МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО

ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«Национальный исследовательский Нижегородский государственный

университет им. Н.И. Лобачевского»

**С.Н.Стребуляев**

**Н.Л. Oрлова**

COMPUTER ALGEBRA «MAPLE» IN

ENGINEERING

КОМПЬЮТЕРНАЯ АЛГЕБРА «MAPLE» В

ИНЖЕНЕРИИ

Учебно-методическое пособие

Рекомендовано методической комиссией ИИТММ

для иностранных студентов ННГУ, обучающихся по направлению подготовки

02. 03. 02 “Фундаментальная информатика и информационные

технологии”

Нижний Новгород

2018

УДК 519. (075)

ББК В19

С-84

С 84 Стребуляев С.Н., Орлова Н.Л. КОМПЬЮТЕРНАЯ АЛГЕБРА «MAPLE» В ИНЖЕНЕРИИ: учебно-методическое электронное пособие. – Нижний Новгород: Ниже­город­ский госуни­верситет, 2018. – 152 с.

Рецензент: профессор кафедры «Программная

инженерия», д.ф.-м.н., профессор **Федоткин М.А.**

Учебно-методическое пособие разработано в соответст­вии с требованиями учебной программы по дисциплинам «Математические модели в естествознании» и «Методы оптимизации». В пособии приведены алгоритмы, программные средства , которые использовались авторами работы в течение ряда лет для решения инженерных задач прикладной математики. Представленный материал служит для закрепления лекционного материала и способствует более эффективному использованию вычислительной техники при решении конкретных задач динамики систем, теории колебаний и теории автоматического регулирования.

Учебно-методическое пособие предназначено для иностранных студентов четвертого курса, обучающихся по направлению «Фундаментальная информатика и информационные технологии» в Институте информационных технологий, математики и механики ННГУ им. Н.И. Лобачевского.

УДК 519.(075)

ББК 22

**© Нижегородский государственный**

**университет им. Н.И. Лобачевского, 2018**

**Contents**

Introduction………………………………………………………………………...4

1. The main packages, operators and functions of the system of analytical calculations Maple……………………………………………………………………........6

1.1 The packages of functions……………………………………………………7

1.2 The methods of giving task to functions and drawing of their diagrams……12

1.3 The calculation of limits……………………………………………………...28

1.4 The calculation of derivatives………………………………………………...31

1.5 Integrals calculus……………………………………………………...............37

1.6 Operations with rows…………………………………………………….. …43

1.7 Solving equations, inequalities and their systems……………………………53

1.8 The analysis of functions………………………………………………….…62

1.9 Solving differential equations and their systems…………………………….71

1.10 Operations with vectors and matriсes ………………………………………90

1.11 The transformation of complex members, analytical expressions and functions of complex variables……………………………………………………………..101

2. Test assignments for doing laboratory tests on computer…………………...109

The list of used sources…………………………………………………………150

The appendix………………………………………………………………….…151

**Introduction**

One of the stages of the process of mathematical models is development of software. Much attention is always paid to this stage in the chain “object – calculation – mathematical model – software” – analysing the results by an electronic computer. In the process of the development of software in the first stage particularly this software is set up. The second stage is devoted to the most crucial task, using this software, namely, the identification of the parameters of the mathematical model. That is why, great demands are made to software in working of the user with it: convenient interface, high speed in calculations, syntactic and semantic control in the process of introducing the parameters, etc. Nowadays there are lots of opportunities for personal computers such as high speed, memory volume, the saturation of programming means, which enable the user to model sophisticated dynamic systems, avoiding the simplification (sometimes without any grounds) of their mathematical models.

The permanent growth of calculating opportunities of modern computers enables the user to hope for successful solving more and more complicated tasks. Parallel with programming languages there appeared the systems of analytical calculations (SAC): Maple, Mathematica, Matlab, etc. The usage of SAC (the systems of computer algebra with broadened opportunities in the field of symbolic calculations) enables the user to automatize the process of writing programs even on a more lavish scale.

The system of Maple |1| ranks high among SAC. The systems of computer mathematics of Maple class were made, by the Waterloo Maple corporation (Canada) as the systems of computer algebra with broadened opportunities in the field of symbolic calculations. Maple is a typical integrated system.

It comprises:

- high-powered programming language;

- editing device for preparing; and editing documents and programs;

- modern multi-windowed user’s interface with opportunities of working in the dialogue conditions;

- high-powered information system with thousands of examples;

-the main body of algorithms and transformation rules of mathematical expressions;

- memory and symbolic processors;

- the system of diagnosing;

- the libraries of installed and supplementary functions;

- the package of functions of other producers and backing and support of other programming languages and programs.

On examining a large number of publications devoted to describing SAC such terms as "high- powered language", "information system", etc. are often mentioned, which actually corresponds to reality. The author’s experience confirms it, for instance, while solving a number of complicated tasks of dynamics of systems, connected with transformation of cumbersome expressions, we were bound to give up using the programming language Fortran in favour of the system of analytical calculations Maple.

While writing this manual, the authors used the material, given in the list of the used sources; there are described the examples of using the systems, received by the authors, while solving particular problems. The first section tackles a brief description of opportunities of this system and the examples of its usage. The second section touches upon mathematical models of particular systems and the complexes of programs for these calculations. The third section is devoted to particular tasks for laboratory work.

1. The main packages ,operators and functions. The systems of analytical calculations MAPLE

This section tackles the opportunities of the system for solving problems from different branches of mathematics: mathematical analysis,algebra,geometry, the theory of functions & others.A brief description of packages & the examples of using functions for solving various problems are given:drawing diagrams of functions, calculating the limits,derivatives & integrals, operations with lines, solving equations , inequalities& their systems, the analysis of functions , solving differential equations ,operations with vectors & matrices, transformation of complex numbers & expressions.These examples are taken from well-known sources for various branches of mathematics , so that the user might control the correctness of solving problems ,implementing the functions of the MAPLE system

The basis for working with symbolic transformations into MAPLE is the nucleus of the system. It contains hundreds of basic functions & algorithms of symbolic transformations. The volume of nucleus can reach 6 or 7 Mbite.Besides; the main library of operators, commands& functions is available. Many functions, inserted into it, as well the functions of the nucleus can be used without any declaration, and others need declaring. A great number of functions, as well as the functions, rarely used in calculations of special types are implemented in problem – oriented packages. The package is switched on by the command: with < the name of the package >.

We shall start describing the most widely used packages with the package of two- or three – dimensional diagrams < plots >. The package “plots “contains 50 diagram functions, essentially broadening the opportunities of drawing two- or three- dimensional diagrams. Among these functions one should mark the means of drawing a number of new types in terms of level lines, vector fields & others, as well as the means of combining several graphics. While solving the problems of the system dynamics, special attention should be paid to the functions, enabling to draw two- or one – dimensional graphics.

Below we turn to the package of diagrams& a list of functions, included into this package.

A unique opportunity of the MAPLE system is the opportunity of solving the problems of linear algebra in terms of symbols. Numerical methods of solving the problems of linear algebra are also implemented. They are widely used in the main spheres of its implementation- mathematical models, the system dynamics, the theory of control & others. The package of solving the problems of linear algebra LINALG is one of the most widely used & powerful packages in this field. To look through the functions, located in this package it is sufficient to use the command: WITH /LINALG/;

In new implementations of the system the main emphasis was placed upon using fast algorithms of linear algebra. Their usage provides effective implementation of the system of symbolic mathematics in solving the problems, referring to linear algebra. The mentioned above programming means are located in the package LINEAR ALGEBRA. To load this package one should use the command > WITH / LINEAR ALGEBRA/<

.

1.1.

> **restart:**

> **with(plots);**



.

:

> **with(linalg);**



В новых реализациях системы была сделана ставка на использование быстрых алгоритмов линейной алгебры. Их применение обеспечивает эффективное использование систем символьной математики в решении задач, сводящихся к задачам линейной алгебры. Указанные программные средства реализованы в пакете LinearAlgebra. Для загрузки этого пакета используется команда:

> **with(LinearAlgebra);**



The package Curve Fitting implements the procedure of approximating curves. It contains a number of functions:

* **with(Curve Fitting);**



These functions include: the function for calculating B-spline curves,p olynomial & spline approximation,the method of the least squares ,the functions of rational approximation & approximation of continued fractions,approximation by Taylor line & polynomic, by Chebyshev polynomials & Chebyshev –Pade approximation.

The package INTTRANS is intended for expansive support of integral transformations. This package is called by the command:

* **with(inttrans);**



& contains a number of functions, covering such important fields of mathematics as Fourier’ rows, direct & indirect transformations of Fourier, Laplace & Gilbert, integral transformations of Henkel & Melvin.

The package STUDENT is one of the most attractive ones for students & post-graduates.It contains the most widely used & necessary functions,used by this category of users in their practical work.

* **with(student);**



Orthogonal polynomials are widely used in various mathematical calculations while making up interpolation algorithms, extrapolations of various functional dependencies, where the characteristics of orthogonality provide high estimation of errors. The package ORTHOPOLY contains orthogonal polynomials: of Hagenщ, щawe, Hermit, Lager, Legendre, Jacobi & Chebyshev. The package is called by the command:

> **with(orthodoxy);**



The package Vector Calculus provides the assess to different commands & functions of vector analysis, theory of field & supplements of differential calculation. This package is oriented in the first place to solving the problems of mathematical physics, where the methods of the theory of field & supplements of differential calculation are used. This Package is called by the command:

* **with(Vector Calculus);**
* 

It should be noted, that after loading this package modifies most operators, commands & functions, installed into the nucleus of the system. That is why one should be very careful while using the package. To restore the role of functions one may use the command RESTART.

To carry out a number of special operations with polynomial or set up polynomials with particular qualities there is a package Polynomial Tools. This package has a small number of functions:

* **with(Polynomial Tools);**

This package contains the functions of splitting, sorting & transforming polynomials & others.

Solving differential equations of various types is one of the advantages of the system of analytical calculations MAPLE. The package DEtools offers a number of useful functions for solving differential equations & the systems of equations:

* **with(DEtools);**



This package offers the most exquisite means for analytical & numerical solution of differential equations & their systems. For solving differential equations with quotient derivatives its visualization a special instrumental package is implemented <PDEtools>:

* **with(PDEtools);**



A large number of physical constants, implemented in calculations are given with using the following package:

* **with(Scientific Constants);**



> **Get Constant(g);**



> **g:=evalf(Constant(g,units));**



This section tackles only a small part of packages of applied programs, used for solving the problems of the class, mentioned above. The information about other packages can be received in the reference book of the system Maple, as well as in a large number of books, devoted to the description of this system.

1.2. The methods of giving tasks to functions & the drawing of their graphs

In the system of analytical calculations there exist 4 methods of giving tasks to functions.

Method 1. Defining the function with the help of the operator of appropriating (:=), for example> **restart;**

> **f[1]:=sin(x)^3+1;**



To calculate the meaning of the function in some particular point , it is necessary to implement the following 2 operators:

> **x:=Pi/4; f[1];**





> **1/4\*2^(1/2)+Pi;**



After carrying out these commands a variable X has the given meaning Pi/4. Not to give the variable a particular value, it is more convenient to use the command of substitution SUBS in the following way:

> **a:=subs({x=Pi/4},f[1]);**



To get an approximate meaning of the former expression in terms of the number with floating comma, we use the command:

> **evalf(%); # The symbol(%) is used for Calling the former command;**



*Method 2 .* Defining the function with the help of functional operator who correlates the independent variable < argument > & the meaning of the dependent variable < function>:

> **f[2]:=(t)->t^3+exp(t);**



Turning to this function is carried out by the usual for specialists method:

> **f[2](1); f[2](3);**





The approximate meanings are received in the following way :

> **evalf(f[2](1)); evalf(f[2](3));**





*Method 3Giving the task to the function with the help of the command .*

> **f[3]:=unapply(y^6+y,y);**



> **f[3](1); f[3](5);**





*Method 4 The task of piece linear function. For example: The*

> **f[4]:=piecewise(z<0,1,z>=0 and z<1, z, z>=1,cos(z));**



The above mentioned methods of giving tasks to functions can be applied for the functions from several variables.

