные графические объекты объединяются в i-тый кадр анимации, `frame_i`, с помощью встроенной процедуры `plots[display](a_i1,a_i2,...,a_ini)`, с присвоением имени `A[i]`. Наконец, все кадры анимации объединяются в анимационную последовательность с помощью той же процедуры, но с добавочной опцией: `plots[display]([A[1],A[2],.. ,A[n]], insequence=true):`

В результате получается, вообще говоря, трехмерная, интерактивная анимационная картина, состоящая из n кадров. Дискретной временной переменной t при этом является номер кадра i, длительностью анимации, как и степенью ее непрерывности, управляет параметр n - число кадров. При создании анимации необходимо выбирать золотую середину между качеством анимации (большие значения n) и скоростью ее загрузки (малые значения n). Управление скоростью анимации, ее направлением, как и переходом в режим покадрового просмотра, можно осуществить непосредственно из графического меню окна Maple после выделения графического объекта. Для управления же параметрами компьютерной модели эти параметры должны быть введены в программные процедуры графических объектов. Такими параметрами могут быть, например, уравнения линии, способ ее представления, тип ее оснащения, количество кадров анимации, предельные значения параметров кривой и т.п.

Динамическое оснащение кривой, как и других графических объектов, может быть трех типов:

1. Графическое оснащение - оснащение с помощью дополнительных графических объектов: изображением точек, касательных и других векторов, плоскостей и т.п.;

2. Текстовое оснащение - оснащение с помощью динамических текстовых вставок;

3. Цифровое оснащение - оснащение с помощью динамических цифровых вставок.

Рис.I.3. *Схема создания динамической графики. Светлые квадраты - простые графические объекты, темные - сложные, включающие в себя несколько простых, и, возможно, анимированные структуры. Цифры в правой части рисунка - номера кадров анимации. Неизменяющиеся слева квадраты - графическая структура фона – эти элементы должны присутствовать в каждом кадре.*

При этом первые два типа динамического оснащения достигаются простым добавлением графических или текстовых объектов в каждый кадр с введением их функциональной зависимости от номера кадра, i, с помощью процедуры `display`. Цифровые динамические вставки величины s достигаются добавлением в каждый кадр процедуры вычисления этой величины, S(i), которую предварительно необходимо конвертировать в строковую переменную: `convert(s=S(i),string)`. При этом данная величина будет отображаться в форме s=SD(i), где SD(i) - динамически обновляющееся цифровое значение. Для динамического вывода упорядоченного списка величин [s=SD(i),q=QD(i),p=PD(i)] можно использовать формат convert([s=S(i),q=Q(i),p=P(i)],string). При этом необходимо помнить о том, что в приближенных вычислениях с помощью функции evalf(S) Maple по умолчанию выводит 10 значащих цифр. Поэтому до применения конвертирования в строковую переменную необходимо определить формат вывода числовых значений.

Общая идеологию построения анимационных моделей представлена на схемах Рис.V.56, Рис.I.3 и подробно разрабатывалась в работах [97], [101], а затем была применена при создании оснащенных динамических моделей нелинейных обобщенно-динамических систем [98, 99, 100].

I.5 Управление цветом, динамическая раскраска

I.5.1 Векторная модель цветового круга

Для управления цветом в стандартных графических командах Maple типа `plot`, `plot3d` и др. существует необязательная опция `color=parametr`. В качестве параметра может быть использовано название одного из стандартных цветов: `white,yellow,orange,red` и т.д. Однако для более тонкого управления цветом можно использовать цветовой параметр в формате: `color=COLOR(RGB,r,g,b)`, где `r,g,b` – десятичные дроби на интервале `[0,1]`. Для понимания процесса управления цветом в `RGB` - схеме необходимо иметь представление о *цветовом круге* Рис.I.4.

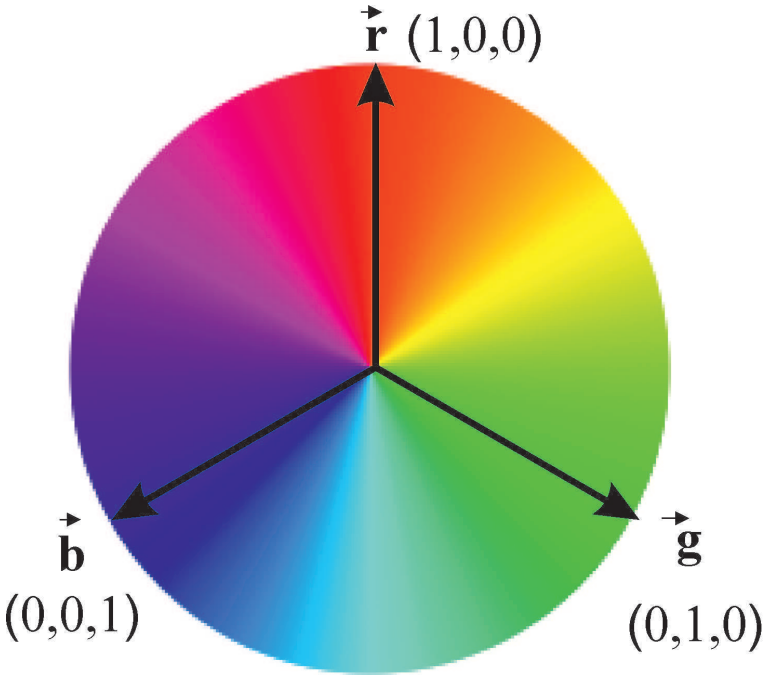


Рис.I.4 *Модель цветового круга в цветовой схеме RGB.*

В векторной модели `RGB` цветового круга каждый цвет представляется двумерным вектором, причем его интенсивность определяется его длиной. На единичном цветовом круге заданы 3 базисных орта, $\vec{r}, \vec{g}, \vec{b}$, соответствующие 3-м чистым основным цветам: красному (`R`), зеленому (`G`) и голубому (`B`). Сумма этих базисных векторов равна $\vec{0}$:

$$\vec{r} + \vec{g} + \vec{b} = \vec{0}. \tag{I.1}$$

0-вектор соответствует цветонейтральной комбинации, т.е., соответствующий 0-вектору цвет может изменяться от белого до черного, через различные градации серого. Так, например, вектор (1,1,1) соответствует белому цвету, (0,0,0) – черному, (0.5,0.5,0.5) – серому цвету.

Таким образом, каждый цвет определяется формулой:

$$\vec{r} = x\vec{r} + y\vec{g} + z\vec{b}, \tag{I.2}$$

где

$$x, y, z \in [0, 1]. \tag{I.3}$$

При этом интенсивность цвета определяется величиной:

$$I = \max\{x, y, z\}, \quad I \in [0, 1] \tag{I.4}$$

– и также изменяется на интервале от 0 до 1 (100%).

I.5.2 Пример: оснащенная динамическая модель цветового круга

Создадим динамическую модель цветового круга. Для этого разделим единичную окружность на N сегментов, в каждом из которых, i-том, зададим функции цветов `rr,gg,bb` с помощью моделирования их кусочно заданными функциями:

```
treug0:=proc(i,N) local f,M:
f:=(i)->2*Pi/N*i:
M:=(i)->[sin(f(i-1)),cos(f(i-1))]:
plots[polygonplot]([[0,0],M(i),M(i+1)]):
end proc:
treug0(7,24):
treug:=proc(i,N) local f,M,rr,gg,bb,xx:
f:=(i)->2*Pi/N*i:
M:=(i)->[sin(f(i-1)),cos(f(i-1))]:
rr:=(xx)->evalf(piecewise(xx>=0 and
xx<2*Pi/3,1-3*xx/(2*Pi),xx>=2*Pi/3
and xx<4*Pi/3,0,xx>=4*Pi/3 and xx<2*Pi,-2+3/(2*Pi)*xx)):
gg:=(xx)->evalf(piecewise(xx>=0 and
xx<2*Pi/3,3*xx/(2*Pi),xx>=2*Pi/3
and xx<4*Pi/3,2-3*xx/(2*Pi),
xx>=4*Pi/3 and xx<2*Pi,0)):
bb:=(xx)->evalf(piecewise(xx>=0 and
```

```
xx<2*Pi/3,0,xx>=2*Pi/3
and xx<4*Pi/3,-1+3*xx/(2*Pi),
xx>=4*Pi/3 and xx<2*Pi,3-3/(2*Pi)*xx)):
plots[polygonplot]([[0,0],M(i),M
(i+1)],color=COLOR(RGB,rr(f(i)),gg(f(i)),bb(f(i)))):
end proc:
```

Исполнение команды

```
>treug(8,24);
```

представлено на Рис.I.5.

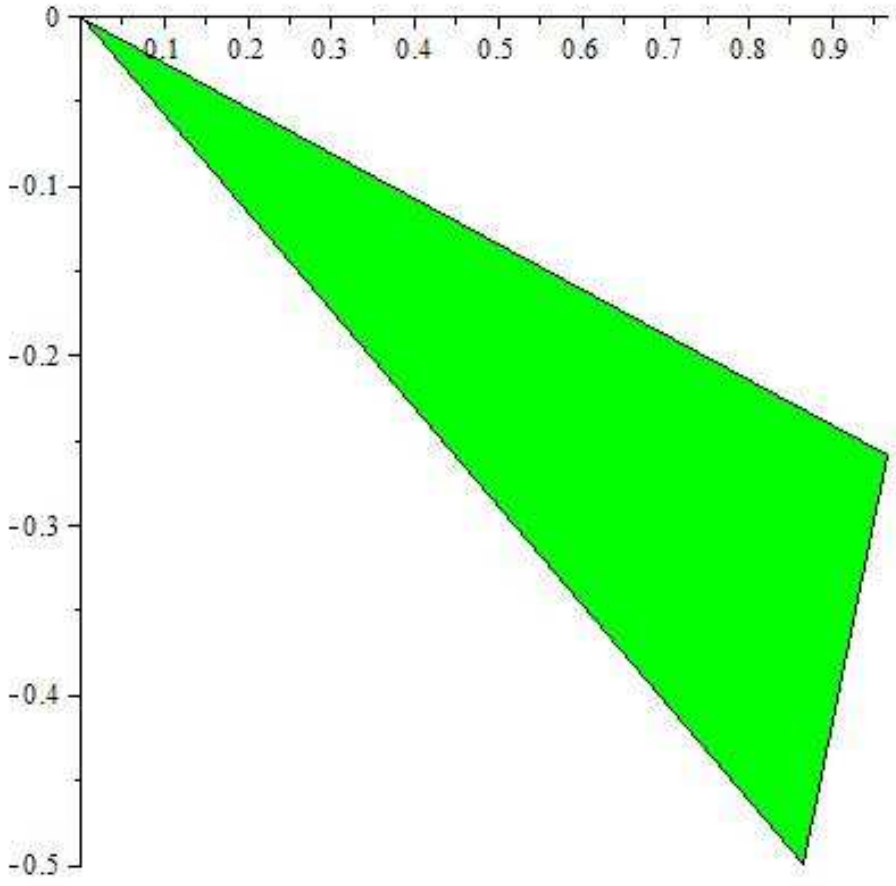


Рис.I.5 *8-й сектор из 24-цветного цветового «круга».*

С помощью созданной процедуры построим статическую графическую модель N-цветного круга:

```
>color_circle:=(N)->
```

```
plots[display](seq(treug(i,N),
i=0..N),axes=NONE):

>color_circle(24);
```

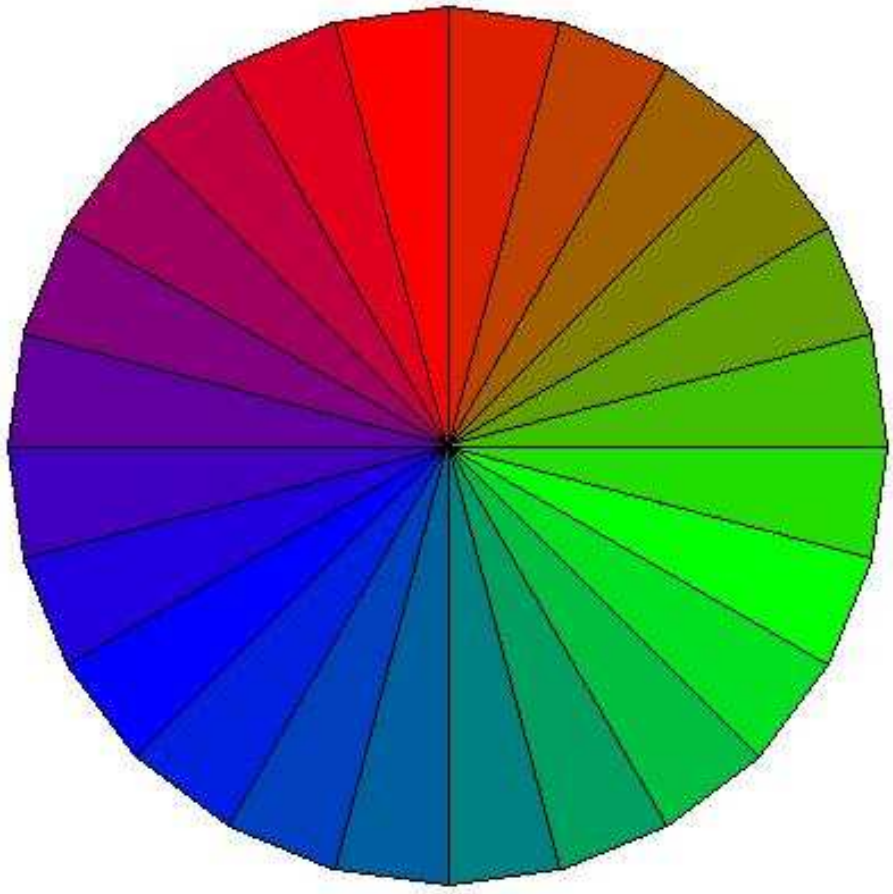


Рис.I.6 *24-цветный цветовой «круг».*

и затем, опять используя команду `display`, создадим процедуру построения k-секторов цветового круга. При этом мы конвертируем номер сектора k В строковую переменную и вставим его в заголовок рисунка:

```
color_part:=(k,N)->plots[display](seq(treug(i,N),
i=0..k),axes=NONE,scaling=CONSTRAINED,title=[i=convert(k,string)]):
```

Исполнение этой процедуры показано на Рис.I.7

```
>color_part(15,24);
```

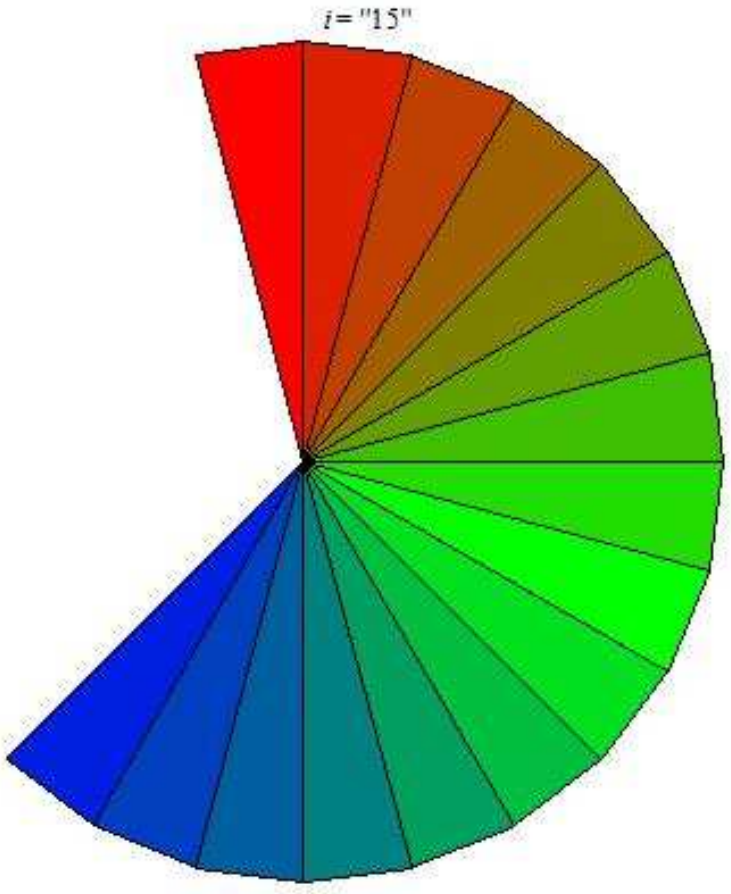


Рис.I.7 *15-ть сегментов 24-цветного цветового «круга».*

Наконец, объединяя еще раз полученные графические структуры с помощью команды `display` с необязательным параметром `insequence=true`, получим нужную динамическую модель, оснащенную динамической информацией о номере сегмента Рис.I.8.

```
Anim_color:=(N)->
plots[display](seq(color_part(k,N),k=0..N),insequence=true):

>Anim_color(36);
```

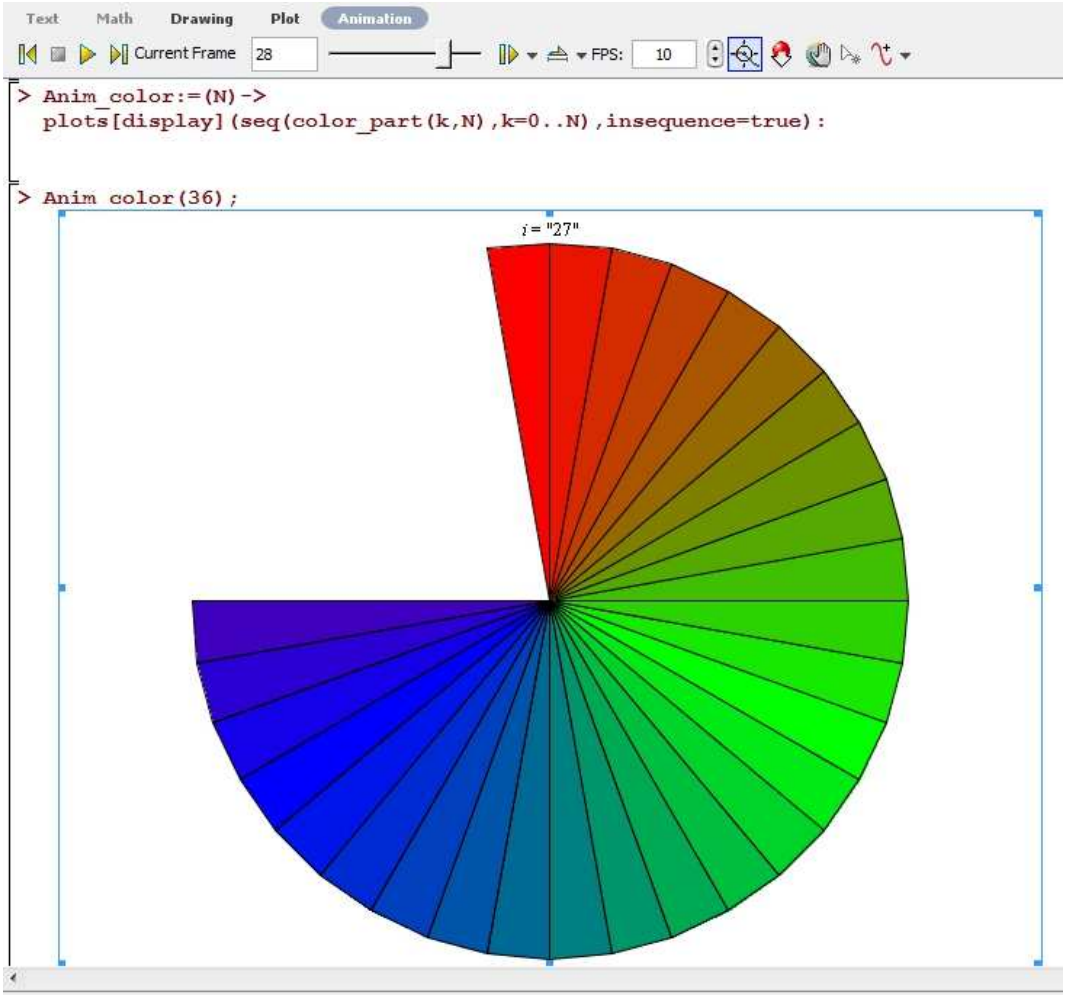


Рис.I.8 *Вид окна Maple 17 с анимационной моделью цветового «круга». Показан 28-й кадр из 36-кадрового фильма.*

I.6 Информационные технологии изучения физико - математических курсов на основе математического моделирования в системе компьютерной математики

I.6.1 Необходимость внедрения информационных технологий в структуру физико - математического образования

Существует ряд весьма веских причин необходимости внедрения информационных технологий в структуру физико - математического образования. Эти причины, в основном, имеют внешний по отношению к физико-математическому образованию характер и вызваны глобальными изменени-

ями в структуре общества, общественного сознания и интенсивным процессом информатизации общества. Среди этих причин:

1. непрерывно и быстро растущие потоки информации и быстрое ее устаревание;

2. сокращение учебных часов на изучение фундаментальных дисциплин с одновременным расширением списка изучаемых вопросов;

3. перенос центра тяжести учебного процесса на самостоятельную работу студентов и учащихся;

4. недостаточное финансирование фундаментальных направлений науки и соответствующих им направлений высшего образования;

5. интеграция различных областей знаний и появление новых направлений науки и технологий;

6. увеличение числа специальностей при одновременном уменьшении числа студентов;

7. деморализация общества, притяжение молодежи к масс-медиа, наркотикам, интернет-зависимости, выхолащивание системы ценностей.

Но помимо этих причин, на наш взгляд, существуют и внутренние причины именно российского математического образования, приводящие в последнее время к его застою и низкой эффективности. Среди этих причин:

1. формализованный характер математического образования;

2. утрата связей математического образования с современными задачами, как прикладных, так и фундаментальных наук;

3. перегруженность математических курсов абстрактным теоретическим материалом в ущерб решению конкретных задач, исторически являющихся целевыми для данных курсов;

4. оторванность математических курсов от современных компьютерных технологий.

Аналогичные проблемы свойственны и многим современным российским научным математическим школам. Известны, например, потребности многих

областей, как фундаментальных, так и прикладных наук, в создании методов исследования нелинейных континуальных систем, описываемых нелинейными дифференциальными и интегро - дифференциальными уравнениями с частными производными. Однако, подавляющее большинство кандидатских и докторских диссертаций по этой специальности посвящено методам решение линейных дифференциальных и интегральных уравнений, причем зачастую исследования завершаются доказательством существования и единственности решения. На наш взгляд, преодолеть указанные противоречия между запросами современных науки и технологий, с одной стороны, и потенциалом российского математического образования, с другой стороны, возможно на пути интенсивного применения методов математического и компьютерного моделирования при изучении всех базовых курсов математики с последующим интегрированием целевых задач этих курсов с задачами фундаментальных и прикладных наук. При этом компьютерное моделирование следует осуществлять в среде систем компьютерной математики (СКМ), а соответствующие курсы формировать, как исследовательские, направленные на построение математических и компьютерных моделей, в ходе создания которых студенты будут овладевать необходимыми фундаментальными знаниями предметов и учиться их практическому применению. Следует обратить внимание на тот факт, что построение математической модели и её компьютерная реализация воспитывают строгость математического мышления, его культуру и технологичность. Построение и исследование компьютерной модели, кроме всего прочего, воспитывает трудолюбие, аккуратность и добросовестность – качества, которых так не хватает постсоветским генерациям молодежи. Кроме всего прочего, этот путь является наиболее эффективным способом вовлечения молодежи в современную науку и инженерию.

I.6.2 Основная идея внедрения информационных технологий в структуру физико - математического образования на основе систем компьютерной математики

Согласно одному из основоположников математического моделирования, академику А.А. Самарскому, (см., например, [1]) «... математическая модель – это эквивалент объекта, отражающий в математической форме важнейшие его свойства – законы, которым он подчиняется, связи, присущие составляющим его частям, и т.д., причем ... сама постановка задачи о математическом моделировании какого-либо объекта порождает четкий план действий. Его

можно условно разбить на три этапа: модель > алгоритм > программа[5]. На первом этапе выбирается (или строится) «эквивалент» объекта, отражающий в математической форме важнейшие его свойства – законы, которым он подчиняется, связи присущие составляющим его частям и т.д.. Математическая модель (или ее фрагменты) исследуется теоретическими методами, что позволяет получить важные предварительные знания об объекте. Второй этап - выбор (или разработка) алгоритма для реализации модели на компьютере. Модель представляется в форме, удобной для применения численных методов, определяется последовательность вычислительных и логических операций, которые нужно произвести, чтобы найти искомые величины с заданной точностью. Вычислительные алгоритмы должны не искажать основные свойства модели и, следовательно, исходного объекта, быть экономичными и адаптирующимися к особенностям решаемых задач и используемых компьютеров. На третьем этапе создаются программы, «переводящие» модель и алгоритм на доступный компьютеру язык. К ним также предъявляются требования экономичности и адаптивности. . . Создав триаду «модель > алгоритм > программа», исследователь получает в руки универсальный, гибкий и недорогой инструмент,который вначале отлаживается, тестируется в «пробных» вычислительных экспериментах. После того, как адекватность (достаточное соответствие)триады исходному объекту удостоверена, с моделью проводятся разнообразные и подробные «опыты», дающие все требуемые качественные и количественные свойства и характеристики объекта. Процесс моделирования сопровождается улучшением и уточнением, по мере необходимости, всех звеньев триады...».

На наш взгляд, эта триада математического моделирования и должна быть положена в основу математического образования. Для внедрения информационных технологий в структуру физико-математического образования необходимо решить следующие научно- методические задачи:

1. Создать информационное обеспечение учебного процесса:
   (a) создать электронные учебники;
   (b) создать генераторы индивидуальных заданий;
   (c) создать автоматизированную систему проверки индивидуальных заданий;
   (d) создать электронные библиотеки.

[5] См. Рис..1

2. Создать демонстрационное сопровождение лекций и практических занятий:

   (a) создать интерактивные 3D иллюстраций геометрических и физических объектов;

   (b) создать интерактивные видеоматериалы, сопровождающие вычисления;

   (c) создать анимационные математические модели объектов и явлений.

3. Встроить компьютерные вычисления в структуру практических занятий:

   (a) создать классы для комплексных учебных занятий по всем физико - математическим предметам;

   (b) встроить параллельное сопровождение практических занятий студентов компьютерными вычислениями;

   (c) создать программы аналитического тестирования и самотестирования учащихся.

4. Встроить компьютерные вычисления в структуру спецкурсов, курсовых и выпускных квалификационных работ:

   (a) сделать построение компьютерных математических моделей учащимися основой специальных курсов;

   (b) сделать создание учащимися авторских программных и научных продуктов, а также интерактивных учебных пособий обязательным элементом выпускных квалификационных работ.

При этом основной идеей внедрение информационных технологий в структуру физико-математического образования является компьютерное моделирование в системах компьютерной математики на Рис.I.9. Многочисленные исследования, проведенные различными авторами (см., например, [97], [102], [12] [112], [107],[16]), показывают, что среди известных систем компьютерной математики Maple является наиболее приемлемой для физико-математического образования СКМ, как по стоимости, так и по простоте интерфейса, а также соответствию языка программирования стандартному математическому языку.

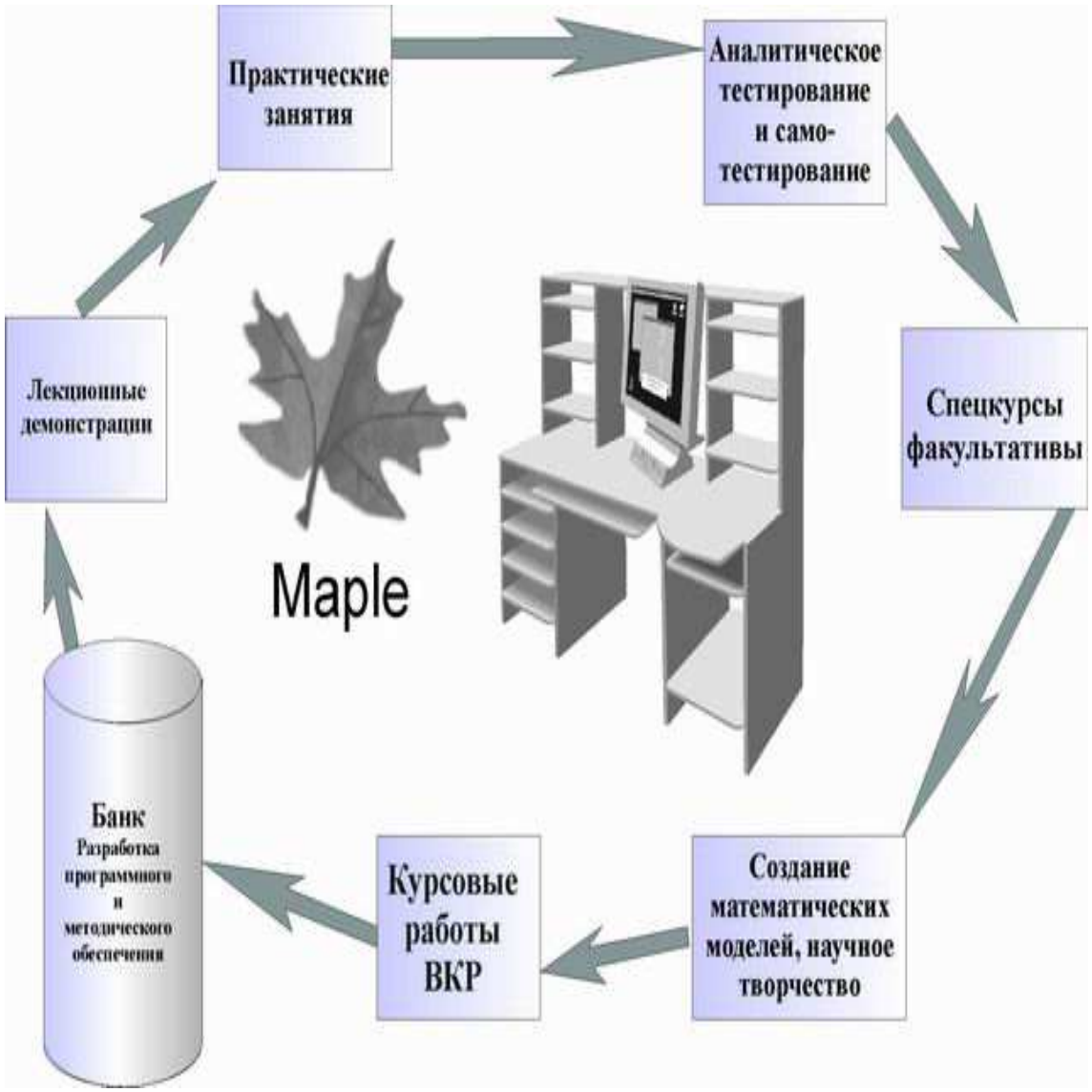


Рис.I.9 *Организация учебного процесса по физико-математическим дисциплинам на основе СКМ.*

Представленная на Рис.I.9 схема учебного процесса предполагает решение следующих учебно-научных задач:

1. (a) создание компьютерных моделей изучаемых явлений, привлечение информационных технологий в процесс преподавания предмета;
   (b) создание интерактивных учебных пособий и систем аналитического тестирования;
   (c) привлечение методов символьной математики для описания сложных явлений;

(d) замена академического метода преподавания интерактивным с использованием информационных технологий;

(e) использование компьютерных технологий для переориентации интересов молодежи к научному творчеству;

I.6.3 Методическое и программное обеспечение внедрения информационных технологий в структуру физико - математического образования

Организация вышеуказанного учебного цикла с глубоким использованием информационных технологий на основе СКМ требует больших наукоемких вложений, как на стадии запуска учебного процесса, так и на всех его дальнейших стадиях. Уже на первых стадиях учебного процесса требуется наличие большого количества заранее разработанных компьютерных моделей изучаемых объектов, как для лекционных демонстраций, так и для семинарских и самостоятельных занятий студентов. Разрабатываемые для обеспечения учебного процесса компьютерные модели должны удовлетворять ряду обязательных требований:

1. они должны быть наглядными;
2. они должны отображать все основные свойства исследуемой модели;
3. они должны быть интерактивными, т.е., позволять пользователю манипулировать ими с помощью внешних устройств;
4. они должны быть многопараметрическими для обеспечения возможности проведения численных экспериментов.

Проблема обеспечения наглядности математических структур играет важную роль в высшем образовании, так как усвоение фундаментальных геометрических понятий подготавливает фундамент для понимания процесса математического моделирования и овладения методами компьютерного моделирования, что в свою очередь, создает предпосылки для инновационного развития современного образования. Заметим, что многопараметричность создаваемых компьютерных моделей является важнейшим фактором компьютерных моделей, позволяющим управлять математической моделью, т.е., проводить компьютерное моделирование. В связи с этим важную роль играет компьютерная визуализация математических моделей, а особенно, *оснащенная динамическая визуализация*, основные принципы которой разрабо-

таны в работах [97], [101], [105], [107],[17]. Создание таких сложных компьютерных моделей возможно формате независимых пакетов программ (библиотек программ), которые могут использованы, как преподавателями, так и студентами вызовом соответствующих библиотек и содержащихся в них многопараметрических команд, имеющих простой синтаксис (см., например, [98], [99], [108],[16]). Необходимо подчеркнуть, что увеличение степени наглядности и интерактивности учебных материалов, созданных средствами ИТ, требует вложения больших интеллектуальных затрат и высокой степени профессионализма преподавателей.

Решение проблемы компьютерной реализации объектов линейной алгебры и аналитической геометрии и создания наглядных геометрических образов (интерпретаций) объектов, структур и свойств предполагает решение трех основных задач:

1. построение математических моделей основных алгебраических структур, объектов и свойств;

2. построение их геометрических интерпретаций, т.е., сопоставление им геометрических моделей;

3. построение многопараметрических компьютерных моделей графических образов объектов.

Заметим, что многопараметричность создаваемых компьютерных моделей является важнейшим фактором компьютерных моделей, позволяющим управлять математической моделью, т.е., проводить компьютерное моделирование. Наиболее эффективное решение этих задач возможно в системах компьютерной математики (СКМ), среди которых для целей образования наиболее удобна система Maple. Основными достоинствами этой системы применительно к задачам образования являются: относительно невысокая стоимость (по сравнению с MatLab и Mathematica), дружественный и интерактивный интерфейс, великолепные графические возможности, в частности, интерактивная трехмерная графика и динамическая (анимация). В этой статье мы рассмотрим основные принципы математического и компьютерного моделирования объектов линейной алгебры и аналитической геометрии в СКМ Maple. Заметим, что для рассмотренных здесь программных процедур конкретная версия Maple, начиная с версии 6, не имеет значения.

Необходимо подчеркнуть, что увеличение степени наглядности и интерактивности учебных материалов, созданных средствами ИТ, требует вложения больших интеллектуальных затрат и высокой степени профессио-

нализма преподавателей. В первую очередь, сказанное касается предметов физико-математического цикла. Здесь центральной идеей создания высококачественных электронных учебных материалов является математическое моделирование изучаемых объектов и явлений. Создание математической модели изучаемого объекта во многом определяет наглядность и степень усвоения изучаемого материала. Поэтому основными образовательными требованиями к математической модели должны быть: ее многопараметричность, возможность графической трехмерной реализации, интерактивность, возможность построения анимационных (графических динамических) представлений.

Системы компьютерной математики (СКМ), в первую очередь Maple, предоставляют уникальные программные и графические возможности для реализации этой идеи [97]. Однако, попытка прямого применения стандартных процедур СКМ далеко не всегда дает желаемый результат. Для получения качественных графических и анимационных моделей основных математических структур анализа функций приходится создавать пользовательские многопараметрические программные процедуры, простые для неискушенного в программировании пользователя, которые удобно объединять в специализированные библиотеки пользовательских процедур [16].

I.6.4 Технология совместного изучения математики и компьютерного моделирования

Основными методами реализации идеи информатизации предметов физико - математического цикла на основе математического и компьютерного моделирования среде компьютерной математики являются следующие:

1. профильная направленность курсов математики;
2. использование метода математического моделирования как основного метода изучения специальных предметов;
3. выстраивание всей системы подготовки специалистов вокруг решения научно-технических проблем и подготовки дипломного проекта;
4. встраивание компьютерного моделирования во все специальные курсы;
5. организация занятий по специальным предметам в форме лабораторных комплексных научных исследований с применением компьютерной математики и ИТ;

6. главным критерием для получения диплома необходимо считать квалификационную работу с обязательным применением методов компьютерного моделирования и возможностью научной публикации либо прямого использования в учебном процессе.

Необходимыми организационными мероприятиями для материального обеспечения внедрения информационных технологий в структуру физико - математического образования являются:

1. пересмотреть учебные программы специальных предметов;
2. создать учебно-методическое обеспечение специальных курсов;
3. организовать переподготовку преподавателей в области компьютерного моделирования и ИТ;
4. оборудовать современные компьютерные лаборатории ИТ;
5. обеспечить эти лаборатории лицензионными пакетами Mathematica, Maple, MatLab, CorelDraw, Delfi, WinEdt, MicrosoftOffice и другими;
6. переоборудовать классы под семинарские занятия в классы для комплексных занятий с применением компьютеров;
7. организовать систему летних научных школ для студентов и аспирантов по математическому и компьютерному моделированию.

На Рис.I.10 показан возможный вариант такого компьютерного класса, составленного из модулей Рис.I.11, позволяющих проводить комплексные занятий с применением компьютеров.

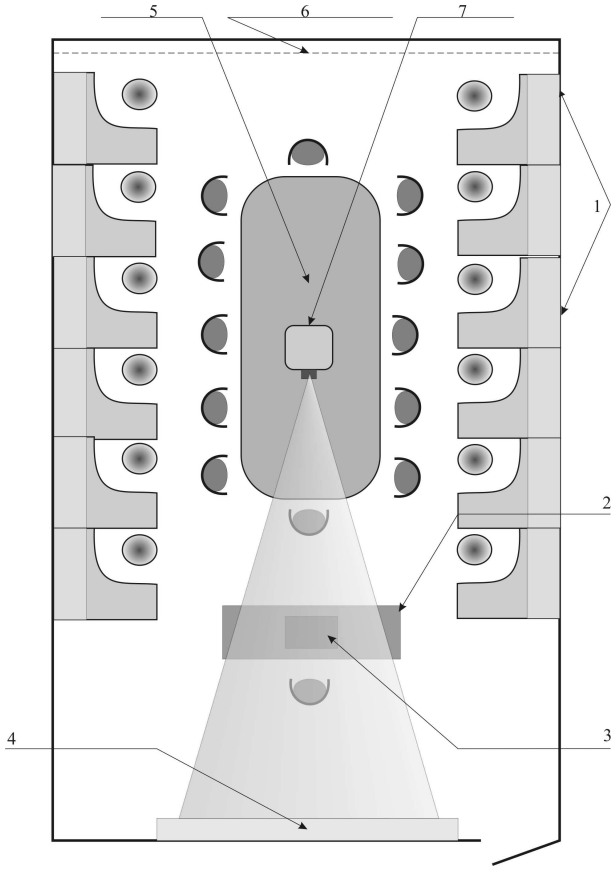


Рис.I.10 *Компьютерный класс для комплексного изучения предметов физико-математического цикла: 1 - модули; 2 - стол преподавателя;3 - компьютер преподавателя; 4 - интерактивная доска; 5 - дискуссионный стол; 6 – жалюзи; 7 – проектор.*

Рис.I.11 *Рабочее место для студентов (модуль) для комплексных занятий с применением компьютера.*

Глава II

Моделирование объектов алгебры и аналитической геометрии и математического анализа в Maple

II.1 Программные процедуры решения систем линейных алгебраических уравнений

II.1.1 Введение

Учебный модуль высшей алгебры является составной и базовой частью курса высшей математики для нематематических факультетов. Основная цель этого модуля – изучение теории линейных алгебраических уравнений, необходимой, как в других модулях курса высшей математики и предметах естественно - научного цикла, так и имеющей самостоятельную ценность для решения многочисленных прикладных задач. Кроме того, в этот модуль включается изучение основ матричного исчисления и теории определителей, необходимых для изучения теории систем линейных алгебраических уравнений (СЛАУ) и также имеющих многочисленные приложения в других модулях курса высшей математики и имеющих самостоятельную ценность. Поэтому информатизация этого модуля имеет большое значение для изучения курса высшей математики.

В работе [151] были рассмотрены основные принципы математического и компьютерного моделирования объектов линейной алгебры и аналитической геометрии в системе компьютерной математики (СКМ) Maple. В частности, в указанной работе была представлена программа автоматизированного решения системы линейных алгебраических уравнений в СКМ Maple с выводом решений в стандартной для Российской системы образования форме. Однако, описанная в [151] программная процедура, обладает существенным

недостатком: при наличии нулевых коэффициентов перед неизвестными эти переменные не считываются и соответствующие нули не попадают в расширенную матрицу системы. Кроме того, встроенные программные процедуры Maple не содержат программы нахождения фундаментальных решений СЛАУ, имеющих большую ценность в многочисленных приложениях и составляющих основу математической культуры. Решение указанных двух задач программными способами в СКМ Maple потребовало значительного усложнения программных процедур. Здесь следует отметить общее правило создания программных продуктов: чем менее профессиональным является пользователь программного продукта, тем большая степень программного сервиса требуется от этого продукта, а, значит, и более сложные соответствующие программные процедуры. Ниже мы укажем пути решения этих задач и опишем библиотеку соответствующих программных процедур `Algebra`, [152]. Указанные ниже коды описаны в программе [153].[1]

II.1.2 Математическая модель теории линейных алгебраических уравнений

Итак, рассмотрим систему алгебраических линейных уравнений из m уравнений относительно n неизвестных:

$$AX = B \Rightarrow \sum_{i=1}^{n} a_{ki} x_i = b_k; \quad (k = \overline{1, m}), \tag{II.1}$$

где**A** -основная матрица системы, **B** - матрица-столбец свободных членов, **X** – матрица-столбец неизвестных:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}; \quad B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}; \quad X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{II.2}$$

Пользователь вводит указанную систему в Maple не в матричной, а в стандартной форме упорядоченного списка уравнений:

$$\text{Sys} := [\alpha_1 * x + \alpha_2 * y + \dots + \alpha_n * u = b_1, \beta_1 * x + \beta_2 * y + \dots + \beta_n * u = b_2, \\ \dots, \delta_1 * x + \delta_2 * y + \dots + \delta_n * u = b_m], \tag{II.3}$$

где $\alpha_i, \beta_i, \dots, \delta_i, b_j$ - коэффициенты уравнений (V.5), т.е., конкретные числа, а $x, y, \dots, u$ —- имена переменных, которые в конкретном задании могут быть

[1] При использовании библиотеки программ `Algebra` ссылка на нее обязательна.

произвольными. Поэтому первой задачей является компьютерное распознавание системы (III.3) и приведение её к стандартному виду (V.5), а также нахождение расширенной матрицы системы:

$$\tilde{A} = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right) \tag{II.4}$$

Итак, опишем библиотеку `Algebra` автоматизированного решения систем линейных алгебраических уравнений.Для создания библиотеки вводим пустую таблицу с указанным наименованием.

```
>restart:
Algebra:=table():
```

II.1.3 Процедуры распознавания системы линейных алгебраических уравнений

Для распознавания системы (V.5) библиотека `Algebra` имеет программную процедуру `Algebra[InfoEq]`, которая предоставляет информацию о одиночном линейном уравнении в виде упорядоченного списка, первый элемент которого есть упорядоченный список имен переменных уравнения, второй элемент - упорядоченный список коэффициентов при этих переменных, третий элемент - правую часть уравнения:

```
> Algebra[InfoEq]:=proc(Eq) local n,i,xxx:
n:=Algebra[number_members](Eq):
xxx:=(i)->Algebra[coef_var](Eq,i):
[[seq(xxx(i)[2],i=1..n)],[seq(xxx(i)[1],i=1..n)],
rhs(Eq)]:end proc:
```

Здесь использована процедура `number_coef`, которая находит коэффициент у i-той переменной, причем если коэффициент равен 1, т.е., множитель перед переменной отсутствует, то результат действия программы будет «1», в других ненулевых случаях результат равен «2»:

Процедура `number_members` находит количество переменных в левой части уравнения

```
> Algebra[number_members]:=(Eq)->nops(lhs(Eq)):
```

Пример. Количество переменных уравнения x+2*y-3*z=7 равно 3: (x,y,z)

Продемонстрируем действие этой процедуры:

```
> Algebra[number_members](x+2*y-3*z=7);
```

$$3$$

Процедура `number_coef` определяет фактор коэффициента y i - той переменной, причем, если коэффициент не записан в численной форме, то программа дает значение 1, в других ненулевых случаях равен фактор равен 2

```
> Algebra[number_coef]:=(Eq,i)->
> nops([op([op(lhs(Eq))])][i]):
```

Примеры:

```
> Algebra[number_coef](x+2*y-3*z=7,1);
```

$$1$$

Процедура `coef_var` создает упорядоченный список [k,z] для i-той переменной в уравнении, где k-коэффициент перед i-той переменной в уравнении, z-имя этой переменной

```
> Algebra[coef_var]:=proc(Eq,i) local xx,xxx:
> xx:=[op([op([op(lhs(Eq))])][i])]:
> if Algebra[number_coef](Eq,i)=2 then xxx:=xx:
> else xxx:=[1,op(xx)]:end if:xxx:end proc:
```

Пример

```
> Algebra[coef_var](x+2*y-3*z=7,2);
```

$$[2, y]$$

```
> Algebra[coef_var](x+2*y-3*z=7,1);
```

$$[1, x]$$

Процедура `InfoEq` представляет информацию о единичном линейном уравнении в виде упорядоченного списка, первый элемент которого представляет собой упорядоченный список имен переменных уравнения, второй элемент - упорядоченный список коэффициентов при этих переменных, третий элемент - правую часть уравнения, т.е., фактически записывает строку расширенной матрицы, соответствующей этому уравнению.

```
> Algebra[InfoEq]:=proc(Eq) local n,i,xxx:
> n:=Algebra[number_members](Eq):
> xxx:=(i)->Algebra[coef_var](Eq,i):
> [[seq(xxx(i)[2],i=1..n)],[seq(xxx(i)[1],i=1..n)],rhs(Eq)]:end
proc:
```

Пример:

```
> Algebra[InfoEq](x+2*y-3*z=7);
```

$$[[x, y, z], [1, 2, -3], 7]$$

Процедура `associate` находит объединение подмножеств `x`, представленных в виде x={{a_1,..,a_r},{b_1,b_2,..,b_s},..,{c_1,..,c_m}}. Конечное множество записывает в виде упорядоченного

списка :

```
> Algebra[associate]:=proc(x) local elem,u:
> u:=x[1]:
> for elem in x do
> u:=u union elem:
> end do:[op(u)]: end proc:
```

Пример:

```
> Algebra[associate]([{x,y,z},{y,u,z},{y,t,w},{r,u,s}]);
```

$$[r, s, t, u, w, y, z, \mathrm{x}]$$

II.1.4 Формирование системы уравнений в стандартном виде

Процедура `StandartSys` представляет систему линейных уравнений в стандартном виде с унифицированными именами переменныхX_i, где i изменяется в пределах от 1 до `nnn` – числа неизвестных:

```
> Algebra[StandartSys]:=proc(Eqs)
> local nn,Eq,i,vars,Vars,nnn,k,SB,
> EQS:
> nn:=nops(Eqs):
> Eq:=(i)->Eqs[i]:
> vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
> Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}):
> nnn:=nops(Vars):
> SB:={seq(Vars[k]=X[k],k=1..nnn)}:
> EQS:=subs(SB,Eqs):
> EQS:
> end proc:
```

Пример:

```
> Algebra[StandartSys]([x+2*y-z=5,\
> x+y-3*z=7,5*x-3*y+2*z=9,x+y+z=1]);
```

```
>[X[1]+2*X[2]-X[3] = 5, X[1]+X[2]-3*X[3] = 7,
5*X[1]-3*X[2]+2*X[3] = 9,
X[1]+X[2]+X[3] = 1]}[[X_{{1}}+2*X_{{2}}-X_{{3}}=5,
X_{{1}}+X_{{2}}-3*X_{{3}}=7, 5*X_{{1}}-3*X_{{2}}+2*X_{{3}}=9,
X_{{1}}+X_{{2}}+X_{{3}}=1]]
```

Процедура `Algebra[MatrSys]` создает упорядоченный список, состоящий из двух матриц - основной матрицы системы и расширенной:

```
Algebra[MatrSys]:=proc(Eqs) local nn,Eq,i,vars,JJ,Vars,nnn,k,SB,
EQS,aa,bb,AA,AB,AAA,BBB:
nn:=nops(Eqs):
```

```
Eq:=(i)->Eqs[i]:
vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}):
nnn:=nops(Vars):
SB:={seq(Vars[k]=X[k],k=1..nnn)}:
EQS:=subs(SB,Eqs):
aa:=(i,k)->coeff(lhs(EQS[i]),X[k]):
bb:=(i)->rhs(EQS[i]):
AA:=[seq([seq(aa(i,k),k=1..nnn)],i=1..nn)]:
AB:=[seq([seq(aa(i,k),k=1..nnn),bb(i)],i=1..nn)]:
JJ:=[convert(AA, matrix),convert(AB,matrix)]:
JJ:
end:
```

Пример:

```
> Algebra[MatrSys]([x+2*y-z=5, y-3*z=7,5*x-3*y+2*z=9]);
```

$$\left[\begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -3 \\ 5 & -3 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 & -1 & 5 \\ 0 & 1 & -3 & 7 \\ 5 & -3 & 2 & 9 \end{bmatrix}\right]$$

Опишем сначала некоторые промежуточные, простые, удобные для пользователя процедуры. Программная процедура `Matrconverb(XX)` конвертирует список `XX` в матрицу `XX`:

```
>Algebra[Matrconverb]:=proc(XX) local nn,mm,AAM,MMM:
nn:=nops(XX):
mm:=nops(XX[1]):
MMM:=array(1..nn,1..mm,XX):
AAM:=convert(MMM,matrix):
end:
```

Программная процедура `convert_vec(YY)` преобразует одномерный список `YY` в матрицу `(matrix)`:

```
>Algebra[convert_vec]:=proc(YY) local ZZ,nn:
nn:=nops(YY):
ZZ:=array(1..nn,YY):
convert(ZZ,matrix):
end proc:
```

Следующие две процедуры конвертируют списки в векторы.

```
>Matrlinalgg(X,Y)
Algebra[Matrlinalgg]:=proc(X,Y) local XX,YY:
XX:=Algebra[Matrconverb](X):
YY:=Algebra[convert_vec](Y):
linalg[linsolve](XX,YY,'_t[]',S):
end proc:
```

```
>Algebra[Matrlinalg]:=proc(X,Y) local XX,YY,ZZ:
XX:=Algebra[Matrconverb](X):
YY:=convert(Y,vector):
ZZ:=linalg[linsolve](XX,YY,'_t[]',S):
convert(ZZ,matrix):
end proc:
```

II.1.5 Получение общего решения СЛАУ в стандартном виде

Программная процедура `Algebra[SolLinGen]` находит общее решение системы линейных алгебраических уравнений, причем значения параметра `t=matr` соответствует вывод решения системы в матричном виде, в котором произвольные константы выводятся в формате C_i , где i номер строки, содержащей только эту константу с коэффициентом 1. В случае, если система не совместна, то вместо решения команда выводит сообщение о несовместности системы уравнений. При любом другом значении параметра t, например, 123a, - решение выводится в списочном виде: `x=x1, y=y1,....`:

```
\begin{maplegroup}
\begin{mapleinput}
\mapleinline{active}{1d}{Algebra[SolLinGen]:=proc(Eqs,t)
local nn,Eq,i,vars,Vars,nnn,k,SB,
EQS,aa,bb,AA,AB,r1,r2,BB,SSS,C,RRR:
nn:=nops(Eqs):
Eq:=(i)->Eqs[i]:
vars:=(i)->\{op(Algebra[InfoEq](Eq(i))[1])\}:
Vars:=Algebra[associate](\{seq(vars(i),i=1..nn)\}):
nnn:=nops(Vars):
SB:=\{seq(Vars[k]=X[k],k=1..nnn)\}:
EQS:=subs(SB,Eqs):
aa:=(i,k)->coeff(lhs(EQS[i]),X[k]):
```

```
bb:=(i)->rhs(EQS[i]):
AA:=convert([seq([seq(aa(i,k),k=1..nnn)],i=1..nn)],Matrix):
AB:=convert([seq([seq(aa(i,k),k=1..nnn),bb(i)],i=1..nn)],Matrix):
BB:=Vector([seq(bb(i),i=1..nn)]):
r1:=linalg[rank](AA):r2:=linalg[rank](AB):
if r1=r2 then
SSS:=LinearAlgebra[LinearSolve](AA,BB,free=‘S‘):
RRR:=seq(Vars[i]=SSS[i],i=1..nnn):
else
("Система не совместна")
 end if:
if t=matr then SB,convert(SSS,Matrix):
else RRR:end if:
end proc:
```

Пример 1:

```
> Eq1:=[x+2*y-z=5, x+y-3*z=7,5*x-3*y+2*z=9,y+2*z=-2];
```

$$Eq1 := [x + 2\,y - z = 5, x + y - 3\,z = 7, 5\,x - 3\,y + 2\,z = 9, y + 2\,z = -2]$$

```
> Algebra[SolLinGen](Eq1,matr);
```

$$\{x = X_1, y = X_2, z = X_3\}, \begin{bmatrix} \frac{29}{11} \\ \frac{6}{11} \\ -\frac{14}{11} \end{bmatrix}$$

```
> Eq111:=[x+y=0, 2*y+z=0,z=0];
```

$$Eq111 := [x + y = 0, 2\,y + z = 0, z = 0]$$

```
> Algebra[SolLinGen](Eq111,matr);
```

$$\{x = X_1, y = X_2, z = X_3\}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Пример 2:

```
> Eq2:=[x-3*y+10*z-t+u=1,x+y+3*z-t-u=3,x-2*y+2*t+2*u=1];
```

$$Eq2 := [x - 3\,y + 10\,z - t + u = 1, x + y + 3\,z - t - u = 3, x - 2\,y + 2\,t + 2\,u = 1]$$

```
> SSEq2:=Algebra[SolLinGen](Eq2,matr);
```

$$SSEq2 := \{t = X_1, u = X_2, x = X_3, y = X_4, z = X_5\}, \begin{bmatrix} 1/3 - S_4 + 9/2\,S_5 \\ -1 + 2\,S_4 - 7/2\,S_5 \\ 7/3 - 2\,S_5 \\ S_4 \\ S_5 \end{bmatrix}$$

```
> Algebra[SolLinGen](Eq2,1);
```

$t = 1/3 - S_4 + 9/2\,S_5,\ u = -1 + 2\,S_4 - 7/2\,S_5,\ x = 7/3 - 2\,S_5,\ y = S_4,\ z = S_5$

```
> SSEq2Matr:=SSEq2[2];
```

$$SSEq2Matr := \begin{bmatrix} 1/3 - S_4 + 9/2\,S_5 \\ -1 + 2\,S_4 - 7/2\,S_5 \\ 7/3 - 2\,S_5 \\ S_4 \\ S_5 \end{bmatrix}$$

```
> subs({S[3]=1,S[5]=0},SSEq2Matr);
```

$$\begin{bmatrix} 1/3 - S_4 \\ -1 + 2\,S_4 \\ 7/3 \\ S_4 \\ 0 \end{bmatrix}$$

```
> SSEq2Vec:=convert(SSEq2Matr,Vector);
```

$$SSEq2Vec := \begin{bmatrix} 1/3 - S_4 + 9/2\,S_5 \\ -1 + 2\,S_4 - 7/2\,S_5 \\ 7/3 - 2\,S_5 \\ S_4 \\ S_5 \end{bmatrix}$$

II.1.6 Процедуры нахождения фундаментального решения системы линейных алгебраических уравнений

Отметим, что хотя в СКМ Maple имеется множество команд для решения систем линейных алгебраических уравнений, но в ней по какой-то непонятной причине отсутствуют процедуры нахождения фундаментальной системы решений. Ниже мы восполним этот пробел.

```
> Algebra[Find]:=proc(X,Y) local XX,XY:
> XY:=op(convert(linalg[transpose](X),list));
> if Y in XY then XX:=1:
> else XX:=0: end if:
> end proc:
```

Пример:

```
> Algebra[Find]([s,h,F[77],5,9,C[2],3],F[77]);
> Algebra[Find]([s,h,F[77],5,9,C[2],3],F[76]);
```

$$1$$

$$0$$

Процедура `Finder(s,a)` отыскивает элемент `a` в списке `s` и определяет его положение; в случае, если элемент `a` не содержится в списке `s`, команда выводит пустой список. Надо заметить, что *Finder - очень удобная команда, она находит все, что можно, если же искомого элемента нет, пишет пустую скобку* [].

```
> Algebra[Finder]:= proc(s,a) local ss,i,j,aa:
> aa:=convert(a,symbol):
> ss:=[]:
> for i from 1 to LinearAlgebra[RowDimension](s)  do
> for j from 1 to LinearAlgebra[ColumnDimension](s) do
> if(convert(s[i,j],symbol)=aa) then
> ss:=[op(ss),[i,j]]:
> end if:
> end do:
> end do:
> ss:
> end proc:
```

Пример:

```
> ssss0:=Algebra[Finder](SSEq2Matr,S[5]):ssss0;
```

$$[[5, 1]]$$

```
> ssss:=Algebra[Finder](SSEq2Matr,S[3]):ssss;
```

$$[]$$

```
> ssss1:=Algebra[Finder](SSEq2Matr,S[6]):ssss1;
```

$$[]$$

```
> delta:=(i,k)->if i=k then 1:else 0:end if:
```

```
> delta(4,3);delta(12,12);
```

$$0$$
$$1$$

Процедура `List_Num` определяет положение элементов в матрице-столбце и записывает это положение в виде упорядоченного списка:

```
> Algebra[List_Num]:=proc(A) local i,ss,nn:
> nn:=nops(A):
> ss:=[]:
> for i from 1 to nn  do
> if A[i]=1 then
> ss:=[op(ss),i]:
> end if:
> end do:
> ss:
> end proc:
```

Пример:

```
> f4:=[0,1,0,0,1,1,0,1];
```

$$f4 := [0, 1, 0, 0, 1, 1, 0, 1]$$

```
> Algebra[List_Num](f4);
```

$$[2, 5, 6, 8]$$

```
> Algebra[List_Num](SSEq2Vec);
```

$$[]$$

Программная процедура `Algebra[SolLinBasic]` находит фундаментальное решение системы линейных алгебраических уравнений, в матричном виде. В случае, если система не совместна, то вместо решения команда выводит сообщение о несовместности системы уравнений.

```
> Algebra[SolLinBasic]:=
> proc(Eqs) local nn,Eq,i,vars,Vars,nnn,k,SB,
> EQS,aa,bb,AA,AB,r1,r2,BB,SSS,S,RRR,SSSM,
> nFC,NC,mmm,j,ii,FSJ,FS:
> nn:=nops(Eqs):
> Eq:=(i)->Eqs[i]:
> vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
> Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}):
> nnn:=nops(Vars):
> SB:={seq(Vars[k]=X[k],k=1..nnn)}:
> EQS:=subs(SB,Eqs):
> aa:=(i,k)->coeff(lhs(EQS[i]),X[k]):
> bb:=(i)->rhs(EQS[i]):
> AA:=convert([seq([seq(aa(i,k),k=1..nnn)],i=1..nn)],
> Matrix):
> AB:=convert([seq([seq(aa(i,k),k=1..nnn),bb(i)],i=1..nn)],
> Matrix):
> BB:=Vector([seq(bb(i),i=1..nn)]):
> r1:=linalg[rank](AA):r2:=linalg[rank](AB):
> if r1=r2 then
> SSS:=LinearAlgebra[LinearSolve](AA,BB,free=‘S‘):
> SSSM:=convert(SSS,Matrix):
> nFC:=[seq(nops(Algebra[Finder](SSSM,S[i])),i=1..nnn)]:
> NC:=Algebra[List_Num](nFC):
> mmm:=nops(NC):
> ii:=(j)->NC[j]:
> FSJ:=(j)->subs({seq(S[ii(k)]=delta(k,j),k=1..mmm)},
> SSSM):
> FS:=convert([seq(FSJ(j),j=1..mmm)],Matrix):
> else
> ("Система не совместна")
> end if:
> FS:
> end proc:
```

Пример:

```
> Algebra[SolLinBasic](Eq2);
```

$$\begin{bmatrix} -2/3 & \frac{29}{6} \\ 1 & -9/2 \\ 7/3 & 1/3 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Пример:

```
> Eq3:=[x+2*y-3*z+u-6*t+8*n=1,x-y+2*z-t-n=0,2*x+y-z+3*u-t=1];
> Eq3S:=Algebra[SolLinBasic](Eq3);
```

$$Eq3S := \begin{bmatrix} -\frac{10}{19} & -\frac{12}{19} & \frac{36}{19} \\ -\frac{18}{19} & -\frac{14}{19} & \frac{42}{19} \\ 1 & 0 & 0 \\ -\frac{28}{19} & -\frac{7}{19} & \frac{40}{19} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Eq3S[4,2];
```

$$-\frac{7}{19}$$

```
> [seq([seq(Eq3S[i,k],i=1..6)],k=1..3)];
```

$$[[-\frac{10}{19}, -\frac{18}{19}, 1, -\frac{28}{19}, 0, 0], [-\frac{12}{19}, -\frac{14}{19}, 0, -\frac{7}{19}, 1, 0], [\frac{36}{19}, \frac{42}{19}, 0, \frac{40}{19}, 0, 1]]$$

```
> %[2];
```

$$[-\frac{12}{19}, -\frac{14}{19}, 0, -\frac{7}{19}, 1, 0]$$

Процедура `Algebra[SolLinBasic]` выводит фундаментальное решение в виде матрицы, каждый столбец которой представляет линейно независимое решение:

```
> Algebra[SolLinBasics]:=proc(Eqs) local mmm,nnn,i,k,SSSS:
> SSSS:=Algebra[SolLinBasic](Eqs):
> mmm:=LinearAlgebra[RowDimension](SSSS):
> nnn:=LinearAlgebra[ColumnDimension](SSSS):
> [seq([seq(SSSS[i,k],i=1..mmm)],k=1..nnn)]:
> end proc:
```

Пример:

```
> Algebra[SolLinBasics](Eq3);
```

$$[[-\frac{10}{19}, -\frac{18}{19}, 1, -\frac{28}{19}, 0, 0], [-\frac{12}{19}, -\frac{14}{19}, 0, -\frac{7}{19}, 1, 0], [\frac{36}{19}, \frac{42}{19}, 0, \frac{40}{19}, 0, 1]]$$

Наконец, сохраним библиотеку «Algebra» в файле «SolLin.m».

```
> save(Algebra,`SolLin.m`):
```

Созданная библиотека процедур приспособлена для студентов нематематических факультетов, удобна и проста в работе. Она может быть использована и преподавателями для проверки работ студентов, а также для генерации заданий. Кроме того, библиотека может быть использована и иссле-

дователями нематематического профиля для проведения исследовательских работ.

II.2 Математическое моделирование трехмерных линейных объектов

II.2.1 Алгебраический этап моделирования

Как известно, линейная алгебра и аналитическая геометрия традиционно тесно связаны между собой, что вызвано, как историческими корнями этих разделов математики, так и определяющей зависимостью структур аналитической геометрии структурами и методами линейной алгебры. Особенно тесной эта связь проявляется в области векторной алгебры. В свою очередь, все наглядные интерпретации алгебраической структуры векторных пространств, фактически, являются геометрическими. Проблема обеспечения наглядности этих структур играет важную роль в высшем образовании, так как усвоение фундаментальных геометрических понятий подготавливает фундамент для понимания процесса математического моделирования и овладения методами компьютерного моделирования, что в свою очередь, создает предпосылки для инновационного развития современного образования. Решение проблемы компьютерной реализации объектов линейной алгебры и аналитической геометрии и создания наглядных геометрических образов (интерпретаций) объектов, структур и свойств предполагает решение трех основных задач:

1. построение математических моделей основных алгебраических структур, объектов и свойств;

2. построение их геометрических интерпретаций, т.е., сопоставление им геометрических моделей;

3. построение многопараметрических компьютерных моделей графических образов объектов.

Заметим, что многопараметричность создаваемых компьютерных моделей является важнейшим фактором компьютерных моделей, позволяющим управлять математической моделью, т.е., проводить компьютерное моделирование.

Системы линейных алгебраических уравнений имеет адекватную геометрическую интерпретацию (см., например, [154]): каждое линейное алгебраическое уравнение в n-мерном пространстве определяет гиперплоскость, а система линейных алгебраических уравнений определяет взаимное расположение этих гиперплоскостей. Если система уравнений не имеет решения, то гиперплоскости не пересекаются, т.е., параллельны; в случае существования решения гиперплоскости пересекаются по k-мерным плоскостям, включая 1-мерные (прямые) и 0-мерные (точки). В случае трехмерных пространств, которые, фактически, и важны для общего среднего и нематематического высшего образования, число различных типов взаимного расположения плоскостей или прямых значительно сокращается.

Как известно, прямая в евклидовом пространстве E_3 определяется опорной точкой $_0$ и ненулевым *направляющим* вектором $\mathbf{q} \neq \mathbf{0}$ как геометрическое место точек E_3:

$d(M_0, q) = \left\{ M \in E_3 \middle| \overrightarrow{M_0M} = \lambda q \right\}$, *где параметр* λ *принимает значения на всем множестве действительных чисел* $\mathbb{R}$. Плоскость в евклидовом пространстве также определяется опорной точкой $_0$ и ненулевым *нормальным* вектором $\mathbf{N} \neq \mathbf{0}$, как геометрическое место точек E_3:

$\Pi(M_0, N) = \left\{ M \in E_3 \middle| \left(N, \overrightarrow{M_0M} \right) = 0 \right\}$.

Соответственно этим определениям, при заданном декартовом репере $R\{O, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, где $\mathbf{e}_i$ – векторы ортонормированного базиса: $(e_i, e_k) = \delta_{ik}$; $\left(i, k = \overline{1,3}\right)$, прямая $d(M0,\mathbf{q})$ описывается параметрическими (или каноническими) уравнениями:

$$d(M_0, q): \quad x^i = x_0^i + \lambda q^i; \qquad \left(i = \overline{1,3}\right), \tag{II.5}$$

где x_0^i, q^i, x^i - координаты опорной точки $M0$, направляющего вектора $\mathbf{q}$, и текущей точки прямой M, соответственно. Плоскость же в евклидовом пространстве *E3* описывается общим уравнением: $\Pi(M_0, N) : (r - r_0, N) = 0 \Rightarrow$

$$A_i(x^i - x_0^i) = 0 \Rightarrow A(x - x_0) + B(y - y_0) + C(z - z_0) = 0, \tag{II.6}$$

где A_i – координаты нормального вектора $\mathbf{N} = (A, B, C)$.

При построении математических моделей линейных объектов в трехмерном евклидовом пространстве принципиально важными являются три теоремы (см. [154] и более подробно [155]):

Теорема 1. *Две прямые,* $d(M_1, \mathbf{q}_1)$ и $d(M_1, \mathbf{q}_1)$, *в евклидовом пространстве* E_3:

1. пересекаются в единственной точке при условии:

$$(\mathbf{q}_1, \mathbf{q}_2, \overrightarrow{M_1M_2}) \neq 0; \tag{II.7}$$

2. параллельны при условиях:

$$(\mathbf{q}_1, \mathbf{q}_2, \overrightarrow{M_1M_2}), \tag{II.8}$$

$$[\mathbf{q}_1, \mathbf{q}_2] \equiv \mathbf{q}_1 \times \mathbf{q}_2 \neq \mathbf{0}; \tag{II.9}$$

3. совпадают при условиях (V.8) и:

$$[\mathbf{q}_1, \mathbf{q}_2] \equiv \mathbf{q}_1 \times \mathbf{q}_2 = \mathbf{0}; \tag{II.10}$$

4. или скрещиваются при условии:

$$(\mathbf{q}_1, \mathbf{q}_2, \overrightarrow{M_1M_2}) = 0. \tag{II.11}$$

Теорема 2. *Две плоскости,* $\Pi_1(_1, \mathbf{N}_1)$ *и* $\Pi_2(_2, \mathbf{N}_2)$*, в евклидовом пространстве* E_3*:*

1. пересекаются по единственной прямой при условии:

$$[\mathbf{q}_1, \mathbf{q}_2] \neq 0; \tag{II.12}$$

2. параллельны при условиях:

$$[q_1, q_2] = 0, \tag{II.13}$$

$$M_2 \notin \Pi_1(M_1, \mathbf{N}_1); \tag{II.14}$$

3. или совпадают при условиях (V.13) и:

$$M_2 \in \Pi_1(M_1, N_1). \tag{II.15}$$

Теорема 3. *Прямая* $d(M_0, \mathbf{q})$ *и плоскость* $\Pi(_1, \mathbf{N})$ *в евклидовом пространстве* E_3*:*

1. пересекаются в единственной точке при условии:

$$(q, N) \neq 0; \tag{II.16}$$

2. параллельны при условиях:

$$(\mathbf{q}, \mathbf{N}) = 0, \tag{II.17}$$

$$M_0 \notin \Pi(M_1, N); \tag{II.18}$$

3. и совпадают при условиях (II.17) и:

$$M_0 \in \Pi(M_1, \mathbf{N}). \tag{II.19}$$

Фактически уравнения (V.6), (II.6) и перечисленные три теоремы с алгебраической точки зрения полностью описывают задачу о взаимном расположении линейных объектов[2]. Однако, с геометрической точки зрения задача о взаимном расположении линейных объектов еще далеко не завершена.

II.2.2 Геометрический этап моделировании

При получении ответа о реализации одного из перечисленных типов взаимного расположения линейных объектов в теоремах 1-3 (4 типа для прямых, 3 - для плоскостей и 3 – для плоскостей и прямых) необходимо далее решить конкретную геометрическую задачу. Такими задачами являются:

1. нахождение координат точки пересечения прямых;
2. вычисление угла между прямыми;
3. вычисление расстояния между параллельными прямыми;
4. нахождение угла между скрещивающимися прямыми;
5. нахождение прямой пересечения плоскостей;
6. вычисление угла между плоскостями;
7. вычисление расстояния между параллельными плоскостями;
8. нахождение координат точки пересечения прямой и плоскости;
9. вычисление угла между прямой и плоскостью;
10. вычисление расстояния между параллельными прямой и плоскостью.

Решение каждой из перечисленных выше стандартных геометрических задач обеспечивается рядом определений и алгоритмов, которые дополняют математическую модель и позволяют дать ей четкую геометрическую интерпретацию. Приведем пример моделирования взаимного расположения прямой и плоскости в пространстве в случае их взаимного пересечения (Рис. 1). На этом рисунке показано определение угла α между прямой d и плоскостью Π как угла между прямой d и ее ортогональной проекцией d1 на

[2] Под линейными объектами понимаются фигуры, описываемые системами линейных уравнений.

плоскость П. при этом численное значение угла определяется выражением: $\sin\alpha = \frac{(qN)}{|q||N|}$.

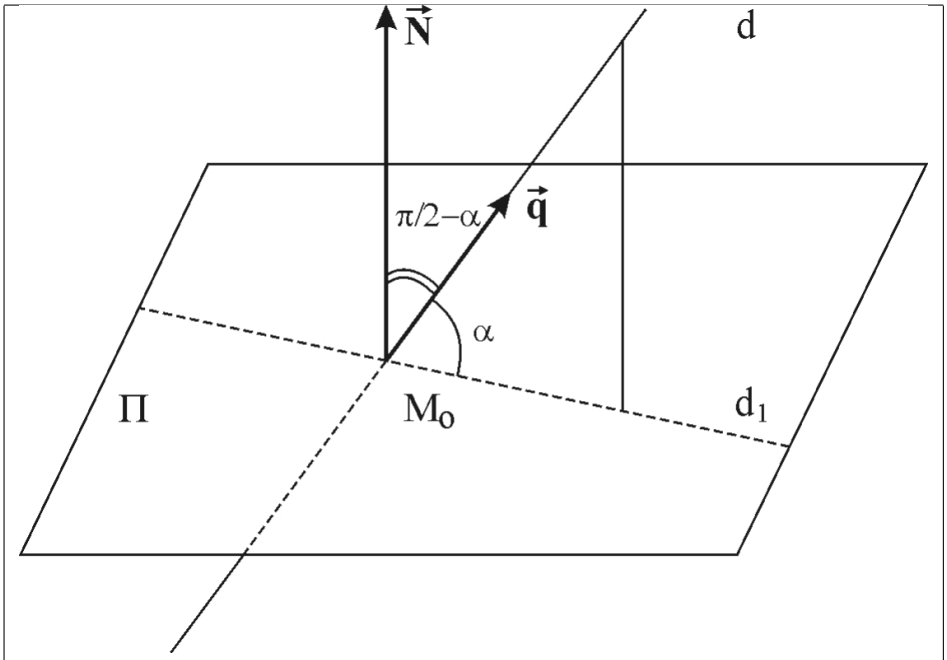


Рис. 1. Определение угла α между прямой d и плоскостью П.

II.2.3 Этап компьютерного моделирования

После завершения этапа математического моделирования переходим к компьютерному моделированию в пакете Maple.

Компьютерное моделирование взаимного расположения плоскостей

Заметим, что несмотря на внешнюю простоту рассмотренных выше математических моделей, их компьютерное моделирование достаточно сложно, поскольку основным требованием к программным продуктам, используемым в учебном процессе или в научной деятельности непрофессиональными математиками, является высокая степень наглядности и простоты их применения. Другими словами, обеспечение сервисного обслуживания ПО создает чаще гораздо больше проблем, чем принципиальное алгоритмическое решение проблемы. Создадим программную процедуру `EqLine3(Eqs,t)` полного исследования и геометрического моделирования взаимного расположения двух плоскостей, заданных общими уравнениями `Eqs` в библиотеке

`Anal_Geom` (см. [156], [157]), позволяющую определить взаимно расположение плоскостей в пространстве[3]:

```
>restart:
AnalGeo:=table():

>AnalGeo[EqLine3]:=proc(Eqs,t)
local n,m,A,B1,B,F,k1,k2,SS,SST,EQS,M0,M1,q,N1,N2,Q1,
Q2,M2,M3,M4,M5,GP1,GP2,GL,z1,z2,P1,P2,W1,W2:
n:=LinearAlgebra[ColumnDimension](convert(Eqs,Matrix)):
A:=convert([seq([coeffs(lhs(Eqs[i]))],i=1..n)],Matrix):
m:=LinearAlgebra[RowDimension](A):
B1:=seq(rhs(Eqs[i]),i=1..n):
B:=Vector([B1]):
F:=convert([seq([coeffs(lhs(Eqs[i])),rhs(Eqs[i])],i=1..n)],
Matrix):
k1:=linalg[rank](A):
k2:=linalg[rank](F):
if k1=2 and k2=2 then
SS:=LinearAlgebra[LinearSolve](A,B,free='C'):
if
SS[1]=C[1] then SST:=subs(C[1]=t,SS):
elif SS[2]=C[2] then SST:=subs(C[2]=t,SS):
elif SS[3]=C[3] then SST:=subs(C[3]=t,SS):
end if:
EQS:=[SST[1],SST[2],SST[3]]:
M0:=convert(subs(t=0,SST),list):
M1:=convert(subs(t=2,SST),list):
N1:=convert(linalg[row](A,1),Vector):
N2:=convert(linalg[row](A,2),Vector):
q:=convert(coeff(SST,t),Vector):
Q1:=convert(linalg[crossprod](q,N1),list):
Q2:=convert(linalg[crossprod](q,N2),list):
M2:=convert(linalg[matadd](M0,Q1,1,1),list):
M3:=convert(linalg[matadd](M1,Q1,1,1),list):
M4:=convert(linalg[matadd](M0,Q2,1,1),list):
M5:=convert(linalg[matadd](M1,Q2,1,1),list):
GL:=plots[spacecurve](EQS,t=0..2,color=red,thickness=2):
```

[3]При использовании библиотеки программ `Anal_Geom` ссылка на нее обязательна.