For drawing the graphs of functions the package PLOTS is used It is given in the following way

plot(f, h,v,g),

where f — visualized function,

h — the argument of the function, with defining the sphere of its change

v — not obligatory variable—

g — a set of parameters, defining the style of drawing the graph of functions.

For two –dimensional graph the following most widely used parameters are possible<g>):

axes — defines the type of coordinate axes, allows the meanings: frame, boxed, normal, none;

axesfont — defines the print for representing the inscriptions of coordinate scale the type of print (times,courier,helvetica,symbol), its style (bold, italic, bolditalic), as well as its size;

color — the color of the line for representing the graph of the functionи: aquamarine, black,

blue, navy, coral, cyan, brown,

gold, green, gray, khaki, magenta,

maroon, orange, pink, plum,

red, sienna, tan, turquoise, violet,

wheat, white, yellow;

cords — the parameter defines the coordinate system, representing the graph: Cartesian — to pass over, bipolar, cardioid, cassinian,elliptic, hyperbolic, invcassinian, invelliptic logarithmic, logcosh, rose & Maxwell, parabolic, tangent;

discont — if the formal parameter of the function equals TRUE, checking of the function for discount is carried out. The meaning of the parameter equals FALSE;

font — the parameter defines the print for representing inscriptions of texts (times, courier, Helvetica или symbol) & the style of the print (roman, bold, italic, bolditalic) on the graph;

labels — the parameter defines the inscriptions for the axes of coordinates; the parameter is given in2 lines for the abcciss axis &for the coordinate axis;

labelfont — the parameter defines the print for the coordinate axes;

legend — the legend of the graph, for several graphs there is given the list each element of which is the legend for a particular graph;

linestyle — line style: solid, dot, dash, dashdot;

numpoints — the parameter presents the minimum quantity of basic points for drawing the graph < this number equals 50>;

scaling — the parameter presents the scale by the coordinate axes (constrained) & (unconstrained);

style — the parameter of the line style (line), (point), (patch), (patchnogrid);

symbol — the type of symbols for representing the basic points(box), (cross), (circle), (point), (diamond);

symbolize — the size of symbols for representing the basic points, it equals 10.

title — the title of the drawing;

title font — the print of the title of the drawing;

thickness — the parameter, defining the thickness of the line for drawing the graph (0–15).

Below there are given the examples of drawing the graphs of functions, given in different forms.

> **restart: with(plots):**

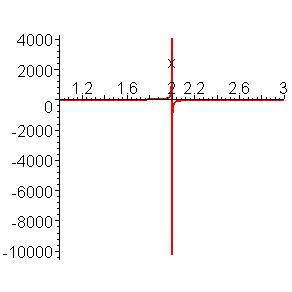
**Warning, the name changecoords has been redefined**

For the explicit function the drawing of the graph is carried out in the following way:

> **f[1]:=(x)->(x^2-x-6)/(x-2);**

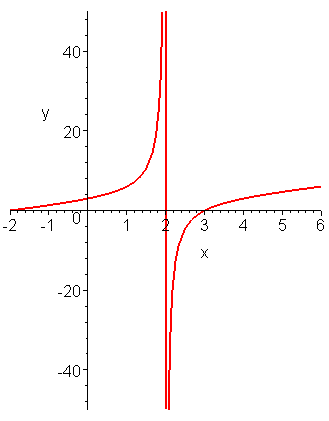


> **plot(f[1](x),x=1..3,color=red,thickness=2);**



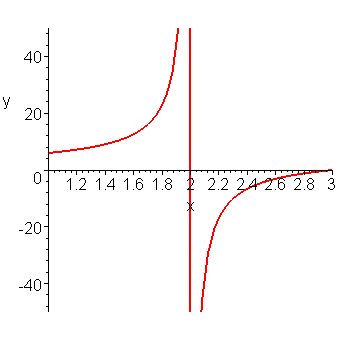
If to give the limits for changing the value of the function, the graphic pattern will become more understandable:

> **plot(f[1](x),x=-2..6,y=-50..50,color=red,thickness=2);**



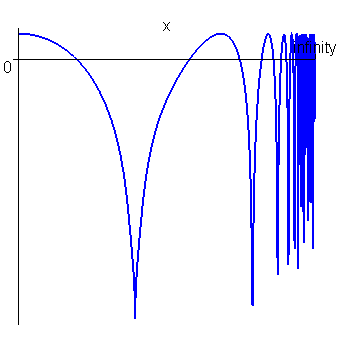
Analytically given function can also be used in the operator plot:

> **plot((x^2-x-6)/(x-2),x=1..3,y=-50..50,thickness=2);**

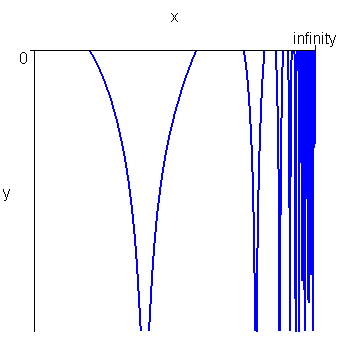


The limit of changing the argument can be infinite (infinity).

> **plot(ln(1+cos(x)),x=0..infinity,color=blue,thickness=2);**

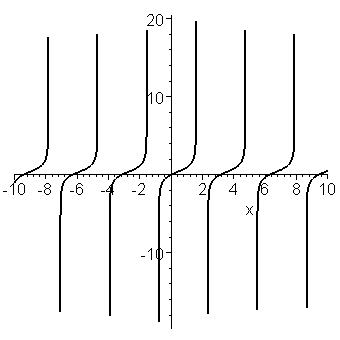


> **plot(ln(1+cos(x)),x=0..infinity,y=-5..0,color=blue,thickness=2);**

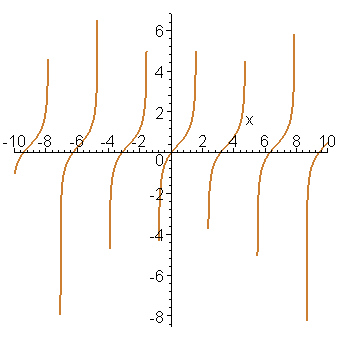


For the function with discont the drawing of continuous graph is possible discont=true (discont=false):

> **plot(ln(1+tan(x)),x=-10..10,discont=true,color=gold,thickness=2);**



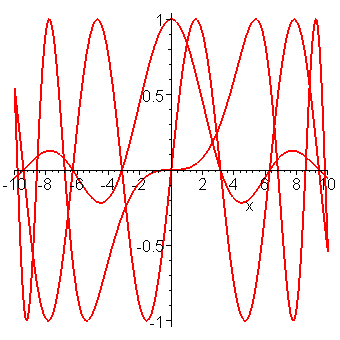
> **plot(ln(1+tan(x)),x=-10..10,color=gold,thickness=2);**



The drawing of several functions in one axes is carried out in the following way:

* **plot([sin(x),sin(x)/x,sin(x^3/100)],**

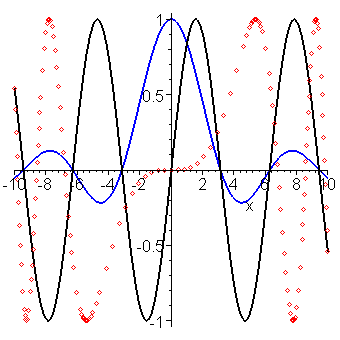
**x=-10..10,color=red,thickness=2);**



> **plot([sin(x),sin(x)/x,sin(x^3/100)],**

**x=-10..10,color=[black,blue,red],**

**style=[line,line,point],thickness=2);**

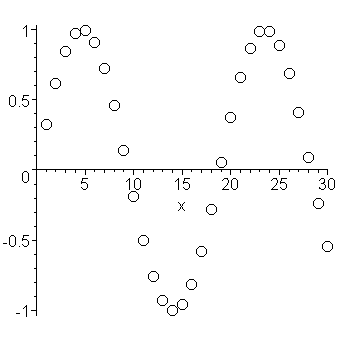


The drawing of the graph of the function, presented by separate points is carried out in the following way:

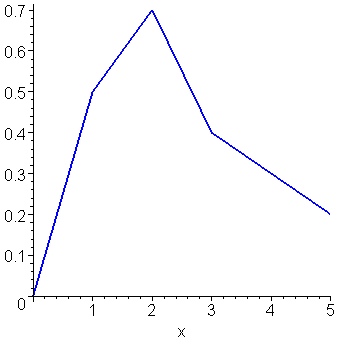
> **restart:**

> **p:=[[i,sin(i/3)]$i=1..30]:# -the procedure giving the sequence of points;**

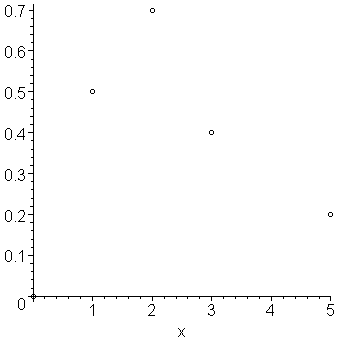
> **plot(p,x=0..30,color=black,style=point,symbol=circle,symbolsize=20,thickness=4);**



* **plot([[0,0],[1,0.5],[2,0.7],[3,0.4],[5,0.2]],x=0..5,style=line,symbol=circle,color=blue,thickness=2);**



* **plot([[0,0],[1,0.5],[2,0.7],[3,0.4],[5,0.2]],x=0..5,style=point,symbol=circle,color=black,thickness=4);**



The drawing of functions, defined by the meaning of vector elements:

> **restart:**

> **x:=[0,1,2,3,4,5,6,7]: y:=[2,1.79,1.27,0.571,-0.2,-0.678,-0.758,-0.538]:**

>

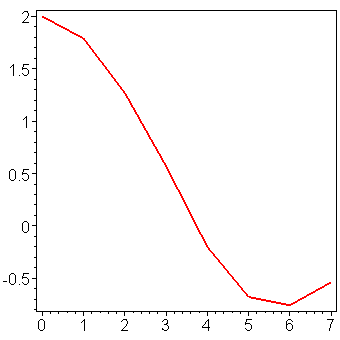
> **pare:=(x,y)->[x,y];**



> **Coordxy:=zip(pare,x,y,2);**



> **plot(Coordxy,style=[line,point],symbol=circle,axes=boxed,thickness=2);**



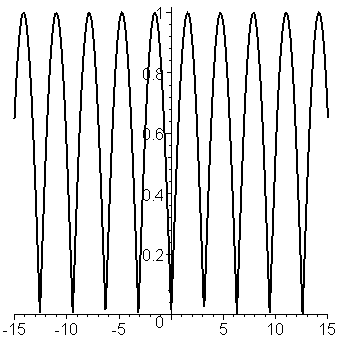
The drawing of graph of the function by the given procedure:

> **restart:**

> **w:=proc(x) if sin(x)>0 then sin(x) else -sin(x) fi end;**



* **plot(w,-15..15,color=black,thickness=2);**



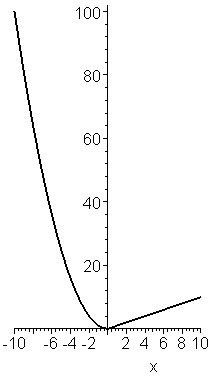
Drawing graphs of piece-linear functions:

> **restart:**

> **f[1]:=piecewise(x<0,x^2,x);**



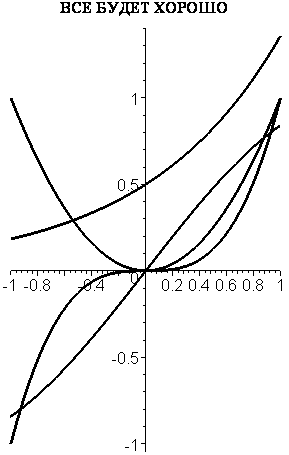
> **plot(f[1],x=-10..10,color=black,thickness=2);**



The drawing of the graphs of the functions presented by functional operators & installed functions:

> **restart:**

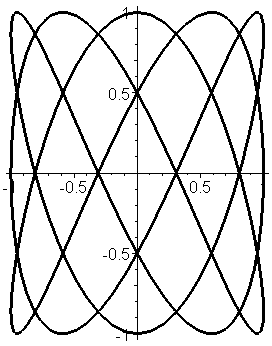
> **plot([exp/2,sin,x->x^2,x->x^3],-1..1,color=black,thickness=3,title="EVERYTHING WILL BE ALL RIGHT ",titlefont=[TIMES,BOLD,10]);**



The drawing of functions, presented parametrically:

> **restart:**

> **plot([sin(3\*t),cos(5\*t),t=0..2\*Pi],color=black,thickness=3);**

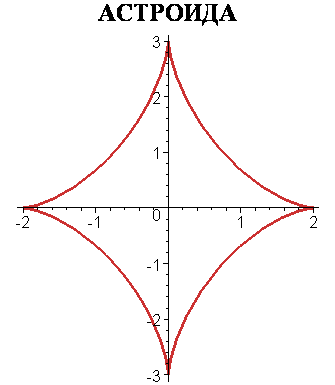


> **x:=2\*(cos(t)^3);y:=3\*(sin(t)^3);**





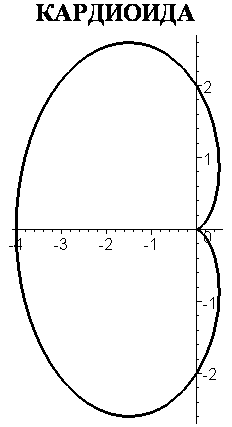
* **plot([x,y,t=0..2\*Pi],color=orange,title="АСТРОИДА",titlefont=[TIMES,BOLD,15],thickness=3);**



The drawing of functions, presented in the polar coordinate system:

> **restart:**

> **plot([2\*(1-cos(phi)),phi,phi=0..2\*Pi],color=black,coords=polar,title="КАРДИОИДА", titlefont=[TIMES,BOLD,15],thickness=3);**

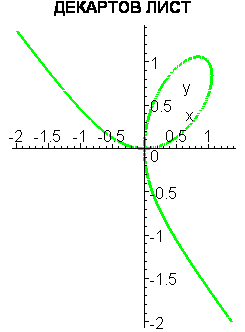


> **polarplot(phi,phi=0..8\*Pi,title="Archimedes spiral ",titlefont=[HELVETICA,BOLD,10],thickness=3);** П

The drawing of the graph of the implicit function :

> **restart:with(plots): implicitplot(x^3+y^3-2\*x\*y=0,**

**x=-2..1.5,y=-2..2,color=green,title="Descartes list",titlefont=[HELVETICA,BOLD,10],numpoints=10000,thickness=3);**



>

The drawing of the graphs of functions from 2 variables is carried out with the help of installed into the nucleus of the function

plot3d.

For drawing the explicit function z = z(x,y) the procedure plot3d(z,x = a..b, y = c..d,p)is used, and for the function with а parametrical form of the task - x = x(t), y = y(t), z = z(t) – plot3d([x,y,z], a..b,c..d,p).

Drawing the explicit functions:

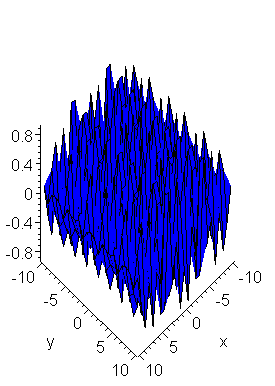
> **restart;**

> **z[1]:=(x,y)->cos(x^2+2)\*sin(y^2+2);**



* **plot3d(z[1](x,y),x=-10..10,**

**y=-10..10,color=blue,thickness=1,axes=frame);**



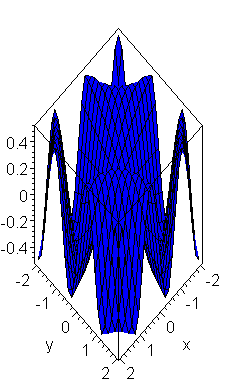
> **restart;**

> **z[2]:=(x,y)->cos(x\*y)\*sin(x\*y);**



> **plot3d(z[2](x,y),x=-2..2,**

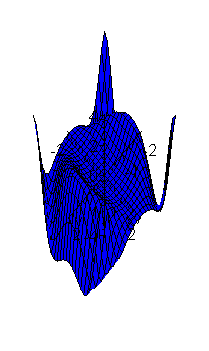
**y=-2..2,color=blue,thickness=1,axes=boxed);**



> **restart;**

> **plot3d(sin(x^2+y^2)\*x^2,x=-2..2,**

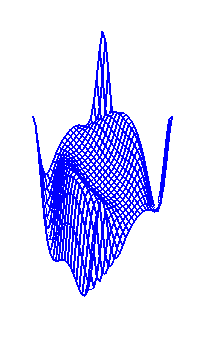
**y=-2..2,color=blue,thickness=1,axes=normal);**



> **restart;**

> **plot3d(sin(x^2+y^2)\*x^2,x=-2..2,y=-2..2,style=hidden,color=blue,thickness=2);**

>



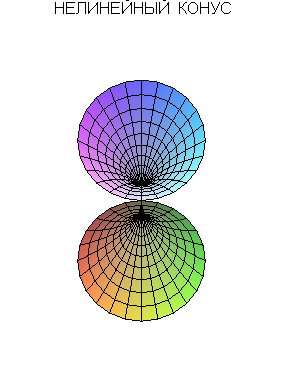
Drawing the graph of the function in the cylindrical coordinate system :

> **restart:**

> **plot3d(h^2,a=-Pi..Pi,**

**h=-5..5,coords=cylindrical,style=patch,**

**title=" NONLINEAR conus",thickness=1);**



The current section tackles the most widely used operators for drawing 2-or 3 –dimensional graphs of functions, used for visualizing the results of the research.

1.3. Calculating the limits

For calculating the limits of the function with x->a the following function is used:

limit(f(x),x = a, g),

where f(x) — algebraic expression,

x — the name of the variable,

g — the parameter, referring to the direction of searching the limit — left, right — , real — in the field of real numbers, complex — complex numbers. a — can be (infinity), both positive ,and negative. The examples of implementing this function are given below:

> **restart;**

> **with(linalg):**

> **Limit(f(x),x=a);**



> **Limit(sin(x)^tan(x),x=0)=limit(sin(x)^tan(x),x=0);**



> **Limit((1+(3/x)^x),x=infinity)=limit((1+(3/x)^x),**

**x=infinity);**



> **Limit(arcsin(sqrt(x^2+x)+x),x=infinity)=**

**=limit(arcsin(sqrt(x^2+x)+x),x=infinity);**



> **Limit((1/x)-(1/exp(x)-1),x=0)=**

**=limit((1/x)-(1/exp(x)-1),x=0);**



> **Limit(((x+2^x)^(1/x)),x=infinity)=**

**=limit(((x+2^x)^(1/x)),x=infinity);**



> **Limit(1-exp(-x),x=infinity)=**

**=limit(1-exp(-x),x=infinity);**



> **Limit(exp(x),x=infinity)=limit(exp(x),x=infinity);**



> **Limit((Pi-2\*x)\*tan(x),x=Pi/2)=**

**=limit((Pi-2\*x)\*tan(x),x=Pi/2);**



> **Limit(tan(x),x=Pi/2)=limit(tan(x),x=Pi/2);**



> **Limit(tan(x),x=Pi/2,right)=limit(tan(x),x=Pi/2,right);**

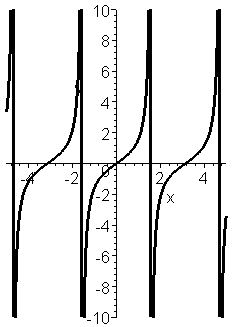


> **Limit(tan(x),x=Pi/2,left)=limit(tan(x),x=Pi/2,left);**



* **plot(tan(x),x=-5..5,**

**y=-10..10,color=black,thickness=3);**



1.4. Calculating the derivatives

Calculating the derivatives is one of the most widely used problems of mathematical analysis.To solve it there exists the following function:

diff(a,x1,x2,..., xn) ,

where a — differential algebraic expression or function,

x1, x2,..., xn — a row of variables,where differentiation is carried out .

Below there are given the examples of using the function differantion.

Calculating the derivatives of explicit functions:

> **restart:**

> **with(linalg): with(plots):**

**Warning, the protected names norm and trace have been redefined and unprotected**

**Warning, the name changecoords has been redefined**

> **diff(sin(x),x);**



> **Diff(sin(x),x)=diff(sin(x),x);**



To calculate the derivative from the function one can use the differential operator D. It enables to produce more compact expressions, than diff.

> **D(sin);**



> **D(sin)(x);**



> **y:=(x)->(x\*sin(alpha)+cos(alpha))\*(x\*cos(alpha)-sin(alpha));**



> **Diff(y(x),x)=diff(y(x),x) ;**



> **D(y);**



>

> **restart: y:=x+x^x+x^(x^x);**



> **diff(y,x);**



> **diff(x^(1/x),x);**



> **collect(%,1/x);**



* **restart; y:=x->floor(x)\*sin(Pi\*x)^2;**
* **# floor(x)-целая часть (Антье) числа;**



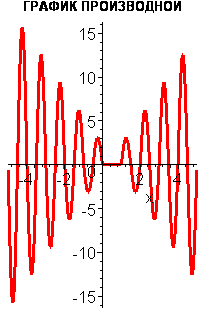
> **diff(y(x),x);**



> **combine(%);**



* **plot(%,x=-5..5,title="ГРАФИК ПРОИЗВОДНОЙ", titlefont=[HELVETICA,BOLD,8],thickness=3);**



Calculating the derivative from piece- linear function is carried out in the following way:

> **restart:**

> **y:=x->piecewise(x<1,1-x,x<=2,(1-x)\*(2-x),**

**-(2-x));"y(x)"=y(x);**





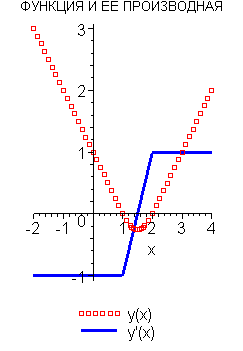
> **f:=x->diff(y(x),x);**



> **"y'(x)"=f(x);**



* **plot([y(x),f(x)],x=-2..4,title="FUNCTION AND ITS DERIVATIVE",titlefont=[HELVETICA,8],color=[RED,BLUE],style=[POINT,LINE],symbol=BOX,legend=["y(x)","y'(x)"],thickness=3);**