```
GP1:=plots[polygonplot3d]([M0,M2,M3,M1]):
GP2:=plots[polygonplot3d]([M0,M4,M5,M1]):
plots[display](GL,GP1,GP2,axes=BOXED,
title="Плоскости пересекаются по красной прямой"):
elif k1=1 and k2=1 then
("Плоскости совпадают")
else
z1:=LinearAlgebra[LinearSolve](Matrix([[linalg[row](A,1)],
[linalg[row](A,1)]]),Vector([B[1],B[1]]),free=‘Z‘):
z2:=LinearAlgebra[LinearSolve](Matrix([[linalg[row](A,2)],
[linalg[row](A,2)]]),Vector([B[2],B[2]]),free=‘U‘):
P1:=subs({Z[1]=0,Z[2]=0,Z[3]=0},z1):
P2:=subs({U[1]=0,U[2]=0,U[3]=0},z2):
if
z1[1]=Z[1] and z1[2]=Z[2] then W1:=subs({Z[1]=u,Z[2]=v},z1[3]):
GP1:=plot3d([u,v,W1],u=0..2,v=0..2,color=green):
W2:=subs({U[1]=u,U[2]=v},z2[3]): GP2:=plot3d([u,v,W2],
u=0..2,v=0..2,color=red):
elif
z1[1]=Z[1] and z1[3]=Z[3] then W1:=subs({Z[1]=u,Z[3]=v},z1[3]):
GP1:=plot3d([u,W1,v],u=0..2,v=0..2,color=green):
W2:=subs({U[1]=u,U[3]=v},z2[3]): GP2:=plot3d([u,W2,v],u=0..2,
v=0..2,color=red):
else
W1:=subs({Z[2]=u,Z[3]=v},z1[1]): GP1:=plot3d([W1,u,v],u=0..2,
v=0..2,color=green):
W2:=subs({U[2]=u,U[3]=v},z2[1]): GP2:=plot3d([W2,u,v],u=0..2,
v=0..2,color=red):
end if:
plots[display](GP1,GP2,axes=BOXED,title="Плоскости параллельны"):
end if:
end proc:
```

Примеры

Случай пересекающихся плоскостей

Покажем пример ввода этой программной процедуры для исследования взаимного расположения двух плоскостей, заданных парой общих уравнений:

$$\{2 * x - 2 * y + z = 21, x + 2 * y - z = -9\} :$$

`>EqLine3([2*x-2*y+z=21,x+2*y-z=-9],t);` и ее исполнения (Рис. Рис.II.13). Как видно из этого рисунка исполнение команды дает изображение двух плоскостей, линию их пересечения и название рисунка, содержащее информацию о взаимном расположении плоскостей. Заметим также, что в результате исполнения команды образуется интерактивный 3D-объект, который можно легко повертеть и рассмотреть в различных ракурсах. Также заметим, что 3D-графика в в версии Maple17 просто великолепна.

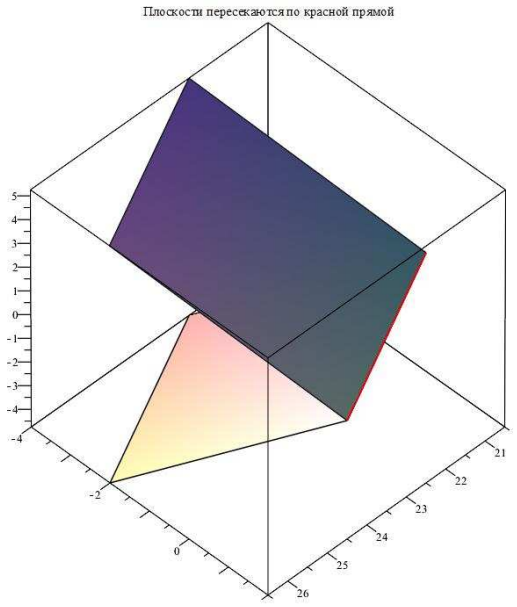


Рис.II.12 Исполнение команды EqLine3([2*x-2*y+z=21,x+2*y-z=-9],t). Плоскости пересекаются по красной прямой.

Случай параллельных плоскостей

EqLine3([x-2*y-3*z=0,x-2*y-3*z=4],t);

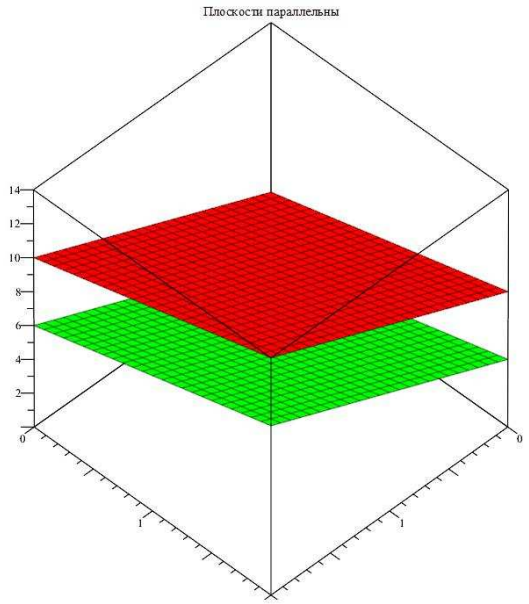


Рис.II.13 Исполнение команды EqLine3([x-2*y-3*z=0,x-2*y-3*z=4],t). Плоскости параллельны.

Случай совпадающих плоскостей

```
>EqLine3([x+2*y+z=0,x+2*y+z=0],t);
```

"Плоскости совпадают"

Компьютерное моделирование взаимного расположения прямых в пространстве

Создадим теперь в библиотеке `Anal_Geom` программную процедуру `graph3D(M1,q1,M2,q2,a,b,A,B,c1,c2,c3)` полного исследования и геометрического моделирования взаимного расположения двух прямых, заданных парой опорных точек M_1, M_2 и парой направляющих векторов $\mathbf{q}_1, \mathbf{q}_2$ (`a,b` - коэффициенты, `A,B` – обозначение прямых, `c1,c2,c3` – цветовые опции прямых и их общего перпендикуляра, если таковой существует):

```
>AnalGeo[d]:=proc(M,q,t)local ddd:
ddd:=linalg[matadd](M,q,1,t): end proc:
AnalGeo[dAB]:=proc(A,B,t) local AB:
AB:=linalg[matadd](A,B,-1,1):
linalg[matadd](A,AB,1,t):end proc:
AnalGeo[graph_d]:=proc(M,q,a,b,c) local dd,t:
dd:=(t)->d(M,q,t):
plots[spacecurve](dd(t),t=a..b,thickness=2,
```

```
color=c,scaling=CONSTRAINED):
end proc:

>AnalGeo[TT]:=proc(M1,q1,t1,M2,q2,t2) local delta_d,Eq1,Eq2:
delta_d:=linalg[matadd](d(M1,q1,t1),d(M2,q2,t2),1,-1):
Eq1:=linalg[innerprod](q1,delta_d):
Eq2:=linalg[innerprod](q2,delta_d):
solve({Eq1,Eq2},{t1,t2}):end proc:

>AnalGeo[M1_M2]:=proc(M1,q1,M2,q2)
local t1,t2,delta_d,Eq1,Eq2,t1t2:
delta_d:=linalg[matadd](d(M1,q1,t1),d(M2,q2,t2),1,-1):
Eq1:=linalg[innerprod](q1,delta_d):
Eq2:=linalg[innerprod](q2,delta_d):
t1t2:=solve({Eq1,Eq2},{t1,t2}):
[subs(t1t2,d(M1,q1,t1)),subs(t1t2,d(M2,q2,t2))]:
end proc:

>AnalGeo[d_cross]:=proc(M1,q1,M2,q2,t) local A,B:
A:=M1_M2(M1,q1,M2,q2)[1]:
B:=M1_M2(M1,q1,M2,q2)[2]:
dAB(A,B,t):end proc:

>AnalGeo[cross_par]:=proc(M1,q1,M2,q2) local a,b:
a:=M1_M2(M1,q1,M2,q2)[1]:
b:=M1_M2(M1,q1,M2,q2)[2]:
[M1_M2(M1,q1,M2,q2)[1],M1_M2(M1,q1,M2,q2)[2],
linalg[matadd](a,b,-1,1)]:
end proc:

>AnalGeo[lincomb]:=proc(a,b,lambda,mu) local n,i:
n:=nops(a):
[seq(lambda*a[i]+mu*b[i],i=1..n)]:
end proc:
>AnalGeo[scalar_prod]:=proc(a,b)local n,i:
n:=nops(a):
simplify(sum(a[i]*b[i],i=1..n)):
end proc:
```

```
>AnalGeo[graph3D]:=proc(M1,q1,M2,q2,a,b,A,B,c1,c2,c3)
local gd1,gd2,
M3,MM3,m3,GM3,q3,gdcr,M4,MM4,m4,GM4,GM5,GM6,Ra1,Ra2,Rb1,Rb2,
x_min,x_max,y_min,y_max,z_min,z_max:
Ra1:=lincomb(M1,q1,1,a):
Ra2:=lincomb(M2,q2,1,a):
Rb1:=lincomb(M1,q1,1,b):
Rb2:=lincomb(M2,q2,1,b):
x_min:=min(Ra1[1],Ra2[1],Rb1[1],Rb2[1]):
y_min:=min(Ra1[2],Ra2[2],Rb1[2],Rb2[2]):
z_min:=min(Ra1[3],Ra2[3],Rb1[3],Rb2[3]):
x_max:=max(Ra1[1],Ra2[1],Rb1[1],Rb2[1]):
y_max:=max(Ra1[2],Ra2[2],Rb1[2],Rb2[2]):
z_max:=max(Ra1[3],Ra2[3],Rb1[3],Rb2[3]):
gd1:=graph_d(M1,q1,a,b,c1):
gd2:=graph_d(M2,q2,a,b,c2):
M3:=cross_par(M1,q1,M2,q2)[1]:
M4:=cross_par(M1,q1,M2,q2)[2]:
MM3:=convert(M3,list):
MM4:=convert(M4,list):
m3:=convert(M(op(MM3)),name):
m4:=convert(N(op(MM4)),name):
q3:=cross_par(M1,q1,M2,q2)[3]:
gdcr:=graph_d(M3,q3,0,1,c3):
GM3:=plots[textplot3d]([op(MM3),m3],align = {ABOVE,RIGHT},
color=NAVY,font=[TIMES,ROMAN,8]);
GM4:=plots[textplot3d]([op(MM4),m4],align = {ABOVE,RIGHT},
color=NAVY,font=[TIMES,ROMAN,8]);
GM5:=plots[textplot3d]([M1[1]+0.2,M1[2]+0.2,M1[3]+0.2,`A`],
align = {ABOVE,LEFT},color=c1,font=[TIMES,ROMAN,10]);
GM6:=plots[textplot3d]([M2[1]+0.2,M2[2]+0.2,M2[3]+0.2,`B`],
align = {BELOW,RIGHT},color=c2,font=[TIMES,ROMAN,10]);
plots[display](gd1,gd2,gdcr,GM3,GM4,GM5,GM6,axes=BOXED,
labels=[X,Y,Z],title=
`Скрещивающиеся прямые a и b;
прямая MN - кратчайший перпендикуляр`,
view=[x_min..x_max,y_min..y_max,z_min..z_max]):
end proc:
```

Приведем пример ввода и исполнения программы исследования взаимного расположения двух прямых (Рис. Рис.II.14):

```
> graph3D([0,1,-2],[-2,0,1],[-1,-1,2],[1,2,-1],-2,2,a,b,
blue,COLOR(RGB,0,0.5,0.5),black);
```

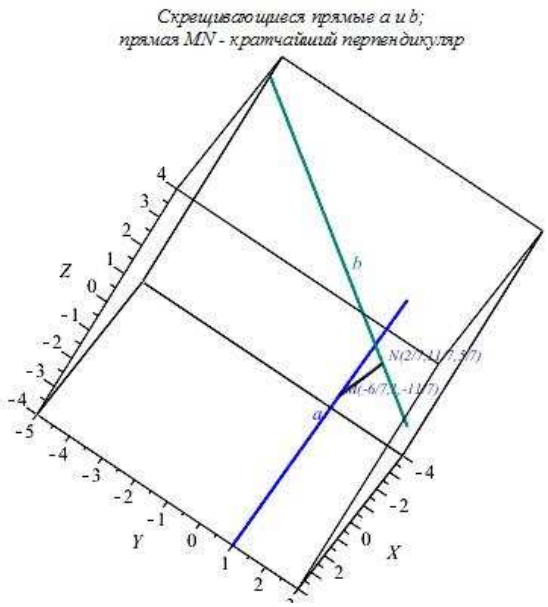


Рис. Рис.II.14. Исполнение команды
graph3D([0,1,-2],[-2,0,1],[-1,-1,2],[1,2,-1],-2,2,a,b,
blue,COLOR(RGB,0,0.5,0.5),black)

Как видно, программная процедура изображает прямые с их названиями, находит кратчайший перпендикуляр и точки его пересечения с данными прямыми. С помощью опции `COLOR(RGB,0,0.5,0.5)` мы придали прямой `b` цвет морской волны, причем сделали его более насыщенным.

II.3 Приведение уравнений кривых второго порядка к каноническому виду

II.3.1 Математическая модель теории кривых второго порядка

Общее уравнение кривой второго порядка и классификация кривых второго порядка

Математическую теорию кривых второго порядка можно найти практически в любых учебниках по аналитической геометрии.

Определение OII.1. *Линия, определяемая в прямоугольной системе координат уравнением*

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0, \tag{II.20}$$

где A, B, C одновременно не обращаются в нуль, называется кривой второго порядка, а уравнение (II.20) — ее общим уравнением.

Совершенно очевидно, что всякая окружность $(x-a)^2+(y-b)^2 = R^2$ есть кривая второго порядка (II.20) при $A = C$, $B = 0$, так как после возведения в квадрат имеем: $x^2 + y^2 - 2ax - 2by + a^2 + b^2 - R^2 = 0$.

Обратно, если в (II.20)
$A = C$, $B = 0$, $\dfrac{F}{A} < 0$, то (II.20) есть уравнение окружности. В самом деле, умножив (II.20) на $\dfrac{1}{A}$, получим уравнение:

$$x^2 + y^2 - 2ax - 2by - c = 0 \left(a = -\frac{D}{A},\ b = -\frac{E}{A},\ c = -\frac{F}{a} > 0\right),$$

которое может быть записано в виде

$$(x - a)^2 + (y - b)^2 = c + a^2 + b^2.$$

Нашей целью является выбор такой новой прямоугольной системы координат
$\Re\{O'; \vec{i}\,', \vec{j}\,', \vec{k}\,'\}$, относительно которой уравнение кривой (II.20) имеет наименьшее число числовых параметров (каноническое уравнение). Переход осуществляется по формулам движения

$$\left.\begin{aligned} x &= x' \cos\alpha - y' \sin\alpha + a \\ y &= x' \sin\alpha + y' \cos\alpha + b \end{aligned}\right\}. \tag{II.21}$$

Подставив вместо x и y в (II.20) их выражения (II.21), получим

$$A(x'^{\,2}\cos^2\alpha + y'^{\,2}\sin^2\alpha + a^2 - 2x'y'\cos\alpha\sin\alpha + 2ax'\cos\alpha - 2ay'\sin\alpha) +$$
$$+2B(x'^{\,2}\cos\alpha\sin\alpha + x'y'\cos^2\alpha + bx'\cos\alpha - x'y'\sin^2\alpha - y'^{\,2}\sin\alpha\cos\alpha - by'\sin\alpha +$$
$$+ax'\sin\alpha + ay'\cos\alpha + ab) + C(x'^{\,2}\sin^2\alpha + y'^{\,2}\cos^2\alpha + b^2 + 2x'y'\cos\alpha\sin\alpha + 2bx'\sin\alpha +$$
$$+2by'\cos\alpha) + 2D(x'\cos\alpha - y'\sin\alpha + \alpha) + 2E(x'\sin\alpha + y'\cos\alpha + b) + F = 0.$$

Приведя подобные члены, имеем

$$A'x'^{\,2} + 2B'x'y' + C'y'^{\,2} + 2D'x' + 2E'y' + F' = 0, \tag{II.22}$$

где новые коэффициенты выражаются следующим образом:

$$\left.\begin{aligned}
A' &= A\cos^2\alpha + 2B\cos\alpha\sin\alpha + C\sin^2\alpha \\
B' &= B\cos 2\alpha + \frac{1}{2}(C-A)\sin 2\alpha \\
C' &= A\sin^2\alpha - 2B\sin\alpha\cos\alpha + C\cos^2\alpha \\
D' &= a(A\cos\alpha + B\sin\alpha) + b(B\cos\alpha + C\sin\alpha) + D\cos\alpha + E\sin\alpha \\
E' &= a(b\cos\alpha - A\sin\alpha) + b(C\cos\alpha - B\sin\alpha) - D\sin\alpha + E\cos\alpha \\
F' &= Aa^2 + 2Bab + Cb^2 + 2Da + 2Eb + F
\end{aligned}\right\}. \tag{II.23}$$

Отметим следующий факт:

> Определитель $\Delta = \begin{vmatrix} A & B \\ B & C \end{vmatrix}$, составленный из коэффициентов при второй степени есть инвариант относительно преобразований координат (II.21).

В самом деле, подставив в выражение $\Delta' = A'C' - B'^2$ значения A', B', C' из (II.23), получим

$$\Delta' = A'C' - B'^2 = (AC-B^2)\sin^4\alpha + (AC-B^2)\cos^4\alpha + 2(AC-B^2)\sin^2\alpha\cos^2\alpha =$$

$$= (AC-B^2)(\sin^2\alpha + \cos^2\alpha)^2 = \Delta. \tag{II.24}$$

Если в уравнении $B \neq 0$, то в преобразованиях (II.21) угол α подберем таким, чтобы коэффициент B' в (II.22) обратился в нуль. Для этого достаточно, чтобы

$$B\cos 2\alpha + \frac{1}{2}(C-A)\sin 2\alpha = 0,$$

т.е.

$$\operatorname{ctg} 2\alpha = \frac{1}{2B}(A-C). \tag{II.25}$$

Распорядимся сейчас выбором параметров a и b для упрощения уравнения (II.22), где $B' = 0$. С этой целью дальнейшее рассмотрение разобьем на два случая в зависимости от того, отличен от нуля определитель $\Delta = AC - B^2$ в уравнении (II.20) или он равен нулю.

(A) Пусть $\Delta = AC - B^2 \neq 0$. Заметив, что определитель, составленный из коэффициентов при a и b для D' и E' в (II.23)

$$\begin{vmatrix} A\cos\alpha + B\sin\alpha & B\cos\alpha + C\sin\alpha \\ B\cos\alpha - A\sin\alpha & C\cos\alpha - B\sin\alpha \end{vmatrix} = AC - B^2 \neq 0, \tag{II.26}$$

выберем a и b таким образом, чтобы $D' = 0$ и $E' = 0$. Для этого, согласно (II.23), необходимо решить систему уравнений относительно неизвестных a и b (значение α фиксировано равенством (II.25))

$$\left.\begin{aligned} a(A\cos\alpha + B\sin\alpha) + b(B\cos\alpha + C\sin\alpha) &= -D\cos\alpha - E\sin\alpha \\ a(B\cos\alpha - A\sin\alpha) + b(C\cos\alpha - B\sin\alpha) &= D\sin\alpha - E\cos\alpha \end{aligned}\right\}.$$

В силу (II.26) для a и b по формулам Крамера имеем единственное решение.

Подставив найденные значения α, a, b, в A', C', F' из (II.23), мы получим в штрихованной системе координат с началом в точке $O'(a;b)$ следующее уравнение:

$$px'^2 + qy'^2 = r. \tag{II.27}$$

Покажем, что начало координат $O'(a;b)$ является центром симметрии кривой (II.27), а штрихованные оси координат — осями симметрии. В самом деле, если в уравнении (II.27) вместо координаты x' подставить $-x'$, то уравнение не изменится. Не изменится оно и после замены y' на $-y'$. Это значит, что точка O' (пересечение осей $O'x'$ и $O'y'$) есть центр симметрии кривой (II.27). Линию второго порядка, имеющую единственный центр симметрии называют *центральной*, остальные носят название нецентральных.

Если $r \neq 0$, то центральная кривая называется *невырожденной*, при $r = 0$ — *вырожденной*.

(*A*.1) *Невырожденные центральные кривые.*Представив уравнение (II.27) в виде

$$\frac{x^2}{\left(\frac{r}{p}\right)} + \frac{y^2}{\left(\frac{r}{q}\right)} = 1$$

(ради упрощения записи штрихи у переменных опустим) и обозначив

$$\frac{r}{p} = \epsilon_1 a^2, \quad \frac{r}{q} = \epsilon_2 b^2 \quad (a, b > 0, \quad \epsilon_1 \pm 1, \quad \epsilon_2 = \pm 1),$$

получим

$$\frac{x^2}{\epsilon_1 a^2} + \frac{y^2}{\epsilon_2 b^2} = 1. \tag{II.28}$$

При $\epsilon_1 = 1$, $\epsilon_2 = 1$ имеем *каноническое уравнение эллипса*

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1. \tag{II.29}$$

Когда $a = b$, эллипс представляет собой окружность. При $\epsilon_1 = 1$, $\epsilon_2 = -1$ имеем *каноническое уравнение гиперболы*

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1. \tag{II.30}$$

При $\epsilon_1 = -1$, $\epsilon_2 = 1$ опять имеем уравнение гиперболы, в котором по сравнению с II.30 роль переменной x играет y и наоборот. Этот случай принципиально не отличается от предыдущего.

При $\epsilon_1 = -1$, $\epsilon_2 = -1$ вещественных точек не существует (*мнимый эллипс*). Очевидно, что первый и четвертый случай соответствуют ситуации, когда $\Delta > 0$, второй и третий, когда $\Delta < 0$.

(A.2) *Вырожденные центральные кривые.* При $r = 0$ уравнение II.27 записывается в виде

$$\epsilon_1 \frac{x^2}{a^2} + \epsilon_2 \frac{y^2}{b^2} = 0 \quad \left(\epsilon_1 a^2 = \frac{1}{p}, \quad \epsilon_2 b^2 = \frac{1}{q} \right). \tag{II.31}$$

При $\epsilon_1 = 1$, $\epsilon_2 = 1\,(\Delta > 0)$ имеем одну точку $O'(0,0)$. При $\epsilon_1 = 1$, $\epsilon_2 = -1\,(\Delta < 0)$ левая часть II.31 распадается на два сомножителя

$$\left(\frac{x}{a} + \frac{y}{b} \right) \left(\frac{x}{a} - \frac{y}{b} \right) = 0.$$

Поэтому имеем пару пересекающихся прямых

$$\frac{x}{a} + \frac{y}{b} = 0, \quad \frac{x}{a} - \frac{y}{b} = 0 \tag{II.32}$$

При $\epsilon_1 = -1$, $\epsilon_2 = 1\,(\Delta < 0)$ ситуация та же самая. Этот случай ничем принципиальным от предыдущего не отличается.

Наконец, при $\epsilon_1 = -1$, $\epsilon_2 = -1\,(\Delta > 0)$ возвращаемся к первой ситуации.

(B). Пусть $\Delta = AC - B^2 = 0$. Взяв $a = 0$, $b = 0$ в (II.31), после выбора угла поворота осей, α, в виде (II.25) мы получим, что $B' = 0$. Из - за инвариантности Δ имеем $\Delta' = A'C' = 0$. Пусть $A' = 0$ (если $C' = 0$, то все рассуждения повторяются буквально, только роль x будет играть y), и, следовательно, $C' \neq 0$. Уравнение (II.22) принимает после умножения на $\dfrac{1}{C'}$ следующий вид:

$$y'^{\,2} + 2px' + 2qy' + r = 0. \tag{II.33}$$

После переноса начала координат в точку

$$O'' \left(\frac{q^2 - r}{2p}; -q \right) \quad x'' = x' + \frac{r - q^2}{2p}, \quad y'' = y' + q$$

(случай $p = 0$ будет рассмотрен ниже) имеем

$$y''^{\,2} = 2px''. \tag{II.34}$$

Получили *каноническое уравнение нецентральной невырожденной* кривой второго порядка — *параболы.*

Если в (II.33) $p = 0$, то после переноса начала координат в точку $O''(0;\ -q)$: $x'' = x'$, $y'' = y' + q$ имеем *каноническое уравнение вырожденной нецентральной кривой второго порядка*

$$(y'')^2 = \epsilon a^2 \quad (\epsilon a^2 = q^2 - r,\ \epsilon = \pm 1). \tag{II.35}$$

При $\epsilon = 1$ уравнение (II.25) описывает *пару параллельных* прямых (при $a = 0$ *слившихся*), при $\epsilon = -1$ — *пару мнимых параллельных прямых.*

На этом классификация всех возможных алгебраических линий второго порядка закончена.

II.3.2 Исследование формы и свойств эллипса

Перейдем к рассмотрению свойств и установлению формы *эллипса* (II.29). При этом мы будем предполагать, что $b < a$ (при $b > a$ роли переменных x и y меняются местами), и называть a — *большой полуосью* эллипса, b — *малой полуосью* эллипса.

Введем параметр $c > 0$, определив его из равенства $c^2 = a^2 - b^2$, и рассмотрим на оси Ox две точки $F_1(-c; 0)$ и $F_2(c; 0)$ так, что начало координат делит отрезок F_1F_2 пополам. Вычислим сейчас расстояния r_1 и r_2 от точек F_1 и F_2 до произвольной точки эллипса $M(x; y)$: $r_1 = \sqrt{(x+c)^2 + y^2}$, $r_2 = \sqrt{(x-c)^2 + y^2}$. Из уравнения (II.29), подставив вместо b^2 его выражение $b^2 = a^2 - c^2$, получим

$$y^2 = a^2 - c^2 - x^2 + \frac{c^2}{a^2}x^2$$

и внесем его под корни для r_1 и r_2

$$\left.\begin{aligned} r_1 &= \sqrt{2cx + a^2 + \frac{c^2}{a^2}x^2} = \sqrt{\left(a + \frac{c}{a}x\right)^2} = \left|a + \frac{c}{a}x\right| \\ r_2 &= \sqrt{-2cx + a^2 + \frac{c^2}{a^2}x^2} = \sqrt{\left(a - \frac{c}{a}x\right)^2} = \left|a - \frac{c}{a}x\right| \end{aligned}\right\}.$$

Из уравнения (II.29) следует, что $|x| \le a$, и так как $c < a$, то $\left|\frac{c}{a}x\right| < a$. Таким образом, $a + \frac{c}{a}x > 0$ и $a - \frac{c}{a}x > 0$. Поэтому для r_1 и r_2 окончательно имеем

$$r_1 = a + ex, \quad r_2 = a - ex, \tag{II.36}$$

где параметр $e = \frac{c}{a} < 1$ называют *эксцентриситетом* эллипса. Складывая равенства (II.36), получим, что

$$r_1 + r_2 = 2a$$

для *любой* точки эллипса. Это свойство позволяет определить эллипс таким образом:

> Эллипс — это геометрическое место точек плоскости, сумма расстояний от которых до двух данных точек F_1 и F_2 (называемых фокусами эллипса) есть величина постоянная (большая чем расстояние между фокусами).

Числа r_1 и r_2 из (II.36) часто называют *фокальными радиусами* эллипса.

Исследуем форму эллипса по его каноническому уравнению. Нами уже установлено, что оси координат являются осями симметрии, а начало координат — центром симметрии. Поскольку $\frac{x^2}{a^2} \leq 1$ и $\frac{y^2}{b^2} \leq 1$, то эллипс расположен в прямоугольнике $|x| \leq a$, $|y| \leq b$.

Для исследования формы эллипса в силу симметрии достаточно рассмотреть ту его часть, что лежит в первой координатной четверти:

$$y = \frac{b}{a}\sqrt{a^2 - x^2}.$$

Получив график этой линии и достроив ее симметрично в остальных четвертях, получим линию эллипса (Рис.II.15)

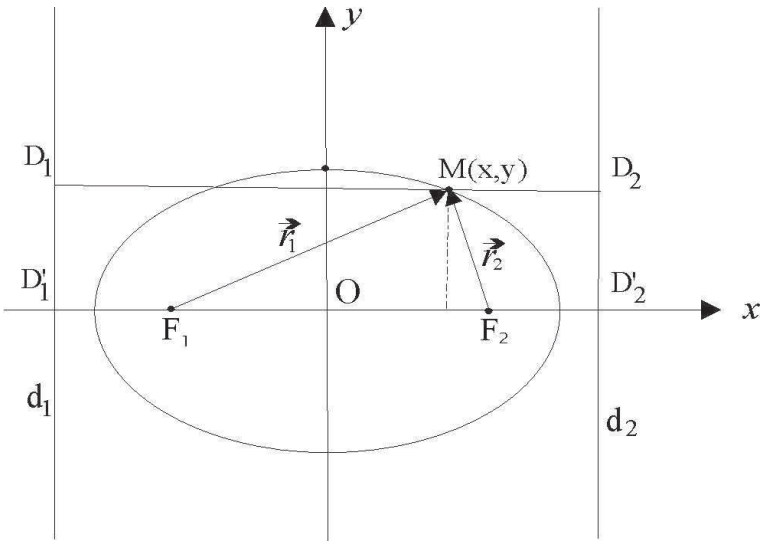


Рис.II.15 *Эллипс.*

При изучении эллипса особую роль играют две прямые, перпендикулярные к оси Ox, с уравнениями $x = \pm\frac{a}{e}$ (Рис.II.15) — *директрисы* эллипса.

Для эллипса
$e < 1$, $\dfrac{a}{e} > a$, и, следовательно, директрисы расположены вне эллипса. Для директрис имеет место теорема, которую можно взять за новое определение эллипса.

Теорема TII.1. *Отношение расстояния от любой точки эллипса до фокуса к расстоянию ее до соответствующей директрисы есть величина постоянная, равная эксцентриситету эллипса.*

Доказательство: $\langle\langle$ Действительно, пусть $M(x;y)$ — точка эллипса, d_1 и d_2 — расстояния до соответствующих дирректрисы. В силу симметрии достаточно доказать теорему для одного из фокусов (например, для F_2).

Точка M' имеет координаты $(x;0)$, а $D_2' - \left(\dfrac{a}{e};0\right)$. Поэтому $d_2 = |\overrightarrow{M'D_2'}| = \dfrac{a}{e} - x$. Поскольку $r_2 = a - ex$, то

$$\frac{r_2}{d_2} = \frac{(a-ex)e}{a-ex} = e.$$

$\rangle\rangle$

В заключении остановимся на параметрическом уравнении эллипса. С этой целью рассмотрим окружность $x^2 + y^2 = a^2$ и произведем точечное сжатие плоскости вдоль ординат точек

$$x = X \qquad y = \frac{a}{b}Y. \tag{II.37}$$

Подставив в уравнение окружности, получим уравнение эллипса

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} = 1.$$

Пусть окружность задана в параметрической форме:

$$x = a\cos t, \quad y = a\sin t, \quad (t \in [0, 2\pi]).$$

Из (II.37) находим $X = a\cos t$, $Y = \dfrac{b}{a}y = b\cos t$ $(t \in [0, 2\pi])$. Поэтому параметрическое уравнение эллипса с полуосями a и b имеют вид

$$\left.\begin{aligned} x &= a\cos t \\ y &= b\sin t \end{aligned}\right\} \quad (0 \le t < 2\pi). \tag{II.38}$$

В следующем разделе мы изучим гиперболу и параболу.

II.3.3 Исследование формы и свойств гиперболы

Перейдем к рассмотрению геометрических свойств и формы гиперболы

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \quad (a, b > 0). \tag{II.39}$$

Введем параметр $c > 0$ с помощью равенства $c^2 = a^2 + b^2$, так что $c > a$, и рассмотрим на оси Ox две точки $F_1(-c; 0)$ и $F_2(c; 0)$, которые назовем *левым* и *правым фокусами* гиперболы. Вычислим фокальные расстояния $|\overrightarrow{F_1M}|$, $|\overrightarrow{F_2M}|$ до произвольной точки гиперболы $M(x; y)$: $|\overrightarrow{F_1M}| = r_1 = \sqrt{(x+c)^2 + y^2}$, $|\overrightarrow{F_2M}| = r_2 = \sqrt{(x-c)^2 + y^2}$. Из (II.39) находим

$$y^2 = \frac{c^2}{a^2}x^2 - x^2 - c^2 + a^2$$

и подставляем в выражение для r_1 и r_2. Имеем

$$\left.\begin{aligned} r_1 &= \sqrt{\left(a + \frac{c}{a}x\right)^2} = |a + ex| \\ r_2 &= \sqrt{\left(a - \frac{c}{a}x\right)^2} = |a - ex| \end{aligned}\right\} \tag{II.40}$$

где $e = \dfrac{c}{a}$ — параметр, называемый *эксцентриситетом гиперболы*. Заметим, что для гиперболы $e > 1$, так как $c > a$.

Исследуем различные возможности, представляемые равенствами (II.40). Из (II.39) следует, что $\dfrac{x^2}{a^2} \geq 1$, т.е. $|x| \geq a$. Поэтому в зависимости от того, лежит ли $M(x; y)$ в правой полуплоскости ($x > 0$), или в левой ($x < 0$) выражения для (II.40) различные.

Если $x > 0$, то $a + ex > 0$ и $a - ex < 0$, так как, $e > 1$, $x > a$, $ex > a$. Поэтому *в правой полуплоскости* выполнено

$$r_1 = a + ex, \quad r_2 = -a + ex. \tag{II.41}$$

Если $x < 0$, то $a + ex < 0$, так как, $|ex| > a$ и $a - ex > 0$. Поэтому в *в левой полуплоскости* выполнено

$$r_1 = -a - ex, \quad r_2 = a - ex. \tag{II.42}$$

Из этих рассуждений вытекает, что гипербола состоит из двух симметричных ветвей, расположенных соответственно в правой и левой полуплоскостях. Из (II.41) и (II.42) для обеих ветвей выводим инвариантное равенство

$$|r_1 - r_2| = 2a. \tag{II.43}$$

На основании (II.43) можно дать новое определение гиперболы:

> Гипербола есть геометрическое место точек плоскости, модуль разности расстояний от которых до двух фиксированных точек F_1, F_2, называемых фокусами гиперболы есть величина постоянная (не равная нулю и меньшая расстояния между фокусами) .

При установлении формы гиперболы отметим, что оси Ox и Oy являются *осями симметрии*, а их пересечение — *центром симметрии* (центр гиперболы). Поскольку $\frac{x^2}{a^2} \geq 1$ ($|x| \geq a$), то ветви гиперболы лежат на плоскости вне полосы $-a < x < a$. Точки пересечения ветвей с осью Ox ($y = 0$, $A_1(-a; 0)$) называются *вершинами* гиперболы. Для гиперболы, заданной уравнением (II.39) ось Ox называется *действительной осью* гиперболы, а ось Oy — *мнимой осью* гиперболы (пересечения нет из - за равенства $-\frac{y^2}{b^2} = 1$). В силу симметрий достаточно исследовать форму гиперболы в первой четверти, $y = \frac{b}{a}\sqrt{x^2 - a^2}$, а затем распространить на всю плоскость. Наряду с указанной ветвью гиперболы целесообразно рассмотреть луч, исходящий из начала координат, с уравнением $Y = \frac{b}{a}x$.

Пусть M — точка гиперболы с абсциссой x, N — точка, указанной прямой с той же абсциссой. Для разности ординат имеем

$$Y - y = \frac{b}{a}(x - \sqrt{x^2 - a^2}).$$

При неограниченном возрастании x данная разность монотонно стремится к нулю, так как

$$\lim_{x\to\infty}(Y - y) = \frac{b}{a}\lim_{x\to\infty}(x - \sqrt{x^2 - a^2}) = \frac{b}{a}\lim_{x\to\infty}\frac{x^2 - x^2 + a^2}{(x + \sqrt{x^2 - a^2})} = 0.$$

Поэтому прямые $y = \pm\frac{b}{a}x$ являются *асимптотами* гиперболы

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1.$$

Для построения асимптот строим прямоугольник со сторонами $x = \pm a$, $y = \pm b$. Асимптоты — прямые, содержащие диагонали прямоугольника. При построении гиперболы удобно построить сначала асимптоты, а затем ветви гиперболы (Рис.II.16).

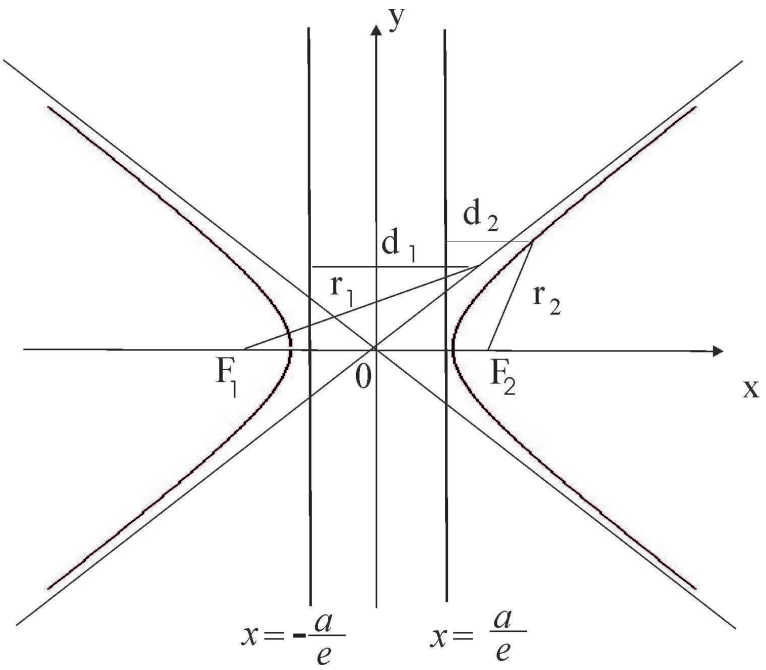


Рис.II.16 *Гипербола.*

Для гиперболы вводятся две прямые, перпендикулярные к оси Ox, с уравнениями $x = \pm\dfrac{a}{e}$, и называемые *директрисами гиперболы.* В силу неравенства $e > 1$ директрисы лежат между вершинами гиперболы. Так же, как и для эллипса, имеет место *утверждение:*

$$\frac{r_1}{d_1} = \frac{r_2}{d_2} = e,$$

где d_1 и d_2 — расстояния от точки гиперболы до соответствующих директрис (Рис.II.16). Доказательство совершенно аналогично тому, что проведено для эллипса, и поэтому мы приводить его не будем. Приведем лишь формулировку теоремы, которая является фактически еще одним определением гиперболы.

Теорема TII.2. *Отношение расстояния от любой точки гиперболы до фокуса к расстоянию ее до соответствующей директрисы есть величина постоянная, равная эксцентриситету гиперболы $e > 1$.*

Отметим, что гипербола

$$\frac{y^2}{b^2} - \frac{x^2}{a^2} = 1.$$

называется *сопряженной* по отношению к гиперболе (II.39). Действительной осью сопряженной гиперболы является ось Oy, асимптоты остаются прежними.

II.3.4 Исследование формы и свойств параболы

Перейдем к установлению геометрических свойств и вида формы параболы

$$y^2 = 2px \quad (p > 0). \tag{II.44}$$

Прежде всего отметим, что кривая лежит в правой полуплоскости ($x \geq 0$) и проходит через начало координат. Рассмотрим точку $F\left(\frac{p}{2}; 0\right)$, называемую *фокусом параболы*, и прямую, перпендикулярную к оси Ox, с уравнением $x = -\frac{p}{2}$, называемую *директрисой параболы*. Пусть $M(x; y)$ — произвольная точка параболы. Вычислим расстояние

$$\left|\overrightarrow{FM}\right| = r = \sqrt{\left(x - \frac{p}{2}\right)^2 + y^2},$$

подставив y^2 из соотношения (II.44). Получим

$$r = \sqrt{\left(x + \frac{p}{2}\right)^2} = \left|x + \frac{p}{2}\right| = x + \frac{p}{2}. \tag{II.45}$$

Расстояние d от M до директрисы $x = -\frac{p}{2}$ также равно $x + \frac{p}{2}$. Поэтому для параболы $\frac{r}{d} = 1$. Таким образом:

> Парабола есть геометрическое место точек, для которых расстояние до фиксированной точки F (фокуса параболы) равно расстоянию до фиксированной прямой (директрисы параболы).

Так как для эллипса и гиперболы отношение $\frac{r}{d}$ есть постоянное число, равное e — эксцентриситету, то и для параболы отношение $\frac{r}{d} = e$ — называют *эксцентриситетом*, т.е. для параболы $e = 1$.

Из приведенного рассуждения следует, что эллипс, гипербола и парабола обладают *общим* геометрическим свойством:

Отношение расстояния r от любой точки M каждой из этих кривых до фокуса к расстоянию d от этой точки M до соответствующей директрисы есть величина постоянная, равная эксцентриситету e кривой.

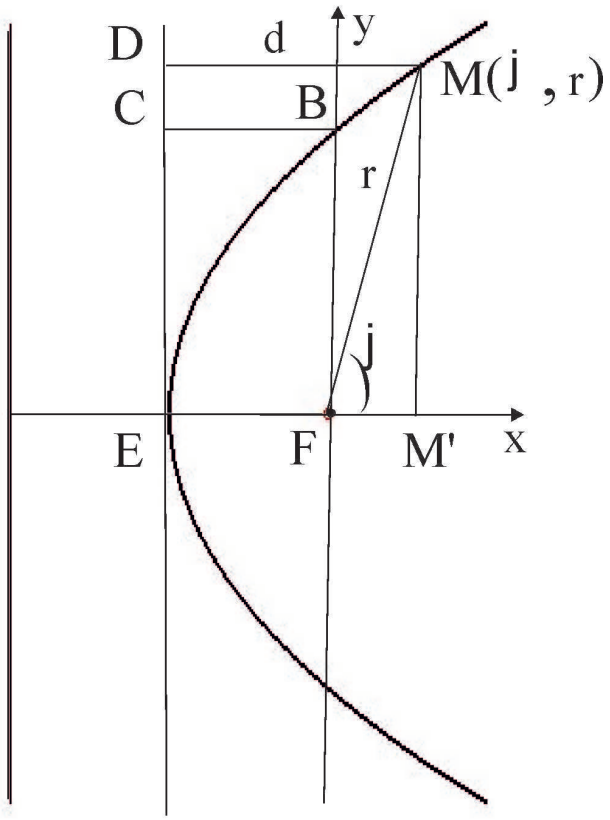


Рис.II.17 *Парабола.*

Форма параболы легко устанавливается с использованием симметрии относительно оси Ox и монотонного возрастания функции $y = \sqrt{2px}$ ((Рис.II.17))

II.3.5 Полярные уравнения кривых второго порядка

Для вывода единого уравнения эллипса, гиперболы и параболы в *полярной системе координат* воспользуемся установленным единым для этих кривых свойством $\frac{r}{d} = e$.

Пусть задана какая - либо из перечисленных кривых. Поместим полюс полярной системы координат в фокус F, а полярную ось — по перпендикуляру от директрисы к фокусу (Рис.II.17).

Пусть B — точка пересечения кривой с перпендикуляром к полярной оси, исходящим из фокуса F, а p — длина вектора $\overrightarrow{FB}$, называемая *фокальным параметром*. Согласно свойству $\frac{r}{d} = e$ имеем, что

$$\frac{p}{\left|\overrightarrow{CB}\right|} = \frac{p}{\left|\overrightarrow{EF}\right|} = e.$$

Поэтому $\left|\overrightarrow{EF}\right| = \frac{p}{e}$, а также

$$d = \left|\overrightarrow{DM}\right| = \left|\overrightarrow{EN}\right| = \left|\overrightarrow{EF}\right| + r\cos\varphi = \frac{p}{e} + r\cos\varphi.$$

Подставляя данное выражение в отношение $\frac{r}{d} = e$, получим

$$\frac{r}{\frac{p}{e} + r\cos\varphi} = e.$$

откуда следует полярное уравнение для рассматриваемых кривых второго порядка ($e < 1$ — эллипс, $e > 1$ — гипербола, $e = 1$ — парабола):

$$r = \frac{p}{1 - e\cos\varphi}. \tag{II.46}$$

Отметим, что при $e > 1$ из - за условия $r > 0$ уравнение (II.46) описывает только одну ветвь гиперболы, внутри которой лежит фокус.

Эллипс, гиперболу и параболу называют *коническими сечениями*, потому что их можно получить путем сечения плоскостями кругового конуса. Если плоскость пересекает только одну полость конуса и параллельна одной из образующих конуса, то получим параболу. Если секущая плоскость пересекает только одну полость конуса и не параллельна ни одной образующей конуса, то коническое сечение будет эллипсом. Когда секущая плоскость перпендикулярна оси конуса, получаем окружность.

Наконец, если секущая плоскость не проходит через вершину и параллельна оси конуса, то получим гиперболу.

II.4 Компьютерное моделирование теории кривых второго порядка

```
> restart:
```

```
> AnalGeo:=table():
```

II.4.1 Изображение координатных осей

```
>AnalGeo[Axes1]:=proc(X,Y,alpha,X0)
local ttr,x_max,y_max,x_min,
y_min,x_end,y_end,XY,XX,YY,TXT,xx,yy,xx1,
yy1,ddx,ddy,ddd,ii,xgr,ygr:
x_min:=floor(X[1]):
x_max:=ceil(X[2]):
ddx:=x_max-x_min:
y_min:=floor(Y[1]):
y_max:=ceil(Y[2]):
ddy:=y_max-y_min:
ddd:=max([ddx,ddy]):
ttr:=(xx1,yy1)->[xx1*cos(alpha)-yy1*sin(alpha)+X0[1],
xx1*sin(alpha)+yy1*cos(alpha)+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=x_min..x_min+ddd],
thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=y_min..y_min+ddd],
thickness=1,color=black):
xgr:=plots[display](plot([op(ttr(x_min,xx)),xx=0..0.015*ddd],
color=black,thickness=1),
seq(plot([op(ttr(x_min+ii,xx)),xx=0..0.015*ddd],color=black,
thickness=1),ii=1..ddd)):
ygr:=plots[display](plot([op(ttr(xx,y_min)),xx=0..0.015*ddd],
color=black,thickness=1),
seq(plot([op(ttr(xx,y_min+ii)),xx=0..0.015*ddd],color=black,
thickness=1),ii=0..ddd)):
x_end:=x_min+ddd:y_end:=y_min+ddd:
TXT:=plots[textplot]({[op(ttr(0.015*ddd,-0.015*ddd)),`O`],
[op(ttr(x_end,-0.02*ddd)),`x`],
[op(ttr(0.05*ddd,y_end)),`y`]},align={below,right}):
plots[display](XX,YY,xgr,ygr,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:
```

```
AnalGeo[Axes0]:=proc(X,Y,alpha,X0) local ttr,XY,XX,
YY,TXT,xx,yy,xx1,yy1,ddx,ddy,ddd:
```

```
ttr:=(xx1,yy1)->[xx1*cos(alpha)-yy1*sin(alpha)+X0[1],
xx1*sin(alpha)+yy1*cos(alpha)+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=X[1]..X[2]],thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=Y[1]..Y[2]],thickness=1,color=black):
ddx:=(X[2]-X[1]):
ddy:=(Y[2]-Y[1]):
ddd:=max([ddx,ddy]):
TXT:=plots[textplot]({[op(ttr(0.015*ddd,-0.015*ddd)),`O1`],
[op(ttr(X[2],-0.015*ddd)),`x1`],
[op(ttr(0.015*ddd,Y[2])),`y1`]},align={below,right}):
plots[display](XX,YY,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:
```

II.4.2 Изображение пар прямых на координатной плоскости

```
>AnalGeo[CrossDir]:=proc(a,b,alpha,X0,c1,c2)
local X,t,Eq1,Eq2,ttr,Eq1_1,Eq2_1,gd1,gd2,M1_1,
M1_2,M2_1,M2_2,X_min,X_max,Y_min,Y_max,CRD,CRD1:
ttr:=(X)->[X[1]*cos(alpha)-X[2]*sin(alpha)+X0[1],
X[1]*sin(alpha)+X[2]*cos(alpha)+X0[2]]:
Eq1:=(a,b,t)->[b*t,a*t]: Eq2:=(a,b,t)->[b*t,-a*t]:
Eq1_1:=(a,b,t,alpha,X0)->ttr(Eq1(a,b,t)):
Eq2_1:=(a,b,t,alpha,X0)->ttr(Eq2(a,b,t)):
M1_1:=Eq1_1(a,b,-1,alpha,X0):
M1_2:=Eq1_1(a,b,1,alpha,X0):
M2_1:=Eq2_1(a,b,-1,alpha,X0):
M2_2:=Eq2_1(a,b,1,alpha,X0):
X_min:=min(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
X_max:=max(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
Y_min:=min(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
Y_max:=max(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
CRD:=AnalGeo[Axes1]([X_min,X_max],[Y_min,Y_max],0,[0,0]):
CRD1:=AnalGeo[Axes0]([X_min-X0[1],X_max-X0[1]],
[Y_min-X0[2],Y_max-X0[2]],alpha,X0):
gd1:=(a,b,alpha,X0)->plot([op(Eq1_1(a,b,t,alpha,X0)),
t=-1..1],thickness=1,color=c1):
gd2:=(a,b,alpha,X0)->plot([op(Eq2_1(a,b,t,alpha,X0)),
t=-1..1],thickness=1,color=c2):
```

```
plots[display](gd1(a,b,alpha,X0),gd2(a,b,alpha,X0),
CRD,CRD1):
end proc:

>AnalGeo[ParalDir]:=proc(a,alpha,X0,c1,c2)
local X,t,Eq1,Eq2,ttr,Eq1_1,Eq2_1,gd1,gd2,M1_1,
M1_2,M2_1,M2_2,X_min,X_max,Y_min,Y_max,CRD,CRD1:
ttr:=(X)->[X[1]*cos(alpha)-X[2]*sin(alpha)+X0[1],
X[1]*sin(alpha)+X[2]*cos(alpha)+X0[2]]:
Eq1:=(a,t)->[t,a]: Eq2:=(a,t)->[t,-a]:
Eq1_1:=(a,t,alpha,X0)->ttr(Eq1(a,t)):
Eq2_1:=(a,t,alpha,X0)->ttr(Eq2(a,t)):
M1_1:=Eq1_1(a,-1,alpha,X0):
M1_2:=Eq1_1(a,1,alpha,X0):
M2_1:=Eq2_1(a,-1,alpha,X0):
M2_2:=Eq2_1(a,1,alpha,X0):
X_min:=min(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
X_max:=max(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
Y_min:=min(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
Y_max:=max(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
CRD:=AnalGeo[Axes1]([X_min,X_max],[Y_min,Y_max],0,[0,0]):
CRD1:=AnalGeo[Axes0]([X_min-X0[1],X_max-X0[1]],
[Y_min-X0[2],Y_max-X0[2]],alpha,X0):
gd1:=(a,alpha,X0)->plot([op(Eq1_1(a,t,alpha,X0)),
t=-1..1],thickness=1,color=c1):
gd2:=(a,alpha,X0)->plot([op(Eq2_1(a,t,alpha,X0)),
t=-1..1],thickness=1,color=c2):
plots[display](gd1(a,alpha,X0),gd2(a,alpha,X0),CRD,CRD1):
end proc:

AnalGeo[Point]:=proc(alpha,X0,c1)local X,ttr,X_min,
X_max,Y_min,Y_max,CRD,CRD1,gp:
ttr:=(X)->[X[1]*cos(alpha)-X[2]*sin(alpha)+X0[1],
X[1]*sin(alpha)+X[2]*cos(alpha)+X0[2]]:
X_min:=min(0,X0[1],-X0[1]):
X_max:=max(0,X0[1],-X0[1]):
Y_min:=min(0,X0[2],-X0[2]):
Y_max:=max(0,X0[2],-X0[2]):
CRD:=AnalGeo[Axes1]([X_min,X_max],[Y_min,Y_max],0,[0,0]):
```

```
CRD1:=AnalGeo[Axes0]([X_min,X_max],[Y_min,Y_max],alpha,X0):
gp:=plots[pointplot](X0,color=c1,symbol=circle,symbolsize=16):
plots[display](CRD,CRD1,gp):
end proc:
```

II.4.3 Изображение фигур второго порядка

```
>AnalGeo[Ell]:=proc(a,b,alpha,X0,c1,c2)
local pp,ee,t,cc,eq2,eq21,X,ttr,
F1,F2,AA1,AA2,BB1,BB2,ells,DD1,DD2,AA12,
BB12,X0A1,X0A2,X0B1,X0B2,
dd,LX,LY,PP,TT,dd1,dd2,gdd1,gdd2:
#####
ttr:=(X)->[X[1]*cos(alpha)-X[2]*sin(alpha)+X0[1],
X[1]*sin(alpha)+X[2]*cos(alpha)+X0[2]]:
if a^2>b^2 then
pp:=b^2/a:ee:=sqrt(a^2-b^2)/a:
cc:=sqrt(a^2-b^2):dd:=evalf(a^2/cc):
else
pp:=a^2/b:ee:=sqrt(b^2-a^2)/b:
cc:=sqrt(b^2-a^2):dd:=evalf(b^2/cc):
end if:
if a^2>b^2 then
F1:=ttr([-cc,0]):F2:=ttr([cc,0]):
else
F1:=ttr([0,-cc]):F2:=ttr([0,cc]):
end if:
AA1:=ttr([-a,0]):AA2:=ttr([a,0]):
BB1:=ttr([0,-b]):BB2:=ttr([0,b]):
AA12:=[AA2[1]-AA1[1],AA2[2]-AA1[2]]:
BB12:=[BB2[1]-BB1[1],BB2[2]-BB1[2]]:
if a^2>b^2 then
DD1:=ttr([-dd,0]):DD2:=ttr([dd,0]):
else
DD1:=ttr([0,-dd]):DD2:=ttr([0,dd]):
end if:
#LX:=AnalGeo[Axes0]([-a,a],[-dd,dd],alpha,X0):
#LY:=AnalGeo[Axes0]([0,0],[-b,b],alpha,X0):
```

```
if a^2>b^2 then
LX:=AnalGeo[Axes0]([-dd,dd],[-b,b],alpha,X0):
#LY:=AnalGeo[Axes1]([-0.2*BB2[1]+1.2*BB1[1],
-0.2*BB1[1]+1.2*BB2[1]],[-0.2*AA2[1]+1.2*AA1[1],
-0.2*AA1[1]+1.2*AA2[1]],alpha,X0):
else
LX:=AnalGeo[Axes0]([-a,a],[-dd,dd],alpha,X0):
#LY:=AnalGeo[Axes1]([-0.2*AA2[1]+1.2*AA1[1],
-0.2*AA1[1]+1.2*AA2[1]],[-0.2*BB2[1]+1.2*BB1[1],
-0.2*BB1[1]+1.2*BB2[1]],alpha,X0):
end if:
LY:=AnalGeo[Axes1]([-dd,dd],[-dd,dd],0,[0,0]):
if a^2>b^2 then
eq2:=(pp,ee,t)->[pp*cos(t)/(1-ee*cos(t))-cc,
pp*sin(t)/(1-ee*cos(t))]:
dd1:=(t)->[DD1[1]+BB12[1]*t,DD1[2]+BB12[2]*t]:
dd2:=(t)->[DD2[1]+BB12[1]*t,DD2[2]+BB12[2]*t]:
else
eq2:=(pp,ee,t)->[pp*sin(t)/(1-ee*cos(t)),
pp*cos(t)/(1-ee*cos(t))-cc]:
dd1:=(t)->[DD1[1]+AA12[1]*t,DD1[2]+AA12[2]*t]:
dd2:=(t)->[DD2[1]+AA12[1]*t,DD2[2]+AA12[2]*t]:
end if:
eq21:=(pp,ee,t,alpha,X0)->ttr(eq2(pp,ee,t)):
ells:=plot([op(eq21(pp,ee,t,alpha,X0)),t=0..2*Pi],
color=c1,thickness=2,scaling=CONSTRAINED,
numpoints=500,axes=normal,scaling=CONSTRAINED):
gdd1:=plot([op(dd1(t)),t=-1/2..1/2],thickness=1,
color=black):
gdd2:=plot([op(dd2(t)),t=-1/2..1/2],thickness=1,
color=black):
TT:=plots[textplot]([[F1[1]+0.5,F1[2],"F1"],
[F2[1]+0.5,F2[2],"F2"],
[AA1[1]+0.3,AA1[2],"A1"],[AA2[1]+0.3,AA2[2],"A2"],
[BB1[1]+0.3,BB1[2],"B1"],[BB2[1]-0.3,BB2[2],"B2"],
[DD1[1]+0.4,DD1[2],"d1"],[DD2[1]+0.4,DD2[2],"d2"]],
title="Э Л Л И П С",font=[TIMES,ROMAN,10],color=c2):
PP:=plots[pointplot]([F1,F2,AA1,AA2,BB1,BB2,X0,DD1,DD2],
```

```
color=black,symbol=circle,symbolsize=14,color=c2):
plots[display](LX,LY,ells,gdd1,gdd2,TT,PP,scaling=CONSTRAINED):
end proc:


AnalGeo[Hyp]:=proc(a,b,q,alpha,X0,c1,c2,d)
local pp,ee,t,cc,eq2,eq21,X,ttr,
F1,F2,AA1,AA2,BB1,BB2,DD1,DD2,AA12,
BB12,X0A1,X0A2,X0B1,X0B2,
x_min,x_max,y_min,y_max,l_x,l_y,f,f_k,
TTXY,dd,LX,LY,PP,TT,dd1,dd2,gdd1,gdd2,
eq1,hypb,eq22,g21,g22,M1,M2,M3,M4,p1,p2,
p3,p4,xt1,xt2,asimp1,asimp2,ppg,PPG,TTit:
ttr:=(X)->[X[1]*cos(alpha)-X[2]*sin(alpha)+
X0[1],X[1]*sin(alpha)+X[2]*cos(alpha)+X0[2]]:
M1:=ttr([-a,-b]):M2:=ttr([a,-b]):M3:=ttr([a,b]):
M4:=ttr([-a,b]):
ppg:=plots[polygonplot]([M1,M2,M3,M4],
color=COLOR(RGB,0.75,0.75,0.75),scaling=CONSTRAINED):
PPG:=plots[textplot]([[M1[1]+0.2,M1[2]+0.2,"M1"],
[M2[1]+0.2,M2[2]+0.2,"M2"],
[M3[1]+0.2,M3[2]+0.2,"M3"],
[M4[1]+0.2,M4[2]+0.2,"M4"]],font=[TIMES,ROMAN,10],
color=c2,scaling=CONSTRAINED):
if q=1 then
pp:=b^2/a:ee:=sqrt(a^2+b^2)/a:
cc:=sqrt(a^2+b^2):dd:=a^2/cc:
else
pp:=a^2/b:ee:=sqrt(b^2+a^2)/b:
cc:=sqrt(b^2+a^2):dd:=b^2/cc:
end if:
if q=1 then
F1:=ttr([-cc,0]):F2:=ttr([cc,0]):
else
F1:=ttr([0,-cc]):F2:=ttr([0,cc]):
end if:
p1:=ttr([-(1+d)*cc,-cc*b*(1+d)/a]):
p2:=ttr([(1+d)*cc,cc*b*(1+d)/a]):
p3:=ttr([(1+d)*cc,-cc*b*(1+d)/a]):
```

```
p4:=ttr([-(1+d)*cc,cc*b*(1+d)/a]):
xt1:=ttr([cc*(1+d),0])[1]:
xt2:=ttr([-cc*(1+d),0])[1]:
asimp1:=plot([p1[1]+(p2[1]-p1[1])*t,
p1[2]+(p2[2]-p1[2])*t,t=0..1],
style=point,symbol=point,color=black,
scaling=CONSTRAINED):
asimp2:=plot([p3[1]+(p4[1]-p3[1])*t,
p3[2]+(p4[2]-p3[2])*t,t=0..1],
style=point,symbol=point,color=black,
scaling=CONSTRAINED):
AA1:=ttr([-a,0]):AA2:=ttr([a,0]):
BB1:=ttr([0,-b]):BB2:=ttr([0,b]):
AA12:=[AA2[1]-AA1[1],AA2[2]-AA1[2]]:
BB12:=[BB2[1]-BB1[1],BB2[2]-BB1[2]]:
x_min:=min(F1[1],F2[1],AA1[1],AA2[1],
BB1[1],BB2[1]):
x_max:=max(F1[1],F2[1],AA1[1],AA2[1],
BB1[1],BB2[1]):
l_x:=x_max-x_min:
y_min:=min(F1[2],F2[2],AA1[2],AA2[2],
BB1[2],BB2[2]):
y_max:=max(F1[2],F2[2],AA1[2],AA2[2],
BB1[2],BB2[2]):l_y:=y_max-y_min:
TTXY:=plots[textplot]([[2*(x_max+l_x*0.1),
-2*(l_y*0.05),"x"],[-2*(l_x*0.05),
2*(l_y*0.1+y_max),"y"]],color=black,
font=[TIMES,ROMAN,10]):
if q=1 then
DD1:=ttr([-dd,0]):DD2:=ttr([dd,0]):
else
DD1:=ttr([0,-dd]):DD2:=ttr([0,dd]):
end if:
#Изображение новых осей координат
LX:=plot([(F2[1]-F1[1])*t+F1[1],(F2[2]-F1[2])*t+F1[2],
t=-0.2..1.2],color=black):
if q=1 then
LY:=plot([(BB2[1]-BB1[1])*t+BB1[1],(BB2[2]-BB1[2])*t+BB1[2],
```

```
t=-1..2],color=black):
else
LY:=plot([(AA2[1]-AA1[1])*t+AA1[1],(AA2[2]-AA1[2])*t+AA1[2],
t=-1..2],color=black):
end if:
if q=1 then
#-------------------------------
#Главная ось OX
#Уравнения левой(eq1) и правой(eq2) ветвей
#?гиперболы в полярных координатах
#-------------------------------
eq1:=(pp,ee,t)->[-pp*cos(t)/(1-ee*cos(t))-cc,
pp*sin(t)/(1-ee*cos(t))]:
eq2:=(pp,ee,t)->[pp*cos(t)/(1-ee*cos(t))+cc,
pp*sin(t)/(1-ee*cos(t))]:
#--------------------------------------
#Уравнения директрисс для случая главной оси OX
#--------------------------------------
dd1:=(t)->[DD1[1]+BB12[1]*t,DD1[2]+BB12[2]*t]:
dd2:=(t)->[DD2[1]+BB12[1]*t,DD2[2]+BB12[2]*t]:
else
#Главная ось OY
#Уравнения нижней(eq1) и верхней(eq2) ветвей
#гиперболы в полярных координатах
eq1:=(pp,ee,t)->[pp*sin(t)/(1-ee*cos(t)),
-pp*cos(t)/(1-ee*cos(t))-cc]:
eq2:=(pp,ee,t)->[pp*sin(t)/(1-ee*cos(t)),
pp*cos(t)/(1-ee*cos(t))+cc]:
#--------------------------------------
#Уравнения директрисс для случая главной оси OY
dd1:=(t)->[DD1[1]+AA12[1]*t,DD1[2]+AA12[2]*t]:
dd2:=(t)->[DD2[1]+AA12[1]*t,DD2[2]+AA12[2]*t]:
end if:
#Преобразованные уравнения левой/нижней(eq21)
#и правой/верхней(eq22) ветвей гиперболы
eq21:=(pp,ee,t,alpha,X0)->ttr(eq2(pp,ee,t)):
eq22:=(pp,ee,t,alpha,X0)->ttr(eq1(pp,ee,t)):
f:=evalf(arctan(b/a),4):
```

```
if q=1 then
f_k:=evalf(arctan(b*sqrt(cc^2*(1+d)^2-a^2)/(a*cc*d))):
else
f_k:=evalf(arctan(a*sqrt(cc^2*(1+d)^2-b^2)/(b*cc*d))):
end if:
g21:=plot([op(eq21(pp,ee,t,alpha,X0)),t=f_k..2*Pi-f_k],
color=c1,thickness=2,
scaling=CONSTRAINED,
numpoints=500,axes=Normal):
g22:=plot([op(eq22(pp,ee,t,alpha,X0)),t=f_k..2*Pi-f_k],
color=c1,thickness=2,
scaling=CONSTRAINED,
numpoints=500,axes=Normal):
hypb:=plots[display](g21,g22):
TT:=plots[textplot]([[F1[1]+0.5,F1[2],"F1"],
[F2[1]+0.5,F2[2],"F2"],
[X0[1]+0.3,X0[2],"O1"],
[X0[1]+(AA2[1]-X0[1])*1.2+0.3,X0[2]+(AA2[2]-X0[2])*1.2,"x1"],
[X0[1]+(BB2[1]-X0[1])*1.2+0.3,X0[2]+
(BB2[2]-X0[2])*1.2+0.3,"y1"],
[AA1[1]+0.3,AA1[2],"A1"],[AA2[1]+0.3,AA2[2],"A2"],
[BB1[1]+0.3,BB1[2],"B1"],[BB2[1]-0.3,BB2[2],"B2"],
[DD1[1]+0.4,DD1[2],"d1"],[DD2[1]+0.4,DD2[2],"d2"]],
title="ГИПЕРБОЛА",font=[TIMES,ROMAN,10],color=c2):
PP:=plots[pointplot]([F1,F2,AA1,AA2,BB1,BB2,X0,DD1,DD2],
color=black,symbol=circle,symbolsize=14,color=c2):
gdd1:=plot([dd1(t)[1],dd1(t)[2],t=-1..1],thickness=1,
color=COLOR(RGB,0,0,0),style=line):
gdd2:=plot([op(dd2(t)),t=-1..1],thickness=1,
color=COLOR(RGB,0,0,0)):
plots[display](LX,LY,PP,TT,TTXY,ppg,PPG,
hypb,asimp1,asimp2,gdd1,gdd2,
scaling=CONSTRAINED,labels=["",""]):
#PPG:
#[M1,M2,M3,M4]:
#[x1,y1],ee,cc,t1,t2,f_k:
end proc:
```

```
>AnalGeo[Parbol]:=proc(p,q,alpha,X0,c1,c2)
local pp,ee,t,cc,eq,eq21,X,ttr,
F1,O1,ells,DD1,BB1,BB2,CC1,CC2,DF,BB12,PP1,PP2,PP3,
PP4,X0A1,X0A2,X0B1,X0B2,
x_min,x_max,y_min,y_max,l_x,l_y,TTXY,dd,LX,LY,PP,TT,
dd1,dd2,gdd1,prb,rho:
ttr:=(X)->[X[1]*cos(alpha)-X[2]*sin(alpha)+X0[1],
X[1]*sin(alpha)+X[2]*cos(alpha)+X0[2]]:
dd:=-p/2:
if q=1 then
F1:=ttr([p/2,0]):
DD1:=ttr([dd,0]):
BB1:=ttr([p,0]):
BB2:=ttr([3/2*p,0]):
CC1:=ttr([3/2*p,-sqrt(3)*p]):
CC2:=ttr([3/2*p,sqrt(3)*p]):
PP1:=ttr([-p/2,-sqrt(3)*p]):
PP2:=ttr([-p/2,sqrt(3)*p]):
PP3:=ttr([0,-sqrt(3)*p]):
PP4:=ttr([0,sqrt(3)*p]):
else
F1:=ttr([0,p/2]):
DD1:=ttr([0,dd]):
BB1:=ttr([0,p]):
BB2:=ttr([0,3/2*p]):
CC1:=ttr([-sqrt(3)*p,3/2*p]):
CC2:=ttr([sqrt(3)*p,3/2*p]):
PP1:=ttr([-sqrt(3)*p,-p/2]):
PP2:=ttr([sqrt(3)*p,-p/2]):
PP3:=ttr([-sqrt(3)*p,0]):
PP4:=ttr([sqrt(3)*p,0]):
end if:
rho:=evalf(sqrt((CC2[1]-CC1[1])\symbol{94}2+
(CC2[2]-CC1[2])\symbol{94}2)):
O1:=ttr([0,0]):
DF:=[F1[1]-DD1[1],F1[2]-DD1[2]]:
x_min:=min(CC1[1],CC2[1],DD1[1],PP1[1],PP2[1]):
x_max:=max(CC1[1],CC2[1],DD1[1],PP1[1],PP2[1]):
```

```
l_x:=x_max-x_min:
y_min:=min(CC1[2],CC2[2],DD1[2],PP1[2],PP2[2]):
y_max:=max(CC1[2],CC2[2],DD1[2],PP1[2],PP2[2]):
l_y:=y_max-y_min:
TTXY:=plots[textplot]([[x_max+l_x*0.01,-l_y*0.05,"x"],
[-l_x*0.05,l_y*0.01+y_max,"y"]],color=black,
font=[TIMES,ROMAN,10],align=\{left,above\}):
LX:=plot([(DD1[1]-BB1[1])*t+DD1[1],
(DD1[2]-BB1[2])*t+DD1[2],t=-1.5..1],
color=black):
LY:=plot([(CC2[1]-CC1[1])*t+O1[1],
(CC2[2]-CC1[2])*t+O1[2],t=-0.5..0.5],
color=black):
if q=1 then
eq:=(p,t)->[p*cos(t)/(1-cos(t))+p/2,p*sin(t)/(1-cos(t))]:
dd1:=(t)->ttr([-p/2,t]):
else
eq:=(p,t)->[p*sin(t)/(1-cos(t)),p*cos(t)/(1-cos(t))+p/2]:
dd1:=(t)->ttr([t,-p/2]):
end if:
eq21:=(p,t,alpha,X0)->ttr(eq(p,t)):
prb:=plot([op(eq21(p,t,alpha,X0)),
t=Pi/3..2*Pi-Pi/3],color=c1,thickness=2,
scaling=CONSTRAINED,
numpoints=500,axes=normal,scaling=CONSTRAINED):
gdd1:=plot([dd1(t)[1],dd1(t)[2],t=-rho/2..rho/2],
thickness=1,color=black):
if q=1 then
TT:=plots[textplot]([[F1[1]+0.1,F1[2],"F1"],
[O1[1]+0.1,O1[2],"O1"],[BB2[1]+0.1,BB2[2],"x1"],
[PP4[1]+0.1,PP4[2],"y1"],
[DD1[1]+0.1,DD1[2],"d1"]],title="ПАРАБОЛА",
font=[TIMES,ROMAN,10],color=c2,
align=\{right,above\}): else
TT:=plots[textplot]([[F1[1]+0.1,F1[2],"F1"],
[O1[1]+0.1,O1[2],"O1"],[BB2[1]+0.1,BB2[2],"x1"],
[PP3[1]+0.1,PP3[2],"y1"],
[DD1[1]+0.1,DD1[2],"d1"]],title="ПАРАБОЛА",
```

```
font=[TIMES,ROMAN,10],color=c2,
align=\{right,above\}): end if:
PP:=plots[pointplot]([F1,O1,X0,DD1],color=black,
symbol=circle,symbolsize=14,
color=c2):
plots[display](LX,TTXY,LY,prb,gdd1,TT,PP,
scaling=CONSTRAINED):
#[DD1,dd1(t)]:
#eq2(pp,ee,t):
#eq21(p,t,alpha,X0):
end proc:

>AnalGeo[Axes11]:=proc(X,Y,alpha,X0) local ttr,x_max,
y_max,x_min,y_min,XY,XX,YY,TXT,xx,yy,xx1,
yy1,ddx,ddy,ddd,ii,xgr,ygr:
x_min:=floor(X[1]):x_max:=ceil(X[2]):
ddx:=x_max-x_min:y_min:=floor(Y[1]):
y_max:=ceil(Y[2]):ddy:=y_max-y_min:
ddd:=max([ddx,ddy]):
ttr:=(xx1,yy1)->[xx1*cos(alpha)-yy1*sin(alpha)+X0[1],
xx1*sin(alpha)+yy1*cos(alpha)+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=x_min..x_min+ddd],
thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=y_min..y_min+ddd],
thickness=1,color=black):
xgr:=plots[display](plot([op(ttr(x_min,xx)),xx=0..0.015*ddd],
color=black,thickness=1),
seq(plot([op(ttr(x_min+ii,xx)),xx=0..0.015*ddd],color=black,
thickness=1),ii=1..ddd)):
ygr:=plots[display](plot([op(ttr(xx,y_min)),xx=0..0.015*ddd],
color=black,thickness=1),
seq(plot([op(ttr(xx,y_min+ii)),xx=0..0.015*ddd],color=black,
thickness=1),ii=0..ddd)):
TXT:=plots[textplot](\{[op(ttr(0.015*ddd,-0.015*ddd)),`O`],
[op(ttr(X[2],-0.015*ddd)),`x`],
[op(ttr(0.015*ddd,Y[2])),`y`]\},align=\{below,right\}):
plots[display](XX,YY,xgr,ygr,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:
```

```
>AnalGeo[Axes01]:=proc(X,Y,cosa,sina,X0)
local ttr,XY,XX,YY,TXT,xx,yy,xx1,yy1,ddx,ddy,ddd:
ttr:=(xx1,yy1)->[xx1*evalf(cosa)-yy1*evalf(sina)+
X0[1],xx1*evalf(sina)+yy1*evalf(cosa)+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=X[1]..X[2]],thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=Y[1]..Y[2]],thickness=1,color=black):
ddx:=(X[2]-X[1]):
ddy:=(Y[2]-Y[1]):
ddd:=max([ddx,ddy]):
TXT:=plots[textplot](\{[op(ttr(0.015*ddd,-0.015*ddd)),‘O1‘],
[op(ttr(X[2],-0.015*ddd)),‘x1‘],
[op(ttr(0.015*ddd,Y[2])),‘y1‘]\},align=\{below,right\}):
plots[display](XX,YY,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:

>AnalGeo[Ell1]:=proc(a,b,cosa,sina,X0,c1,c2)
local pp,ee,t,cc,eq2,eq21,X,ttr,
F1,F2,AA1,AA2,BB1,BB2,ells,DD1,DD2,AA12,BB12,O1,
X0A1,X0A2,X0B1,X0B2,
dd,LX,LY,PP,TT,dd1,dd2,gdd1,gdd2:
ttr:=(X)->[X[1]*cosa-X[2]*sina+X0[1],X[1]*sina+
X[2]*cosa+X0[2]]:
if evalf(a^2)>evalf(b^2) then
pp:=b^2/a:ee:=sqrt(a^2-b^2)/a:
cc:=sqrt(a^2-b^2):dd:=a^2/cc:
else
pp:=a^2/b:ee:=sqrt(b^2-a^2)/b:
cc:=sqrt(b^2-a^2):dd:=b^2/cc:
end if:
if evalf(a^2)>evalf(b^2) then
F1:=ttr([-cc,0]):F2:=ttr([cc,0]):
else
F1:=ttr([0,-cc]):F2:=ttr([0,cc]):
end if:
AA1:=ttr([-a,0]):AA2:=ttr([a,0]):
BB1:=ttr([0,-b]):BB2:=ttr([0,b]):
O1:=ttr([0,0]):
AA12:=[AA2[1]-AA1[1],AA2[2]-AA1[2]]:
```

```
BB12:=[BB2[1]-BB1[1],BB2[2]-BB1[2]]:
if evalf(a^2)>evalf(b^2) then
DD1:=ttr([-dd,0]):DD2:=ttr([dd,0]):
else
DD1:=ttr([0,-dd]):DD2:=ttr([0,dd]):
end if:
if evalf(a^2)>evalf(b^2) then
LX:=AnalGeo[Axes01]([-dd,dd],[-b,b],cosa,sina,X0):
else
LX:=AnalGeo[Axes01]([-a,a],[-dd,dd],cosa,sina,X0):
#LY:=AnalGeo[Axes1]([-0.2*AA2[1]+1.2*AA1[1],-0.2*AA1[1]+
1.2*AA2[1]],[-0.2*BB2[1]+1.2*BB1[1],-0.2*BB1[1]+
1.2*BB2[1]],alpha,X0):
end if:
LY:=AnalGeo[Axes11]([-dd,dd],[-b,b],1,0,[0,0]):
if evalf(a^2)>evalf(b^2) then
eq2:=(pp,ee,t)->[pp*cos(t)/(1-ee*cos(t))-cc,
pp*sin(t)/(1-ee*cos(t))]:
dd1:=(t)->[DD1[1]+BB12[1]*t,DD1[2]+BB12[2]*t]:
dd2:=(t)->[DD2[1]+BB12[1]*t,DD2[2]+BB12[2]*t]:
else
eq2:=(pp,ee,t)->[pp*sin(t)/(1-ee*cos(t)),
pp*cos(t)/(1-ee*cos(t))-cc]:
dd1:=(t)->[DD1[1]+AA12[1]*t,DD1[2]+AA12[2]*t]:
dd2:=(t)->[DD2[1]+AA12[1]*t,DD2[2]+AA12[2]*t]:
end if:
eq21:=(pp,ee,t,cosa,sina,X0)->ttr(eq2(pp,ee,t)):
ells:=plot([op(eq21(pp,ee,t,cosa,sina,X0)),t=0..2*Pi],
color=c1,thickness=2,scaling=CONSTRAINED,
numpoints=500,axes=normal,scaling=CONSTRAINED):
gdd1:=plot([op(dd1(t)),t=-1/2..1/2],thickness=1,
color=black):
gdd2:=plot([op(dd2(t)),t=-1/2..1/2],thickness=1,
color=black):
TT:=plots[textplot]([[F1[1],F1[2],"F1"],[F2[1],F2[2],"F2"],
[AA1[1],AA1[2],"A1"],[AA2[1],AA2[2],"A2"],[BB1[1],BB1[2],"B1"],
[BB2[1],BB2[2],"B2"],[DD1[1],DD1[2],"d1"],[DD2[1],DD2[2],"d2"]],
title="? ? ? ? ? ?",font=[TIMES,ROMAN,10],color=c2):
```

```
PP:=plots[pointplot]([F1,F2,AA1,AA2,BB1,BB2,X0,DD1,DD2],
color=black,symbol=circle,symbolsize=14,color=c2):
plots[display](LX,LY,ells,gdd1,gdd2,TT,PP,scaling=CONSTRAINED):
end proc:
```

II.4.4 Команда изображения гиперболы

```
>AnalGeo[Axes11]:=proc(X,Y,cosa,sina,X0)
local ttr,x_max,y_max,x_min,y_min,XY,XX,YY,
TXT,xx,yy,xx1,yy1,ddx,ddy,ddd,ii,xgr,ygr:
x_min:=floor(X[1]):x_max:=ceil(X[2]):
ddx:=x_max-x_min:y_min:=floor(Y[1]):
y_max:=ceil(Y[2]):ddy:=y_max-y_min:
ddd:=max([ddx,ddy]):
ttr:=(xx1,yy1)->[xx1*cosa-yy1*sina+X0[1],
xx1*sina+yy1*cosa+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=x_min..x_min+ddd],
thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=y_min..y_min+ddd],
thickness=1,color=black):
xgr:=plots[display](plot([op(ttr(x_min,xx)),
xx=0..0.015*ddd],color=black,thickness=1),
seq(plot([op(ttr(x_min+ii,xx)),xx=0..0.015*ddd],
color=black,thickness=0),ii=1..ddd)):
ygr:=plots[display](plot([op(ttr(xx,y_min)),
xx=0..0.015*ddd],color=black,thickness=1),
seq(plot([op(ttr(xx,y_min+ii)),xx=0..0.015*ddd],
color=black,thickness=0),ii=0..ddd)):
TXT:=plots[textplot]({[op(ttr(0.015*ddd,-0.015*ddd)),`O`],
[op(ttr(X[2],-0.015*ddd)),`x`],
[op(ttr(0.015*ddd,Y[2])),`y`]},align={below,right}):
plots[display](XX,YY,xgr,ygr,TXT,axes=NONE,
scaling=CONSTRAINED):
end proc:

>AnalGeo[Axes01]:=proc(X,Y,cosa,sina,X0)
```

```
local ttr,XY,XX,YY,TXT,xx,yy,xx1,yy1,ddx,ddy,ddd:
ttr:=(xx1,yy1)->[xx1*evalf(cosa)-yy1*evalf(sina)+X0[1],
xx1*evalf(sina)+yy1*evalf(cosa)+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=X[1]..X[2]],thickness=1,
color=black):
YY:=plot([op(ttr(0,yy1)),yy1=Y[1]..Y[2]],thickness=1,
color=black):
ddx:=(X[2]-X[1]):ddy:=(Y[2]-Y[1]):
ddd:=max([ddx,ddy]):
TXT:=plots[textplot]({[op(ttr(0.015*ddd,-0.015*ddd)),‘O1‘],
[op(ttr(X[2],-0.015*ddd)),‘x1‘],
[op(ttr(0.015*ddd,Y[2])),‘y1‘]},align={below,right}):
plots[display](XX,YY,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:

>AnalGeo[Hyp1]:=proc(a,b,q,cosa,sina,X0,c1,c2,d)
local pp,ee,t,cc,eq2,eq21,X,xx,ttr,F1,F2,AA1,
AA2,BB1,BB2,DD1,DD2,AA12,BB12,X0A1,X0A2,X0B1,X0B2,
x_min,x_max,y_min,y_max,l_x,l_y,f,f_k,TTXY,dd,
coord0,coord1,PP,TT,dd1,dd2,gdd1,gdd2,
eq1,hypb,eq22,g21,g22,M1,M2,M3,M4,asimp1,
asimp2,ppg,PPG,TTit:
ttr:=(X)->[X[1]*cosa-X[2]*sina+X0[1],X[1]*
sina+X[2]*cosa+X0[2]]:
M1:=ttr([-a,-b]): M2:=ttr([a,-b]): M3:=ttr([a,b]):
M4:=ttr([-a,b]):
ppg:=plots[polygonplot]([M1,M2,M3,M4],
color=COLOR(RGB,0.75,0.75,0.75),scaling=CONSTRAINED):
PPG:=plots[textplot]([[M1[1]+0.2,M1[2]+0.2,"M1"],
[M2[1]+0.2,M2[2]+0.2,"M2"],
[M3[1]+0.2,M3[2]+0.2,"M3"],
[M4[1]+0.2,M4[2]+0.2,"M4"]],font=[TIMES,ROMAN,10],
color=c2,scaling=CONSTRAINED):
if q=1 then
pp:=b^2/a:ee:=sqrt(a^2+b^2)/a:
cc:=sqrt(a^2+b^2):dd:=a^2/cc:
else
pp:=a^2/b:ee:=sqrt(b^2+a^2)/b:
```

```
cc:=sqrt(b^2+a^2):dd:=b^2/cc:
end if:
if q=1 then
F1:=ttr([-cc,0]):F2:=ttr([cc,0]):
else
F1:=ttr([0,-cc]):F2:=ttr([0,cc]):
end if:
if q=1 then
asimp1:=plot([op(ttr([xx,b*xx/a])),xx=-(1+d)*cc..(1+d)*cc],
style=point,symbol=point,color=black,scaling=CONSTRAINED):
asimp2:=plot([op(ttr([xx,-b*xx/a])),xx=-(1+d)*cc..(1+d)*cc],
style=point,symbol=point,color=black,scaling=CONSTRAINED):
else
asimp1:=plot([op(ttr([a/b*xx,xx])),xx=-(1+d)*cc..(1+d)*cc],
style=point,symbol=point,color=black,scaling=CONSTRAINED):
asimp2:=plot([op(ttr([-a/b*xx,xx])),xx=-(1+d)*cc..(1+d)*cc],
style=point,symbol=point,color=black,scaling=CONSTRAINED):
end if:
AA1:=ttr([-a,0]):
AA2:=ttr([a,0]):
BB1:=ttr([0,-b]):
BB2:=ttr([0,b]):
AA12:=[AA2[1]-AA1[1],AA2[2]-AA1[2]]:
BB12:=[BB2[1]-BB1[1],BB2[2]-BB1[2]]:
x_min:=min(F1[1],F2[1],AA1[1],AA2[1],BB1[1],BB2[1]):
x_max:=max(F1[1],F2[1],AA1[1],AA2[1],BB1[1],BB2[1]):
l_x:=x_max-x_min:
y_min:=min(F1[2],F2[2],AA1[2],AA2[2],BB1[2],BB2[2]):
y_max:=max(F1[2],F2[2],AA1[2],AA2[2],BB1[2],BB2[2]):
l_y:=y_max-y_min:
if q=1 then
DD1:=ttr([-dd,0]):
DD2:=ttr([dd,0]):
else
DD1:=ttr([0,-dd]):
DD2:=ttr([0,dd]):
end if:
#Изображение новых осей координат
```

```
if q=1 then
coord1:=AnalGeo[Axes11]([-(1+d)*cc,(1+d)*cc],[-b/a*cc*(1+d),
b/a*cc*(1+d)],1,0,[0,0]):
coord0:=AnalGeo[Axes01]([-(1+d)*cc,(1+d)*cc],[-b/a*cc*(1+d),
b/a*cc*(1+d)],cosa,sina,X0):
#LY:=plot([(BB2[1]-BB1[1])*t+BB1[1],(BB2[2]-BB1[2])*t+BB1[2],
t=-1..2],color=black):
else
coord1:=AnalGeo[Axes11]([-a/b*cc*(1+d),a/b*cc*(1+d)],
[-(1+d)*cc,(1+d)*cc],1,0,[0,0]):
coord0:=AnalGeo[Axes01]([-a/b*cc*(1+d),a/b*cc*(1+d)],
[-(1+d)*cc,(1+d)*cc],cosa,sina,X0):
#LY:=plot([(AA2[1]-AA1[1])*t+AA1[1],(AA2[2]-AA1[2])*t+AA1[2],
t=-1..2],color=black):
end if:
#LX:=AnalGeo[Axes11]([-(1+d)*cc,(1+d)*cc],[-b,b],0,[0,0]):
if q=1 then
#-------------------------------
#Главная ось OX
#Уравнения левой(eq1) и правой(eq2) ветвей гиперболы в
#полярных координатах
#-------------------------------
eq1:=(pp,ee,t)->[-pp*cos(t)/(1-ee*cos(t))-cc,
pp*sin(t)/(1-ee*cos(t))]:
eq2:=(pp,ee,t)->[pp*cos(t)/(1-ee*cos(t))+cc,
pp*sin(t)/(1-ee*cos(t))]:
#-------------------------------------
#Уравнения директрисс для случая главной оси OX
dd1:=(t)->[DD1[1]+BB12[1]*t,DD1[2]+BB12[2]*t]:
dd2:=(t)->[DD2[1]+BB12[1]*t,DD2[2]+BB12[2]*t]:
else
#-------------------------------------
#Главная ось OY
#Уравнения нижней(eq1) и верхней(eq2)
#ветвей гиперболы в полярных координатах
#-------------------------------------
eq1:=(pp,ee,t)->[pp*sin(t)/(1-ee*cos(t)),
-pp*cos(t)/(1-ee*cos(t))-cc]:
```

```
eq2:=(pp,ee,t)->[pp*sin(t)/(1-ee*cos(t)),
pp*cos(t)/(1-ee*cos(t))+cc]:
#--------------------------------------
#Уравнения директрисс для случая главной оси OY
#--------------------------------------
dd1:=(t)->[DD1[1]+AA12[1]*t,DD1[2]+AA12[2]*t]:
dd2:=(t)->[DD2[1]+AA12[1]*t,DD2[2]+AA12[2]*t]:
#
end if:
#Преобразованные уравнения левой/нижней(eq21)  и
#правой/верхней(eq22) ветвей гиперболы
eq21:=(pp,ee,t,cosa,sina,X0)->ttr(eq2(pp,ee,t)):
eq22:=(pp,ee,t,cosa,sina,X0)->ttr(eq1(pp,ee,t)):
f:=evalf(arctan(b/a),4):
if q=1 then
f_k:=evalf(arctan(b*sqrt(cc^2*(1+d)^2-a^2)/(a*cc*d))):
else
f_k:=evalf(arctan(a*sqrt(cc^2*(1+d)^2-b^2)/(b*cc*d))):
end if:
g21:=plot([op(eq21(pp,ee,t,cosa,sina,X0)),t=f_k..2*Pi-f_k],
color=c1,thickness=2,
scaling=CONSTRAINED,
numpoints=500,axes=Normal):
g22:=plot([op(eq22(pp,ee,t,cosa,sina,X0)),t=f_k..2*Pi-f_k],
color=c1,thickness=2,
scaling=CONSTRAINED,
numpoints=500,axes=Normal):
hypb:=plots[display](g21,g22):
TT:=plots[textplot]([[F1[1]+0.5,F1[2],"F1"],
[F2[1]+0.5,F2[2],"F2"],
[AA1[1]+0.3,AA1[2],"A1"],[AA2[1]+0.3,AA2[2],"A2"],
[BB1[1]+0.3,BB1[2],"B1"],[BB2[1]-0.3,BB2[2],"B2"],
[DD1[1]+0.4,DD1[2],"d1"],[DD2[1]+0.4,DD2[2],"d2"]],
title="ГИПЕРБОЛА",font=[TIMES,ROMAN,10],color=c2):
PP:=plots[pointplot]([F1,F2,AA1,AA2,BB1,BB2,X0,DD1,DD2],
color=black,symbol=circle,symbolsize=14,color=c2):
gdd1:=plot([dd1(t)[1],dd1(t)[2],t=-1..1],thickness=1,
color=COLOR(RGB,0,0,0),style=line):
```

```
gdd2:=plot([op(dd2(t)),t=-1..1],thickness=1,
color=COLOR(RGB,0,0,0)):
plots[display](coord0,coord1,PP,TT,ppg,PPG,hypb,
asimp1,asimp2,gdd1,gdd2,
scaling=CONSTRAINED):
end proc:
```

Команда изображения оси

```
>AnalGeo[Axes11]:=proc(X,Y,cosa,sina,X0)
local ttr,x_max,y_max,x_min,y_min,XY,XX,YY,TXT,
xx,yy,xx1,yy1,ddx,ddy,ddd,ii,xgr,ygr,x_end,y_end:
x_min:=floor(X[1]):x_max:=ceil(X[2]):
ddx:=x_max-x_min:y_min:=floor(Y[1]):
y_max:=ceil(Y[2]):ddy:=y_max-y_min:
ddd:=max([ddx,ddy]):ttr:=(xx1,yy1)->
[xx1*cosa-yy1*sina+X0[1],xx1*sina+yy1*cosa+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=x_min..x_min+ddd],
thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=y_min..y_min+ddd],
thickness=1,color=black):
xgr:=plots[display](plot([op(ttr(x_min,xx)),
xx=0..0.015*ddd],color=black,thickness=1),
seq(plot([op(ttr(x_min+ii,xx)),xx=0..0.015*ddd],
color=black,thickness=0),ii=1..ddd)):
ygr:=plots[display](plot([op(ttr(xx,y_min)),
xx=0..0.015*ddd],color=black,thickness=1),
seq(plot([op(ttr(xx,y_min+ii)),xx=0..0.015*ddd],
color=black,thickness=0),ii=0..ddd)):
x_end:=x_min+ddd:y_end:=y_min+ddd:
TXT:=plots[textplot]({[op(ttr(0.015*ddd,-0.015*ddd)),`O`],
[op(ttr(x_end,-0.02*ddd)),`x`],
[op(ttr(0.05*ddd,y_end)),`y`]},align={below,right}):
plots[display](XX,YY,xgr,ygr,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:

>AnalGeo[Axes01]:=proc(X,Y,cosa,sina,X0)
local ttr,XY,XX,YY,TXT,xx,yy,xx1,yy1,ddx,ddy,ddd:
```

```
ttr:=(xx1,yy1)->[xx1*evalf(cosa)-yy1*evalf(sina)+X0[1],
xx1*evalf(sina)+yy1*evalf(cosa)+X0[2]]:
XX:=plot([op(ttr(xx1,0)),xx1=X[1]..X[2]],thickness=1,color=black):
YY:=plot([op(ttr(0,yy1)),yy1=Y[1]..Y[2]],thickness=1,color=black):
ddx:=(X[2]-X[1]):
ddy:=(Y[2]-Y[1]):
ddd:=max([ddx,ddy]):
TXT:=plots[textplot]({[op(ttr(0.015*ddd,-0.015*ddd)),‘O1‘],
[op(ttr(X[2],-0.015*ddd)),‘x1‘],
[op(ttr(0.015*ddd,Y[2])),‘y1‘]},align={below,right}):
plots[display](XX,YY,TXT,axes=NONE,scaling=CONSTRAINED):
end proc:
```

Команда изображения параболы

```
>AnalGeo[Parbol1]:=proc(p,q,cosa,sina,X0,c1,c2)
local pp,ee,t,cc,eq,eq21,X,ttr,F1,O1,ells,
DD1,BB1,BB2,CC1,CC2,DF,BB12,PP1,PP2,PP3,PP4,X0A1,X0A2,X0B1,X0B2,
dd,coord0,coord1,PP,TT,dd1,dd2,gdd1,prb,rho:
ttr:=(X)->[X[1]*cosa-X[2]*sina+X0[1],X[1]*sina+X[2]*cosa+X0[2]]:
dd:=-p/2:
if q=1 then
F1:=ttr([p/2,0]):DD1:=ttr([dd,0]):
BB1:=ttr([p,0]):BB2:=ttr([3/2*p,0]):
CC1:=ttr([3/2*p,-sqrt(3)*p]):CC2:=ttr([3/2*p,sqrt(3)*p]):
PP1:=ttr([-p/2,-sqrt(3)*p]):PP2:=ttr([-p/2,sqrt(3)*p]):
PP3:=ttr([0,-sqrt(3)*p]):PP4:=ttr([0,sqrt(3)*p]):
else
F1:=ttr([0,p/2]):DD1:=ttr([0,dd]):
BB1:=ttr([0,p]):BB2:=ttr([0,3/2*p]):
CC1:=ttr([-sqrt(3)*p,3/2*p]):CC2:=ttr([sqrt(3)*p,3/2*p]):
PP1:=ttr([-sqrt(3)*p,-p/2]):PP2:=ttr([sqrt(3)*p,-p/2]):
PP3:=ttr([-sqrt(3)*p,0]):PP4:=ttr([sqrt(3)*p,0]):
end if:
rho:=evalf(sqrt((CC2[1]-CC1[1])^2+(CC2[2]-CC1[2])^2)):
O1:=ttr([0,0]):
DF:=[F1[1]-DD1[1],F1[2]-DD1[2]]:
if  p>=0 then
```

```
coord1:=AnalGeo[Axes11]([-3/2*p,3/2*p],
[-sqrt(3)*p,sqrt(3)*p],1,0,[0,0]):
coord0:=AnalGeo[Axes01]([-3/2*p,3/2*p],
[-sqrt(3)*p,sqrt(3)*p],cosa,sina,X0):
else
coord1:=AnalGeo[Axes11]([3/2*p,-3/2*p],
[sqrt(3)*p,-sqrt(3)*p],1,0,[0,0]):
coord0:=AnalGeo[Axes01]([3/2*p,-3/2*p],
[sqrt(3)*p,-sqrt(3)*p],cosa,sina,X0):
end if:
#LX:=plot([(DD1[1]-BB1[1])*t+DD1[1],
(DD1[2]-BB1[2])*t+DD1[2],t=-1.5..1],color=black):
if q=1 then
eq:=(p,t)->[p*cos(t)/(1-cos(t))+p/2,p*sin(t)/(1-cos(t))]:
dd1:=(t)->ttr([-p/2,t]):
else
eq:=(p,t)->[p*sin(t)/(1-cos(t)),p*cos(t)/(1-cos(t))+p/2]:
dd1:=(t)->ttr([t,-p/2]):
end if:
eq21:=(p,t,cosa,sina,X0)->ttr(eq(p,t)):
prb:=plot([op(eq21(p,t,cosa,sina,X0)),t=Pi/3..2*Pi-Pi/3],
color=c1,thickness=2,scaling=CONSTRAINED,
numpoints=500,axes=normal,scaling=CONSTRAINED):
gdd1:=plot([dd1(t)[1],dd1(t)[2],t=-rho/2..rho/2],
thickness=1,color=black):
if q=1 then
TT:=plots[textplot]([[F1[1]+0.1,F1[2],"F1"],
[DD1[1]+0.1,DD1[2],"d1"]],title="ПАРАБОЛА",font=[TIMES,ROMAN,10],
color=c2,align={right,above}): else
TT:=plots[textplot]([[F1[1]+0.1,F1[2],"F1"],
[DD1[1]+0.1,DD1[2],"d1"]],title="ПАРАБОЛА",font=[TIMES,ROMAN,10],
color=c2,align={right,above}): end if:
PP:=plots[pointplot]([F1,O1,X0,DD1],color=black,symbol=circle,
symbolsize=14,color=c2):
plots[display](coord1,coord0,prb,gdd1,TT,PP,scaling=CONSTRAINED):
end proc:
```

Команда изображение пересекающихся прямых

```
>AnalGeo[CrossDir1]:=proc(a,b,cosa,sina,X0,c1,c2)
local X,t,Eq1,Eq2,ttr,Eq1_1,Eq2_1,gd1,gd2,M1_1,
M1_2,M2_1,M2_2,X_min,X_max,Y_min,Y_max,CRD,CRD1,TT:
ttr:=(X)->[X[1]*cosa-X[2]*sina+X0[1],X[1]*sina+X[2]*cosa+X0[2]]:
Eq1:=(a,b,t)->[b*t,a*t]: Eq2:=(a,b,t)->[b*t,-a*t]:
Eq1_1:=(a,b,t,cosa,sina,X0)->ttr(Eq1(a,b,t)):
Eq2_1:=(a,b,t,cosa,sina,X0)->ttr(Eq2(a,b,t)):
M1_1:=Eq1_1(a,b,-1,cosa,sina,X0):
M1_2:=Eq1_1(a,b,1,cosa,sina,X0):
M2_1:=Eq2_1(a,b,-1,cosa,sina,X0):
M2_2:=Eq2_1(a,b,1,cosa,sina,X0):
X_min:=min(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
X_max:=max(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
Y_min:=min(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
Y_max:=max(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
CRD:=AnalGeo[Axes11]([X_min,X_max],[Y_min,Y_max],1,0,[0,0]):
CRD1:=AnalGeo[Axes01]([X_min-X0[1],X_max-X0[1]],[Y_min-X0[2],
Y_max-X0[2]],cosa,sina,X0):
gd1:=(a,b,cosa,sina,X0)->plot([op(Eq1_1(a,b,t,cosa,sina,X0)),
t=-1..1],color=c1,thickness=1):
gd2:=(a,b,cosa,sina,X0)->plot([op(Eq2_1(a,b,t,cosa,sina,X0)),
t=-1..1],color=c2,thickness=1):
TT:=plots[textplot]([[M1_1[1],M1_1[2],"d1"],
[M2_1[1],M2_1[2],"d2"]],
title="П А Р А  П Е Р Е С Е К А Ю Щ И Х С Я  П Р Я М Ы Х",
font=[TIMES,ROMAN,10],color=c2,align=above):
plots[display](gd1(a,b,cosa,sina,X0),
gd2(a,b,cosa,sina,X0),CRD,CRD1,TT):
end proc:
```

Команда изображения параллельных прямых

```
>AnalGeo[ParalDir1]:=proc(a,cosa,sina,X0,x,c1,c2)
local X,t,Eq1,Eq2,ttr,Eq1_1,Eq2_1,gd1,gd2,M1_1,
M1_2,M2_1,M2_2,X_min,X_max,Y_min,Y_max,CRD,CRD1,TT:
ttr:=(X)->[X[1]*cosa-X[2]*sina+X0[1],X[1]*sina+X[2]*cosa+X0[2]]:
if x=1 then
Eq1:=(a,t)->[t,a]: Eq2:=(a,t)->[t,-a]:
Eq1_1:=(a,t,cosa,sina,X0)->ttr(Eq1(a,t)):
```

```
Eq2_1:=(a,t,cosa,sina,X0)->ttr(Eq2(a,t)):
M1_1:=Eq1_1(a,-1,cosa,sina,X0):
M1_2:=Eq1_1(a,1,cosa,sina,X0):
M2_1:=Eq2_1(a,-1,cosa,sina,X0):
M2_2:=Eq2_1(a,1,cosa,sina,X0):
else
Eq1:=(a,t)->[a,t]: Eq2:=(a,t)->[-a,t]:
Eq1_1:=(a,t,cosa,sina,X0)->ttr(Eq1(a,t)):
Eq2_1:=(a,t,cosa,sina,X0)->ttr(Eq2(a,t)):
M1_1:=Eq1_1(-1,a,cosa,sina,X0):
M1_2:=Eq1_1(1,a,cosa,sina,X0):
M2_1:=Eq2_1(-1,a,cosa,sina,X0):
M2_2:=Eq2_1(1,a,cosa,sina,X0):
end if:
X_min:=min(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
X_max:=max(0,X0[1],M1_1[1],M1_2[1],M2_1[1],M2_2[1]):
Y_min:=min(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
Y_max:=max(0,X0[2],M1_1[2],M1_2[2],M2_1[2],M2_2[2]):
CRD:=AnalGeo[Axes11]([X_min,X_max],[Y_min,Y_max],1,0,[0,0]):
CRD1:=AnalGeo[Axes01]([X_min-X0[1],X_max-X0[1]],[Y_min-X0[2],
Y_max-X0[2]],cosa,sina,X0):
gd1:=(a,cosa,sina,X0)->plot([op(Eq1_1(a,t,cosa,sina,X0)),
t=-1..1],thickness=1,color=c1):
gd2:=(a,cosa,sina,X0)->plot([op(Eq2_1(a,t,cosa,sina,X0)),
t=-1..1],thickness=1,color=c2):
TT:=plots[textplot]([[M1_1[1],M1_1[2],"d1"],
[M2_1[1],M2_1[2],"d2"]],
title="П А Р А  П А Р А Л Л Е Л Ь Н Ы Х  П Р Я М Ы Х",
font=[TIMES,ROMAN,10],color=c2,align=above):
plots[display](gd1(a,cosa,sina,X0),
gd2(a,cosa,sina,X0),CRD,CRD1,TT):
end proc:
```

Команда изображение точки (X=0,Y=0)

```
>AnalGeo[Point1]:=proc(cosa,sina,X0,c1)
local X,ttr,X_min,X_max,Y_min,Y_max,CRD,CRD1,gp:
ttr:=(X)->[X[1]*cosa-X[2]*sina+X0[1],X[1]*sina+X[2]*cosa+X0[2]]:
if X0[1]<>0 and X0[2]<>0 then
```

```
X_min:=min(0,X0[1],-X0[1]):
X_max:=max(0,X0[1],-X0[1],1):
Y_min:=min(0,X0[2],-X0[2]):
Y_max:=max(0,X0[2],-X0[2],1):
elif
X0[1]=0 and X0[2]<>0 then
X_min:=-1:X_max:=1:Y_min:=min(0,X0[2],-X0[2]):
Y_max:=max(0,X0[2],-X0[2],1):
elif
X0[1]<>0 and X0[2]=0 then
X_min:=min(0,-1,X0[1],-X0[1]):
X_max:=max(0,X0[1],-X0[1],1):
Y_min:=min(0,-1,X0[2],-X0[2]):Y_max:=max(0,X0[2],-X0[2],1):
elif
X0[1]=0 and X0[2]=0 then
X_min:=-1:
X_max:=1:
Y_min:=-1:Y_max:=1:
end if:
CRD:=AnalGeo[Axes11]([X_min,X_max],[Y_min,Y_max],1,0,[0,0]):
CRD1:=AnalGeo[Axes01]([X_min,X_max],[Y_min,Y_max],cosa,sina,X0):
gp:=plots[pointplot](X0,color=c1,symbol=circle,symbolsize=16,
title="T O Ч K A (X=0,Y=0)",font=[TIMES,ROMAN,10],color=c1):
plots[display](CRD,CRD1,gp):
end proc:
```

Команда изображения пустого множества

```
>AnalGeo[nullSet]:=proc(cosa,sina,X0,c1)local CRD,CRD1:
CRD:=AnalGeo[Axes11]([-1,1],[-1,1],1,0,[0,0]):
CRD1:=AnalGeo[Axes01]([-1,1],[-1,1],cosa,sina,X0):
plots[display](CRD,CRD1,title="ПУСТОЕ МНОЖЕСТВО",
font=[TIMES,ROMAN,10]):
end proc:
```

II.4.5 Определение коэффициентов общего уравнения второго порядка на плоскости

```
>AnalGeo[Coeffs2]:=proc(Eq,X)
local Q,xx,yy,a_11,a_12,a_22,AA,b_1,b_2,cc,dd,Q2,LQ,BB:
Q:=lhs(Eq):
xx:=X[1]:yy:=X[2]:
a_11:=coeff(Q,xx^2):a_22:=coeff(Q,yy^2):
dd:=1/2*coeff(Q,yy):a_12:=coeff(dd,xx):
AA:=Matrix([[a_11,a_12],[a_12,a_22]]):
Q2:=a_11*xx^2+2*a_12*xx*yy+a_22*yy^2:
LQ:=simplify(Q-Q2):
b_1:=coeff(LQ,xx):b_2:=coeff(LQ,yy):
BB:=Vector([b_1,b_2]):
cc:=simplify(LQ-b_1*xx-b_2*yy):
[AA,BB,cc]:
end proc:
```

II.4.6 Преобразование квадрики с помощью движения

```
>AnalGeo[Trans2]:=proc(Eq,X,X1,s,p)
local xx,yy,xx1,yy1,Q,AA,BB,cc,a_11,a_12,a_22,b_1,b_2,Q2,LQ,LQ1,TR1,
Q21,aa_11,aa_12,aa_22,AA_11,AA_12,AA_22,BB_1,BB_2:
xx:=X[1]:yy:=X[2]:xx1:=X1[1]:yy1:=X1[2]:
Q:=lhs(Eq):AA:=AnalGeo[Coeffs2](Eq,X)[1]:
BB:=AnalGeo[Coeffs2](Eq,X)[2]:
cc:=AnalGeo[Coeffs2](Eq,X)[3]:
a_11:=AA[1,1]:a_12:=AA[1,2]:a_22:=AA[2,2]:b_1:=BB[1]:b_2:=BB[2]:
Q2:=a_11*xx^2+2*a_12*xx*yy+a_22*yy^2:
LQ:=b_1*xx+b_2*yy:
TR1:=(s,p)->{xx=xx1*cos(s)-p*yy1*sin(s),yy=xx1*sin(s)+p*yy1*cos(s)}:
Q21:=(s,p)->collect(collect(subs(TR1(s,p),Q2),xx1),yy1^2):
AA_11:=(s,p)->coeff(Q21(s,p),xx1^2):
AA_12:=(s,p)->simplify(1/2*coeff(coeff(Q21(s,p),yy1),xx1)):
AA_22:=(s,p)->coeff(Q21(s,p),yy1^2):
LQ1:=(s,p)->collect(collect(subs(TR1(s,p),LQ),xx1),yy1):
AA:=(s,p)->Matrix([[AA_11(s,p),AA_12(s,p)],[AA_12(s,p),AA_22(s,p)]]):
BB_1:=(s,p)->simplify(coeff(LQ1(s,p),xx1)):
BB_2:=(s,p)->simplify(coeff(LQ1(s,p),yy1)):
```

```
BB:=(s,p)->[BB_1(s,p),BB_2(s,p)]:
[AA(s,p),BB(s,p),cc]:
end proc:
```

II.4.7 Определение sin и cos угла поворота

```
>AnalGeo[Angle2]:=proc(Eq,X,s)
local Q,xx,yy,dd,a_11,a_12,a_22,AA,b_1,b_2,cc,Q2,LQ,BB,sss:
Q:=lhs(Eq):
xx:=X[1]:yy:=X[2]:
dd:=1/2*coeff(Q,yy):
a_11:=coeff(Q,xx^2):a_22:=coeff(Q,yy^2):a_12:=coeff(dd,xx):
if a_12<>0 then
sss:=[cos(s)=1/sqrt(2)*sqrt(1+abs(a_11-a_22)
/sqrt((a_11-a_22)^2+4*a_12^2)),
sin(s)=sign((a_11-a_22)*a_12)/sqrt(2)*
sqrt(1-abs(a_11-a_22)/sqrt((a_11-a_22)^2+4*a_12^2))]:
elif a_12=0 then
sss:=[cos(s)=1,sin(s)=0]:
end if:
sss:
end proc:


>AnalGeo[Trans3]:=proc(Eq,X,X1,s):
simplify(combine(subs(\{op(AnalGeo[Angle2](Eq,X,s)),p=1\},
AnalGeo[Trans2](Eq,X,X1,s,p))):end proc:
```

II.4.8 Канонические уравнения фигуры

```
>AnalGeo[CanonF]:=proc(Eq,X,X1,s)
local TR3,QQ2,LQ1,lambda1,lambda2,b1,
b2,c1,c2,Ang,cosa,sina,QQ3,ttr,xx,yy,xx1,yy1,txt,
EqF,GF,aa,bb,aaa1,bbb1,aa2,bb2,aaf,bbf,cosaf,sinaf,
pp,ppf,txtt,cc,cc2,ee,dd,acc,x0,y0,AAA:
TR3:=AnalGeo[Trans3](Eq,X,X1,s):
QQ2:=TR3[1]:LQ1:=TR3[2]:c1:=TR3[3]:
lambda1:=QQ2[1,1]:lambda2:=QQ2[2,2]:
b1:=LQ1[1]:b2:=LQ1[2]:
```

```
Ang:=AnalGeo[Angle2](Eq,X,s):
cosa:=rhs(Ang[1]):sina:=rhs(Ang[2]):
xx:=X[1]:yy:=X[2]:xx1:=X1[1]:yy1:=X1[2]:
#-----------------------------
if evalf(lambda1)<>0 and evalf(lambda2)<>0 then
#--------------
c2:=simplify(combine(c1-b1^2/(4*lambda1)-b2^2/(4*lambda2))):
QQ3:=lambda1*X1[1]^2+lambda2*X1[2]^2+c2:
ttr:=[xx=xx1*cosa-yy1*sina-simplify(combine(b1/(2*lambda1))),
yy=xx1*sina+yy1*cosa-simplify(combine(b2/(2*lambda2)))]:
x0:=-simplify(combine(b1/(2*lambda1))):
y0:=simplify(combine(b2/(2*lambda2))):
#--------------случай ненулевых lambda
if evalf(lambda1*lambda2)<0 and c2=0
then txt:=print("Пара пересекающихся прямых"):
EqF:=lambda1*X1[1]^2+lambda2*X1[2]^2=0:
aa:=NULL:bb:=NULL:cc:=NULL:ee:=NULL:
dd:=NULL:pp:=NULL:
aaa1:=sqrt(abs(lambda1)):
bbb1:=sqrt(abs(lambda2)):
GF:=AnalGeo[CrossDir1](aaa1,bbb1,cosa,sina,[x0,y0],red,blue):
elif evalf(lambda1*lambda2)<0 and evalf(lambda1*c2)<0
then
aa:=simplify(sqrt(abs(c2/lambda1))):
bb:=simplify(sqrt(abs(c2/lambda2))):
EqF:=X1[1]^2/aa^2-X1[2]^2/bb^2=1:
txt:=print("Гипербола с главной осью OX"):
cc:=simplify(sqrt(aa^2+bb^2)):
ee:=simplify(cc/aa):
dd:=simplify(aa/ee):
pp:=NULL:
GF:=AnalGeo[Hyp1](aa,bb,1,cosa,sina,[x0,y0],red,blue,0.5):
elif evalf(lambda1*lambda2<0) and evalf(lambda1*c2)>0
then
aa:=simplify(sqrt(abs(c2/lambda1))):
bb:=simplify(sqrt(abs(c2/lambda2))):
EqF:=-X1[1]^2/aa^2+X1[2]^2/bb^2=1:
txt:=print("Гипербола с главной осью OY"):
```

```
cc:=simplify(sqrt(aa^2+bb^2)):
ee:=simplify(cc/aa):
dd:=simplify(sqrt(aa/ee)):
pp:=NULL:
GF:=AnalGeo[Hyp1](aa,bb,-1,cosa,sina,[x0,y0],red,blue,0.5):
elif evalf(lambda1*lambda2)>0 and c2=0 then
txt:=print("????? (X=0,Y=0)"):
aa:=NULL:bb:=NULL:cc:=NULL:ee:=NULL:dd:=NULL:pp:=NULL:
GF:=AnalGeo[Point1](cosa,sina,[x0,y0],red):
elif evalf(lambda1*lambda2)>0 and evalf(lambda1*c2)<0
then
aa2:=evalf(abs(c2/lambda1)):
bb2:=evalf(abs(c2/lambda2)):
cc2:=simplify(aa^2-bb^2):
aa:=sqrt(abs(c2/lambda1)):
bb:=sqrt(abs(c2/lambda2)):
aaf:=evalf(aa):bbf:=evalf(bb):
cc:=simplify(sqrt(aa^2-bb^2)):
ee:=simplify(cc/aa):
dd:=simplify(aa/ee):
pp:=NULL:
EqF:=X1[1]^2/aa^2+X1[2]^2/bb^2=1:
if evalf(aa^2)>evalf(bb^2) then
txt:=print("Эллипс с главной осью OX"):
else txt:=print("Эллипс с главной осью OY"):
cc:=simplify(sqrt(bb^2-aa^2)):
ee:=simplify(cc/aa):
dd:=simplify(aa/ee):
pp:=NULL:
end if:
GF:=AnalGeo[Ell1](aa,bb,cosa,sina,[x0,y0],blue,red):
#----------------------
end if:
#-------------------конец подслучая ненулевых lambda
elif evalf(lambda1)=0 and evalf(lambda2)<>0 then
c2:=simplify(combine(c1-b2^2/(4*lambda2))):
#--------------------
if b1=0 then
```

```
QQ3:=lambda2*X1[2]^2+c2:
ttr:=[xx=xx1*cosa-yy1*sina,yy=xx1*sina+yy1*cosa+
simplify(combine(b2/(2*lambda2)))]:
x0:=0:
y0:=simplify(combine(b2/(2*lambda2))):
#----------------------------
if evalf(c2*lambda2)>0 then txt:=
print("Пустое множество"):
aa:=NULL:bb:=NULL:cc:=NULL:ee:=NULL:
dd:=NULL:pp:=NULL:EqF:=NULL:
GF:=AnalGeo[nullSet](cosa,sina,[x0,y0],red):
elif evalf(c2*lambda2)<0 then txt:=
print("Пара параллельных прямых"):
aa:=NULL:bb:=NULL:cc:=NULL:ee:=NULL:dd:=NULL:pp:=NULL:
aaa1:=sqrt(-c2/lambda2):
EqF:=[X1[2]=sqrt(-c2/lambda2),X1[2]=-sqrt(-c2/lambda2)]:
GF:=AnalGeo[ParalDir1](aaa1,cosa,sina,[x0,y0],1,blue,red):end if:
##конец случая b1=0-------------------------------------
elif evalf(b1)<>0 then
QQ3:=b1*xx1+lambda2*yy1^2:
x0:=simplify(combine(c2/b1)):
y0:=simplify(combine(b2/(2*lambda2))):
ttr:=[xx=xx1*cosa-yy1*sina+x0,yy=xx1*sina+yy1*cosa+y0]:
txt:=print("Парабола с главной осью OX1"):
aa:=NULL:bb:=NULL:cc:=NULL:ee:=1:
pp:=-b1/(2*lambda2):dd:=-pp/2:
EqF:=X1[2]^2=-b1*X1[1]/lambda2:
GF:=AnalGeo[Parbol1](pp,1,cosa,sina,[x0,y0],blue,red):
end if:
#----------------------------
elif evalf(lambda1)<>0 and evalf(lambda2)=0 then
c2:=simplify(combine(c1-b1^2/(4*lambda1))):
#--------------------
if b2=0 then
QQ3:=lambda1*X1[1]^2+c2:
ttr:=[xx=xx1*cosa-yy1*sina-simplify(combine(b1/(2*lambda1))),
yy=xx1*sina+yy1*cosa]:
x0:=-simplify(combine(b1/(2*lambda1))):
```

```
y0:=0:
if evalf(c2*lambda1)>0 then txt:=print("Пустое множество"):
aa:=NULL:bb:=NULL:cc:=NULL:ee:=NULL:dd:=NULL:pp:=NULL:
EqF:=NULL:
GF:=AnalGeo[nullSet](cosa,sina,[x0,y0],red):
elif evalf(c2*lambda1)<0 then txt:=print("Пара параллельных прямых"):
aaa1:=sqrt(-c2/lambda1):
EqF:=[X1[1]=sqrt(-c2/lambda1),X1[1]=-sqrt(-c2/lambda1)]:
GF:=AnalGeo[ParalDir1](aaa1,cosa,sina,[x0,y0],2,blue,red):
end if:
elif evalf(b2)<>0 then
QQ3:=b2*yy1+lambda1*xx1^2:
ttr:=[xx=xx1*cosa-yy1*sina+simplify(combine(b1/(2*lambda1))),
yy=xx1*sina+yy1*cosa+
simplify(combine(c2/b2))]:
x0:=simplify(combine(b1/(2*lambda1))):
y0:=simplify(combine(c2/b2)):
txt:=print("Парабола с главной осью OY1"):
aa:=NULL:bb:=NULL:
ee:=1:pp:=-b2/(2*lambda1):
ppf:=evalf(-b2/(2*lambda1)):
dd:=-pp/2:cc:=NULL:
EqF:=X1[1]^2=-b2*X1[2]/lambda1:
GF:=AnalGeo[Parbol1](ppf,-1,cosa,sina,[x0,y0],blue,red):
end if:
#---------------------------
end if:
AAA:=array(1..8,1..3,sparse):
AAA[1,1]:=[lambda[1],lambda[2]]:
AAA[1,2]:=‘=‘:AAA[1,3]:=[lambda1,lambda2]:
AAA[2,1]:=‘Уравнение‘:AAA[2,2]:=‘:‘:
AAA[2,3]:=EqF:AAA[3,1]:=‘x’‘:
AAA[3,2]:=‘=‘:AAA[3,3]:=rhs(ttr[1]):
AAA[4,1]:=‘y’‘:AAA[4,2]:=‘=‘:
AAA[4,3]:=rhs(ttr[2]):AAA[5,1]:=[c,epsilon,d]:
AAA[5,2]:=‘=‘:AAA[5,3]:=[cc,ee,dd]:
AAA[6,1]:=[‘a‘,‘b‘,‘p‘]:AAA[6,2]:=‘=‘:
AAA[6,3]:=[aa,bb,pp]:AAA[7,1]:=[cos(alpha),sin(alpha)]:
```

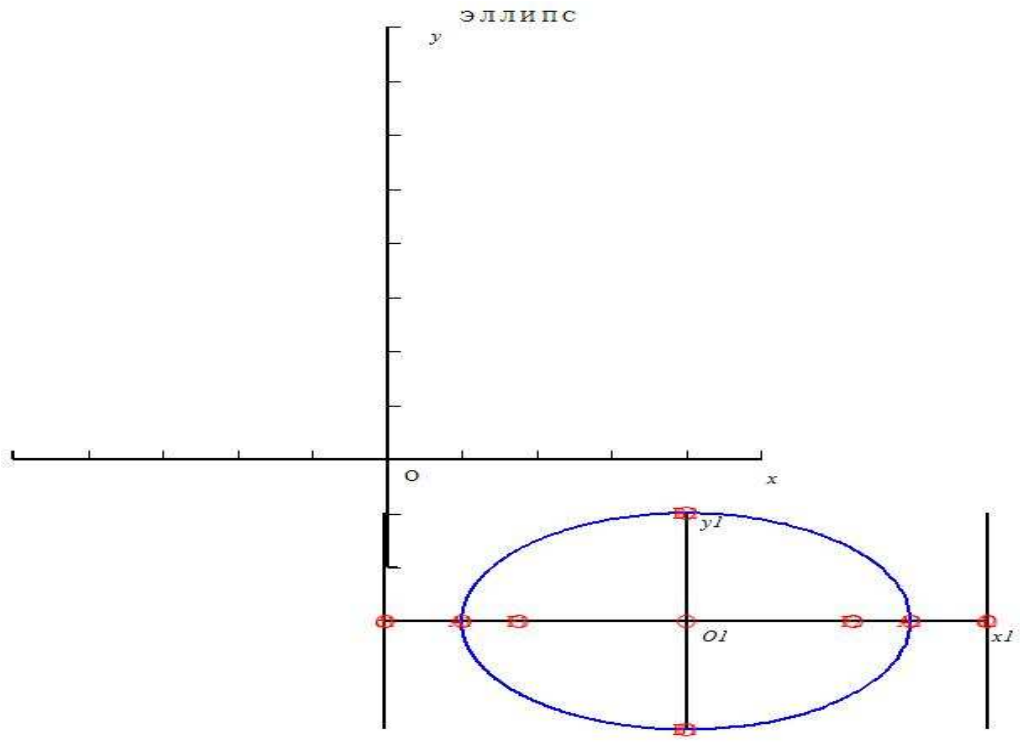
```
AAA[7,2]:='=':AAA[7,3]:=[cosa,sina]:
AAA[8,1]:=[x[0],y[0]]:AAA[8,2]:='=':
AAA[8,3]:=[x0,y0]:print(AAA),GF:
end proc:
```

II.4.9 Примеры

```
> AnalGeo[CanonF](4*x^2+\
> 9*y^2-32*x-54*y+109=0,[x,y],[xi,eta],alpha);
```

$$OX''$$

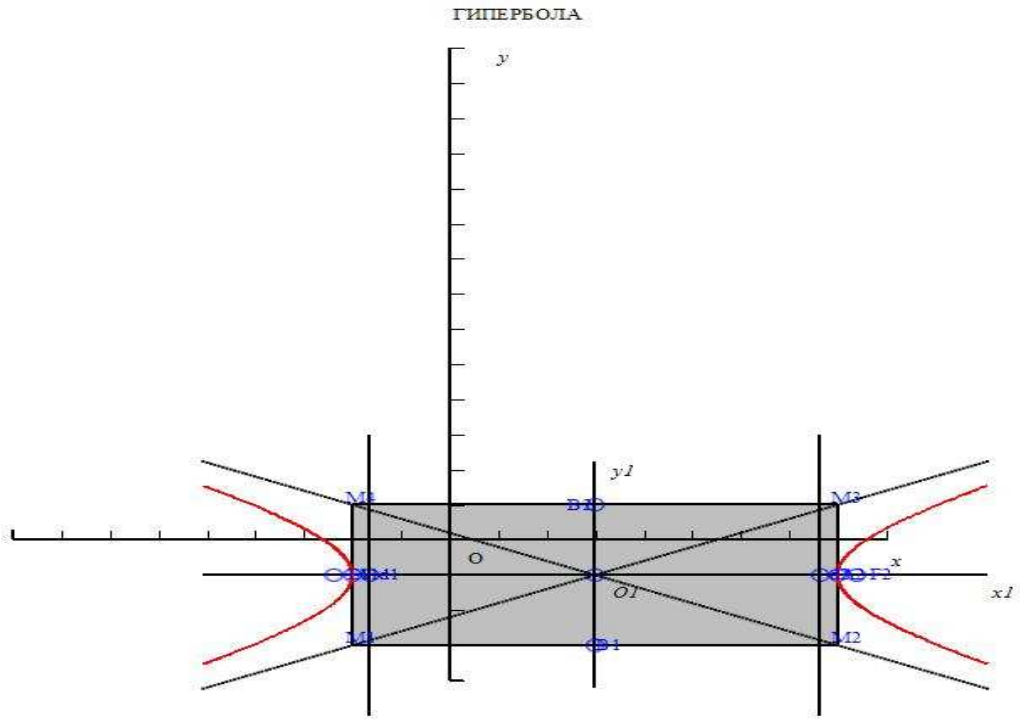
$$\begin{bmatrix} [\lambda_1, \lambda_2] & = & [4, 9] \\ \text{Уравнение} & : & 1/9\,\xi^2 + 1/4\,\eta^2 = 1 \\ x' & = & \xi + 4 \\ y' & = & 3 + \eta \\ [c, \epsilon, d] & = & [\sqrt{5}, 1/3\,\sqrt{5}, 9/5\,\sqrt{5}] \\ [a, b, p] & = & [3, 2] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [4, -3] \end{bmatrix}$$




```
> AnalGeo[CanonF](4*x^2-25*y^2-24*x+50*y-\
> 89=0,[x,y],[xi,eta],alpha);
```

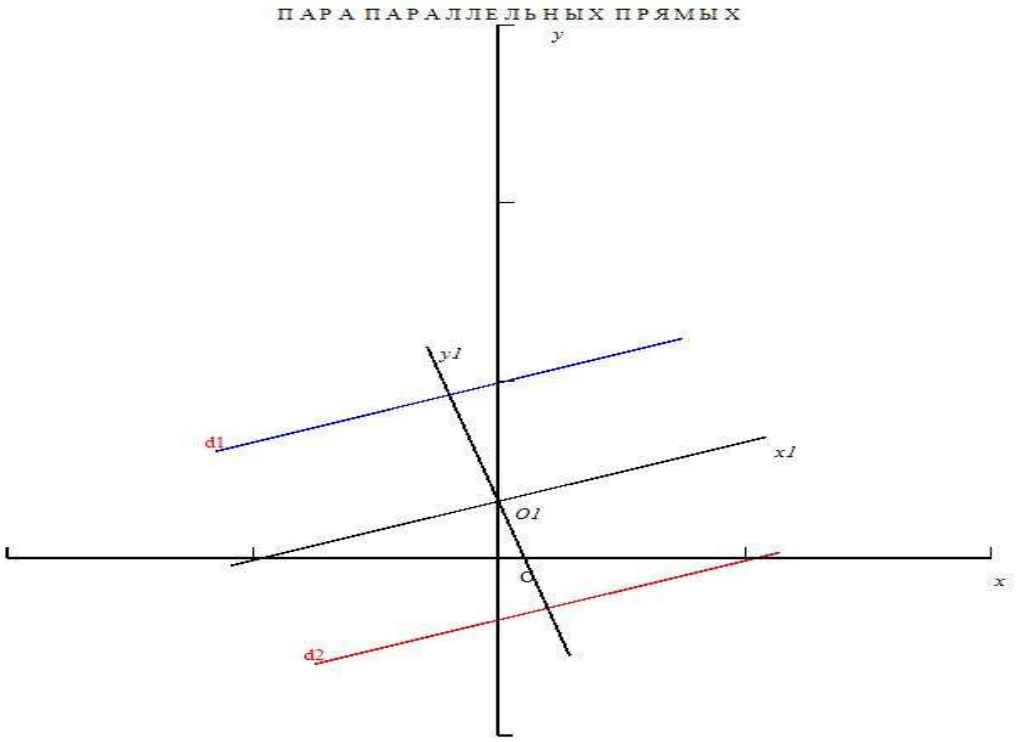
$$OX''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [4, -25] \\ \texttt{Уравнение} & : & 1/25\,\xi^2 - 1/4\,\eta^2 = 1 \\ x' & = & \xi + 3 \\ y' & = & 1 + \eta \\ [c, \epsilon, d] & = & [\sqrt{29}, 1/5\,\sqrt{29}, \frac{25}{29}\,\sqrt{29}] \\ [a, b, p] & = & [5, 2] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [3, -1] \end{array}\right]$$




```
> AnalGeo[CanonF](x^2-6*x*y+9*y^2-2*x+\
> 6*y-3=0,[x,y],[xi,eta],alpha);
                                    ''
```

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [0, 10] \\ \texttt{Уравнение} & : & [\eta = 1/5\sqrt{10}, \eta = -1/5\sqrt{10}] \\ x' & = & 3/10\,\xi\sqrt{2}\sqrt{5} - 1/10\,\eta\sqrt{2}\sqrt{5} \\ y' & = & 1/10\,\xi\sqrt{2}\sqrt{5} + 3/10\,\eta\sqrt{2}\sqrt{5} + 1/10\sqrt{10} \\ [c, \epsilon, d] & = & [] \\ [a, b, p] & = & [] \\ [\cos(\alpha), \sin(\alpha)] & = & [3/10\sqrt{2}\sqrt{5}, 1/10\sqrt{2}\sqrt{5}] \\ [x_0, y_0] & = & [0, 1/10\sqrt{10}] \end{array}\right]$$



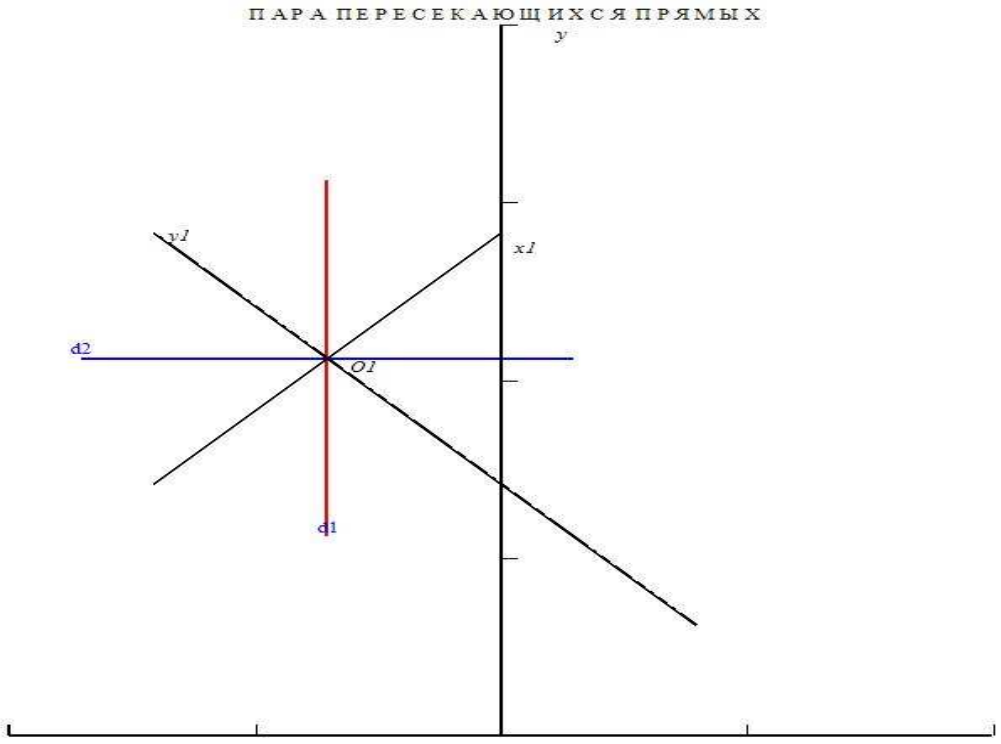

```
> QQQ19:=x*y+2*x-y-2=0;
```

$$QQQ19 := xy + 2\,x - y - 2 = 0$$

```
> AnalGeo[CanonF](QQQ19,[x,y],[xi,eta],alpha);
```

″

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1/2, -1/2] \\ \text{Уравнение} & : & 1/2\,\xi^2 - 1/2\,\eta^2 = 0 \\ x' & = & 1/2\,\xi\,\sqrt{2} - 1/2\,\eta\,\sqrt{2} - 1/2\,\sqrt{2} \\ y' & = & 1/2\,\xi\,\sqrt{2} + 1/2\,\eta\,\sqrt{2} - 3/2\,\sqrt{2} \\ [c, \epsilon, d] & = & [] \\ [a, b, p] & = & [] \\ [\cos(\alpha), \sin(\alpha)] & = & [1/2\,\sqrt{2}, 1/2\,\sqrt{2}] \\ [x_0, y_0] & = & [-1/2\,\sqrt{2}, 3/2\,\sqrt{2}] \end{array}\right]$$



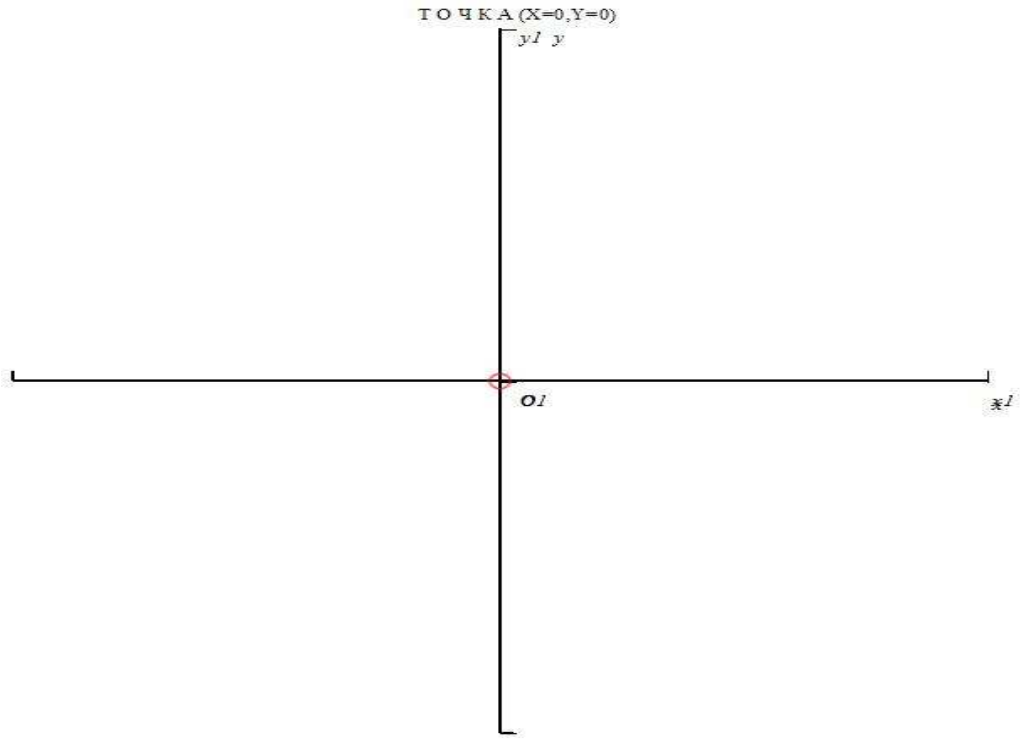

```
> QQQ20:=x^2+4*y^2=0;
```

$$QQQ20 := x^2 + 4\,y^2 = 0$$

```
> AnalGeo[CanonF](QQQ20,[x,y],[xi,eta],alpha);
```

$$(X = 0, Y = 0)''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1, 4] \\ \text{Уравнение} & : & EqF \\ x' & = & \xi \\ y' & = & \eta \\ [c, \epsilon, d] & = & [] \\ [a, b, p] & = & [] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [0, 0] \end{array}\right]$$




```
> AnalGeo[CanonF](x^2+9*y^2+3=0,[x,y],[xi,eta],alpha);
```

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1, 9] \\ \text{Уравнение} & : & EqF \\ x' & = & \xi \\ y' & = & \eta \\ [c, \epsilon, d] & = & [cc, ee, dd] \\ [a, b, p] & = & [aa, bb, pp] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [0, 0] \end{array}\right]$$

$$GF$$

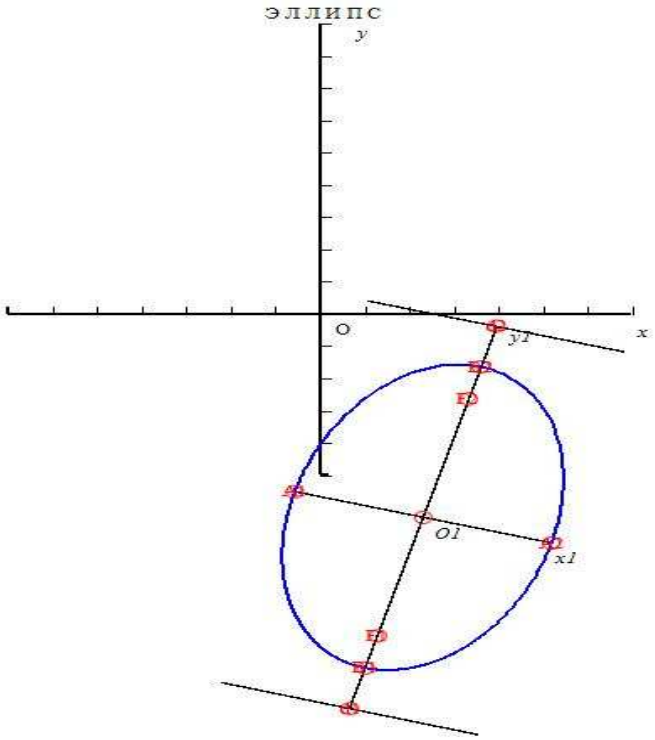
```
> qqqqq1:=9*x^2-3*x*y+4*y^2-54*x-32*y+109=0;
```

$$qqqqq1 := 9\,x^2 - 3\,xy + 4\,y^2 - 54\,x - 32\,y + 109 = 0$$

```
> AnalGeo[CanonF](9*x^2-3*x*y+4*y^2-54*x-32*y+\
> 109=0,[x,y],[xi,eta],alpha);
```

$$OY''$$

$$\left[\begin{array}{lcl}
[\lambda_1, \lambda_2] & = & [13/2 + 1/2\sqrt{34}, 13/2 - 1/2\sqrt{34}] \\
\text{Уравнение} & : & \left(\frac{15}{194} + \frac{15}{2522}\sqrt{34}\right)\xi^2 + \left(\frac{15}{194} - \frac{15}{2522}\sqrt{34}\right)\eta^2 = 1 \\
x' & = & \frac{1}{68}\,\xi\sqrt{2}\sqrt{1156 + 170\sqrt{34}} + \frac{1}{68}\,\eta\sqrt{2}\sqrt{1156 - 170\sqrt{34}} + 1/17\,\frac{27\sqrt{578+85\sqrt{34}} - 16\sqrt{578-85\sqrt{34}}}{13+\sqrt{34}} \\
y' & = & -\frac{1}{68}\,\xi\sqrt{2}\sqrt{1156 - 170\sqrt{34}} + \frac{1}{68}\,\eta\sqrt{2}\sqrt{1156 + 170\sqrt{34}} - 1/17\,\frac{16\sqrt{578+85\sqrt{34}} + 27\sqrt{578-85\sqrt{34}}}{-13+\sqrt{34}} \\
[c, \epsilon, d] & = & [\frac{2}{45}\sqrt{1261}\sqrt[4]{34}, 1/45\sqrt[4]{34}\sqrt{15}\sqrt{26 + 2\sqrt{34}}, \frac{3}{34}\,\frac{\sqrt{1261}34^{3/4}}{13+\sqrt{34}}] \\
[a, b, p] & = & [2/15\,\frac{\sqrt{18915}}{\sqrt{26+2\sqrt{34}}}, 2/15\,\frac{\sqrt{18915}}{\sqrt{26-2\sqrt{34}}}] \\
[\cos(\alpha), \sin(\alpha)] & = & [\frac{1}{68}\sqrt{2}\sqrt{1156 + 170\sqrt{34}}, -\frac{1}{68}\sqrt{2}\sqrt{1156 - 170\sqrt{34}}] \\
[x_0, y_0] & = & [1/17\,\frac{27\sqrt{578+85\sqrt{34}} - 16\sqrt{578-85\sqrt{34}}}{13+\sqrt{34}}, 1/17\,\frac{16\sqrt{578+85\sqrt{34}} + 27\sqrt{578-85\sqrt{34}}}{-13+\sqrt{34}}]
\end{array}\right]$$




```
> qqq1:=5*x^2+4*x*y+8*y^2+8*x+14*y+5=0;
```

$$qqq1 := 5\,x^2 + 4\,xy + 8\,y^2 + 8\,x + 14\,y + 5 = 0$$

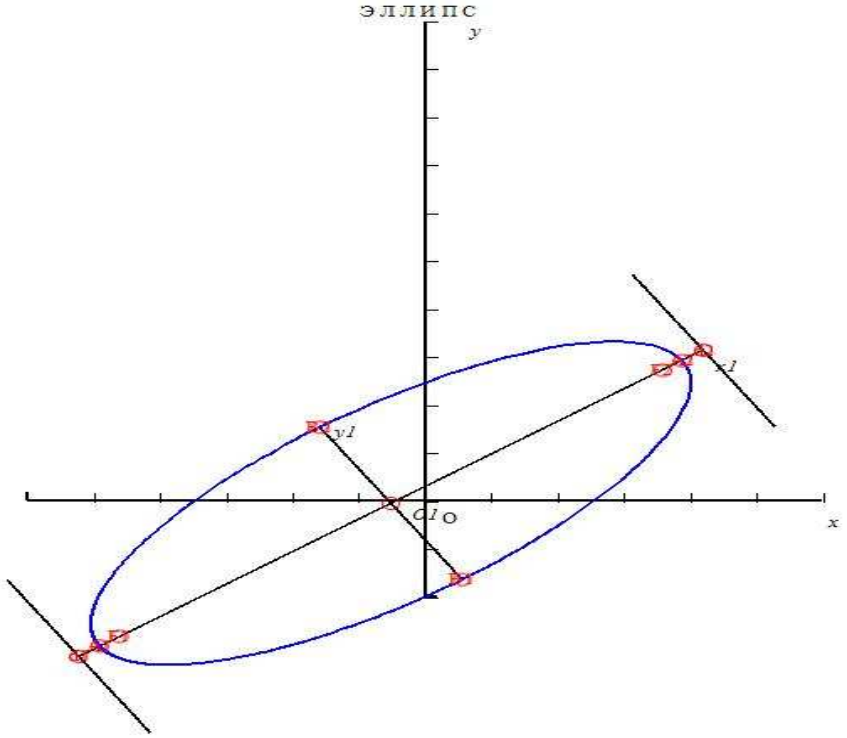
```
> qqqqq:=5*x^2-10*x*y+9*y^2+2*x-45=0;
```

$$qqqqq := 5\,x^2 - 10\,xy + 9\,y^2 + 2\,x - 45 = 0$$

```
> AnalGeo[CanonF](qqqqq,[x,y],[xi,eta],alpha);
```

$$OX''$$

$$\left[\begin{array}{ccc}
[\lambda_1,\lambda_2] & = & [7-\sqrt{29},7+\sqrt{29}] \\
\texttt{Уравнение} & : & \left(\frac{140}{909}-\frac{20}{909}\sqrt{29}\right)\xi^2+\left(\frac{140}{909}+\frac{20}{909}\sqrt{29}\right)\eta^2=1 \\
x' & = & \frac{1}{58}\,\xi\,\sqrt{2}\sqrt{841+58\sqrt{29}}-\frac{1}{58}\,\eta\,\sqrt{2}\sqrt{841-58\sqrt{29}}+\frac{1}{58}\,\frac{\sqrt{1682+116\sqrt{29}}}{-7+\sqrt{29}} \\
y' & = & \frac{1}{58}\,\xi\,\sqrt{2}\sqrt{841-58\sqrt{29}}+\frac{1}{58}\,\eta\,\sqrt{2}\sqrt{841+58\sqrt{29}}+\frac{1}{58}\,\frac{\sqrt{1682-116\sqrt{29}}}{7+\sqrt{29}} \\
[c,\epsilon,d] & = & [\frac{3}{20}\,\sqrt{202}\sqrt[4]{29},1/10\,\sqrt{2}\sqrt[4]{29}\sqrt{5}\sqrt{7-\sqrt{29}},-\frac{3}{58}\,\frac{\sqrt{101}\sqrt{2}29^{3/4}}{-7+\sqrt{29}}] \\
[a,b,p] & = & [3/10\,\frac{\sqrt{505}}{\sqrt{7-\sqrt{29}}},3/10\,\frac{\sqrt{505}}{\sqrt{7+\sqrt{29}}}] \\
[\cos(\alpha),\sin(\alpha)] & = & [\frac{1}{58}\,\sqrt{2}\sqrt{841+58\sqrt{29}},\frac{1}{58}\,\sqrt{2}\sqrt{841-58\sqrt{29}}] \\
[x_0,y_0] & = & [\frac{1}{58}\,\frac{\sqrt{1682+116\sqrt{29}}}{-7+\sqrt{29}},-\frac{1}{58}\,\frac{\sqrt{1682-116\sqrt{29}}}{7+\sqrt{29}}]
\end{array}\right]$$




```
> evalf(45*29^(1/2));
```

$$242.3324163$$

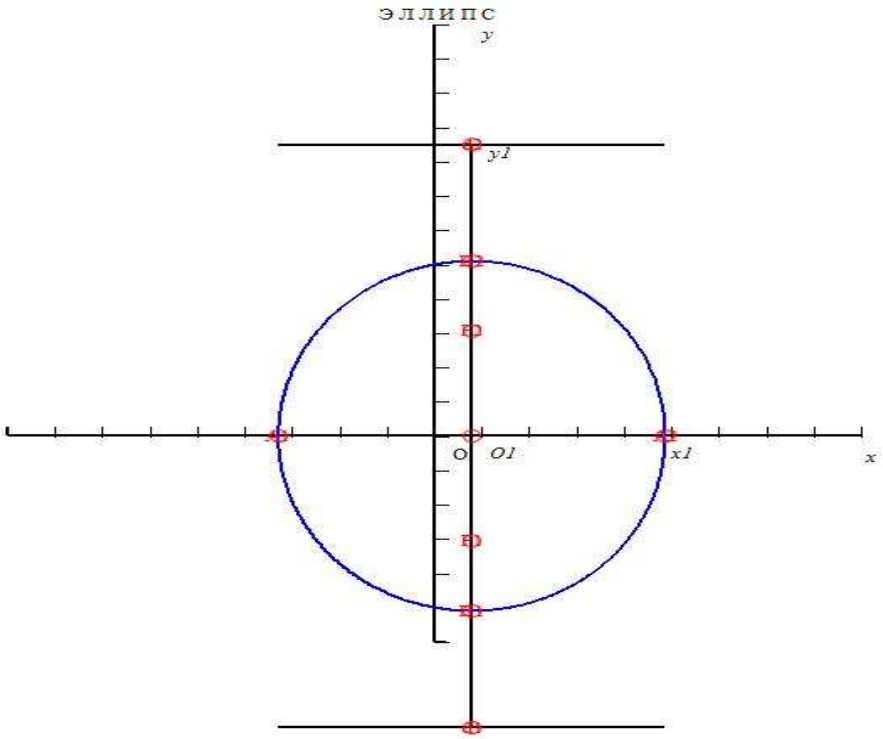
```
> qq15:=25*x^2+16*y^2-40*x-400=0;
```

$$qq15 := 25\,x^2+16\,y^2-40\,x-400=0$$

```
> AnalGeo[CanonF](qq15,[x,y],[xi,eta],alpha);
```

$$OY''$$

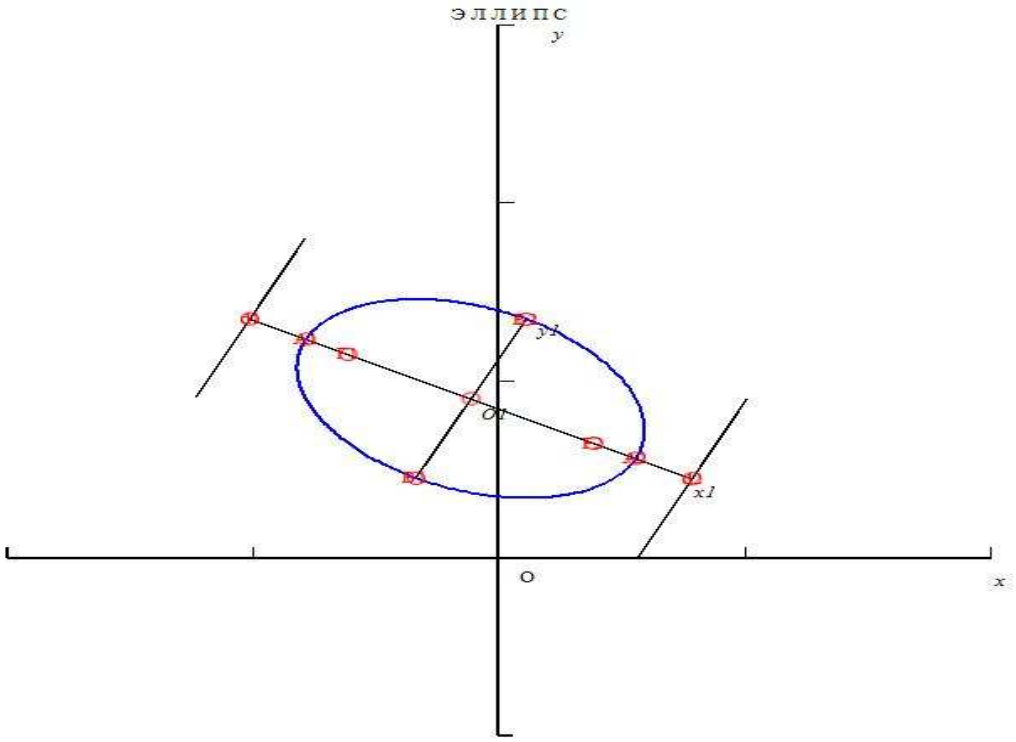
$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [25, 16] \\ \texttt{Уравнение} & : & \frac{25}{416}\,\xi^2 + 1/26\,\eta^2 = 1 \\ x' & = & \xi + 4/5 \\ y' & = & \eta \\ [c, \epsilon, d] & = & [3/5\,\sqrt{26}, 3/4, \frac{16}{15}\,\sqrt{26}] \\ [a, b, p] & = & [4/5\,\sqrt{26}, \sqrt{26}] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [4/5, 0] \end{array}\right]$$




```
> AnalGeo[CanonF](5*x^2+4*x*y+\
> 8*y^2+8*x+14*y+5=0,[x,y],[xi,eta],alpha);
```

$$OX''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [4, 9] \\ \texttt{Уравнение} & : & \frac{16}{9}\,\xi^2 + 4\,\eta^2 = 1 \\ x' & = & 1/5\,\xi\,\sqrt{2}\sqrt{10} + 1/10\,\eta\,\sqrt{2}\sqrt{10} - 1/20\,\sqrt{5} \\ y' & = & -1/10\,\xi\,\sqrt{2}\sqrt{10} + 1/5\,\eta\,\sqrt{2}\sqrt{10} - 2/5\,\sqrt{5} \\ [c, \epsilon, d] & = & [1/4\,\sqrt{5}, 1/3\,\sqrt{5}, \frac{9}{20}\,\sqrt{5}] \\ [a, b, p] & = & [3/4, 1/2] \\ [\cos(\alpha), \sin(\alpha)] & = & [1/5\,\sqrt{2}\sqrt{10}, -1/10\,\sqrt{2}\sqrt{10}] \\ [x_0, y_0] & = & [-1/20\,\sqrt{5}, 2/5\,\sqrt{5}] \end{array}\right]$$

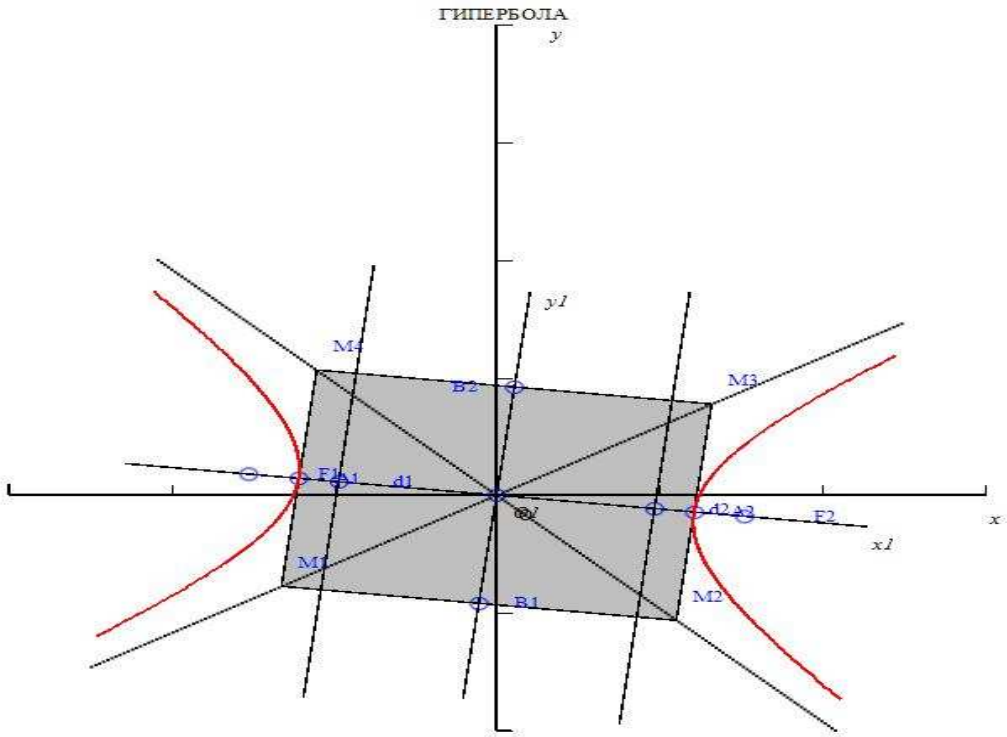



```
> qq81:=9*x^2-6*x*y-16*y^2-14=0;
```

$$qq81 := 9\,x^2 - 6\,xy - 16\,y^2 - 14 = 0$$

```
> AnalGeo[CanonF](qq81,[x,y],[xi,eta],alpha);
```

$$OX''$$

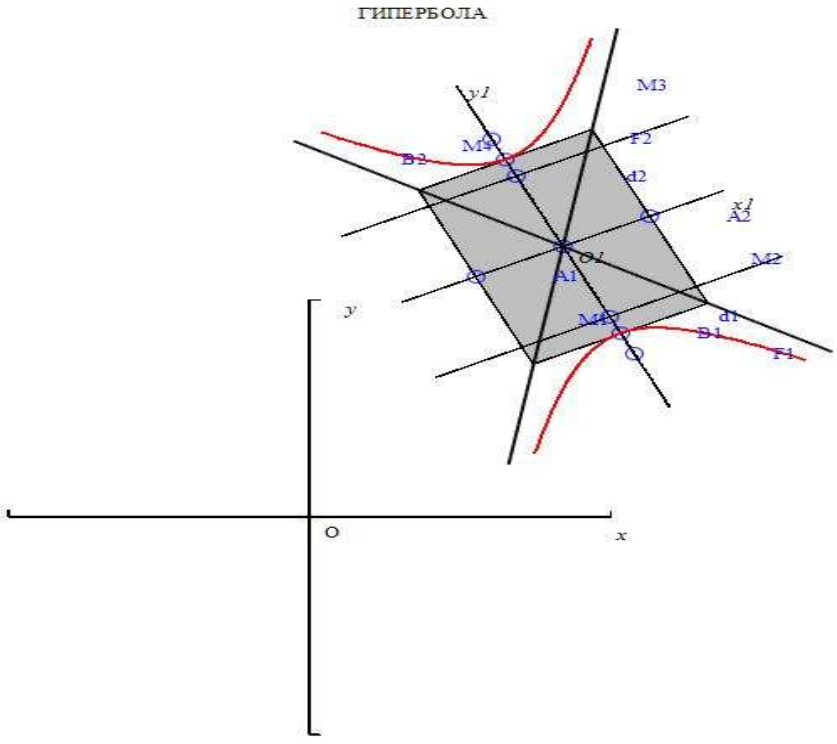



```
> qqqqqq:=3*x^2+5*x*y-y^2-4*x-8*y=0;
```

$$qqqqqq := 3\,x^2 + 5\,xy - y^2 - 4\,x - 8\,y = 0$$

```
> AnalGeo[CanonF](qqqqqq,[x,y],[xi,eta],alpha);
```

$$OY''$$



Пример из учебника Гусак А.А. с.80

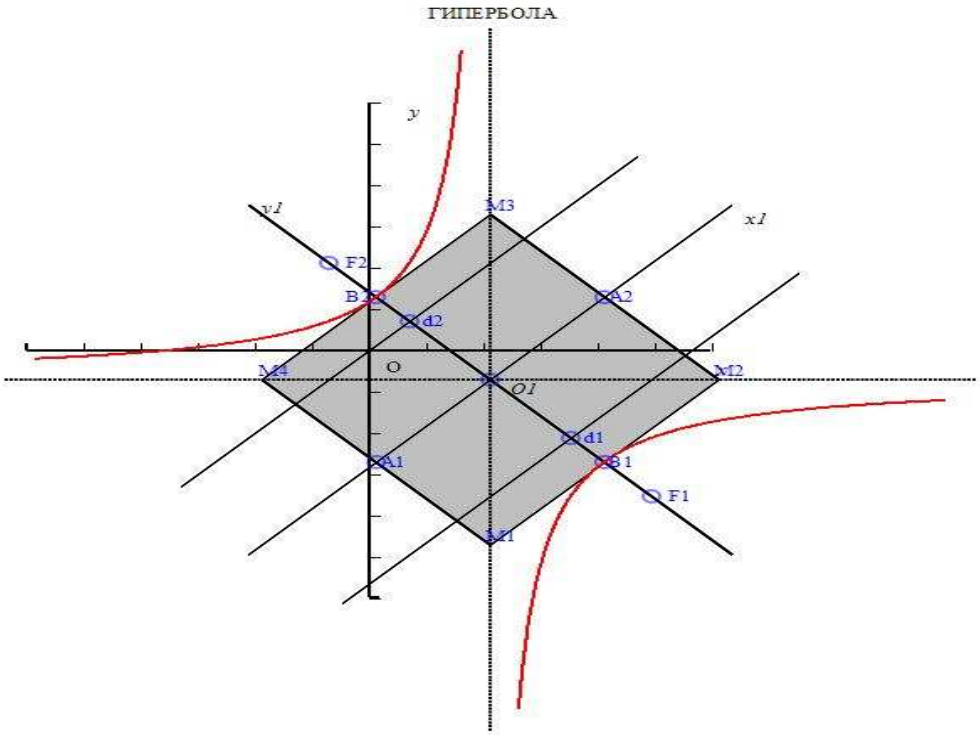
```
> QQ5:=x*y-2*x-y+6=0;
```

$$QQ5 := xy - 2\,x - y + 6 = 0$$

```
> AnalGeo[CanonF](QQ5,[x,y],[xi,eta],alpha);
```

$$OY''$$

$$\begin{bmatrix} [\lambda_1, \lambda_2] & = & [1/2, -1/2] \\ \text{Уравнение} & : & -1/8\,\xi^2 + 1/8\,\eta^2 = 1 \\ x' & = & 1/2\,\xi\,\sqrt{2} - 1/2\,\eta\,\sqrt{2} + 3/2\,\sqrt{2} \\ y' & = & 1/2\,\xi\,\sqrt{2} + 1/2\,\eta\,\sqrt{2} + 1/2\,\sqrt{2} \\ [c, \epsilon, d] & = & [4, \sqrt{2}, \sqrt{2}] \\ [a, b, p] & = & [2\,\sqrt{2}, 2\,\sqrt{2}] \\ [\cos(\alpha), \sin(\alpha)] & = & [1/2\,\sqrt{2}, 1/2\,\sqrt{2}] \\ [x_0, y_0] & = & [3/2\,\sqrt{2}, -1/2\,\sqrt{2}] \end{bmatrix}$$



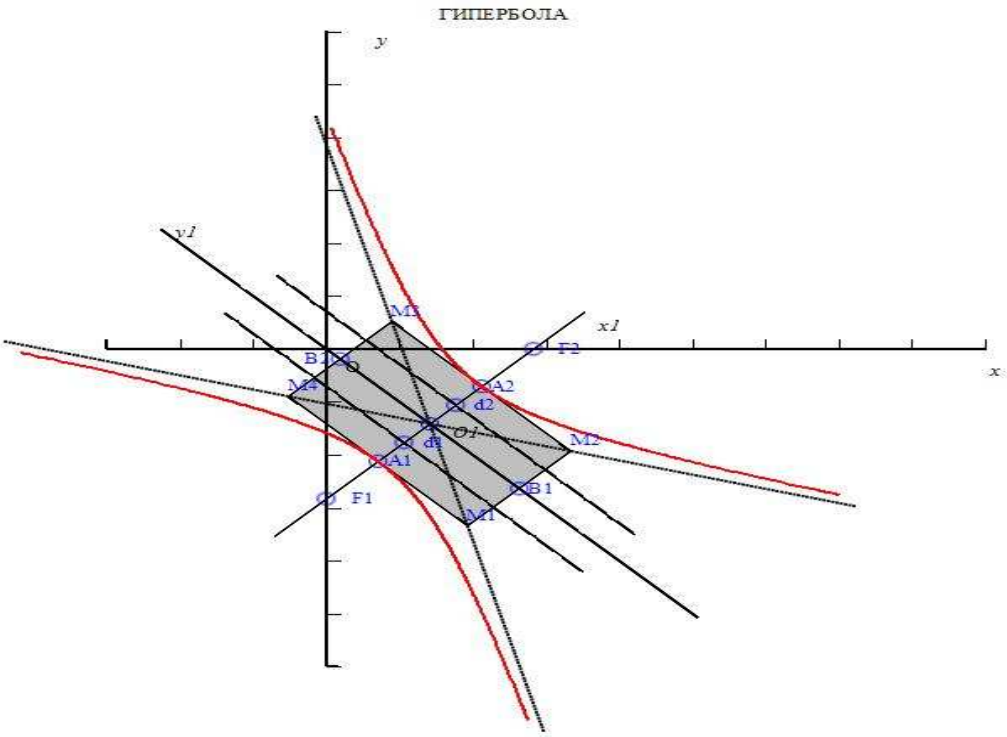

```
> QQ6:=x^2+y^2+4*x*y-8*x-4*y+1=0;
```

$$QQ6 := x^2 + 4\,xy + y^2 - 8\,x - 4\,y + 1 = 0$$

```
> AnalGeo[CanonF](QQ6,[x,y],[xi,eta],alpha);
```

$$OX''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [3, -1] \\ \text{Уравнение} & : & \xi^2 - 1/3\,\eta^2 = 1 \\ x' & = & 1/2\,\xi\,\sqrt{2} - 1/2\,\eta\,\sqrt{2} + \sqrt{2} \\ y' & = & 1/2\,\xi\,\sqrt{2} + 1/2\,\eta\,\sqrt{2} + \sqrt{2} \\ [c, \epsilon, d] & = & [2, 2, 1/2] \\ [a, b, p] & = & [1, \sqrt{3}] \\ [\cos(\alpha), \sin(\alpha)] & = & [1/2\,\sqrt{2}, 1/2\,\sqrt{2}] \\ [x_0, y_0] & = & [\sqrt{2}, -\sqrt{2}] \end{array}\right]$$



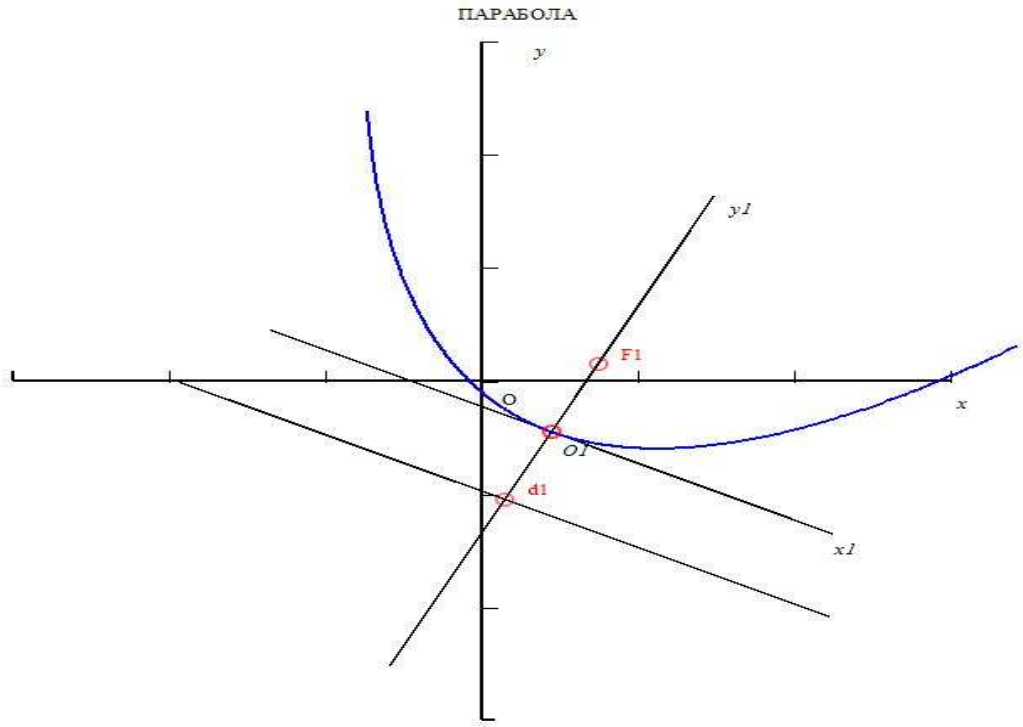

```
> QQ6:=4*x^2-4*x*y+y^2-2*x-14*y+7=0;
```

$$QQ6 := 4\,x^2 - 4\,xy + y^2 - 2\,x - 14\,y + 7 = 0$$

```
> AnalGeo[CanonF](QQ6,[x,y],[xi,eta],alpha);
```

$$OY1''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [5, 0] \\ \texttt{Уравнение} & : & \xi^2 = 6/5\sqrt{5}\eta \\ x' & = & 1/5\,\xi\sqrt{2}\sqrt{10} + 1/10\,\eta\sqrt{2}\sqrt{10} + 1/5\sqrt{5} \\ y' & = & -1/10\,\xi\sqrt{2}\sqrt{10} + 1/5\,\eta\sqrt{2}\sqrt{10} - 1/5\sqrt{5} \\ [c, \epsilon, d] & = & [1, -3/10\sqrt{5}] \\ [a, b, p] & = & [3/5\sqrt{5}] \\ [\cos(\alpha), \sin(\alpha)] & = & [1/5\sqrt{2}\sqrt{10}, -1/10\sqrt{2}\sqrt{10}] \\ [x_0, y_0] & = & [1/5\sqrt{5}, -1/5\sqrt{5}] \end{array}\right]$$




```
> q1:=x^2-6*x*y+9*y^2-2*x+6*y-3=0;
```

$$q1 := x^2 - 6\,xy + 9\,y^2 - 2\,x + 6\,y - 3 = 0$$

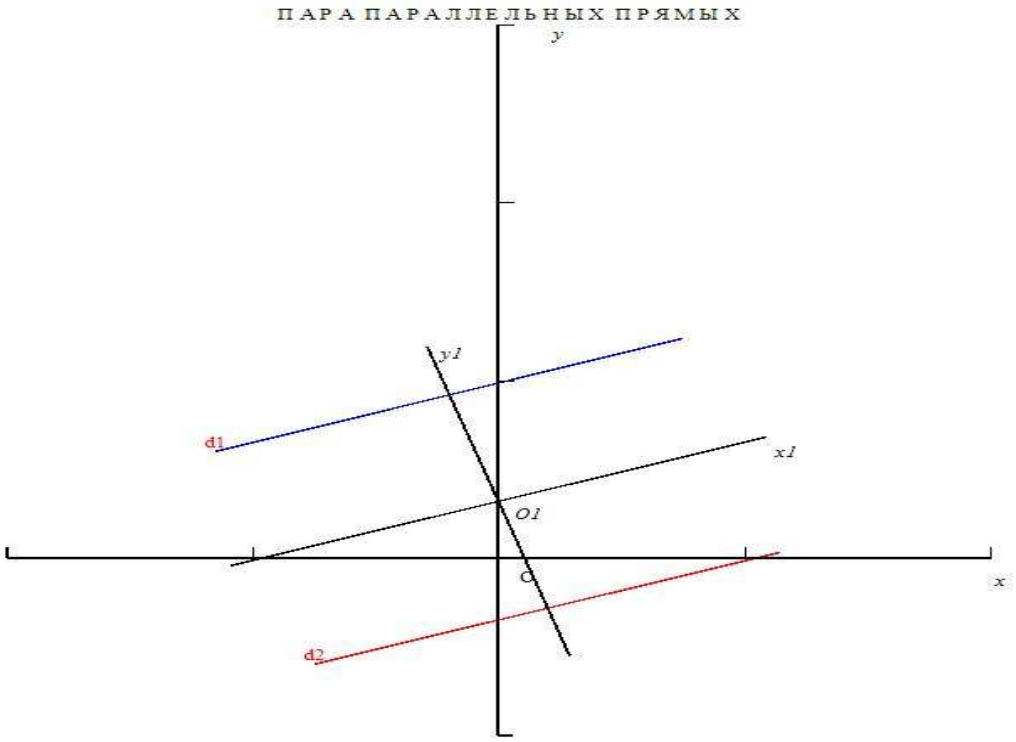
```
> Zam1222:=AnalGeo[Angle2](q1,[x,y],alpha);
```

$$Zam1222 := [\cos(\alpha) = 3/10\sqrt{2}\sqrt{5}, \sin(\alpha) = 1/10\sqrt{2}\sqrt{5}]$$

```
> AnalGeo[CanonF](x^2-6*x*y+\
> 9*y^2-2*x+6*y-3=0,[x,y],[xi,eta],alpha);
```

″

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [0, 10] \\ \texttt{Уравнение} & : & [\eta = 1/5\sqrt{10}, \eta = -1/5\sqrt{10}] \\ x' & = & 3/10\,\xi\sqrt{2}\sqrt{5} - 1/10\,\eta\sqrt{2}\sqrt{5} \\ y' & = & 1/10\,\xi\sqrt{2}\sqrt{5} + 3/10\,\eta\sqrt{2}\sqrt{5} + 1/10\sqrt{10} \\ [c, \epsilon, d] & = & [] \\ [a, b, p] & = & [] \\ [\cos(\alpha), \sin(\alpha)] & = & [3/10\sqrt{2}\sqrt{5}, 1/10\sqrt{2}\sqrt{5}] \\ [x_0, y_0] & = & [0, 1/10\sqrt{10}] \end{array}\right]$$



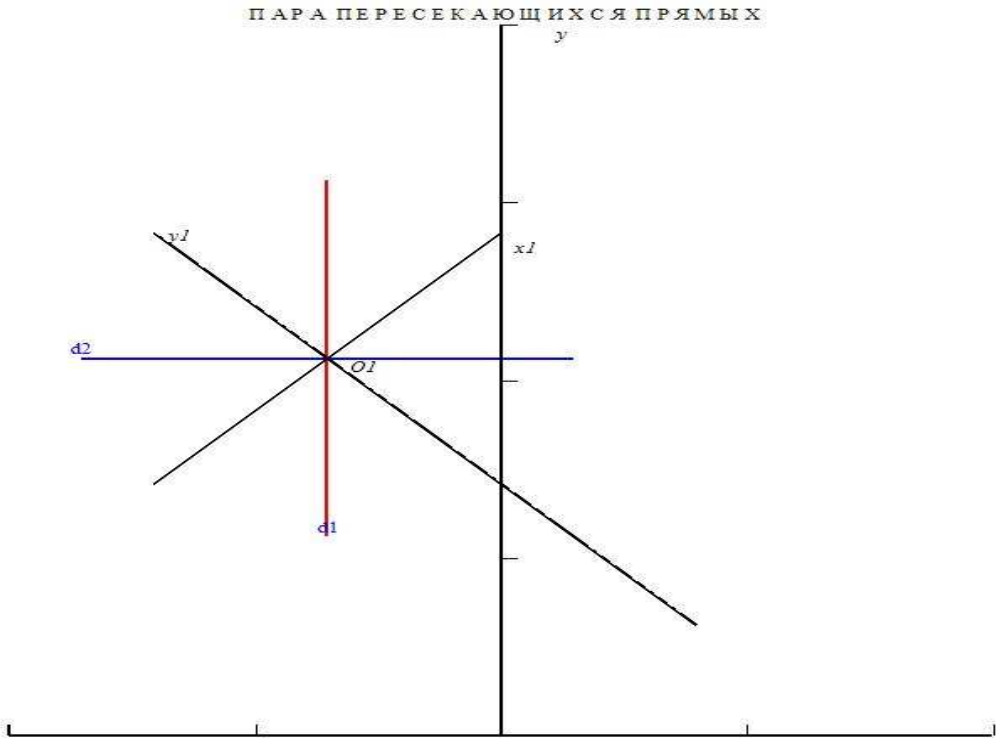

```
> QQQ19:=x*y+2*x-y-2=0;
```

$$QQQ19 := xy + 2\,x - y - 2 = 0$$

```
> AnalGeo[CanonF](QQQ19,[x,y],[xi,eta],alpha);
```

$$''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1/2, -1/2] \\ \text{Уравнение} & : & 1/2\,\xi^2 - 1/2\,\eta^2 = 0 \\ x' & = & 1/2\,\xi\,\sqrt{2} - 1/2\,\eta\,\sqrt{2} - 1/2\,\sqrt{2} \\ y' & = & 1/2\,\xi\,\sqrt{2} + 1/2\,\eta\,\sqrt{2} - 3/2\,\sqrt{2} \\ [c, \epsilon, d] & = & [] \\ [a, b, p] & = & [] \\ [\cos(\alpha), \sin(\alpha)] & = & [1/2\,\sqrt{2}, 1/2\,\sqrt{2}] \\ [x_0, y_0] & = & [-1/2\,\sqrt{2}, 3/2\,\sqrt{2}] \end{array}\right]$$



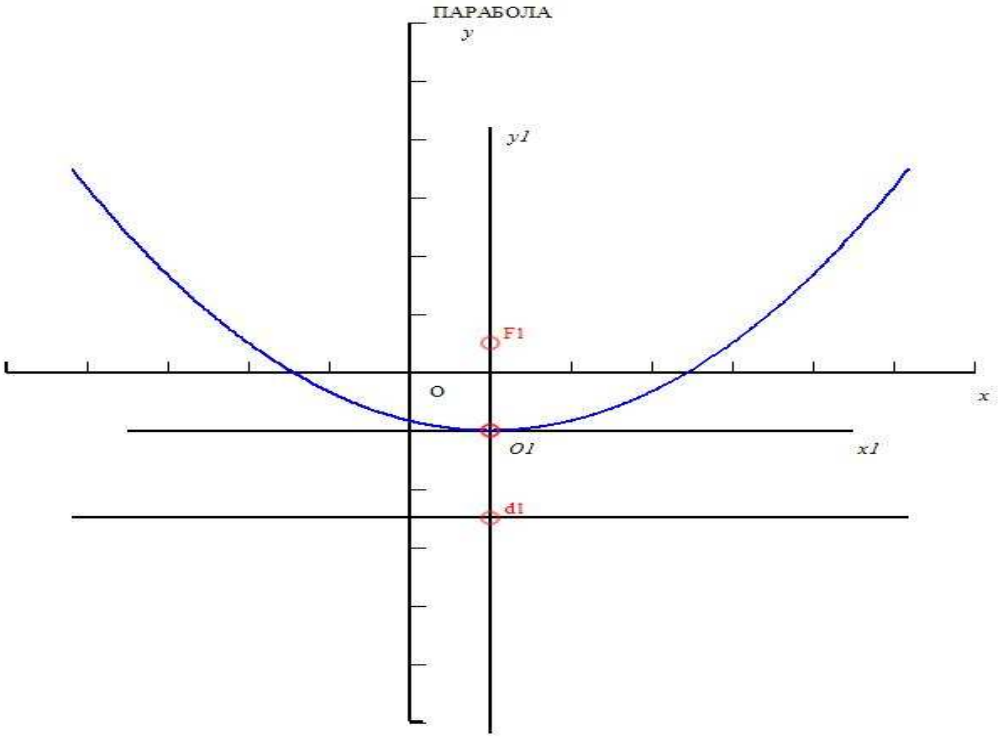

```
> QQQ22:=x^2+2*x-6*y+7=0;
```

$$QQQ22 := x^2 + 2\,x - 6\,y + 7 = 0$$

```
> AnalGeo[CanonF](QQQ22,[x,y],[xi,eta],alpha);
```

$$OY1''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1, 0] \\ \text{Уравнение} & : & \xi^2 = 6\,\eta \\ x' & = & \xi + 1 \\ y' & = & -1 + \eta \\ [c, \epsilon, d] & = & [1, -3/2] \\ [a, b, p] & = & [3] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [1, -1] \end{array}\right]$$




```
> QQQ23:=y^2+3*x+7=0;
```

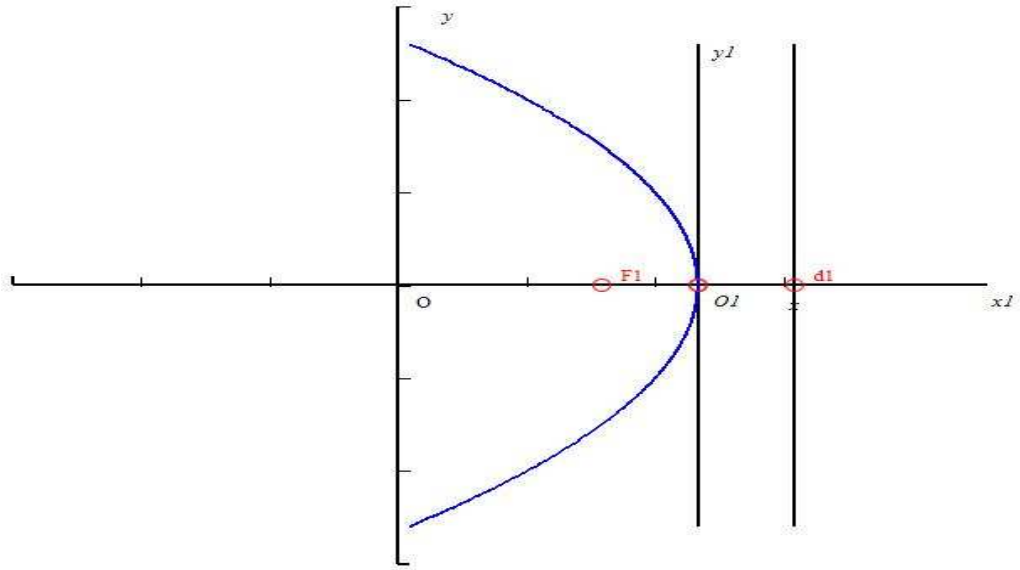
$$QQQ23 := y^2 + 3\,x + 7 = 0$$

```
> AnalGeo[CanonF](QQQ23,[x,y],[xi,eta],alpha);
```

$$OX1''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [0, 1] \\ \text{Уравнение} & : & \eta^2 = -3\,\xi \\ x' & = & \xi + 7/3 \\ y' & = & \eta \\ [c, \epsilon, d] & = & [1, 3/4] \\ [a, b, p] & = & [-3/2] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [7/3, 0] \end{array}\right]$$

ПАРАБОЛА



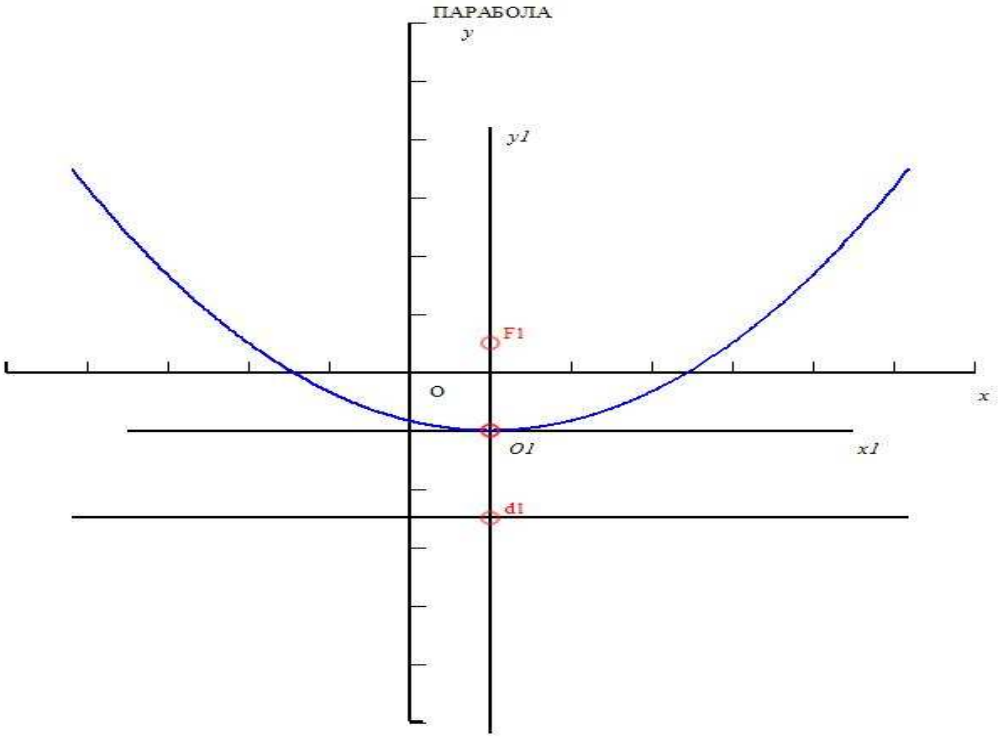

```
> QQ7:=x^2+2*x-6*y+7=0;
```

$$QQ7 := x^2 + 2\,x - 6\,y + 7 = 0$$

```
> AnalGeo[CanonF](QQ7,[x,y],[xi,eta],alpha);
```

$$OY1''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1, 0] \\ \texttt{Уравнение} & : & \xi^2 = 6\,\eta \\ x' & = & \xi + 1 \\ y' & = & -1 + \eta \\ [c, \epsilon, d] & = & [1, -3/2] \\ [a, b, p] & = & [3] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [1, -1] \end{array}\right]$$



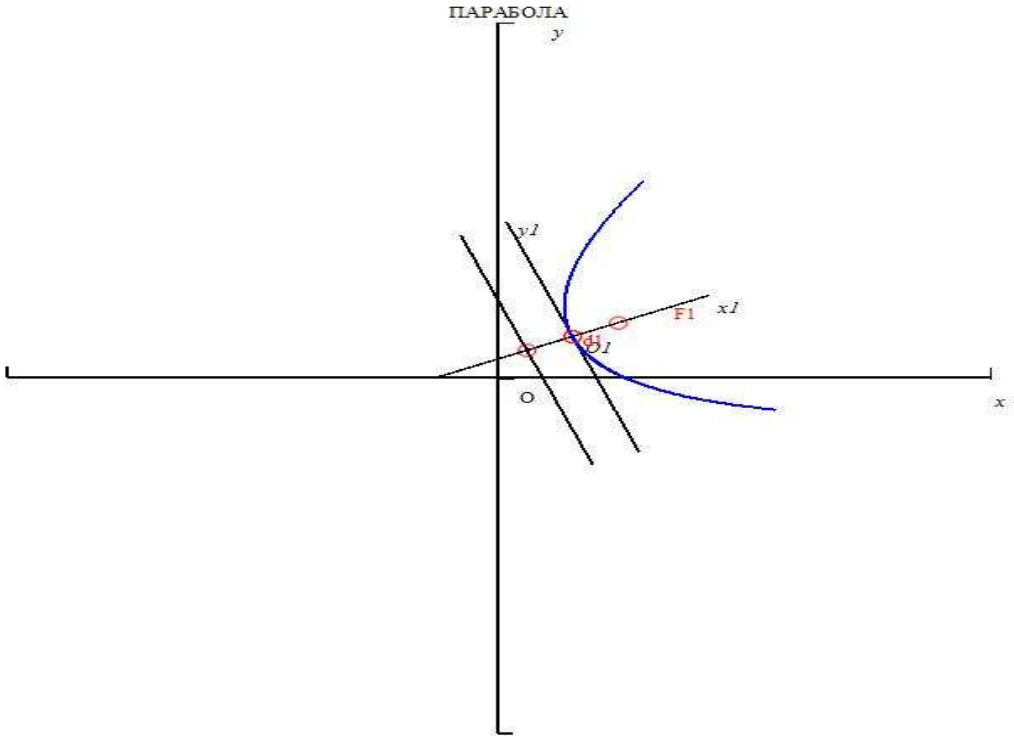

```
> QQQ233:=25*x^2-120*x*y+144*y^2-78*x+10*y-8=0;
```

$$QQQ233 := 25\,x^2 - 120\,xy + 144\,y^2 - 78\,x + 10\,y - 8 = 0$$

```
> AnalGeo[CanonF](QQQ233,[x,y],[xi,eta],alpha);
```

$$OX1''$$

$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [0, 169] \\ \texttt{Уравнение} & : & \eta^2 = \frac{886}{2197}\,\xi \\ x' & = & \frac{12}{13}\,\xi - \frac{5}{13}\,\eta + \frac{293513}{1946542} \\ y' & = & \frac{5}{13}\,\xi + \frac{12}{13}\,\eta + \frac{255}{2197} \\ [c, \epsilon, d] & = & [1, -\frac{443}{4394}] \\ [a, b, p] & = & [\frac{443}{2197}] \\ [\cos(\alpha), \sin(\alpha)] & = & [\frac{12}{13}, \frac{5}{13}] \\ [x_0, y_0] & = & [\frac{293513}{1946542}, \frac{255}{2197}] \end{array}\right]$$



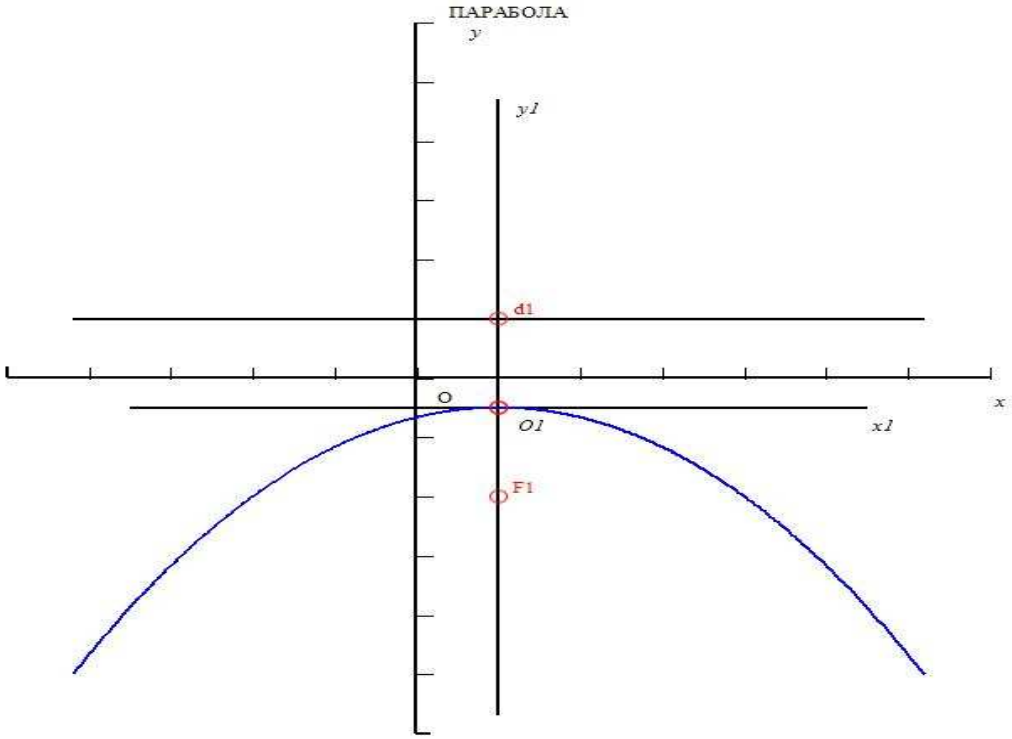

```
> Q233:=x^2+2*x+6*y-2=0;
```

$$Q233 := x^2 + 2\,x + 6\,y - 2 = 0$$

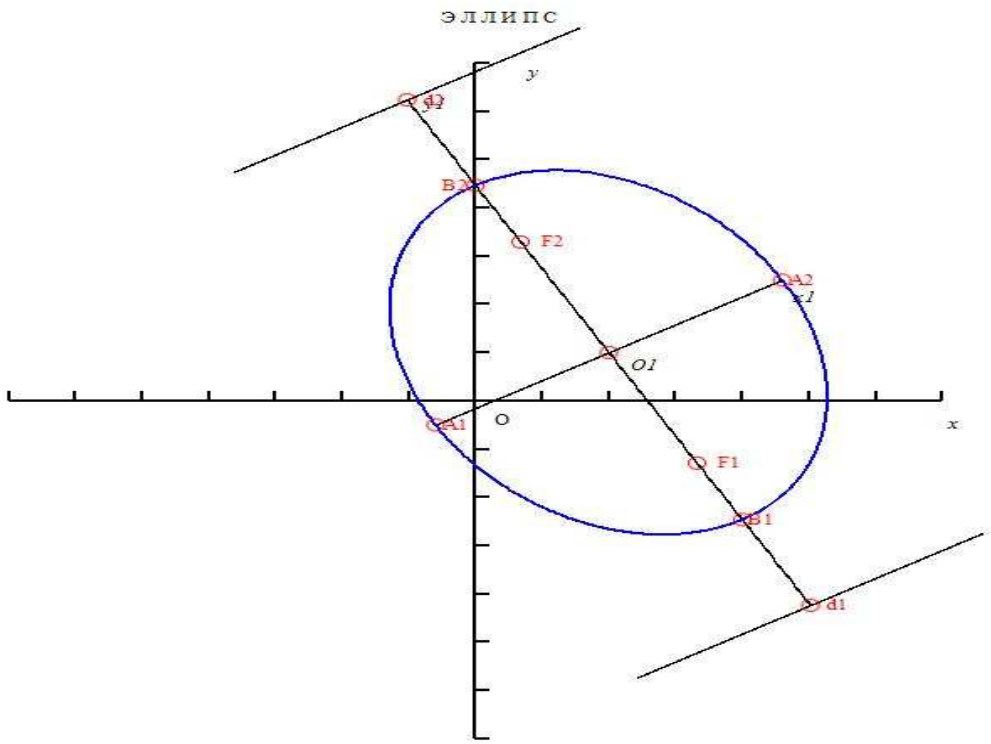
```
> AnalGeo[CanonF](Q233,[x,y],[xi,eta],alpha);
```

$$OY1''$$

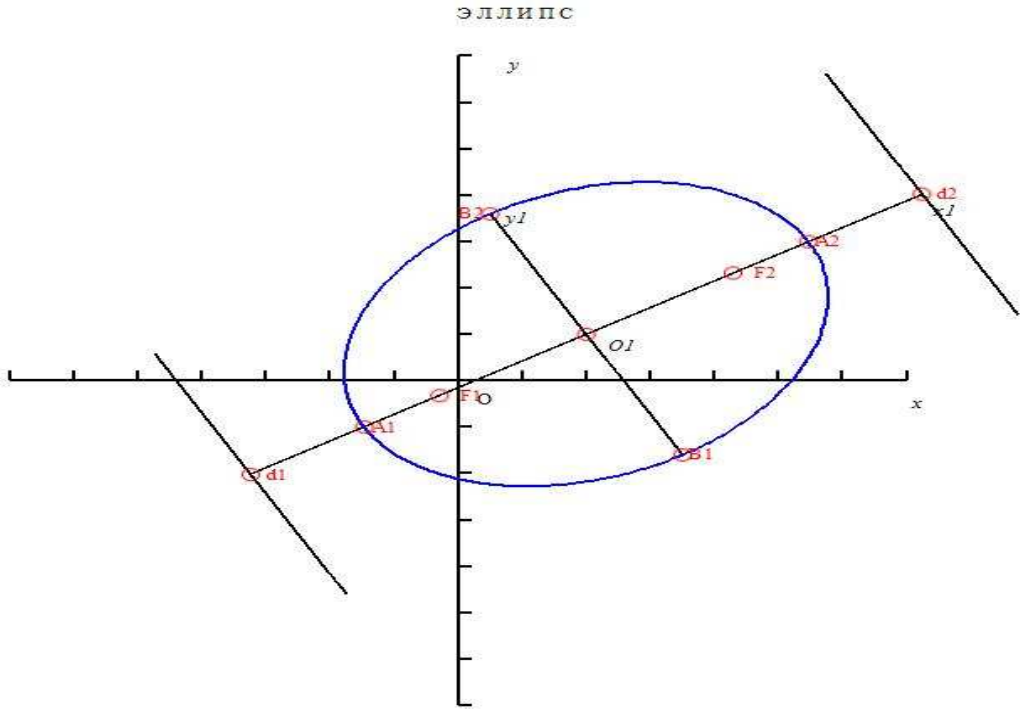
$$\left[\begin{array}{ccc} [\lambda_1, \lambda_2] & = & [1, 0] \\ \text{Уравнение} & : & \xi^2 = -6\,\eta \\ x' & = & \xi + 1 \\ y' & = & -1/2 + \eta \\ [c, \epsilon, d] & = & [1, 3/2] \\ [a, b, p] & = & [-3] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [1, -1/2] \end{array}\right]$$