FUNCTION AND ITS DERIVATIVE

Calculating derivative functions by the parameters:

> **restart;**

> **x:=t->exp(2\*t)\*cos(t)^2;**



> **y:=t->exp(2\*t)\*sin(t)^2;**



> **diff(y(t),t)/diff(x(t),t);**



The derivative from the implicit function & its graphic expression:

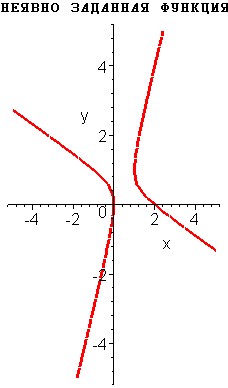
>

> **restart: with(plots):**

> **F:=x^2+2\*x\*y-y^2-2\*x;**



* **implicitplot(F,x=-5..5,y=-5..5,thickness=3,title="IMPLICIT FUNCTION",titlefont=[COURIER,BOLD,8]);**



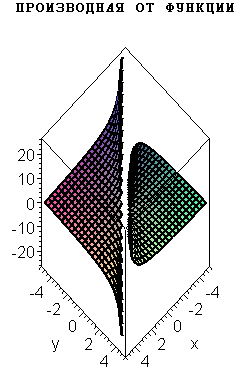
IMPLICIT FUNCTION

> **F[1]:=implicitdiff(F,y,x);**



* **plot3d(F[1],x=-5..5,y=-5..5,thickness=3,title="DERIVATIVE OF FUNCTION",titlefont=[COURIER,BOLD,4],axes=boxed);**

DERIVATIVE OF FUNCTION



>

Derivatives from higher orders :

> **restart:**

> **y:=x->x\*sinh(x);**



> **diff(y(x),x$100);**



> **y1:=2\*cos(5\*t)+3\*sin(9\*t);**



> **diff(y1,t$4);**



Quotient derivatives:

> **restart;**

> **u:=(x,y)->x^4+y^4-4\*x^2\*y^2;**



> **Diff(u(x,y),x)=diff(u(x,y),x);**



> **Diff(u(x,y),y)=diff(u(x,y),y);**



> **Diff(u(x,y),x,y)=diff(u(x,y),x,y);**



> **Diff(u(x,y),x$2)=diff(u(x,y),x$2);**



> **Diff(u(x,y),y,y)=diff(u(x,y),y,y);**

1.5. Integral calculus

Calculating indefinite & definite integrals is one of the most widely spread operations of mathematical analysis. To calculate these integrals the following functions are presented:

int(f,x), int(f,x = a..b), int(f,x = a..b,c),

где f — subintegral function , x — a variable of integration, a, b — low & upper limits of integration, c — non-obligatory additional condition. Below the examples of using these functions for calculating various integrals are given.

Calculating indefinite integrals:

> **restart:**

> **Int(exp(alpha\*x)\*cos(b\*x),x)=int(exp(alpha\*x)\*cos(b\*x),x);**



> **collect(%,exp); # Put beyond the bracket exp(x);**



> **Seq:=f(x),x; # to the variable Seq sequences of 2 elements are given as the meaning -- the result of the operator”s F[ ] influence on the variable X f() & the influence of the variable x;**



> **Int(Seq)=int(Seq);**



Finding the integrals of discount functions:

> **restart: with(plots):**

**Warning, the name changecoords has been redefined**

> **f:=(x)->piecewise(x<0,1,x<=1,x+1,2\*x);**



> **f(x);**



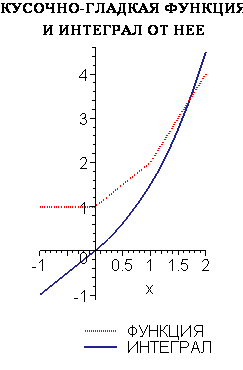
> **F:=x->int(f(x),x);**



> **F(x);**



* **plot([f(x),F(x)],x=-1..2,color=[RED,NAVY],linestyle=[DOT,SOLID],thickness=2,title="КУСОЧНО-ГЛАДКАЯ ФУНКЦИЯ \n И ИНТЕГРАЛ ОТ НЕЕ",titlefont=[TIMES,BOLD,9],legend=["ФУНКЦИЯ","ИНТЕГРАЛ"]);**



Calculating definite integrals:

> **restart:**

> **Int(x^2\*cos(x),x=0..2\*Pi)=int(x^2\*cos(x),x=0..2\*Pi);**



> **Int(1/((x^3+x^2+2\*x+2)^2),x=0..1)=int(1/((x^3+x^2+2\*x+2)^2),x=0..1); evalf(%);**





> **# Integral with a parameter;**

> **s:=m->int(sin(x)^m,x=0..Pi/2);**



> **assume(m::integer>0); # Assume m>0 -integer;**

> **s(m); # Г(x) - gamma-function of Euler;**



Calculating non-personal integrals:

> **restart: with(plots):**

> **y:=(x)->(x\*ln(x))/(1+x^2)^2;**



> **Int(y(x),x=0..infinity)=int(y(x),x=0..infinity);**



> **evalf(%);**



* **plot(y(x),x=0..infinity,title="Graph of function y(x)",titlefont=[HELVETICA,BOLD,10],thickness=3);**



> **Int(1/(x^2+1)^2,x=-infinity..infinity)=int(1/(x^2+1)^2,x=-infinity..infinity);**



Calculating double integrals:

> **restart: with(student):**

> **Doubleint(x\*y^2,y=x^2..x,x=0..1); # ATTENTION !! The tasks for the limits of integration should be given in the following way ;**



> **value(%);**



> **Int(Int(x\*y\*(x+y),y=0..x^3),x=0..2)=int(int(x\*y\*(x+y),y=0..x^3),x=0..2);**



> **Int(Int(1/(x^2+y^2+5)^2,y=0..infinity),x=0..infinity);**



> **value(%);**



Calculating double integrals by transferring to polar coordinates:

> **restart: with(student):**

> **A:=Doubleint(sqrt(x^2+y^2),x,y,Omega);**



> **# При переходе к полярным используем r и phi;**

> **assume(r>0); # assume r>o;**

> **changevar({x=r\*cos(phi),y=r\*sin(phi)},A,[r,phi]);# chage variables;**



> **# assume the field Omega;**

> **Omega:=x^2+y^2<=a^2; x:=r\*cos(phi);y:=r\*sin(phi); Omega;# Omega in polar coordinates;**









> **simplify(%);**



> **Int(Int(r^2,r=0..a),phi=0..2\*Pi); value(%);**





1.6. Operations with rows

Implementing the system of symbolic mathematics is particularly effective for solving one of the problems of mathematical analysis –operations with rows .This section tackles the problems of analyzing numerical & functional rows,as well as dividing functions into Tailor & Fourier rows , asymptotic division.

Numerical sequences with a given number of members:

> **restart:with(plots):**

> **sum(k^2,k=1..4);**



> **Sum(k^2,k=1..4)=evalf(sum(k^2,k=1..4));**



> **sum(k,k=1..n);**



Infinite sequences:

> **restart:sum(-exp(-k),k=1..n);**



> **sum(-exp(-k),k=1..infinity);**



> **Sum(k\*a^k,k)=evalf(sum(k\*a^k,k));**



> **Sum(1/n!,n=1..infinity)=evalf(sum(1/n!,n=1..infinity));**



> **Sum(n^2\*exp(-sqrt(n)),n=1..5000)=evalf(sum(n^2\*exp(-sqrt(n)),n=1..5000));**



> **Sum(n!/n^sqrt(n),n=1..1000)=evalf(sum(n!/n^sqrt(n),n=1..500));**



Calculating the factorial of number:

* **3!;**



> **factorial(50);**



Double sums:

> **Sum(Sum(k^2,k=1..m),m=1..N);**



> **factor(simplify(value(%)));**



> **subs(N=100,%);**



Calculating the products of member sequences:

> **restart:**

> **Product(k^2,k=1..5)=product(k^2,k=1..5);**



> **Product(k^2,k)=product(k^2,k);**



> **product(a[k],k=1..6);**



> **f:=[1,2,3,4,5]; product(f[k],k=1..5);**





> **product(n+k,k=1..4);**



> **Product(2/i,i=1..infinity)=product(2/i,i=1..infinity);**



The total scheme of analyzing the coincidence of numerical rows.

Rows with constant signs:

> **restart:with(linalg):**

> **U:=(n)->(1/(ln(n)^2))\*cos(Pi\*n^2/(n+1));**



> **S:=Sum(U(n),n=2..infinity);**



Checking the necessary condition of coincidence:

> **Limit(U(n),n=infinity)=limit(U(n),n=infinity);**



To define the coincidence the Dalamber sign is used:

> **if limit(U(n+1)/U(n),n=infinity)<1 then print(R-coincide) else print (R- not coincide) fi;**

R-not coincide

Rows of variables of signs:

> **restart:**

> **V:=(n)-> (-1)^n\*sin(n)^2/n;**



> **S:=Sum(V(n),n=2..infinity);**



Checking the necessary condition of coincidence:

> **Limit(V(n),n=infinity)=limit(V(n),n=infinity);**

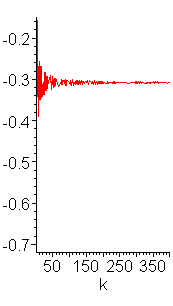


Drawing the graph of partial sums of a row :

> **H:=(k)->sum((-1)^n\*sin(n)^2/n,n=1..k);**



> **plot(H(k),k=1..400);**

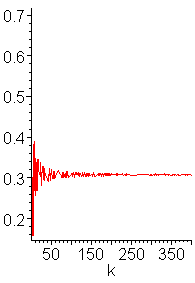


Drawing the graph of partial sums from modules:

> **G:=(k)->abs(H(k));**



> **plot(G(k),k=1..400);**



Defining the sum of the row:

> **Sum((-1)^n\*sin(n)^2/n,n=1..1000)=evalf(sum((-1)^n\*sin(n)^2/n,n=1..1000));**



Degree rows:

> **restart:**

> **a:=(n)->(n^n)/(n!);**



> **Sum(a(n)\*x^n,n=1..infinity);**



> **R:=limit(a(n)/a(n+1),n=infinity); evalf(%); # field of coincidence -e^(-1) < x < e^(-1);**





Diving functions into Tailor rows:

The function of one variable:

> **restart: with(plots):with(linalg):**

> **# Dividing into Tailor row to member с x^4 of function f(x);**

> **f:=(1+x+x^2)/(1-x+x^2);**



> **taylor(f,x=0);**



> **# Dividing x=0 to the order of the remainder = 5(non obligatory parameter);**

> **taylor(f,x=0,5);f1:=convert(%,polynom);f1;**





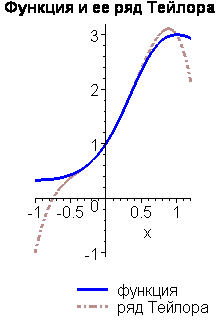


> **f1:=convert(%,polynome);# f1- the main part of the row**



> **# Let’s analyse how the initial function differs from its approximation by Tailor row;**

> **plot([f,f1],x=-1..1.2,color=[BLUE,PINK],linestyle=[SOLID,DASHDOT],title="Функция и ее ряд Тейлора",titlefont=[HELVETICA,BOLD,10],legend=["функция","ряд Тейлора"],thickness=3);**



Function from 2 variables:

> **restart:**

> **mtaylor(f(x,y),[x,y],3):**

> **mtaylor(sin(x^2+y^2),[x,y]);**



> **mtaylor(sin(x^2+y^2),[x,y],8);**



> **mtaylor(sin(x^2+y^2),[x,y],8,[2,1]);**



> **mtaylor(sin(x^2+y^2),[x=1,y],3);**





Asymtotic division of functions> **restart:with(linalg): with(plots):**

**Warning, the protected names norm and trace have been redefined and unprotected**

**Warning, the name changecoords has been redefined**

> **y[1]:=1/(1-x-x^2);**



> **asympt(y[1],x);**

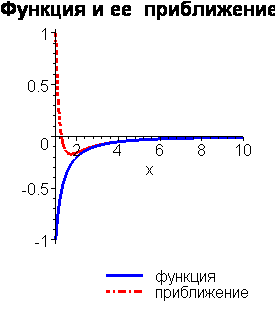


> **y[2]:=evalf(convert(%,polynomial));**



> **# Let’s define how the initial function differs from its asymptotic approximation;**

> **plot([y[1],y[2]],x=1..10,color=[BLUE,RED],linestyle=[SOLID,DASHDOT],title="The function& its approximation",titlefont=[HELVETICA,BOLD,12],legend=["function"appriximation"],thickness=3);**



Dividing the function into Fourier row:

Method 1

>

> **# Expansion into Fourie row of function y(x)=abs(1-x) on the interval (-1,1);**

> **restart;**

> **with(linalg): with(plots):**

**Warning, the protected names norm and trace have been redefined and unprotected**

**Warning, the name changecoords has been redefined**

> **y := x -> abs(1 - x);**



> **# Let’s assume integer conditions on variables n и k;**

> **assume(n, integer); assume(k, integer);**

> **l := 2;**



> **# Calculate Fourier coefficients;**

> **a[0] := 1/2/l\*int(y(x), x = -l..l);**



> **a[n] := n -> 1/l\*int(y(x)\*cos(n\*Pi\*x/l), x = -l..l);**



> **a[n](n);**



> **b[n] := n -> (1/l)\*int(y(x)\*sin(n\*Pi\*x/l), x = -l..l);**



> **b[n](n);**



> **F := (x, k) -> a[0] + sum(a[n](n)\*cos(n\*Pi\*x/l) + b[n](n)\*sin(n\*Pi\*x/l), n = 1..k);**

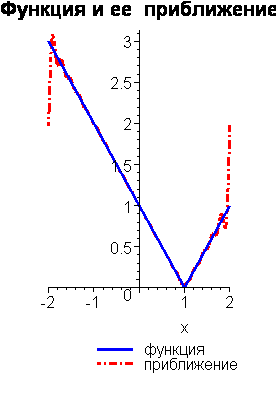


> **F(x, infinity);**



> **# Compare:;**

> **plot([y(x),F(x,20)],x=-2..2,color=[BLUE,RED],linestyle=[SOLID,DASHDOT],title="function & its approximation",titlefont=[HELVETICA,BOLD,12],legend=["function ","approxomation"],thickness=3);**



1.7. Solving equations, inequalities & their systems

Solving linear & non- linear equations & inequalities is one of the main fields of mathematical analysis. To solve the mentioned above type of equations a universal & flexible function is used(eqn, var) or

solve({eqn1,eqn2.....},{var1,var2....}) ,

where eqn is the equation , containing the function of several variables , var — variable ,to which the solution is found.If on writing eqn the sign of equality or ratio signs are not used,it is considered < solve> looks for roots of equation . For getting the numerical solution of non-linear equation or the system of non-linear equations in terms of material numbers it is convenient to use the function fsolve(eqns, vars, options). Sometimes it is reasonable to use other modifications of the function< solve>; rsolve — to solve recurrent equations, isolve — to solve integer equations, msolve — for solving by modules <m> & others.

>

Solving non-linear equations :

> **restart;with(linalg):with(plots):**

> **f:=(2\*x^2+3\*x+1)\*(3\*x+1)=8\*x^4;**



> **fsolve(f,x);**



> **fsolve(f,x,complex);**



> **solve(f,x);**



> **evalf(%);**



> **fsolve(x^5-x,x,complex);**



> **AA:=fsolve(2\*x+(x+2)^2+(x^3-2\*x)=0,x,complex);**



> **# Assuming of the received values of roots for their further analysis is carried out in the following way;**

> **AA[1];AA[2];AA[3];**







Solving transcendental equations:

> **restart;**

> **f1:=sin(x); f2:=cos(x)-1;G1:=f1-f2; s:=fsolve(G1,x);**









> **fsolve(sin(x)=Pi/4,x,complex);**



> **A:=solve(a\*x^2+b\*x+c,x):**

> **A[1];**



> **A[2];**



> **BB:=evalf(solve(a\*x^3+b\*x^2+c\*x+d,x)):**

> **x[1]:=BB[1];**



> **x[2]:=BB[2];**



> **x[3]:=BB[3];**



>

Solving systems of linear equations:

> **restart: with(linalg): with(plots):**

> **sys:={3\*x+5\*y=15,y=x-1};**



> **sols:=solve(sys,{x,y});sols[1];sols[2];**







> **# 8951Assuming the received meanings of the solution is carried out in the following way :**

> **x2:=rhs(sols[2]);# right-hand side (**

> **y2:=rhs(sols[1]);**



> **# or using the operator assign:;**

> **assign(sols);x;y;**





> **# C**



> **implicitplot(sys,x=-10..10,y=-10..10,thickness=3,color=blue);**

> **restart: with(plots):**

> **sys:={3\*x-3\*y=10,2\*x+y=7,y=2\*x+z+4};**



> **sols:=solve(sys,{x,y,z});**



> **# Checking the correctness of the received solution;**

> **subs(sols,sys);**



> **x2:=subs(sols,x);**



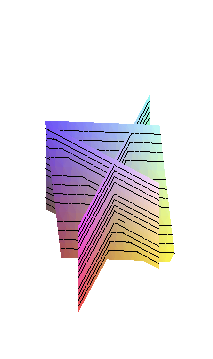
> **y2:=subs(sols,y);**



> **z2:=subs(sols,z);**



> **implicitplot3d(sys,x=-10..10,y=-10..10,z=-15..0,style=patchcontour,orientation=[-25,30]);**



Solving systems of non-linear & transcendental equations:

> **restart;**

> **solve({x\*y=a,x+y=b},{x,y});**



> **allvalues(%);**



> **restart: with(plots):**

Warning, the name changecoords has been redefined

> **z1:=2\*x+4\*y-6;**



> **z2:=y+(1/x)-1;**



> **eqs:={z1,z2};**



> **r:=solve(eqs,[x,y]);**



> **# Checking the correctness of the received solution;**

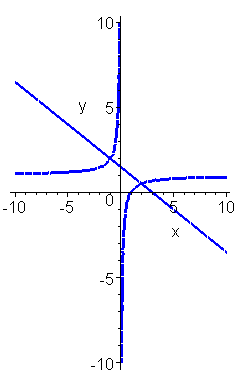
> **eval(eqs,r[1]);**



> **eval(eqs,r[2]);**



> **implicitplot(eqs,x=-10..10,y=-10..10,thickness=3,color=blue);**



> **# Solving the system of non-linear equations{x^2-y^2=1,x^2+xy=2};**

> **eqq:={x^2-y^2=1,x^2+x\*y=2};**



> **\_EnvExplicit:=true;**



> **ss:=solve(eqq,{x,y});**



> **simplify(ss[1]);**



> **simplify(ss[2]);**



> **# Solution of trigonometrical equation {5sin(x)+12cos(x)=13};**

> **\_EnvAllSolutions:=true:**

> **solve(5\*sin(x)+12\*cos(x)=13,x);**



> **evalf(%);**



Solving inequalities:

> **restart:**

> **solve(5\*x>0,x);**



> **solve(5\*x>=10,x);**



> **solve(a\*x>b,{x});**



> **restart:**

> **eqns:=abs(z)^2/(z+1)<exp(2)/(exp(1)-1);**



> **AA:=solve(eqns,{z});**



> **evalf(AA);**



> **restart:**

> **eqns:=exp(x)\*x^2>=1/2;**



> **SS:=solve(eqns,{x});**



> **evalf(SS);**



> **restart:**

> **eqns:=abs((z+abs(z+2))^2-1)^2=9;**



> **solve(eqns,{z});**



Solving the inequality systems:

> **eqns:={x^2<=1,y^2<=1,x+y<1/2};**



> **solve(eqns,{x,y});**



> **solve({x\*y\*z>0,x>-1,y+z>10},{x,y,z});**



> **{z = 0, 10 < y, -1 < x}, {10 < z, y = 0, -1 < x};**

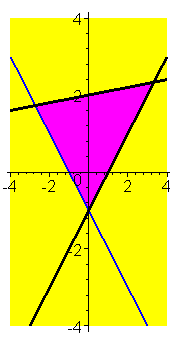


> **# Representing the fields,presented by inequality systems;**

> **restart:**

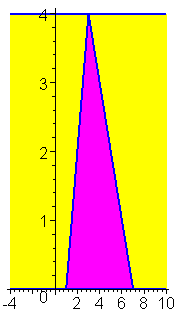
> **with(plots):**

> **inequal({x+y>-1,x-y<=1,y<=2+0.1\*x},x=-4..4,y=-4..4,optionsfeasible=(color=magenta),optionsopen=(color=blue,thickness=2),optionsclosed=(color=black,thickness=3),optionsexcluded=(color=yellow));**



>

> **inequal({0<y,y<4,y/2+1<x,x<7-y},x=-4..10,y=0..4,optionsfeasible=(color=magenta),optionsopen=(color=blue,thickness=2),optionsclosed=(color=black,thickness=3),optionsexcluded=(color=yellow));**



1.8. The analysis of functions

While writing course, diploma & research papers one is bound to analyze functions’behaviour. In some cases it is a complicated & labour-consuming procedure. It is possible to automatize this process by the given below functions. Searching the extremes of functions

> **restart: with(plots):with(linalg):**

> **# Let’s use the function extrema(expr,constrs,vars,'s'), where expr - type of function, constr - constraints,vars- arguments of the function, s – found absciss of extremum point;**

> **y:=(x)->a\*x^2+b\*x+c;**



> **extrema(y(x),{},x,'s');# in case of no limitations empty list is written {};**



> **s;subs(x=s,y(x));**





>

> **extrema(x\*exp(-x),{},x,'s');evalf(%);**

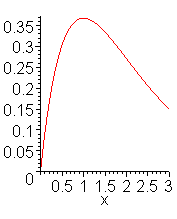




> **s;**



> **plot(x\*exp(-x),x=0..3);**

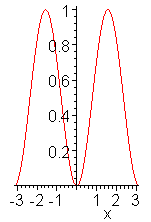


> **evalf(extrema(sin(x)^2,{},x,'s'));s;**





> **plot(sin(x)^2,x=-Pi..Pi);**



Searching the minimum & maximum of 2 variables:

> **restart: with(plots):**

> **z[1]:=(x,y)->x^2-3\*x+y^2+3\*y+3;**



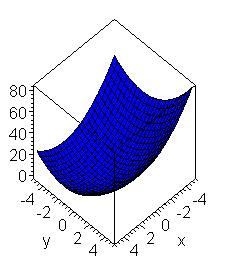
> **minimize(z[1](x,y));**



> **minimize(z[1](x,y),location);**



> **plot3d(z[1](x,y),x=-5..5,y=-5..5,color=blue,thickness=1,axes=boxed);**



> **z[2]:=(x,y)->sin(y)\*exp(-x);**



> **maximize(z[2](x,y));**



> **maximize(z[2](x,y),location);**



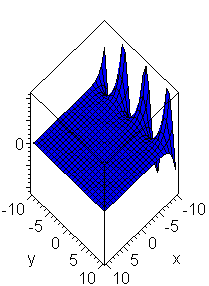
> **maximize(z[2](x,y),x=-10..10,y=-10..10,location);**