```
> AnalGeo[Ell](3,4,Pi/6,[2,1],blue,red);
```




```
> AnalGeo[Ell](4,3,Pi/6,[2,1],blue,red);
```

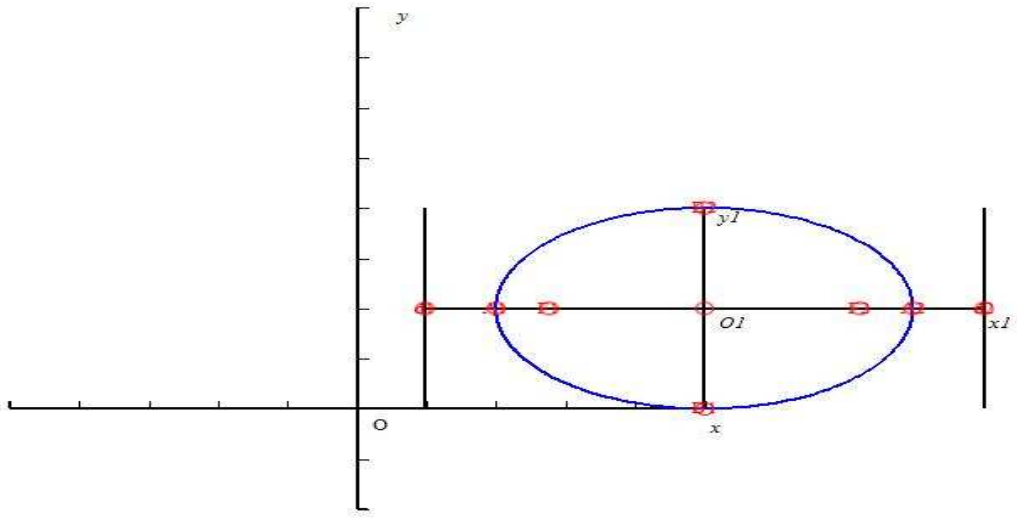



```
> AnalGeo[CanonF](4*x^2+\
> 9*y^2-40*x+36*y+100=0,[x,y],[xi,eta],alpha);
```

$$OX''$$

$$\begin{bmatrix} [\lambda_1, \lambda_2] & = & [4, 9] \\ \text{Уравнение} & : & 1/9\,\xi^2 + 1/4\,\eta^2 = 1 \\ x' & = & \xi + 5 \\ y' & = & -2 + \eta \\ [c, \epsilon, d] & = & [\sqrt{5}, 1/3\,\sqrt{5}, 9/5\,\sqrt{5}] \\ [a, b, p] & = & [3, 2] \\ [\cos(\alpha), \sin(\alpha)] & = & [1, 0] \\ [x_0, y_0] & = & [5, 2] \end{bmatrix}$$

эллипс




```
> save(AnalGeo,"Geom.m"):
```

II.5 Создание процедуры интегрирования с графическим сопровождением

В книге В.П. Дьяконова [80] приведен пример анимации процедуры приближенного вычисления определенного интеграла частного вида. Мы рассмотрим ниже многопараметрическую процедуру, позволяющую демонстрировать вычисление определенного интеграла от произвольной функции на

произвольном отрезке с произвольным числом разбиения N.

II.5.1 Вычисление интегральной суммы Дарбу

Подключим библиотеку Maple `student`:

```
> restart:
> with(student):
```

Определим интегральную сумму Дарбу для произвольной пока функции с помощью команды **middlesum** этой библиотеки:

```
> Sigma:=(n,f,x,a,b)->middlesum(f,x=a..b,n);
```

$$\Sigma := (n, f, x, a, b) \mapsto \frac{(b-a)\sum_{i=0}^{n-1} f}{n}$$

```
> Sigma(n,x^3,x,0,1);
```

$$\sum_{i=0}^{n-1} \frac{(i+1/2)^3}{n^3} n^{-1}$$

Вычислим, например, предел суммы Дарбу пр $n \to \infty$:

```
>Limit(Sigma(n,sin(x)/x,x,0,2*Pi),
n=infinity)=limit(Sigma(n,sin(x)/x,x,0,2*Pi),
n=infinity);
```

$$\lim_{n\to\infty} \left(2\,\pi \sum_{i=0}^{n-1} 1/2\, \sin\left(2\,\frac{(i+1/2)\,\pi}{n} \right) n\,(i+1/2)^{-1}\,\pi^{-1} n^{-1} \right) = Si\,(2\,\pi)$$

Вычислим теперь точное значение соответствующего определенного интеграла:

$$\int_0^{2\pi} \frac{\sin x}{x} dx :$$

```
> int(sin(x)/x,x=0..2*Pi);
```

$$Si\,(2\,\pi)$$

Таким образом, точное значение интеграла равно $\mathrm{Si}(2\pi)$, где $\mathrm{Si}(x)$ – интегральный синус. С помощью команды `middlebox` библиотеки `student` изобразим теперь процесс итерации соответствующего интеграла методом прямоугольников при разбиении отрезка на 20 равных частей:

```
> middlebox(sin(x)/x,x=0..2*Pi,20);
```

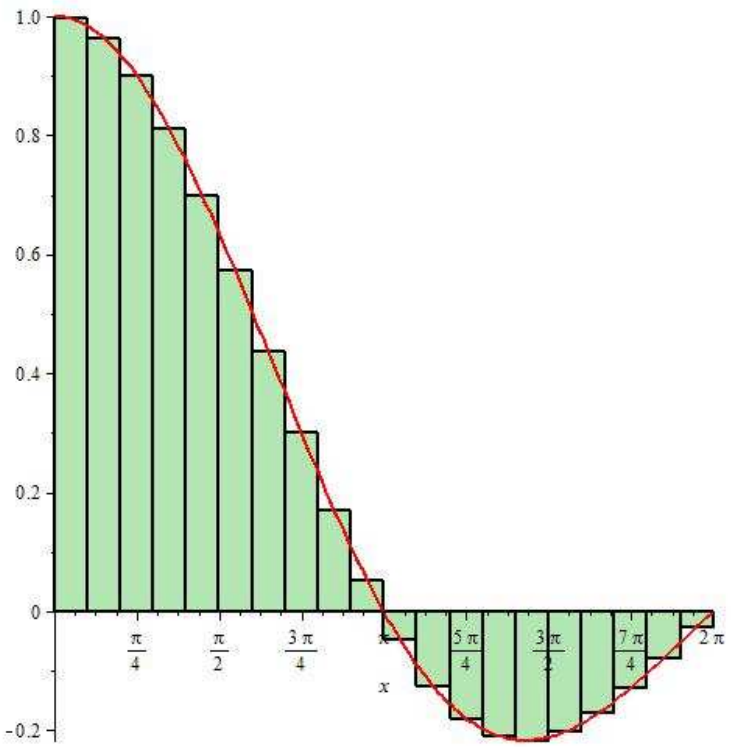


Рис.II.18 *Визуализация интегральной суммы Дарбу для функции* $\sin(x)/x$ *на отрезке* $[0, 2\pi]$ *при* $N = 20$.

II.5.2 Анимация процедуры вычисления определенного интеграла

Создадим процедуру итераций функции $f(x)$ на отрезке $[a, b]$, разбивая его на n частей методом прямоугольников, причем высоту прямоугольников определяет левое значение функции на каждом из интервалов:

```
>AA:=(f,x,a,b,n,d)->
middlebox(f,x=a..b,n,title=d,titlefont=[TIMES,BOLD,14]):
```

Приведем пример:

```
> AA(sin(x)/x,x,0,2*Pi,10,‘Рисунок‘);
```

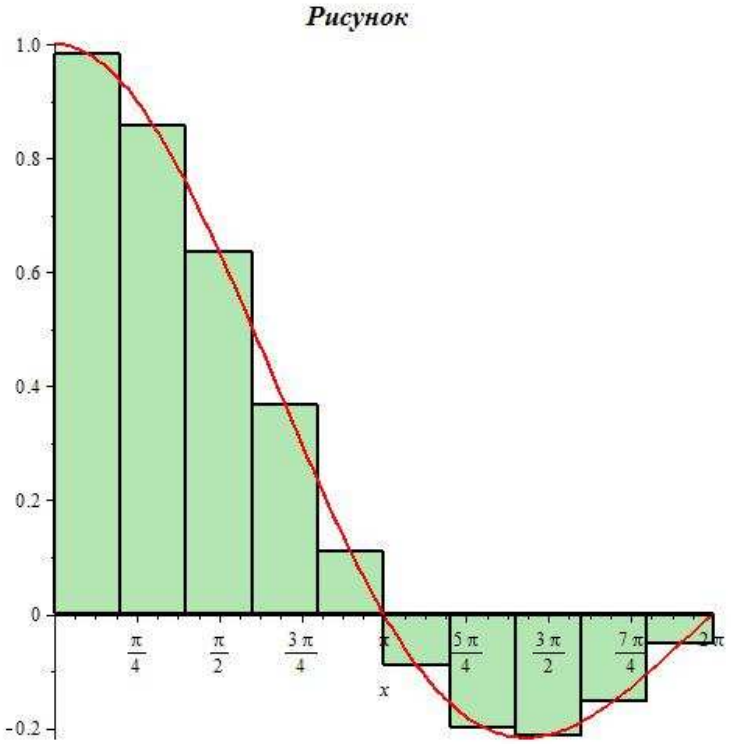


Рис.II.19 *Визуализация интегральной суммы Дарбу для функции* $\sin(x)/x$ *на отрезке* $[0, 2\pi]$ *при* $N = 10$ *с выводом названия рисунка «Рисунок».*

Команда `middlesum` вычисляет среднюю сумму для заданного разбиения, которая аппроксимирует определенный интеграл от заданной функции на заданном промежутке:

```
> LS:=(f,x,a,b,n)->middlesum(f,x=a..b,n);
> LS(f,x,a,b,n);
```

$$LS := (f, x, a, b, n) \mapsto \frac{(b-a)\sum_{i=0}^{n-1} f}{n}$$

$$\frac{(b-a)\sum_{i=0}^{n-1} f}{n}$$

```
> evalf(LS(sin(x)/x,x,0,2*Pi,5));
> evalf(LS(sin(x)/x,x,0,2*Pi,10));
> evalf(LS(sin(x)/x,x,0,2*Pi,100));
```

1.407258804
1.415507839
1.418125394

- таким образом, мы все точнее и точнее будем вычислять указанный интеграл, увеличивая число n. Конвертируем теперь полученную величину интегральной суммы в строковую переменную для ее динамического отображения в названии рисунка и создадим команду кадра анимации, `Cadr`:

```
> LSN:=(f,x,a,b,n)->convert(evalf(LS(f,x,a,b,n)),string);
```

$$LSN := (f, x, a, b, n) \mapsto \text{“}(b - 1. * a)/n * Sum(f, i = 0..n - 1)\text{”}$$

```
> LSN(sin(x)/x,x,0,2*Pi,10);
```

“1.415507839”

```
> Cadr:=(f,x,a,b,n)->AA(f,x,a,b,n,LSN(f,x,a,b,n));
> Cadr(sin(x)/x,x,0,2*Pi,12);
```

Покажем пример применения этой процедуры для указанной суммы Дарбу при разбиении отрезка на 12 интервалов:

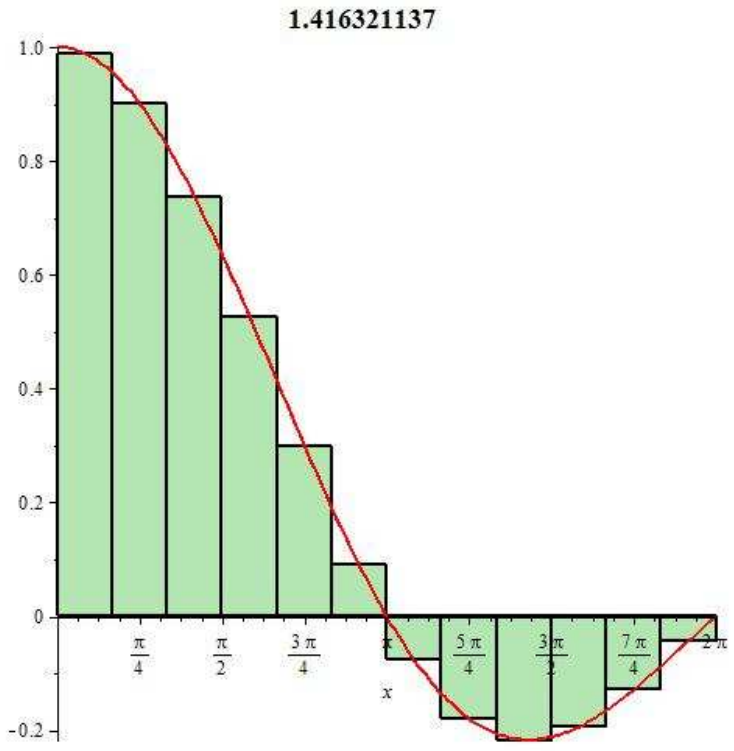


Рис.II.20 *Применение процедуры* `Cadr(sin(x)/x,x,0,2*Pi,12)`.

Создадим последовательность с помощью команды `seq` этих интегральных сумм, задав в качестве заголовка значение этой суммы при заданном числе разбиений:

```
> S:=(f,x,a,b,N)->seq(Cadr(f,x,a,b,n),n=1..N);
```

И создадим окончательно анимацию процесса вычисления определенного интеграла: $J = \int_a^b f dx$ при делении промежутка интегрирования на N частей:

```
> GS:=(f,x,a,b,N)->plots[display](S(f,x,a,b,N),insequence=true):
```

II.5.3 Примеры

1. Вычислим интеграл

$$\int_0^{2\,\pi} \frac{\sin(x)}{x} dx \tag{II.47}$$

делением промежутка интегрирования на 100 частей:

```
> Int(sin(x)/x,x = 0 .. 2*Pi)=int(sin(x)/x,x = 0 .. 2*Pi);
```

$$\int_0^{2\,\pi} \frac{\sin(x)}{x} dx = Si(2\,\pi)$$

Его приближенное значение равно:

```
> Int(sin(x)/x,x = 0 .. 2*Pi)=evalf(int(sin(x)/x,x = 0 .. 2*Pi));
```

$$\int_0^{2\,\pi} \frac{\sin(x)}{x} dx = 1.418151576$$

```
> GS(sin(x)/x,x,0,2*Pi,100);
```

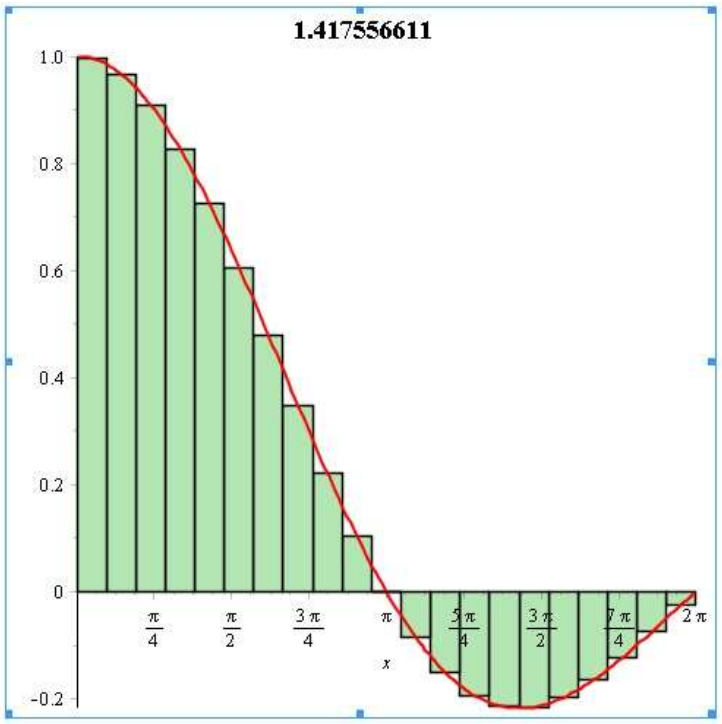


Рис.II.21 *21-й кадр анимации процесса вычисления интеграла (II.47).*

2. Вычислим интеграл

$$\int_{-1}^{3} x^3 - 4\,x^2 + x - 1 dx \tag{II.48}$$

разбивая промежуток интегрирования на 100 частей: Его точное значение равно:

```
> Int(x^3-4*x^2+x-1,x = -1 .. 3)=int(x^3-4*x^2+x-1,x = -1 .. 3);
```

$$\int_{-1}^{3} x^3 - 4\,x^2 + x - 1 dx = -\frac{52}{3}$$

и приближенное -

```
> Int(x^3-4*x^2+x-1,x = -1 .. 3)=evalf(int(x^3-4*x^2+x-1,x = -1
.. 3));
```

$$\int_{-1}^{3} x^3 - 4\,x^2 + x - 1 dx = -17.33333333$$

```
> GS(x^3-4*x^2+x-1,x,-1,3,100);
```

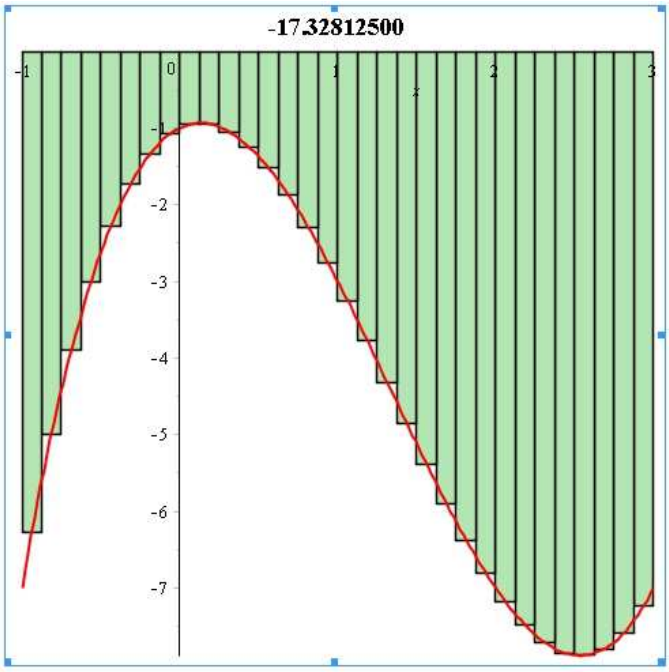


Рис.II.22 *32-й кадр анимации процесса вычисления интеграла (II.48).*

II.6 Создание процедуры анимации вычисления предела суммы

Создадим процедуру вычисления **частичной суммы ряда** д ля произвольной пока последовательности

f(n)с помощью процедуры PartSum:

```
> AnimLimit[PartSum]:=proc(f,i,n) local F:
> F:=(k)->subs(i=k,f): sum(F(k),k=1..n):end proc:
```

Проверим действие процедуры:

```
> AnimLimit[PartSum]((-1)^i,i,101);
```

$$-1$$

Вычислим теперь с помощью полученной процедуры предел последовательности:

```
> Limit(AnimLimit[PartSum](1/i^2,i,n),
> n=infinity)=limit(AnimLimit[PartSum](1/i^2,i,n),n=infinity);
```

$$\lim_{n\to\infty} -\Psi(1,\, n+1) + \frac{\pi^2}{6} = \frac{\pi^2}{6}$$

Договоримся далее графически отображать частичную сумму n первых членов ряда с помощью прямоугольника единичного основания, высотой, равной, этой частичной сумме, и позиционированным координатой i на оси абсцисс. Создадим диаграмму частичных сумм ряда от n=1 до n=N:

```
>AnimLimit[GraphSum]:=proc(f,i,n0,N,c) local
F,k,n,Sn,Sgn,j,S,S0,SG,Sj:
F:=(k)->subs(i=k,f):Sn:=(n0,n)->evalf(sum(F(k),k=n0..n)):
S:=(n0)->evalf(limit(Sn(n0,n),n=infinity)):
S0:=(n0)->limit(sum(F(k),k=n0..n),n=infinity):
SG:=plot([t,S(n0),t=n0..N],color=red,thickness=2):
#Sj:=evalf(subs(n=N,Sn)):
Sgn:=seq(plots[polygonplot]([[j,0],[j+1,0],
#[j+1,evalf(subs(n=j,Sn))],
#[j,subs(n=j,Sn)]
[j+1,Sn(n0,j)],[j,Sn(n0,j)]],color=c,
titlefont=[TIMES,ROMAN,14],title=
convert([n=N,Sn(n0,N),Lim=S0(n0)],string)
),j=n0..N):
plots[display](Sgn,SG):
end proc:
```

Создадим анимационную процедуру, предварительно выведя информацию о представленном процессе, а затем создавая последовательность всех кадров анимации и объединяя их в процедуре display с опцией insequence=true:

```
> AnimLimit[AnimSum]:=proc(f,i,n0,N1,c) local N,F,k,prtext:
> F:=(k)->subs(i=k,f):
> prtext:=print(‘Процесс вычисления предела суммы‘,
> Sum(F(k),k=n0..infinity)=sum(F(k),k=n0..infinity)):
> prtext,
> plots[display](seq(AnimLimit[GraphSum](f,i,n0,N,c),
> N=n0+1..N1),insequence=true):
> end proc:
```

```
> save(AnimLimit,‘AnimLimit.m‘);
```

II.7 Создание процедуры анимации вычисления предела последовательности

Пусть f(n) - числовая последовательность, тогда ее предел, если он существует, равен:

```
> AnimLimit[Lim]:=proc(f,n) local F,k:
> F:=subs(n=k,f): limit(F,k=infinity):end proc:
```

Например, рассмотрим первый замечательный предел:

```
> AnimLimit[Lim](n*sin(1/n),n);
```

$$1$$

Создадим теперь n-й кадр анимационного фильма, состоящий из двух прямо угольников высотой h1=f(n) и высотой h2=limit(f(n),n=infinity), причем нижнему придадим цвет c, верхнему - (RGB,0.5,0.8,0.8).

Далее поместим информацию о номере кадра n, приближенном значении последовательности для этого значения n и точное значение предела последовательности.

```
> AnimLimit[Cadr]:=proc(f,n,i,c)
> local F,k,Fi,Fn,lim,limf,pr,p0,ttx:
> F:=(k)->eval(subs(n=k,f)):
> Fn:=evalf(F(i)):
> lim:=limit(F(k),k=infinity):
> limf:=evalf(limit(F(k),k=infinity)):
> pr:=plots[polygonplot]([[0,0],[1,0],[1,Fn],[0,Fn]],color=c):
> p0:=plots[polygonplot]([[0,Fn],[1,Fn],[1,limf],[0,limf]],
> color=COLOR(RGB,0.5,0.8,0.8)):
> ttx:=plots[textplot]([1.5,Fn,convert([n=i,f=evalf(Fn)],
> string)],color=c,font=[TIMES,ROMAN,12],align=ABOVE):
> plots[display](pr,p0,ttx,axes=FRA
> ME,titlefont=[TIMES,ROMAN,12],title=convert([n=i,f=Fn,
> Limit(f)=lim],string)):end proc:
```

Создадим анимационную процедуру, предварительно выведя информацию о представленном процессе, а затем создавая последовательность всех кадров анимации и объединяя их в процедуре display с опцией `insequence = true`:

```
>AnimLimit[Anim_Seq]:=proc(f,n,N,c)
local prtext:
prtext:=print(‘Процесс вычисления предела последовательности‘,
Limit(f,n=infinity)=limit(f,n=infinity)):
plots[display](seq(AnimLimit[Cadr](f,n,i,c),i=1..N),
insequence=true):
end proc:
```

II.8 Создание процедуры анимации вычисления предела суммы

Создадим процедуру вычисления *частичной суммы ряда* для произвольной пока последовательности $f(n)$ с помощью процедуры `PartSum`:

```
> AnimLimit[PartSum]:=
> proc(f,i,n) local F:
> F:=(k)->subs(i=k,f): sum(F(k),k=1..n):end proc:
```

Проверим действие процедуры:

```
> AnimLimit[PartSum]((-1)^i,i,101);
```

$$-1$$

Вычислим теперь с помощью полученной процедуры предел последовательности:

```
> Limit(AnimLimit[PartSum](1/i^2,i,n),
> n=infinity)=limit(AnimLimit[PartS
> um](1/i^2,i,n),n=infinity);
```

$$\lim_{n\to\infty} -\Psi(1,\, n+1)+\frac{\pi^2}{6}=\frac{\pi^2}{6}$$

Договоримся далее графически отображать частичную сумму n первых членов ряда с помощью прямоугольника единичного основания, высотой, равной, этой частичной сумме, и позиционированной координатой `i` на оси абсцисс. Создадим диаграмму частичных сумм ряда от n=1 до n=N:

```
>AnimLimit[GraphSum]:=proc(f,i,n0,N,c) local
F,k,n,Sn,Sgn,j,S,S0,SG,Sj:
F:=(k)->subs(i=k,f):Sn:=(n0,n)->evalf(sum(F(k),k=n0..n)):
S:=(n0)->evalf(limit(Sn(n0,n),n=infinity)):
S0:=(n0)->limit(sum(F(k),k=n0..n),n=infinity):
SG:=plot([t,S(n0),t=n0..N],color=red,thickness=2):
Sgn:=seq(plots[polygonplot]([[j,0],[j+1,0],
```

```
[j+1,Sn(n0,j)],[j,Sn(n0,j)]],color=c,
titlefont=[TIMES,ROMAN,14],title=
convert([n=N,Sn(n0,N),Lim=S0(n0)],
string)),j=n0..N):
plots[display](Sgn,SG):
end proc:
```

Создадим анимационную процедуру, предварительно выведя информацию о представленном процессе, а затем создавая последовательность всех кадров анимации и объединяя их в процедуре `display` с опцией `insequence = true`:

```
> AnimLimit[AnimSum]:=
> proc(f,i,n0,N1,c) local N,F,k,prtext:
> F:=(k)->subs(i=k,f):
> prtext:=print(‘Процесс вычисления предела суммы‘,
> Sum(F(k),k=n0..infinity)=sum(F(k),k=n0..infinity)):
> prtext,
> plots[display](seq(AnimLimit[GraphSum](f,i,n0,N,c),N=n0+1..N1),
> insequence=true):
> end proc:
```

Сохраним теперь созданные процедуры в библиотеке «AnimLimit» с одноименным названием файла «AnimLimit.m»

```
> save(AnimLimit,‘AnimLimit.m‘);
```

II.9 Примеры анимации процесса вычисления пределов сумм и последовательностей

Загрузим созданную библиотеку:

```
>restart:
read "AnimLimit.m";
with(AnimLimit);
       [AnimSum, Anim_Seq, Cadr, GraphSum, Lim, PartSum]
```

Приведем пример процесса вычисления замечательного предела:

II.9.1 Вычисление первого замечательного предела

Приведем пример исполнения процедуры `Anim_Seq` вычисления первого замечательного предела для последовательности

$$f(n) = n \sin\left(\frac{1}{n}\right) = 1,$$

полагая число кадров равным 100, а цвет вычисляемого элемента - красным:

```
> AnimLimit[Anim_Seq](n*sin(1/n),n,100,red);
```

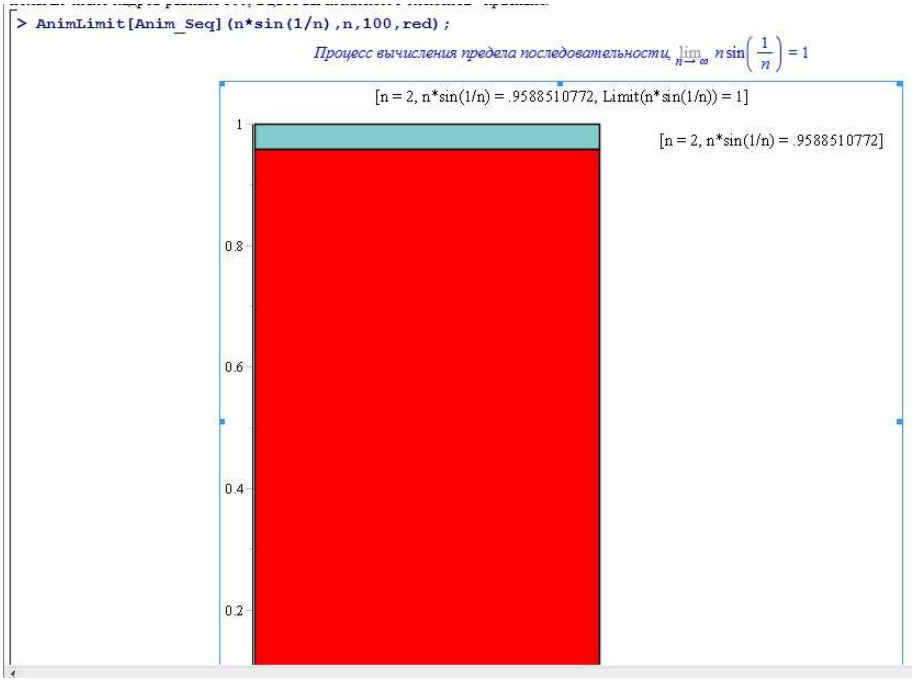


Рис.II.23 *2-й кадр анимации процесса вычисления предела* $\lim\limits_{n\to\infty} n \sin 1/n$.

II.9.2 Пример вычисления предела суммы знакопостоянного ряда

Вычислим сумму знакопостоянного ряда:

$$\lim_{n=\infty} \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 2.$$

```
> AnimLimit[AnimSum](1/2^i,i,0,100,green);
```

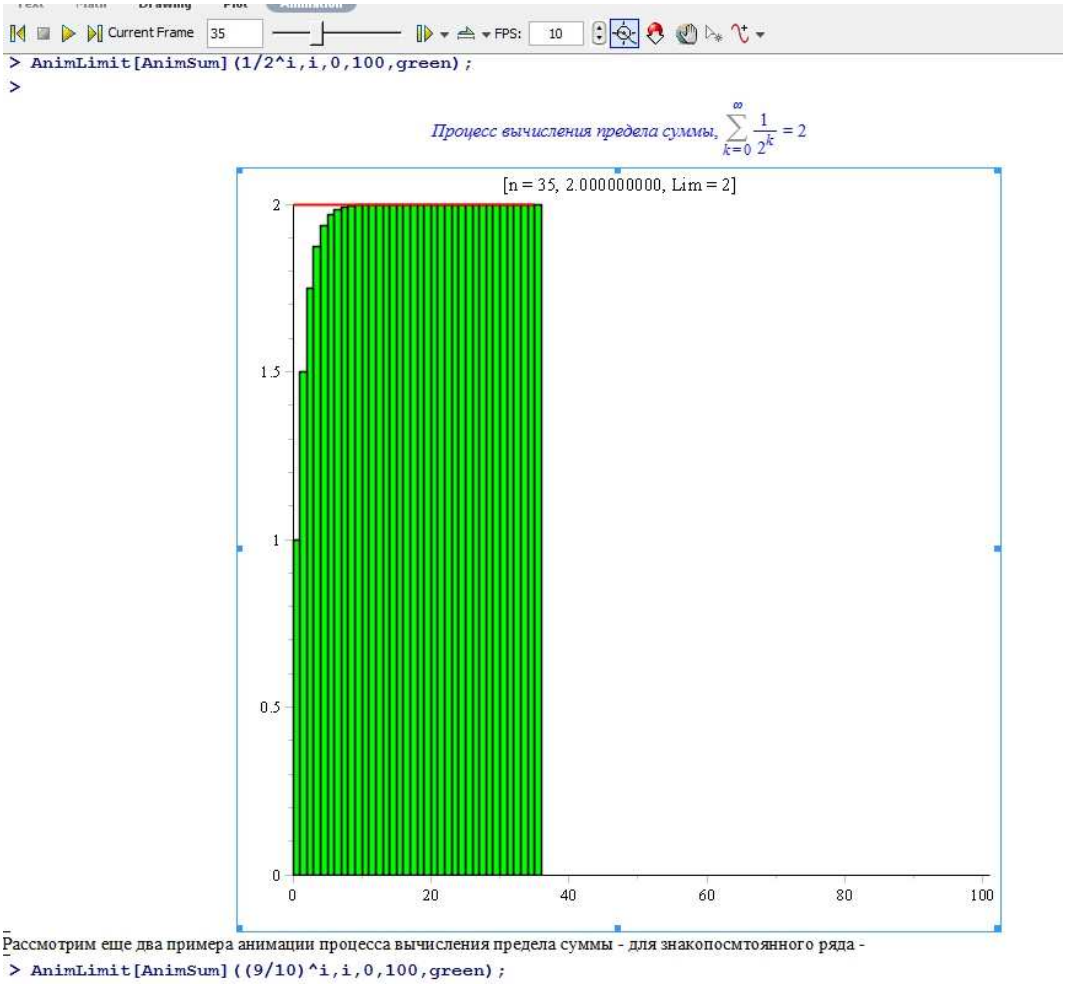


Рис.II.24 *35-й кадр анимации процесса вычисления суммы* $\sum_{i=0}^{100} \frac{1}{2^i}$.

Заметим, что созданные программные процедуры прежде вывода объекта динамической графики выводят исследуемую формулу предела и его точное значение.

II.9.3 Еще один пример вычисления предела суммы знакопостоянного ряда

Вычислим сумму знакопостоянного ряда:

$$\lim_{n=\infty} \sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i = 10.$$

```
>AnimLimit[AnimSum]((9/10)^i,i,0,100,green);
```

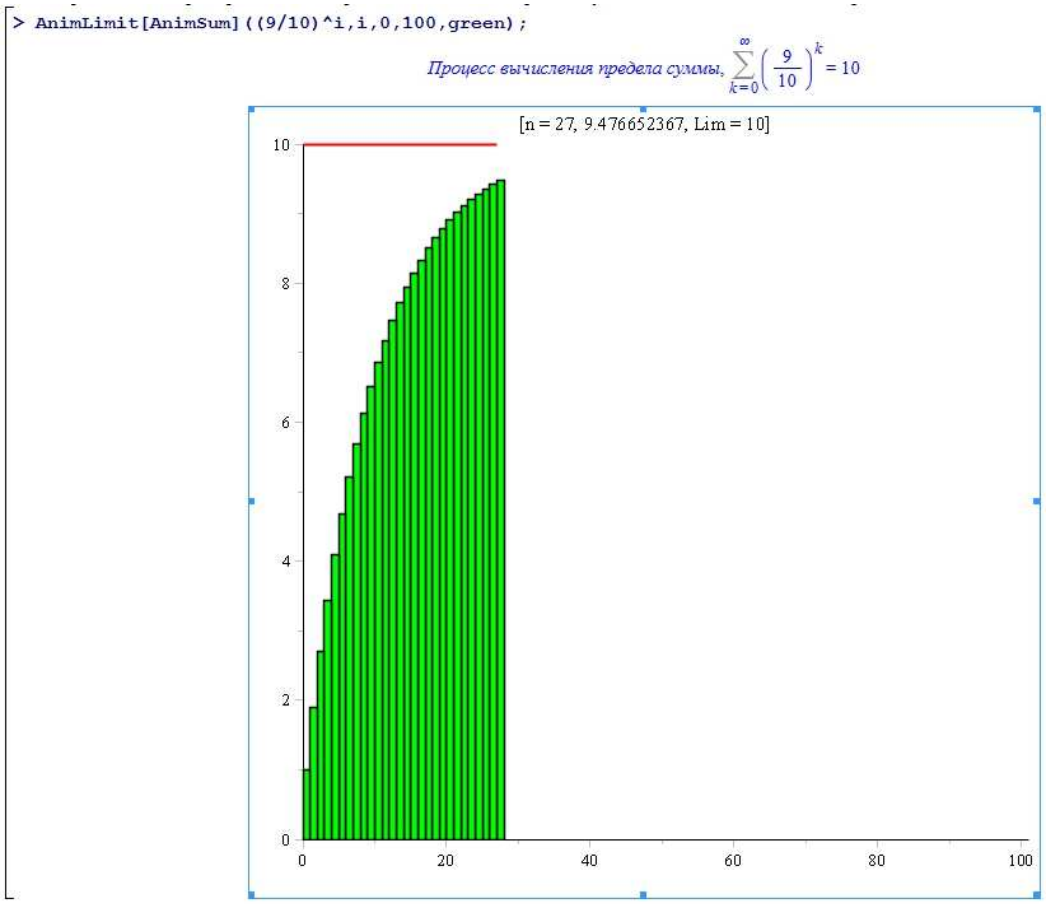


Рис.II.25 *27-й кадр анимации процесса вычисления суммы* $\sum_{i=0}^{100}\left(\frac{9}{10}\right)^i$.

II.9.4 Пример вычисления предела суммы знакопеременного ряда

Вычислим сумму знакопеременного ряда:

$$\lim_{n=\infty}\sum_{i=0}^{\infty}\left(-\frac{9}{10}\right)^i = \frac{10}{19}.$$

```
>AnimLimit[AnimSum]((-9/10)^i,i,0,100,COLOR(RGB,0.4,1,1));
```

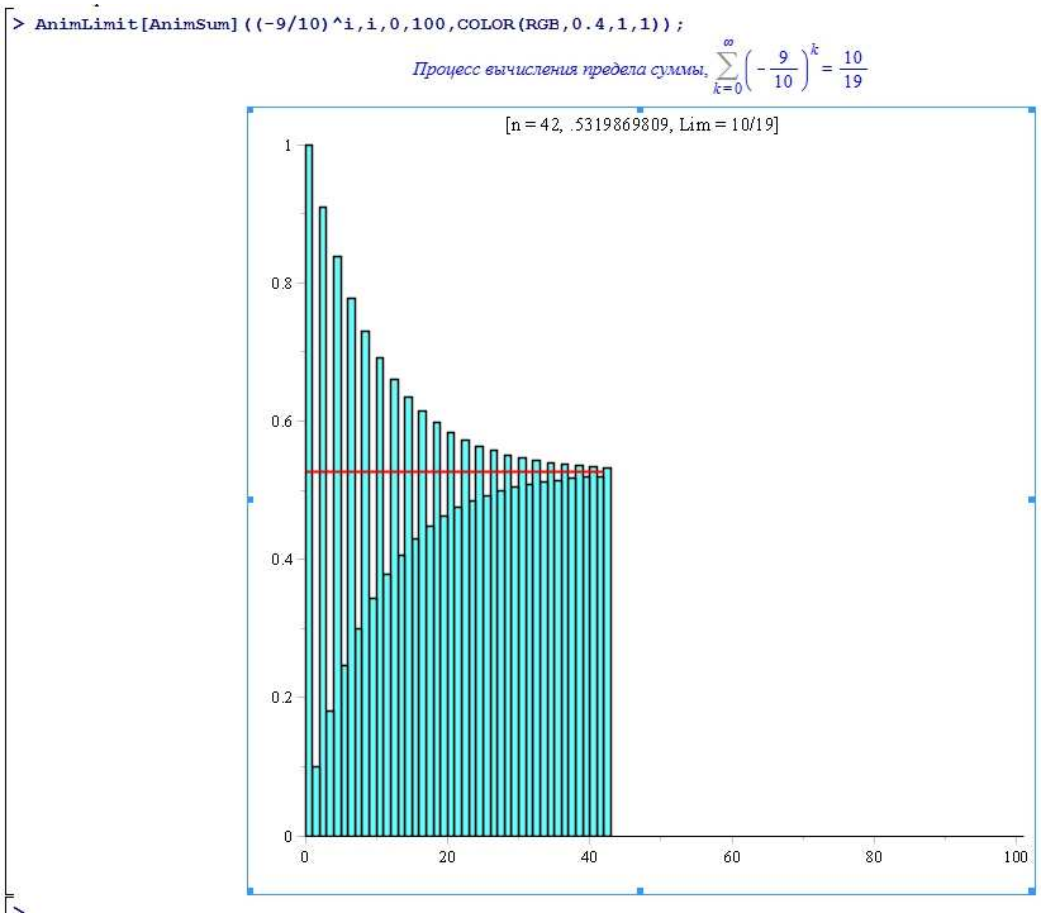


Рис.II.26 *42-й кадр анимации процесса вычисления суммы* $\sum_{i=0}^{100}\left(-\frac{9}{10}\right)^i$.

II.10 Построение анимационной процедуры нахождения производной функции

Сначала определим процедуру вычисления дифференциала функции f(x) в точке x1 по формуле:

$$dx = x2 - x1; dy = y(x2) - y(x1).$$

```
> dif:=proc(x,f,x1,x2) local
> dX,y1,y2,dY:dX:=x2-x1:
> y1:=subs(x=x1,f):y2:=subs(x=x2,f):
> dY:=y2-y1:[dX,dY]:end proc:
```

Вычислим с помощью этой процедуры дифференциал функции x^3 в точке x1=1, если x2=3:

```
> dif(x,x^3,1,3);
```

$$[2,\ 26]$$

Создадим процедуру построения явного уравнения секущей, проходящей через точки графика с абсциссами x1 и x2:

```
> eqline:=proc(x,f,x1,x2)
> local dX,y1,y2,dY,x0:
> dX:=x2-x1: y1:=subs(x=x1,f):
> y2:=subs(x=x2,f):
> dY:=y2-y1: x0:=x1-dX/dY*y1:
> [x0,[dX,dY],y1+dY/dX*(x-x1)]:
> end proc:
```

Построим с помощью этой процедуры явное уравнение секущей к кривой $y = x^3$ при $x1 = 1, x2 = 3$:

```
> QQ:=eqline(x,x^3,1,3);
```

$$QQ := [\frac{12}{13},\ [2,\ 26],\ -12 + 13\,x]$$

Покажем, как с помощью этой процедуры найти значения: приближенного значения производной, дифференциалов координат, левой части уравнения секущей:

```
> QQ[3];
```

$$-12 + 13\,x$$

```
> QQ[2];
```

$$[2,\ 26]$$

```
> QQ[2,1];
```

$$2$$

```
> QQ[1];
```

$$\frac{12}{13}$$

Построим, наконец, анимационную процедуру вычисления предела для произвольной функции $f(x)$ в точке x_1. Точка x_2 выбирается произвольным образом, как некоторая затравочная. Затем происходит деление промежутка $x_2 - x_1$ на 2, 3,.., n частей, в результате этого процесса точка x_k сближается с началом интервала: xk=x1*(x2-x1)/n*k.

```
> graph_n:=proc(x,f,x1,x2,n,c1,c2)
> local DF,x0,d,dxn,
> xn,y1,y2,yn,Q,dyn,dify,difyn,fdifyn,
> x_min,x_max,i,yy_min,y_min,yy_max,y_max,xd,
> X_min,X_max,yd,Y_min,Y_max,EQ,t:
> y1:=subs(x=x1,f):y2:=subs(x=x2,f):
> DF:=subs(x=x1,diff(f,x)):
> if DF=0 then x0:=x1:
> else x0:=x1-y1/DF:
> end if:
> x_min:=min(x0,x1,x2):
> x_max:=max(x0,x1,x2):
> xd:=(x_max-x_min):
> d:=(x_max-x_min)/n:
> Q:=seq(evalf(subs(x=x_min+i*d,f)),i=0..n):
> yy_min:=min(0,Q):
> yy_max:=max(0,Q):
> y_min:=min(y1,y2,yy_min):
> y_max:=max(y1,y2,yy_max):
> yd:=(y_max-y_min):
> X_min:=evalf(x_min-0.1*xd):
> X_max:=evalf(x_max+0.1*xd):
> Y_min:=evalf(y_min-0.1*yd):
> Y_max:=evalf(y_max+0.1*yd):
> dxn:=evalf((x2-x1)/n,4):
> #dxn:=evalf((x2-x1)/(1.2^n-0.2),4):
> xn:=x1+dxn: yn:=subs(x=xn,f):
> dyn:=yn-y1:difyn:=dyn/dxn:
> fdifyn:=evalf(difyn,4):
> EQ:=y1+difyn*(x-x1):
> plot([[x1,t,t=0..y1],[xn,t,t=0..yn],f,EQ,0],
> x=X_min..X_max,y=Y_min..Y_max,
> color=[blue,blue,c1,c2,black],axes=FRAME,
> title=convert([N=n,dx=dxn,dy/dx=fdifyn,
> Dy/dx=evalf(DF,4)],string),numpoints=200
> ):
> end proc:
```

Таким образом, получим и опробуем процедуру построения n-го графика:

```
> graph_n(x,x^3,1,3,1,red,blue);
```

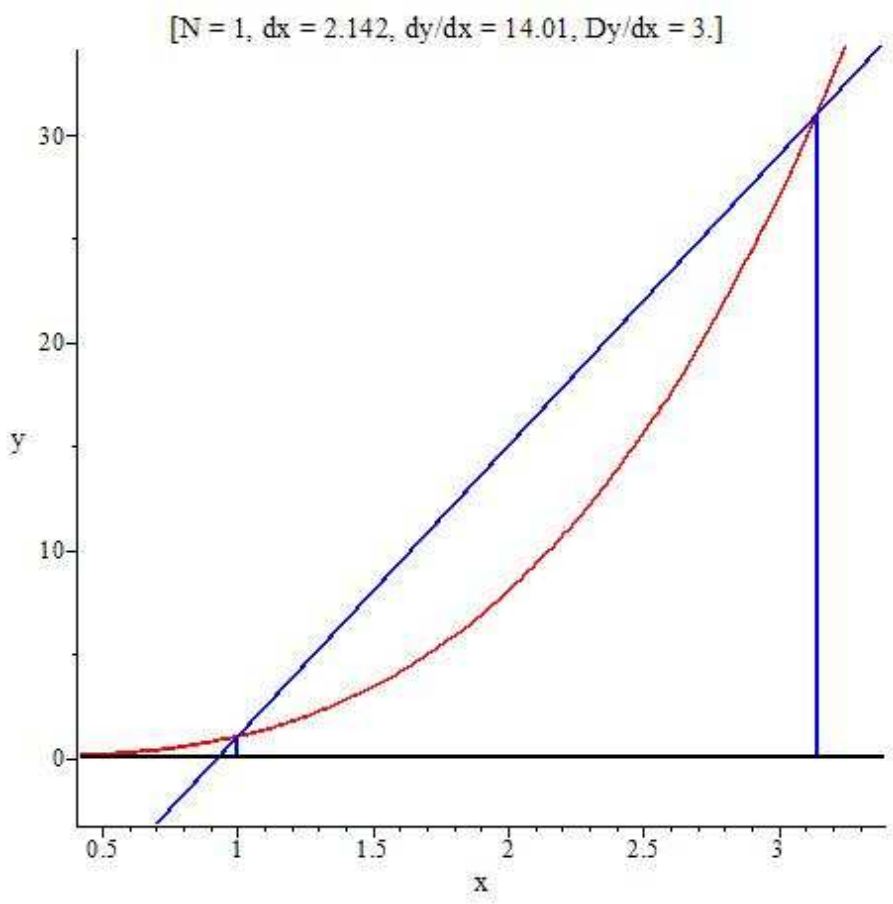


Рис.II.27 *Первый кадр анимации процесса построения касательной к графику функции* $y = x^3$.

На рисунке показан первый шаг вычисления производной. График функции изображен красным цветом, секущей - синим. Собирая последовательность таких графиков в процедуре display с опцией insequence=true, получаем искомую анимационную процедуру процесса вычисления производной. Тангенс угла наклона секущей на каждом шаге дает приближенное значение производной на этом этом шаге, которое отображается в титрах графика.

```
> Anim:= proc(x,f,x1,x2,N,c1,c2) local SS,i:
> SS:=seq(graph_n(x,f,x1,x2,i,c1,c2),i=1..N):
> plots[display](SS,insequence=true):end proc:

> Anim(x,x^3,1,Pi,200,red,blue);
```

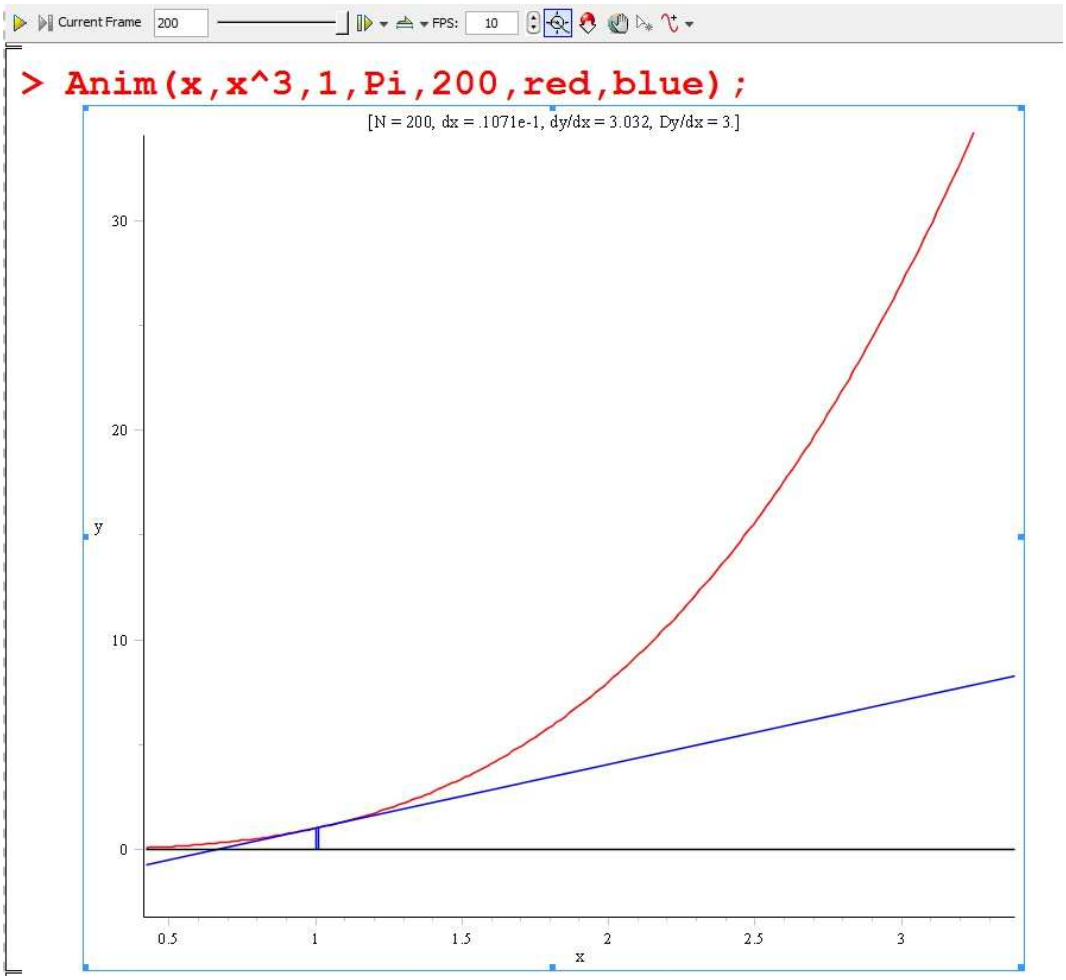


Рис.II.28 *200-й кадр анимации процесса построения касательной к графику функции* $y = x^3$.

Глава III

Моделирование объектов дифференциальной геометрии в Maple

III.1 Адаптированный репер кривой и дифференциальные инварианты кривой в пространстве

III.1.1 Ведение

Дифференциальная геометрия традиционно считается одним из наиболее трудных предметов на математических факультетах вузов. В ней интегрируются знания, полученные студентами на курсах алгебры, аналитической геометрии, математического анализа одной и нескольких переменных, дифференциальных уравнений. Кроме того, геометрические образы, а также соответствующие вычислительные процедуры, сопровождающие этот курс, являются чрезвычайно сложными и громоздкими. Все это делает указанный курс мало наглядным и сложным для усвоения, несмотря на чрезвычайную простоту и прозрачность основных идей дифференциальной геометрии. С другой стороны, методы дифференциальной геометрии кривых и поверхностей находят огромное количество приложений в самых различных областях прикладной математики, теории поля, механики, оптики и т.п. Поэтому, задача повышения наглядности этого курса становится чрезвычайно важной. Эта задача может быть решена графическими средствами компьютерной математики, в частности, пакета Maple.

К стандартным задачам дифференциальной геометрии следует отнести задачи, полностью построенные на аппарате векторной алгебры и дифференциального исчисления функций одной (дифференциальная геометрия кривых) и двух (дифференциальная геометрия поверхностей) переменных. К таковым задачам относятся: построение адаптированного репера кривой, вычисление кривизны и кручения кривой, нахождение первой и второй квад-

ратичных форм поверхностей, ее векторной оснастки, геодезической, нормальной, главных кривизн поверхностей. Именно эти задачи и изучаются в стандартных курсах дифференциальной геометрии. К задачам повышенной сложности, которые за редким исключением практически не изучаются в стандартных курсах дифференциальной геометрии относятся такие, как восстановление кривой по ее натуральным уравнениям и нахождение линий главных направлений кривизн и геодезических на поверхностях. Замечательным является тот факт, что как раз на решения этих задач, как основных, и направлены стандартные курсы дифференциальной геометрии. Не решаются же эти заявленные и очень поучительные задачи в стандартных курсах дифференциальной геометрии именно потому, что задачи эти связаны с решением системы обыкновенных *существенно нелинейных* дифференциальных уравнений, для которых не существует общей теории интегрирования, за исключением некоторых качественных методов исследования. Эти задачи также могут быть решены средствами компьютерной математики. В 90-х годах А. Грэй посвятил специальную монографию изложению дифференциальной геометрии с помощью пакета компьютерной математики «Mathematica 2» [83]. Однако, в этой книге не отражены вопросы, как динамической визуализации объектов дифференциальной геометрии, так и вопросы, связанные с интегральными нелинейными задачами дифференциальной геометрии.

III.1.2 Математическая модель дифференциальной геометрии кривых

Как уже было сказано, дифференциальная геометрия кривых основана на применении методов математического анализа функции одной переменной к структуре аналитической геометрии. Основным объектом дифференциальной геометрии кривых является гладкая параметризованная кривая γ в трехмерном евклидовом пространстве E_3, определяемая относительно декартового репера векторными уравнениями:[1]

$$\mathbf{r} = \mathbf{r}(t), \tag{III.1}$$

где $\mathbf{r} \equiv r$ – радиус-вектор точки кривой, заданный упорядоченным набором непрерывных, дважды дифференцируемых декартовых координат точки $x(t), y(t), z(t)$[2]. С помощью производных по параметру радиуса вектора,

[1] Описание математической модели адаптированного репера можно найти, например, в книге [126].
[2] См., например, [126].

$\dot{\mathbf{r}}(t), \ddot{\mathbf{r}}(t)$ в каждой точке кривой определяются векторы ортонормированного базиса $\{e\}_\gamma = \{\tau, \nu, \beta\}$[3]:

$$\tau = \frac{\dot{r}}{|\dot{r}|}; \quad \beta = \frac{[\dot{r}, \ddot{r}]}{|[\dot{r}, \ddot{r}]|}; \quad \nu = \frac{[\dot{r}, [\dot{r}, \ddot{r}]]}{|[\dot{r}, [\dot{r}, \ddot{r}]]|} \tag{III.2}$$

– единичные векторы касательной, бинормали и нормали, соответственно; [**a**,**b**] – знак векторного произведения векторов **a**, **b**. Присоединение к текущей точки кривой M(t) ортонормированного базиса (V.6) дает нам так называемый адаптированный репер, $R_t(M(t); \tau(t), \nu(t), \beta(t))$, – правый прямоугольный декартов репер, скользящий вдоль параметризованной кривой и являющийся основным инструментом исследования ее геометрии не только в трехмерном евклидовом, но, вообще говоря, и в римановом пространстве произвольной размерности. В связи с этим понятие адаптированного репера является чрезвычайно важным для дифференциальной геометрии и требует наглядного представления в курсе дифференциальной геометрии[4]. Далее, для гладкой кривой γ вводится ее натуральный параметр, s, как длина этой кривой:

$$s(t) = \int_{t_0}^{t} |\dot{r}(t')| dt', \tag{III.3}$$

являющийся монотонно возрастающей функцией параметра t. Разложение производных по натуральному параметру от векторов базиса по векторам же $\{e\}_\gamma$ представляется формулами Френе-Серре:

$$\tau' = k\nu; \quad \nu' = -k\tau + \kappa\beta; \quad \beta' = -\kappa\nu, \tag{III.4}$$

где штрих означает дифференцирование по натуральному параметру а два скаляра:

$$k = k(s); \quad \kappa = \kappa(s), \tag{III.5}$$

– кривизна и кручение кривой, соответственно, определяются формулами:

$$k = \frac{|[\dot{r}, \ddot{r}]|}{|\dot{r}|^3}; \quad \kappa = \frac{(\dot{r}, \ddot{r}, \dddot{r})}{|[\dot{r}, \ddot{r}]|^2}; \tag{III.6}$$

[3] Мы придерживаемся классического обозначения векторов адаптированного репера, см., например, [126].

[4] Здесь следует отметить, что в советской системе высшего образования курс дифференциальной геометрии сопровождался большим количеством наглядных учебных пособий. В частности, понятия соприкосновения кривых и адаптированного (натурального) репера кривых демонстрировались на специальных приборах, снабженными подвижной кареткой, скользящей вдоль макета кривой, выполненного из проволоки.

(**a**,**b**.**c**) – знак смешанного произведения векторов **a**,**b**.**c**. Соотношения (V.8) называются натуральными уравнениями кривой. Теорема о конгруэнтности двух кривых с одинаковыми натуральными уравнениями приводит к тому, что знание в каждой точке кривой ее кривизны и кручения полностью определяет внутреннюю геометрию этой кривой. В дифференциальной геометрии кривых можно различить две задачи, – прямую и обратную. В прямой задаче, которую только обычно и решают в стандартных курсах дифференциальной геометрии, требуется определить натуральные уравнения кривой по заданному ее параметрическому уравнению. В обратной задаче необходимо определить с точностью до движения ее параметрическое уравнение. Программной реализации второй задачи для произвольной кривой мы посвятим следующий раздел. В этом же разделе мы рассмотрим программные процедуры реализации прямой задачи.

III.1.3 Визуализация натуральных уравнений кривой

Задача визуализации натуральных уравнений произвольной кривой решается является простой – согласно (V.8)–(V.9) для этого необходимо лишь вычислить зависимости (V.8) в каждой точке кривой и изобразить их на совместном графике. В созданном нами пакете программных процедур `YuDifGeo` эта задача решается с помощью команд
`GraphicNatural Curvature(Line,param,param0,max_param,cc)`,
`YuDifGeo(Line, param,param0,max_param,cc)` и
`GraphicNaturalEquations (Line,param,param0,max_param,cc1,cc2)`,
где Line – параметрические уравнения кривой, задаваемые в списочном виде `[x(t),y(t),z(t)]`; `param, param0` – имя параметра кривой и начальное значение параметра, max_param – максимальное значение параметра, cc, cc1, cc2 – цвета изображения кривых. При этом первые две команды изображают кривизну и кручения кривой на отдельном графике, третья команда совмещает эти изображения. Приведем фрагмент программы:

```
>YuDifGeo[GraphicNaturalCurvature]:=
proc(Line,param,param0,max\_param,cc) local param1:
plot([YuDifGeo[CurvatureLine](Line,param,param1),
YuDifGeo[LengthLine](Line,param,param0,param1),
param1=param0..max\_param],color=cc,
labelfont=[TIMES,ROMAN,12],
labels=[‘s‘,‘‘]):end:
```

Указанные программные процедуры используют внутренние процедуры пакета – `CurvatureLine, TorsionLine и LengthLine`, ответственные за вычисление длины дуги кривой, ее кривизны и кручения, соответственно. Продемонстрируем выполнение процедуры изображения натуральных уравнений конической винтовой линии $r = (t\cos(t), t\sin(t), t)$ на отрезке $[0,4\pi\ ]$, причем график кривизны изображается синим цветом, кручения – красным:
>YuDifGeo[GraphicNaturalEquations]([t*cos(t),t*sin(t),t],t,0,4*Pi,blue,red);

III.1.4 Оснащенная динамическая визуализация адаптированного репера произвольной параметризованной кривой

Аналогичная задача нахождения натуральных уравнений кривых и их визуализации в системе Mathematica2 приведена и в цитированной монографии Грэя [83]. Перейдем теперь к решению гораздо более сложной задачи – построению оснащенной динамической визуализации адаптированного репера произвольной параметризованной кривой. Отметим, что задача построения изображения адаптированного репера кривой также решается достаточно просто:

```
>YuDifGeo[NaturalReper]:=
proc(Line,param,param0,t1,cc1,cc2,cc3)
local rt1,rt2,rt3,tt,bb,nn,xi,zeta,chi:
tt:=YuDifGeo[NormTangentLine](Line,param,param0):
rt1:=YuDifGeo[ParamLine](subs(param=param0,Line),
tt,xi,t1,cc1,scaling=CONSTRAINED):
bb:=YuDifGeo[BinormLine](Line,param,param0):
rt2:=YuDifGeo[ParamLine](subs(param=param0,Line),
bb,zeta,t1,cc2,scaling=CONSTRAINED):
nn:=YuDifGeo[MainNormLine](Line,param,param0):
rt3:=YuDifGeo[ParamLine](subs(param=param0,Line),
nn,chi,t1,cc3,scaling=CONSTRAINED):
plots[display](rt1,rt2,rt3):end:
```

Как видно, здесь использован ряд внутренних процедур пакета, ответственных за изображение отдельных элементов репера кривой в ее точке, соответствующей значению параметра t1.

Значительно более сложно решается задача построения оснащенной динамической модели адаптированного репера кривой. Эта задача решается в созданном пакете программной процедурой

`YuDifGeo[AnimNatural Reper](Line,param,mini,maxi,c,N,p)`,
где кроме уже введенных параметров mini – минимальное значение параметра t кривой, maxi – максимальное, c – цвет изображения кривой, N – число кадров динамической модели. Параметр p может принимать значение 1 или любое другое, в том числе и буквенное. При p=1 в заголовок рисунка выносятся динамические значения кривизны, кручения и длины дуги. При любом другом значении этого параметра кривизна, кручение и длина дуги кривой отображаются динамическими объемными гистограммами с приложенными к ним динамически обновляемыми значениями указанных скаляров. Для автоматизации процесса построения оптимальной динамической модели необходимо было дать определения ряду графических параметров оснащенной графики, зависящих от конкретного уравнения кривой и заданных границ ее воспроизведения. Для решения этой задачи использовался следующий алгоритм. В каждом кадре определялись минимальные и максимальные значения координат для всех элементов фигуры с помощью встроенной процедуры min и max, затем составлялся список минимальных и максимальных значений координат фигур по всем кадрам и находились максимальные и минимальные координаты по всей динамической модели. Таким образом, можно было определить максимальные размеры динамического объекта по всем измерениям, что позволило оптимизировать размеры его добавочных элементов. На Рис. 3 показан один кадр исполнения этой программной процедуры.

III.2 Компьютерное моделирование адаптированного репера пространственной кривой

Итак, будем задавать параметризованную кривую правой частью ее параметрических уравнений, т.е., радиусом-вектором [f(t),g(t),h(t)], Line - будет ее условное имя, param - ее параметр. Зададим для удобства в дальнейшем команду вычисления длины вектора, ModVector(x) и команду нормирования вектора, NormVector(x):

```
> restart:

> YuDifGeo:=table():
> YuDifGeo[ModVector]:=proc(x):
> simplify(sqrt(linalg[innerprod](x,x))):end:
> YuDifGeo[NormVector]:=proc(x):
> simplify(linalg[scalarmul](x,1/YuDifGeo[ModVector](x))):end:
```

Проверим, как это работает:

```
> YuDifGeo[NormVector]([sin(t),cos(t),1]);
```

$$\left[\begin{array}{ccc} 1/2\,\sqrt{2}\sin\left(t\right) & 1/2\,\sqrt{2}\cos\left(t\right) & 1/2\,\sqrt{2} \end{array}\right]$$

Определим команду вычисления касательного вектора к кривой в точке с параметром param0:

```
> YuDifGeo[TangentLine]:=proc(Line,param,param0):
> eval(subs(param=param0,diff(Line,param))):end:
> YuDifGeo[TangentLine]([a*cos(t),a*sin(t),b*t],t,0);
```

$$[0, a, b]$$

Определим команду вычисления длины дуги кривой:

```
> YuDifGeo[LengthLine]:=proc(Line,param,param0,param1)
> local t:
> simplify(int(YuDifGeo[ModVector](YuDifGeo
> [TangentLine](Line,param,t1)),
> t1=param0..param1)):end:
> YuDifGeo[LengthLine]([a*cos(t),a*sin(t),b*t],t,0,t);
```

$$\sqrt{a^2+b^2}t$$

Определим команду вычисления единичного касательного вектора кривой tau в точке с параметром param0:

```
> YuDifGeo[NormTangentLine]:=
> proc(Line,param,param0):
> YuDifGeo[NormVector](YuDifGeo[TangentLine](#
> Line,param,param0)):
> end:
> YuDifGeo[NormTangentLine]([a*cos(t),a*sin(t),b*t],t,t);
```

$$\left[\begin{array}{ccc} -\frac{a\sin(t)}{\sqrt{a^2+b^2}} & \frac{a\cos(t)}{\sqrt{a^2+b^2}} & \frac{b}{\sqrt{a^2+b^2}} \end{array}\right]$$

Определим команду нахождения единичного вектора бинормали:

```
beta=[r',r'']/|[r',r'']:|
> YuDifGeo[BinormLine]:=
> proc(Line,param,param0):
> YuDifGeo[NormVector](linalg[crossprod](eval(subs(param=
> param0,diff(Line,param))),
> eval(subs(param=param0,diff(Line,param$2))))):
> end:
> YuDifGeo[BinormLine]([a*cos(t),a*sin(t),b*t],t,t);
```

$$\left[\begin{array}{ccc} \frac{ba\sin(t)}{\sqrt{a^2(a^2+b^2)}} & -\frac{ba\cos(t)}{\sqrt{a^2(a^2+b^2)}} & \frac{a^2}{\sqrt{a^2(a^2+b^2)}} \end{array}\right]$$

Определим команду нахождения единичного вектора главной нормали n=[tau,beta]:

```
> YuDifGeo[MainNormLine]:=
> proc(Line,param,param0):
> simplify(linalg[crossprod](#
> YuDifGeo[NormTangentLine](Line,param,param0),
> YuDifGeo[BinormLine](Line,param,param0))):end:
```

```
> YuDifGeo[MainNormLine]([a*cos(t),a*sin(t),b*t],t,t);
```

$$\left[\begin{array}{ccc} \frac{a\cos(t)\sqrt{a^2+b^2}}{\sqrt{a^2(a^2+b^2)}} & \frac{a\sin(t)\sqrt{a^2+b^2}}{\sqrt{a^2(a^2+b^2)}} & 0 \end{array}\right]$$

Определим команду вычисления кривизны кривой в точке param0:

```
> YuDifGeo[CurvatureLine]:=proc(Line,param,param0)
> local T1,N1:
> T1:=subs(param=param0,diff(Line,param)):
> N1:=subs(param=param0,diff(Line,param$2)):
> simplify(YuDifGeo[ModVector](linalg[crossprod](T1,N1))/
> YuDifGeo[ModVector](T1)^3):
> end:
```

```
> YuDifGeo[CurvatureLine]([a*cos(t),a*sin(t),b*t],t,tau);
```

$$\frac{\sqrt{a^2\left(a^2+b^2\right)}}{\left(a^2+b^2\right)^{3/2}}$$

Определим команду смешанного произведения трех векторов:

```
> YuDifGeo[CombProd]:=proc(x,y,z):
> simplify(linalg[innerprod](x,linalg[crossprod](y,z))):
> end proc:
> YuDifGeo[CombProd](#
> [x,y,z],[1,2,3],[4,5,6]);
```

$$-3\,x+6\,y-3\,z$$

Определим команду вычисления кручения кривой в точке param0:

```
> YuDifGeo[TorsionLine]:=
> proc(Line,param,param0) local T1,T2,T3,TTT:
> T1:=subs(param=param0,diff(Line,param)):
> T2:=subs(param=param0,diff(Line,param$2)):
> T3:=subs(param=param0,diff(Line,param$3)):
> TTT:=
> simplify(YuDifGeo[ModVector](linalg[crossprod](T1,T2))):
> simplify(YuDifGeo[CombProd](T1,T2,T3)/TTT^2):end:
```

Посмотрим, как это работает:

```
> YuDifGeo[TorsionLine]([a*cos(t),a*sin(t),b*t],t,tau);
```

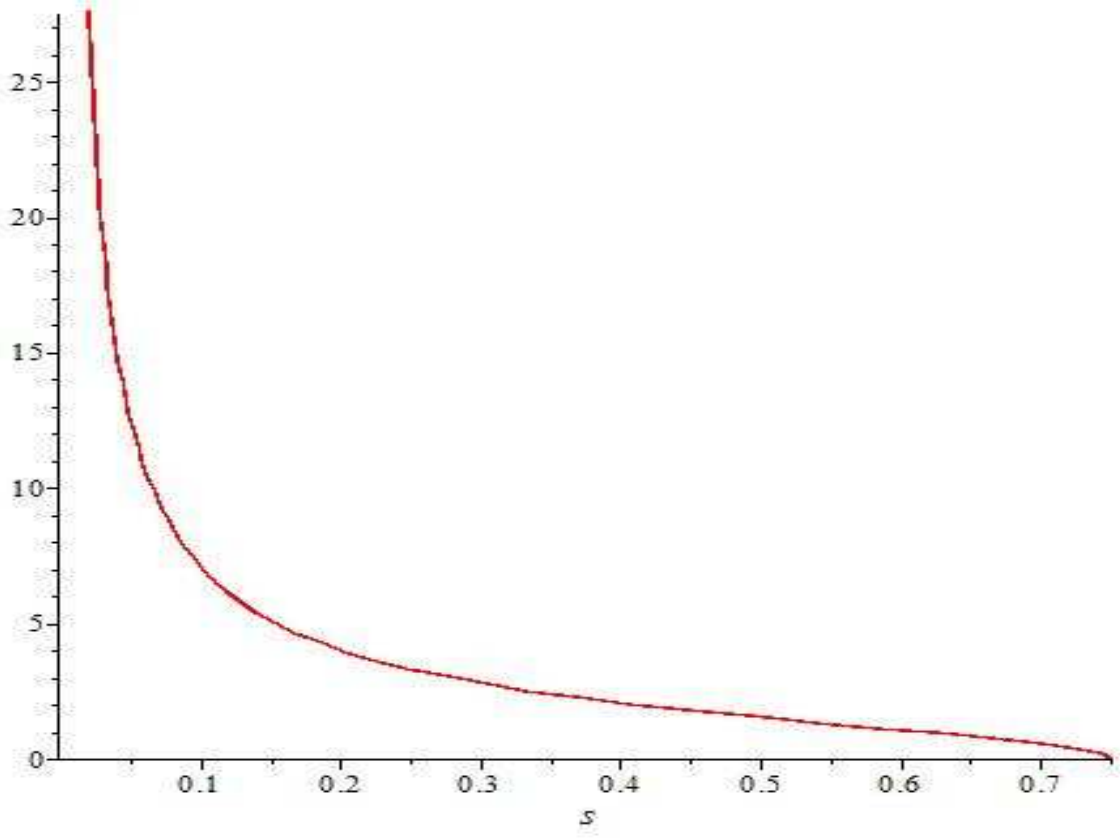
$$\frac{b}{a^2+b^2}$$

III.2.1 Команды изображения в теории дифференциальной геометрии кривых

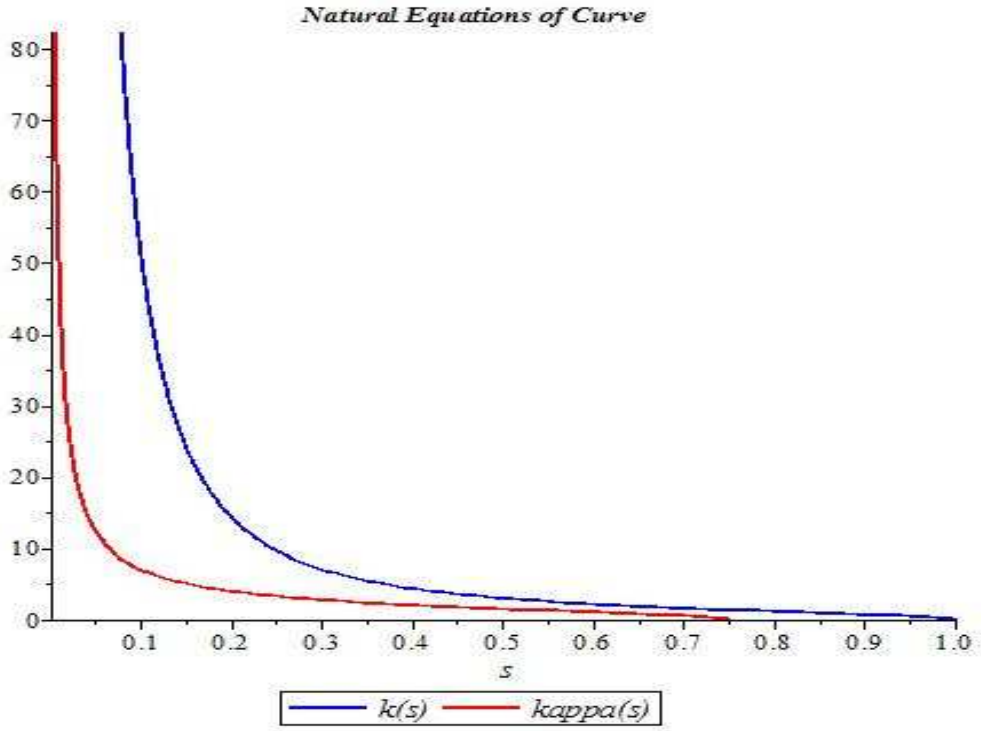
Команда изображения натуральных уравнений кривых

$$k = k(s); kappa = kappa(s) :$$

```
> YuDifGeo[GraphicNaturalCurvature]:=
> proc(Line,param,param0,max_param,cc)
> local param1:
> plot([YuDifGeo[CurvatureLine](Line,param,param1),
> YuDifGeo[LengthLine](Line,param,param0,param1),
> param1=param0..max_param],color=cc
> ,labelfont=[TIMES,ROMAN,12],labels=[‘s‘,‘‘]):end:
> YuDifGeo[GraphicNaturalTorsion]:=
> proc(Line,param,param0,max_param,cc)
> local param1:
> plot([YuDifGeo[TorsionLine](Line,param,param1),
> YuDifGeo[LengthLine](Line,param,param0,param1),
> param1=param0..max_param],color=cc
> ,labelfont=[TIMES,ROMAN,12],labels=[‘s‘,‘‘]):end:
> #
> YuDifGeo[GraphicNaturalTorsion]([t*cos(t),t*sin(t),t],\
> t,0,8*Pi,red);
```

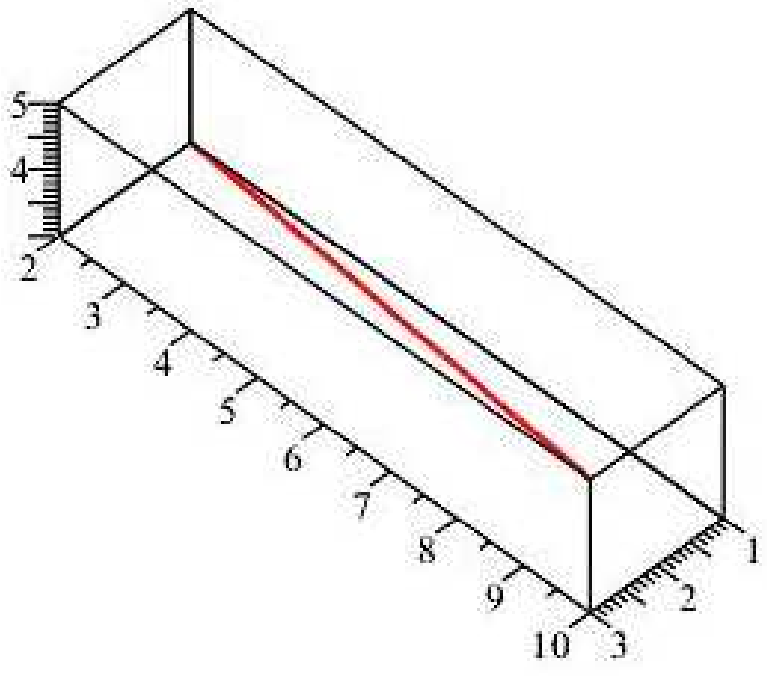


```
>YuDifGeo[GraphicNaturalEquations]:=
proc(Line,param,param0,max_param,cc1,cc2)
local param1:
plot([[YuDifGeo[CurvatureLine](Line,param,param1),
YuDifGeo[LengthLine](Line,param,param0,param1),
param1=param0..max_param],
[YuDifGeo[TorsionLine](Line,param,param1),
YuDifGeo[LengthLine](Line,param,param0,param1),
param1=param0..max_param]],color=[cc1,cc2],
labelfont=[TIMES,ROMAN,12],labels=[‘s‘,‘‘],
legend=[‘k(s)‘,‘kappa(s)‘],
titlefont=[TIMES,ROMAN,BOLD],
title=‘Natural Equations of Curve‘):end:
  > YuDifGeo[GraphicNaturalEquations]([t*cos(t),
  > t*sin(t),t],t,0,4*Pi,blue,red);
```



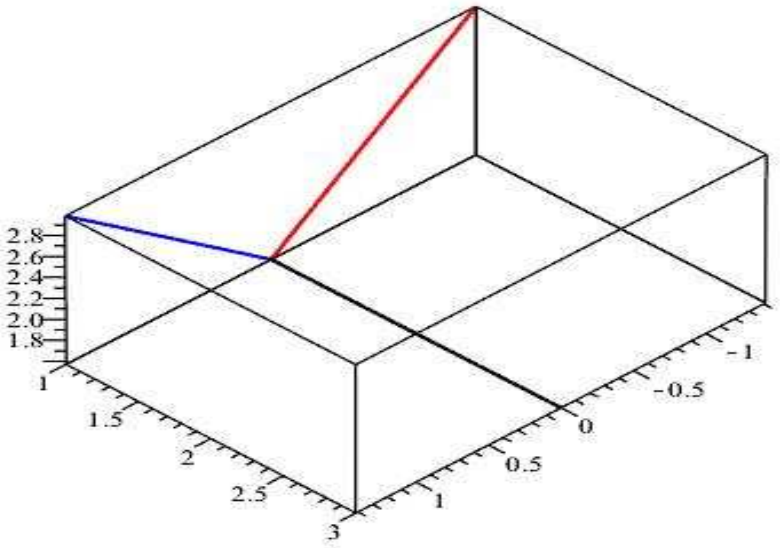
Команда изображения отрезка

```
  > YuDifGeo[ParamLine]:=proc(X,q,param,t1,c) local EQ:
  > EQ:=linalg[matadd](X,linalg[scalarmul](q,param),1,1):
  > plots[spacecurve](EQ,param=0..t1,thickness=2,
  > color=c,scaling=CONSTRAINED):
  > end:
  > YuDifGeo[ParamLine]([1,2,3],[1,4,1],xi,2,red);
```



III.2.2 Команды изображения адаптированного репера с началом в точке t0

```
> YuDifGeo[NaturalReper]:=
> proc(Line,param,param0,t1,cc1,cc2,cc3)
> local rt1,rt2,rt3,tt,bb,nn,xi,zeta,chi:
> tt:=YuDifGeo[NormTangentLine](Line,param,param0):
> rt1:=YuDifGeo[ParamLine](subs(param=param0,Line),
> tt,xi,t1,cc1,scaling=CONSTRAINED):
> bb:=YuDifGeo[BinormLine](Line,param,param0):
> rt2:=YuDifGeo[ParamLine](subs(param=param0,Line),
> bb,zeta,t1,cc2,scaling=CONSTRAINED):
> nn:=YuDifGeo[MainNormLine](Line,param,param0):
> rt3:=YuDifGeo[ParamLine](subs(param=param0,Line),
> nn,chi,t1,cc3,scaling=CONSTRAINED):
> plots[display](rt1,rt2,rt3):end:

> YuDifGeo[NaturalReper]([cos(t),sin(t),t],\
> t,Pi/2,2,red,blue,black);
```



Команда анимации натурального репера кривой:

Сначала вычислем длину линии

```
> SS:=(Line,param,mini,maxi,i)->
> evalf(YuDifGeo[LengthLine](Line,param,mini,\
> (maxi-mini)/71*i),6):
> SS([cos(t),sin(t),0.2*t],t,0,8*Pi,10);
```

3.60991

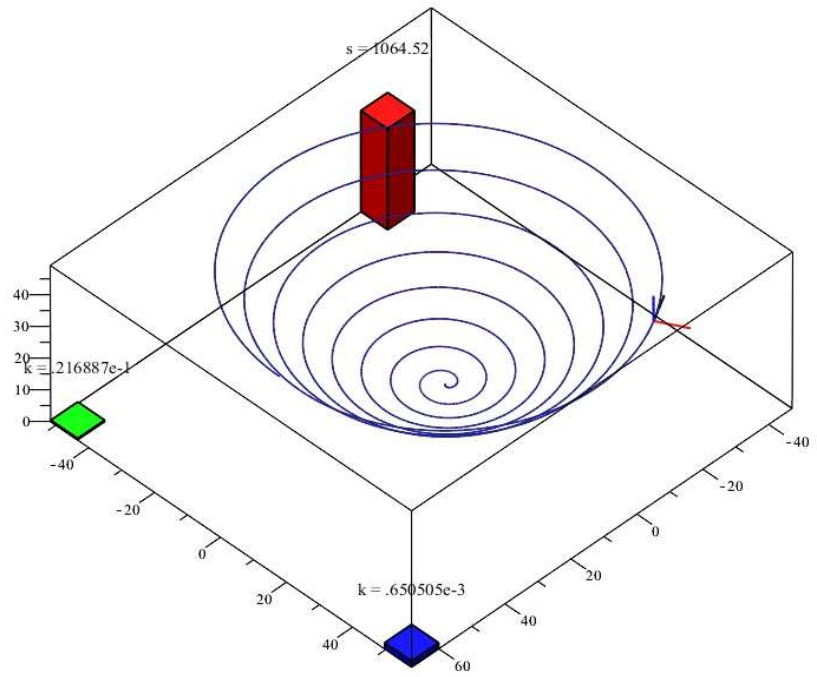
Строим процедуру динамической визуализации натурального репера кривой. Задаем параметры репера:

```
>YuDifGeo[AnimNaturalReper]:=
proc(Line,param,mini,maxi,c,N,p) local
LINE,T,GL,TTT,S,kk,KKK,XXX,
x_min,x_max,y_min,y_max,z_min,z_max,l_x,l_y,l_z,
SS,a1,a2,i,t1,s_max,s_norm,k_max,k_min,k_norm,
K_max,K_min,K_norm,CC,SSS,TTS,kkk,KKKK,TTk,TTK:
LINE:=(T)->subs(param=T,Line):
GL:=plots[spacecurve](LINE(param),param=mini..maxi,thickness=1,
color=navy,scaling=CONSTRAINED,numpoints=1000):
#Вычисляем значение
#длины линии
SS:=(i)->
```

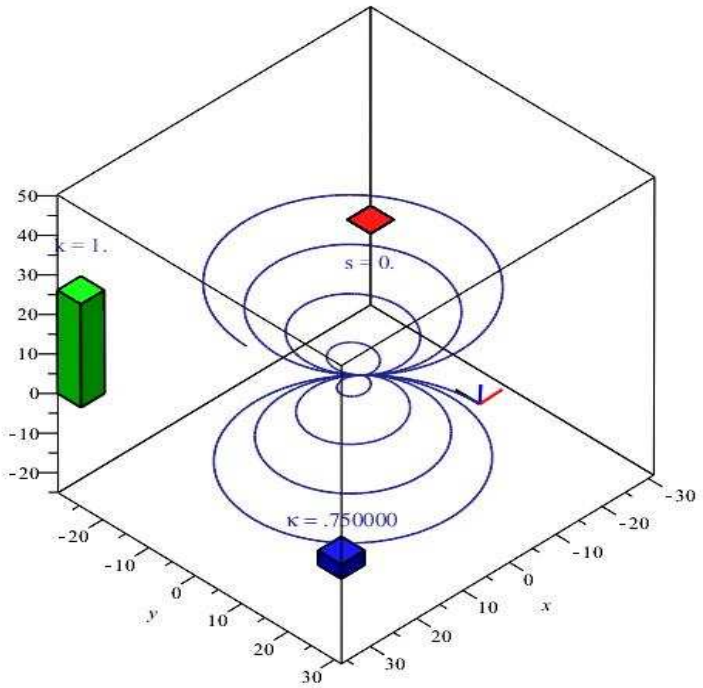
```
evalf(YuDifGeo[LengthLine](Line,param,mini,
mini+(maxi-mini)/N*i),6):
#кривизну линии
kk:=(i)->
evalf(YuDifGeo[CurvatureLine](Line,param,(maxi-mini)/N*i),6):
#кручение линии
KKK:=(i)->
evalf(YuDifGeo[TorsionLine](Line,param,(maxi-mini)/N*i),6):
#строим график
TTT:=(i)->
plots[textplot3d]([0,0,0,convert(SS(i),string)],\\
color=c,align=LEFT):
XXX:=(i)->
evalf(LINE(mini+i*(maxi-mini)/N)):
#вычислим минимальные и максимальные x, y, z
x_min:=min(seq(XXX(i)[1],i=0..N)):
x_max:=max(seq(XXX(i)[1],i=0..N)):
y_min:=min(seq(XXX(i)[2],i=0..N)):
y_max:=max(seq(XXX(i)[2],i=0..N)):
z_min:=min(seq(XXX(i)[3],i=0..N)):
z_max:=max(seq(XXX(i)[3],i=0..N)):
#затем вычисляем длины  векторов по x,y,z
l_x:=x_max-x_min:
l_y:=y_max-y_min:
l_z:=z_max-z_min:
#Задаем величину репера по отношению к рисунку
t1:=0.1*max(l_x,l_y,l_z):
####
s_max:=YuDifGeo[LengthLine](Line,param,mini,maxi):
s_norm:=(i)->evalf(SS(i)/s_max*l_z):
k_max:=max(seq(kk(i),i=0..N)):
k_min:=min(seq(kk(i),i=0..N)):
k_norm:=(i)->evalf(kk(i)/(k_max-k_min)*l_z):
K_max:=max(seq(KKK(i),i=0..N)):
K_min:=min(seq(KKK(i),i=0..N)):
K_norm:=(i)->evalf(KKK(i)/(K_max-K_min)*l_z):
CC:=[x_min+l_x/2,y_min+l_y/2,z_min]:
SSS:=(i)->plot3d(s_norm(i)+0.01,x=x_min-1.2*t1..x_min-1.2*t1+t1,
```

```
y=y_min-1.2*t1..y_min-1.2*t1+t1,filled=true,
color=red,grid=[2,2],scaling=CONSTRAINED):
TTS:=(i)->plots[textplot3d]([x_min-1.2*t1,y_min-1.2*t1,
z_min+s_norm(i+1)+t1,convert(s=SS(i),string)],
align=above,color=NAVY,font=[TIMES,ROMAN,12]):
kkk:=(i)->plot3d(z_min+k_norm(i),
x=x_max+1.2*t1..x_max+1.2*t1+t1,
y=y_min-1.2*t1..y_min-1.2*t1+t1,filled=true,color=green,
grid=[2,2],scaling=CONSTRAINED):
TTk:=(i)->plots[textplot3d]([x_max+1.2*t1,y_min-1.2*t1,
z_min+k_norm(i)+t1,convert(k=kk(i),string)],
align=above,color=NAVY,font=[TIMES,ROMAN,12]):
KKKK:=(i)->plot3d(z_min+K_norm(i+1),
x=x_max+1.2*t1..x_max+1.2*t1+t1,
y=y_max+1.2*t1..y_max+1.2*t1+t1,filled=true,color=blue,
grid=[2,2],scaling=CONSTRAINED):
TTK:=(i)->plots[textplot3d]([x_max+1.2*t1,
y_max+1.2*t1,z_min+K_norm(i+1)+t1,convert(k=KKK(i),string)],
align=above,color=NAVY,font=[SYMBOL,12]):
#Строим график динамической визуализации репера кривой
if p=1 then
S:=(i)->plots[display](
YuDifGeo[NaturalReper](Line,param,mini+
(maxi-mini)/N*i,t1,black,blue,red),
GL,title=convert([s=SS(i),k=kk(i),
kappa=KKK(i)],string)):
else
S:=(i)->plots[display](YuDifGeo[NaturalReper](Line,
param,mini+(maxi-mini)/N*i,t1,black,blue,red),
GL,SSS(i),TTS(i),kkk(i),TTk(i),KKKK(i),TTK(i)):
end if:
plots[display](seq(S(i),i=0..N),insequence=true,
scaling=CONSTRAINED):
#[seq([s_norm(i)],i=1..N),l_z]:#k_norm(2):
end proc:
```

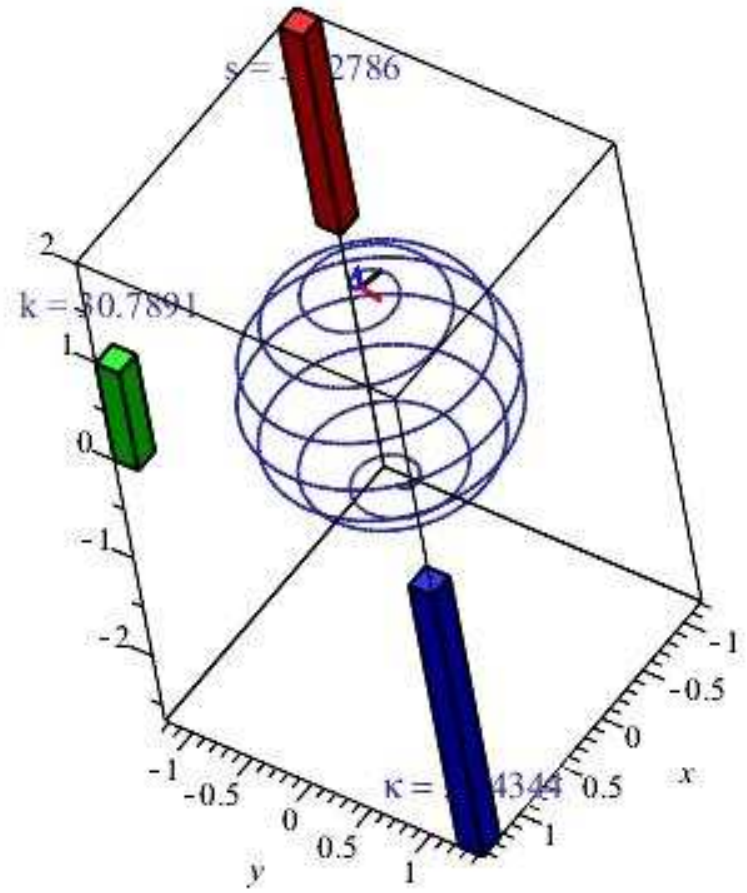
```
> YuDifGeo[AnimNaturalReper]([t*cos(t),t*sin(t),0.015*t^2],
> t,0,16*Pi,red,24,xxx);
```



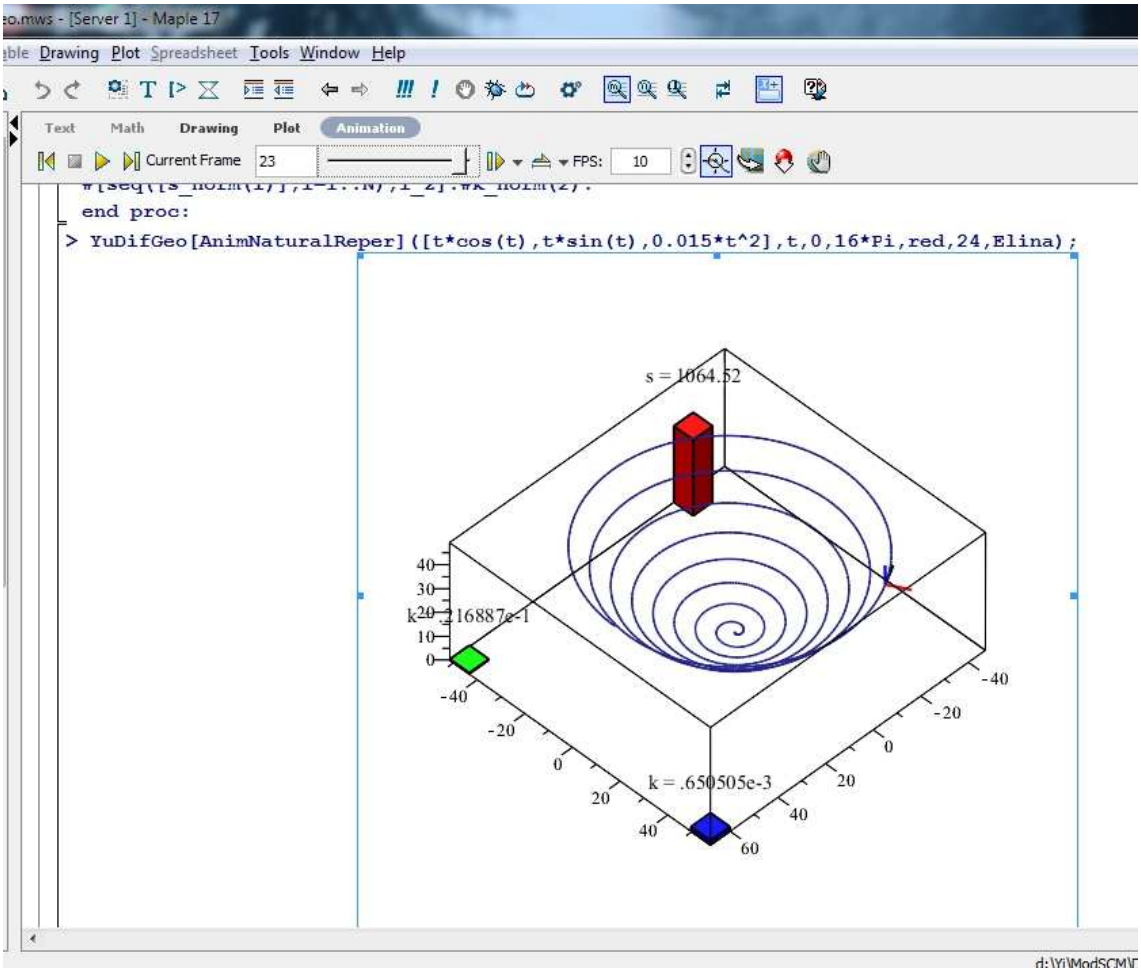

```
> YuDifGeo[AnimNaturalReper]([t*cos(t),t*sin(t),t],
> t,-8*Pi,8*Pi,red,32,0.1);
```



```
> YuDifGeo[AnimNaturalReper]([cos(t)*cos(-Pi/2+t/16),
> sin(t)*cos(-Pi/2+t/16),sin(-Pi/2+t/16)],\
> t,0.1,16*Pi-0.1,red,64,xxx);
```




```
> save(YuDifGeo,‘DifGeo.m‘);
```



III.3 Восстановление кривой по ее натуральным уравнениям

Инвариантами произвольной кривой является ее длина дуги, кривизна и кручение. Если кривая отнесена к натуральному параметру, то ее кривизна и кручение являются функциями этого параметра

$$k = k(s); \quad \varkappa = \varkappa(s). \tag{III.7}$$

Система этих двух соотношений называется *натуральными уравнениями кривой*. Так как натуральное уравнение связывает инварианты кривой, то оно не меняется при преобразовании координат или при перемещении кривой относительно этой системы. Основное значение натуральных уравнений состоит в том, что задание их вполне характеризует форму кривой, так что *две кривые с одинаковыми натуральными уравнениями необходимо совпадают по своей форме и могут отличаться только положением пространства,*

т.е., являются конгруэнтными фигурами[5]. С другой стороны, кривизна и кручение кривой, являются ее двумя локально измеряемыми характеристиками, полностью описывающими свойства кривой: кривизна, $k(s)$, совпадает с абсолютной величиной скорости поворота касательной к кривой, а абсолютная величина кручения, $\varkappa$, – с абсолютной величиной скорости поворота бинормали кривой. Эти две физически измеряемые характеристики кривой являются основой для ориентации на местности и обеспечивают мгновенную привязку к местности мобильных технических средств. Фактическое измерение этих скаляров осуществляется с помощью гиросистем. Математически вопрос привязки перемещаемого по некоторой пространственной кривой $\vec{r} = \vec{r}(t)$ объекта заключается в разрешении натуральных уравнений кривой относительно радиуса-вектора перемещаемой точки. На первый взгляд кажется, что невозможно однозначно восстановить векторную функцию $\vec{r}(t)$ по системе двух натуральных уравнений (III.7). Однако, это возможно сделать при задании начальной точки и начального направления – как раз вследствие указанного свойства конгруэнтности.

Кривизна и кручение кривой, заданной своим параметрическим уравнением, описываются формулами (см., например, [125]):

$$k(t) = \frac{|\dot{\mathbf{r}} \times \ddot{\mathbf{r}}|}{|\dot{\mathbf{r}}^3}; \tag{III.8}$$

$$\varkappa(t) = \frac{(\dot{\mathbf{r}}, \ddot{\mathbf{r}}, \dddot{\mathbf{r}})}{|\dot{\mathbf{r}} \times \ddot{\mathbf{r}}|^2}. \tag{III.9}$$

Таким образом, задача восстановления кривой (с точностью до пространственного движения) сводится к решению системы двух обыкновенных, существенно нелинейных дифференциальных уравнений (III.8), (III.9):

$$k(s) = \frac{|\dot{\mathbf{r}} \times \ddot{\mathbf{r}}|}{|\dot{\mathbf{r}}^3}; \tag{III.10}$$

$$\varkappa(s) = \frac{(\dot{\mathbf{r}}, \ddot{\mathbf{r}}, \dddot{\mathbf{r}})}{|\dot{\mathbf{r}} \times \ddot{\mathbf{r}}|^2}. \tag{III.11}$$

с определением связи производных по натуральному параметру$<$ s, (длине) и временем, t:

$$\frac{dt}{ds} = \frac{1}{\dot{\mathbf{r}}}. \tag{III.12}$$

[5]См., например, [35]

III.3.1 Задание кривой и вычисление кривизны и кручения по натуральному параметру

Предположим, что нам известны параметрические уравнения кривой в натуральной параметризации, т.е., заданы три функции:

$$x = X(s);\ y = Y(s);\ z = Z(s) : \tag{III.13}$$

Зададим эту операцию в Maple:

```
> restart:
> R(s):=[X(s),Y(s),Z(s)];
```

$$\mathrm{R}(s) := [\mathrm{X}(s),\ \mathrm{Y}(s),\ \mathrm{Z}(s)]$$

Производные первого порядка:

```
> tau_1(s):=[diff(X(s),s),diff(Y(s),s),diff(Z(s),s)];
```

$$\mathrm{tau\_1}(s) := [\tfrac{d}{ds}\,\mathrm{X}(s),\ \tfrac{d}{ds}\,\mathrm{Y}(s),\ \tfrac{d}{ds}\,\mathrm{Z}(s)]$$

Применим теперь следующий прием Ю.Г. Игнатьева [126]. Учтем, что вектор τ_1 имеет единичную длину:

$$\left|\tau_1\right| = 1. \tag{III.14}$$

Вследствие этого как раз и исчезает лишняя степень свободы, поэтому координаты этого вектора можно параметризовать двумя переменными, $\Phi(s)$ и $\Theta(s)$ (аналогичным сферическим координатам):

$$\frac{dX}{ds} = \cos\Phi\cos\Theta;\ \frac{dY}{ds} = \sin\Phi\cos\Theta;\ \frac{dZ}{ds} = \sin\Theta : \tag{III.15}$$

$\{\ \frac{d}{ds}\,\mathrm{X}(s) = \cos(\Phi(s))\cos(\Theta(s))\ ;\ \frac{d}{ds}\,\mathrm{Y}(s) = \sin(\Phi(s))\cos(\Theta(s))\ ;$
$\frac{d}{ds}\,\mathrm{Z}(s) = \sin(\Theta(s))\ ,$

где $\Phi(s)$, $\Theta(s)$ - произвольные пока функции натурального параметра.

Сделаем указанные подстановки и вычислим затем производную радиуса вектора в новых переменных:

```
> DR:=
{diff(X(s),s)=cos(Phi(s))*cos(Theta(s)),diff(Y(s),s)
=sin(Phi(s))*cos(Theta(s)), diff(Z(s),s)=sin(Theta(s))};
```

$$DR := \{\frac{d}{ds}\,\mathrm{X}(s) = \cos(\Phi(s))\cos(\Theta(s)),$$
$$\frac{d}{ds}\,\mathrm{Y}(s) = \sin(\Phi(s))\cos(\Theta(s)),\ \frac{d}{ds}\,\mathrm{Z}(s) = \sin(\Theta(s))\}$$

Запишем производную радиуса вектора, $\tau(s)$, с помощью новых переменных:

```
> tau(s):=subs(DR,tau_1(s));
> u(s):=tau(s)[1];v(s):=tau(s)[2];w(s):=tau(s)[3];
```

$$\tau(s) := [\cos(\Phi(s))\cos(\Theta(s)),\ \sin(\Phi(s))\cos(\Theta(s)),\ \sin(\Theta(s))]$$

$$\mathrm{u}(s) := \cos(\Phi(s))\cos(\Theta(s))$$

$$\mathrm{v}(s) := \sin(\Phi(s))\cos(\Theta(s))$$

$$\mathrm{w}(s) := \sin(\Theta(s))$$

Производные второго порядка:

```
> R2(s):=  [diff(u(s),s),diff(v(s),s),diff(w(s),s)];
```

$$\mathrm{R2}(s) := [-\sin(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))\cos(\Theta(s)) -$$
$$\cos(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s)),$$
$$\cos(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))\cos(\Theta(s)) - \sin(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s)),$$
$$\cos(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))]$$

Производные третьего порядка:

```
> R3(s):=  [diff(u(s),s$2),diff(v(s),s$2),diff(w(s),s$2)];
```

$$\mathrm{R3}(s) := [-\cos(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))^2\cos(\Theta(s)) -$$
$$\sin(\Phi(s))\,(\tfrac{d^2}{ds^2}\,\Phi(s))\cos(\Theta(s))$$
$$+\,2\sin(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s)) -$$
$$\cos(\Phi(s))\cos(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))^2$$
$$-\cos(\Phi(s))\sin(\Theta(s))\,(\tfrac{d^2}{ds^2}\,\Theta(s)), -\sin(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))^2\cos(\Theta(s))$$
$$+\cos(\Phi(s))\,(\tfrac{d^2}{ds^2}\,\Phi(s))\cos(\Theta(s)) -$$
$$2\cos(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))$$
$$-\sin(\Phi(s))\cos(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))^2 - \sin(\Phi(s))\sin(\Theta(s))\,(\tfrac{d^2}{ds^2}\,\Theta(s)),$$
$$-\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))^2 + \cos(\Theta(s))\,(\tfrac{d^2}{ds^2}\,\Theta(s))]$$

III.3.2 Вычисление кривизны и кручения кривой

Предполагая, теперь, что параметрические уравнения кривой Γ известны, вычислим ее кривизну и кручение по стандартным формулам дифференциальной геометрии (III.8), (III.9). При этом скалярное и вектор-

ное произведения векторов **a** и **b** будем вычислять с помощью команд `innerprod(a,b)` и `crossprod(a,b)` библиотеки `linalg`, соответственно:

```
> k(s):=sqrt(simplify(linalg[innerprod](R2(s),R2(s))));
```

$$\mathrm{k}(s) := \sqrt{(\tfrac{d}{ds}\,\Phi(s))^2\cos(\Theta(s))^2 + (\tfrac{d}{ds}\,\Theta(s))^2}$$

С помощью команды `scallarmul(a,alpha` библиотеки `linalg` производем умножение вектора $\mathbf{r}''(s)$ на число α – тем самым получаем единичный вектор ν сопровождающего трехгранника кривой:.

```
> nu(s):=linalg[scalarmul](R2(s),1/k(s));
```

$$\nu(s) := \left[\frac{-\sin(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))\cos(\Theta(s)) - \cos(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))}{\sqrt{\%1}},\right.$$
$$\frac{\cos(\Phi(s))\,(\tfrac{d}{ds}\,\Phi(s))\cos(\Theta(s)) - \sin(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))}{\sqrt{\%1}},$$
$$\left.\frac{\cos(\Theta(s))\,(\tfrac{d}{ds}\,\Theta(s))}{\sqrt{\%1}}\right]$$
$$\%1 := (\tfrac{d}{ds}\,\Phi(s))^2\cos(\Theta(s))^2 + (\tfrac{d}{ds}\,\Theta(s))^2$$

Убедимся в том, что первая и вторая производные радиуса-вектора ортогональны (так как вторая производная есть производная от единичного вектора τ).

```
> simplify(linalg[innerprod](tau(s),nu(s)));
```

$$0$$

Вычислим векторное произведение этих же векторов:

```
> V_DR_D2R(s):=linalg[crossprod](tau(s),R2(s));
```

$$\mathrm{V\_DR\_D2R}(s) := \Big[\sin(\Phi(s))\cos(\Theta(s))^2\,(\tfrac{d}{ds}\Theta(s))$$
$$-\sin(\Theta(s))\,(\cos(\Phi(s))\,(\tfrac{d}{ds}\Phi(s))\cos(\Theta(s))-$$
$$\sin(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\Theta(s))),$$
$$\sin(\Theta(s))\,(-\sin(\Phi(s))\,(\tfrac{d}{ds}\Phi(s))\cos(\Theta(s))-$$
$$\cos(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\Theta(s)))$$
$$-\cos(\Phi(s))\cos(\Theta(s))^2\,(\tfrac{d}{ds}\Theta(s)),$$
$$\cos(\Phi(s))\cos(\Theta(s))\,(\cos(\Phi(s))\,(\tfrac{d}{ds}\Phi(s))\cos(\Theta(s))$$
$$-\sin(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\Theta(s)))-\sin(\Phi(s))\cos(\Theta(s))$$
$$(-\sin(\Phi(s))\,(\tfrac{d}{ds}\Phi(s))\cos(\Theta(s))-\cos(\Phi(s))\sin(\Theta(s))\,(\tfrac{d}{ds}\Theta(s)))\Big]$$

С помощью скалярного умножения полученного результата и вектора третьей производной получим кручение кривой: $\kappa(s)$:

```
> kappa(s):=
> simplify(linalg[innerprod](V_DR_D2R(s),R3(s))/k(s)^2);
```

$$\kappa(s) := (\sin(\Theta(s))\,(\tfrac{d}{ds}\Phi(s))^3\cos(\Theta(s))^2 + 2\sin(\Theta(s))\,(\tfrac{d}{ds}\Theta(s))^2\,(\tfrac{d}{ds}\Phi(s))$$
$$-(\tfrac{d}{ds}\Theta(s))\,(\tfrac{d^2}{ds^2}\Phi(s))\cos(\Theta(s)) + (\tfrac{d}{ds}\Phi(s))\cos(\Theta(s))\,(\tfrac{d^2}{ds^2}\Theta(s)))\Big/($$
$$(\tfrac{d}{ds}\Phi(s))^2\cos(\Theta(s))^2 + (\tfrac{d}{ds}\Theta(s))^2)$$

$$\kappa(s) := ((\tfrac{d}{ds}\Phi(s))\cos(\Theta(s))\,(\tfrac{d^2}{ds^2}\Theta(s)) - (\tfrac{d}{ds}\Theta(s))\,(\tfrac{d^2}{ds^2}\Phi(s))\cos(\Theta(s))$$
$$+ 2\sin(\Theta(s))\,(\tfrac{d}{ds}\Theta(s))^2\,(\tfrac{d}{ds}\Phi(s)) + \sin(\Theta(s))\,(\tfrac{d}{ds}\Phi(s))^3\cos(\Theta(s))^2)\Big/($$
$$(\tfrac{d}{ds}\Phi(s))^2\cos(\Theta(s))^2 + (\tfrac{d}{ds}\Theta(s))^2)$$

III.3.3 Формирование системы дифференциальных уравнений

Напомним, что натуральные уравнения кривой — это пара уравнений:

$$k = k(s); \tag{III.16}$$
$$\kappa = \kappa(s), \tag{III.17}$$

где $k(s)$ и $\kappa(s)$ в правых частях заданы как функции натурального параметра s. Правые части этих уравнений мы будем задавать, а левые части

этих уравнений определяются дифференциальными выражениями, которые мы получили выше. Таким образом, мы получаем систему двух обыкновенных дифференциальных уравнений второго порядка относительно двух неизвестных функций — $\Phi(s)$ и $\Theta(s)$.

Для численного интегрирования этой системы необходимо привести ее к нормальному виду, т.е., к системе ОДУ 1-го порядка, разрешенных относительно производных. Для этого обозначим первые производные от искомых функций с помощью новых функций, $\Phi_S(s)$ и $\Theta_S(s)$:

$$\frac{d\Phi(s)}{ds} = \Phi_S(s); \tag{III.18}$$

$$\frac{d\Theta(s)}{ds} = \Theta_S(s)\,: \tag{III.19}$$

```
D1:=diff(Phi(s),s)=Phi[S](s);D2:=diff(Theta(s),s)=Theta[S](s);
```

Система ОДУ решается с помощью команды
`>dsolve(система ОДУ union Начальные условия, type=numeric, method=classical, output=listprocedure):`
— при этом получаем численные решения классическим методом, который является комбинацией методов Эйлера, Хейца, Рунге - Куддта и Адамса - Бэшфорда, причем решения выводятся в виде списка. Как показывают вычисления, этот комбинированный метод дает лучшие результаты, чем стандартный метод Рунге - Кудта в случаях, когда кривая имеет много изгибов.

```
> Sol[1]:= dsolve(System1 union Inits,
{X(s),Y(s),Z(s),Phi(s),Theta(s),Phi[S](s),Theta[S](s)},
type=numeric, method=classical,output=listprocedure);
Sol[2]:= dsolve(System2 union Inits, {X(s),Y(s),Z(s),
Phi(s),Theta(s),Phi[S](s),Theta[S](s)},
type=numeric,method=classical, output=listprocedure);
Sol[3]:= dsolve(System3 union Inits,
{X(s),Y(s),Z(s),Phi(s),Theta(s),Phi[S](s),Theta[S](s)},
type=numeric,method=classical, output=listprocedure);
Sol[4]:= dsolve(System4 union Inits, {X(s),Y(s),Z(s),
Phi(s),Theta(s),Phi[S](s),Theta[S](s)},
type=numeric,method=classical, output=listprocedure);
```

III.3.4 Компьютерное моделирование натуральных уравнений кривой

Подставляя теперь в правую часть натуральных уравнений кривой полученные выражения для кривизны и кручения, а в левые – заданные функции параметра s, получим систему двух нелинейных обыкновенных дифференциальных уравнений 3-го порядка относительно двух углов $\Phi(s)$ и $\Theta(s)$. Как мы отмечали выше, точка начала кривой и ее начальное направление могут быть заданы произвольно. Соответственно этому произволу зададим начальные условия в виде:

```
>InitConditions:={X(0)=0,Y(0)=0,Z(0)=0,Phi(0)=0,
        Theta(0)=0,Phi[S](0)=1,Theta[S](0)=0};
```

Зададим теперь конкретные натуральные уравнения кривой, т.е., конкретизируем функции $\overset{\circ}{(}s)$ и $\varkappa(s)$.Рассмотрим несколько примеров. На последующих рисунках представлены результаты компьютерного восстановления кривых по их натуральным уравнениям.

Кривая 1:

$$k(s) = 1, \varkappa(s) = 0. \tag{III.20}$$

Эта кривая, как понятно из геометрического смысла кривизны и кручения, является единичной окружностью. Одновременно – это хороший пример для проверки работы комплекса программ.

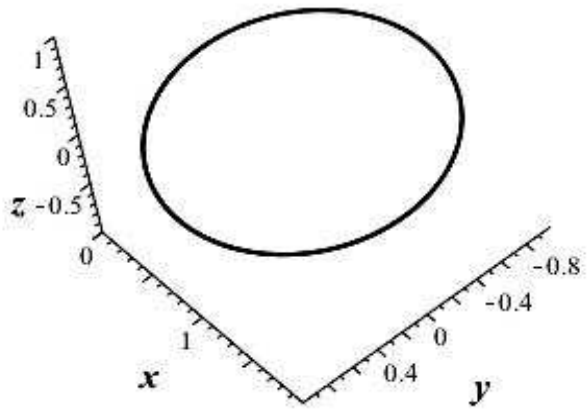


Рис.III.29. Восстановление кривой 1 по ее натуральным уравнениям (III.20). Кривая Γ - единичная окружность в пространстве, как и следует из теории [125].

Кривая 2:

$$k(s) = \frac{s^2}{1+s^4};\ \varkappa(s) = -\frac{s^2}{1+s^2}. \tag{III.21}$$

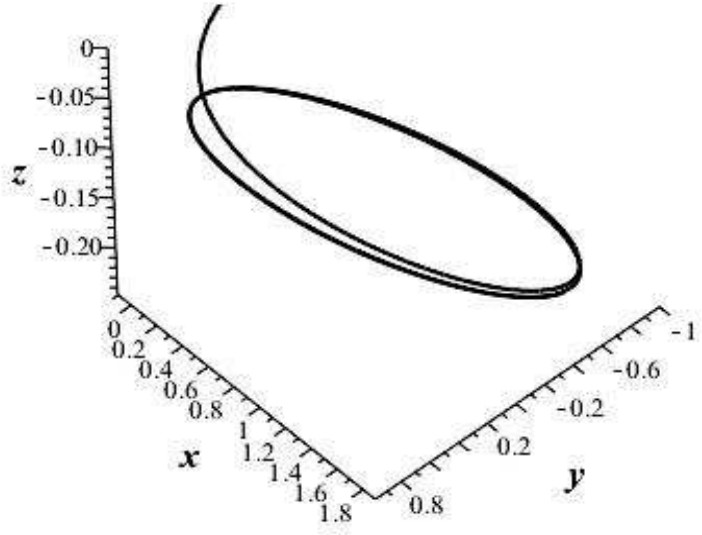


Рис.III.30. Восстановление кривой 2 по ее натуральным уравнениям (III.21). Кривая Γ - искаженная винтовая линия.

Из рисунка видно, что эту кривую можно охарактеризовать как деформированную винтовую линию с переменным шагом и наклонной осью.

Кривая 3:

$$k(s) = \cos s;\ \varkappa(s) = \sin s. \tag{III.22}$$

Натуральные уравнения этой кривой приведены в книге А.П. Нордена [125].

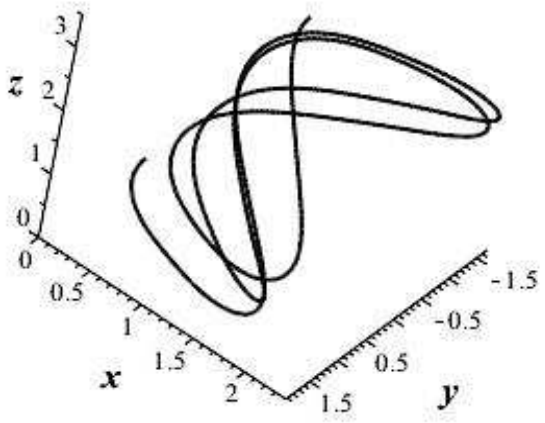


Рис.III.31. Восстановление кривой 3 по ее натуральным уравнениям (III.22).

Глава IV

Моделирование геодезических линий

IV.1 Математическая модель геодезических линий

С излагаемой здесь математической теорией геодезических линий можно ознакомиться в учебниках по дифференциальной и римановой геометрии, например, [126].

IV.1.1 Геодезическая кривизна и геодезические линии

Определение геодезической кривизны

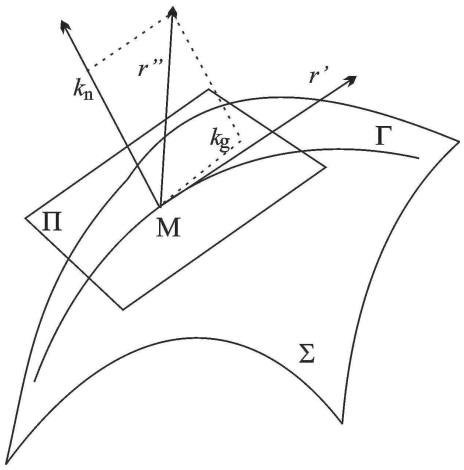


Рис.IV.32. *Геодезическая и нормальная кривизна линий на поверхности*

Определение OIV.1. *Геодезической кривизной k_g линии Γ на поверхности в некоторой ее точке называется абсолютная величина проекции второй производной радиуса-вектора этой кривой на касательную плоскость:*

$$k_g = |Пр_{кас.\ пл.}\vec{r}''| \qquad \text{(IV.1)}$$

.

Получим выражение для геодезической кривизны линии

$$\Gamma: \quad \vec{r} = \vec{r}(u(t), v(t)) \qquad \text{(IV.2)}$$

на поверхности Σ. Нормальный вектор к поверхности выражается через касательные векторы к координатным линиям:

$$\vec{N} = \left[\vec{r}_u \cdot \vec{r}_v \right]. \tag{IV.3}$$

Нормальная проекция вектора ускорения равна:

$$\mathrm{Пр}_{\vec{N}}\, \vec{r}'' = \frac{\left(\vec{r}'' \cdot \vec{N} \right)}{|\vec{N}|}.$$

Таким образом, согласно определению IV.1 по теореме Пифагора получим:

$$k_g^2 = (\vec{r}'')^2 - \frac{\left(\vec{r}'' \cdot \vec{N} \right)^2}{|\vec{N}|^2} \equiv \frac{1}{\vec{N}^2} \left(\vec{N}^2 - \left(\vec{r}'' \cdot \vec{N} \right)^2 \right).$$

Применяя здесь тождество Лагранжа:

$$\vec{a}^{\,2} \vec{b}^{\,2} = \left[\vec{a}\ \vec{b} \right]^2 + \left(\vec{a}\ \vec{b} \right)^2,$$

получим более удобное выражение для геодезической кривизны линии:

$$k_g^2 = \frac{1}{\vec{N}^2} \left[\vec{r}'' \cdot \vec{N} \right]^2 \Longrightarrow k_g = \frac{1}{|\vec{N}|} \left| \left[\vec{r}'' \cdot \vec{N} \right] \right|. \tag{IV.4}$$

Используя IV.3 и затем — формулу раскрытия двойного векторного произведения:

$$\left[\vec{a} \ \left[\vec{b}\ \vec{c} \right] \right] = \vec{b} \left(\vec{a}\ \vec{c} \right) - \vec{c} \left(\vec{a}\ \vec{b} \right),$$

получим с учетом определения коэффициентов первой квадратичной формы:

$$k_g^2 = \frac{1}{EG - F^2} \left(E \left(\vec{r}'' \cdot \vec{r}_v \right)^2 - 2F \left(\vec{r}'' \cdot \vec{r}_v \right) \left(\vec{r}'' \cdot \vec{r}_u \right) + G \left(\vec{r}'' \cdot \vec{r}_u \right)^2 \right). \tag{IV.5}$$

Заметим, что поскольку геодезическая кривизна определяется проекцией вектора ускорения на касательную плоскость, то она должна полностью определяться коэффициентами первой квадратичной формы и их производными[1], таким образом, геодезическая кривизна поверхности не изменяется при изгибании.

[1] В этом факте мы убедимся ниже.

IV.1.2 Геодезические линии и уравнения геодезических линий

Определение OIV.2. *Линия* Γ *поверхности* Σ *называется геодезической, если ее геодезическая кривизна во всех точках равна нулю.*

Таким образом, согласно IV.4 для геодезической линии:

$$\left|\left[\vec{r}'' \cdot \vec{N}\right]\right| = 0.$$

Но вектор нулевой длины имеет и нулевые координаты, следовательно:

$$\left[\vec{r}'' \cdot \vec{N}\right] = \vec{0}. \tag{IV.6}$$

Применяя здесь формулу упрощения двойного векторного произведения, найдем:

$$\vec{r}_u\left(\vec{r}'' \cdot \vec{r}_v\right) - \vec{r}_v\left(\vec{r}'' \cdot \vec{r}_u\right) = \vec{0}.$$

Поскольку векторы $\vec{r}_u$ и $\vec{r}_v$ в неособых точках поверхности неколлинеарны, то в последнем выражении должны обращаться в нуль скалярные произведения. Таким образом, получим для геодезической линии уравнения:

$$\left(\vec{r}'' \cdot \vec{r}_u\right) = 0; \tag{IV.7}$$

$$\left(\vec{r}'' \cdot \vec{r}_v\right) = 0. \tag{IV.8}$$

Вычисляя скалярные произведения, получим, например:

$$\left(\vec{r}'' \cdot \vec{r}_u\right) =$$

$$\left(\vec{r}_{uu} \cdot \vec{r}_u\right)u'^{\,2} + 2\left(\vec{r}_{uv} \cdot \vec{r}_u\right)u'v' + \left(\vec{r}_{vv} \cdot \vec{r}_u\right)v'^{\,2} + \vec{r}_u{}^2u'' + \left(\vec{r}_u \cdot \vec{r}_v\right)v''$$

Таким образом получаем систему двух уравнений:

$$\frac{1}{2}\partial_u E u'^{\,2} + \partial_v E u'v' + \left(\vec{r}_{vv} \cdot \vec{r}_u\right)v'^{\,2} + Eu'' + Fv'' = 0; \tag{IV.9}$$

$$\frac{1}{2}\partial_v G v'^{\,2} + \partial_u G u'v' + \left(\vec{r}_{uu} \cdot \vec{r}_v\right)u'^{\,2} + Fu'' + Gv'' = 0. \tag{IV.10}$$

Далее произведем преобразования:

$$\left(\vec{r}_{vv} \cdot \vec{r}_u\right) = \partial_v\left(\vec{r}_v \cdot \vec{r}_u\right) - \left(\vec{r}_v \cdot \vec{r}_{uv}\right) \Longrightarrow$$

$$\left(\vec{r}_{vv} \cdot \vec{r}_u \right) = \partial_v F - \frac{1}{2}\partial_u G. \tag{IV.11}$$

Аналогично:

$$\left(\vec{r}_{uu} \cdot \vec{r}_v \right) = \partial_u F - \frac{1}{2}\partial_v E. \tag{IV.12}$$

Используя соотношения IV.11, IV.12 в уравнениях IV.9, IV.11, приведем последние к виду:

$$\frac{1}{2}\partial_u E u'^{\,2} + \partial_v E u'v' + \left(\partial_v F - \frac{1}{2}\partial_u G\right) v'^{\,2} + Eu'' + Fv'' = 0; \tag{IV.13}$$

$$\frac{1}{2}\partial_v G v'^{\,2} + \partial_u G u'v' + \left(\partial_u F - \frac{1}{2}\partial_v E\right) u'^{\,2} + Fu'' + Gv'' = 0. \tag{IV.14}$$

Введем теперь так называемые *символы Кристоффеля I-го рода*:

$$\Gamma_{ij,k} = \frac{1}{2}\left(\partial_i g_{jk} + +\partial_j g_{ik} - \partial_k g_{ij}\right). \tag{IV.15}$$

По определению символы Кристоффеля I-го рода симметричны по первым двум индексам:

$$\Gamma_{ij,k} = \Gamma_{ji,k}. \tag{IV.16}$$

Таким образом, вычисляя, найдем:

$$\Gamma_{11,1} = \frac{1}{2}\partial_u E;\ \Gamma_{11,2} = \partial_u F - \frac{1}{2}\partial_v E;\ \Gamma_{12,1} = \Gamma_{21,1} = \partial_v E;$$

$$\Gamma_{12,2} = \Gamma_{21,2} = \frac{1}{2}\partial_u G;\ \Gamma_{22,1} = \partial_v F - \frac{1}{2}\partial_v G. \tag{IV.17}$$

Сравнивая IV.17 с уравнениями IV.13, IV.14, приведем последние к виду:

$$g_{jk}\frac{d^2x^k}{ds^2} + \Gamma_{kl,j}\frac{dx^l}{ds}\frac{dx^k}{ds} = 0, \qquad (j,k,l = \overline{1,2}). \tag{IV.18}$$

Пусть g^{ij} - коэффициенты матрицы G^{-1}, обратной к матрице первой квадратичной формы:

$$g^{ij}g_{kj} = \delta ik. \tag{IV.19}$$

Можно показать, что величины g^{ij} образуют координаты симметричного контрвариантного тензора второй валентности.

Умножим уравнения IV.18 на g^{ij} и просуммируем результат по индексу j, в результате получим уравнения:

$$\frac{d^2x^i}{ds^2} + \Gamma^i_{kl}\frac{dx^k}{ds}\frac{dx^l}{ds} = 0, \qquad (i = \overline{1,2}), \tag{IV.20}$$

где введены так называемые *символы Кристоффеля II-го рода*:

$$\Gamma^i_{kl} = g^{ij}\Gamma_{kl,j}, \tag{IV.21}$$

также симметричные по двум нижним индексам.

Уравнения IV.21 называются *уравнениями геодезических линий*; они являются системой обыкновенных дифференциальных уравнений второго порядка относительно координат $x^i(s)$ точки геодезической линии.

Так как кривая Γ параметризована натуральным параметром $t = s$, то должны выполняться условия:

$$|\vec{r}\,'\,| = 1; \tag{IV.22}$$

$$\left(\vec{r}\,' \cdot \vec{r}\,''\right) = 0. \tag{IV.23}$$

Расписывая соотношение IV.22,получим так называемое *соотношение нормировки*:

$$Eu'^{\,2} + 2Fu'v' + Gv'^{\,2} = 1 \Longrightarrow g_{ik}\frac{dx^i}{ds}\frac{dx^k}{ds} = 1. \tag{IV.24}$$

Однако, вследствие уравнений геодезических IV.7, IV.8 соотношение IV.23, являющееся дифференциальным следствием соотношения нормировки IV.22, выполняется тождественно. Таким образом, можно утверждать, что интегралом уравнений геодезических IV.21 является:

$$g_{ij}\frac{dx^i}{ds}\frac{dx^j}{ds} = \text{Const}\,. \tag{IV.25}$$

Это означает, что соотношение нормировки IV.24 лишь уточняет значение этой постоянной, и, следовательно, одно из этих уравнений геодезических может быть заменено соотношением нормировки.

В заключении раздела заметим, что уравнения геодезических полностью определяются коэффициентами первой квадратичной формы и их первыми частными производными. Таким образом, исследование геодезических поверхности является задачей внутренней геометрии поверхности.

IV.1.3 Геодезическая линия как кратчайшая

Пусть γ — вещественная кривая поверхности Σ, заданная уравнениями $x^i = f^i(t)$, где x^i - внутренние координаты поверхности, t — вещественный параметр, и пусть A и B — две точки этой кривой, соответствующие значениям параметра t_0 и t_1. Пусть далее

$$ds^2 = g_{ik}dx^i dx^k \tag{IV.26}$$

— первая квадратичная форма поверхности. Тогда длина этой кривой, заключенная между точками A и B равна:

$$s = \int_{t_1}^{t_2} \sqrt{g_{ij}\frac{dx^i}{dt}\frac{dx^j}{dt}}dt. \tag{IV.27}$$

Уравнения

$$\overline{x}^i = x^i + \epsilon\delta x^i,$$

где ϵ — бесконечно малая величина, а δx^i — функции от x^i, такие, что

$$\delta x^i = 0 \quad \text{при}\, t = t_0, t_1 \tag{IV.28}$$

определяет новую кривую $\overline{C}$, близкую к γ и проходящую через те же точки A и B.
Рассмотрим интеграл

$$I = \int_{t_0}^{t_1} \varphi(x^1, \ldots, x^n, \dot{x}^1, \ldots, \dot{x}^n)dt, \tag{IV.29}$$

где $\dot{x}^i = \dfrac{dx^i}{dt}$, а φ — аналитические функции от $2n$ аргументов. Если $\overline{I}$ — соответствующий интеграл для кривой $\overline{C}$, то разлагая функцию φ в ряд Тейлора, получим:

$$\overline{I} - I = \epsilon \int_{t_0}^{t_1} \left[\frac{\partial\varphi}{\partial x^i}\delta x^i + \frac{\partial\varphi}{\partial\dot{x}^i}\dot{\delta}x^i\right] dt + \ldots,$$

где $\dot{\delta}x^i = \dfrac{\partial\delta x^i}{\partial x^j}\dot{x}^j$, а ненаписанные члены имеют второй и более высокий порядок относительно ϵ. Если мы запишем, что

$$\delta I = \epsilon \int_{t_0}^{t_1} \left[\frac{\partial\varphi}{\partial x^i}\delta x^i + \frac{\partial\varphi}{\partial\dot{x}^i}\dot{\delta}x^i\right] dt, \tag{IV.30}$$

то, интегрируя второе подинтегральное слагаемое по частями используя формулу IV.28, получим

$$\delta I = \epsilon \int_{t_0}^{t_1} \left[\frac{\partial \varphi}{\partial x^i} - \frac{d}{dt} \left(\frac{\partial \varphi}{\partial \dot{x}^i} \right) \right] \delta x^i dt. \tag{IV.31}$$

Об интеграле говорят, что он *стационарен*, а соответствующая ему кривая Γ называется *экстремалью*, если первая вариация δI равна нулю для любой системы функций δx^i, удовлетворяющих условиям IV.28. Из формулы IV.31 следует, что, необходимым и достаточным для этого условием будет равенство

$$\frac{d}{dt} \left(\frac{\partial \varphi}{\partial \dot{x}^i} \right) - \frac{\partial \varphi}{\partial x^i} = 0. \tag{IV.32}$$

Это условие известно под названием уравнений *Эйлера*. Применим этот общий результат к интегралу IV.27: В этом случае

$$\frac{\partial \varphi}{\partial \dot{x}^i} = \frac{g_{ij}\dot{x}^j}{\sqrt{g_{ij}\dot{x}^i\dot{x}^j}} = \frac{g_{ij}\dot{x}^j}{\dfrac{ds}{dt}}, \qquad \frac{\partial \varphi}{\partial x^i} = \frac{1}{2} \frac{\dfrac{\partial g_{jk}}{\partial x^i}\dot{x}^j\dot{x}^k}{\dfrac{ds}{dt}}.$$

Подставив эти значения в уравнение IV.32, получим

$$g_{ij}\ddot{x}^j + \frac{\partial g_{ij}}{\partial x^k}\dot{x}^j\dot{x}^k - \frac{1}{2}\frac{\partial g_{jk}}{\partial x^i}\dot{x}^j\dot{x}^k - g_{ij}\dot{x}^j \frac{\dfrac{d^2 s}{dt^2}}{\dfrac{ds}{dt}} = 0.$$

Применяя символы Кристоффеля, составленные для формы IV.26, можно этим уравнениям придать вид

$$g_{ij}\frac{d^2x^j}{dt^2} + \Gamma_{jk,i}\frac{dx^j}{dt}\frac{dx^k}{dt} - g_{ij}\frac{dx^j}{dt}\frac{\dfrac{d^2 s}{dt^2}}{\dfrac{ds}{dt}} = 0. \tag{IV.33}$$

Умножая на g^{il} и суммируя по i, найдем, что

$$\frac{d^2x^l}{dt^2} + \Gamma^l_{jk}\frac{dx^j}{dt}\frac{dx^k}{dt} - \frac{dx^l}{dt}\frac{\dfrac{d^2 s}{dt^2}}{\dfrac{ds}{dt}} = 0. \tag{IV.34}$$

Если вместо произвольного параметра t использовать длину дуги s, то уравнения IV.34 принимают вид

$$\frac{d^2x^i}{ds^2} + \Gamma^i_{jk}\frac{dx^j}{ds}\frac{dx^k}{ds} = 0. \tag{IV.35}$$

Таким образом, экстремалями интеграла IV.27, в котором параметр t представляет длину дуги s, являются интегральные кривые системы n обыкновенных дифференциальных уравнений IV.35.

Умножим уравнения IV.35 на

$$g_{il}\frac{dx^l}{ds}$$

и просуммируем результат. При этом надо учесть, что вследствие симметричности квадратичной формы

$$g_{il}\frac{dx^i}{ds}\frac{dx^l}{ds}$$

$$g_{il}\frac{dx^l}{ds}\frac{d^2x^i}{ds^2} \equiv \frac{1}{2}\frac{d}{ds}g_{il}\frac{dx^i}{ds} - \frac{dx^i}{ds}\frac{dx^l}{ds}\frac{dg_{il}}{ds}.$$

Таким образом, получим тождество:

$$g_{il}\frac{dx^l}{ds}\frac{d^2x^i}{ds^2} \equiv \frac{1}{2}\frac{d}{ds}g_{il}\frac{dx^i}{ds} - \partial_m g_{il}\frac{dx^i}{ds}\frac{dx^l}{ds}\frac{dx^m}{ds}. \tag{IV.36}$$

Учтем также, что

$$g_i l\Gamma^i_{jk} = \Gamma_{jk,l} = \frac{1}{2}(\partial_j g_{kl} + \partial_k g_{jl} - \partial_l g_{jk}).$$

Таким образом, получим из IV.35:

$$\frac{d}{ds}\left(g_{il}\frac{dx^i}{ds}\frac{dx^l}{ds}\right) = 0 \Longrightarrow$$

$$g_{ij}\frac{dx^i}{ds}\frac{dx^j}{ds} = \text{Const}\,. \tag{IV.37}$$

Это означает, что величина:

$$\chi = g_{ij}\frac{dx^i}{ds}\frac{dx^j}{ds}$$

является *интегралом* уравнений геодезических линий. Таким образом соотношение нормировки:

$$g_{ij}\frac{dx^i}{ds}\frac{dx^j}{ds} = 1,$$

возникающее при натуральной параметризации кривой и означающее факт нормировки вектора касательной, не противоречит уравнениям геодезической линии, а лишь уточняет значение постоянной в формуле IV.37.

В качестве примера рассмотрим геодезические линии евклидовой плоскости и кругового цилиндра. Первая квадратичная форма евклидовой плоскости есть:

$$ds^2 = dx^2 + dy^2, \tag{IV.38}$$

следовательно, матрица первой квадратичной формы единичная и постоянная. Вследствие этого символы Кристоффеля как первого, так и второго рода евклидовой плоскости *в декартовых координатах* равны нулю. Но тогда уравнения геодезических линий евклидовой плоскости принимают вид:

$$\frac{d^2x}{ds^2} = 0; \qquad \frac{d^2y}{ds^2} = 0. \tag{IV.39}$$

Как известно из математического анализа, необходимым и достаточным условием равенства нулю второй производной функции на отрезке является линейность этой функции, т.е:

$$x = \alpha s + x_0; \qquad y = \beta s + y_0, \tag{IV.40}$$

где α, β, x_0, y_0 - произвольные константы. Но уравнения IV.40 относительно декартовых координат являются параметрическими уравнениями прямой линии на плоскости.

Таким образом, справедлива теорема:

Теорема TIV.1. *Геодезическими линиями евклидовой плоскости являются прямые линии и только они.*

Рассмотрим теперь геодезические линии на круговом цилиндре радиуса a, метрика которого в цилиндрических координатах имеет вид:

$$ds^2 = a^2 d\varphi^2 + dz^2. \tag{IV.41}$$

Поскольку матрица первой квадратичной формы цилиндра постоянна, то и в этом случае символы Кристоффеля обращаются в нуль, а, следовательно, решениями уравнений геодезических будут линейные функции:

$$\varphi = \alpha s + \varphi_0; \qquad z = \beta s + z_0, \tag{IV.42}$$

Исключая отсюда натуральный параметр, получим:

$$z = \sigma\varphi + z_0. \tag{IV.43}$$

Возвращаясь к цилиндрическим координатам в пространстве, найдем параметрические уравнения геодезических цилиндра:

$$\begin{aligned} x &= a\cos\varphi\,; \\ y &= a\sin\varphi\,; \quad , \\ z &= \sigma\varphi + z_0 \end{aligned} \tag{IV.44}$$

т.е., получим уравнения винтовой цилиндрической линии.

Таким образом, справедлива теорема:

Теорема TIV.2. *Геодезическими линиями кругового цилиндра являются винтовые линии и только они.*

IV.2 Построение геодезической сети на псевдосфере средствами Maple

В качестве примера достаточно подробно рассмотрим процедуру построения геодезической сети на псевдосфере (см. [126]).

IV.2.1 Задание псевдосферы и вычисление производных радиуса-вектора

Зададим параметрические уравнения верхней части псевдосферы, для чего введем **функции** x(t, ϕ), y(t, ϕ), z(t, ϕ) двух параметров (t, ϕ), которые принимают следующие значения на верхней половине псевдосферы: $t = 0..\frac{\pi}{2}$, $\phi = 0..2\,\pi$:

```
> restart:with(linalg):with(tensor):
```

```
> x:=(t,phi)->sin(t)*cos(phi);
> y:=(t,phi)->sin(t)*sin(phi);
> z:=(t,phi)->-(cos(t)+ln(tan(t/2)));
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
```

$$x := (t,\, \phi) \to \sin(t)\cos(\phi)$$
$$y := (t,\, \phi) \to \sin(t)\sin(\phi)$$
$$z := (t,\, \phi) \to -\cos(t) - \ln(\tan(\frac{1}{2}\,t))$$

Зададим теперь радиус-вектор r произвольной точки псевдосферы в виде упорядоченного множества функций и вычислим от него частные производные с помощью команд **diff(r(t, ϕ),t)** и **diff(r(t, ϕ), ϕ)**:

```
> r:=(t,phi)->[x(t,phi),y(t,phi),z(t,phi)];
> r_t:=diff(r(t,phi),t);
> r_phi:=diff(r(t,phi),phi);
```

$$r := (t,\, \phi) \to [\mathrm{x}(t,\, \phi),\, \mathrm{y}(t,\, \phi),\, \mathrm{z}(t,\, \phi)]$$
$$r\_t := \left[\cos(t)\cos(\phi),\, \cos(t)\sin(\phi),\, \sin(t) - \frac{\frac{1}{2} + \frac{1}{2}\tan(\frac{t}{2})^2}{\tan(\frac{t}{2})}\right]$$
$$r\_phi := [-\sin(t)\sin(\phi),\, \sin(t)\cos(\phi),\, 0]$$
$$r := (t,\, \phi) \to [\mathrm{x}(t,\, \phi),\, \mathrm{y}(t,\, \phi),\, \mathrm{z}(t,\, \phi)]$$
$$r\_t := \left[\cos(t)\cos(\phi),\, \cos(t)\sin(\phi),\, \sin(t) - \frac{\frac{1}{2} + \frac{1}{2}\tan(\frac{1}{2}\,t)^2}{\tan(\frac{1}{2}\,t)}\right]$$
$$r\_phi := [-\sin(t)\sin(\phi),\, \sin(t)\cos(\phi),\, 0]$$

IV.2.2 Нахождение матрицы первой квадратичной формы псевдосферы

Сформируем теперь симметричную матрицу первой квадратичной формы и вычислим ее определитель.

```
> G:=array(1..2,1..2,symmetric,sparse);
```

$$G := \mathrm{array}(\mathit{symmetric},\, \mathit{sparse},\, 1..2,\, 1..2,\, [])$$

Зададим коэффициенты матрицы первой квадратичной формы псевдосферы $G_{i,k}$ с помощью скалярных произведений производных радиуса-вектора ($r_i\, r_k$), вычисляемых с помощью команды **innerprod(a,b)** библиотеки **linalg,** где a и b - векторы.

```
> G[1,1]:=simplify(innerprod(r_t,r_t));
> G[1,2]:=innerprod(r_t,r_phi);
> G[2,2]:=simplify(innerprod(r_phi,r_phi));
```

$$G_{1,1} := \frac{\cos(t)^2}{\sin(t)^2}$$

$$G_{1,2} := 0$$

$$G_{2,2} := 1 - \cos(t)^2$$

Вычислим определитель матрицы G:

```
> DetG:=simplify(det(G));
```

$$DetG := \cos(t)^2$$

и определим метрический тензор g_{ik} псевдосферы:

```
> g:=create([-1,-1],eval(G));
```

$$g := \mathrm{table}([compts = \begin{bmatrix} \frac{\cos(t)^2}{\sin(t)^2} & 0 \\ 0 & 1-\cos(t)^2 \end{bmatrix}, index\_char = [-1,\ -1]])$$

IV.2.3 Вычисление символов Кристоффеля псевдосферы

Определим внутренние координаты псевдосферы как упорядоченную последовательность

```
> coord:=[t,phi]:
```

Теперь с помощью команды **d1metric{G,coord)** библиотеки **tensor** вычислим первые частные производные, dG, метрического тензора, а также вычислим координаты контрвариантного метрического тензора g_1 (валентности [+1,+1]), обратного к метрическому тензору g с помощью команды **invert(G,'detG')** библиотеки **tensor**:

```
> dg:=d1metric(g,coord);
> g_1:=invert(g,'detG');
```

$$
\begin{aligned}
&dg := \mathrm{table}([compts = \mathrm{array}(cf1,\ 1..2,\ 1..2,\ 1..2,\ [\\
&(1,\ 1,\ 1) = -\frac{2\cos(t)}{\sin(t)^3}\\
&(1,\ 1,\ 2) = 0\\
&(1,\ 2,\ 1) = 0\\
&(1,\ 2,\ 2) = 0\\
&(2,\ 1,\ 1) = 0\\
&(2,\ 1,\ 2) = 0\\
&(2,\ 2,\ 1) = 2\cos(t)\sin(t)\\
&(2,\ 2,\ 2) = 0\\
&]),\\
&index\_char = [-1,\ -1,\ -1]\\
&])
\end{aligned}
$$

$$
g\_1 := \mathrm{table}([compts = \begin{bmatrix} \frac{\sin(t)^2}{\cos(t)^2} & 0 \\ 0 & \frac{1}{\sin(t)^2} \end{bmatrix},\ index\_char = [1,\ 1]])
$$

С помощью найденных значений первых частных производных вычислим теперь символы Кристоффеля I-го рода, Cf1, применяя команду **Christoffel1(dG)** библиотеки **tensor**:

```
> CR1:=tensor[Christoffel1](dg):
```

На основе полученных значений для символов Кристоффеля I-го рода найдем символы Кристоффеля II-го рода, Cf2, с помощью команды **Christoffel2(g_1,Cf1)** библиотеки **tensor**:

```
> CR2:=tensor[Christoffel2](g_1,CR1);
```

$$
\begin{aligned}
&CR2 := \mathrm{table}([compts = \mathrm{array}(cf2,\ 1..2,\ 1..2,\ 1..2,\ [\\
&(1,\ 1,\ 1) = -\frac{1}{\sin(t)\cos(t)}\\
&(1,\ 1,\ 2) = 0\\
&(1,\ 2,\ 1) = 0\\
&(1,\ 2,\ 2) = -\frac{\sin(t)^3}{\cos(t)}\\
&(2,\ 1,\ 1) = 0\\
&(2,\ 1,\ 2) = \frac{\cos(t)}{\sin(t)}\\
&(2,\ 2,\ 1) = \frac{\cos(t)}{\sin(t)}\\
&(2,\ 2,\ 2) = 0\\
&]),\\
&index\_char = [1,\ -1,\ -1]\\
&])
\end{aligned}
$$

Наконец, с помощью команды **geodesic_eqn(coord,c,Cf2)** библиотеки получим уравнения геодезических:

```
> Geo:= tensor[geodesic_eqns](coord, s, CR2 );
```

$$
Geo := \left\{ (\tfrac{d^2}{ds^2}\,\mathrm{t}(s)) - \frac{(\frac{d}{ds}\,\mathrm{t}(s))^2}{\sin(t)\cos(t)} - \frac{\sin(t)^3\,(\frac{d}{ds}\,\phi(s))^2}{\cos(t)} = 0,\ (\tfrac{d^2}{ds^2}\,\phi(s)) + \frac{2\cos(t)\,(\frac{d}{ds}\,\mathrm{t}(s))\,(\frac{d}{ds}\,\phi(s))}{\sin(t)} = 0 \right\}
$$

IV.2.4 Приведение уравнений геодезических к нормальной системе ОД У

Обозначим первые производные по натуральному параметру от внутренних координат псевдосферы за новые переменные, $\mathrm{T}(s)$ и $\Phi(s)$:

```
> EQT:=diff(t(s),s)=T(s);
> EQP:=diff(phi(s),s)=Phi(s);
```

$$
EQT := \tfrac{d}{ds}\,\mathrm{t}(s) = \mathrm{T}(s)
$$

$$
EQP := \tfrac{d}{ds}\,\phi(s) = \Phi(s)
$$

Подставим эти величины в уравнения геодезических:

```
> GEOT:=subs({EQT,EQP,t=t(s)},Geo[1]);
```

```
> GEOP:=subs({EQT,EQP,t=t(s)},Geo[2]);
```

$$GEOT := (\tfrac{d}{ds}\,\mathrm{T}(s)) - \frac{\mathrm{T}(s)^2}{\sin(\mathrm{t}(s))\cos(\mathrm{t}(s))} - \frac{\sin(\mathrm{t}(s))^3\,\Phi(s)^2}{\cos(\mathrm{t}(s))} = 0$$

$$GEOP := (\tfrac{d}{ds}\,\Phi(s)) + \frac{2\cos(\mathrm{t}(s))\,\mathrm{T}(s)\,\Phi(s)}{\sin(\mathrm{t}(s))} = 0$$

При этом мы получили нормальную систему дифференциальных уравнений, т.е., систему дифференциальных уравнений I-го порядка (EQT,EQP,GEOT,GEOP), разрешенных относительно производных: $\frac{d}{ds}\,\mathrm{t}(s)$, $\frac{d}{ds}\,\phi(s)$, $\frac{d}{ds}\,\mathrm{T}(s)$, $\frac{d}{ds}\,\Phi(s)$.

Запишем теперь эти уравнения в виде системы уравнений:

```
> GEO:={EQT,EQP,GEOT,GEOP};
```

$$GEO := \{\tfrac{d}{ds}\,\phi(s) = \Phi(s),\ (\tfrac{d}{ds}\,\mathrm{T}(s)) - \frac{\mathrm{T}(s)^2}{\sin(\mathrm{t}(s))\cos(\mathrm{t}(s))} - \frac{\sin(\mathrm{t}(s))^3\,\Phi(s)^2}{\cos(\mathrm{t}(s))} = 0,$$

$$(\tfrac{d}{ds}\,\Phi(s)) + \frac{2\cos(\mathrm{t}(s))\,\mathrm{T}(s)\,\Phi(s)}{\sin(\mathrm{t}(s))} = 0,\ \tfrac{d}{ds}\,\mathrm{t}(s) = \mathrm{T}(s)\}$$

IV.2.5 Ввод группы начальных условий

а). Линии "Φ"

- здесь изменяются только значения t(0), при этом вектор касательной направлен вдоль параллелей:

```
> In[P0]:={t(0)=0.1,T(0)=0,phi(0)=0,Phi(0)=1};
> In[P1]:={t(0)=0.2,T(0)=0,phi(0)=0,Phi(0)=1};
> In[P2]:={t(0)=0.3,T(0)=0,phi(0)=0,Phi(0)=1};
> In[P3]:={t(0)=0.4,T(0)=0,phi(0)=0,Phi(0)=1};
```

$$In_{P0} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = 1,\ \mathrm{t}(0) = 0.1\}$$

$$In_{P1} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = 1,\ \mathrm{t}(0) = 0.2\}$$

$$In_{P2} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = 1,\ \mathrm{t}(0) = 0.3\}$$

$$In_{P3} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = 1,\ \mathrm{t}(0) = 0.4\}$$

б). Линии "Φ_1 здесь также изменяются только значения t(0), но при этом вектор касательной направлен в противоположную сторону вдоль параллелей:

```
> In[P_0]:={t(0)=0.1,T(0)=0,phi(0)=0,Phi(0)=-1};
> In[P_1]:={t(0)=0.2,T(0)=0,phi(0)=0,Phi(0)=-1};
> In[P_2]:={t(0)=0.3,T(0)=0,phi(0)=0,Phi(0)=-1};
> In[P_3]:={t(0)=0.4,T(0)=0,phi(0)=0,Phi(0)=-1};
```

$$In_{P\_0} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \mathrm{t}(0) = 0.1,\ \Phi(0) = -1\}$$
$$In_{P\_1} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = -1,\ \mathrm{t}(0) = 0.2\}$$
$$In_{P\_2} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = -1,\ \mathrm{t}(0) = 0.3\}$$
$$In_{P\_3} := \{\phi(0) = 0,\ \mathrm{T}(0) = 0,\ \Phi(0) = -1,\ \mathrm{t}(0) = 0.4\}$$

в). Линии "T"

- здесь изменяются лишь значения $\phi(0)$, при этом вектор касательной направлен вдоль меридианов:

```
> In[T0]:={t(0)=Pi/12,T(0)=1,phi(0)=0,Phi(0)=0};
> In[T1]:={t(0)=Pi/12,T(0)=1,phi(0)=Pi/3,Phi(0)=0};
> In[T2]:={t(0)=Pi/12,T(0)=1,phi(0)=2*Pi/3,Phi(0)=0};
> In[T3]:={t(0)=Pi/12,T(0)=1,phi(0)=Pi,Phi(0)=0};
> In[T4]:={t(0)=Pi/12,T(0)=1,phi(0)=4*Pi/3,Phi(0)=0};
> In[T5]:={t(0)=Pi/12,T(0)=1,phi(0)=5*Pi/3,Phi(0)=0};
```

$$In_{T0} := \{\phi(0) = 0,\ \mathrm{T}(0) = 1,\ \Phi(0) = 0,\ \mathrm{t}(0) = \frac{\pi}{12}\}$$
$$In_{T1} := \{\mathrm{T}(0) = 1,\ \Phi(0) = 0,\ \mathrm{t}(0) = \frac{\pi}{12},\ \phi(0) = \frac{\pi}{3}\}$$
$$In_{T2} := \{\mathrm{T}(0) = 1,\ \Phi(0) = 0,\ \mathrm{t}(0) = \frac{\pi}{12},\ \phi(0) = \frac{2\,\pi}{3}\}$$
$$In_{T3} := \{\mathrm{T}(0) = 1,\ \Phi(0) = 0,\ \mathrm{t}(0) = \frac{\pi}{12},\ \phi(0) = \pi\}$$
$$In_{T4} := \{\mathrm{T}(0) = 1,\ \Phi(0) = 0,\ \mathrm{t}(0) = \frac{\pi}{12},\ \phi(0) = \frac{4\,\pi}{3}\}$$
$$In_{T5} := \{\mathrm{T}(0) = 1,\ \Phi(0) = 0,\ \mathrm{t}(0) = \frac{\pi}{12},\ \phi(0) = \frac{5\,\pi}{3}\}$$

IV.2.6 Изображение верхней половины псевдосферы и линий начальных условий

Изобразим параллели и меридианы, соответствующие заданию нач альных условий, с помощью команды **spacecurve** библиотеки **plots**. Параллели, соответствующие большим значениям параметра t находятся ниже, а параллели с меньшим значением параметра t - выше:

```
> Ps[u]:=plot3d(r(t,phi),t=0..Pi/2,phi=0..2*Pi,style=PATCH,color=white,
> thickness=0): cir_1:=plots[spacecurve](r(Pi/12,phi),phi=0..2*Pi,
> color=black,thickness=2):
> cir_2:=plots[spacecurve](r(Pi/8,phi),phi=0..2*Pi,
> color=black,thickness=2):
> cir_3:=plots[spacecurve](r(Pi/4,phi),phi=0..2*Pi,
> color=black,thickness=2):
> cir_4:=plots[spacecurve](r(Pi/3,phi),phi=0..2*Pi,
> color=black,thickness=2):
> mer_1:=plots[spacecurve](r(t,0),t=0..Pi/2,color=black,thickness=2):
> mer_2:=plots[spacecurve](r(t,Pi/3),t=0..Pi/2,color=black,thickness=2):
> mer_3:=plots[spacecurve](r(t,2*Pi/3),t=0..Pi/2,color=black,thickness=2
> ):
> mer_4:=plots[spacecurve](r(t,Pi),t=0..Pi/2,color=black,thickness=2):
> mer_5:=plots[spacecurve](r(t,4*Pi/3),t=0..Pi/2,color=black,thickness=2
> ):
> mer_6:=plots[spacecurve](r(t,5*Pi/3),t=0..Pi/2,color=black,thickness=2
> ):

> plots[display](cir_1,cir_2,cir_3,cir_4,mer_1,mer_2,mer_3,mer_4,mer_5,
> mer_6,axes=FRAME, labels=[x,y,z],labelfont=
> [TIMES,BOLD,14],thickness=0,color=white,titlefont=
> [TIMES,BOLD,12])
```

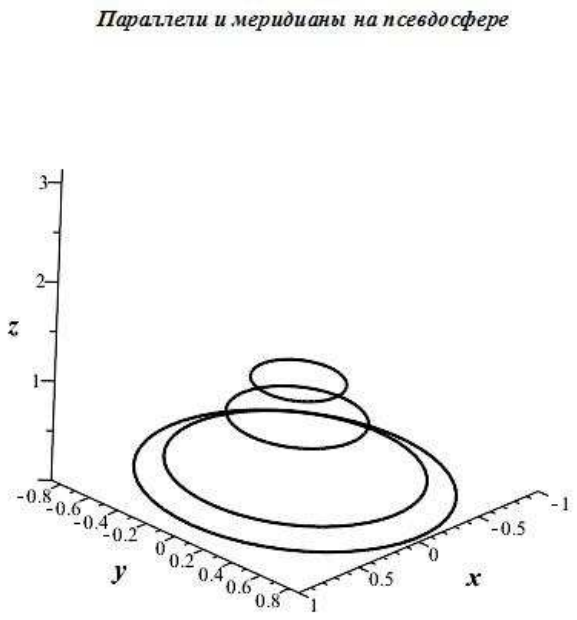


Рис.IV.33 *Параллели и меридианы на псевдосфере.*

IV.2.7 Процедуры численного интегрирования нормальной системы ОДУ

Здесь изложение идет по специальной методике профессора Игнатьева Ю.Г. решения системы ОДУ численными методами в Maple. Численные решения нормальной системы дифференциальных уравнений получаются с помощью команды
`dsolve(Система union Начальные Условия, {Переменные}, type=numeric, method=` **classical,output=listprocedure**).
При этом создается процедура численного решения нормальной системы дифференциальных уравнений классическим методом, комбинирующим известные основные методы численного решения нормальных систем дифференциальных уравнений с последующим выводом результатов интегрирования в виде списков: **а. Решения для линий** “Ф”

```
Sol0[Phi]:=
dsolve(GEO union In[P0],{t(s),phi(s),T(s),Phi(s)},
type=numeric,method=classical,output=listprocedure);
Sol1[Phi]:= dsolve(GEO union In[P1],
{t(s),phi(s),T(s),Phi(s)},
type=numeric,method=classical,output=listprocedure);
Sol2[Phi]:= dsolve(GEO union In[P2],
{t(s),phi(s),T(s),Phi(s)},
type=numeric,method=classical,output=listprocedure);
Sol3[Phi]:= dsolve(GEO union In[P3], {t(s),phi(s),T(s),Phi(s)},
```

б). Решения для линий Ф_1"

```
Sol_0[Phi]:= dsolve(GEO union In[P_0],{t(s),phi(s),T(s),Phi(s)},
```

в). Решения для линий «Т»

```
>Sol0[T]:= dsolve(GEO union In[T0],{t(s),phi(s),T(s),Phi(s)},
type=numeric,method=classical,output=listprocedure);
type=numeric,method=classical,output=listprocedure);
type=numeric,method=classical,output=listprocedure);
type=numeric,method=classical,output=listprocedure);
Sol4[T]:= dsolve(GEO union In[T4], {t(s),phi(s),T(s),Phi(s)},
type=numeric,method=classical,output=listprocedure);
type=numeric,method=classical,output=listprocedure);
```

Запись решений и проверка решений

Запись решений осуществляется с помощью подстановки решения в соответствующую величину:

```
t_P0:=subs(Sol0[Phi],t(s)):
phi_P0:=subs(Sol0[Phi],phi(s)):
t_P1:=subs(Sol1[Phi],t(s)):
phi_P1:=subs(Sol1[Phi],phi(s)):
t_P2:=subs(Sol2[Phi],t(s)):
phi_P2:=subs(Sol2[Phi],phi(s)):
t_P3:=subs(Sol3[Phi],t(s)):
phi_P3:=subs(Sol3[Phi],phi(s)):
```

```
> t_P_0:=subs(Sol_0[Phi],t(s)): phi_P_0:=subs(Sol_0[Phi],phi(s)):
> t_P_1:=subs(Sol_1[Phi],t(s)):
> phi_P_1:=subs(Sol_1[Phi],phi(s)):
> t_P_2:=subs(Sol_2[Phi],t(s)): phi_P_2:=subs(Sol_2[Phi],phi(s)):
> t_P_3:=subs(Sol_3[Phi],t(s)): phi_P_3:=subs(Sol_3[Phi],phi(s)):
> t_P_3(0.5);t_P_2(1);phi_P_2(1);phi_P_2(2);
```

$$0.407972228887300558$$
$$0.313565954681080339$$
$$-0.972274405931570329$$
$$-1.79692851570137124$$

```
> t_T0:=subs(Sol0[T],t(s)):phi_T0:=subs(Sol0[T],phi(s)):
> t_T1:=subs(Sol1[T],t(s)):phi_T1:=subs(Sol1[T],phi(s)):
> t_T2:=subs(Sol2[T],t(s)):phi_T2:=subs(Sol2[T],phi(s)):
> t_T3:=subs(Sol3[T],t(s)):phi_T3:=subs(Sol3[T],phi(s)):
> t_T4:=subs(Sol4[T],t(s)):phi_T4:=subs(Sol4[T],phi(s)):
> t_T5:=subs(Sol5[T],t(s)):phi_T5:=subs(Sol5[T],phi(s)):
> t_T0(0.1);t_T0(0.2);t_T0(0.3);t_T0(0.35);phi_T0(0.1);
> t_T3(0.1);t_T3(0.2);t_T3(0.3);t_T5(0.35);phi_T3(0.1);
```

$$0.383780149720260311$$
$$0.571777403808498619$$
$$0.893618550265845024$$
$$1.19269837961553014$$
$$0.$$
$$0.383780149720260311$$
$$0.571777403808498619$$

0.8936185502658450 24
1.19269837961553014
3.14159265358980022

Создание последовательностей полученных решений

Последовательности решений создаем с помощью команды **seq([],)**:

```
> r_T0:=[seq([x(t_T0(i/40),phi_T0(i/40)),y(t_T0(i/40),phi_T0(i/40)),z(t
> _T0(i/40),phi_T0(i/40))], i=0..15)]:
> r_T1:=[seq([x(t_T1(i/40),phi_T1(i/40)),y(t_T1(i/40),phi_T1(i/40)),z(t_
> T1(i/40),phi_T1(i/40))], i=0..15)]:
> r_T2:=[seq([x(t_T2(i/40),phi_T2(i/40)),y(t_T2(i/40),phi_T2(i/40)),z(t_
> T2(i/40),phi_T2(i/40))], i=0..15)]:
> r_T3:=[seq([x(t_T3(i/40),phi_T3(i/40)),y(t_T3(i/40),phi_T3(i/40)),z(t_
> T3(i/40),phi_T3(i/40))], i=0..15)]:
> r_T4:=[seq([x(t_T4(i/40),phi_T4(i/40)),y(t_T4(i/40),phi_T4(i/40)),z(t_
> T4(i/40),phi_T4(i/40))], i=0..15)]:
> r_T5:=[seq([x(t_T5(i/40),phi_T5(i/40)),y(t_T5(i/40),phi_T5(i/40)),z(t_
> T5(i/40),phi_T5(i/40))], i=0..15)]:
```

Создание графиков Т-геодезических

Для создания графиков Т-геодезических задаем их с помощью команды **spacecurve** а затем совмещаем их с помощью команды **display:**

```
> T0:=plots[spacecurve](r_T0,color=red,thickness=3):
> T1:=plots[spacecurve](r_T1,color=red,thickness=3):
> T2:=plots[spacecurve](r_T2,color=red,thickness=3):
> T3:=plots[spacecurve](r_T3,color=red,thickness=3):
> T4:=plots[spacecurve](r_T4,color=red,thickness=3):
> T5:=plots[spacecurve](r_T5,color=red,thickness=3):
```

```
> plots[display](T0,T1,T2,T3,T4,T5,Ps[u],axes=FRAME,labels=[x,y,z],labe
> lfont= [TIMES,BOLD,14],titlefont= [TIMES,BOLD,14],orientation=[-120,45]);
```

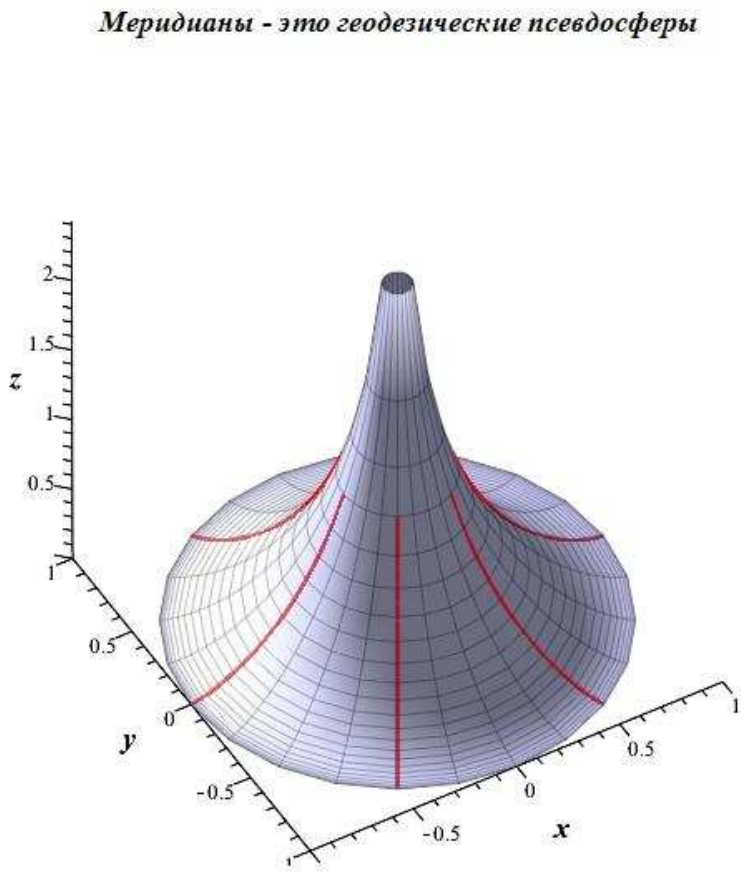


Рис.IV.34 *Меридианы - это геодезические псевдосферы.*

Таким образом, в результате численного интегрирования убеждаемся в том, что меридианы псевдосферы являются ее геодезическими. Это является общим результатом для поверхностей вращения (А.П.Норден, «Дифференциальная геометрия»).

Создание (Φ , $\Phi_{\_1}$)- сети

Создадим сеть с помощью геодезических, направленных в начальных точках параллельно параллелям псевдосферы, соответственно вдоль них и противоположно им:

```
> r_P0:=[seq([x(t_P0(i/20),phi_P0(i/20)),y(t_P0(i/20),phi_P0(i/20)),z(t
> _P0(i/20),phi_P0(i/20))], i=0..600)]:
> r_P1:=[seq([x(t_P1(i/20),phi_P1(i/20)),y(t_P1(i/20),phi_P1(i/20)),z(t_
> P1(i/20),phi_P1(i/20))], i=0..220)]:
> r_P2:=[seq([x(t_P2(i/20),phi_P2(i/20)),y(t_P2(i/20),phi_P2(i/20)),z(t_
> P2(i/20),phi_P2(i/20))], i=0..120)]:
> r_P3:=[seq([x(t_P3(i/20),phi_P3(i/20)),y(t_P3(i/20),phi_P3(i/20)),z(t_
> P3(i/20),phi_P3(i/20))], i=0..80)]:
```

```
> r_P_0:=[seq([x(t_P_0(i/20),phi_P_0(i/20)),y(t_P_0(i/20),phi_P_0(i/20)
> ), z(t_P_0(i/20),phi_P_0(i/20))], i=0..600)]:
> r_P_1:=[seq([x(t_P_1(i/20),phi_P_1(i/20)),y(t_P_1(i/20),phi_P_1(i/20))
> , z(t_P_1(i/20),phi_P_1(i/20))], i=0..220)]:
> r_P_2:=[seq([x(t_P_2(i/20),phi_P_2(i/20)),y(t_P_2(i/20),phi_P_2(i/20))
> , z(t_P_2(i/20),phi_P_2(i/20))], i=0..120)]:
> r_P_3:=[seq([x(t_P_3(i/20),phi_P_3(i/20)),y(t_P_3(i/20),phi_P_3(i/20))
> , z(t_P_3(i/20),phi_P_3(i/20))], i=0..80)]:

> P0:=plots[spacecurve](r_P0,color=black,thickness=3):
> P1:=plots[spacecurve](r_P1,color=black,thickness=3):
> P2:=plots[spacecurve](r_P2,color=black,thickness=3):
> P3:=plots[spacecurve](r_P3,color=black,thickness=3):

> P_0:=plots[spacecurve](r_P_0,color=black,thickness=3):
> P_1:=plots[spacecurve](r_P_1,color=black,thickness=3):
> P_2:=plots[spacecurve](r_P_2,color=black,thickness=3):
> P_3:=plots[spacecurve](r_P_3,color=black,thickness=3):

> plots[display](P0,P1,P2,P3,P_0,P_1,P_2,P_3,Ps[u],
> axes=FRAME,labels=[x,y,z],labelfont= [TIMES,BOLD,14],titlefont=
> [TIMES,BOLD,14],
> title=‘Геодезическая сеть на псевдосфере‘,orientation=[20,60]);
```

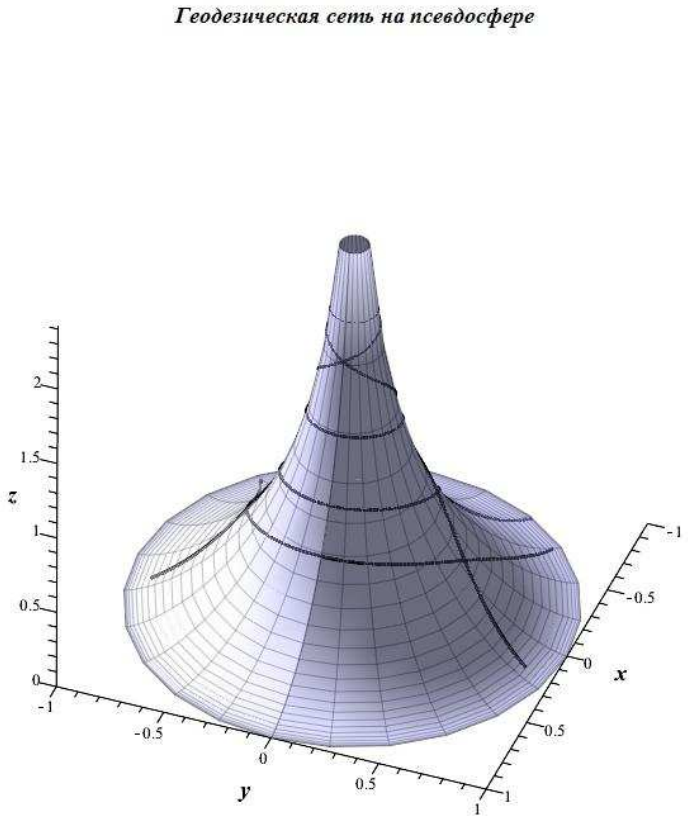


Рис.IV.35 *Геодезическая сеть на псевдосфере.*

IV.3 Пакет программ `Geodesic_lines`

Указанные программные процедуры можно значительно усовершенствовать и практически полностью автоматизировать процесс построения геодезической линии, проходящей через произвольную точку произвольной гладкой поверхности в произвольном направлении [110][2].

Единственная 9-параметрическая процедура этой библиотеки `Geodesic(Coord,S,S1,P,NewCoord,Inits,N,m,gr)` при соответствующем выборе параметров может создавать самые разнообразные 3d-модели геодезических, включая динамические модели. Здесь `Coord` - список внутренних координат поверхности и период изменения каждой из этих координат; S - канонический параметр кривой; S1 - интервал изменения параметра S в формате [a,b]; P - метрика; `NewCoord` - список новых координат, `Inits` - координаты начальной точки M0 геодезической и направляющего вектора V0 геодезической в этой точке, N - число кадров анимации, m - параметр, принимающий значения: 1 - (в этом случае геодезическая оканчивается на заданной границе поверхности, определяемой Coord), 0 - геодезическая выходит за пределы поверхности) gr - параметр, принимающий значения: graphic - для вывода графика, animate - для вывода анимации.

Продемонстрируем возможности пакета.

Пример 1. Геодезическая на торе:

Векторные параметрические уравнения тора имеют вид:

$$\vec{r} = [(2 + \cos(v)) * \cos(u), (2 + \cos(v)) * \sin(u), \sin(v)].$$

```
g1:=(2+cos(v))*cos(u),(2+cos(v))*sin(u),sin(v):
A:=Geodesic_lines[Geodesic]([[u,v],[0,2*Pi],
[0,2*Pi]],s,50,[g1],[U,V],[[2,2],[1,0]],150,1,graphic):
A[4];
```

[2]При использовании авторских программных процедур ссылка на них обязательна.

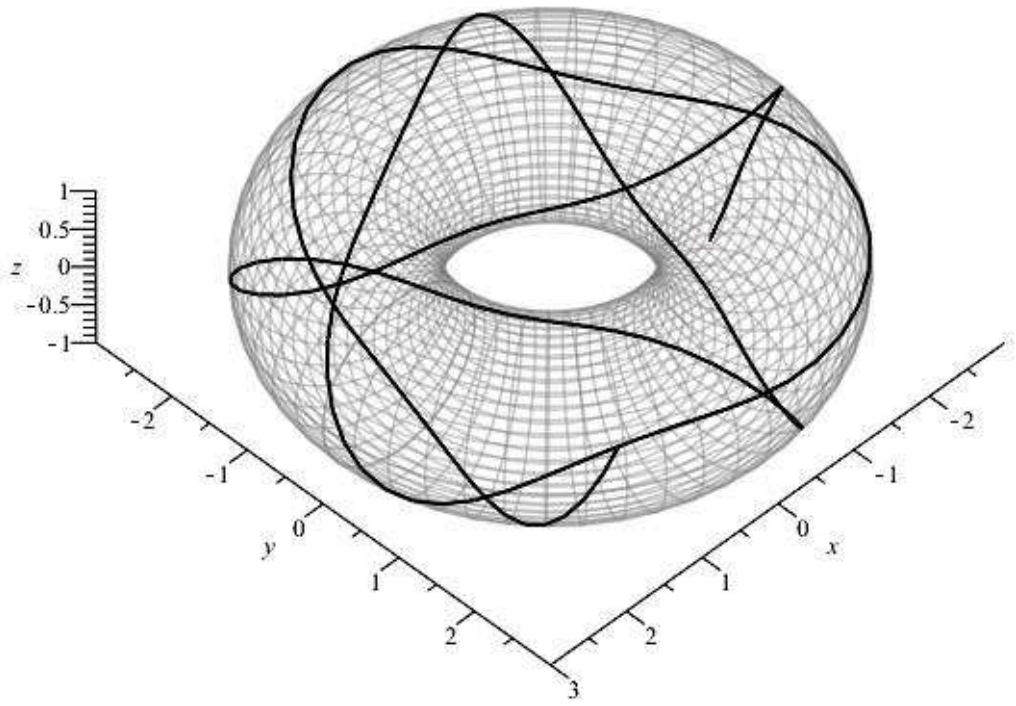


Рис.IV.36 *Применение процедуры* `Geodesic_lines[Geodesic]( [[u,v], [0,2*Pi], [0,2*Pi]], s, 50, [g1], [U,V],[[2,2],[1,0]],150,1,graphic)` *в графическом формате: геодезическая на торе.*

Пример 2. Геодезическая на однополостном гиперболоиде вращения:

Векторные параметрические уравнения тора имеют вид:

$$\vec{r} = [2 * \cosh(u) * \cos(v), 2 * \cosh(u) * \sin(v), 2 * \sinh(u)].$$

```
> g2:=2*cosh(u)*cos(v),2*cosh(u)*sin(v),2*sinh(u):
B:=Geodesic_lines[Geodesic]([[u,v],[-1.1,1.1],[0,2*Pi]],
```

```
s,10,[g2],[U,V],[[0,1/2],[Pi/32,1]],100,0,graphic):
> B[4];
```

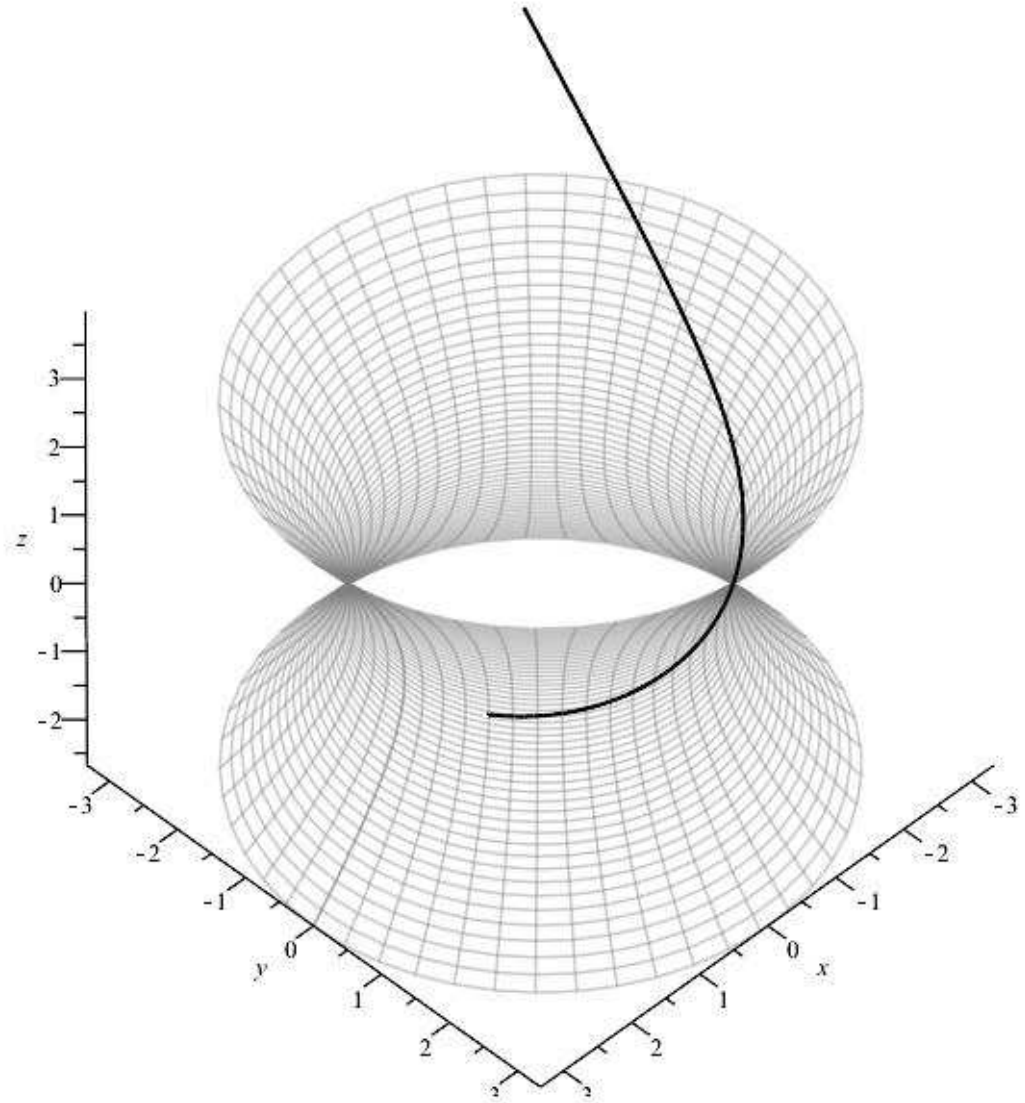


Рис.IV.37 *Применение процедуры* `Geodesic_lines[Geodesic]( [[u,v], [-1.1,1.1], [0,2*Pi]], s,10, [g2],[U,V], [[0,1/2], [Pi/32,1]] ,100,0, graphic)` *в графическом формате: геодезическая на гиперболоиде вращения. В данном случае параметр m=0 – геодезическая вышла за пределы заданной поверхности.*

Пример 3. Геодезическая на сфере:

Векторные параметрические уравнения сферы радиуса 2 имеют вид:

$$\vec{r} = [2 * \cos(u) * \cos(v), 2 * \cos(u) * \sin(v), 2 * \sin(u)].$$