> **evalf(%);**



> **plot3d(z[2](x,y),x=-10..10,y=-10..10,color=blue,thickness=1,axes=boxed);**



Analysis of functions for continuity & singularity:

> **restart:**

> **iscont(1/x^2,x=-1..1);# is** function **cont**inuity?**;**





> **iscont(1/x^2,x=0..1);**



> **iscont(1/x^2,x=0..1,'closed');**



> **iscont(1/x^2,x=-1..1,'closed');**



> **iscont(1/(x+2),x=-1..1);**



> **singular(ln(x)/(x^2-x));**



> **singular(tan(x));evalf(%);**





> **singular(1/sin(x));evalf(%);**





> **singular(x+y+1/x,{x,y});**



Operations with polynomials:

> **# Defining the coefficient of polynomial p(x);**

> **restart:**

> **p:=a[4]\*x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0];**



> **coeff(p,x);# Defining the coefficient with x^1;**



> **coeff(p,x^3); coeff(p,x,3); # Defining the coefficient with x^3;**





> **coeffs(p,x);# Defining all coefficients in terms of raising the degree;**



> **collect(p,x);**



> **# Defining coefficient of polynomial q(x,t);**

> **q:=x^2+2\*(t^2)+3\*x+4\*t+5;**



> **coeffs(q);**



> **coeffs(q,t);**



> **collect(q,x);**



> **collect(q,x,t);**



> **# Defining coefficients of polynomials by degrees;**

> **restart:**

> **q:=1/x^2+2/x+3+4\*x+5\*x^2;**



> **lcoeff(q,x);# coefficient with senior degree x;**



> **lcoeff(q,x,'t'); t;**





> **coeffs(q,x,'t');t;**





>

> **p:=a[4]\*x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0];**



> **degree(p,x);# higher degree of polynomial**



> **ldegree(p,x);# lower degree of polynomial;**



> **# expanding polynomial into degrees;**

> **restart:**

> **evala(AFactor(2\*x^2+4\*x-6));**



> **evala(AFactor(x^2+2\*y^2));**



> **expand((x-1)\*(x-2)\*(x-3)\*(x-4));**



> **AFactor(%);**



> **evala(%);**



> **expand((x-1+I+2)\*(x+1-I\*2)\*(x-3));**



> **evala(AFactor(x^2-2\*y^2));**



> **# defining the roots of polynomial;**

> **restart:**

> **p:=x^4+9\*x^3+31\*x^2+59\*x+60;**



> **solve(p,x);# all roots of polynomial;**



> **roots(p,x);# material roots of polynomial in terms of divisibility;**



> **expand((x-1)\*(x-2)\*(x-3)\*(x-4));**



> **roots(%,x);**



> **# main operations with polynomials;**

> **restart:**

> **readlib(psqrt);# including libraries**

> **readlib(proot);**



> **psqrt(x^2+2\*x\*y+y^2); # square root from the expression in the function psqrt;**



> **proot(x^3+3\*x^2+3\*x+1,3); # cube root from the expression in the function proot;**



> **psqrt(x+y);**



> **proot(x+y,3);**



> **p1:=a1\*x^3+b1\*x^2+c1\*x+d1;**



> **p2:=a2\*x^3+b2\*x^2+c2\*x+d2;**



> **p1+p2;# sum of polynomials**



> **p1\*p2;# producing polynomials;**



> **collect(%,x);# ordering polynomials in terms of decreasing degrees x;**



> **p3:=p1/p2; # dividing polynomials;**



> **expand(%,x);**



> **normal(%);# normalizing the fraction;**



> **diff(p1,x);# derivative from polynomial;**



> **diff(p1,x$2);**



> **Int(p1,x)=int(p1,x);# indefinite integral from polynomial;**



> **Int(p1,x=0..1)=int(p1,x=0..1); # definite integral from polynomial;**



> **p3;**



> **denom(p3);# denominator of the fraction;**



> **numer(p3); # numerator of the fraction;**



> **collect(denom(p3),x);**



> **collect(numer(p3),x);**

**For multipliers of polynomial;**

> **restart:**

> **p:=x^4+4;**



> **factor(p);** 

> **# division**



> **factor(p,complex);**



> **factor(x^5-6\*x^4+9\*x^3-6\*x^2+8\*x,complex);**



> **ifactor(453);**



**# the least common divisor of 2 polynomials**

> **s[1]:=x^2-x\*sqrt(3)-sqrt(2)\*x+sqrt(6);**



> **s[2]:=x^2-2;**



> **gcd(s[1],s[2]);# of 2 polynomialsв;**



1.9. Solving differential equations & their systems

Differential equations is the basis of mathematical modelling of different by their nature dynamic systems & processes. The current section tackles the functions, enabling to solve linear & non- linear differential equations & their systems analytically & numerically.Ordinary differential equations ,systems of equations & equations in quotient derivatives.To solve the system of differential equations < Cauchy problem>the function <dsolve> in different forms is used.

dsolve(ode); dsolve(ode, y(x), extra\_args); dsolve({ode,ics}, y(x), extra\_args); dsolve({sysode,ics}, {funcs}, extra\_args),

where < ode> — ordinary differential equation or the system of equations, y(x) — unknown function of one variable ics — expression for initial conditions, {sysode} — quantity of differential equations <system>, {funcs} — quantity of determinant functions , extra\_args — option,defining the type of solution.Parameter< extra\_args> can have the following meanings: exact — аnalytical solution < accepted a priori>explicit , system — solving the system of differential equations, formal series — solution in terms of power polynomial, series — solution in terms of series, numeric — numerical solution. A priori function < dsolve> chooses the most suitable method for solving differential equations automatically.However in function parameters <dsolve>the preferable method of solving differential equations can be given in brackets.The following methods are allowed :quadrature, linear, Bernoulli, inverse\_linear, separable, homogeneous, Chini, lin\_sym, exact, Abel, pot\_sym.

Solving ordinary differential equations of the first order:

> **restart:with(linalg):with(DEtools):with(plots):**

>

> **dsolve(diff(y(x),x)-a\*x=0,y(x));**



> **dsolve(diff(y(x),x)-y(x)=exp(-x),y(x));**



> **dsolve(diff(y(x),x)-y(x)=sin(x)\*x,y(x));**



> **ode\_L:=sin(x)\*diff(y(x),x)-cos(x)\*y(x)=0;**



> **dsolve(ode\_L,[linear],useInt);**



> **value(%);**



> **dsolve(ode\_L,[separable],useInt);**



> **value(%);**



> **mu:=intfactor(ode\_L);**



> **dsolve(mu\*ode\_L,[exact],useInt);**

**G**

> **# Graphic solution of differential equation,usage OF THE FUNCTION odeplot PACKAGE plots;**

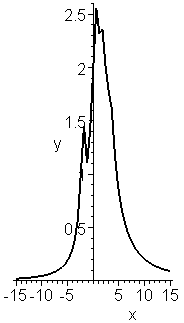
>

> **restart:with(plots):**

> **P:=dsolve({diff(y(x),x)=cos(x^2\*y(x)),y(0)=2},y(x),type=numeric);**



> **odeplot(P,[x,y(x)],-15..15,labels=[x,y],color=black,thickness=2);**



>

Solving differential equations of the second order:

> **restart:**

> **od1:=diff(y(x),x$2)-diff(y(x),x)=sin(x);**



> **dsolve({od1},y(x));**



> **dsolve({diff(y(x),x$2)-diff(y(x),x)=sin(x)},y(x));**



> **dsolve({diff(y(x),x$2)-diff(y(x),x)=sin(x),y(0)=0,D(y)(0)=11},y(x));# equation with initial conditions;**



> **y(x):=rhs(%);**



> **y(x);**



> **restart:**

> **od2:=m\*diff(y(x),x$2)-k\*diff(y(x),x);**



> **yxo:=y(0)=0,D(y)(0)=1; # Assuming initial conditions;**



> **dsolve({od2,yxo},y(x));**



Solving the systems of differential equations <explicitly ,in series ,using Laplace transformations>:

> **restart:**

> **sys:=diff(y(x),x)=2\*z(x)-y(x)-x,diff(z(x),x)=y(x);**



> **fncs:={y(x),z(x)};**



> **A:=dsolve({sys,y(0)=0,z(0)=1},fncs); # solving the system of equations explicitly (exact);**



> **A[1];**



> **z(x):=rhs(%);**



> **A[2];**



> **y(x):=rhs(%);**



> **restart:**

> **sys:=diff(y(x),x)=2\*z(x)-y(x)-x,diff(z(x),x)=y(x);**



> **fncs:={y(x),z(x)};**



> **B:=dsolve({sys,y(0)=0,z(0)=1},fncs,series); # solving the system of equations in terms of row;**



> **y(x):=rhs(B[1]);**



> **z(x):=rhs(B[2]);**



> **restart:**

> **sys:=diff(y(x),x)=2\*z(x)-y(x)-x,diff(z(x),x)=y(x);**



> **fncs:={y(x),z(x)};**



> **Order:=8;# assume the order of approximation by row (a priori 6);**



> **B:=dsolve({sys,y(0)=0,z(0)=1},fncs,series);**



> **y(x):=rhs(B[1]);**



> **z(x):=rhs(B[2]);**



> **restart:**

> **sys:=diff(y(x),x)=2\*z(x)-y(x)-x,diff(z(x),x)=y(x);**



> **fncs:={y(x),z(x)};**



>

> **dsolve({sys,y(0)=0,z(0)=1},fncs,method=laplace);# solving the system of equations ,using Laplace approximation;**



Numerical solution of the system of differential equations:

> restart:

> sys:=diff(y(x),x)=2\*z(x)-y(x)-x,diff(z(x),x)=y(x);



> fncs:={y(x),z(x)};



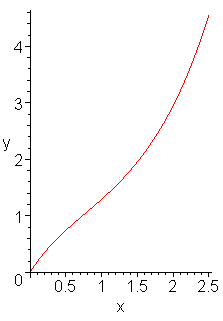
> F:=dsolve({sys,y(0)=0,z(0)=1},fncs,numeric); # solving the equation by the method of Runge,Kutt,Felberg



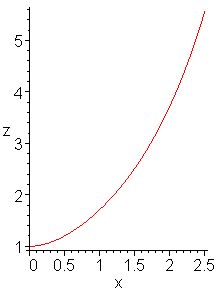
> **F(2);**



* plots[odeplot](F,[x,y(x)],0..2.5,labels=[x,y],color=red); # drawing the graph y(x) of solution;



> plots[odeplot](F,[x,z(x)],0..2.5,labels=[x,z],color=red); # drawing the graph of the solution z(x) ;



> restart:with(plots):

> sys:=diff(y(x),x)=z(x),diff(z(x),x)=3\*sin(y(x));



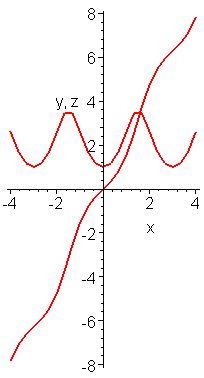
> fncs:={y(x),z(x)};



> P:=dsolve({sys,y(0)=0,z(0)=1},fncs,type=numeric);



**> odeplot(P,[[x,y(x)],[x,z(x)]],-4..4,numpoints=25,color=red,thickness=2);# drawing graphs y(x) и z(x) in one axis;**

****

Solving the system of 2 differential equations with the phase portrait of the solution:

> restart:with(plots):

> sys:=diff(y(x),x)=z(x),diff(z(x),x)=3\*sin(y(x));

> fncs:={y(x),z(x)};

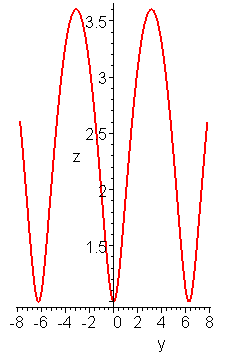


****

> **P:=dsolve({sys,y(0)=0,z(0)=1},fncs,type=numeric);**



* **odeplot(P,[y(x),z(x)],-4..4,numpoints=150,color=red,thickness=2);**



Solving differential equations with piece – linear functions:

> restart:

> eq:=diff(y(x),x)+piecewise(x<x^2-3,exp(x/2))\*y(x);



**> dsolve(eq,y(x));**



Harmonious oscillator:

> restart:

> F:=(t)->A\*sin(nu\*t);



> ode:=diff(q(t),t$2)+2\*delta\*diff(q(t),t)+omega[0]^2\*q(t)=F(t)/m;



> m:=3; delta:=0.1; omega[0]:=150; A:=20; nu:=2;# Assuming the parameters of the system;



****

****

****

****

**> yx0:=q(0)=1,D(q)(0)=0; # Assigning initial conditions;**

****

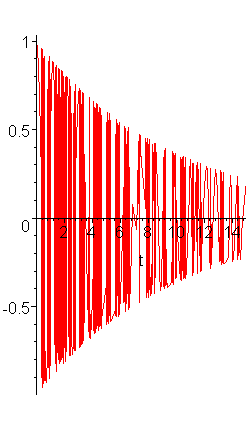
**> dsolve({ode,yx0},q(t)); # Solution of differential equation ( exact);**

****

> **q(t):=rhs(%);**



> **plot(q(t),t=0..15); # Drawing graph of the solution;**



Solving Rikkati equation

> **restart:**

> **eq:=diff(y(x),x)=piecewise(x>0,x)\*y(x)^2;**



> **dsolve({y(0)=1,eq},y(x));**



> **# Checking of the solution;**

> **simplify(eval(subs(%,eq)));**



Solving differential equations in quotient derivatives:

> **restart:**

> **Diff(u(x,t),t,t)=a^2\*Diff(u(x,t),x,x);**



> **# initial conditions u(x,0)=f(x),Diff(u(x,t),t)=0 with t=0, - infinity < x < infinity,t>=0;**

> **eqn:=diff(u(x,t),t$2)-a^2\*diff(u(x,t),x$2)=0;**



> **pdsolve(eqn);**



> **# \_F1(at+x) и \_F2(at-x) – arbitrary functions, twice differentiated;**

> **u:=unapply(rhs(%),x,t); # assign u as a function of 2 parameters x & t;**



> **# avail that derivative of time from u(x,t) with t=0 равна нулю;**

> **D[2](u)(x,0)=0;**



> **dsolve(%,\_F1(x));**



> **\_F1:=F;**



> **\_F2:=x->F(-x);**



> **u(x,t);**



> **u(x,0);**



> **f:=x->1/5\*(1-abs(x))\*Heaviside(1-abs(x));**



> **F:=(1/2)\*f;**



> **a:=1;**



> **evalf(u(x,t));**



Van der Pole oscillator

Classical model of non – linear system, showing periodical autooscillations.

Under different initial conditions phase path is driving to the attractor — the limit cycle.

Established movements are periodical oscillations ,which mathematical cycle in the phase space is the limit cycle.

> restart;with(DEtools):with(plots):

> **vdp:=diff(x(t),t,t)-2\*delta\*diff(x(t),t)\*(1-alpha\*x(t)^2)+omega^2\*x(t)=0;**



> **alpha:=1;omega:=1;d:=0.2;**







> **sys:=[diff(x(t),t)=y(t),diff(y(t),t)-2\*delta\*y(t)\*(1-alpha\*x(t)^2)+omega^2\*x(t)=0];**



> **ff:=dsolve({sys[1],subs(delta=d,sys[2]),x(0)=1,y(0)=1},**

**{x(t),y(t)}, type=numeric, output=listprocedure);**

> **fp := subs(ff,x(t)): fw := subs(ff,y(t)):**



> **steps:=100; init\_t:=0; fin\_t:=15\*Pi;**







> **g:=seq([fp((fin\_t-init\_t)/steps\*i),fw((fin\_t-init\_t)/steps\*i)],i=0..steps):**

> **h:=seq([(fin\_t-init\_t)/steps\*i,fp((fin\_t-init\_t)/steps\*i)],i=0..steps):**

Phase portrait;

* **pointplot([g],connect=true,color=red,title="phase portrait",labels=[coordinate,velocity]):**

Solution

> **pointplot([h],connect=true,color=red,title="oscillations",labels=[t,coordinate]):**

> **ic:=[[x(0)=3,y(0)=2],[x(0)=2,y(0)=0],[x(0)=0.5,y(0)=0.5],[x(0)=0,y(0)=0]];**

> **DEplot(subs(delta=d,sys),[x(t),y(t)],t=0..15\*Pi,ic,method=rkf45,linecolor=black,color=blue,stepsize=0.1,title="Van-der-Pole Oscillator"):**



> **# On changing the parameter <delta> there is a change of the form of attractor, while the topology doesn’t change;**

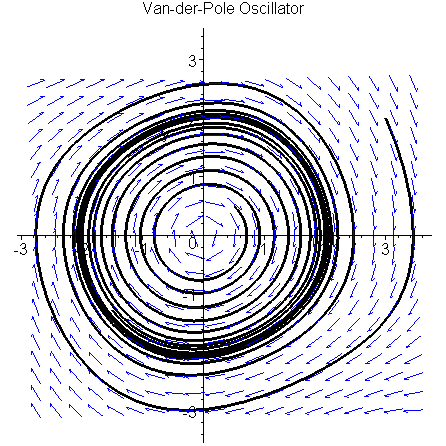
> **for i from 1 by 1 to 15 do**

> **#delta:=i/20;**

> **fp[i]:=DEplot(subs(delta=i/20,sys),[x(t),y(t)],t=0..15\*Pi,ic,method=rkf45,linecolor=black,color=blue,stepsize=0.1,title="Van-der-Pole Oscillator"):**

> **end do:**

> **display(seq(fp[i],i=1..15),insequence=true);**



>

> **restart;with(DEtools):**

> **mu:=0.01:alpha:=0.1:beta:=0.1:sys:=diff(x(t),t)=y(t),diff(y(t),t)=-x(t)+mu\*(-1+alpha\*(1+beta\*x(t)-x(t)^2))\*y(t);**



> **DEplot3d({sys},[x(t),y(t)],t=0..80, [[x(0)=10,y(0)=10]],stepsize=0.1,orientation=[0,90],method=classical[abmoulton],corrections=3,axes=NORMAL,linecolor=blue);**

>

> **restart;with(DEtools):**

> **mu:=0.01:sys:=diff(x(t),t)=y(t),diff(y(t),t)=-x(t)+mu\*(1-x(t)^2)\*y(t);**



> **f1:= (1-R^2\*cos(u)\*cos(u))\*(-R\*sin(u))\*sin(u);**



> **FR:=-1./(2.\*Pi)\*Int(f1,u=0..2\*Pi)=-1./(2.\*Pi)\*int(f1,u=0..2\*Pi);**



> **f2:= (1-R^2\*cos(u)\*cos(u))\*(-R\*sin(u))\*cos(u);**



> **\_FR:=1./(2.\*Pi)\*Int(f2,u=0..2\*Pi)=1./(2.\*Pi)\*int(f2,u=0..2\*Pi);**



>

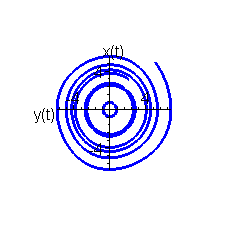
>

>

> **with(DEtools):**

>

> **DEplot3d({sys},[x(t),y(t)],t=0..25, [[x(0)=2,y(0)=2],[x(0)=0.5,y(0)=0.5],[x(0)=5,y(0)=5]],stepsize=0.1,orientation=[0,90],method=classical[abmoulton],corrections=3,axes=NORMAL,linecolor=blue);**



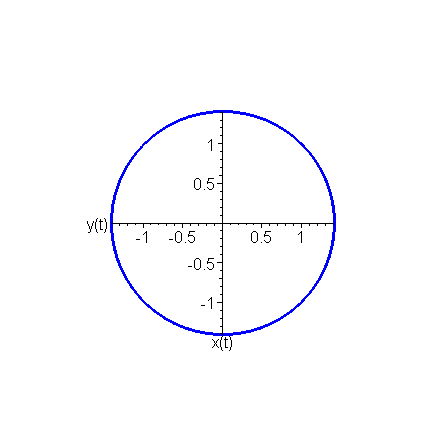
>

> **restart;with(DEtools):**

> **mu:=0.01:alpha:=0.1:beta:=0.1:sys:=diff(x(t),t)=y(t),diff(y(t),t)=-mu\*(alpha\*y(t)^4-beta\*y(t)^2-1)-x(t);**



> **DEplot3d({sys},[x(t),y(t)],t=0..10, [[x(0)=1,y(0)=1]],stepsize=0.1,orientation=[0,90],method=classical[abmoulton],corrections=3,axes=NORMAL,linecolor=blue);**



> **restart;with(DEtools):**

> **fun:=mu\*(1+b\*y(t)^2-a\*y(t)^4);**



> **sys:=diff(x(t),t)=y(t),diff(y(t),t)=-x+fun;**



> **\_y:=-R\*sin(u);**

>

Error, attempting to assign to `\_y` which is protected

> **f1:=(1+b\*\_y^2-a\*\_y^4);**



> **FR:=-1./(2.\*Pi)\*int(f1\*sin(u),u=0..2\*Pi);**



> **\_FR:=-1./(2.\*Pi)\*int(f1\*cos(u),u=0..2\*Pi);**



>

> **restart;with(DEtools):**

The system of Van der Pole equations in approximating lamp characteristic by the third degree polynomial

> **sys:={diff(x(t),t)=y(t),diff(y(t),t)=-x(t)+mu\*(alpha+2\*beta\*x(t)-3\*delta\*x(t)^2)\*y(t)};**



Now we get shortened Van der Pole equations

> **f1:= (alpha+2\*beta\*R\*cos(u)-3\*gamma\*R^2\*cos(u)\*cos(u))\*(-R\*sin(u))\*sin(u);**



> **FR:=-1./(2.\*Pi)\*Int(f1,u=0..2\*Pi)=-1./(2.\*Pi)\*int(f1,u=0..2\*Pi);**



> **f2:= (alpha+2\*beta\*R\*cos(u)-3\*gamma\*R^2\*cos(u)\*cos(u))\*(-R\*sin(u))\*cos(u);**



> **PR:=1./(2.\*Pi)\*Int(f2,u=0..2\*Pi)=1./(2.\*Pi)\*int(f2,u=0..2\*Pi);**



Find radii of limited cycles

> **h:=R\*alpha/2-(3\*delta\*R^3)/8;**



> **solve(h,R);**



> **mu:=0.01:alpha:=0;beta:=0.5;delta:=0.3;sys;**

>

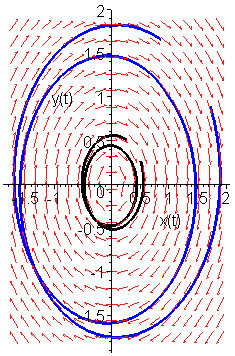








> **phaseportrait(sys,[x(t),y(t)],t=-6..6, [[x(0)=0.3,y(0)=0.3],[x(0)=1,y(0)=1]],stepsize=0.1,scene=[x(t),y(t)],method=classical[foreuler],linecolor=[black,blue]);**



1.10. Оperations with vectors & matrices

The unique opportunity of Maple system is the opportunity of solving problems of linear algebra in terms of symbols. However, this solution is interesting rather from the point of view of theory ,than practice, because even with a small size of matrices symbol results turn out to be very cumbersome & boundless. They are useful only for solving special analytical problems. That is why, MAPLE elaborators implement numerical methods of solving the problems of linear algebra, besides symbolic ones, which are widely used in modelling various technical systems. One of the main packages of solving the problems of linear algebra is the package linalg. It is based on functions, implementing operations with vectors & matrices.