```
> g3:=2*cos(u)*cos(v),2*cos(u)*sin(v),2*sin(u):
C:=Geodesic_lines[Geodesic]([[u,v],[0,2*Pi],[0,2*Pi]],s,12.6,[g3],[U,V],
> C[4];
```

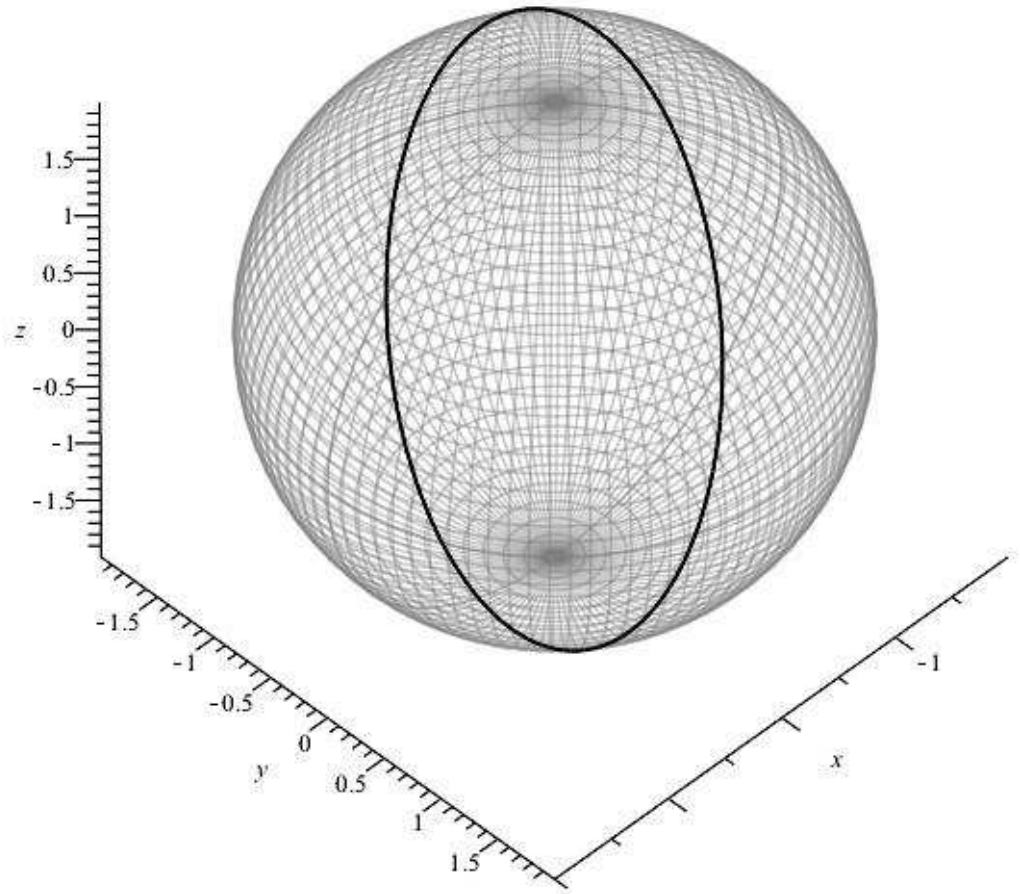


Рис.IV.38 *Применение процедуры* `Geodesic_lines[Geodesic]( [[u,v],[0,2*Pi], [0,2*Pi]], s,12.6, [g3], [U,V],[[2,2], [0,1]], 150,1,graphic)` *в графическом формате:геодезическая на сфере – большая окружность.*

Заметим, что мы всюду использовали только 4-ю опцию программной процедуры `Geodesic_lines` – именно она ответственна за вывод графической модели. Применение других опций позволяет, например, выписать в явном виде уравнения геодезических линий.

IV.4 Математическая модель геометрической оптики

IV.4.1 Функция эйконала и геометро-оптический предел

Электромагнитные волны в среде описываются *самосогласованной* системой уравнений, состоящей из уравнений Максвелла и уравнений движения материальной среды в поле электромагнитного излучения. По этим, последним, уравнениям определяются вектор плотности тока и плотности зарядов, индуцированных в материальной среде электромагнитной волной. Найденные из последних уравнений токи как функции электромагнитного поля подставляются затем в уравнения Максвелла, которые решаются тем или иным способом. В этом и состоит самосогласованный подход в теории электродинамики материальных сред. В общем случае получаются нелинейные интегро - дифференциальные уравнения, с трудом поддающиеся анализу. Мы ограничимся здесь *линейной электродинамикой среды*, когда электромагнитные волны настолько слабы, что не изменяют существенно характеристик среды (температуры, структуры и т.п.). Но даже и в этом случае самосогласованные уравнения электродинамики среды, становясь линейными, сохраняют интегро - дифференциальный характер и требуют специального исследования. Значительное упрощение их достигается лишь в пределе *геометрической оптики*, когда характерные размеры исследуемой среды, в том числе и характерный масштаб неоднородности, значительно превышают длину волны исследуемого света.

Опишем в самых общих чертах идеи геометрической оптики. Уравнения Максвелла среде имеют вид[3]:

$$(\vec{\nabla}\vec{E}) = 4\pi\varrho; \quad [\vec{\nabla}\vec{E}] + \frac{1}{c}\frac{\partial\vec{B}}{\partial t} = 0;$$

$$[\vec{\nabla}\vec{B}] - \frac{1}{c}\frac{\partial\vec{E}}{\partial t} = \frac{4\pi}{c}\vec{j}; \quad (\vec{\nabla}\vec{B}) = 0. \tag{IV.45}$$

где c - скорость света в вакууме, ϱ и $\vec{j}$ - плотности зарядов и токов, индуцированные в материальной среде электромагнитной волной; $\vec{E}$ и $\vec{E}$ - напряженность электрического поля и магнитная индукция. Следствием уравнений Maksvella является закон сохранения заряда, который для соответствующих плотностей имеет локальный вид *уравнения непрерывности*:

[3]Представленный в этом разделе материал частично можно найти в книге [31], частично в книге [32]. Геометрический аспект этого вопроса освещен в книге [33]. Вообще же говоря, основная часть излагаемого здесь материала имеет оригинальный характер.

$$\left(\vec{\nabla}\,\vec{j}\right) + \frac{\partial \varrho}{\partial t} = 0. \tag{IV.46}$$

В уравнениях (IV.45), (IV.46) $\vec{\nabla}$ - линейный дифференциальный оператор, который в декартовых координатах евклидова пространства имеет вид:

$$\vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \right) \tag{IV.47}$$

в криволинейных же координатах этот оператор совпадает с оператором ковариантного дифференцирования. Заметим, что в криволинейных координатах векторное произведение понимается следующим образом:

$$\left[\vec{a}\,\vec{b}\right]_i = \eta_{ijk} a^j b^k, \tag{IV.48}$$

где η_{ijk} - *ковариантно постоянный дискриминантный тензор*:

$$\eta_{ijk} = \frac{1}{\sqrt{g}} \varepsilon_{ijk}, \tag{IV.49}$$

а ε_{ijk} - единичный абсолютно антисимметричный *тензор перестановок*.

Введем новую векторную величину:

$$\vec{D}(\vec{r}, t) = \vec{E}(\vec{r}, t) + 4\pi \int\limits_{-\infty}^{t} dt' \vec{j}(\vec{r}, t'), \tag{IV.50}$$

называемую *вектором электрической индукции*.

С введением этой величины уравнения (IV.45) с учетом уравнения непрерывности (IV.46) принимают более симметричный вид:

$$\left(\vec{\nabla}\,\vec{D}\right) = 0; \quad \left[\vec{\nabla}\,\vec{E}\right] + \frac{1}{c}\frac{\partial \vec{B}}{\partial t} = 0;$$

$$\left[\vec{\nabla}\,\vec{B}\right] - \frac{1}{c}\frac{\partial \vec{D}}{\partial t} = 0; \quad \left(\vec{\nabla}\,\vec{B}\right) = 0. \tag{IV.51}$$

Если мы решим относительно вектора $\vec{E}$ некоторые уравнения движения среды в электромагнитном поле, то сможем установить некоторую *функциональную* связь между вектором плотности тока, $\vec{j}$ и напряженностью электрического поля, $\vec{E}$. Тогда мы сможем, в принципе, представить соотношение (IV.57) в виде:

$$\vec{D} = \vec{E} + \int\limits_{-\infty}^{t} \vec{\Phi}(\vec{E})dt'. \tag{IV.52}$$

Функциональное соотношение (IV.52) называются *материальным уравнением поля*. Отметим, что нахождение материального уравнения само по себе представляет сложную проблему теоретической физики.

Ограничиваясь рассмотрением распространения в среде слабых электромагнитных волн, мы можем полагать функциональное соотношение (IV.52) линейным. В этом случае материальное уравнение (IV.52) принимает вид:

$$D_i(\vec{r},t) = \int\limits_{-\infty}^{t} dt' \int d^3\vec{r}'\varepsilon_{ij}(\vec{r},\vec{r}';t,t')E_j(\vec{r}',t'), \tag{IV.53}$$

где введен *тензор диэлектрической проницаемости среды*, ε_{ij}. В дальнейшем будем полагать оптическую среду *стационарной*, т.е., считать, что все ее характеристики в отсутствие электромагнитной волны не зависят от времени. Рассмотрим *монохроматическую* электромагнитную волну, т.е.:

$$\vec{E}(\vec{r},t) = \vec{E}(\vec{r},\omega)e^{-i\omega t}, \tag{IV.54}$$

где

$$\omega \longrightarrow \omega + i\sigma; \quad \sigma \to +0,$$

- *частота* электромагнитной волны, имеющая бесконечно малую отрицательную мнимую часть, так чтобы $\vec{E}(\vec{r},t \to -\infty) \to 0$. Предполагая аналогичную зависимость от времени всех векторов электромагнитного поля и подставляя их в уравнения Максвелла (IV.51), приведем последние к виду:

$$(\vec{\nabla}\vec{D}) = 0; \quad [\vec{\nabla}\vec{E}] = \frac{i\omega}{c}\vec{B};$$

$$[\vec{\nabla}\vec{B}] = -\frac{i\omega}{c}\vec{D}; \quad (\vec{\nabla}\vec{B}) = 0. \tag{IV.55}$$

Выражая вектор магнитной индукции, $\vec{B}$, из второго уравнения (IV.55) и подставляя результат в третье уравнение с учетом правила вычисления двойного векторного произведения:

$$\big[\vec{\nabla}\ \big[\vec{\nabla}\vec{E}\big]\big] = \vec{\nabla}\big(\vec{\nabla}\vec{E}\big) - \big(\vec{\nabla}\vec{\nabla}\big)\vec{E} =$$

$$= \vec{\nabla}\big(\vec{\nabla}\vec{E}\big) - \Delta\vec{E},$$

получим:

$$\vec{\nabla}\big(\overrightarrow{\nabla}\,\overrightarrow{E}\big) - \Delta\vec{E} = \frac{\omega^2}{c^2}\vec{D}, \tag{IV.56}$$

где Δ - оператор Лапласса, который в криволинейных координатах является оператором Бельтрами II-го рода (см, например, [34]) и имеет вид:

$$\Delta E^i = g^{jk}E^i_{,jk}. \tag{IV.57}$$

Заметим, что подстановка вектора $\vec{B}$ из второго уравнения (IV.55) в четвертое обращает последнее уравнение в тождество вследствие коммутационных соотношений (**??**).

Таким образом:

> В стационарной анизотропной среде уравнения Максвелла для монохроматической электромагнитной волны при известном материальном уравнении поля сводятся к двум уравнениям: первому уравнению из системы (IV.55) и уравнению (IV.56) относительно векторов $\vec{E}$ и $\vec{D}$.

Далее представим координаты трехмерных векторов $E_i(\vec{r},\omega)$, $D_i(\vec{r},\omega)$ в виде:

$$E_i(\vec{r},\omega) = \mathcal{E}_i(\vec{r},\omega)e^{iU(\vec{r})}, \tag{IV.58}$$

где согласно процедуре приближения геометрической оптики векторы типа $\mathcal{E}_i$ полагаются медленно меняющимися функциями координат, а скалярная функция $U(\vec{r})$ полагается большой величиной вместе со своими первыми производными; вторые же производные этой функции полагаются малыми величинами. Пусть далее: L - характерный размер неоднородности системы, λ - длина электромагнитной волны. Будем полагать отношение $\lambda/L = \epsilon$ малым параметром, так что:

$$\partial_i\mathcal{E} \sim \frac{\mathcal{E}}{L}; \quad \partial_i U \sim \frac{U}{\lambda}; \quad \partial_{ij}U \sim \frac{U}{\lambda L}. \tag{IV.59}$$

Функцию $U(\vec{r})$ в дальнейшем будем называть *функцией эйконала, или просто — эйконалом*. Частные производные этой функции являются компонентами ковариантного вектора; будем называть этот вектор *волновым вектором*: $k_i(\vec{r}) = \partial_i U$. Векторы же типа $\mathcal{E}_i$ будем называть *амплитудами* электромагнитной волны.

Согласно (IV.54) и (IV.58) *поверхность постоянной фазы электромагнитной волны* описывается уравнением:

$$U(\vec{r}) - \omega t = \text{Const}\,. \tag{IV.60}$$

Вычисляя дифференциал от обеих частей этого уравнения, получим:

$$k_i dx^i - \omega dt = 0.$$

Таким образом, найдем:

$$k_i v^i = \omega, \tag{IV.61}$$

где

$$v^i = \frac{dx^i}{dt}$$

- так называемый *вектор фазовой скорости электромагнитной волны*. Если положить, например, $dx^i = (dx^1, 0, 0)$, то получим:

$$v = \frac{\omega}{k}, \tag{IV.62}$$

где k - длина волнового вектора. Обычно фазовая скорость волны больше или равна скорости света. Однако, фазовая скорость электромагнитной волны не является реальной физической характеристикой скорости волны. Такой характеристикой является *групповая скорость волны*:

$$u = \frac{\partial \omega}{\partial k}, \tag{IV.63}$$

- именно с групповой скоростью переносится энергия электромагнитной волны, с этой же скоростью в оптической среде движется и квант электромагнитных колебаний - *фотон*, который в пределе геометрической оптики можно представить как элементарную частицу, движущуюся по световому лучу.

Волновой вектор электромагнитной волны $k_i\vec{r}$ выделяет в каждой точке $M(\vec{r})$ среды единичное направление:

$$n_i(\vec{r}) = \frac{k_i}{k}; \quad k^2(\vec{r}) = g^{ij}k_i k_j. \tag{IV.64}$$

Положим далее в соответствие с приближением геометрической оптики материальное уравнение поля для амплитуд $\mathcal{E}$ в виде:

$$\mathcal{D}_i = \varepsilon_{ik}(\omega, \vec{k}(\vec{r}); \vec{r})\mathcal{E}^k. \tag{IV.65}$$

Разрешая (IV.65) относительно вектора $\vec{\mathcal{D}}$, найдем:

$$\mathcal{E}_i = \tilde{\varepsilon}_{ik}(\omega, \vec{k}(\vec{r}); \vec{r})\mathcal{D}^k, \tag{IV.66}$$

где $\tilde{\varepsilon}_{ik}$ -тензор, обратный к тензору ε_{ik}:

$$\tilde{\varepsilon}^{ij}\varepsilon_{jk} = \delta^i_k.$$

Учитывая приближение геометрической оптики $\lambda/L \to 0$ и оценки (IV.59), получим в первом приближении геометрической оптики из первого уравнения (IV.55) и (IV.56):

$$\left(\vec{k}\,\vec{\mathcal{D}}\right) = 0; \tag{IV.67}$$

$$\left(k^2\tilde{\varepsilon}^{ij} - k^i k_m \tilde{\varepsilon}^{mj} - \frac{\omega^2}{c^2}\delta^{ij}\right)\mathcal{D}_j = 0. \tag{IV.68}$$

Вследствие взаимной ортогональности векторов $\vec{k}$ и $\vec{\mathcal{D}}$ выполняется тождество:

$$\mathcal{D}^i \equiv \left(\delta^i_j - \frac{k^i k_j}{k^2}\right)\mathcal{D}^j\,. \tag{IV.69}$$

Представим обратный тензор диэлектрической проницаемости в виде:

$$\tilde{\varepsilon}^{ij} = \delta^{ij} + \delta\varepsilon^{ij}\,. \tag{IV.70}$$

Тогда с учетом (IV.67) уравнения (IV.68) можно записать в виде:

$$\left(\left(k^2 - \frac{\omega^2}{c^2}\right)\delta^{ij} + \delta\varepsilon^{mj}\left(k^2\delta^i_m - k^i k_m\right)\right)\mathcal{D}_j = 0. \tag{IV.71}$$

Уравнения (IV.71) представляют собой систему однородных линейных алгебраических уравнений относительно вектора $\mathcal{D}_j$. Как известно из алгебры, для их нетривиальной разрешимости необходимо и достаточно равенства нулю определителя системы:

$$\det\left\|\left(k^2 - \frac{\omega^2}{c^2}\right)\delta^{ij} + \delta\varepsilon^{mj}\left(k^2\delta^i_m - k^i k_m\right)\right\| = 0. \tag{IV.72}$$

Уравнение (IV.72) можно рассматривать как алгебраическое уравнение относительно волнового вектора, $k(\vec{r})$. Его возможные решения имеют вид:

$$k = k(\vec{r}, \omega) \tag{IV.73}$$

и описывают возможные моды электромагнитных волн в анизотропной среде. Это уравнение называется *дисперсионным уравнением*. Подставляя найденные из дисперсионного уравнения решения в уравнения (IV.71), соответствующие каждой моде колебаний фундаментальные решения, в которых независимых каждый раз будет не более двух, соответствующих двум различным *состояния поляризации электромагнитной волны*.

IV.4.2 Принцип Ферма

Пусть параметрические уравнения движения фотона имеют вид:

$$\vec{r} = \vec{r}(t) \Rightarrow x^i = x^i(t), \tag{IV.74}$$

где параметр t есть время. Бесконечно малая *оптическая* длина пути фотона, $d\tau$, в анизотропной оптической среде с *тензором преломления* n_{ik} определяется формулой:

$$d\tau^2 = n_{ik} dx^i dx^k, \tag{IV.75}$$

где $d\vec{r} = (dx^1, dx^2, dx^3)$ - вектор бесконечно малого смещения фотона. Абсолютная же величина скорости фотона в среде, v, в направлении $d\vec{r}$ определяется соотношением:

$$v = \sqrt{n_{ik} \dot{x}^i \dot{x}^k}. \tag{IV.76}$$

IV.5 Вывод уравнений световых лучей из принципа Ферма

Итак, зададим функцию Лагранжа светового луча в виде:

$$L = \sqrt{n_{ik} dx^i dx^k}. \tag{IV.77}$$

Дифференцируя (IV.74) по времени, получим:

$$dx^i = \dot{x}^i dt\,,$$

где:

$$\dot{x}^i = \frac{dx^i}{dt}\,.$$

Таким образом, функционал действия для световых лучей принимает вид:

$$S = \int_1^2 \sqrt{n_{ik} \dot{x}^i \dot{x}^k} dt\,. \tag{IV.78}$$

Сравнивая формулы этого раздела с соответствующими формулами предыдущего раздела, а также (IV.1.3): (IV.78), (IV.75) и т.д., замечаем, что при подстановках соответствующих выражений:

$$\tau \to s; \quad n_{ik} \to g_{ik} \tag{IV.79}$$

задача о нахождении траектории фотона в анизотропной и неоднородной оптической среде ничем формально не отличается от задачи о геодезической линии в римановом пространстве. Итак, можно сделать следующее утверждение:

> В пределе геометрической оптики траектория фотона (луч света) в оптически прозрачной анизотропной и неоднородной среде с тензором преломления $n_{ik}(x)$ совпадает с геодезической линией в римановом пространстве с метрическим тензором $g_{ik}(x) = n_{ik}(x)$.

Таким образом устанавливается максимально тесная связь между геометрической оптикой и римановой геометрией.

В связи с существованием указанного *взаимно однозначного соответствия* введем объекты $\Omega_{ij,k}$ и Ω^i_{jk}, индуцированные тензором преломления, как метрикой, -

$$\Omega_{ij,k} = \frac{1}{2}\left(\partial_i n_{jk} + \partial_j n_{ik} - \partial_k n_{ij}\right); \tag{IV.80}$$

$$\Omega^i_{jk} = n^{jl}\Omega_{jk,l}, \tag{IV.81}$$

где контравариантный тензор n^{jl} является обратным к тензору преломления n_{ik}, т.е.:

$$n^{jl}n_{il} = \delta^j_i. \tag{IV.82}$$

Объекты $\Omega_{ij,k}$ и Ω^i_{jk}, индуцированные тензором преломления, будем в дальнейшем для простоты называть *оптическими символами Кристоффеля I-го и II-го рода*, соответственно.

Отметим следующее важное обстоятельство. Сами эти объекты, как и символы Кристоффеля, не являются тензорными объектами. Однако, с другой стороны хорошо известно, что *разность соответствующих символов Кристоффеля, индуцированных различными метриками, в общей координации представляет тензор III-го ранга.* Так как само физическое пространство, в котором находится и оптическая среда, является евклидовым, то в декартовых координатах [4] символы Кристоффеля этого пространства обращаются в нуль. В криволинейных же координатах символы Кристоффеля евклидова пространства, $\Gamma_{ij,k}$ и Γ^i_{jk}, вообще говоря, отличны от нуля. Разности же

$$\delta\Omega_{ij,k} = \Omega_{ij,k} - \Gamma_{ij,k}; \quad \delta\Omega^i_{jk} = \Omega^i_{jk} - \Gamma^i_{jk} \tag{IV.83}$$

[4] А более точно, в аффинных координатах.

являются компонентами тензоров и описывают поэтому тензорные свойства оптической среды.

Таким образом, воспользовавшись результатами раздела (IV.1.3), мы сразу можем выписать *уравнения распространения света в оптически прозрачной неоднородной и анизотропной среде*, которые, как было указано выше, являются уравнениями геодезических линий в метрике (IV.75):

$$\frac{d^2x^i}{d\tau^2} + \Omega^i_{jk}\frac{dx^j}{d\tau}\frac{dx^k}{d\tau} = 0. \tag{IV.84}$$

Заметим, что дифференцирование в этих уравнениях осуществляется не по параметру времени, t, а по оптической длине пути, τ. Вследствие (IV.75) выполняется *соотношение нормировки*:

$$n_{ik}\frac{dx^i}{d\tau}\frac{dx^k}{d\tau} = 1. \tag{IV.85}$$

Для решения системы обыкновенных дифференциальных уравнений (IV.84), как известно, необходимо задать начальные условия в некоторой точке $M(x_0) = M_0$:

$$x^i(\tau_0) = x^i_0; \qquad \frac{dx^i}{d\tau}(\tau_0) = \left.\frac{dx^i}{d\tau}\right|^0 = u^i_0, \tag{IV.86}$$

где u^i - единичный в смысле метрики n_{ik} и касательный к световой траектории вектор:

$$u^i = \frac{dx^i}{d\tau}. \tag{IV.87}$$

Соотношение нормировки (IV.85) должно выполняться в любой точке светового луча, в том числе, и в начальной:

$$n_{ik}(x_0)u^i_0u^k_0 = 1. \tag{IV.88}$$

При численном решении задачи удобнее всего воспользоваться соотношением (IV.88), выражая одну из компонент вектора u^i_0 через две другие. При этом согласно теории геодезических линий (раздел (IV.1.3)) соотношение нормировки (IV.85) автоматически будет выполняться в каждой точке световой кривой.

IV.5.1 Свойства тензора преломления

Наряду с квадратичной формой оптической среды (IV.75) можно рассмотреть и квадратичную форму пустого риманова пространства[5]:

$$ds^2 = g_{ik}(x)dx^idx^k, \tag{IV.89}$$

[5]Которое в рассматриваемом нами случае является евклидовым.

где $||g_{ik}|| = G$ - невырожденная матрица, причем квадратичная форма (IV.89) положительно определена, вследствие чего:

$$g = \det \|G\| > 0. \tag{IV.90}$$

Таким образом, в каждой точке $M(x^i)$ множества точек $\mathcal{M}$ имеем пару квадратичных форм, (IV.75) и (IV.89), определяемых симметричными тензорами 2-го ранга n_{ik} и g_{ik}. [6] Согласно теории квадратичных форм[7] пару квадратичных форм, из которых одна положительно определена, невырожденными преобразованиями координат в каждой произвольной, фиксированной точке M_0 можно *одновременно* привести к каноническому виду:

$$G_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad N_0 = \begin{pmatrix} n_{(1)} & 0 & 0 \\ 0 & n_{(2)} & 0 \\ 0 & 0 & n_{(3)} \end{pmatrix}, \tag{IV.91}$$

где n_i - собственные значения матрицы N, определяемые характеристическим уравнением:

$$\det \|N - nG\| = 0. \tag{IV.92}$$

Собственные же векторы, $\underset{(\alpha)}{k}{}^i$; $\alpha = \overline{1,3}$, определяемые уравнениями:

$$(N - nG)K = 0 \Rightarrow \left(n_{ij} - n_{(\alpha)} g_{ij}\right) \underset{(\alpha)}{k}{}^j = 0, \tag{IV.93}$$

будем называть *оптическими осями среды.*

IV.6 Пакет `optica`

Для построения световых лучей и их трубок создана пакет программных процедур `optica` [110][8]. Покажем некоторые коды программных процедур. Введем параметры процедуры:
`optica(Coord,S,S1,G,M0,V0,Init,N,a,graf)`: `Coord` - список координат, S - канонический параметр – оптическая длина пути, S1- период изменения параметра S (число шагов, т.е. S=0..S1), G - метрика (тензор преломления, связанный с тензором диэлектрической проницаемости), M0 - координаты центральной точки M0 геодезической трубки, V0 -кординаты начального

[6] Напомним, что матрица квадратичной формы всегда симметрична.
[7] См., например, [35]
[8] При использовании этого пакета ссылка на него обязательна

направляющего вектора K0 трубки, Init- определяет максимальные и минимальные координаты, в пределах которых график будет отображаться на экране, A=[xmin..xmax], B=[ymin..ymax], C=[zmin..zmax] дает поле обзора получаемого изображения; N - количество кадров анимации; a - начальный радиус трубки геодезических линий (лучей), graf - параметр, принимающий значения graphic - для построения графика, animate - для построения анимации.

```
> restart:
> with(tensor):
> Optic:=table():
> Optic[optica]:=proc(Coord,S,S1,G,M0,V0,Init,N,a,graf)
local nn,i,xx,GG,j,g,GG_1,g_1,dg,Cf1,Cf2,Geod,XX,U,dU,u,
du,v,dc,Geo,GEO,norm,dd,Geod1,Geod2,Norm,GEOD,Geodesic,XV,
Point,K0,k,K,KK,kk0,V,VV,Inits0,SS,XVS,M,kk,Inits,A,B,C,D,E,
Sol,points,STR,gr,vpoints,STV,grf,SumGSS,GeoU0,u_U0,L,
GSS,r,XV1,ss,kol,nnn,ssk,t,S_k,num,Points,SSK,kl,Points1,
gr1,T,ll,J,ii,GeoUL:
nn:=nops(Coord):
for i from 1 to nn do:
xx||i:=Coord[i]:
end do:
GG:=array(1..nn,1..nn,symmetric,sparse):
for i from 1 to nn do:
for j from 1 to nn do:
GG[i,j]:=G[i,j]:
end do:end do:
g:=create([-1,-1],eval(GG)):
GG_1:=linalg[inverse](GG):
g_1:=create([1,1],eval(GG_1)):
dg:=d1metric(g,Coord):
Cf1:=Christoffel1(dg):
Cf2:=Christoffel2(g_1,Cf1):
Geod:=geodesic_eqns(Coord,S,Cf2):
XX(S):=subs(seq(Coord[i]=Coord[i](S),i=1..nn),Coord):
U(S):=diff(XX(S),S):
dU(S):=diff(U(S),S):
u:=create([1],array(1..nn,U(S))):
```

```
du:=create([1],array(1..nn,dU(S))):
v:=create([1],array(1..nn,[seq(v[i](S),i=1..nn)])):
dc:=get_compts(prod(prod(Cf2,u,[2,1]),u,[2,1])):
Geo:=linalg[matadd](dU(S),dc):
GEO:=[seq(Geo[i]=0,i=1..nn)]:
norm:=get_compts(prod(prod(g,u,[1,1]),u,[1,1]))=1:
dd:=get_compts(prod(prod(Cf2,v,[2,1]),v,[2,1])):
Geod1:=[seq(diff(XX(S)[i],S)=v[i](S),i=1..nn)]:
Geod2:=[seq(diff(v[i](S),S)=-dd[i],i=1..nn)]:
Norm:=get_compts(prod(prod(g,v,[1,1]),v,[1,1]))=1:
GEOD:=subs(seq(Coord[i]=Coord[i](S),i=1..nn),Geod2):
Geodesic:=[op(Geod1),op(GEOD)]:
XV:=seq(Coord[i](S),i=1..nn),seq(v[i](S),i=1..nn):
XV1:=seq(Coord[i](S),i=1..nn):
Point:=seq(Coord[i](0)=M0[i],i=1..nn):
K0:=seq(k[i]=V0[i],i=1..nn):
K:=array(1..nn,[seq(k[i],i=1..nn)]):
k:=create([1],eval(K)):
KK:=eval(subs(S=0,subs(seq(Coord[i]=Coord[i](S),
i=1..nn),get_compts(prod(prod(g,k,[1,1]),k,[1,1])))))):
kk0:=eval(subs(Point,KK)):
V:=linalg[scalarmul](K,1/sqrt(kk0)):
VV:=subs(K0,op(V)):
Inits0:={Point,seq(v[i](0)=VV[i],i=1..nn)}:
SS:=dsolve({op(Geodesic)} union Inits0,{XV},
type=numeric,output=listprocedure):
XVS:=subs(SS,[XV]):
A:=XVS(1):
for i from 0 to 7 do M(i):=x1(0)=rhs(Point[1])+a,
x2(0)=rhs(Point[2])+i*Pi/4,Point[3]: od:
kk:=(i)->eval(subs(M(i),KK)):
V:=(i)->linalg[scalarmul](K,1/sqrt(kk(i))):
VV:=(i)->subs(K0,V(i)): VV(3):
Inits:=(i)->[M(i),v[1](0)=VV(i)[1],v[2](0)=VV(i)[2],
v[3](0)=VV(i)[3]]:
B:=Inits(3):
for i from 0 to 7 do SS:=(i)->dsolve({op(Geodesic)}
union {op(Inits(i))},{XV},type=numeric,output=listprocedure):
```

```
XVS:=(i)->subs(SS(i),[XV]): od:
C:=XVS(1)(3):
nnn:=proc(S) local SS,SSS:
SSK:=floor(S):
SSS:=floor(S*5):
25+piecewise(SSK<10 or SSK=10,20,SSK>10,SSS):
end proc:
num:=nnn(S1):
S_k:=(kl)->evalf(kl*S1/num):
points:={seq([rhs(M(i)[1])*cos(rhs(M(i)[2])),
rhs(M(i)[1])*sin(rhs(M(i)[2])),
cos(rhs(M(i)[1]))*cos(rhs(M(i)[3]))],i=0..7)}:
STR:=plots[pointplot3d](points,color=BLUE,
symbol=CIRCLE,symbolsize=24,axes=BOXED): STR:
if graf=graphic then
for i from 0 to 7 do
Points:=(i)->[seq([evalf(XVS(i)(S_k(kl))[1]*
cos(XVS(i)(S_k(kl))[2]),4),
evalf(XVS(i)(S_k(kl))[1]*sin(XVS(i)(S_k(kl))[2]),4),
evalf(cos(XVS(i)(S_k(kl))[1])*cos(XVS(i)(S_k(kl))[3]),4)],
kl=0..num)]:
gr:=(i)->plots[pointplot3d](Points(i),style=line,color=black);
od:
D:=plots[display](gr(0),gr(1),gr(2),gr(3),gr(4),
gr(5),gr(6),gr(7),STR):
elif graf=animate then
#ssk(K) - высчитывает число шагов, необходимое для вывода всего
#интервала изменения параметра анимации t
#где S изменяется от [0,S1], N - количество шагов анимации
ssk:=(K)->evalf(S1/N*K,4): ssk(4):
for i from 0 to 7 do
#L(i,S_k) - выводим последовательность точек -
#решений для каждой i-той геодезической
L:=(i,ll)->[seq([evalf(XVS(i)(S_k(kl))[1]*
cos(XVS(i)(S_k(kl))[2]),4),
evalf(XVS(i)(S_k(kl))[1]*sin(XVS(i)(S_k(kl))[2]),4),
evalf(cos(XVS(i)(S_k(kl))[1])*cos(XVS(i)(S_k(kl))[3]),4)],
kl=0..ll)]:
```

```
gr:=(i,ll)->plots[pointplot3d](L(i,ll),style=line,
color=black);
od:
GeoUL:=(ll)->plots[display]([seq(gr(i,ll),i=0..7),STR]):
GeoU0:=plots[display](seq(GeoUL(ll),ll=0..N),
insequence=true,title=convert(t=ssk(N),string));
end if:
end proc:
```

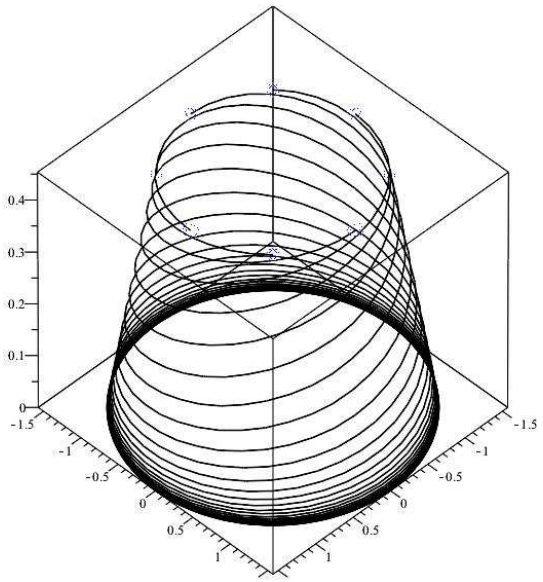


Рис.IV.39. *Применение процедуры* `optica( [x1,x2,x3], s,30 , gg1, [0.1,0,0], [0,-1,0], [[-1.6,1.6], [-1.6,1.6], [0,1]], 50, 1, graphic)` *в графическом формате:луч света вращается в оптической среде.*

Продемонстрируем исполнение программной процедуры.Введем метрику gg1=

$$\begin{pmatrix} \frac{2\sin x1+\cos x1}{\cos^2 x1} & 0 & 0 \\ 0 & \sin^2 x1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

и изобразим график геодезических трубок по сгенерированным начальным условиям: `> AA:=optica([x1,x2,x3], s,30,gg1,[0.1,0,0],[0,-1,0], [[-1.6,1.6], [-1.6,1.6],[0,1]],50,1, graphic):` `>AA[8]` – (полученные геодезические в данном случае совмещаются на одном графике, причем начальные положения геодезических отмечаются синими кружочками):

Введем метрику:

```
> gg4:=[[sin(x1)/(2*cos(x1)^2+1),0,0],[0,sin(x1),0],[0,0,1]]:
```

Введем параметры процедуры:

```
>AA:=optica([x1,x2,x3],s,30,gg4,[0.1,0,0],[0,-1,0],
[[-2.2,2.2],[-2.2,2.2],[-0.5,0.5]],50,1,graphic):

> AA[8];
```

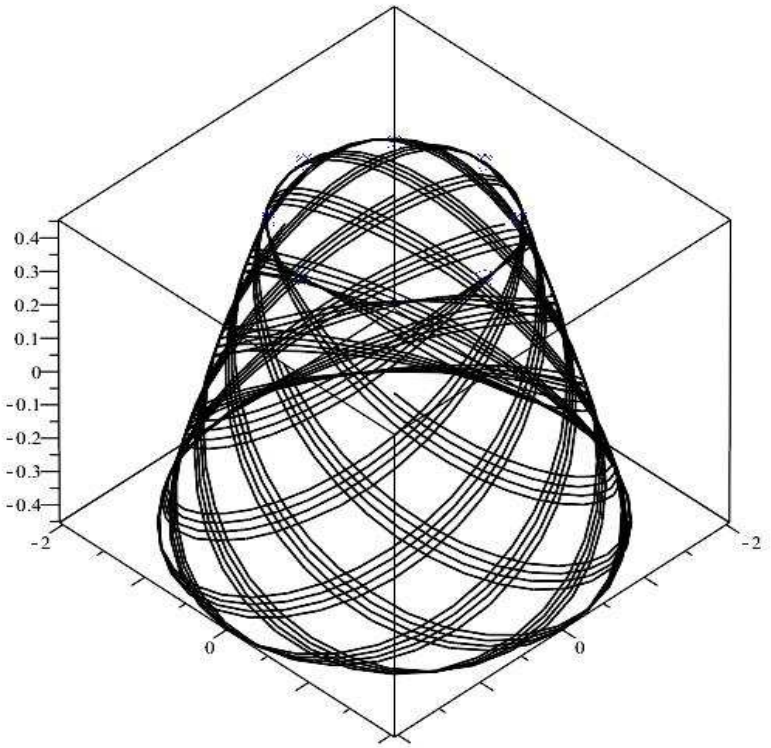


Рис.IV.40 *Изобразим график геодезических трубок по сгенерированным начальным условиям (полученные геодезические совмещаются на одном графике, причем положения геодезических отмечаются синими кружочками).*

Введем метрику:

```
> gg3:=[[1,0,0],[0,x1^2*cos(x3)^2,0],[0,0,x1^2]]:
> AA:=optica([x1,x2,x3],s,10,gg3,[0.1,0,0],
[0,-1,0],[[-1.6,1.6],[-1.6,1.6],[0,1]],50,1,graphic):
```

Построим график геодезических трубок по сгенерированным начальным условиям:

```
> AA[8];
```

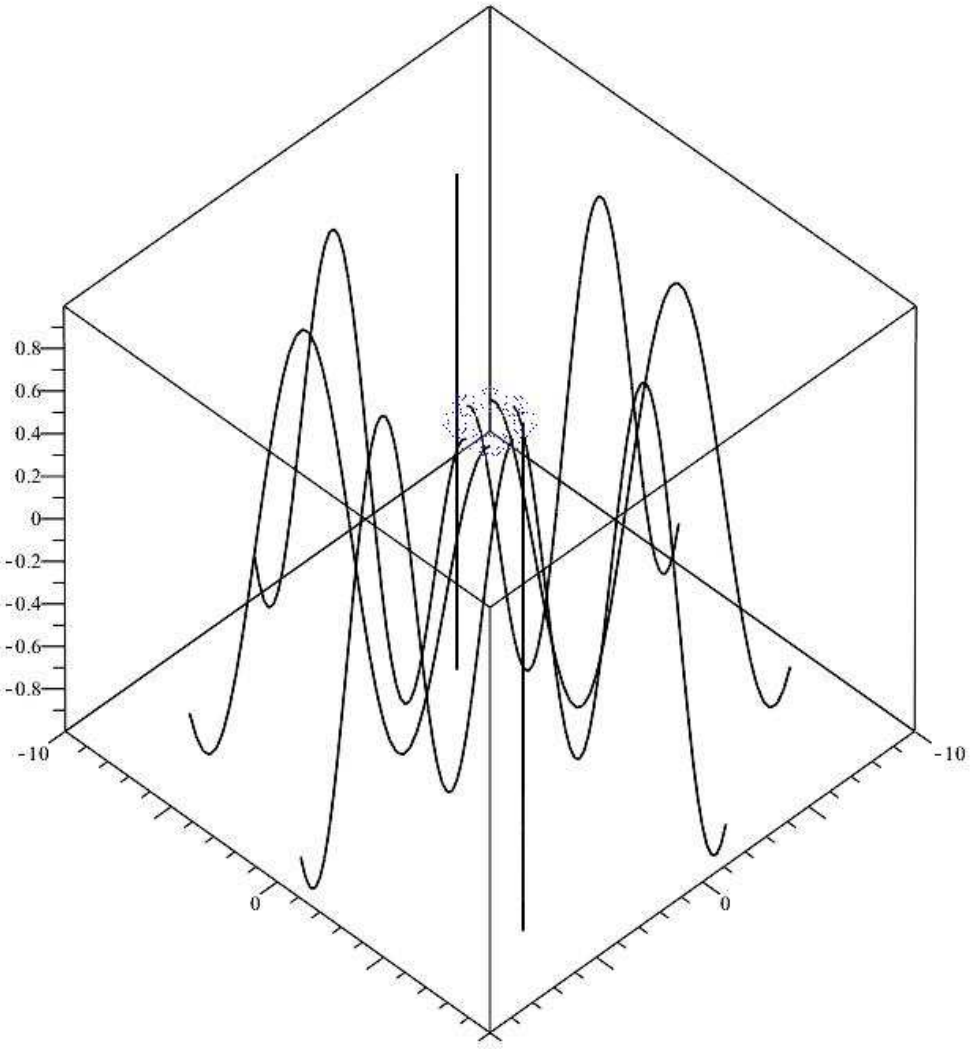


Рис.IV.41 *Изобразим график геодезических трубок по сгенерированным начальным условиям (полученные геодезические совмещаются на одном графике, причем положения геодезических отмечаются синими кружочками).*

Построим динамическую визуализацию геодезических трубок по сгенерированным начальным условиям:

```
>AA:=optica([x1,x2,x3],s,10,gg3,[0.1,0,0],
[0,-1,0],[[-1.6,1.6],[-1.6,1.6],[0,1]],50,1,animate)[8]:AA;
```

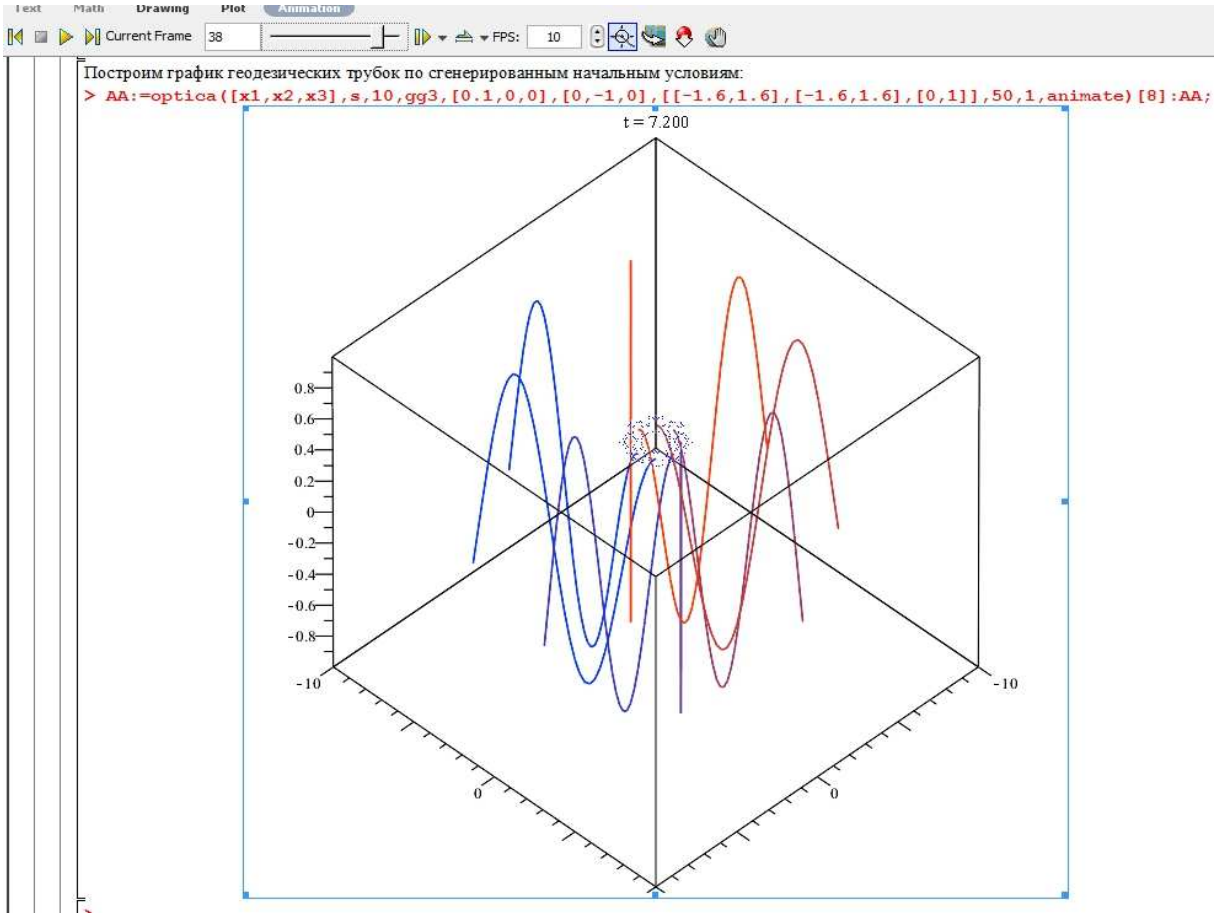


Рис.IV.42 *Окно Maple 17 с 38-м кадром анимации движения световых лучей.*

Введем метрику:

```
> gg4:=[[1/(1+x1^2),0,0],[0,cos(x3)^2/(1+x1^2),0],[0,0,x1^2]]:
> AA:=optica([x1,x2,x3],s,7,gg4,[0.1,0,0],[0,-1,0],
[[-1.6,1.6],[-1.6,1.6],[0,1]],50,1,graphic):
```

Построим график геодезических трубок по сгенерированным начальным условиям:

```
> AA[8];
```

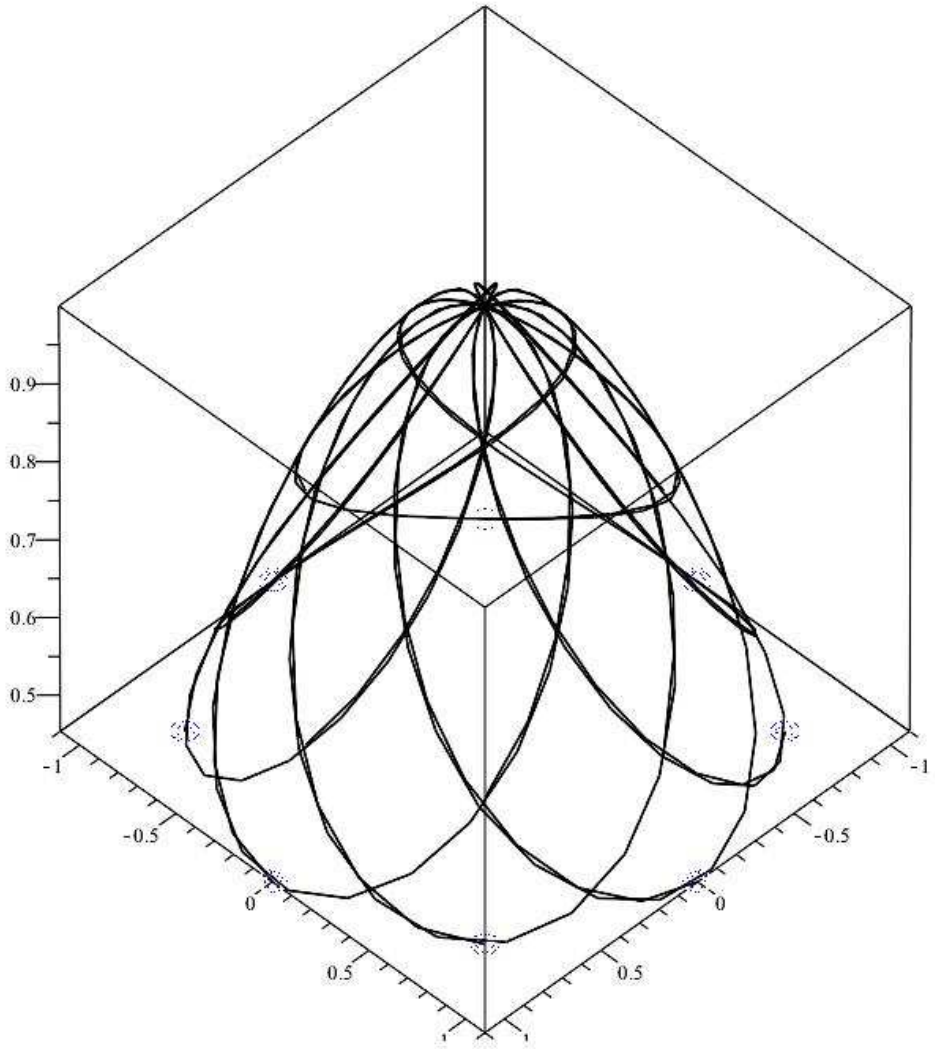


Рис.IV.43 *Трубки световых лучей в метрике gg4.*

Дисперсия – расхождение световых лучей с различной длинной волны (частотой).

```
> g:=[[exp(-omega),0,0],[0,x1^2*cos(x3)^2/(1+omega^2),0],
[0,0,x1^2*exp(-omega)]]:
> dispersion([x1,x2,x3],s,10,g,[1,3,0],[0,1,-1],0);
```

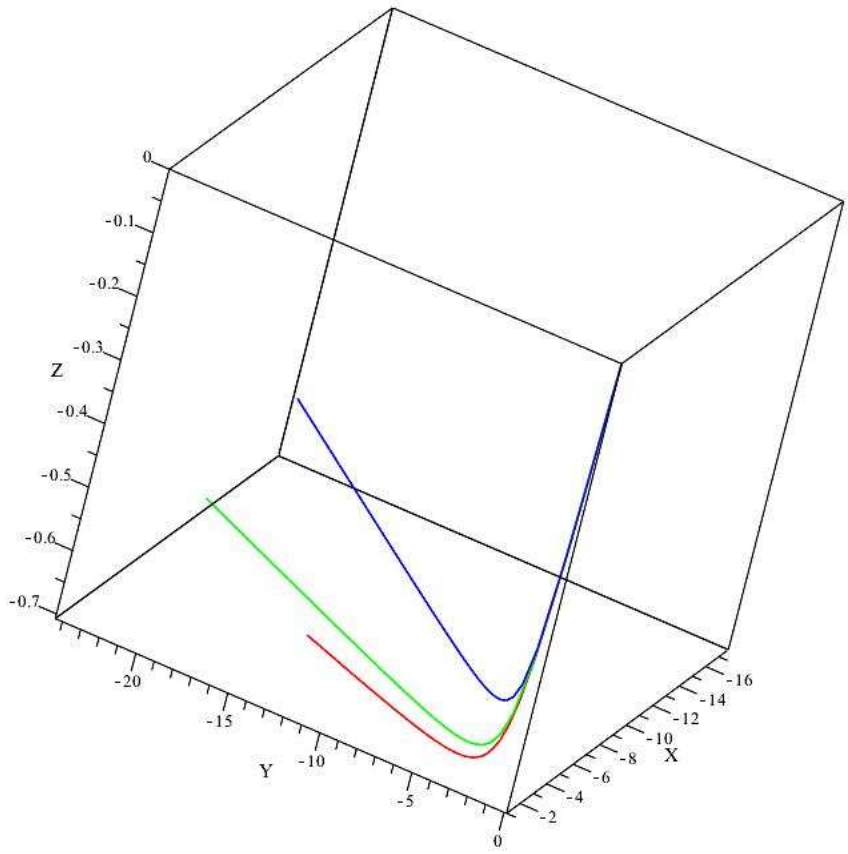


Рис.IV.44 *Дисперсия световых лучей в метрике g.*

IV.7 Геодезические линии в гравитационных полях

Пробные частицы, как и лучи света, движутся в гравитационном поле по геодезическим линиям псевдориманова пространства, метрика которого, g_{ik}, определяется уравнениями Эйнштейна. В математической модели, таким образом, добавляется, во-первых, 4-х мерность, во-вторых, псевдоримановость пространства и, в третьих, фактор массы частицы: массивные частицы движутся по времениподобным геодезическим, безмассовые – по изотропным. Поэтому возможно изобразить лишь 2-мерную или трехмерную проекцию геодезической линии. Это позволяет сделать команда указанного пакета `geodesic4d`, которая также может быть реализована, как в графической, так и в динамической моде.

Приведем пример реализации этой команды для метрики Шварцшильда

(см., например, [32]):

$$ds^2 = \left(1 - \frac{2r_g}{r}\right) dt^2 - \frac{dr^2}{1 - \frac{2r_g}{r}} - r^2(\sin^2\theta d\varphi^2 + d\theta^2)$$

Итак, введем метрику Шварцшильда, выбирая систему единиц, в которой $r_g = 1$:

```
> gg:=[[-1/(1-2/r),0,0,0],[0,-r^2,0,0],
[0,0,-r^2*sin(theta)^2,0],[0,0,0,1-2/r]]:
```

Введем параметры процедуры:

```
> A1:=geodesic4d([r,theta,phi,t],s,gg,[R,Theta,Phi,T],
[40,0,0,0],[0,0,0,1],4,5,38,1,0):
```

и исполним `>A1;`

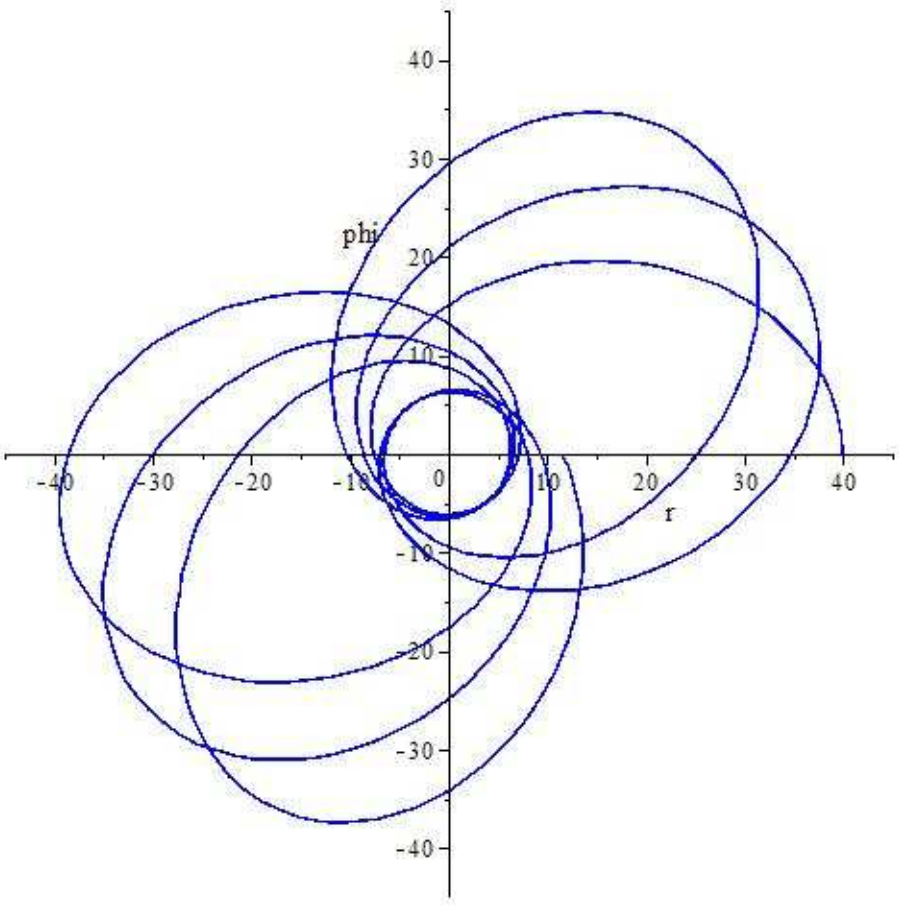


Рис.IV.45 *Движение массивной частицы в поле Шваршильда. Получается «розетка» – смещение перигелия планеты.*

Чтобы получить геодезическую трубку, необходимо применить следующий формат команды:

```
A11:=geodesic4d([r,theta,phi,t],s,gg,[R,Theta,Phi,T],
[40,0,0,0],[0,0,0,1],4,5,38,1,1):
A11;
```

В результате получим следующий рисунок.

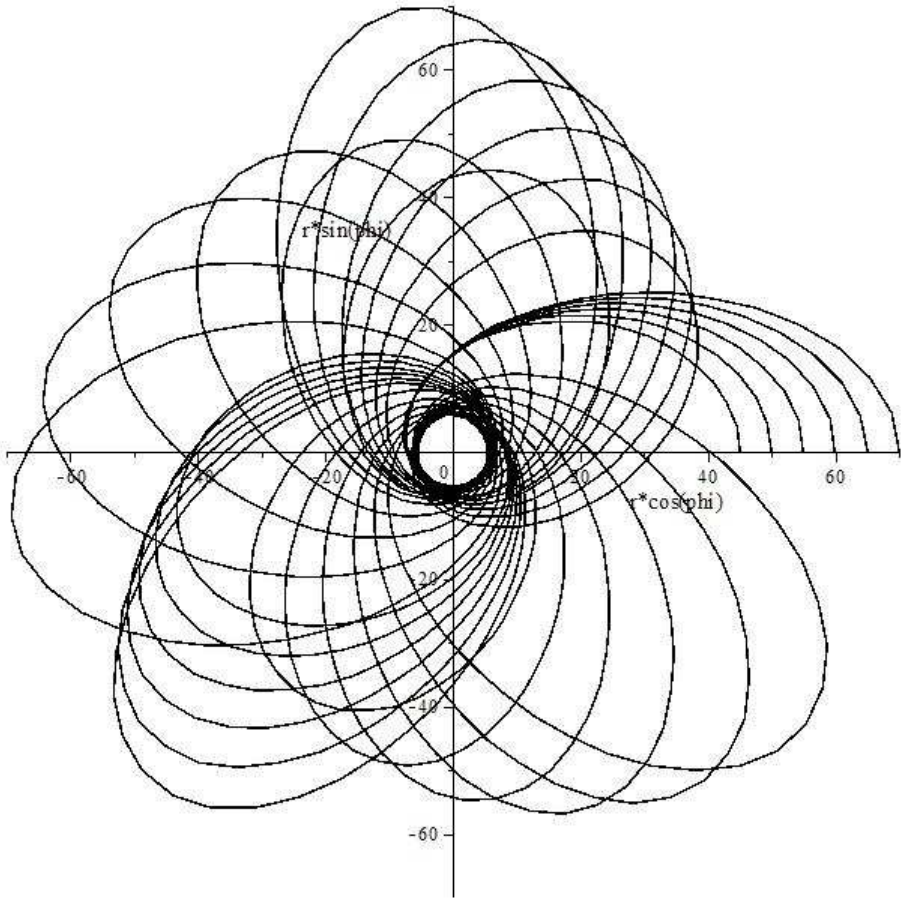


Рис.IV.46 *Движение трубки из 6-ти массивных частиц в поле Шварцшильда. Видно, что вблизи гравитационного радиуса геодезические сгущаются и перепутываются.*

Приведем еще один пример построения модели движения в поле Шварцшильда:

```
A22:=geodesic4d([r,theta,phi,t],s,gg,[R,Theta,Phi,T],
[40,0,0,0],[0,0,0,1],6,5,28,1,1):
A22;
```

В результате получим следующий рисунок.

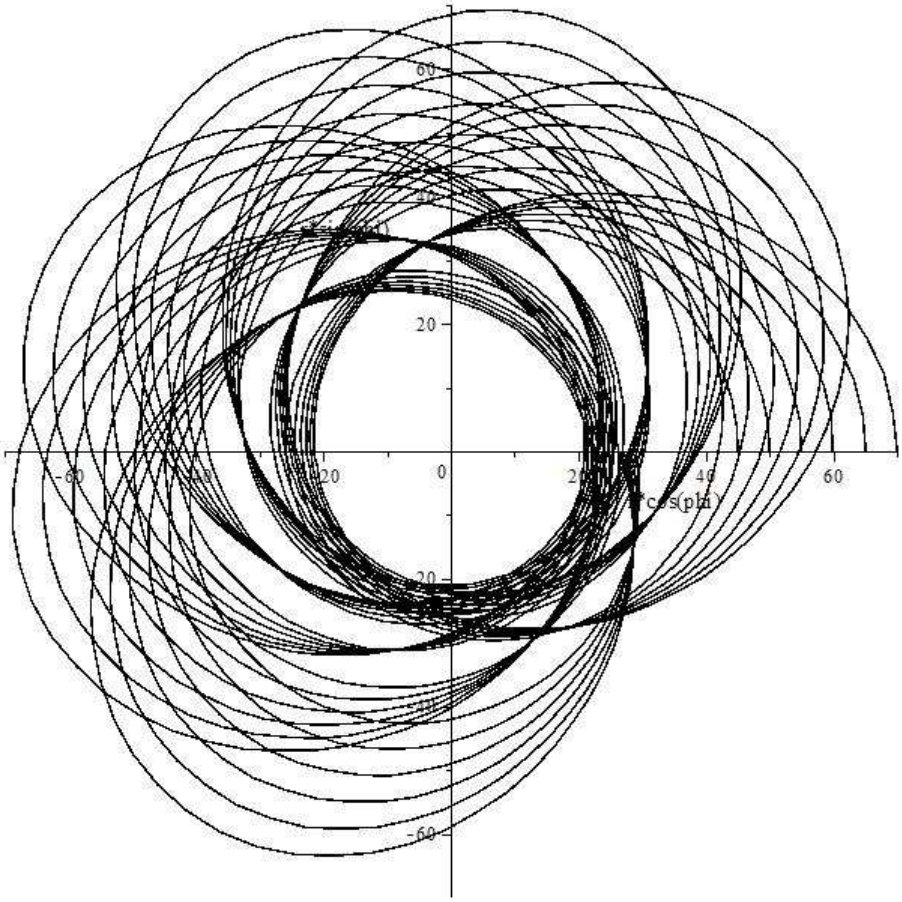


Рис.IV.47 *Движение трубки из 6-ти массивных частиц в поле Шваршильда. Из него также видно, что вблизи гравитационного радиуса геодезические сгущаются и перепутываются.*

Глава V

Моделирование динамических систем в Maple

V.1 Баллистика: артиллерийский тренажер

V.1.1 Уравнения движения

Продемонстрируем основные принципа построения сложных динамических структур. Поскольку в этом разделе нас не интересуют проблемы интегрирования уравнений, в качестве примера создания оснащенной динамической графической модели рассмотрим простую модель движения тела в однородном поле тяготения с линейной силой сопротивления среды:

$$m\frac{d^2\vec{r}}{dt^2} = -k\vec{v} - m\vec{g}, \tag{V.1}$$

где m - масса тела, k - коэффициент линейного трения, $\vec{g}$ - постоянный вектор ускорения свободного падения. Решение этих уравнений с начальными условиями:

$$\vec{r}(0) = \vec{0}; \quad \dot{\vec{r}}(0) = \vec{v}_0 \tag{V.2}$$

известно:

$$x = \frac{mv_0\cos\alpha}{k}\left(1 - e^{-\frac{kt}{m}}\right); \tag{V.3}$$

$$y = -\frac{mgt}{k} + \frac{m(mg + v_0 k\sin\alpha)}{k^2}\left(1 - e^{-\frac{kt}{m}}\right). \tag{V.4}$$

Найдем время в секундах, соответствующее наивысшей точки полета:

```
> T_h:=proc(v,alpha,m,k,g):
> diff(Y(t,v,alpha,m,k,g),t):
> solve(%,t):end proc:
> T_h(100,Pi/3,1,0.1,10);
```

6.238107164

Найдем также радиус-вектор тела как функцию времени и параметров движен ия:

```
> R:=(tau,v,a,M,K,G)->
> [evalf(X(tau,v,a,M,K,G)),evalf(Y(tau,v,a,M,K,G))]:
> evalf(R(5,100,0.45,1,0.1,10));
```

$$[354.2983274,\ 64.6149421]$$

Найдем радиус-вектор точки траектории, соответствующей максимальной высоте:

```
> R_h:=(v,alpha,m,k,g)->R(T_h(v,alpha,m,k,g),v,alpha,m,k,g):
> R_h(100,Pi/3,1,0.1,10);
```

$$[232.0508077,\ 242.2146873]$$

Найдем максимальную высоту полета:

```
> %[2];
```

$$242.2146873$$

Построим график траектории полета тела:

```
> plot([X(t,100,0.45,1,0.1,10),
Y(t,100,0.45,1,0.01,10),t=0..9],
labels=[‘x, метров‘,‘y, метров‘],
labeldirections=[HORIZONTAL,VERTICAL],
titlefont=[TIMES,ROMAN,12],
title=‘Trajectory:
alpha=0,45;v0=100;m=1;k=0,1.‘,
color=BLACK,thickness=2);
```

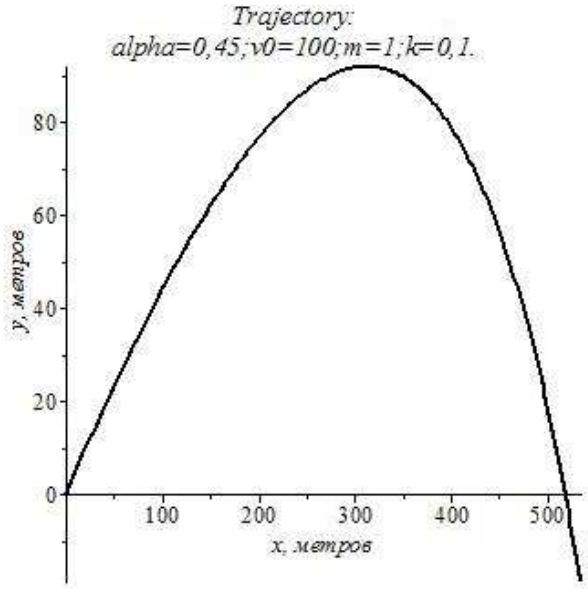


Рис.V.48. Траектория полета тела в поле тяжести.

Безразмерная переменная падения камня:

$\Xi = \frac{k\,t}{m}$

V.1.2 Программные процедуры

Создадим процедуру приближенного вычисления времени полета тела, т.е., такого времени, при котором $y = 0$.

```
> T_max:=(v0,alpha,g)->2*v0*sin(alpha)/g:
> T_max(100,0.5,10);
```

Вычислим это время: 9.588510772

```
> Xi1:=(v0,alpha,m,k,g)->
> fsolve(xi=(1+k*v0*sin(alpha)/(m*g))*(1-exp(-xi)),xi,0..
> 1+k*v0*sin(alpha)/(m*g)):
> Xi1(100,0.5,1,0.1,10);
```

0.8420497943

```
> exp(-Xi1(100,0.5,1,0.1,10));
```

0.4308265120

Время в секундах полета тела: Максимально возможное, при отсутствии сопротивления:

```
> T_max(300,0.5,10);
```

28.76553232

Реальное:

```
> T1:=(v0,alpha,m,k,g)->Xi1(v0,alpha,m,k,g)*m/k:
T1(300,0.5,1,0.1,10);
```

21.55937352

X_max-реальная координата x падения снаряда - дальность стрельбы:

```
> X_max:=(v0,alpha,m,k,g)->
X0(v0,alpha,m,k)*(1-exp(-Xi1(v0,alpha,m,k,g))):
X0(600,Pi/3,10,0.2);
X_max(600,Pi/3,10,0.2,10);
```

15000.00000

12107.83296

Построим управляемую процедуру вычисления радиуса-вектора тела в зависимости от времени, полагая, что после падения (y=0) тело лежит на земле.

```
> RR:=(t,v0,alpha,m,k,g)->piecewise(t<T1(v0,alpha,m,k,g),
evalf(R(t,v0,alpha,m,k,g)),t>=T1(v0,alpha,m,k,g),[evalf(X_max(v0,alpha
,m,k,g)),0]):
```

Этот факт мы учли с помощью кусочно заданной функции `piecewise`. Проверим работу процедуры:

```
> RR(10,300,0.5,1,0.1,10);
```

[1664.213938, 541.2847770]

```
> RR(100,300,0.5,1,0.1,10);
```

[2327.889722, 0]

```
> op(RR(100,300,0.5,1,0.1,10));
```

2327.889722, 0

Проверим исполнение графической процедуры:

```
> plot([RR(t,300,0.5,1,0.1,10)[1],RR(t,300,0.5,1,0.1,10)[2],t=0..10]);
```

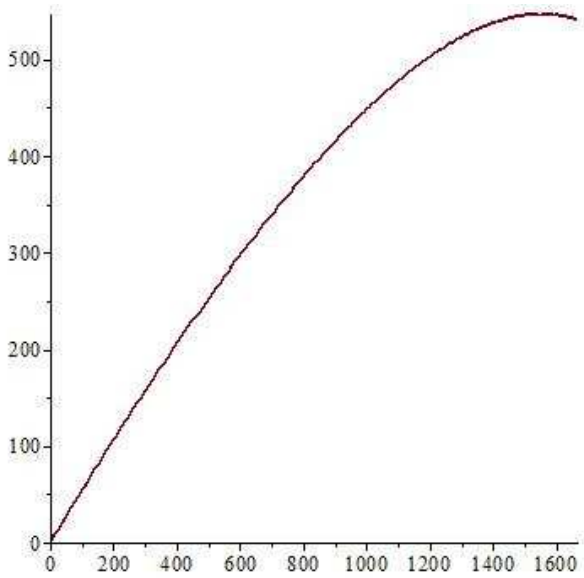


Рис.V.49. Траектория полета тела в поле тяжести.

V.1.3 Создание оснащенной управляемой анимационной процедуры полета шара

Загрузим библиотеку `plottools` для отображения круга, имитирующего тело.

```
> with(plots):
> with(plottools):
```

```
Warning, the name changecoords has been redefined

Warning, the name arrow has been redefined
```

```
> display(plot(x^2,x=0..3,color=blue,thickness=3,scaling=CONSTRAINED),
disk([2,3],1,color=red));
```

Создадим процедуру построения графика полета ядра радиуса d.

```
> Yadro:=proc(t,v0,alpha,m,k,g,d,c) local l,L,h,H:
l:=evalf(-2*d):
h:=evalf(-2*d):
L:=evalf(X_max(v0,alpha,m,k,g)+2*d):
H:=evalf(R_h(v0,alpha,m,k,g)[2]+2*d):
display(disk(RR(t,v0,alpha,m,k,g),d,color=c),
plot([RR(tau,v0,alpha,m,k,g)[1],RR(tau,v0,alpha,m,k,g)[2],tau=0..t],co
lor=c,scaling=CONSTRAINED,view=[l..L,
h..H])): end proc:
```

```
> Yadro(50,100,0.5,1,0.1,10,10,red);
```

Проверим исполнение процедуры.

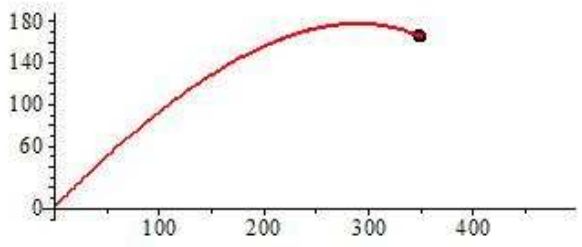


Рис.V.50. Полет ядра в поле тяжести.

Создадим последовательность времен полета:

```
> tt:=(i,v0,alpha,m,k,g)->evalf(T1(v0,alpha,m,k,g)*i/100):
tt(10,100,0.5,1,0.1,10);
```

$$0.8420497943$$

Проверим приближенное обнуление высоты в «момент времени» `i=100`, явно превышающий время полета ядра:

```
> evalf(Y(tt(100,100,0.5,1,0.1,10),100,0.5,1,0.1,10));
```

$$-0.4\,10^{-6}$$

Создадим последовательность графиков полета ядра:

```
> Jadro:=(i,v0,alpha,m,k,g,d,c)->Yadro(tt(i,v0,alpha,m,k,g),
v0,alpha,m,k,g,d,c):
Jadro(58,100,0.8,1,0.1,10,5,red);
```

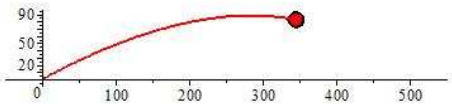


Рис.V.51. Исполнение процедуры `Yadro`.

Создадим еще одну процедуру `Yadro2`.

```
> Yadro2:=proc(t,v0,alpha,m,k,g,d,c)
local l,L,h,H:
l:=evalf(-0.1*X_max(v0,alpha,m,k,g)):
h:=evalf(-0.1*R_h(v0,alpha,m,k,g)[2]):
L:=evalf(1.1*X_max(v0,alpha,m,k,g)):
H:=evalf(1.1*R_h(v0,alpha,m,k,g)[2]):
display(disk(RR(t,v0,alpha,m,k,g),d,color=c),
plot([RR(tau,v0,alpha,m,k,g)[1],RR(tau,v0,alpha,m,k,g)[2],tau=0..t],co
lor=c,scaling=CONSTRAINED,view=[l..L,h..H])):end proc:

> Yadro2(5,100,0.5,1,0.1,10,10,red);
```


Рис.V.52. Проверка исполнения процедуры `Yadro2`.

Создадим теперь процедуру `Yadro1`, оснащенную локальной информацией о времени полета, вычисляемого в заглавии документа с помощью строковой переменной:

```
> Yadro1:=proc(t,v0,alpha,m,k,g,d,c)
local l,L,h,H,ttt,T:
l:=evalf(-0.1*X_max(v0,alpha,m,k,g)):
h:=evalf(-0.1*R_h(v0,alpha,m,k,g)[2]):
L:=evalf(1.1*X_max(v0,alpha,m,k,g)):
H:=evalf(1.1*R_h(v0,alpha,m,k,g)[2]):
T:=T1(v0,alpha,m,k,g):
ttt:=piecewise(t<T,t,t>=T,T):
display(disk(RR(t,v0,alpha,m,k,g),d,color=c),
plot([RR(tau,v0,alpha,m,k,g)[1],RR(tau,v0,alpha,m,k,g)[2],tau=0..t],co
lor=c,scaling=CONSTRAINED,title=convert(evalf(ttt,4),string)),
view=[l..L,h..H]):end proc:
```

Проверим исполнение этой процедуры:

```
> Yadro1(5,100,0.5,1,0.1,10,10,red);
```

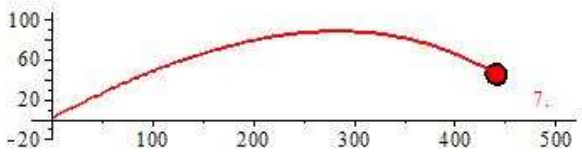


Рис.V.53. Проверка исполнения процедуры `Yadro1`. В титуле отображается локальное время полета 5 секунд.

Создадим теперь процедуру `Yadro3`, в которой с помощью строковой переменной, занесенной в точку с координатой $x(t)$ тела отображается время полета.

```
> Yadro3:=proc(t,v0,alpha,m,k,g,d,c)
local l,L,h,H,T,ttt:
l:=evalf(-2*d):
h:=evalf(-2*d):
L:=evalf(X_max(v0,alpha,m,k,g)+2*d):
H:=evalf(R_h(v0,alpha,m,k,g)[2]+2*d):
T:=T1(v0,alpha,m,k,g):
ttt:=piecewise(t<T,t,t>=T,T):
display(disk(RR(t,v0,alpha,m,k,g),d,color=c),
plot([RR(tau,v0,alpha,m,k,g)[1],RR(tau,v0,alpha,m,k,g)[2],tau=0..t],co
lor=c,scaling=CONSTRAINED,view=[l..L,h..H]),
plots[textplot]([X_max(v0,alpha,m,k,g),2*d,convert(evalf(ttt,4),string
)],color=c,align=LEFT)):end proc:
```

Проверим исполнение этой процедуры:

```
> Yadro3(7,100,0.5,1,0.1,10,10,red);
```

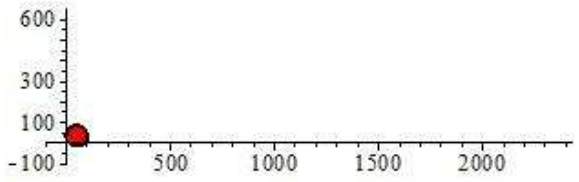


Рис.V.54. Проверка исполнения процедуры `Yadro3`. Рядом с изображением тела отображается локальное время полета 7 секунд.

С помощью команды display с опцией insequence=true создадим процедуру анимации полета тела:

```
> Yadro_fly:=(v0,alpha,m,k,g,d,c)->display(seq(Jadro(i,v0,alpha,m,k,g,d
,c),i=1..100),insequence=true,scaling=CONSTRAINED):
> Yadro_fly(300,0.5,1,0.1,10,50,red);
```

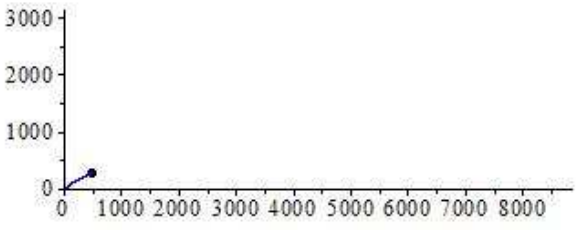


Рис.V.55. Проверка исполнения процедуры `Yadro_fly`.

Последовательно повторяя такие шаги, мы построим окончательную управляемую процедуру динамической графики, позволяющую производить сразу 3 выстрела с разными начальными условиями и параметрами. Таким образом, удается создать тренажер, на котором можно изучать свойства траектории движущегося в поле тяжести тела.

```
> Fly_3Yadro(SS1,SS2,SS3);
```

```
>Fly_3Yadro:=proc(S1,S2,S3)
local L1,L2,L3,L,H1,H2,H3,H,T,T_1,T_2,T_3,ttt_1,ttt_2,ttt_3,d,i,
Ya1_i,Ya2_i,Ya3_i,c1,c2,c3,d1,d2,d3,s1,s2,s3,tau,t_i,G_i,Hm,Lm:
s1:=S1[1],S1[2],S1[3],S1[4],S1[5]:
s2:=S2[1],S2[2],S2[3],S2[4],S2[5]:
s3:=S3[1],S3[2],S3[3],S3[4],S3[5]:
d1:=S1[6]:c1:=S1[7]:d2:=S2[6]:c2:=S2[7]:d3:=S3[6]:c3:=S3[7]:
d:=evalf(1.2*max(d1,d2,d3)):
H1:=evalf(R_h(s1)[2]):
L1:=evalf(X_max(s1)):
H2:=evalf(R_h(s2)[2]):
L2:=evalf(X_max(s2)):
H3:=evalf(R_h(s3)[2]):
L3:=evalf(X_max(s3)):
H:=max(H1,H2,H3): L:=max(L1,L2,L3):
Hm:=evalf(H+2*d):Lm:=evalf(L+2*d):
T_1:=T1(s1):T_2:=T1(s2):T_3:=T1(s3):
T:=max(T_1,T_2,T_3):
plots[display](Yadro4_fly(T,op(S1),L,H),
Yadro4_fly(T,op(S2),L,H),Yadro5_fly(T,op(S3),L,H)):end proc:
```

V.1.4 Численные эксперименты на основе многопараметрической модели

Итак, в созданной программной процедуре Yadro_fly(v0,alpha,m,k,g,d,c) - первый параметр, v0, - скорость в м/сек, alpha - угол вылета в радианах, m - масса тела в кг, k - коэффициент трения в кг/сек, g - ускорение свободного падения в м/^{2}, d - радиус ядра в м, c- цвет ядра.

Зависимость траектории от угла бросания:
Выпустим 3 снаряда под разными углами Pi/6, Pi/4, Pi/3 и прочими одинаковыми характеристиками

```
>SS1:=[1000,Pi/4,1,0.1,10,50,blue];
SS2:=[1000,Pi/6,1,0.1,10,50,black];
SS3:=[1000,Pi/3,1,0.1,10,50,red];
```

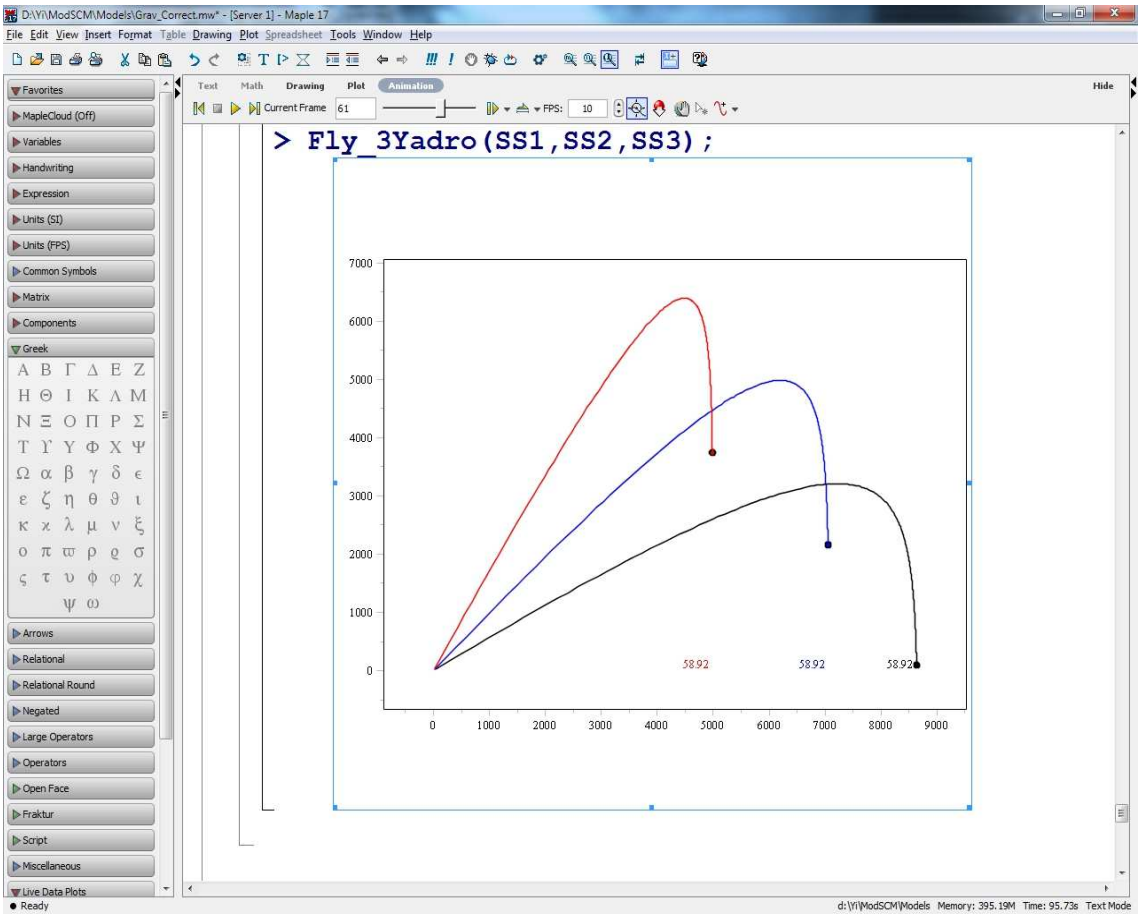


Рис.V.56. Окно Maple 17 с демонстрацией работы тренажера по изучению движения тела в поле тяжести. Показан 61-й кадр из 100-кадроволго фильма.

V.2 Компьютерная модель линейных колебаний

```
> restart:
> CompMod:=Table():
```

Зададим уравнение линейных колебаний с линейной силой трения:

$$\frac{d^2x}{dt^2} + m\beta_0\frac{dx}{dt} + m\omega_0^2 x = f(t),$$

где m – масса груза, β_0 – коэффициент линейного трения, ω_0 – собственная частота колебаний ($omega_0^2 = k/m$, где k - коэффициент упругости, $f(t)$ - внешняя (вынуждающая сила).

```
> CompMod[EqOscillations]:=
> proc(x,t,f,omega0,beta0) local X,F,T,dX,d2X:
> X:=(T)->subs(t=T,x(t)):
> F:=(T)->subs(t=T,f):
> dX:=(T)->subs(t=T,diff(x(t),t)):
> d2X:=(T)->subs(t=T,diff(x(t),t$2)):
> d2X(t)+beta0*dX(t)+omega0^2*X(t)=F(t):
> end proc:
```

```
> CompMod[EqOscillations](xi,tau,exp(-tau)*cos(tau),3,0.1);
```

$$\frac{d^2}{d\tau^2}\xi(\tau) + 0.1\,\frac{d}{d\tau}\xi(\tau) + 9\,\xi(\tau) = \mathrm{e}^{-\tau}\cos(\tau)$$

$$x(t0) = x0,\ v(t0) = v0$$

Зададим начальные условия в стандартном виде: :

```
> CompMod[Inits]:=proc(x,t,InVal) local X,T,dX:
> X:=(T)->subs(t=T,x(t)):
> dX:=subs(t=InVal[1],D(x)(t)):
> [X(InVal[1])=InVal[2],dX=InVal[3]]:
> end proc:
```

```
> CompMod[Inits](xi,tau,[1,2,3]);
```

$$[\xi(1) = 2, \mathrm{D}(\xi)(1) = 3]$$

Создадим процедуру точного решения (в квадратурах) уравнения колебаний t – список вида [t,t1], где t – обозначение переменной, t1 – значение, при котором вычисляется решение:

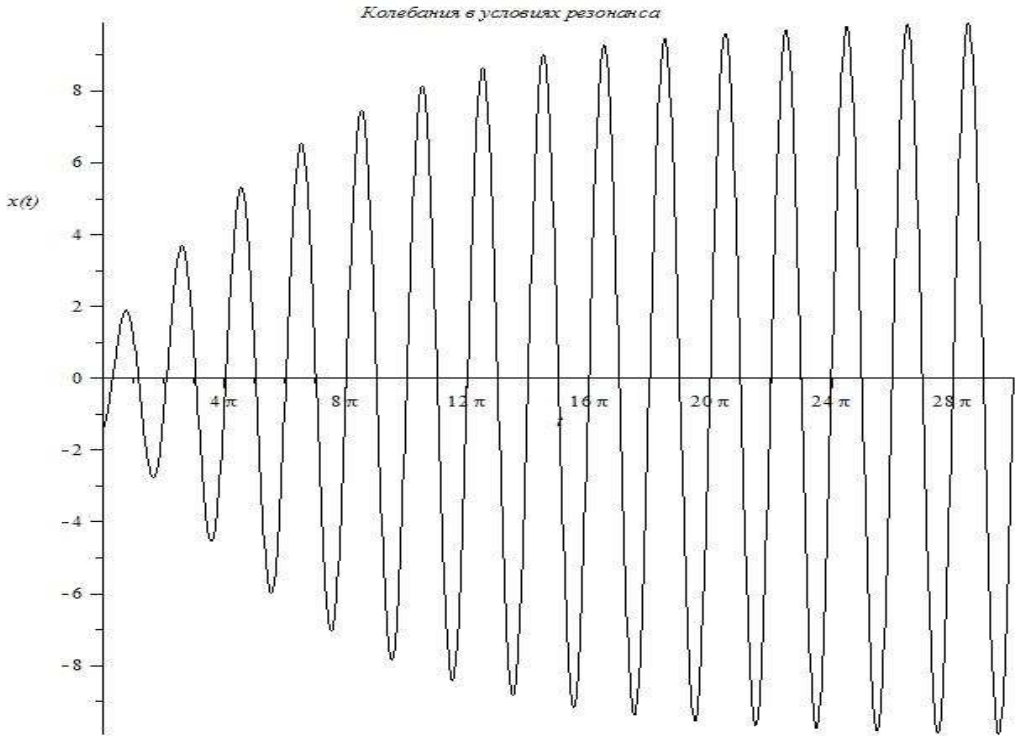
```
> CompMod[SolveOscillations]:=
> proc(x,t,f,omega0,beta0,InVal)
> local X,dX,F,T,T1,IN,S1,S2,SS1,SS2,T2:
> T2:=t[2]:
> X:=(T)->subs(t[1]=T,x(t[1])):
> dX:=(T)->subs(t[1]=T,diff(x(t[1]),t[1])):
> F:=(T)->subs(t[1]=T,f):
> IN:=(T)->subs(t[1]=T,CompMod[Inits](x,t[1],InVal)):
> S1:=(T)->dsolve({CompMod[EqOscillations](X,T,F(T),
> omega0,beta0),op(IN(T))},X(T)):
> S2:=(T1)->subs(T=T1,simplify(diff(S1(T),T))):
> SS1:=(T2)->subs(T=T2,S1(T)):
> SS2:=(T2)->subs(T=T2,S2(T)):
> [rhs(SS1(T2)),rhs(SS2(T2))]:
> end proc:
```

В этой процедуре: x – неизвестная функция (координата точки массы), t=[t,t1] – t – переменная t, t1 – ее значение, при котором находится решение, f(t)=F(t)/m – вынуждающая внешняя сила, точнее – ускорение, omega0

– частота собственных колебаний, beta0 – коэффициент линейного трения, InVal=[x0,v0] – начальные значения координаты и скорости массы. При этом решение выводится в списочном формате [x(t1),v(t1)].

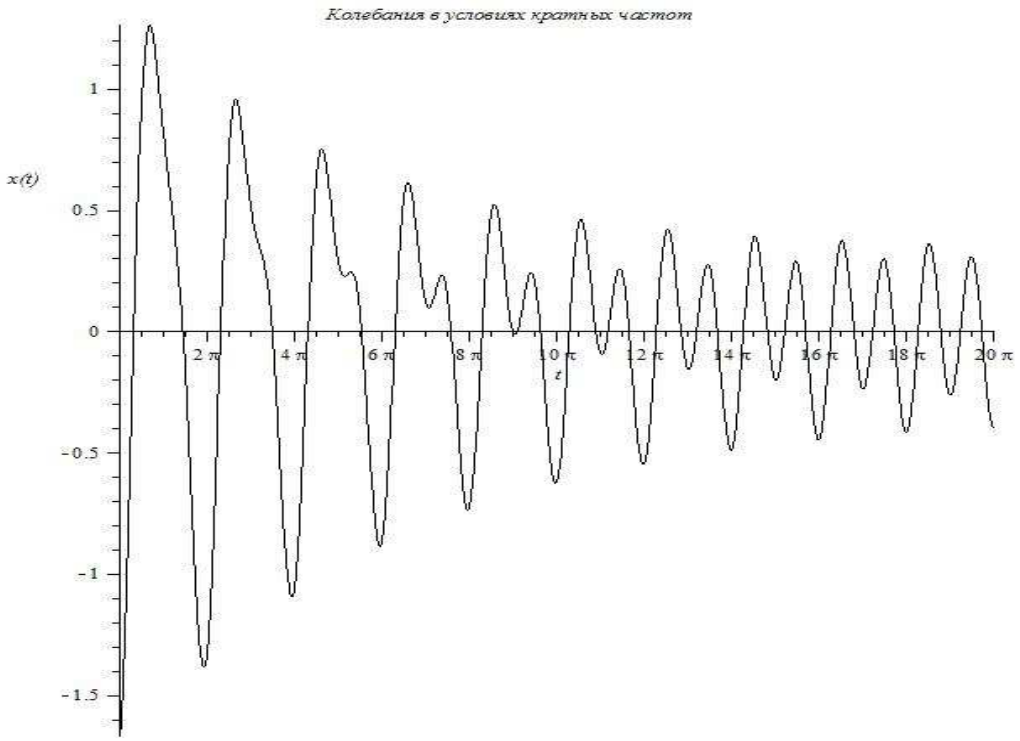
Пример 1.Рассмотрим пример резонанса:

```
> SSS:=(t1)->
> CompMod[SolveOscillations](xi,[tau,t1],cos(tau),1,0.1,[1,0,2]):

> SSS(1):

> plot(SSS(t)[1],t=0..30*Pi,numpoints=1000,
> color=black,labels=[t,‘x(t)‘],
> title=‘Колебания в условиях резонанса‘);
```

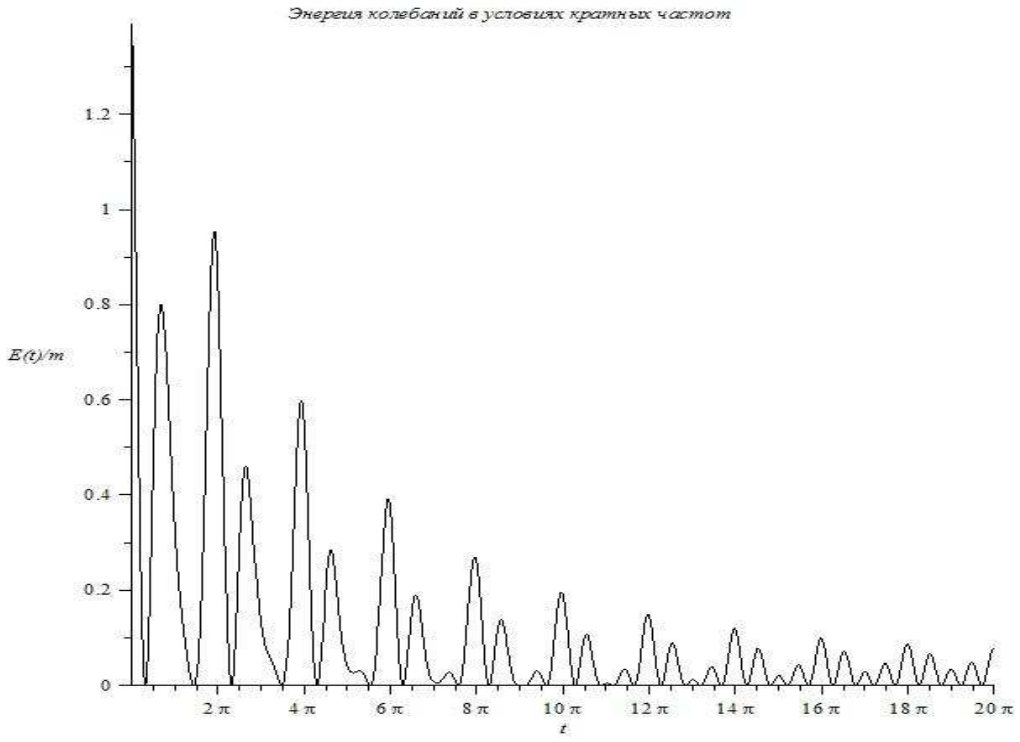


Пример 2. Рассмотрим пример двухкратной частоты вынуждающей силы:

```
> SSS1:=(t1)->CompMod[SolveOscillations](xi,
> [tau,t1],cos(2*tau),1,0.1,[1,0,2]);

> plot(SSS1(t)[1],t=0..20*Pi,numpoints=1000,
> color=black,labels=[t,‘x(t)‘],
> title=‘Колебания в условиях кратных частот‘);
```




```
> plot(SSS1(t)[1]^2/2,t=0..20*Pi,numpoints=1000,
> color=black,labels=[t,‘E(t)/m‘],
> title=‘Энергия колебаний в условиях кратных частот‘);
```



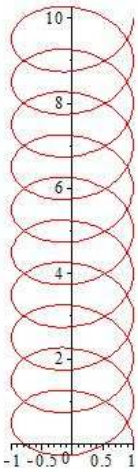
V.3 Программа тренажера линейных колебаний

Создадим инструменты для тренажера:

V.3.1 Пружина

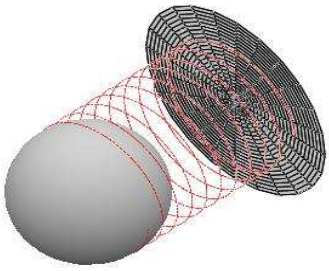
Создадим команду пружины длинной l, радиуса r, количеством витков n, цвета c с опциями h или v - h- горизонтальная, v - вертикальная:

```
> plot([cos(tt),10-sin(tt)-10/(2*Pi*10)*tt,
> tt=0..2*Pi*10],color=red);
```




```
> CompMod[Spring]:=proc(l,r,n,c,d)
> local VV,EQ,EQ_opora,tt,GG,opora,u,EQ_Shar,v,Shar:
> if d=h then
> EQ:=[l/(2*Pi*n)*tt,r*cos(tt),r*sin(tt)]:
> EQ_opora:=[0,u*cos(tt),u*sin(tt)]:
> EQ_Shar:=[l+r*sin(v),r*cos(u)*cos(v),r*sin(u)*cos(v)]:
> elif
> d=v then
> EQ:=[r*cos(tt),r*sin(tt),l-l/(2*Pi*n)*tt]:
> EQ_opora:=[u*cos(tt),u*sin(tt),l]:
> EQ_Shar:=[r*cos(u)*cos(v),r*sin(u)*cos(v),r*sin(v)]:
> end if:
> opora:=plot3d(EQ_opora,tt=0..2*Pi,
> u=0..1.5*r,scaling=CONSTRAINED,color=gray):
> GG:=plots[spacecurve](EQ,tt=0..2*Pi*n,
> color=c,scaling=CONSTRAINED,numpoints=100*n):
> Shar:=plot3d(EQ_Shar,u=0..2*Pi,
> v=-Pi/2..Pi/2,style=PATCHNOGRID,
> shading=ZGRAYSCALE,scaling=CONSTRAINED):
> plots[display](opora,GG,Shar):
> end proc:
```

```
> CompMod[Spring](5,2,10,red,h);
```



V.4 Компьютерная модель нелинейных колебаний

Зададим теперь потенциал в виде полинома четвертого порядка четных степеней по x:

```
> U:=(x,k,b)->b^2/2*(x^2-k/(2*b^2))^2;
```

$$U := (x,k,b) \mapsto 1/2\, b^2 \left(x^2 - 1/2\, \frac{k}{b^2} \right)^2$$

Тогда функция силы, $F(x) = -dU/dx$ равна:

```
> F:=(x,k,b)->-diff(U(x,k,b),x);
```

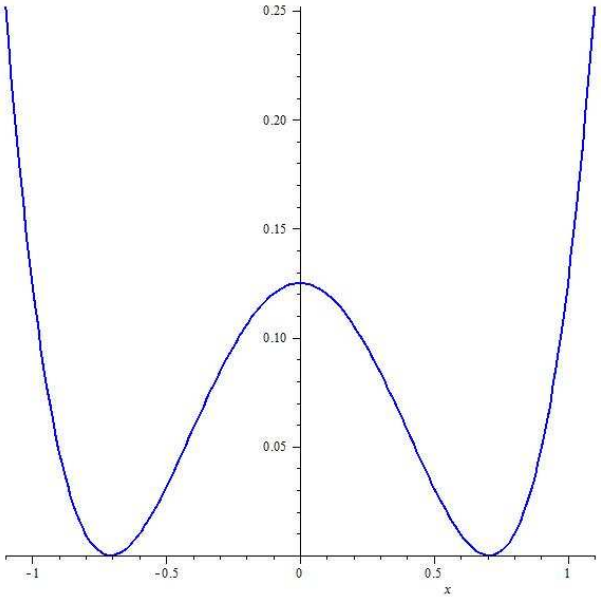
$$F := (x,k,b) \mapsto -2\, b^2 \left(x^2 - 1/2\, \frac{k}{b^2} \right) x$$

```
> F(x,k,b);
```

$$-2\, b^2 \left(x^2 - 1/2\, \frac{k}{b^2} \right) x$$

Сила является нечетной по x функцией и описывается полиномом третьего порядка. Это простейший вид нелинейной силы упругости. Такой вид силы приводит к различным нелинейным явлениям, в том числе к спонтанному нарушению симметрии, заключающегося в нарушении симметрии колебательной системы в результате действия диссипативных сил.

```
> plot(U(x,1,1),x=-1.1..1.1);
```




```
> Eq_n:=(m,k,b,beta0)->diff(x(t),t$2)-
> subs(x=x(t),F(x,k,b)/m)+beta0*diff(x(t),t)=0:

> Eq_n(m,k,b,beta0);
```

$$\frac{d^2}{dt^2}x\left(t\right)+2\,b^2\left(\left(x\left(t\right)\right)^2-1/2\,\frac{k}{b^2}\right)x\left(t\right)m^{-1}+beta0\,\frac{d}{dt}x\left(t\right)=0$$

```
> expand(%);
```

$$\frac{d^2}{dt^2}x\left(t\right)+2\,\frac{b^2\left(x\left(t\right)\right)^3}{m}-\frac{x\left(t\right)k}{m}+beta0\,\frac{d}{dt}x\left(t\right)=0$$

```
> IC:=x(0)=0,D(x)(0)=1;
```

$$IC:=x\left(0\right)=0,\,\mathrm{D}\left(x\right)\left(0\right)=1$$

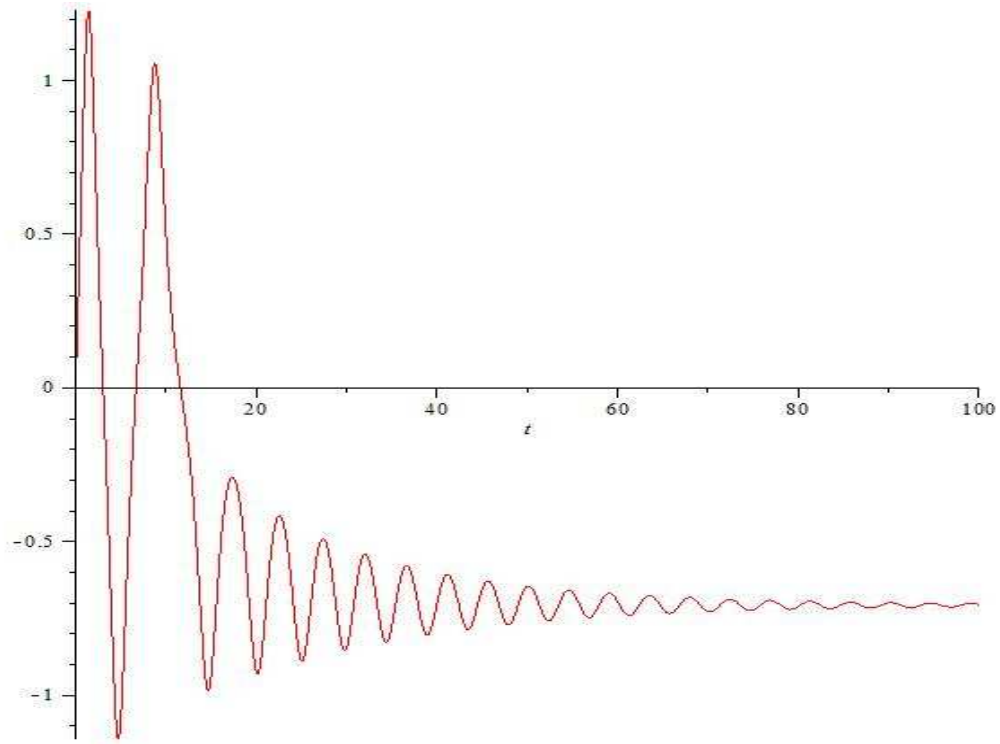
```
> S1:=dsolve({Eq_n(1,1,1,0.1),IC},x(t),
> type=numeric,output=listprocedure):

> S1_1_1_0_1:=subs(S1,x(t));
```

$$S1\_1\_1\_0\_1:=\mathrm{proc}\left(t\right)\ldots\mathrm{endproc}$$

```
> S1_1_1_0_1(1);
```

$$1.012310420639\,06$$

```
> plot(S1_1_1_0_1(t),t=0..100,numpoints=5000);
```

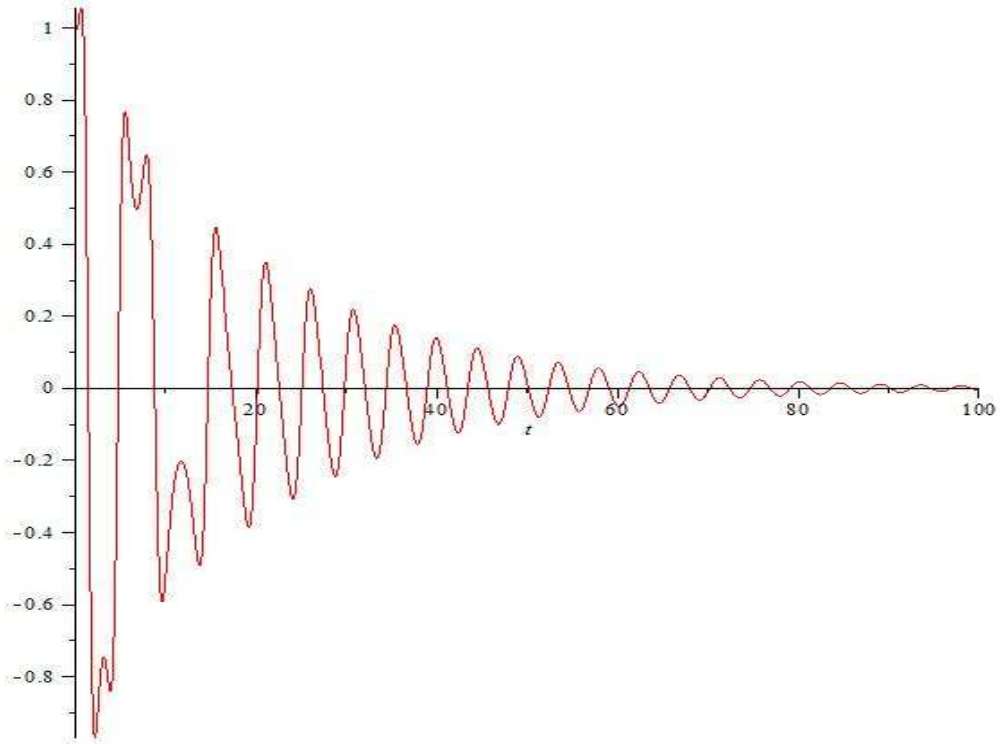



```
> V1_1_1_0_1:=subs(S1,diff(x(t),t));
> V1_1_1_0_1(1);
```

$$V1\_1\_1\_0\_1 := \mathrm{proc}\,(t) \ldots \mathrm{endproc}$$

$$0.877150673964212$$

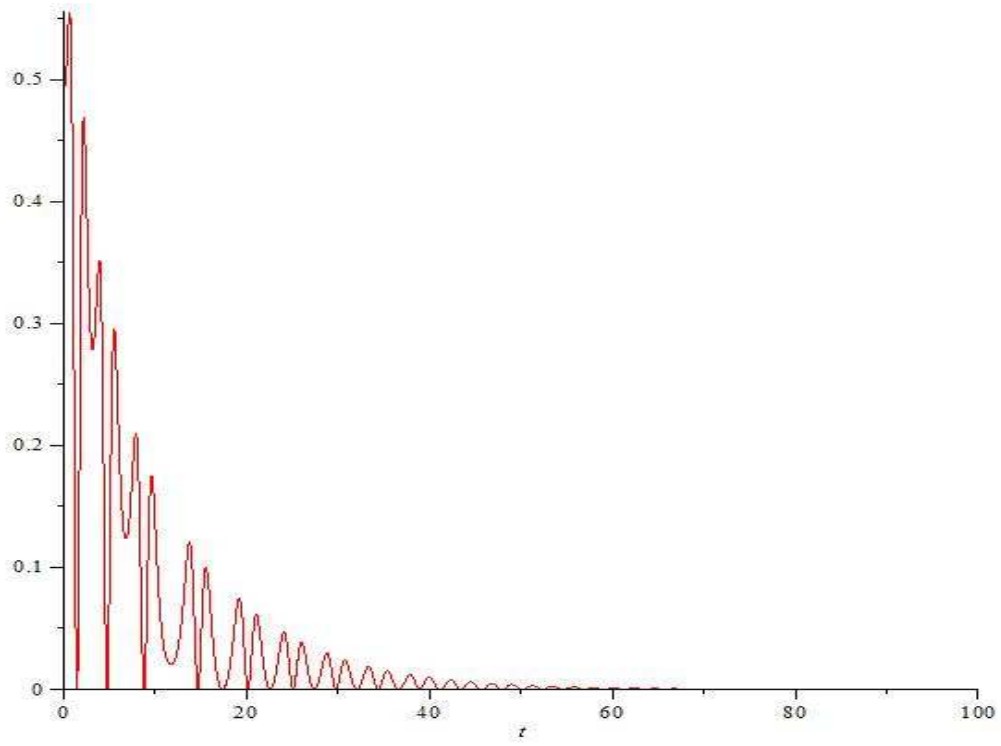
```
> plot(V1_1_1_0_1(t),t=0..100,numpoints=5000);
```




```
> E1:=V1_1_1_0_1(t)^2/2;
```

$$E1 := 1/2\ (V1\_1\_1\_0\_1\ (t))^2$$

```
> plot(E1(t),t=0..100,numpoints=5000);
```




```
> IC2:=x(0)=0,D(x)(0)=10;
```

$$IC2 := x\,(0) = 0,\ \mathrm{D}\,(x)\,(0) = 10$$

```
> S2:=dsolve({Eq_n(1,1,1,0.1),IC2},x(t),
> type=numeric,output=listprocedure);
```

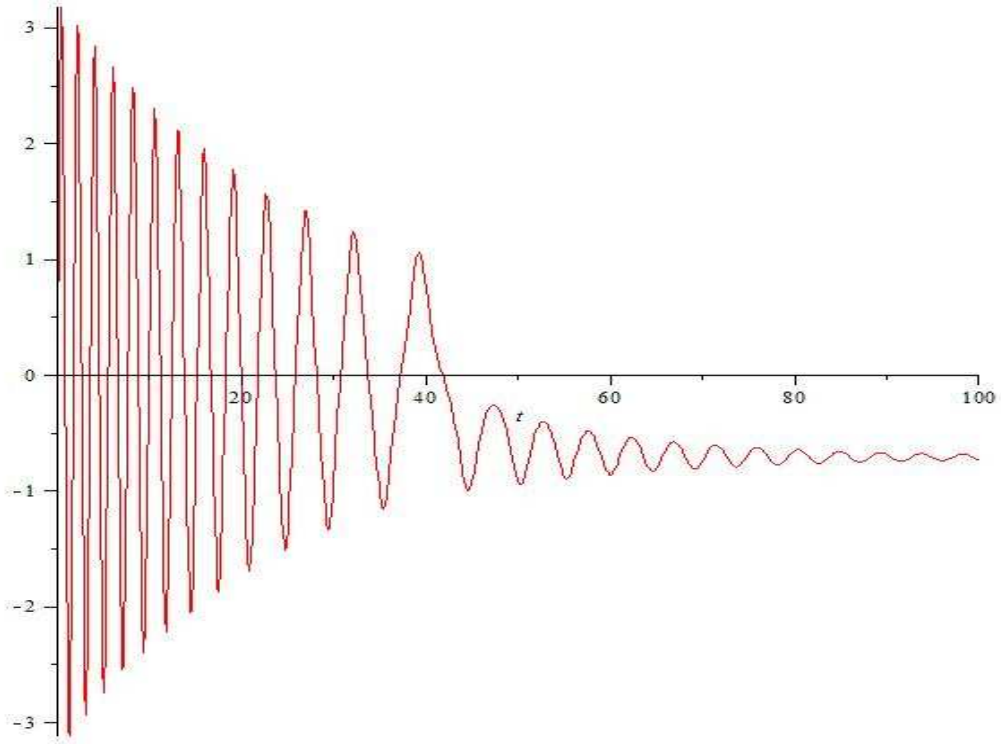
```
> S2_1_1_1_0_1:=subs(S2,x(t));
```

$$S2\_1\_1\_1\_0\_1 := \mathrm{proc}\,(t) \ldots \mathrm{endproc}$$

```
> S2_1_1_1_0_1(1);
```

$$-1.39545993174939$$

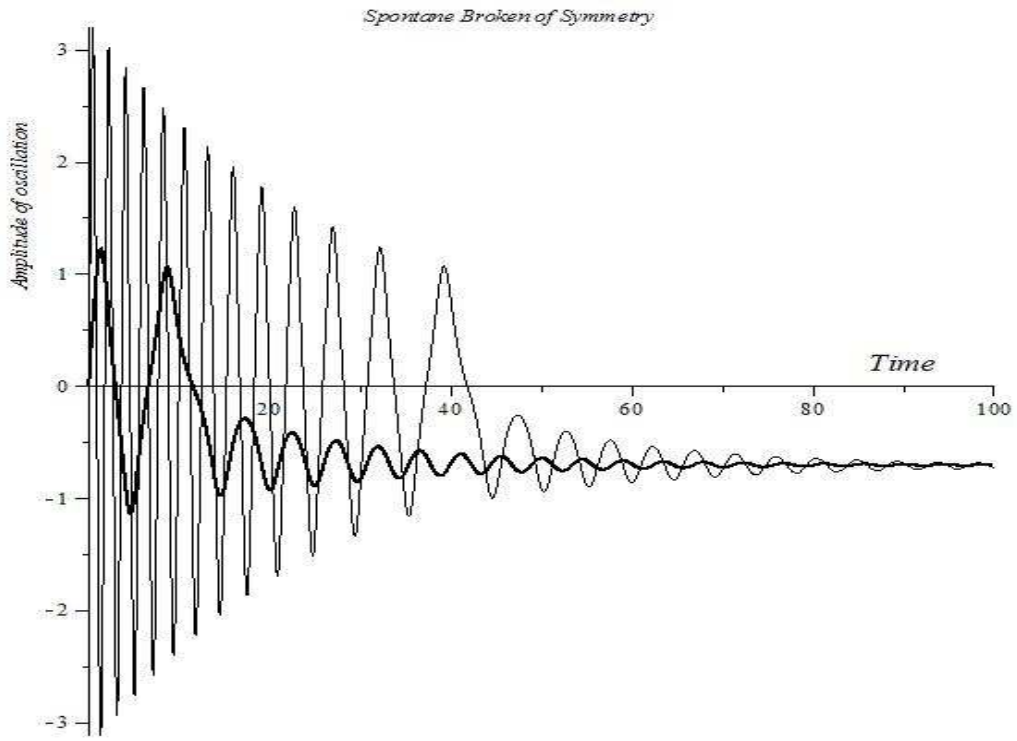
```
> plot(S2_1_1_1_0_1(t),t=0..100);
```



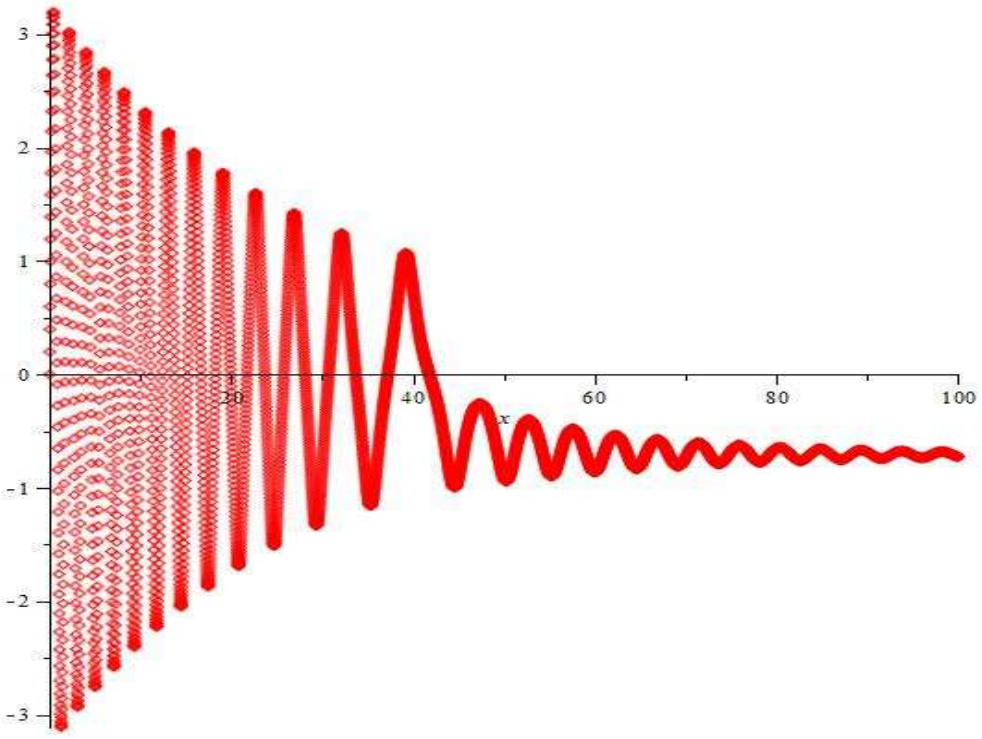

```
> G1:=plot([S1_1_1_0_1(t),S2_1_1_1_0_1(t)],
> t=0..100,numpoints=5000,thickness=[2,1],color=black,
> labels=[‘‘,‘Amplitude of oscillation‘],
> labeldirections=[HORIZONTAL,VERTICAL],title=
> ‘Spontane Broken of Symmetry‘):
```

```
> Txt:=plots[textplot]([90,0.2,‘Time‘],font=[TIMES,ROMAN,14]):
```

```
> plots[display](G1,Txt);
```

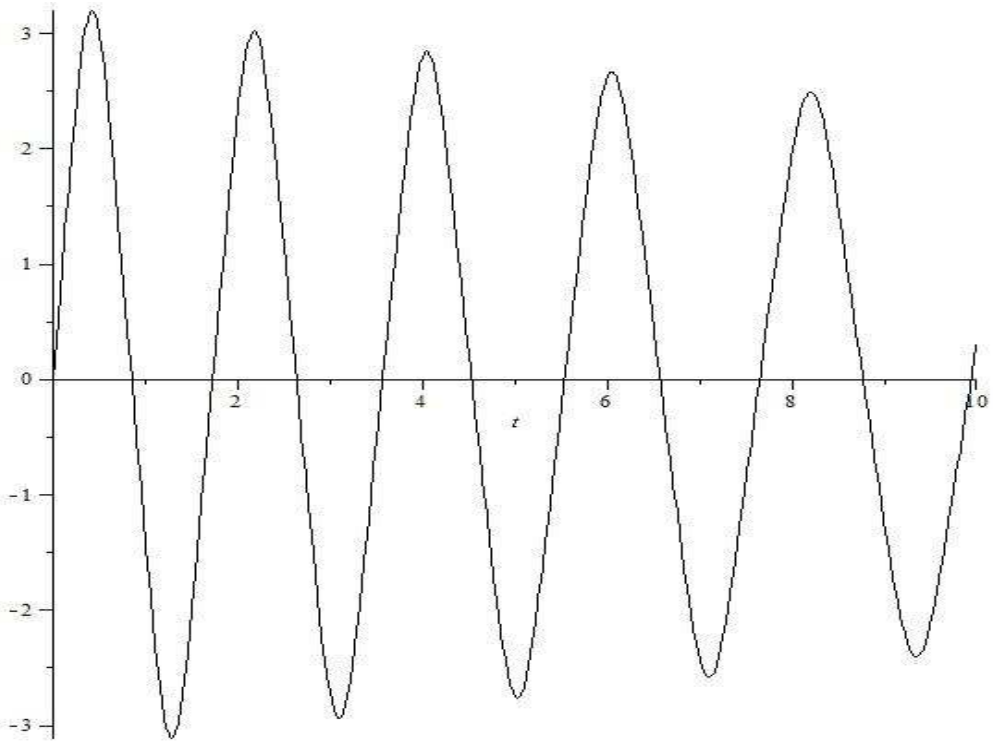



```
> plots[animate](S2_1_1_1_0_1(x*t),x=0..100,t=0..1,style=POINT,
> numpoints=
> 5000,frames=32);
```

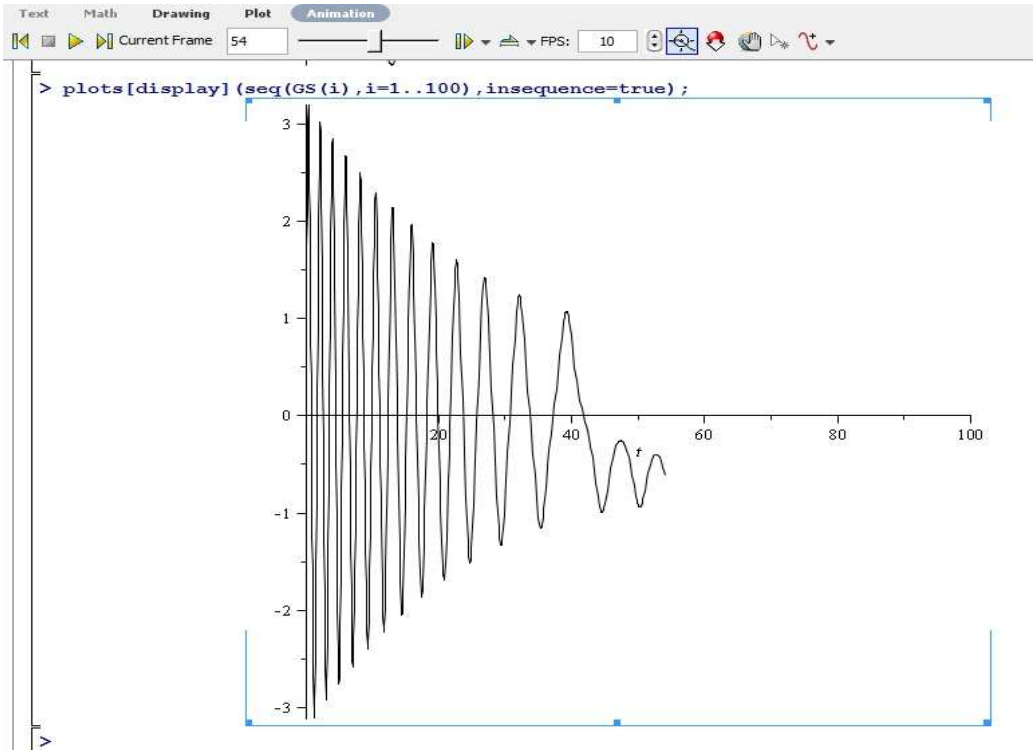



```
> GS:=(T)->plot(S2_1_1_1_0_1(t),t=0..T,color=black);
```

```
> GS(10);
```

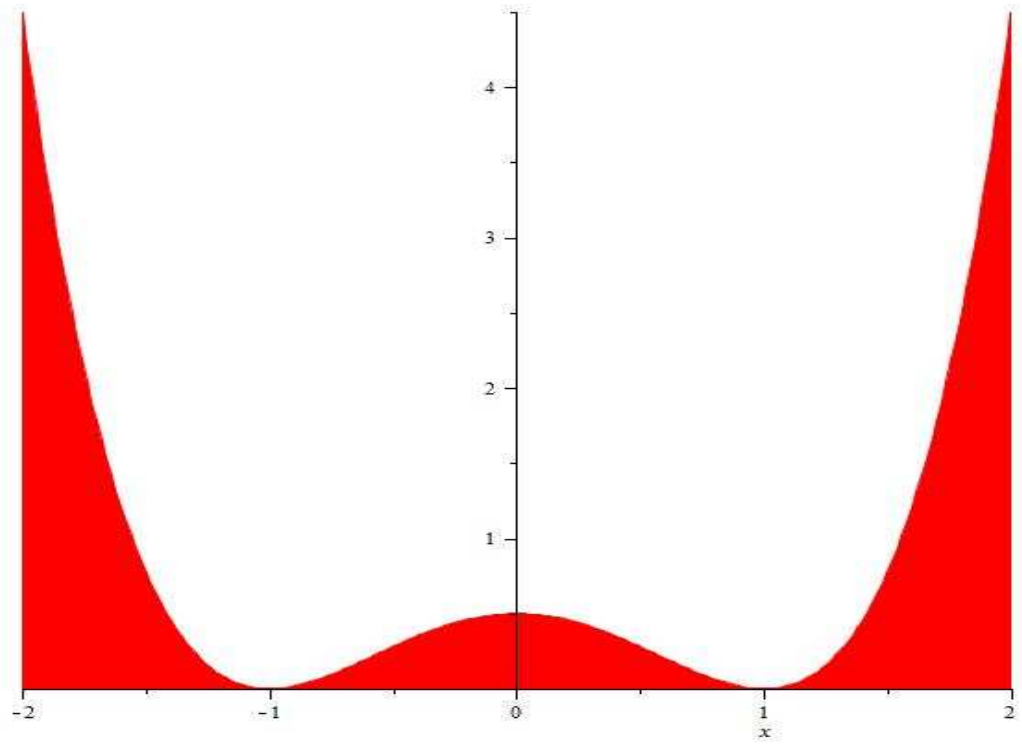



```
> plots[display](seq(GS(i),i=1..100),insequence=true);
```



V.5 Создание процедур анимации для демонстраций

```
> UG:=plot(U(x,2,1),x=-2..2,filled=true,color=red):
> UG;
```




```
> IC3:=x(0)=0,D(x)(0)=1;
```

$$IC3 := x(0) = 0,\ \mathrm{D}(x)(0) = 1$$

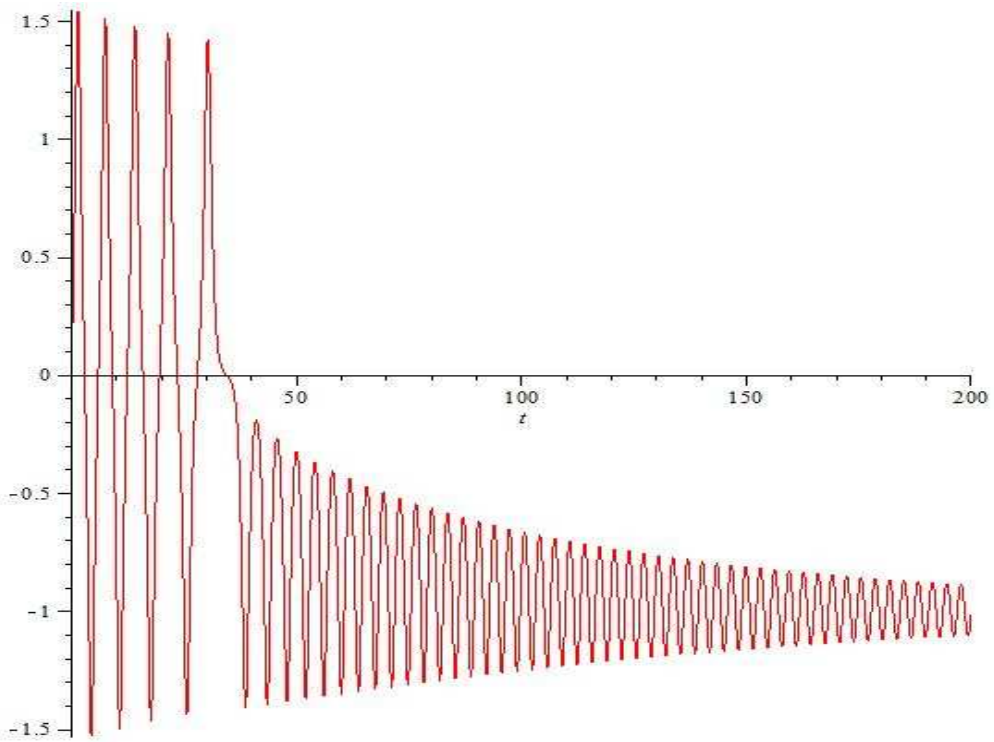
```
> S3:=dsolve({Eq_n(1,2,1,0.02),IC3},x(t),
> type=numeric,output=listprocedure);
```

$$S3\_1\_2\_0\_02 := \mathrm{proc}(t) \ldots \mathrm{endproc}$$

```
> S3_1_2_0_02(1);
```

$$1.20632747147795$$

```
> plot(S3_1_2_0_02(t),t=0..200,numpoints=5000);
```




```
> V:=subs(S3,diff(x(t),t));
```

$$V := \mathrm{proc}\,(t) \ldots \mathrm{endproc}$$

```
> E:=(t)->V(t)^2/2+U(S3_1_2_0_02(t),2,1);
```

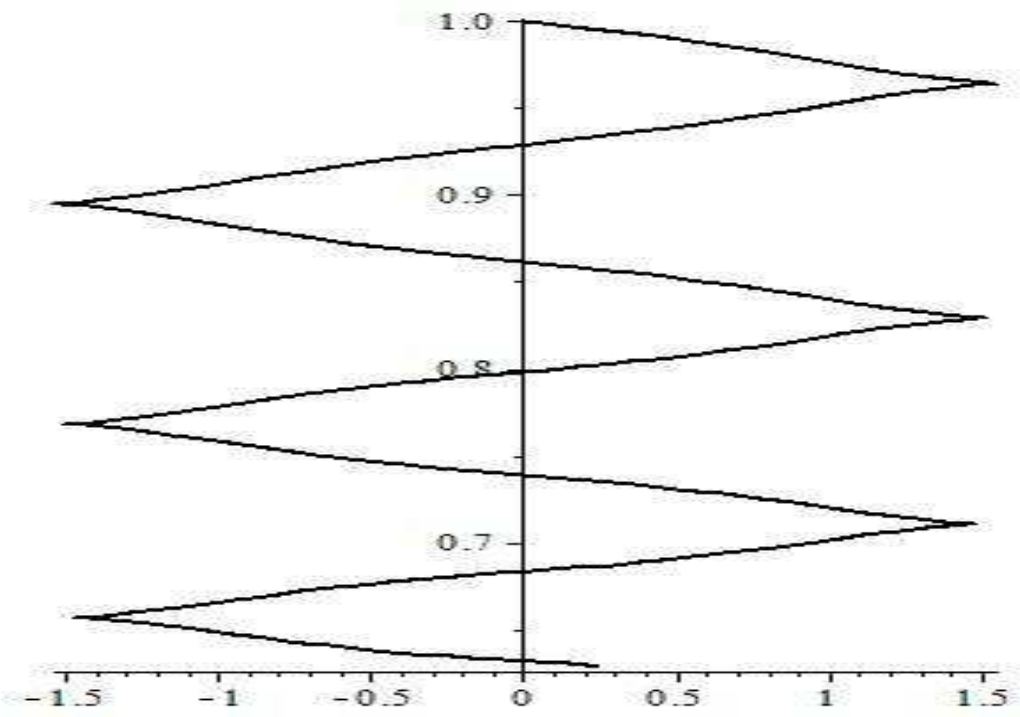
$$E := t \mapsto 1/2\,(V\,(t))^2 + 1/2\,\left((S3\_1\_2\_0\_02\,(t))^2 - 1\right)^2$$

```
> E(1);
```

$$0.970469914859435$$

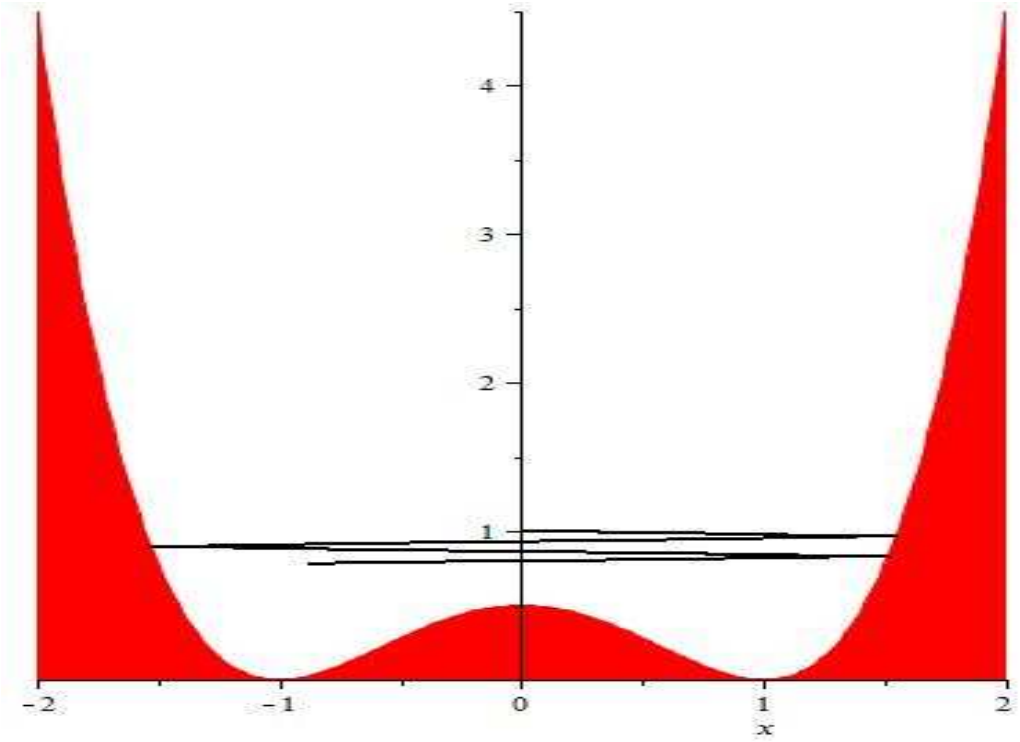
```
> GG:=(T)->plot([S3_1_2_0_02(t),E(t),t=0..T],
> color=black,thickness=2);
```

```
> GG(20);
```

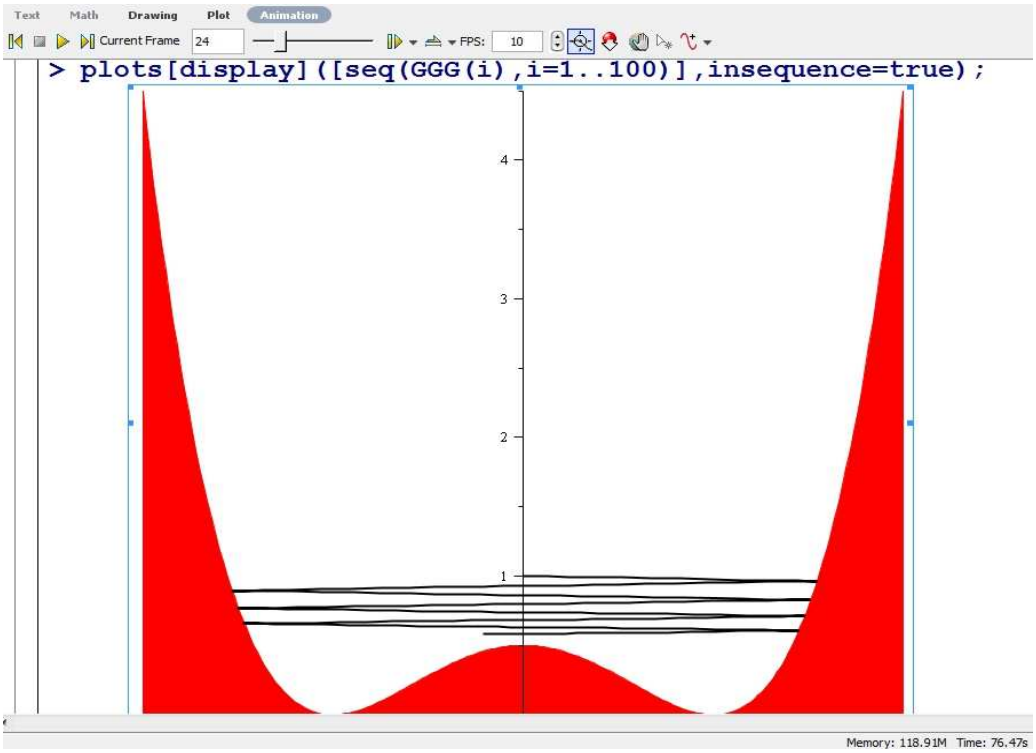



```
> GGG:=(t)->plots[display](UG,GG(t));
```

```
> GGG(10);
```




```
> plots[display]([seq(GGG(i),i=1..100)],insequence=true);
```



V.6 Математическое моделирование нелинейных обобщенно - механических систем в системе компьютерной математики Maple

V.6.1 Введение

Уникальные графические возможности системы компьютерной математики (СКМ) Maple, в частности, возможности создания трехмерных анимационных моделей, хорошо проработанные программные процедуры численного интегрирования систем обыкновенных дифференциальных уравнений (ОДУ), сплайновой и B - сплайновой интерполяции функций позволяют рассматривать СКМ Maple в качестве мощного современного инструмента математического моделирования [9], [4], [6], [8]. В настоящее время методы математического моделирования в СКМ начали эффективно применяться в исследованиях математических моделей как фундаментальных физических явлений, так и прикладных задач. В частности, монография Д.П. Голоскокова [4] целиком посвящена проблемам математического моделирования в СКМ Maple объектов математической физики – физических полей, гидродинамических процессов, процессов теплопереноса и диффузии; в фун-

даментальной монографии В.П. Дьяконова [6] обширная глава посвящена применению СКМ Maple в математическом моделировании, в частности, моделированию электронных схем и измерительных систем на основе эффекта Доплера; монография М.Н. Кирсанова [8] содержит материалы по математическому моделированию сложных механических систем со связями и визуализации математических моделей этих систем. В [20], [22], [23], [24], [26] – методы математического моделирования с помощью СКМ Maple успешно применяются для решения весьма сложных задач релятивистской кинетики, теории гравитации и космологии ранней вселенной. Важным преимуществом СКМ Maple является и возможность интеграции этой системы с СКМ MatLab, которая приспособлена к моделированию электронных систем и технологических процессов.

Объектом исследования этого раздела является математическое моделирование нелинейных обобщенно-механических систем, (НОМС), в среде компьютерной математики Maple [98, 99, 100]. Такие системы в наиболее общем случае описываются системой нелинейных ОДУ, разрешенных относительно старших производных функций $y_i(t)$, вида:

$$y_i^{(n_i)} = F_i(y_1, ..., y_N, y_1', ..., y_N', y_1'', ..., y_N'', ..., y_1^{(n_i-1)}, t); \quad (i = \overline{1, N}), \qquad \text{(V.5)}$$

где y(n) =dnf/dtn – обозначение n-той производной функции f по независимой переменной t, – времени, а Fi – непрерывно-дифференцируемые функции своих переменных. В большинстве случаев n=2, однако, например, при рассмотрении движения релятивистской заряженной частицы в магнитном поле с учетом магнито-тормозного излучения n=3 (см., например, [32]); а в ряде случаев n может достигать и значения 4. К случаю n=3 сводится, например, и важная задача геодезии ориентирования на местности - задача о восстановлении кривой по ее натуральным уравнениям, т.е., по заданным функциям кривизны и кручения (см., например, [35]). В дальнейшем к обобщенно-механическим системам мы будем относить в дальнейшем любые системы, которые полностью описываются уравнениями вида (V.5)[1]. К таким системам могут относиться, в частности, и геометро - оптические системы, в которых фотон может рассматриваться как материальная точка, движущаяся по траектории, сложные электронные схемы и электрические сети, нейронные сети, гомогенные химические системы, в которых протекают химические реакции и т.п. Будем в дальнейшем полагать выполненными начальные условия для системы (V.5):

[1] В дальнейшем такие системы для простоты будем также называть механическими.

$$y_i^{(k)}(t)\Big|_{t=t_0} = C_i^k \ ; \quad (k = \overline{1, n_i - 1}; \ i = \overline{1, N}), \tag{V.6}$$

соответствующие стандартной задаче Коши, где C_i^k- начальные значения производных k-го порядка функций yi(t).

Как известно, достаточно эффективных и общих методов аналитического исследования поведения НОМС, описываемых задачей Коши (V.5)-(V.6), не существует. Применение методов качественной теории дифференциальных уравнений требует, во-первых, автономность системы(V.5), а, во-вторых, с увеличением числа степеней свободы, S, системы (V.5):

$S = \sum_{i=1}^{N} n_i$ (III.3) сложность исследования с помощью качественной теории дифференциальных уравнений, а тем более, их визуализации, резко возрастает при S>2 [2]. Фактически, единственным надежным методом исследования нелинейных механических систем является численное решение задачи Коши, которое сводится обычно к численному интегрированию нормальной системы ОДУ, соответствующей системе (V.5) с соответствующими начальными условиями, полученными из (V.6). Необходимость предварительного преобразования системы (V.5) к нормальному виду вызвано высокой степенью разработанности теории и численных методов именно для нормальных систем ОДУ. Необходимость применения достаточно сложных численных методов, связанная с этим обстоятельством необходимость профессионального программирования, сложность манипуляций с численными решениями, в частности, сложности визуализации динамики нелинейных механических систем являются совокупным фактором, резко ограничивающим область исследуемых нелинейных механических систем, как математикам, так и специалистам в приложениях математики, не являющихся профессиональными программистами. Системы компьютерной математики, в принципе, заметно приближают таких специалистов к применению методов компьютерного моделирования, но все же и здесь применительно к исследованию систем нелинейных ОДУ для таких специалистов сохраняется заметная диспропорция между затраченными усилиями и получением результата. Кроме того, при прямом применении программных процедур СКМ, по-прежнему, остаются слабыми возможности проведения компьютерного моделирования систем с большим числом степеней свободы и параметров.

Целью нашего исследования является разработка алгоритмов и пакетов (комплекса) программ в СКМ Maple для компьютерного исследования нелинейных обобщенно-механических систем и построения многопараметриче-

[2] см., например, [36], [37].

ских программных процедур визуализации математических моделей этих систем, в том числе, и процедур динамической визуализации (анимации) математических моделей НОМС. При этом ставились задачи: 1). достижения максимальной внешней простоты программных процедур для непрофессиональных в программировании пользователей; 2). предоставления пользователю возможности управления процессом численного интегрирования; 3). возможности вывода численных решений и манипуляции ими в формах, наиболее близких к функциональным, 4). достижения максимальной скорости вычислений при сохранении их точности; 5). разработки методов и инструментов визуализации и анимации математических моделей НОМС. В рамках данного раздела будет описано решение первых четырех задач, - решению пятой задачи посвящены другие разделы. Отметим, что важной идеей построения пакета программ явилась идея использования сплайновой и В - сплайновой интерполяции функций, позволившей создать автоматизированный вывод численных решений ОДУ в форме кусочно-заданных функций.

V.6.2 Блок-схема комплекса программ

Для обеспечения гибкой работы с численными решениями были созданы специальные внутренние программные процедуры, позволяющие проводить стандартные операции анализа функции одной переменной со сплайнами, а также конвертировать их в кусочно-заданные функции. В результате был получен программный аппарат аналитического (приближенного) исследования НОМС в СКМ Maple [113][3]. Важным достоинством разработанного комплекса программ является его независимость от версии Maple, начиная с версии 1997 года Maple5.5 и кончая версией 2013 года Maple 17. Представленный комплекс программ состоит из двух независимых пакетов программ.

Пакет программ DifEq содержит программы распознавания введенных системы дифференциальных уравнений и начальных условий (a), программы автоматического преобразования введенных данных в задачу Коши для нормальной системы относительно унифицированных переменных, (b), программы численного решения задачи Коши на заданном интервале с возможностью контроля и переключения метода численного интегрирования при заданном значении независимой переменной, (c). Пакет программ Splines содержит программы генерации равномерных кубических сплайнов и B-сплайнов по заданной функции на заданном интервале, (d,f), программы конвертирования сплайнов и B-сплайнов в кусочно-непрерывные функции и

[3]При использовании программного комплекса ссылка на авторство обязательна.

обратные операции, (e,g), программы операций над сплайнами (h). Операции указанных пакетов интегрируются в программе конвертирования численных решений в кусочно-заданные функции, (i).Ниже мы опишем основные алгоритмы разработанного комплекса программ и продемонстрируем на примерах исполнение программных процедур.

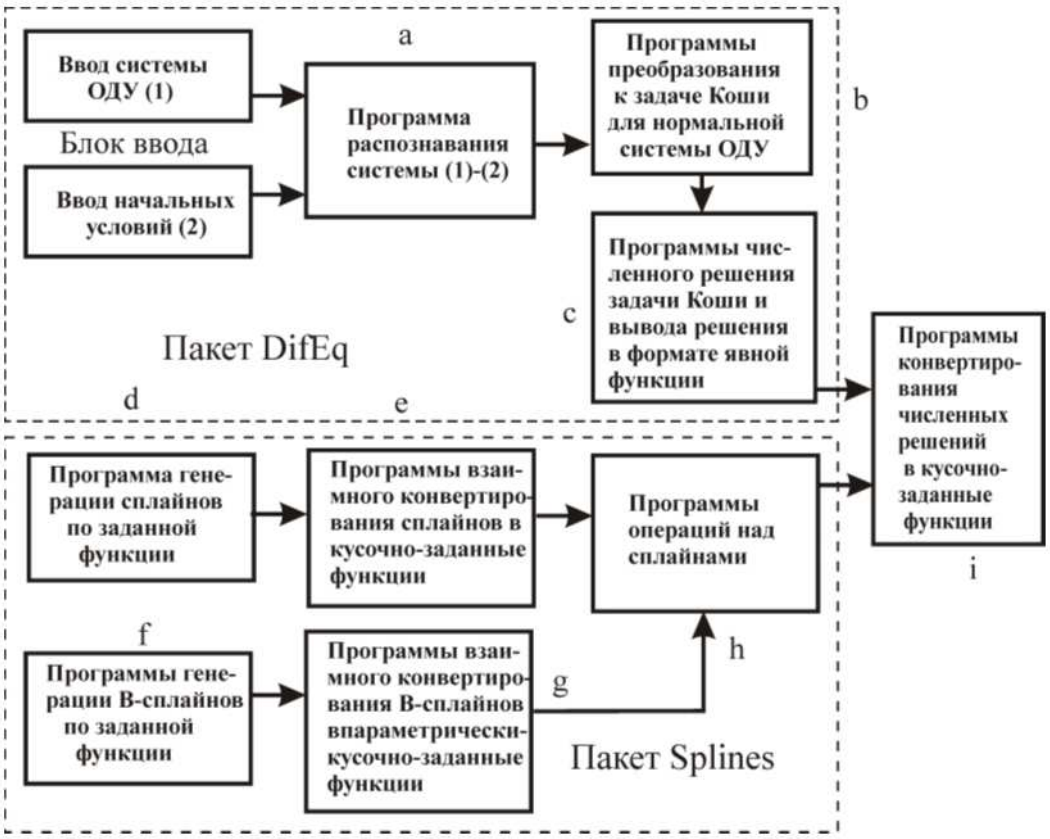


Рис.V.57 *Блок-схема комплекса программ исследования нелинейных обобщенно механических систем.*

V.7 Пакет программ преобразования системы уравнений и решения задачи Коши

V.7.1 Распознавание информации об ОДУ (блок a)

Рассмотрим ОДУ n-го порядка, разрешенного относительно старшей производной функции y(x):

$$y^{(n)} = F\left(x, y, y', ..., y^{(n-1)}\right) y^{(n)} = F\left(x, y, y', ..., y^{(n-1)}\right). \qquad \text{(V.7)}$$

Для задания n-той производной функции y(x) по переменной x в СКМ Maple существуют две альтернативные процедуры: diff(y(x),x$n) и (D@@n)(y)(x). Вторая из этих процедур предусматривает возможность вычисления производной в заданной точке, что весьма удобно для задания начальных условий. Создадим программную процедуру DifEq[DiffOp], позволяющую извлечь полную информацию о старшей производной в правой части уравнения (V.7)[4]:

```
[> DifEq[DiffOp]:=proc(X) local var,s,i,ss,ff,tt:
if op(0,X)=diff then: var:=op(X)[2];s:=op(X)[1];
for i from 1 do
if(nops(s)=1) then
ff:=(op(0,s)); tt:=(op(1,s)); break:
else s:=op(s)[1]; end if; end do:
[diff,i,ff,tt]:
elif op(1,op(0,op(0,X)))=D then
[D,op(2,op(0,op(0,X))),
op(-1,op(0,X)),op(-1,X)]:
elif op(0,op(0,X))=D then
[D,1,op(-1,op(1,op(0,X))),op(-1,X)]:
else ff:= op(0,X):tt:=op(-1,X): [0,ff,tt]:
end if:end proc:
```

Продемонстрируем исполнение этой процедуры на двух примерах:

```
[> DifEq[DiffOp]((D@@4)(F)(tau));
```

[D, 4, F, τ]

```
[> DifEq[DiffOp](diff(Sigma(zeta),zeta));
```

[*diff*, 1, Σ, ζ]

Таким образом, при действии на производную данная программная процедура создает упорядоченный список - тип оператора производной, порядок производной, имя неизвестной функции и имя независимой переменной. Это позволяет перейти в дальнейшем от обозначений пользователя к некоторым унифицированным внутренним именам зависимых и независимых переменных, что позволяет автоматизировать процессы работы с уравнениями.

[4] Идею этого алгоритма подсказал А.В. Матросов, которому Ю.Г. Игнатьев выражает признательность.

V.7.2 Приведение задачи Коши для системы ОДУ произвольного порядка к задаче Коши для нормальной системы ОДУ с унифицированными именами переменных (блок b)

Блок (b) включает три программные процедуры, осуществляющие поэтапное конвертирование системы ОДУ (V.5) с начальными условиями (V.6) к задаче Коши для нормальной системы ОДУ с унифицированными именами переменных. Знание информации о старшей производной уравнения (V.7) позволяет конвертировать это уравнение к нормальной системе ОДУ с унифицированными обозначениями, что обеспечивает автоматизацию манипуляций с этими уравнениями. Договоримся о следующем порядке присвоения унифицированных имен зависимых переменных. Для этого рассмотрим упорядоченный список DifEq[MatAlf] из восьми имен: [X,Y,Z,U,V,W,Phi,Theta], который, конечно, можно продолжить, но для этого нет никаких причин практического характера. Обозначим переменную, введенную пользователем, посредством X, ее первую производную – Y, вторую производную - Z, и т.д., а независимую переменную посредством t. Полученная система и будет являться искомой нормальной системой ОДУ с унифицированными именами переменных:

$$\frac{dX_i^1}{dt} = X_i^2;\ \ \frac{dX_i^2}{dt} = X_i^3; ...;\ \ \frac{dX_i^{n_i-1}}{dt} = F_i,\ \ (i = \overline{1,N}) \qquad \text{(V.8)}$$

где $F = F(t, X_1^1, ..., X_N^{n_N})$ и для краткости положено $X^k = \text{DifEq[MatAlf]}_k \to X^1 = X, X^2 = Y, \ldots$. и системой начальных условий для этих уравнений, соответствующих начальным условиям (V.6) :

$$X_i^k(t_0) = C_i^k\ ;\ \ (k = \overline{1, n_i - 1};\ \ i = \overline{1,N}), \qquad \text{(V.9)}$$

V.7.3 Приведение ОДУ произвольного порядка к нормальной системе ОДУ с унифицированными именами переменных

Первый из указанных алгоритмов – алгоритм приведения ОДУ произвольного порядка к нормальной системе ОДУ ОДУ произвольного порядка к нормальной системе ОДУ реализуется программной процедурой DifEq[Oden_ConvNorm]:

```
[> DifEq[DiffOp]:=proc(X) local var,s,i,ss,ff,tt:
if op(0,X)=diff then: var:=op(X)[2];s:=op(X)[1];
for i from 1 do
```

```
if(nops(s)=1) then
ff:=(op(0,s)); tt:=(op(1,s)); break:
else s:=op(s)[1]; end if; end do:
[diff,i,ff,tt]:
elif op(1,op(0,op(0,X)))=D then
[D,op(2,op(0,op(0,X))),
op(-1,op(0,X)),op(-1,X)]:
elif op(0,op(0,X))=D then
[D,1,op(-1,op(1,op(0,X))),op(-1,X)]:
else ff:= op(0,X):tt:=op(-1,X): [0,ff,tt]:
end if:
end proc:
```

Как видно, процедура DifEq[Oden_ConvNorm] использует две промежуточные процедуры, DifEq[DiffOp] и DifEq[MatAlf]. Рассмотрим исполнение этой процедуры на примере уравнения в обозначениях пользователя:

$$S^{IV}(\xi) = \left(S^{III}(\xi)\right)^2 S^3(\xi) + S'(\xi) + \sin\xi. \tag{V.10}$$

Пользователь может ввести это уравнение в такой, например, форме:

```
[>Eq1:=(D@@4)(S)(xi)=((D@@3)(S)(xi))^2*S(xi)^3+D(S)(xi)+sin(xi);
```

Результат применения нашей процедуры к уравнению (V.10) получается следующим:

```
[>DifEq[Oden_ConvNorm]:=proc(EQ,k)
local lsd,rsd,ls,dd,nn,ff,xx,zz,i,sbs:
lsd:=lhs(EQ):rsd:=rhs(EQ):ls:=DifEq[DiffOp](lsd);
dd:=op(1,ls):nn:=op(2,ls):ff:=op(3,ls):xx:=op(4,ls):
zz:=op(nn,DifEq[MatAlf]):
if dd=D then
sbs:=(t)->[ff(xx)=X[k](t),seq((dd@@i)(ff)(xx)=
(DifEq[MatAlf][i+1])[k](t),i=1..nn-1),xx=t]:
elif dd=diff then
sbs:=(t,k)->[ff(xx)=X[k](t),seq(diff(ff(xx),xx$i)=
op(i,DifEq[MatAlf])[k](t),i=1..nn-1),xx=t]:
end if:
[[nn,ff,xx,dd],[seq(op(i,DifEq[MatAlf])[k],i=1..nn)],
sbs(t,k),[seq(diff((DifEq[MatAlf][i])[k](t),t)=
(DifEq[MatAlf][i+1])[k](t),i=1..nn-1),
diff(zz[k](t),t)=subs(sbs(t,k),rsd)]]:
```

```
end proc:
```

- он состоит из упорядоченного списка четырех упорядоченных списков: первый - содержит: [порядок уравнения, искомую функцию, независимую переменную, тип дифференциального оператора], введенные пользователем. Второй список содержит упорядоченную систему новых функций. Третий список содержит правила замены переменных в ОДУ, четвертый список содержит упорядоченную нормальную систему ОДУ. Как видно, процедура DifEq[Oden_ConvNorm] содержит два обязательных параметра, первый – дифференциальное уравнение, а второй, k, – любое имя или число, необходимое для пометки уравнения и входящих в него переменных. Соответствующая метка помещается как нижний индекс у соответствующих величин. В рассмотренном примере k=2. В частности, если мы хотим отказаться от индексирования функций, достаточно вместо индекса ввести " - два апострофа.

V.7.4 Приведение системы ОДУ произвольного порядка к нормальной системе ОДУ с унифицированными именами переменных

Аналогично вводится и однопараметрическая процедура приведения системы ОДУ (V.5) к нормальной системе ОДУ DifEq[SysOden_ConvNorm]

```
> DifEq[SysOden_ConvNorm]:=proc(SysEq)
local NN,i,k,FF,EQI,ParEq,nn,NP,ff,fff,XX,xx,zz,dd,sbs,SBS:
EQI:=(i)->SysEq[i]:
z:=(i,XX)->(DifEq[MatAlf][nn(i)])[i](XX):
dd:=ParEq(1,1)[4]:
if dd=D then
sbs:=(i,t)->[fff(i,xx)=X[i](t),seq((D@@k)(ff(i))(xx)=
FF(i,k+1,t),k=1..nn(i)-1)]:
elif dd=diff then
sbs:=(i,t)->[fff(i,xx)=X[i](t),seq(diff(fff(i,xx),xx$k)=
FF(i,k+1,t),k=1..nn(i)-1)]:
end if:
SBS:=(t)->[seq(op(sbs(i,t)),i=1..NN),xx=t]:
NP:=DifEq[summ]([seq(nn(i),i=1..NN)]):
[NP,SBS(t),[seq(seq(diff((#
DifEq[MatAlf][k])[i](t),t)=
```

```
(DifEq[MatAlf][k+1])[i](t),k=1..nn(i)-1),i=1..NN),
seq(diff(zz(i,t),t)=subs(SBS(t),rhs(EQI(i))),i=1..NN)]]:
end proc:
```

Результат применения этой процедуры, который для краткости мы не приводим, представляется в виде трех упорядоченных списков: в первом содержится один элемент S - число уравнений нормальной системы (число степеней свободы НОМС); во втором – преобразования пользовательских функций к унифицированным функциям нормальной системы; третий список - есть упорядоченная нормальная система обыкновенных дифференциальных уравнений, причем первые S-N уравнений представляют собой результат стандартной замены переменных вида y'=z.

V.7.5 Приведение задачи Коши (V.5)-(V.6) к задаче Коши для нормальной системы ОДУ с унифицированными именами переменных

На основе программ распознавания системы ОДУ (блок «a») и программ приведения ОДУ к нормальному виду строится двухпараметрическая программная процедура
DifEq[SysCauchy_ConvNorm](System ODE, Inits Conditions)
приведения системы ОДУ произволь-ного порядка (V.5) с начальными условиями (V.6) к задаче Коши для нормальной системы ОДУ относительно унифицированных переменных $X_i^k(t)$ (5) с начальными условиями (V.9), т.е., к задаче Коши для нормальной системы ОДУ. Результат применения процедуры выводит упорядоченный список из шести упорядоченных списков: в первом содержится два числа: S - число уравнений нормальной системы и *M=max(n1,..,nN)* - максимальный порядок уравнений в исходной системе уравнений, во втором - упорядоченные списки новых переменных, - число этих списков равно *M*, выбранных по следующему принципу - в первом списке содержатся новые переменные, *X[i]*, полученные из независимых функций пользователя, во втором - первые производные от этих переменных, *Y[i]*, если вторые производные от этих переменных содержатся в системе ОДУ, и т.д., - до *M-1*-го списка. Таким образом, количество внутренних списков независимых функций совпадает с максимальным порядком уравнений исходной системы, M. Третий список содержит преобразования пользовательских функций к унифицированным функциям нормальной системы, четвертый список - есть упорядоченная нормальная система обыкновенных дифференциальных уравнений, причем первые *S-N* уравнений представля-

ют собой результат стандартной замены переменных вида $y'=z$, в пятом - начальные условия для нормальной системы ОДУ в форме (V.9), в шестом - один элемент - начальное значение независимой переменной. Созданная процедура удобна для извлечения различной информации об исходной системе дифференциальных уравнений и начальных условий.

V.7.6 Программные процедуры численного решения задачи Коши для системы (V.5) (блок c)

Блок «c» содержит две программные процедуры численного решения системы ОДУ (V.5) с начальными условиями (V.6), - трехпараметрическую программную процедуру
DifEq[NumDsolve](System ODE, Inits Conditions, Method)
и пятипараметрическую процедуру
DifEq[ReNumDsolve](System ODE, Inits Conditions, Method1,x1,Method2).
Первая команда создает процедуру решения системы ОДУ с помощью метода Method, встроенного в пакет Maple; значение этого параметра *45* соответствует методу Рунге-Кутта 4-5 порядков, *78* – методу Рунге-Кутта 7-8 порядков, *rosenbrock* – методу Розенброка, *stiff* – методу stiff интегрирования жестких уравнений, *classic* – классическому методу (по умолчанию методом Эйлера), *taylor* – методом разложения в ряды Тейлора (см., например [17,18]). При этом Вывод решений осуществляется в виде упорядоченного списка в вложенными в него M упорядоченными списками. При этом первый упорядоченный список содержит численные значения N искомых функций: [X[1](t),..X[N](t)], порядок записи значений функций в списке совпадает с порядком записи дифференциальных уравнений системы. Второй упорядоченный список содержит значения первых производных тех функций, производные которых не ниже второго порядка содержатся в системе ОДУ. Порядок записи значений первых производных функций в списке совпадает с порядком записи дифференциальных уравнений системы - при этом пропускаются значения производных тех функций, производные выше первого порядка которых не содержатся в системе ОДУ. Третий упорядоченный список содержит значения первых производных тех функций, производные которых не ниже третьего порядка содержатся в системе ОДУ. Порядок записи значений вторых производных функций в списке совпадает с порядком записи дифференциальных уравнений системы - при этом пропускаются значения вторых производных тех функций, производные выше второго порядка которых не содержатся в системе ОДУ, и т.д..

Для демонстрации формата ввода системы и вывода решений рассмотрим *пример интегрирования существенно нелинейной системы ОДУ* с максимальным 3-м порядком производных:

$$\frac{d^2F}{dx^2} = \Phi\frac{d^2Z}{dx^2};\quad \frac{d^3Z}{dx^3} = -F^2;\quad \frac{d\Phi}{dx} = F^3 = \sin x \tag{V.11}$$

с начальными условиями:

$$F(\pi) = 1;\quad F'(\pi) = 0;\quad \mathrm{Z}(\pi) = 1;\quad Z'(\pi) = -1;\quad Z''(\pi) = -1/3;\quad \Phi(\pi) = 0. \tag{V.12}$$

Систему (V.11) пользователь может ввести, например, таким способом[5]:

```
>ODE1:=[(D@@2)(F)(x)=Phi(x)*(D@@2)(Z)(x),(D@@3)(Z)(x)=
-F(x)\^{}2,D(Phi)(x) = F(x)\^{}3+sin(x)];
```

а начальные условия (V.12) для этой системы вводятся следующим образом:

```
> Inits1:=[F(Pi)=1,D(F)(Pi)=0,Z(Pi)=1,D(Z)(Pi)=-1,
(D@@2)(Z)(Pi)=1/3,Phi(Pi)=0];
```

Процедура решения системы (V.11) с начальными условиями (V.12) методом Розенброка осуществляется командой:

```
>SS:=DifEq[NumDsolve](Eqs,Inits1,rosenbrock);
```

– при этом решению мы произвольно присвоили имя SS. Численное решение выводится функцией SS(x):

```
>SS\eqref{GrindEQ__1_};

[[-1.56150852237610560,5.32544568814217101,-2.15254989920542972],
[3.95149696118901738,-3.48591778864530966],[1.81299087038185602]]
```

В данном случае список решений состоит из трех внутренних списков (*M=3*), в первом из них содержится 3 числа (*N=3*) – это значения функций [F(V.5),Z(V.5),Φ(V.5)], во втором списке содержится два числа, поскольку в системе (V.11) содержатся лишь две произ-водные не ниже 2-го порядка, - это значения первых производных [F'(V.5),Z'(V.5)], наконец, третий список содержит лишь одно число – это значение второй производной функции Z"(V.5). Для вывода одного из этих списков, i-го, достаточно применить процедуру SS(V.5)[i]:

[5] Уравнения вводятся упорядоченным списком.

```
>SS\eqref{GrindEQ__1_}[1];

[-1.56150852237610560, 5.32544568814217101, -2.15254989920542972]
```

График траектории системы в конфигурационном пространстве $E_3 : \{X_1 = F, X_2 = Z, X_3 = \Phi\}$ можно получить простой командой Maple, как и для трехмерного графика обычной функции:

```
>plots[spacecurve](SS[1](t),t=0..10,color=black,axes=BOXED,
labels=[‘X[1]‘,‘X[2]‘,‘X[3]‘]);
```

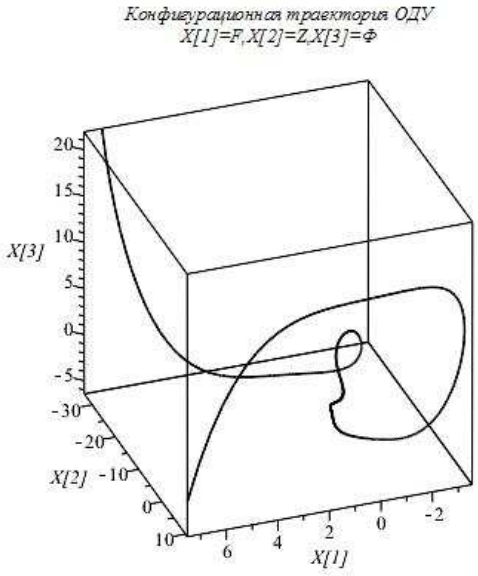


Рис.V.58 *Трехмерная конфигурационная траектория системы* (V.11)-(V.12) *[F(t),Z(t),Φ(t)] при изменении «времени» на интервале [0,10]*

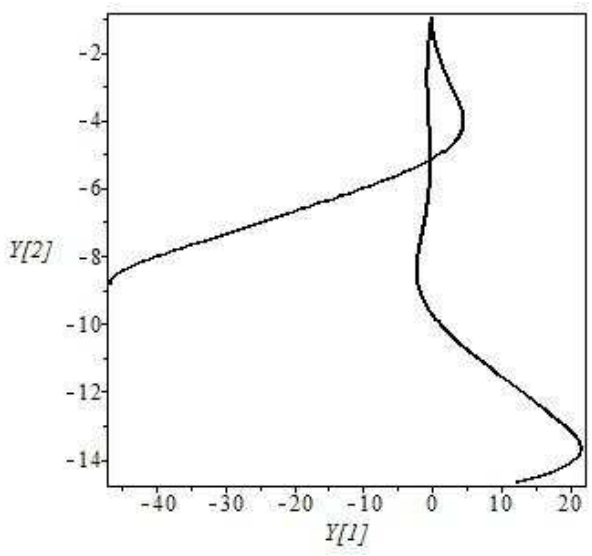


Рис.V.59 *Двумерная фазовая траектория системы* (V.11)-(V.12) *[F'(t),Z'(t)] при изменении «времени» на интервале [0,10].*

Комментируя графики (Рис.V.58, Рис.V.59), заметим, что изломы траектории на них – кажущийся эффект масштабов графиков – при увеличении масштаба изломы пропадают. Далее, программная процедура
`DifEq[ReNumDsolve](Eqs, Inits, Method1,x1, Method2)` построена на основе рассмотренный выше процедуры
`DifEq[NumDsolve]` и встроенной в Maple программной процедуры

`piecewise(x>=x0 and x<x1,f1, x>x1,f2)` кусочно - заданной функции, так что на интервале [x0,x1] при численном интегрировании системы ОДУ применяется метод Method1, а на интервале (x1,...) – метод Method2. При этом начальными условиями для численного интегрирования методом Method2 являются результаты численного интегрирования системы ОДУ в точке x1, полученные методом Method2. Указанный метод можно назвать *методом интегрирования с перезагрузкой начальных условий*. Данный метод следует применять в тех случаях, когда на некотором отрезке стандартные методы интегрирования не дают хороших результатов. Следует отметить, что программные процедуры DifEq[NumDsolve] и DifEq[ReNumDsolve] не требуют выполнения пользователем предварительных операций по приведению системы ОДУ к нормальному виду, так как эти операции являются встроенными процедурами в указанные, – нормирование системы ОДУ и извлечение всей необходимой информации производится автоматически.

V.7.7 Точность и скорость вычислений

Для тестирования пакета программ DifEq на точность вычислений, а также выяснения немаловажного для задач компьютерного моделирования вопроса о скорости вычислений различными численными методами можно провести численное интегрирование различных точно решаемых нелинейных систем ОДУ различными методами с помощью программных DifEq[NumDsolve] и DifEq[ReNumDsolve]. При этом реальное время, необходимое для интегрирования системы и представления численных результатов в графическом виде можно вычислить с помощью встроенной в Maple функции time(), которая определяет точное время в секундах по встроенному в компьютер таймеру. Точность и скорость вычислений тестировались по целому ряду точно решаемых нелинейных систем ОДУ, примеры которых можно найти, во многих учебниках по дифференциальным уравнениям. Рассмотрим следующий пример задачи Коши для нелинейной системы ОДУ [129]:

$$\frac{dy}{dx} = \frac{y}{x}; \quad \frac{dz}{dx} = -\frac{y}{z} \tag{V.13}$$

с начальными условиями:

$$y\left(\frac{1}{2}\right) = \frac{1}{2}; \quad z\left(\frac{1}{2}\right) = \frac{\sqrt{3}}{2}. \tag{V.14}$$

Указанная задача Коши точно решается:

$$y(x) = x; \quad z(x) = \sqrt{1-x^2}; \quad (\mathrm{z} \geq 0), \tag{V.15}$$

так что $x^2 + y^2 = 1$, т.е. конфигурационной траекторией системы является окружность единичного радиуса. Хотя найденное точное решение задачи Коши имеет весьма простой вид, с точки зрения применения численных методов интегрирования система (V.13) является достаточно неудобной: она не является автономной и имеет особые точки при x=0, z=0. Поэтому система ОДУ (V.13) и подобные ей удобны для тестирования численных методов решения ОДУ. Рассмотрим пример численного решения задачи Коши (V.14)-(V.15) и манипуляции с этими решениями. Сразу оговоримся, что для созданного комплекса программ исследование системы (V.14)-(V.15) является элементарной задачей, но мы продемонстрируем на этой задачи все особенности применения пакета программ, чтобы не загромождать изложение громоздкими выражениями. Далее, хотя система (V.14) уже является нормальной, тем интереснее будет проверить для этого случая исполнение процедур преобразования ее к задаче Коши для нормальной системы с унифицированными именами переменных:

```
>DifEq[SysCauchy\_ConvNorm]([diff(y(x),x)=y(x)/x,
sqrt\eqref{GrindEQ__3_}/2]);
```

$$[[2,1],[[X_1(t),X_2(t)]],[],[y(x)=X_1(t),z(x)=X_2(t)],$$
$$\left[\tfrac{d}{dt}X_1(t)=\tfrac{X_1(t)}{t},\tfrac{d}{dt}X_2(t)=-\tfrac{X_1(t)}{X_2(t)}\right],$$
$$\left[X_1\left(\tfrac{1}{2}\right)=\tfrac{1}{2},X_2\left(\tfrac{1}{2}\right)=\tfrac{\sqrt{3}}{2}\right],\tfrac{1}{2}\Big]$$

Следует заметить, что данная команда не является обязательной для решения системы уравнений, но она хорошо демонстрирует структуру внутренних программных процедур. Само численное решение, например, методом Розенброка сразу достигается командой:

```
>SS1:=DifEq[NumDsolve]([diff(y(x),x)=y(x)/x,
diff(z(x),x)=-y(x)/z(x)],
[y(1/2)=1/2,z(1/2)=sqrt\eqref{GrindEQ__3_}/2],
rosenbrock);
```

и выводится в функциональном списочном виде, соответствующем векторной функции скалярного аргумента. Продемонстрируем построение *конфигурационной траектории* системы (V.14)-(V.15)

$$r = [y(x), z(x)] = [X_1(t), X_2(t)]$$

при интегрировании методом Розенброка. Построение траектории достигается простой стандартной командой Maple, в которой мы указали две необязательные опции: scaling = CONSTRAINED – для сохранения масштаба, и color=black, для определения цвета кривой:

```
>plot([SS4(t)[1,1],SS4(t)[1,2],t=-1..1],
scaling=CONSTRAINED,color=black);
```

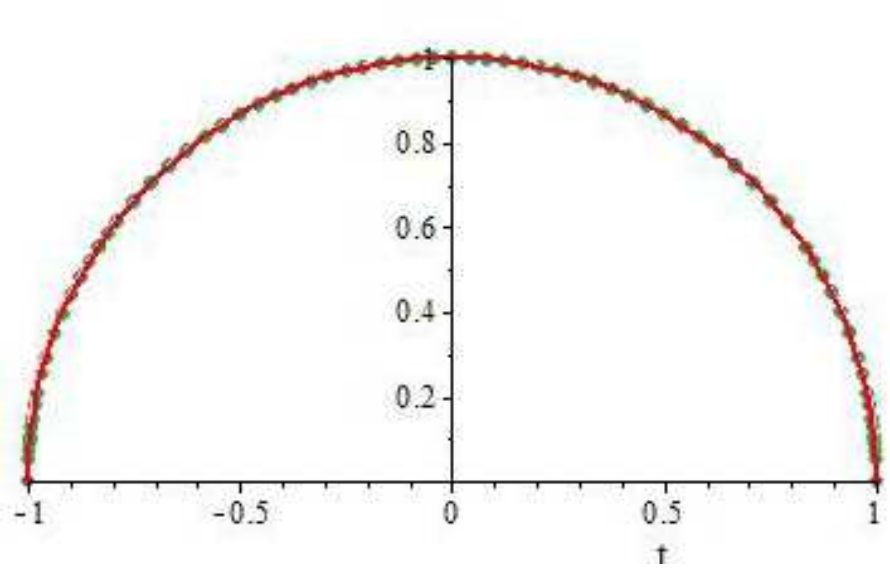


Рис.V.60 *Конфигурационная траектория системы* (V.14) - (V.15) *[X1(t),X2(t)] при изменении «времени» t на интервале [-1,1].*

Заметим, что визуально изменение фазовых траектории в зависимости от применяемых из перечисленных выше методов численного интегрирования обнаружить не удается. Для выяснения вопроса о точности методов интегрирования будем вычислять логарифм модуля разности численного, Z(t), и точного, Z0(t), решений (V.15):

$$L(t) = \lg |Z(t) - Z0(t)|. \qquad \text{(V.16)}$$

Время `T` - расчета и построения конфигурационных диаграмм типа Рис.V.61 можно вычислить с помощью встроенной в Maple команды `time()`. На гистограмме Рис.V.64 показана зависимость времени выполнения данной операции от метода численного интегрирования (см. [98] – [100]).

Вычисления проводились на компьютере AMD Athlon^{TM} 64×2 Dual Core Processor 4200+ 2.2 ГГц 2,00 ГБ ОЗУ. Данные о скорости вычислений методом разложения в ряды Тейлора, taylor, не приведены на гистограмме Рис.V.64, поскольку это время несопоставимо велико по сравнению с другими методами.

Некоторые результаты определения точности решения созданных программных процедур представлены на Рис.V.61, Рис.V.62. Сравнительная точность методов численного интегрирования показаны на гистограмме Рис.V.63.

Всюду на гистограммах – rk45 – метод Рунге – Кутта 4-5 порядков, rk78 – метод Рунге-Кутта 7-8 порядков, Rosn – метод Розенброка, Stif – метод stiff, Tayl – метод разложений Тейлора, Clas – классический метод.

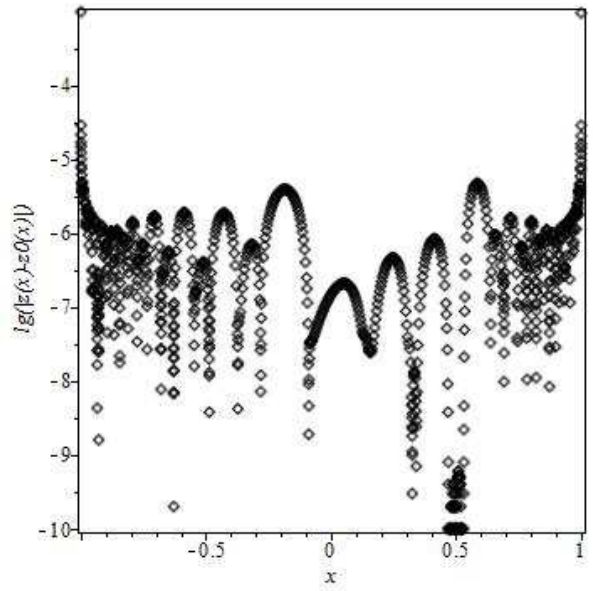


Рис.V.61 *Логарифм абсолютной точности численного интегрирования системы* (V.14)-(V.15) *методом Рунге-Кутта 4-5 порядков.*

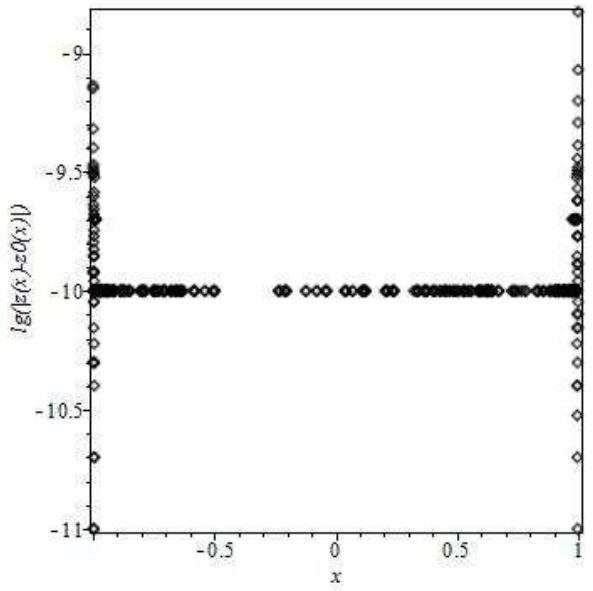


Рис.V.62 *Логарифм абсолютной точности численного интегрирования системы* (V.14)-(V.15) *методом Рунге-Кутта 7-8 порядков.*

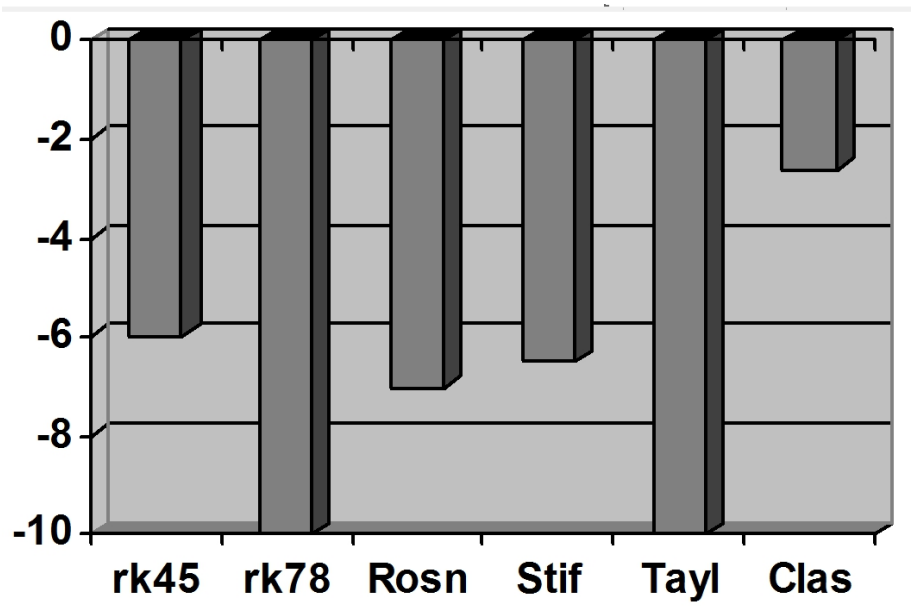


Рис.V.63 *Зависимость среднего логарифма абсолютной погрешности, $\langle L \rangle$, метода численного решения задачи Коши.*

Так, например, время, затрачиваемое на вычисления с помощью метода Рунге-Кутта составляет для рассмотренного примера порядка 0,062 сек, в то время, как эта же операция, выполненная с помощью метода Тейлора занимает 66,2 сек, т.е., в 1000 раз больше. Классический метод дает слишком плохие результаты для нелинейных систем ОДУ. Таким образом, оптимальными методами вычислений (достаточно хорошая точность при высокой скорости вычислений можно признать методы Розенброка и stiff-метод, а методом, дающим наибольшую точность при средней скорости вычислений можно признать метод Рунге-Кутта 7-8 порядков.

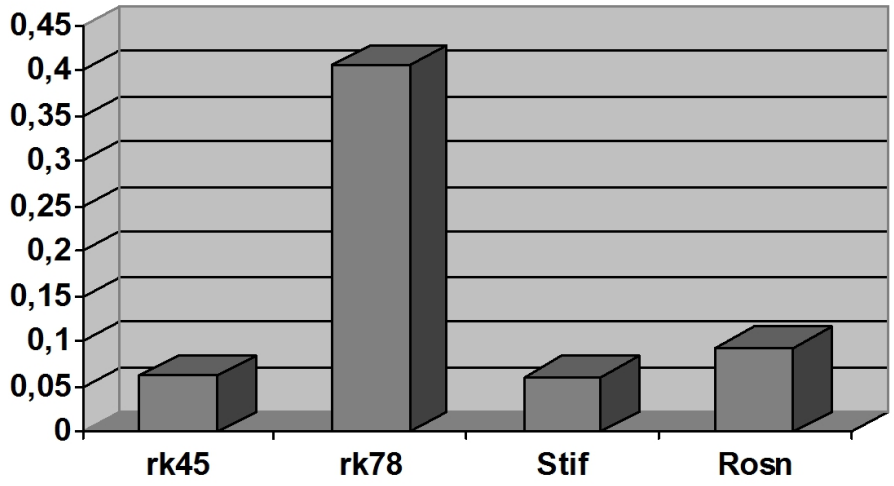


Рис.V.64 *Зависимость среднего времени, T (в секундах), расчета и построения конфигурационной диаграммы для системы (V.14)-(V.15) от метода численного решения задачи Коши: rk45 – метод Рунге-Кутта 4-5 порядков, rk78 – метод Рунге-Кутта 7-8 порядков, Rosn- метод Розенброка, Stif – метод stiff, Clas – классический метод.*

V.8 Программные процедуры сплайновой интерполяции функций

V.8.1 Сплайны

Опишем теперь алгоритмы и программные процедуры пакета программ Splines, позволяющие автоматически получать решение задачи Коши для нелинейной системы ОДУ произвольного порядка в форме равномерных кубических сплайнов и B-сплайнов. Для создания аппарата аналитического исследования решений нелинейной системы ОДУ и проведение компьютерного моделирования нелинейных обобщенно-механических систем были созданы специализированные программные процедуры, позволяющие выполнять над сплайнами алгебраические и интегро-дифференциальные операции, а также операции взаимообратного конвертирования сплайнов и B-сплайнов в кусочно- и кусочно-параметрически- заданные функции. Таким образом,

удалось добиться важного результата – создать комплекс программ, позволяющий проводить аналитическое компьютерное исследование нелинейных обобщенно-механических систем.

Дадим некоторые необходимые в дальнейшем определения и понятия теории сплайнов (см., например, [130], [128]).

Определение OV.1. *Пусть некоторая функция f(x) задана на отрезке [a,b], и пусть для некоторого целого $n \in \mathbb{Z}$ для внутренних точек отрезка [a,b] выполнено условие:*

$x_0 < x_1 < x_2 < \cdots x_{n-1} < x_n; \quad (x_0 \equiv a, x_n \equiv \mathrm{b}).$

Кубическим сплайном относительно функции f(x) на отрезке [a,b] называется функция S(x), удовлетворяющая трем условиям:

1. *на каждом отрезке $[x_{i-1}, x_i]$ $(i = \overline{1,n})$ S(x) является многочленом третьей степени;*
2. *S(x) является функцией класса C3 на всем отрезке [a,b];*
3. *в точках xi $(i = \overline{0,n})$выполняется равенство $S(x_i) = f(x_i), \quad (i = \overline{0,n})$.*
   (14)*Определение 2. Натуральным кубическим сплайном называется кубический сплайн, удовлетворяющий также граничным условиям вида:*
   $S''(a) = S''(b) = 0.$

Для натуральных кубических сплайнов справедлива теорема Шенберга – Уитни:

Теорема TV.1. *Для любой функции f(x) и любого разбиения отрезка [a,b] существует один и только один натуральный сплайн S(x).*

Определение 3. Равномерным сплайном называется кубический сплайн, полученный равномерным разбиением отрезка [a,b]:

$x_i - x_{i-1} = \Delta, \quad (\forall i \in \overline{1,n})$.

Сплайн порядка n представляется многочленом n-го порядка на каждом отрезке. Таким образом, сплайн n-го порядка, $S(x,p)$, можно рассматривать как кусочно-заданную на отрезке $[a,b]$ функцию, представленную многочленом p-го порядка на каждом отрезке, с коэффициентами такими, чтобы обеспечивать принадлежность $S(x,p)$ функциям класса C_p на всем отрезке $[a,b]$.

В пакете `CurveFitting` системы Maple имеются встроенные процедуры построения сплайновой интерполяции Spline (см., например, [3]), которые могут иметь формат Spline(`xydata`, `v`, `dgr`, `endpts`) или `Spline`(`xdata`, `ydata`, `v`, `dgr`, `endpts`), где `xydata` - двумерный массив вида [. . . ,[xi,yi],. . .], `xdata`, `ydata` – одномерные массивы вида [. . . ,xi,..], [. . . ,yi,. . .]; `v` – имя независимой переменной, `dgr` – необязательный параметр, задание которого осуществляется в форме degree=p, где $p \in \mathbb{Z}$ – наивысшая степень интерполяционных полиномов. По умолчанию p=3, и получается кубический сплайн. Наконец, `endpts` – необязательный параметр, управляющий типом сплайна, например, endpoints='natural' дает натуральный сплайн. Применение процедуры Spline к двумерному или двум одномерным массивам генерирует сплайн, который в СКМ Maple, представляется, фактически, кусочно-заданной функций с вертикальным форматом вывода ее кусков. Несмотря на то, что программные процедуры сплайновой интерполяции функций системы Maple весьма достоверно и надежно осуществляют построение сплайновой интерполяции числовых баз данных указанных массивов, программный аппарат действий со сплайнами в СКМ отсутствует, что не позволяет непосредственно применять результаты сплайновой интерполяции численных решений для аналитических исследований моделей. Поэтому для создания эффективного аппарата компьютерного моделирования нелинейных обобщенно-механических систем в СКМ необходимо разработать программное обеспечение операций над сплайнами.

V.8.2 Процедуры конвертирования сплайнов

Используя процедуру Spline, создадим необходимую в дальнейшем промежуточную 6-ти параметрическую обратную к Spline простую процедуру генерации равномерных n-кусочных кубических сплайнов относительно функции f(x), заданной на отрезке [a,b], Splines[SplineF](f,x,a,b,n,z) с передачей имени z ее независимому аргументу:

```
>Splines[SplineF]:=proc(f,x,a,b,n,z) local F,X,Basa:
Basa:=[seq([a+i*(b-a)/n,eval(F(a+i*(b-a)/n))],i=0..n)]:
CurveFitting[Spline](Basa,z): end proc:
```

Введем предварительно трехпараметрическую процедуру Splines[Conv_List](sp,t,z) конвертирования сплайна *sp* функции f на заданном отрезке в упорядоченный список, состоящий из четного числа элементов, в котором каждая пара представляет собой упорядоченный набор эле-

ментов, первый из которых – неравенство, устанавливающее верхнюю границу интервала, а второй – сплайн на данном интервале:

```
>Splines[Conv\_List]:=proc(sp,t,z)
local LSS0,LSS,T,ddd,NN0,NN1,t1,DDD:
LSS0:=convert(sp,list):LSS:=(T)-> subs(t=T,LSS0):
NN0:=nops(LSS0):NN1:=(NN0+1)/2:
ddd:=rhs(op(3,LSS(t)))-rhs(op(1,LSS(t))):
t1:=rhs(op(1,LSS(t)))-ddd:DDD:=t1+NN1*ddd:
[seq(op(i,LSS(z)),i=1..NN0-1),(z<DDD,op(-1,LSS(z)))]:
end proc:
```

Заметим, что процедура автоматически распознает параметры сплайна. Создадим также и обратную процедуру конвертирования списка в кусочно-заданную функцию, `piecewise`:

```
>Splines[Conv_Piece]:=proc(Ls,t,z) local T,LS0,LSS0:
LS0:=(T)->subs(t=T,Ls):
LSS0:=(t)->subs(op(-2,LS0(t))=NULL,LS0(t)):
piecewise(op(LSS0(z))):
end proc:
```

Здесь применена встроенная процедура Maple `NULL` – команда пустого множества Ш. В качестве примеров генерации сплайнов, на которых мы будем демонстрировать применение более сложных операций рассмотрим следующие три:

```
>PPS6:=(z)->Splines[SplineF](sin(x),x,0,2*Pi,6,z):
PPS12:=(z)->Splines[SplineF](sin(x),x,0,2*Pi,12,z):
PPS24:=(z)->Splines[SplineF](sin(x),x,0,2*Pi,24,z):
```

Все три сплайна созданы по отношению к функции *f(x)=sin(x)* на отрезке [0,2π] с именем независимой переменной z: сплайн PPS6 содержит 6 интервалов, PPS12 – 12 интервалов и PPS24 – 24 интервала. Результат генерации сплайнов мы не приводим из-за громоз-дкости результата. Заметим только, что уже 6-кусочный сплайн, PPS6, дает график, ко-торый визуально невозможно отличить от графика порождающей его функции *sin(x)*. Эффективную проверку корректности созданных программных процедур можно провести, дважды конвертируя сгенерированный сплайн с помощью процедур Splines[Conv_List] и Splines[Conv_Piece]. Проверка показывает результаты, которые графически неотличимы от оригинала.

V.9 Программные процедуры операций над сплайнами

V.9.1 Процедура дифференцирования сплайнов

Операция вычисления производной n-го порядка в точке t0 от сплайна можно ввести простой программной процедурой с помощью встроенной в Maple программной процедурой convert(DSP(T),piecewise,T), конвертирующей сплайн в кусочно-заданную функцию:

```
>Splines[SplineDiff]:=proc(spline,t,t0,n) local F,X,T,DSP,DS:
F:=(X)->subs(t=X,spline):DSP:=(T)->subs(X=T,diff(F(X),X\n)):
DS:=(T)->convert(DSP(T),piecewise,T):DS(t0):
end proc:
```

Вычислим, например, с помощью построенной процедуры первую производную сплайна PPS6 в точке $\pi/4$:

```
>Splines[SplineDiff](PPS6(t),t,Pi/4,1);
```

$\frac{207\sqrt{3}}{160\pi} \approx 0,7133 \frac{207\sqrt{3}}{160\pi} \approx 0,7133$

Прямое вычисление значения $\sin(\pi/4) =$ дает $\sqrt{2}/2 \approx 0,7071$, т.е., отличается от предыдущего в третьем знаке. Однако, уже при увеличении числа интервалов в два раза (сплайн PPS12) точность интерполяции возрастает на порядок: PPS12($\pi/4$)?0,7072. Для использования производной от заданного сплайна как функции необходимо теперь просто определить эту функцию:

```
>F:=(x)->Splines[SplineDiff](PPS(t),t,x,1):
```

V.9.2 Процедуры вычисления функций от сплайнов

Процедуры нахождения функции *f(S,t)* от сплайна S можно создать двумя путями.

Первый, самый простой, основан на первоначальном конвертировании исходной аппроксимации в список и построении на основе его новой базы для сплайна. Второй, более сложный, состоит в начальном конвертировании исходного сплайна в список, преобразование функций списка и конвертирование этого списка в кусочно-заданную функцию. Полученная вторым способом функция уже не является, строго говоря, сплайном n-го порядка, так как на каждом интервале представляется уже не многочленом n-го порядка, а функцией от этого многочлена. Однако, при этом сохраняется принадлежность кусочно-заданной функции *f(S)* классу Cn, если сама функция *f(x)*

принадлежит этому классу. Первая из указанных процедур 4-х параметрическая процедура Splines[SplineFunctionPoint]имеет вид:

```
>Splines[SplineFunctionPoint]:=proc(sp,t,f,z)
local LS,n,N,t1,d,F,T,k,X,FF,FFF,TT,tN,Basa,LS1:
LS:=(sp,t,z)->Splines[Conv_List](sp,t,z):
n:=(sp,t,z)->nops(LS(sp,t,z)):
N:=(sp,t,z)->1/2*n(sp,t,z):
d:=(sp,t,z)->rhs(LS(sp,t,z)[3])-rhs(LS(sp,t,z)[1]):
t1:=(sp,t,z)->rhs(LS(sp,t,z)[1]):
TT:=(k,sp,t,z)->t1(sp,t,z)+(k-1)*d(sp,t,z):
tN:=(sp,t,z)->TT(N(sp,t,z)-1,sp,t,z):
F:=(k,sp,t,z)->subs(z=TT(k,sp,t,z),LS(sp,t,z)[2*k]):
FF:=(sp,X,T,f)->subs({sp=X,t=T},f):
FFF:=(k,sp,t,z)->eval(FF(sp,F(k,sp,t,z),TT(k,sp,t,z),f)):
Basa:=[seq([TT(k,sp,t,z),FFF(k,sp,t,z)],k=1..N(sp,t,z))]:
CurveFitting[Spline](Basa,z):
end proc:
```

Здесь первый параметр есть сплайн, второй – его независимая переменная, третий – функция от сплайна, четвертый – присвоенное новое имя независимой переменной. Вторая из указанных процедур с такими же параметрами имеет вид:

```
>Splines[SplineFunction]:=proc(sp,t,f,z)
local LS,n,N,t1,d,F,T,k,X,FF,FFF,TT,tN,SBS,LS1:
LS:=(sp,t,z)->Splines[Conv_List](sp,t,z):
n:=(sp,t,z)->nops(LS(sp,t,z)): N:=(sp,t,z)->1/2*n(sp,t,z):
d:=(sp,t,z)->rhs(LS(sp,t,z)[3])-rhs(LS(sp,t,z)[1]):
t1:=(sp,t,z)->rhs(LS(sp,t,z)[1]):
TT:=(k,sp,t,z)->t1(sp,t,z)+(k-1)*d(sp,t,z):
tN:=(sp,t,z)->TT(N(sp,t,z)-1,sp,t,z):
F:=(k,sp,t,z)->LS(sp,t,z)[2*k]: FF:=(sp,X,T,f)
->subs(\{sp=X,t=T\},f):FFF:=(k,sp,t,z)
->eval(FF(sp,F(k,sp,t,z),z,f)):
SBS:={seq(LS(sp,t,z)[2*k]=FFF(k,sp,t,z),k=1..N(sp,t,z))}:
LS1:=subs(SBS,LS(sp,t,z)):Splines[Conv_Piece](LS1,t,z):
end proc:
```

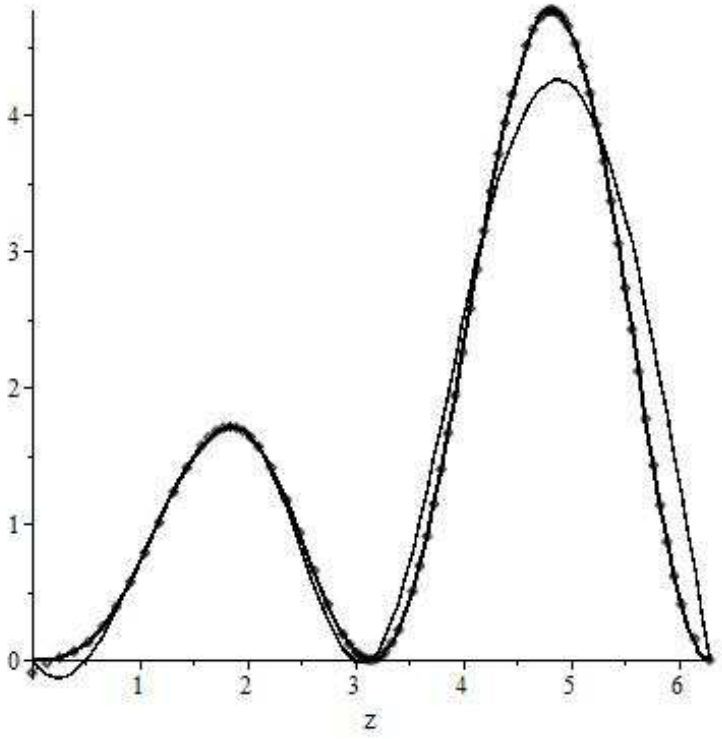


Рис.V.65 *Сравнение функций от сплайна* `PPS6(z)`*, полученной процедурой* `Splines [SplineFunctionPoint]` *(тонкая линия), процедурой* `Splines[SplineFunction]` *(жирная линия) с функцией* $x$ffl $\sin^2(x)$ *(точечная линия).*

Этот результат, как и многие другие, наглядно показывают практически полное совпадение результатов интерполяции функции сплайнов с помощью процедуры Splines[SplineFunction] с функцией-оригиналом даже при небольшом числе интервалов и значительное расхождение результатов при применении процедуры Splines[SplineFunctionPoint][6].

V.9.3 Процедура вычисления функций от двух сплайнов

В дальнейшем нам понадобятся функции от двух сплайнов, `S1` и `S2`, - `f(S1,S2,t)`. В частности, такие функции будут полезными при вычислении определенных интегралов, когда по формуле Ньютона необходимо вычислить разность двух сплайнов. Создадим процедуру вычисления

[6] Заметим, что с увеличением числа интервалов процедура Splines[SplineFunctionPoint]также начинает давать неплохие результаты.

таких функций. При этом важно, чтобы сплайны совпадали как внешними, так и внутренними интервалами. Будем предполагать сплайны равномерными. Для ввода процедуры вычисления функции от пары сплайнов, `Splines[BiSplineFunction`, используем рассмотренный выше второй метод построения программной процедуры вычисления функции от сплайна. Созданная программная процедура `Splines[BiSplineFunction](S1,S2,t)` автоматически распознает вводимые пользователем сплайны `S1, S2` и при их несоответствии другу выдает пользователю сообщение об этом и прекращает выполнение операций.

```
>Splines[BiSplineFunction]:=proc(sp1,sp2,t,f,z)
local LS_1,LS_2,N1,N2,n1,n2,t1_1,t1_2,d1,d2,d,
t1,N,n,F1,F2,T,k,TT,tN,FF,FFF,SBS,LSS:
LS_1:=(sp1,t,z)->Splines[Conv_List](sp1,t,z):
LS_2:=(sp2,t,z)->Splines[Conv_List](sp2,t,z):
N1:=(sp1,t,z)->1/2*nops(LS_1(sp1,t,z)):
n1:=(sp1,t,z)->nops(LS_1(sp1,t,z)):
d1:=(sp1,t,z)->rhs(LS_1(sp1,t,z)[3])-rhs(LS_1(sp1,t,z)[1]):
t1_1:=(sp1,t,z)->rhs(LS_2(sp1,t,z)[1]):
N2:=(sp2,t,z)->1/2*nops(LS_2(sp2,t,z)):
n2:=(sp2,t,z)->nops(LS_2(sp2,t,z)):
d2:=(sp2,t,z)->rhs(LS_2(sp2,t,z)[3])-rhs(LS_2(sp2,t,z)[1]):
t1_1:=(sp1,t,z)->rhs(LS_1(sp1,t,z)[1]):
t1_2:=(sp2,t,z)->rhs(LS_2(sp2,t,z)[1]):
if N1(sp1,t,z)<>N2(sp2,t,z) or
d1(sp1,t,z)<>d2(sp2,t,z) or t1_1(sp1,t,z)<>t1_2(sp2,t,z)
then print("Интервалы сплайнов не совпадают"):
else
N:=(sp1,t,z)->N1(sp1,t,z):n:=(sp1,t,z)->n1(sp1,t,z):
d:=(sp1,t,z)->d1(sp1,t,z):t1:=(sp1,t,z)->t1_1(sp1,t,z):
TT:=(k,sp1,t,z)->t1(sp1,t,z)+(k-1)*d(sp1,t,z):
tN:=(sp1,t,z)->TT(N(sp1,t,z)-1,sp1,t,z):
F1:=(k,sp1,t,z)->LS_1(sp1,t,z)[2*k]:
F2:=(k,sp2,t,z)->LS_2(sp2,t,z)[2*k]:
FF:=(sp1,sp2,X,Y,T,f)->subs({sp1=X,sp2=Y,t=T},f):
FFF:=(k,sp1,sp2,t,z)->
eval(FF(sp1,sp2,F1(k,sp1,t,z),F2(k,sp2,t,z),z,f)):
SBS:={seq(LS_1(sp1,t,z)[2*k]=FFF(k,sp1,sp2,t,z),
k=1..N(sp1,t,z))}:LSS:=subs(SBS,LS_1(sp1,t,z)):
```

```
Splines[Conv_Piece](LSS,t,z):
end if:end proc:
```

V.9.4 Процедура вычисления определенного интеграла от функции сплайна

Создадим теперь процедуру вычисления определенного интеграла от функции f, заданной в сплайновой интерполяции, вида $\int_a^b f(S,t)dt$, где `S(t)` – равномерный `N`- кусочный сплайн, определенный на некотором отрезке `[t1,tn]`, и `[a,b]` – произвольный вещественный отрезок. Предварительно введем процедуру вычисления ступенчатой функции, которая в отличие от стандартных Maple-функций хорошо определена на границах интервалов:

```
>Splines[MyHeaviside]:=(x,a,b)->
piecewise(x<a or x>b,0,x=a,1,x=b,1,x>a and x<b,1):
```

на основе которой определим 4-х параметрическую программную процедуру вычисления определенного интеграла на отрезке `[a,b]` от функции `f` сплайна `sp(t)` .

```
>Splines[SplineDefInt]:=proc(sp,t,f,a,b) local
#
LS,N,n,t1,tn,ST,d,K1,K2,F,T,k,FG,FGN,FF,FFF,TT,LS1,SP1,NumInt:
LS:=convert(sp,list):N:=1/2*(nops(LS)+1):n:=nops(LS)+1:
d:=rhs(LS[3])-rhs(LS[1]):t1:=rhs(LS[1])-d:tn:=t1+N*d:
TT:=(k)->t1+(k-1)*d:ST:=(T)->Splines[MyHeaviside](T,t1,tn):
K1:=floor((a-t1)/d):K2:=ceil((b-t1)/d):
if ST(a)=0 then
print("Нижний предел интегрирования меньше
нижней границы интервала",a<t1):
elif ST(b)=0 then
print("Верхний предел интегрирования больше
верхней границы интервала",b>tn):
elif ST(a)=1 and ST(b)=1 then
F:=(T,k)->piecewise(k<N,FG(T,k),k=N,FGN(T)):
FF:=(sp,X,T,f)->subs(\{sp=X,t=T\},f):
FFF:=(T,k)->eval(FF(sp,F(T,k),T,f)):
FG:=(T,k)->eval(subs(t=T,LS[2*k])):
FGN:=(T)->eval(subs(t=T,LS[n-1])):
```

```
if K1=K2-1 then NumInt:=Splines[Integral](a,b,20,6,T,FFF(T,K1+1)):
else
NumInt:=Splines[Integral](a,TT(K1+1),20,6,T,FFF(T,K1+1))+
Splines[Integral](TT(K2),b,100,6,T,FFF(T,K2))
+sum(Splines[Integral](TT(k),TT(k+1),20,6,T,FFF(T,k+1)),
k=K1+1..K2-1):
end if:
end if:
NumInt:
end proc:
```

Созданная процедура автоматически распознает параметры сплайна и в случаях, когда границы отрезка `[a,b]` выходят за границы определения сплайна, выдает пользователю сообщение и прекращает выполнение операции. Созданная процедура использует промежуточную процедуру приближенного интегрирования методом прямоугольников, `Splines[Integral]`. Рассмотрим пример интегрирования сплайна, для чего будем сравнивать точное значение интеграла: $\int_0^{2\pi} x\sin^2(x)dx = \pi^2 \approx 9,869604404$ со значениями интеграла, вычисленного на рассмотренных выше сплайнах PPS6(x),PPS12(x):

```
>Splines[SplineDefInt](PPS6(t),t,t*PPS6(t)\^{}2,0,2*Pi);
```

9.319671503

```
>Splines[SplineDefInt](PPS12(t),t,t*PPS12(t)\^{}2,0,2*Pi);
```

9.810990839.

Таким образом, при увеличении в 2 раза числа интервалов сплайна точность вычисления определенного интеграла возросла более, чем на порядок, и ошибка вычислений для рассмотренного примера не превышает 1%.

V.10 Программные процедуры операций над B - сплайнами

Определение OV.2. *B-сплайном (базисным сплайном) называют сплайн-функцию, имеющую наименьший носитель для заданной степени, порядка гладкости и разбиения области определения.*

Фундаментальная теорема устанавливает, что любая сплайн-функция для заданной степени, гладкости и области определения может быть представлена как линейная комбинация B-сплайнов той же степени и гладкости на той же области определения (см., например, [130], [128]). В СКМ Maple функция вычисления B-сплайнов, `BSplineCurve`, содержится, как и функция `Spline`, в пакете `CurveFitting`. Программная процедура `BSplineCurve` имеет формат ввода, аналогичный формату ввода процедуры Spline:
BSplineCurve(`xydata`,`v`,`opts`) или BSplineCurve(`xdata`, `ydata`,`v`,`opts`),
где теперь, необязательные параметры имеют формат `order=k` или `knots =knotlist`, а `knowlist` – список узлов. Однако, формат вывода процедуры `BSplineCurve` существенно отличается от формата вывода процедуры `Spline` – вместо кусочно - заданной на отрезке `[a,b]` функции
`S(x) BSplineCurv` выводит два списка кусочно- заданных функций `S(t)` и `x(t)` в вертикальном формате, где `t` – новый параметр, который принимает значения на некотором интервале `[k,n] n>k`, где $k, n \in \mathbb{Z}$, при этом отрезок `[k,n]` разбивается на целочисленные отрезки: $[i, i{+}1]$, $i = \overline{k, n}$. Фактически, формат вывода B-сплайна эквивалентен выводу кусочно-параметрически-заданной функции:

$$x = x(t);\ S = S(t) \Rightarrow r = [x(t), S(t)], \tag{V.17}$$

хотя и не совпадает с ней, что создает большие неудобства для пользователя.

Для устранения указанного недостатка СКМ Maple введем 6-ти параметрическую процедуру
`Splines[BSplineF(f,x,a,b,n,z)` генерации n-кусочного B-сплайна на основе заданной на интервале `[a,b]` функции `f(x)` и последующего конвертирования полученного B-сплайна в кусочно – параметрически заданную функцию вида (V.17):

```
>Splines[BSplineF]:=proc(f,x,a,b,n,z) local
F,X,Basa,BS,BSy,BSx,x\_min,x\_max,Xt,Yt:
F:=(X)->subs(x=X,f):
Basa:=[[a,eval(F(a))],seq([a+i*(b-a)/n,
eval(F(i*(b-a)/n))],i=0..n),[b,eval(F(b))],[b,eval(F(b))]]:
BS:=CurveFitting[BSplineCurve](Basa,X,order=4):
BSy:=(X)->op(-1,BS(X)):
BSx:=(X)->op(-2,BS(X)):
x_min:=min(\rhs(BSy(X)))-1:
x_max:=max(\rhs(BSy(X))):
```

```
Yt:=evalf(convert(op(subs(X=z,BS))[2],piecewise,z)):
Xt:=evalf(convert(op(subs(X=z,BS))[1],piecewise,z)):
[Xt,Yt,x\_min,x\_max]:
end proc:
```

Также создадим трехпараметрическую процедуру
`Splines[ConvBSpline_Piece](BS,x,z)` конвертирования B-сплайна BS(x) в кусочно – параметрически заданную функцию с именем z параметра:

```
>Splines[ConvBSpline\_Piece]:=proc(BS,x,z)
local X,BBS,BSy,BSx,x\_min,x\_max,Xt,Yt:
BBS:=(X)->subs(x=X,BS):
BSy:=(X)->op(-1,BBS(X)):
BSx:=(X)->op(-2,BBS(X)):
x_min:=min(\rhs(BSy(X)))-1:
x_max:=max(\rhs(BSy(X))):
Yt:=evalf(convert(op(subs(X=z,BBS(X)))[2],piecewise,z)):
Xt:=evalf(convert(op(subs(X=z,BBS(X)))[1],piecewise,z)):
[Xt,Yt,x_min,x_max]:
end proc:
```

Созданные процедуры конвертирования устанавливают связь между B - сплайновым и сплайновым представлениями интерполяции и позволяют использовать мощный аппарат B-сплайнов *аналитизации* процесса численного интегрирования нелинейных систем ОДУ.

V.11 Сплайновое представление численного решения нелинейной системы ОДУ

Интегрируем, наконец, программные процедуры описанного здесь пакета программ Splines с программными процедурами пакета DifEq в блоке «i» конвертирования численных решений нелинейной системы ОДУ в кусочно-заданные функции (см. Рис. 1 в работе [1]). Этот блок содержит четыре программные процедуры:
`DifEq[ODESpline], DifEq[ODEBSpline],`
`DifEq[ReODESpline], DifEq[ReODEBSpline]`. Процедура автоматизированной сплайновой обработки численного решения задачи Коши для нелинейной системы произвольного числа ОДУ произвольного порядка
`DifEq[ODESpline](Eqs,Inits,Method,x1,n,i,k,x)` содержит 8 параметров:

Eqs - упорядоченный список системы ОДУ, Inits – упорядоченный список начальных условий, Method – Метод интегрирования, x1 – конечная точка интервала интегрирования, n – число интервалов сплайна, i – порядковый номер выводимой функции[7], k – порядок производной этой функции, x - присваиваемое имя независимой переменной:

```
>DifEq[ODESpline]:=proc(Eqs,Inits,Method,x1,n,i,k,x)
local AAA,NN,XX,nn,nnn,l,x0,Eqs1,Inits1,SS,SD,
SD1,SD11,t,xj,SD1j,XXik,SXXik,z1,BS:
AAA:=DifEq[SysCauchy_ConvNorm](Eqs,Inits):
NN:=AAA[1,1]:XX:=AAA[2]:nn:=nops(XX):
nnn:=(i)->nops(AAA[2,i]):x0:=AAA[6]:
xj:=(l)->x0+(x1-x0)*l/n:Eqs1:={op(AAA[4])}:
Inits1:={op(AAA[5])}:
if Method=45 then
SS:=dsolve(Eqs1 union Inits1,type=numeric,
method=rkf45,output=listprocedure):
elif Method=78 then
SS:=dsolve(Eqs1 union Inits1,type=numeric,
method=dverk78,output=listprocedure):
elif Method=stiff then
SS:=dsolve(Eqs1 union Inits1,type=numeric,
stiff=true,output=listprocedure):
elif Method=rosenbrock then
SS:=dsolve(Eqs1 union Inits1,type=numeric,
method=rosenbrock,output=listprocedure):
elif Method=taylor then
SS:=dsolve(Eqs1 union Inits1,type=numeric,
method=taylorseries,output=listprocedure):
elif Method=classic then
SS:=dsolve(Eqs1 union Inits1,type=numeric,
method=classical,output=listprocedure):
end if:
SD:=subs(SS,XX):SD1:=[rhs(SS[1]),SD]:SD1j:=(l)->SD1(xj(l)):
XXik:=(i,k,l)->[SD1j(l)[1],(SD1j(l)[2])[i,k]]:
SXXik:=(i,k)->[seq(XXik(i,k,l),l=0..n)]:
BS:=(i,k,x)->CurveFitting[Spline](SXXik(i,k),x):
```

[7] Совпадает с порядковым номером уравнения в системе ОДУ.

```
BS(i,k,x):
end proc:
```

Аналогично строится и 8-ми параметрическая процедура `DifEq[ODEBSpline]`, но в в ней использована процедура `Splines[ConvBSpline_Piece]`,позволяющая автоматически осуществлять вывод решений в виде кусочно - параметрически – заданных функций (V.7). Аналогично строятся и 10-параметрические программные процедуры `DifEq[ReODESpline]` и `DifEq[ReODEBSpline]` с перезагрузкой метода интегрирования в точке x2. Заметим, что все рассмотренные в этом разделе процедуры являются конечными и не требуют от пользователя применения каких-либо других процедур для получения решений в системы нелинейных ОДУ в форме сплайнов или B-сплайнов. Кроме того, решения выводятся в формате аналитической функции или упорядоченной пары функций в случае B-сплайна.

V.12 Пример компьютерного исследования системы нелинейных ОДУ

Для демонстрации комплекса программ рассмотрим простой пример получения решения системы ОДУ в виде сплайнов. В качестве системы ОДУ рассмотрим нелинейную систему оду 1-го порядка (V.13)-(V.14). Отметим, что выбор простой системы, а также малое количество интервалов сплайна, которое мы собираемся ввести, продиктованы лишь необходимостью краткости изложения. Тестирование показывает весьма хорошие результаты и при большом числе уравнений системы (десятки), высокого порядка уравнений системы (до 6-го) и большом количестве интервалов сплайнов (десятки). Однако, вывод таких результатов занимает весьма большое место, не совместимое с форматом книги. Вся операция численного решения системы с его сплайновым представлением производится простой командой:

```
F6:=(xi)->Splines[ODESpline]([diff(y(x),x)=y(x)/x,
diff(z(x),x)=-y(x)/z(x)],[y(-1/2)=-1/2,
z(-1/2)=sqrt(3)/2],45,1,12,1,2,xi):evalf(F6(xi),4);
```

Формат вывода результата интегрирования мы также упростили, сократив число выводимых значащих цифр до 4:

>F6(xi);

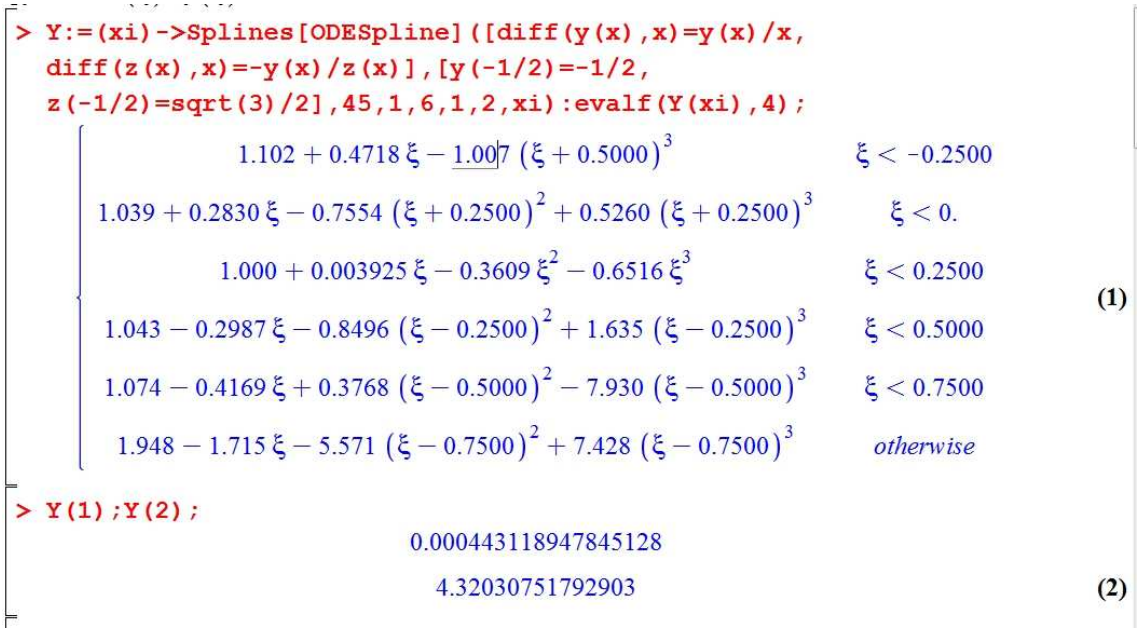


Рис.V.66 *Вывод численного решения задачи Коши (V.13) – (V.14) на отрезке [-1/2,1] методом Рунге-Кутта 4-5 порядков в виде кубического 6-кусочного сплайна с назначением имени независимой переменной ζ. Выведена функция z(x)=X2(ζ)*

Вычислим вторую производную от полученного решения в точке ζ=1/8:

```
>DifEq[SplineDiff](F6(t),t,1/8,2);
```

-1.210034572

Вычислим определенный интеграл от полученного решения $\int_0^1 \sin z(\xi) d\xi$:

```
>DifEq[SplineDefInt](F6(xi),xi,sin(xi),0,1);
```

.4596988780

Графическое представление решения интервале [-1/2,1] черным цветом в жестком формате производится стандартной командой Maple:

```
> plot([F6(t),sqrt(1-t^2)],t=-1/2..1,color=black,
style=[line,point],scaling=CONSTRAINED,title=
"Точное решение (точки) и численное (линия)
в формате 6-ти кусочного сплайна",titlefont=[TIMES,ROMAN,12]);
```

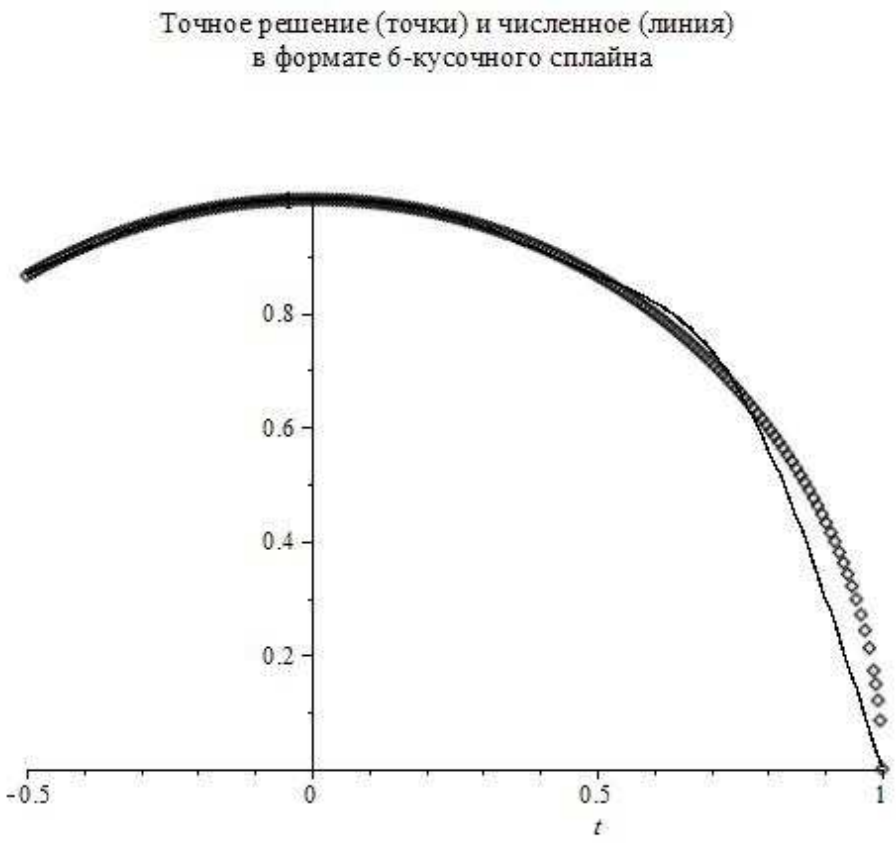


Рис.V.67 *Вывод графика численного решения задачи задачи Коши (V.13) – (V.14) на отрезке [-1/2,1] методом Рунге-Кутта 4-5 порядков для функции z(x)=X2(t) в виде кубического 6-ти кусочного сплайна.*

```
> plot([Z(t),sqrt(1-t^2)],t=-1/2..1,color=black,
style=[line,point],scaling=CONSTRAINED,title=
"Точное решение (точки) и численное (линия)
в формате 48-кусочного сплайна",titlefont=[TIMES,ROMAN,12]);
```

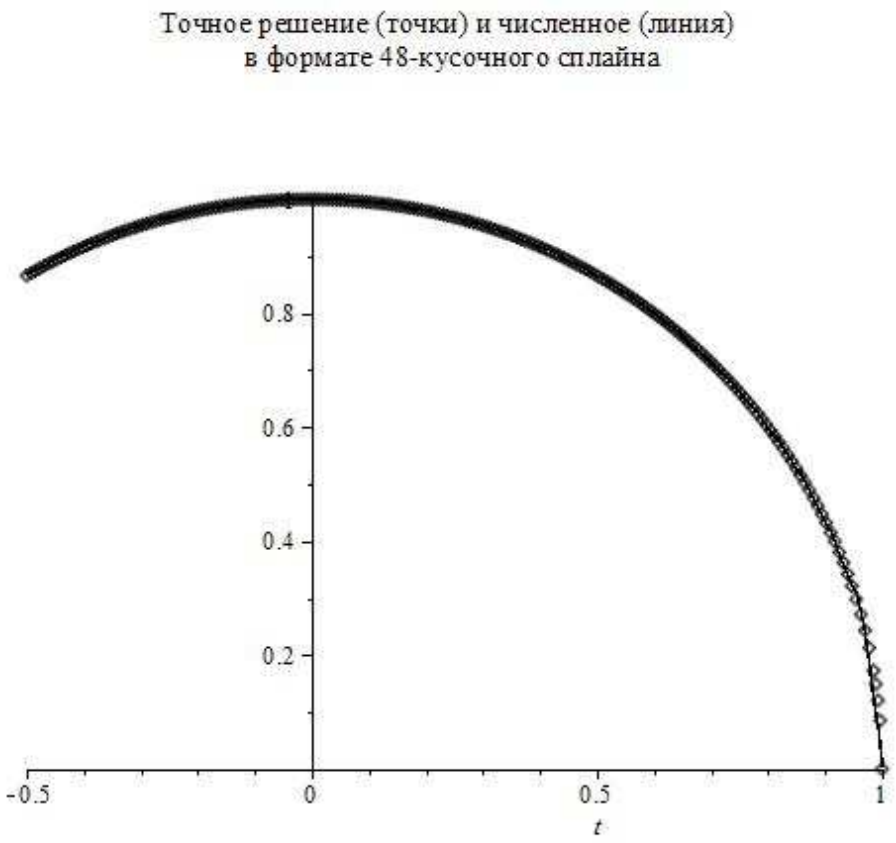


Рис.V.68 *Вывод графика численного решения задачи задачи Коши (V.13) – (V.14) на отрезке [-1/2,1] методом Рунге-Кутта 4-5 порядков для функции z(x)=X2(t) в виде кубического 48-ми кусочного сплайна.*

> plot(F6(t),t=-1/2..1,color=black,scaling=CONSTRAINED);

Несоответствие фигуры на Рис.V.67 дуге окружности вызвано лишь малым числом интервалов сплайна, продиктованным указанными выше соображениями. На Рис.V.68 продемонстрировано изображение сплайна, аналогичного F6(t), но 48-кусочного.

V.13 Компьютерная модель движения релятивистского заряда в электромагнитном поле с учетом магнитотормозного излучения

V.13.1 Динамические уравнения движения

Построим на основе созданных программных процедур компьютерную модель движения релятивистского заряда в сильном электромагнитном поле с учетом магнитотормозного излучения заряда. Такая задача имеет многочисленные приложения в теории электромагнитного ускорения элементарных частиц, релятивистской астрофизике, теории образования космических лучей. Пусть $\eta = Diag(-1,-1,-1,+1)$ - тензор Минковского. Здесь и в дальнейшем латинские символы принимают значения $i,j,.. = \overline{(1,4)}$, греческие -$\alpha,\beta,.. = \overline{(1,3)}$; $x^4 \equiv ct$, скорость света равна единице (c=*1*). Уравнения движения релятивистского заряда e в электромагнитном поле с учетом магнитотормозного излучения имеют вид [10]:

$$mc\frac{du^i}{ds} = \frac{e}{c}F^i_{.k}u^k + f^i, \tag{V.18}$$

где:

$$u^i = \frac{dx^i}{ds} \tag{V.19}$$

- 4-х мерный времениподобный единичный вектор скорости:

$$(u,u) \equiv \eta_{ik}u^iu^k = 1, \;\; \Rightarrow \;\; u_i\frac{du^i}{ds} = 0. \tag{V.20}$$

Вектор магнитотормозной силы, возникающей за счет реакции на магнитотормозное излучение, **f**, в отличие от общепринятой формы, запишем в виде:

$$f^i = \frac{2e^2}{3c}P^i_k\frac{d^2u^k}{ds^2}, \tag{V.21}$$

где мы ввели симметричный тензор ортогонального проектирования на направление ui - Pik :

$$P^i_k = \delta^i_k - u^iu_k; \quad P^i_ku^k \equiv 0; \quad P^i_kP^k_{\rm j} \equiv P^i_j. \tag{V.22}$$

Обычно уравнения движения (V.18) с магнитотормозной силой (V.21) сводят приближенно к дифференциальным уравнениям 2-го порядка, исполь-

зуя в качестве нулевого приближения уравнения (V.18) без учета торможения и получая, таким образом, выражения для вторых производных[8]:

$$f^i \ll \frac{e}{c}F^i{}_{.k}u^k \Rightarrow mc\frac{du^i}{ds} \approx \frac{e}{c}F^i{}_{.k}u^k \Rightarrow \frac{d^2u^i}{ds^2} \approx \frac{e}{mc^2}\frac{d}{ds}\left(F^i{}_{.k}u^k\right) \tag{V.23}$$

Мы этого делать не будем и получим *точные* динамические уравнения 3-го порядка. Подставляя (V.21) в уравнения движения (V.18), запишем их в виде:

$$P^i_k\frac{d^2u^k}{ds^2} = G^i \equiv \frac{3mc^2}{2e^2}\frac{du^i}{ds} - \frac{3}{2e}F^{\mathrm{i}}{}_{.k}u^k. \tag{V.24}$$

Вследствие (V.20) и антисимметричности тензора Максвелла выполняется тождество:

$$(G,u) \equiv G_iu^i = 0 \Rightarrow P^k_iG_k = G_i, \tag{V.25}$$

Поэтому, сворачивая уравнения (V.24) с тензором ортогонального проектирования, приведем их к эквивалентному виду:

$$P^i_k\xi^k = 0, \tag{V.26}$$

где

$$\xi^i = \frac{d^2u^i}{ds^2} - G^i. \tag{V.27}$$

Таким образом, задача сводится к нахождению нетривиальных решений однородной алгебраической системы линейных уравнений (V.26). Прямым вычислением можно показать, что вследствие соотношения нормировки 4-вектора скорости определитель матрицы P равен нулю:

$$\det(P^i_k) \equiv 0. \tag{V.28}$$

Поскольку ранг матрицы тензора является инвариантом, то выбирая синхронную систему отсчета, в которой ui = (0,0,0,1), получим для матрицы P выражение в этой системе отсчета:

$$\left\|P^i_k\right\| \stackrel{*}{=} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

[8] См., например, классическую книгу Л.Д.Ландау, Е.М. Лифшица [1].

вследствие чего

$$rank \left\| P^i_k \right\| = 3. \tag{V.29}$$

Но это означает, что существует всего одно нетривиальное решение системы (V.26) . С учетом (V.22) это решение легко находится:

$\xi^i = \alpha u^i$

Таким образом, мы получим:

$\frac{d^2u^i}{ds^2} = \alpha u^i + G^i$.

Для нахождения коэффициента α в этом соотношении свернем его с ui, учитывая соотношение нормировки (V.24) и (V.28). Тогда получим:

$\alpha = \eta_{ik} u^i \frac{d^2u^k}{ds^2}$.

Учитывая теперь дифференциальное следствие соотношения нормировки (V.24):

$\eta_{ik} \left(u^i \frac{d^2u^k}{ds^2} + \frac{du^i}{ds}\frac{du^k}{ds} \right) \equiv 0,$

найдем окончательно:

$$\frac{d^2u^i}{ds^2} = -\eta_{kl}\frac{du^k}{ds}\frac{du^l}{ds}u^i + \frac{3mc^2}{2e^2}\frac{du^i}{ds} - \frac{3}{2e}F^{\mathrm{i}}_{\cdot\, k}u^k. \tag{V.30}$$

Подставляя в (V.30) определение 4-скорости (V.19), получим искомые дифференциальные уравнения 3-го порядка, разрешенные относительно старших производных и описывающие движение релятивистского заряда в сильных электромагнитных полях с учетом магнитотормозного излучения.

V.13.2 Масштабирование уравнений

Переменное электромагнитное поле будем задавать тензором Максвелла [32][9]:

$$F_{ik} = \begin{pmatrix} 0 & -H_3 & H_2 & -E_1 \\ H_3 & 0 & -H_1 & -E_2 \\ -H_2 & H_1 & 0 & -E_3 \\ E_1 & E_2 & E_3 & 0 \end{pmatrix}; \; F^i_{\cdot\, k} = \begin{pmatrix} 0 & H_3 & -H_2 & E_1 \\ -H_3 & 0 & H_1 & E_2 \\ H_2 & -H_1 & 0 & E_3 \\ -E_1 & -E_2 & -E_3 & 0 \end{pmatrix};$$
$$F^{ik} = \begin{pmatrix} 0 & -H_3 & H_2 & E_1 \\ H_3 & 0 & -H_1 & E_2 \\ -H_2 & H_1 & 0 & E_3 \\ -E_1 & -E_2 & -E_3 & 0 \end{pmatrix} \tag{V.31}$$

[9] Здесь первый индекс – номер столбца матрицы, второй – номер строки.

где:

$$E_\alpha = E_\alpha(r,t); \quad H_\alpha = H_\alpha(r,t) \tag{V.32}$$

- 3-векторы напряженности электрического и магнитного поля – произвольные функции радиуса-вектора **r** и времени t. Таким образом, выполняется известное соотношение [см., например, [10]]: $\frac{1}{2}F_{ik}F^{ik} = H^2 - E^2$. Заметим, во-первых, что 4-вектор скорости ui в отличие от 3-вектора скорости vα

$$u^\alpha = \frac{v^\alpha}{c\sqrt{1-\frac{v^2}{c^2}}}; \quad u^4 = \frac{1}{\sqrt{1-\frac{v^2}{c^2}}} \Rightarrow u^\alpha \in (-\infty,+\infty);\ u^4 \in [0,+\infty) \tag{V.33}$$

является безразмерной величиной, так как четырехмерный интервал s имеет размерность длины: $ds^2 = (dx^4)^2 - (dr)^2 \geq 0$. Во-вторых, уравнения (V.29) имеют естественный масштаб – это классический радиус электрона: $r_0 = \frac{e^2}{mc^2} = 2.81 \cdot 10^{-13} cm$. В связи с этим произведем перенормировку масштабов, переходя к безразмерному интервалу σ и безразразмерным четырехмерным координатам ζi:

$$\sigma = {}^{s}\!/_{r_0}; \quad \xi^i = {}^{x^i}\!/_{r_0}. \tag{V.34}$$

Введем также безразмерные напряженности электрического, **e**, и магнитного, **b**, полей:

$$\varepsilon = \frac{e^3}{m^2c^4}E = r_0\frac{eE}{mc^2}; \quad \beta = \frac{e^3}{m^2c^4}H = r_0\frac{eH}{mc^2}. \tag{V.35}$$

Таким образом:

$$|\varepsilon| = {}^{r_0}\!/_{r_E}; \quad |\beta| = {}^{r_0}\!/_{r_H}, \tag{V.36}$$

где r_E, r_H - характерные радиусы кривизны траектории заряда в электрическом и магнитном полях:

$$k_E \equiv r_E^{-1} = \frac{e\,|E|}{mc^2}; \quad k_H \equiv r_H^{-1} = \frac{e\,|H|}{mc^2}. \tag{V.37}$$

Таким образом, уравнения (V.29) в безразмерных переменных принимают вид:

$$\frac{d^3\xi^i}{d\sigma^3} = -\eta_{kl}\frac{d^2\xi^k}{d\sigma^2}\frac{d^2\xi^l}{d\sigma^2}\frac{d\xi^{\,i}}{d\sigma} + \frac{3}{2}\frac{d^2\xi^i}{d\sigma^2} - \frac{3}{2}\Phi^{i}_{\cdot\,k}\frac{d\xi^{\,k}}{d\sigma}, \tag{V.38}$$

где $\Phi^i_{.k}$ - тензор Максвелла, определенный относительно нормированных напряженностей **e** и **b** аналогично тензору Максвелла (V.30). Из (V.29) и (V.37) следует, что при постоянных электромагнитных полях ($F^i_{.k} = Const$)

порядок релятивистских динамических уравнений понижается на 1. Заметим также, что релятивистски сильными электромагнитными полями можно считать поля с нормированными напряженностями порядка 1:

$$\max(|\varepsilon|, |\beta|) \sim 1. \tag{V.39}$$

В таких полях радиус кривизны траектории заряженной частицы сравним с ее классическим радиусом r0. Это – очень сильные поля, в которых указанное выше и обычно используемое приближение (V.23) [32] непригодно.

V.13.3 Результаты исследования движения релятивистского заряда

Приведем результаты исследования движения релятивистского заряда в скрещенных электромагнитных полях с учетом магнитотормозного излучения.

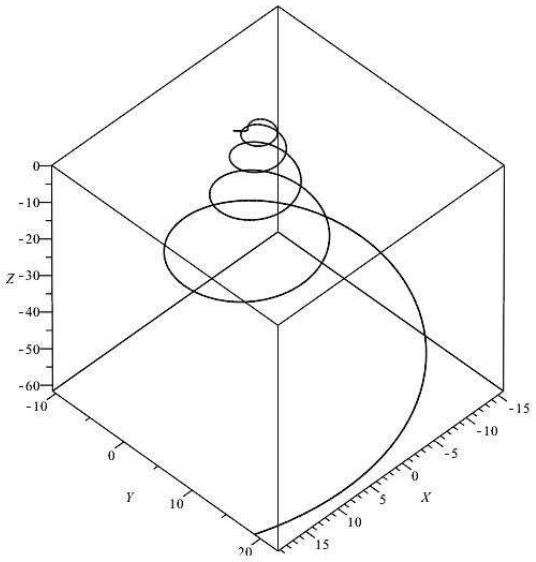


Рис.V.69 *Траектория заряда в сильных скрещенных электромагнитных полях. Магнитное поле направлено по оси OZ.*

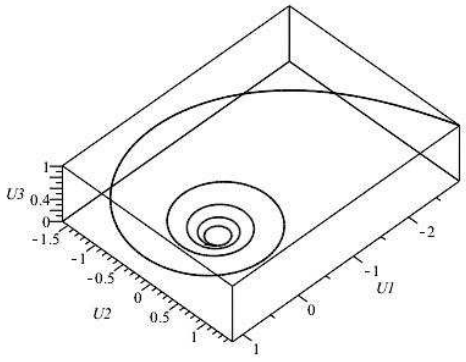


Рис.V.70 *Траектория заряда в сильных скрещенных электромагнитных полях. Магнитное поле направлено по оси OZ.*

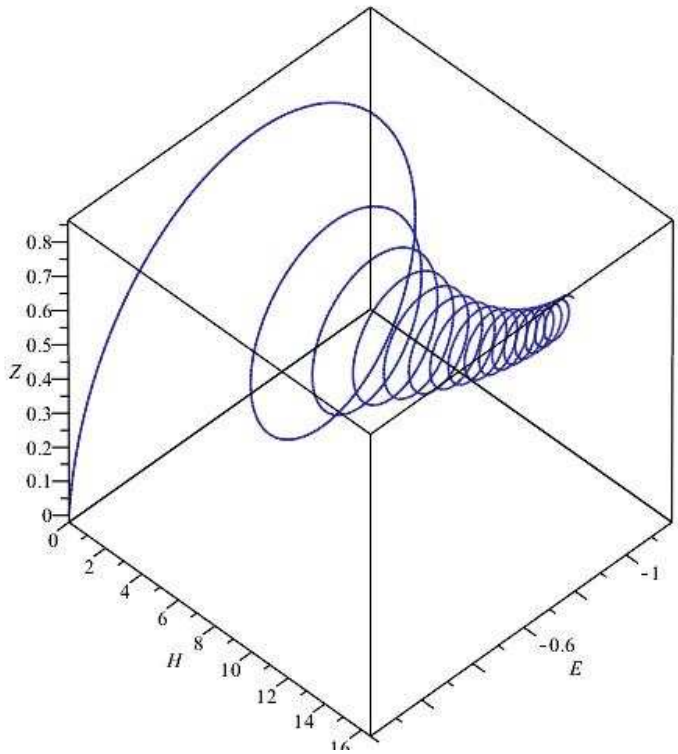


Рис.V.71 *Траектория заряда в сильных скрещенных электромагнитных полях. Магнитное поле направлено по оси OZ. Заметен дрейф заряда в направлении* $\mathbf{E} \times \mathbf{H}$

V.13.4 Заключение

Подводя результаты, отметим, что нами создан в системе компьютерной математики Maple система алгоритмов и комплекс программ автоматизированного численного решения задачи Коши для нелинейной системы произвольного числа обыкновенных дифференциальных уравнений произвольного порядка, разрешенных относительно старших производных и его аналитического представления в виде сплайнов и B-сплайнов. Одновременно создан комплекс программ, позволяющий осуществлять аналитические алгебраические и интегро - дифференциальные операции над сплайнами. Таким образом, комплекс представленных программ можно рассматривать как инструмент численно-аналитического исследования математических моделей нелинейных обобщенно-механических систем. Использование сплайновой интерполяции численных решений позволяет существенно (на порядки) сократить

время создания компьютерных динамических моделей. На Рис.V.72 приведен пример демонстрации такой модели.

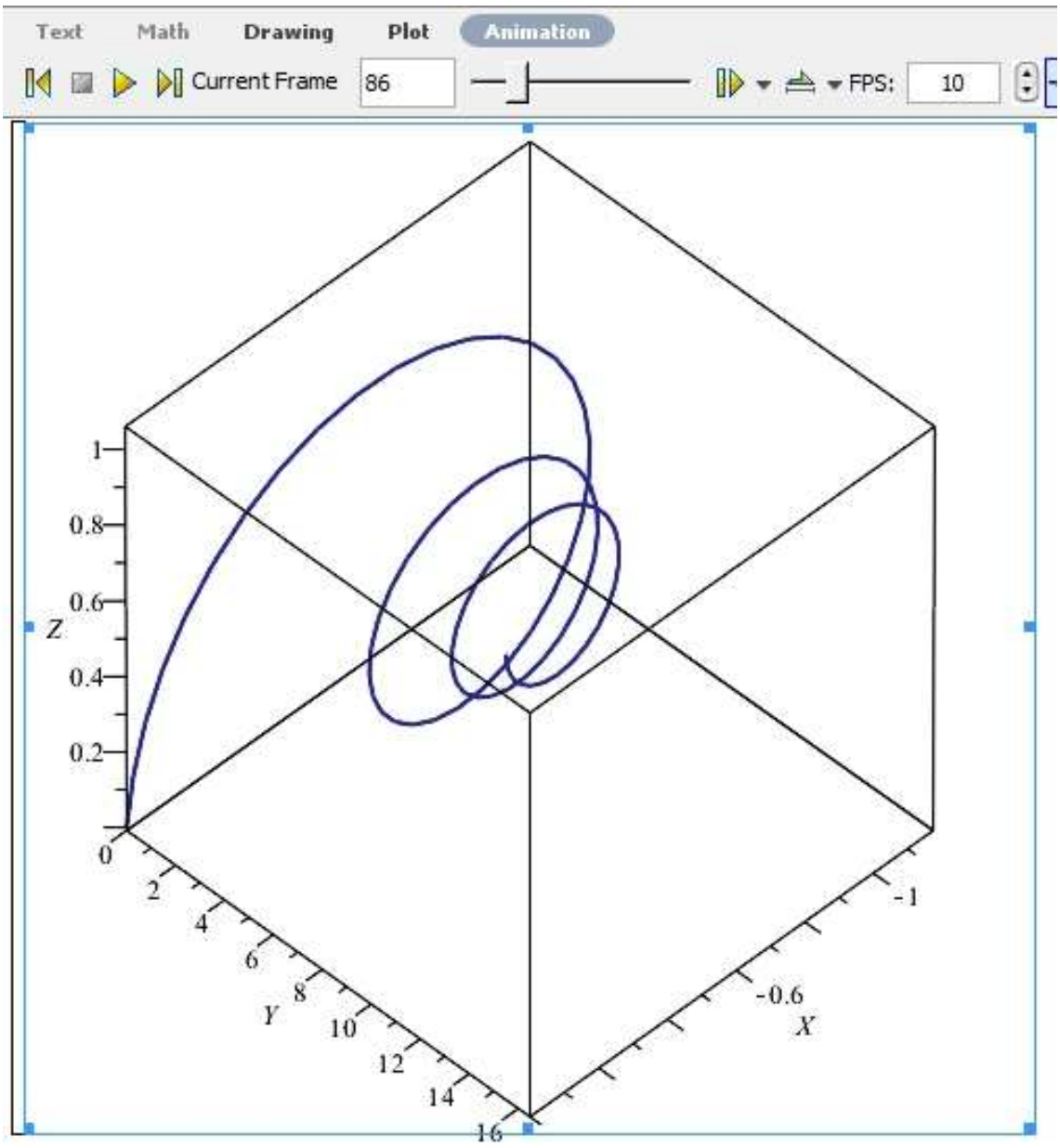


Рис.V.72 *Окно Maple 17 с динамической моделью движения заряда в сильных скрещенных электромагнитных полях. Показан 86-й кадр 400-кадрового трехмерного интерактивного фильма.*

Литература

[1] Самарский А. А., Михайлов А. П. Математическое моделирование: Идеи. Методы. Примеры. — 2-е изд., испр. — М.: Физматлит, 2005. - 320 с.

[2] Введение в математическое моделирование: Учеб. пособие / Под ред. П. В. Трусова. - М.: Логос, 2005. - 440 с.

[3] Матросов А.В. Maple 6. Решение задач высшей математики и механики. – СПб.: БХВ-Петербург, 2001, 528 с.

[4] Голоскоков Д.П. Уравнения математической физики. Решение задач в системе Maple. – СПб.: Питер, 2004, – 539 с.

[5] Голоскоков Д.П. Практический курс математической физики в системе Maple. – СПБ.: ООО «ПаркКом». - 2010. – 643 с.

[6] Дьяконов В.П. Maple 9.5/10 в математике, физике и образовании. – М.: Солон-Пресс, 2006, 720 с.

[7] Кирсанов М.Н. Графы в Maple. – М.: Физматлит. – 2007. – 292с.

[8] Кирсанов М.Н. Maple 13 и Maplet. Решение задач механики. М.: Физматлит, 2010, 349 с.

[9] *Матросов А.В.*. Maple 6. Решение задач высшей математики и механики. – СПб.: БХВ-Петербург. – 2001.– 528 с.

[10] *Дьяконов В.П.* Maple 7: Учебный курс. – СПб.: Питер. – 2002. – 672 с.

[11] *Дьяконов В.П.* Maple 9.5/10 в математике, физике и образовании. – М: Солон-пресс. – 2006. – 720 с.

[12] Дьяконов В.П. Компьютерная математика // Соровский образовательный журнал. 2001. № 1. С. 116–121.

[13] *Аладьев В.З.* Основы программирования в Maple. – Таллинн: Международная Академия Ноосферы, – 2006. – 301 с.

[14] *Gray A.* Modern Differential Geometry of Curves and Surfaces with Mathematica, Second Edition. – New-York:CRC Press. – 1997. – 432 p.

[15] *Аладьев В.З., Бойко В.К., Ровба Е.А.* Программирование в пакетах Maple и Mathematica: Сравнительный аспект. – Гродно: Изд-во Гродненский госуниверситет, Беларусь, – 2011. – 517 с.

[16] *Игнатьев Ю.Г.* Проблемы информационных технологий в математическом образовании. – Казань: Изд. - во ТГГПУ, 2005. – 118 с.

[17] Розакова Л.И. Методы математического и компьютерного моделирования в СКМ Maple графических и анимационных материалов для школьных курсов математики // Вестник ТГГПУ, 2010. № 3(21). С. 64-68.

[18] *Кирсанов М.Н.* Практика программирования в системе Maple. - М.: Издательский дом МЭИ. - 2011. - 208 с.

[19] Ignat'ev Yu.G., Chepkunova E.G. The moving semibounded magnetoactive plasma in field of a plane gravitational wave. // Gravitation & Cosmology, Vol.10, 2004г.,No 4, p. 123-127.

[20] Ignatyev Yu.G., Alsmadi K. A complect relativistic kinetic model of symmetry violation in an anisotropic expanding plasma. II. X-boson distribution function. // Gravitation and Cosmology, 2005, vol. 11, No 4, p. 363.

[21] Ignatyev Yu.G. and Miftakhov R.F. Statistical systems of particles with scalar interaction in cosmology. // Gravitation and Cosmology, Vol. 12 (2006), No 2-3, 179.

[22] Ignatyev Yu.G., Alsmadi K. A complect relativistic kinetic model of symmetry violation in an anisotropic expanding plasma. III. Specific entropy calculation. // Gravitation and Cosmology, 2007, vol. 13, No 2, p. 114.

[23] Игнатьев Ю.Г., Зиатдинов Р.А. Асимптотическое приближение модели Фоккера-Планка космологической эволюции сверхтепловых ультрарелятивистских частиц при наличии скейлинга взаимодействий. // Известия Вузов, сер. Физика, 2009, т. 42, No 2, с. 87.

[24] Игнатьев Ю.Г., Эльмахи Н. Динамическая модель сферических возмущений во вселенной Фридмана. III. Автомодельные решения. // Известия Вузов, сер.Физика, 2008, т. 42, №1, с. 69.

[25] Ignatyev Yu.G., Ignatyev D.Yu.. Kinetics of the nonequilibrium Universe. III. Stability of Nonequilibrium Scenario. // Gravitation and Cosmology, 2008, vol. 14, No 4, p.286-292.

[26] Ignatyev Yu.G., Agafonov A.A. Bremsstrahlung Response of a Homogeneous Magnetoactive Plasma to a Gravitational Wave. // Gravitation and Cosmology, Gravitation and Cosmology, 2010, Vol. 16, No. 1. p.16-24.

[27] Игнатьев Ю.Г. Релятивистская кинетика неравновесных процессов в гравитационных полях. - Казань: Фолиант. – 2010. - 505 с.

[28] Игнатьев Ю.Г. Использование аналитических возможностей пакета Maple для создания программ аналитического тестирования, самотестирования и генерации индивидуальных заданий в курсах высшей математики. // В сб. «Проблемы информационных технологий в математическом образовании» под ред. Ю.Г. Игнатьева. Казань: Изд-во ТГГПУ. - 2005. – с. 9-24.

[29] Адиятуллина Г.Р. Система аналитического тестирования в форме маплетов. //Системы компьютерной математики и их приложения. – Смоленск: Изд-во СмолГУ. – 2010. Вып. 11. – с. 5-8.

[30] Адиятуллина Г.Р., Игнатьев Ю.Г. Принципы моделирования системы аналитического тестирования знаний на основе системы компьютерной математики Maple. // Вестник ТГГПУ, – 2010, вып. 2(20). – с. 6-12.

[31] Гинзбург В.Л., Рухадзе А.А. Волны в магнитоактивной плазме. – М.: Наука. – 1970. – 368 с.

[32] Ландау Л.Д., Лифшиц Е.М. Теория поля. – М.: Наука, 1973. – 504 с.

[33] Синг Дж.Л. Общая теория относительности. – М.: ИЛ. – 1963. – 432 с.

[34] Петров А.З. Новые методы в общей теории относительности. – М.: Наука. – 1966. – 542 с.

[35] Б.А. Дубровин, С.П. Новиков, А.Т.Фоменко. Современная геометрия. – М.: Наука, 1979, 769 с.

[36] Арнольд В.И., Ильяшенко Ю.С. Обыкновенные дифференциальные уравнения. // Современные проблемы математики. Фундаментальные направления. т. 1. М.: Итоги науки и техники, 1985, 244 с.

[37] Богоявленский О.И. Методы качественной теории динамических систем в астрофизике и газовой динамике. – М.: Наука, 1980, 320 с.

[38] Глушков В.М., Бондарчук В.Г., Гривченко Т.А.. Аналик – алгоритмический язык для описания процессов с использованием аналитических преобразований. // Кибернетика. – 1971. - № 3, с. 127 - 152.

[39] Дэвенпорт Дж., Сирэ И., Турнье Э. Компьютерная алгебра. - М.: Мир. – 1991. - 352 с.

[40] Акритас А. Основы компьютерной алгебры. - М.: Мир. - 1994. - 262 с.

[41] Грэхэм Р., Кнут Д., Поташник О. Конкретная математика. Основание информатики. – М.: Мир. – 1998. - 324 с.

[42] Дьяконов В.П. Компьютерная математика. Теория и практика. – М.: Нолидж. - 2001. – 396 с.

[43] Дьяконов В.П. Система MathCAD. Справочник. М.: Радио и связь. 1993. - 128 с.

[44] Дьяконов В.П. Компьютерная математика. //Соросовский образовательный журнал. – **7**, No 1. – 2001. – с. 116-121.

[45] Дьяконов В.П. Справочник по MathCAD PLUS 6.0 PRO. – М.: СК-ПРЕСС. – 1997. – 336 с.

[46] Дьяконов В.П. Справочник по MathCAD PLUS 7.0 PRO. – М.: СК-ПРЕСС. – 1998. – 352 с.

[47] Дьяконов В.П. Энциклопедия Mathcad 2001i и Mathcad 11. – М.: СОЛОН-Пресс. – 2004.–646 с.

[48] Дьяконов В.П. MathCAD 11/12/13 в математике. Справочник. – М.: Горячая линия. – Телеком. – 2007. – 584с.

[49] Ефремов Л.В.. Практика вероятностного анализа надежности техники с применением компьютерных технологий. – СПб.: Наука. – 2008. – 216 с.

[50] Ефремов Л.В.. Теория и практика исследований крутильных колебаний силовых установок с применением компьютерных технологий. – СПб.: Наука. – 2008. – 276 с.

[51] Охорзин В.А.. Прикладная математика в системе MATHCAD. Учебное пособие. 3-е изд. – СПб.: Лань. – 2009. – 352 с.

[52] Охорзин В.А.. Компьютерное моделирование в системе Mathcad. – М.: Финансы и статистика. – 2006. – 144с.

[53] Охорзин В.А.. Оптимизация экономических систем. Примеры и алгоритмы в среде Mathcad. – М.: Финансы и статистика. – 2005. – 144с.

[54] Ивановский Р. Теория вероятностей и математическая статистика. Основы, прикладные аспекты с примерами и задачами в среде Mathcad. – М.: БХВ-Петербург/ – 2008/ – 528с.

[55] Фриск В.В. Mathcad. Расчеты и моделирование цепей на ПК. – М.: Солон-Пресс. – 2006. – 242 с.

[56] Гурский Д. Вычисления в MATHCAD 12. – С-Пб: Питер. – 2006. – 544с.

[57] Кирьянов Д.В.. Mathcad 13. – С-Пб: БХВ-Петербург. – 2006. 598 с.

[58] Кирьянов Д.В.. Самоучитель Mathcad 13. – С-Пб: БХВ-Петербург. – 2006. – 528 с.

[59] Поршнев Д.С., Беленкова И.С.. Численные методы на базе Mathcad. – С-Пб: БХВ-Петербург. – 2005. – 456 с.

[60] Дьяконов В.П. Mathematica 2.0 под MS-DOS и под Windows. // Монитор-Аспект.-1993.- **№ 2**.- с.52-74.

[61] Дьяконов В.П. Системы символьной математики Mathematica 2 и Mathematica 3. – М.: СК ПРЕСС/PC Week. – 1998. - 484с.

[62] Капустина Т.В. Компьютерная система «Mathematica 3.0 для пользователей». М.: - Солон-Р. - 1999. - 302 с.

[63] Дьяконов В.П. Mathematica 4 с пакетами расширений. – М.: Нолидж. – 2000. – 608 с.

[64] Дьяконов В.П. Mathematica 3/4 с пакетами расширений. – М.: Нолидж. – 2004. – 612с.

[65] Дьяконов В.П. Mathematica 4: Учебный курс. – СПб.: Питер. – 2001. - 624 с.

[66] Дьяконов В.П. Mathematica 5/6/7. Полное руководство. – М.: ДМК Пресс. – 2009. – 624 с.

[67] Половко А.М. Mathematica для студентов. – СПб.: БХВ Санкт-Петербург. – 2007. – 524 с.

[68] Шмидский Я.К. Mathematica 5%. Самоучитель. – М. Издательский дом «Вильямс». – 2004. - 402 с.

[69] Дьяконов В.П. Справочник по применению PC MatLAB. – М.: Наука, Физматлит. – 1993. – 488 с.

[70] Дьяконов В.П. MATLAB 6.5 SP1/7.0+Simulink5/6. Основы применения. - М.: СОЛОН Пресс. – 2005. – 602 с.

[71] Дьяконов В.П. MATLAB 6.5 SP1/7.0+Simulink5/6. Работа с изображениями. - М.: СОЛОН Пресс. – 2005. – 588 с.

[72] Дьяконов В.П., Круглов В.В. MATLAB 6.5 SP1/7/7 SP1/7 SP2+Simulink5/6. Инструменты искусственного интеллекта и биоинформатики. - М.: СОЛОН Пресс. – 2006. – 512 с.

[73] Дьяконов В.П.. Derive - жемчужина символьной математики. // Монитор-Аспект.-1993. **№ 2**.-с.36 -51.

[74] Дьяконов В.П.. Справочник по применению системы Derive. М.: Наука. Физматлит. – 1996. - 258 с.

[75] Дьяконов В.П.. Справочник по системе символьной математики Derive. – М.: СК - ПРЕСС, – 1998. – 256с.

[76] Дьяконов В.П.. Системы компьютерной алгебры Derive. Самоучитель. М.: Солон-Р. – 2002. - 442с.

[77] Говорухин В.Н. , Цибулин В. Г.. Введение в Maple. Математический пакет для всех. - М.: Мир, - 1997. 287 с.

[78] Дьяконов В.П. Математическая система Maple V R3/R4/R5. - М.: Солон. - 1998. - 400 с.

[79] Дьяконов В.П. Maple 6. Учебный курс. – СПб.: ПИТЕР. – 2001. - 586 с.

[80] Дьяконов В.П. Maple 7. Учебный курс. – СПб.: ПИТЕР. – 2001. - 588 с.

[81] Дьяконов В.П. Maple 8 в математике, физике и образовании. М.: СОЛОН-Пресс. – 2003. - 612 с.

[82] Дьяконов В.П. Maple 9 в математике, физике и образовании. М.: СОЛОН-Пресс. – 2004. - 624 с.

[83] Gray A. Modern Differential Geometry of Curves and Surfaces with Mathematica, Second Edition. – New-York:CRC Press. – 1997. – 432 p.

[84] Gray A., Mezzino M., Pinsky M.. Introdution to Ordinary Differential Equations with Mathematica. – John Wiley & Sons. – 1998. – 544 p.

[85] Дьяконов В.П. Mathematica 4.1/4.2/5.0 в математических и научно - технических расчетах. – М.: СОЛОН-Пресс. – 2004. Аннотация. 628с.

[86] Эдвардс Чарльз Генри, Пенни Дэвид Э.. Дифференциальные уравнения и краевые задачи: моделирование и вычисление с помощью Mathematica, Maple и MATLAB. 3-е издание. – Киев.: Диалектика-Вильямс. – 2007. – 584с.

[87] Тан К. Символьный C ++: Введение в компьютерную алгебру с использованием объектно - ориентированного программирования. – М.: Мир. – 2001. – 432 с.

[88] Семененко М. Математическое моделирование в MathCad. – М.: Альтекс-А. – 2003. – 264 с.

[89] Бертяев В.Д.. Теоретическая механика на базе Mathcad. Практикум. – С-Пб: БХВ-Петербург. – 2005. – 752с.

[90] Поршнев Д.С.. Компьютерное моделирование физических процессов с использованием пакета Mathcad. Учебное пособие. – М.: Горячая линия - Телеком. – 2002. 334 с.

[91] Глушаков С., Жакин И., Хачиров Т.. Математическое моделирование. Mathcad 2000. Matlab 5.3. – М.: АСТ. – 2001. – 432 с.

[92] Каганов В. Радиотехника + компьютер + Mathcad. – М.:Горячая линия - Телеком. – 2001. 346 с.

[93] Шестаков Н., Власов А.. Расчеты процессов обработки металлов давлением в среде Mathcad. Учебное пособие. – М.:МГИУ. – 2000. –182с.

[94] Панько М.. Расчет и моделирование автоматических систем регулирования в среде Mathcad. – М.: Изд-ство МЭИ. – 2001. 156 с.

[95] Очков В.. Физические и экономические величины в Mathcad и Maple. – М.: Финансы и статистика. – 2002. – 372 с.

[96] Тарасевич Ю. Математическое и компьютерное моделирование. Вводный курс. – М.: Едиториал-УРСС. – 2001. – 342с.

[97] *Игнатьев Ю.Г.* Пользовательские графические процедуры для создания анимационных моделей нелинейных физических процессов // Системы компьютерной математики и их приложения. Материалы 10-й международной конференции. – 2009. – вып. 10. – Смоленск: изд-во СмолГу. – С. 43 – 44.

[98] *Игнатьев Ю.Г., Абдулла Х.Х.* Математическое моделирование нелинейных обобщенно-механических систем в системе компьютерной математики Maple // Известия ВУЗов, Поволжский регион, физ.-мат. науки. – 2010. – №2 (14) – С. 66–75.

[99] *Игнатьев Ю.Г., Абдулла Х.Х.* Математическое моделирование нелинейных электродинамических систем в системе компьютерной математики Maple // Вестник ТГГПУ. – 2010. – вып. 2(20) – С. 22–27/

[100] *Игнатьев Ю.Г., Абдулла Х.Х.* Математическое моделирование нелинейных обобщённо - механических систем в системе компьютерной математики MAPLE // Вестник РУДН. – 2010. – № 4(3). – С. 65–76.

[101] *Игнатьев Ю.Г., Исрафилова Э.Г.* Математическое моделирование объектов дифференциальной геометрии кривых в системе компьютерной математики MAPLE // Вестник ТГГПУ. – 2011. – № 4(26). – С. 11–16.

[102] Бушкова В.А. Библиотека программных процедур создания управляемой оснащенной динамической визуализации геодезических линий в СКМ Maple // Вестник ТГГПУ, 2011. №4(26). С. 8–10.

[103] *Черкасова В.В.* Мультипликативный интеграл в дифференциальной геометрии и прикладных задачах.// Вестник ТГГПУ. – 2010. – № 3(21). – С. 57–62.

[104] *Игнатьев Ю.Г., Черкасова В.В.* Программа динамической визуализации процесса качения шара произвольного радиуса вдоль произвольной кривой произвольной параметризованной поверхности с динамическим

выводом числовой информации о моментальных координатах векторов адаптированного репера с началом в центре шара, гауссовой и средней кривизн, а также матрицы преобразования репера между начальным и конечным положениями шара. // св. о гос. рег. прог. для ЭВМ Рос. Фед. № 2012615999, 3 июля 2012 г. Федеральная служба по интеллектуальной собственности; RU ОБПБТ №3(80). 20.09.2012. - с. 699.

[105] *Игнатьев Ю.Г., Самигуллина А.Р.* Программное обеспечение теории кривых второго порядка в пакете компьютерной математики // Вестник ТГГПУ. – 2010. – вып. 4(26) – С. 24–29.

[106] *Игнатьев Ю.Г., Самигуллина А.Р.* Программа автоматизированного полного исследования общего уравнения второго порядка на плоскости с выводом результатов исследования в табличном и графическом форматах всех элементов кривых, описываемых общим уравнением, включая формулы их преобразования к каноническому виду, изображения директрис, асимптот, фокусов, исходной и преобразованной системы координат, в математическом пакете Maple. //св. о гос. рег. прог. для ЭВМ Рос. Фед. № 2012611664, 14 февраля 2012 г. Федеральная служба интеллектуальной собственности; RU ОБПБТ №2(79). 20.06.2012.

[107] Игнатьев Ю.Г., Самигуллина А.Р. Программа точного вычисления фундаментальных решений системы линейных алгебраических уравнений произвольного порядка и представления их в стандартном, списочном виде в математическом пакете Maple // Свидетельство о государственной регистрации программ для ЭВМ РФ №2011614976. 2011. Бюл.: RU ОБПБТ, № 3(76). С. 547.

[108] Игнатьев Ю.Г., Самигуллина А.Р. Библиотека программных процедур для методического обеспечения курса высшей алгебры в системе компьютерной математики «Maple // Вестник ТГГПУ, 2011.№1(23). С. 20 – 24.

[109] *Игнатьев Ю.Г., Бушкова В.А.* Программа автоматизированного построения геодезических линий на произвольной параметризованной поверхности и их оснащенной динамической визуализации с автоматической оптимизацией графических параметров в системе компьютерной математики Maple.: св. о гос. рег. прог. для ЭВМ Рос. Фед. № 2012612380 от 30.05.12.

[110] *Игнатьев Ю.Г., Бушкова В.А.* Программный комплекс автоматизированного нахождения и построения геодезических линий и геодезических трубок в трехмерных (локально евклидовых) римановых пространствах по заданной метрике и их оснащенной динамической визуализации с автоматической оптимизацией графических параметров в системе компьютерной математики Maple.: св. о гос. рег. прог. для ЭВМ Рос. Фед. № 2012613444 от 28.05.13.

[111] *Игнатьев Ю.Г., Бушкова В.А.* Программный комплекс автоматизированного нахождения и построения геодезических линий и геодезических трубок в четырехмерных (псевдоевклидовых) римановых пространствах по заданной метрике и их оснащенной динамической визуализации с автоматической оптимизацией графических параметров в системе компьютерной математики Maple.: св. о гос. рег. прог. для ЭВМ Рос. Фед. № 2012613445 от 28.05.13.

[112] Игнатьев Ю.Г., Михайлов М.Л. Программный комплекс автоматизированного упорядочения двумерных и трехмерных массивов по первой (двум первым) координате(ам) и построения линии (поверхности) отформатированного массива, а также автоматизированного создания двумерных и трехмерных массивов по результатам численного интегрирования однократных и двукратных определенных интегралов с выводом результатов в виде двух(трех)-мерных графиков с управляемыми параметрами в системе компьютерной математики Maple // Свидетельство о государственной регистрации программ для ЭВМ РФ № 2012611332. 2012.

[113] Игнатьев Ю.Г. Программа автоматизированного распознавания системы обыкновенных дифференциальных уравнений произвольного порядка, разрешенных относительно старших производных, автоматизированного управляемого численного интегрирования задачи Коши для нее и выводом решений в функциональной, сплайновой форме в системе компьютерной математики Maple // Свидетельство о государственной регистрации программ для ЭВМ № 2012613751, 20 июня 2012 г. Федеральная служба по интеллектуальной собственности. RU ОБПБТ №3(80). 20.09.2012. - с. 606

[114] Самигуллина А.Р. Математическое моделирование объектов линейной алгебры ианалитической геометрии в системе компьютерной математики Maple // Вестник ТГГПУ, №3 (21), 2010. С.69 – 74.

[115] Дьяконов В.П.. VisSim+Mathcad+MATLAB. Визуальное математическое моделирование. – М.: СОЛОН-Пресс. – 2004. – 482 с.

[116] Дьяконов В.П. Mathematica 4.1/4.2/5.0 в математических и научно-технических расчетах. – М.: СОЛОН ПРЕС. – 2004. - 608 с.

[117] Дьяконов В.П. MATLAB 6.5 SP1/7.0+Simulink5/6 в математике и моделировании. - М.: СОЛОН Пресс. – 2005. – 598 с.

[118] Дьяконов В.П. MATLAB 6.5 SP1/7.0+Simulink5/6. Обработка сигналов и проектирование фильтров. - М.: СОЛОН Пресс. – 2005. – 534 с.

[119] Обыкновенные дифференциальные уравнения. – М.: Наука, Физматлит. – 1965. – 332 с.

[120] Эльсгольц Л.Э. Дифференциальные уравнения и вариационное исчисление. – М.: Наука, Физматлит. – 1965. – 424 с.

[121] Федорюк М.В. Обыкновенные дифференциальные уравнения. Санкт-Петербург·Москва·Краснодар: Лань. – 2003. – 448 с.

[122] Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. - М.: Бином, 2001,525 с.

[123] http://vuz.exponenta.ru/

[124] Абдулла Х.Х. Программные процедуры численного решения задачи Коши для нормальных систем обыкновенных дифференциальных уравнений в пакете компьютерной математики Maple. // Труды математического центра имени Н.И. Лобачевского– Казань: Казанское математическое общество, 2009, т. 39, с. 388.

[125] Норден А.П. Дифференциальная геометрия. – М.: Учпедгиз. – 1948. – 216 с.

[126] Игнатьев Ю.Г. Дифференциальная геометрия кривых поверхностей в евклидовом пространстве [Текст: Электронный ресурс]: учебное пособие. IV семестр: курс лекций для студентов математического факультета // <URL:http://libweb.ksu.ru/ebooks/05_120_000327.pdf.

[127] Стечкин С.Б., Субботин Ю.Н. Сплайны в вычислительной математике. - М.: Наука. - 1976. - 248 с.

[128] Fox L. and Mayers D.F. Numerical Solution of Ordinary Differential Equations for Scientists and Engineers. - New-York: Springer, 1987, 624 p.

[129] Филиппов А.Ф. Сборник задач по дифференциальным уравнениям. – Ижевск.: НИЦ «Регулярная и хаотическая динамика», 2000, 176 с.

[130] Роджерс Д., Адамс Дж. Математические основы машинной графики. — М.: Мир, 2001, 452 с.

[131] Brinks R. On the convergence of derivatives of B-splines to derivatives of the Gaussian function, Comp. Appl. Math., **27**, p. 1-17, 2008.

[132] Корнейчук Н.П., Бабенко В.Ф., Лигун А.А. Экстремальные свойства полиномов и сплайнов. — К.: Наукова думка, 1992. — 304 с.

[133] Фихтенгольц Г.М. Курс дифференциального и интегрального исчисления. Том I. – М.: Наука, Физматлит. – 1966. – 608 с.

[134] Фихтенгольц Г.М. Курс дифференциального и интегрального исчисления. Том II. – М.: Наука, Физматлит. – 1966. – 800 с.

[135] Ландау Л.Д., Лифшиц Е.М. Механика. – М.: Наука. Физматлит. – 1965. – 204 с.

[136] J.L. Synge. Classical dynamics. Springer-Verlag/Berlin · Göttingen · Heidelberg. – 1960 – 356 p. Русский перевод: Синг Дж.Л. Классическая динамика. – М.: ГИФМЛ. – 1963. 448 с.

[137] Lee E.T.Y. A Simplified B-Spline Computation Routine. Computing (Springer-Verlag) 29 (4): 365-371. doi:10.1007/BF02246763.

[138] Lee E.T.Y. (1986). Comments on some B-spline algorithms. Computing (Springer-Verlag) 36 (3): 229-238. doi:10.1007/BF02240069.

[139] Yamaguchi F. Curves and Surfaces in Computer Aided Geometric Design. New-Yuork:- Springer, 1988, 476 p.

[140] Лоу А.М. , Кельтон Д.В. Имитационное моделирование. – СПБ.: – Питер-BHV. 2004 – 848 с.

[141] Хемди А. Т. Имитационное моделирование. // Введение в исследование операций. — 7-е изд. —- М.: «Вильямс»/ – 2007. — с. 697-737.

[142] Строгалев В.П., Толкачева И.О. Имитационное моделирование. -— М.: Изд-во МГТУ им. Баумана. – 2008. — с. 697-737.

[143] Рашевский П.К. Риманова геометрия и тензорный анализ. – М.: Наука. – 1964. – 664 с.

[144] Эйзенхарт Л.П. Риманова геометрия. – М.: ГИИЛ. – 1948. – 316 с.

[145] Схоутен Я.А. Тензорный анализ для физиков. – М.: Наука. – 1965. – 456 с.

[146] Игнатьев Ю.Г.// Вопросы современной математики и информационных технологий в математическом образовании. Сб. трудов под ред. Ю.Г. Игнатьева. – Казань: Изд-во КГПУ. – 2004. – 164 с.

[147] Игнатьев Ю.Г. Пользовательские графические процедуры для создания анимационных моделей нелинейных физических процессов. // Системы компьютерной математики и их приложения. Смоленск: Изд-во СмолГу, 2009, Выпуск 10, с. 43.

[148] Абдулла Х.Х. Визуализация математических моделей нелинейных механических систем в системах компьютерной математики. // Системы компьютерной математики и их приложения. Смоленск: Изд-во СмолГу, 2009, Выпуск 10, с. 108.

[149] Ахиезер А.И., Берестецкий В.Б. Квантовая электродинамика. – М.: Наука, 1969, 624 с.

[150] Игнатьев Ю.Г. Проблемы информационных технологий в математическом образовании // Учебное пособие под редакцией Ю.Г. Игнатьева. – Казань: ТГППУ, 2005. - 118 с.

[151] Самигуллина А.Р. Математическое моделирование объектов линейной алгебры и аналитической геометрии в системе компьютерной математики Maple // Вестник ТГГПУ, № 3 (21), 2010. — с. 69 – 74.

[152] Игнатьев Ю.Г., Самигуллина А.Р. Библиотека программных процедур для методического обеспечения курса высшей алгебры в системе компьютерной математики «Maple» // Вестник ТГГПУ, 2011, № 1(23). – с. 20–24.

[153] Игнатьев Ю.Г., Самигуллина А.Р. Программа точного вычисления фундаментальных решений системы линейных алгебраических уравнений произвольного порядка и представления и в стандартном, списочном виде в математическом пакете Maple // Свидетельство о государственной регистрации программ для ЭВМ № 2011614976, 24 июня 2011 г., Федеральная служба по интеллектуальной собственности.

[154] Атанасян Л.С., Базылев В.Т.. Геометрия. Часть I. - М.: Просвещение, 1987. - 352 с.

[155] Игнатьев Ю.Г. Аналитическая геометрия. Часть II. Аффинные и евклидовы пространства. Учебное пособие. II семестр. // 2013, Казань, - 188 с. $http://www.kpfu.ru/docs/F1788036257/Aff_Evk13.pdf$.

[156] Игнатьев Ю.Г., Самигуллина А.Р. Библиотека программных процедур для методического обеспечения курса высшей алгебры в системе компьютерной математики «Maple» // Вестник ТГГПУ, 2011, 1(23). - с. 20-24

[157] Игнатьев Ю.Г., Самигуллина А.Р. Программный комплекс программных процедур по высшей математике в прикладном математическом пакете Maple // Свидетельство о государственной регистрации программ для ЭВМ № 2013617288, 8 августа 2013г. Федеральная служба интеллектуальной собственности.

[158] Игнатьев Ю.Г., Самигуллина А.Р. Программное обеспечение теории кривых второго порядка в пакете компьютерной математики //Вестник ТГГПУ, 2011, 4(26), - с. 24-29.

[159] Игнатьев Ю.Г., Самигуллина А.Р. Программа автоматизированного полного исследования общего уравнения второго порядка на плоскости с выводом результатов исследования в табличном и графическом форматах всех элементов кривых, описываемых общим уравнением, включая формулы их преобразования к каноническому виду, изображения директрис, асимптот, фокусов, исходной и преобразованной системы координат, в математическом пакете Maple программа для ЭВМ Свидетельство о государственной регистрации программ для ЭВМ № 2012611664, 14 февраля 2012 г. Федеральная служба интеллектуальной собственности. RU ОБПБТ №2(79). 20.06.2012

Набор монографии осуществлен в издательском пакете LaTeX2ε в научно-исследовательской лаборатории «Информационных технологий в математическом образовании» КФУ.

Разработка авторского LaTeX-стиля оформления - Ю.Г.Игнатьев

Техническая редакция, набор и верстка: Ю.Г.Игнатьев.

Оформление обложки - Ю.Г. Игнатьев

Объем монографии 298 страниц ,библиография: 159 наименований, иллюстраций – 128.

Отпечатано с готового оригинал-макета
в типографии Казанского университета

Подписано в печать 15.08.12. Формат 60×84/8
Печать офсетная. Бумага офсетная. Печ. л. 19. Тираж 500 экз.

Казанский университет

420008, г. Казань, ул. Профессора Нужина, 1/37