Methods of setting tasks to vectors:

> **restart;with(linalg):with(LinearAlgebra):**

> **v[1]:=array(1..3,[-3,0,4]);**



> **v[2]:=array(1..3,[1,2,7]);**



> **v[3]:=array(1..3,[3,4,-8]);**



> **u[1]:=<3|-2|0>;**



> **u[2]:=<13|-22|0.1>;**



> **u[3]:=<23|-29|54>;**



> **s[1]:=vector[row]([-3,0,4]);**



> **s[2]:=vector[row]([1,2,7]);**



> **s[3]:=vector[row]([3,4,-8]);**



> **alpha[1]:=vector(3,[-3,0,4]);**



> **alpha[2]:=vector(3,[1,2,7]);**



> **alpha[3]:=vector(3,[3,4,-8]);**



Operations with vectors:

> **# Scale product of vectors;**

> **dotprod(v[1],v[2]);**



> **dotprod(s[1],s[2]);**



> **dotprod(alpha[1],alpha[2]);**



> **# Vector product of vectors;**

> **crossprod(v[1],v[2]);**



> **crossprod(s[1],s[2]);**



> **crossprod(alpha[1],alpha[2]);**



> **# Mixed product of vectors;**

> **dotprod(crossprod(alpha[1],alpha[2]),alpha[3]);**



> **# angle between vectors;**

> **angle(alpha[1],alpha[2]);**



> **evalf(%);**



> **norm(alpha[1],3); evalf(%);# Norm of three- dimensional vector;**





> **normalize(alpha[1]);#**



> **Basis([u[1],u[2],u[3]]);# Defining the basis of vectors, only as u[i] (package Linear Algebra);**



> **GramSchmidt([u[1],u[2],u[3]]); # Orthogonalization of vector system only as u[i] (package Linear Algebra);**



> **convert(alpha[1],list[1]);# Преобразование вектора в список;**



Methods of setting tasks to matrices:

> **restart: with(linalg):**

Warning, the protected names norm and trace have been redefined and unprotected

> **C:=matrix(3,3);**



> **C1:=array(1..3,1..3);**



> **A:=matrix(3,3,[1,2,-5,2,8,0,-4,-3,-7]);**



> **B:=matrix(3,3,[1,2,-5,7,6,-8,3,4,-9]);**



> **C2:=matrix(2,3,[1,2,3,4]);**



> **C3:=array(1..3,1..3,[[1,2,-5],[2,8,0],[-4,-3,-7]]);**



> **C4:=array(1..3,1..3,[[1,2,-5],[7,6,-8],[3,4,-9]]);**



Operations with matrices:

> **rowdim(A); # The number of lines of matrix;**



> **coldim(A); # The number of columns of matrix;**



> **evalm(C3+C4);# Addition <subtraction >;**



> **matadd(C3,C4);# Addition of matrices;**



> **multiply(C3,C4);#Multiplication of matrices;**



> **evalm(C3&\*C4);#Multiplication of matrices;**



> **evalm(2\*C3); # Multiplication of matrix by number;**



> **evalm(2+C4);# Addition of matrix with a number , multiplied by a single matrix;**



> **evalm(C3^3);# Raising matrix to power;**



> **evalm(A^(-1));# Finding inverse matrix;**



> **inverse(A); # Finding inverse matrix;**



> **C:=A^3+4\*A+5;# Matrix – the function from matrix;**



> **evalm(C);**



> **evalm(A^0);**



> **transpose(A);# Finding transparent matrix;**



> **det(A);#Finding the determinant of matrix;**



> **rank(A);# Finding the rank of matrix;**



> **trace(A);# Finding the trace of matrix;**



> **norm(B);# Norm of matrix;**



> **kernel(B); # Kernel of matrix;**



Spectral analysis of matrix:

> **evalf(eigenvalues(B,C)); # Proper numbers of matrix;**



> **evalm(C);**



> **evalf(eigenvectors(B)); # Proper numbers +their multiples + proper vectors;**



> **evalf(eigenvals(B,C));**



> **evalm(C);**



> **charpoly(A,lambda); # Characteristic polynomial of matrix ;**



> **charmat(A,lambda); # Characteristic matrix ;**



> **minpoly(A,lambda); # Minimum polynomial (divisor);**



> **jordan(A);# Jordan form of matrix;**



> **gausselim(A); # Giving matrix a triangle type by Gauss method-;**



> **gaussjord(A); # -by Gauss-Jordan method ;**



> **definite(B,'positive\_def'); # Defining positive determination of matrix (yes - true,no - false);**



Symbol operations with matrices:

> **restart: with(linalg):**

> **M1:=array(1..2,1..2,[[a1,b1],[c1,d1]]);**



> **M2:=array(1..2,1..2,[[a2,b2],[c2,d2]]);**



> **evalm(M1+M2);**



> **evalm(M1/M2);**



> **evalm(M1^2);**



> **evalm(sin(M1));**



> **evalm(M1+z);**



> **M3:=array(1..2,1..2,[[x,x^2],[x^3,x^4]]);**



> **map(diff,M3,x);# Function map – uses the given operation to every element of matrix;**



> **map(int,%,x);**



> **map(sin,M3);**



> **# Generation of functional matrix;**

> **f[1]:=(i,j)->x^i+y^j;**



> **AA:=matrix(2,3,f[1]);**



> **restart:**

> **A:=Matrix(3,3,[[omega,2,3],[4,5,6],[7,8,9]],readonly=true);**



> **omega:=1:**

> **as:=Matrix(3,3,readonly=false);**



> **as:=A;**



> **omega:=2:**

> **print(as);**



> **restart:**

> **a:=(omega)->matrix(2,2,[[12,omega],[43,566]]);**



> **a(23);**



Solving the system of linear equations С\*X=B:

> **restart:with(linalg):**

Warning, the protected names norm and trace have been redefined and unprotected

> **C:=matrix(3,3,[[4,8,2],[6,2,3],[3,7,11]]);**



> **B:=matrix(3,1,[5,6,1]);**



> **X:=matrix(3,1);**



> **X:=evalm((C^(-1))\*B);# the first method;**



> **X:=multiply(inverse(C),B); # the second method**

> **X:=linsolve(C,B); # the third method;**



> **# Interactive input of matrix;**

> **restart;**

> **with(linalg):**

Warning, the protected names norm and trace have been redefined and unprotected

> **#A:=array(1..3,1..3);**

> **#entermatrix(A);**

> **1;**



> **3;**



> **2;**



> **5;**



> **7;**



> **5;**



> **2;**



> **5;**



> **9;**

**1.11. Transformation of complex numbers, analytical expressions & functions of a complex variable**

Operations with complex numbers:

> **restart: with(linalg): with(plots):**

> **z::complex:# Announcing z – as a complex number;**

> **z:=3+4\*I;**



> **Im(z);# False part of complex number;**



> **Re(z);# Material part of complex number;**



> **abs(z);#Module of complex number;**



> **argument(z); # Argument of complex number;**



> **evalf(%);**



> **conjugate(z);# Conjugate to complex number;**



> **AA:=polar(z);# Trigonometric form of complex number ;**



> **exp(z);# Exponent of complex number;**



> **A:=evalf(evalc(exp(z)));**



> **Re(A);**



> **Im(A);**



Functions of complex variable:

> **restart:with(plots):**

> **w,p::complex:**

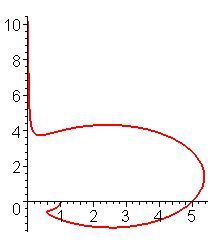
> **w:=(p)->(1+p^4)/(1+2\*p+p^3);**



> **p:=I\*omega;**



> **complexplot(w(p),omega=0..10,thickness=2);**



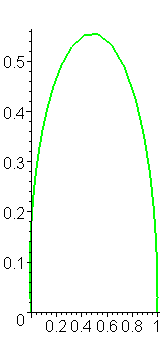
> **# Drawing in (Re w, Im w)of hodograrh:**

> **restart: with(plots):**

> **w:=(1+0.1\*p)/(1+0.01\*p^2+p^3);p:=I\*omega:**



> **complexplot(w,omega=0..100,color=black,thickness=2);**



> **restart:**

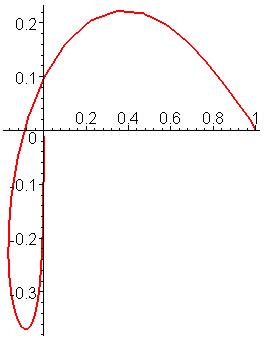
> **p:=I\*omega;**



> **w:=(1+0.1\*p+p^2)/(1+0.01\*p^2+p^3);**



* **plot([Re(w),Im(w),omega=0..100],thickness=2);**



> **restart: with(plots):**

> **u,v,w,p::complex:**

> **p:=I\*omega;**



> **u:=(4+0.1\*p)/(1+0.01\*p^2);**



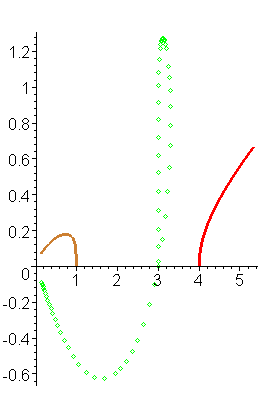
> **v:=(3+p)/(1+0.01\*p^5);**



> **w:=(1+0.1\*p)/(1+0.01\*p^4);**



> **complexplot([u,v,w],omega=0..5,color=[RED,GREEN,GOLD],style=[line,point,line], thickness=3);**



Transformation of complex expressions:

> **restart:**

> **p:=I\*omega;**



> **w:=(a[1]+a[2]\*p+a[3]\*exp(p\*tau)\*p^2+a[4]\*p^5)/(b[1]+b[2]\*p^2+b[3]\*p^3);**



> **simplify(evalf(evalc(w)));**



> **simplify(evalf(evalc(conjugate(w))));# Сопряженное от w;**



>

> **evalf(evalc(Re(w)));**



> **evalf(evalc(Im(w)));**



>

Work with complex functions

>

> **# Find the image of single circle of complex plane z(t)=exp(I\*t) by transformation w(t)=(z(t)-0.5\*z(Pi/4))/(1-0.5\*z(-Pi/4)\*z(t));**

> **restart;**

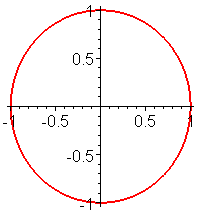
> **z(t),w(t)::complex;**



> **z:=(t)->exp(I\*t);**

 **plot([**

>**evalc(Re(z(t))),evalc(Im(z(t))),t=0..2\*Pi],thickness=2);**



> **w:=(t)->(z(t)-0.5\*z(Pi/4))/(1-0.5\*z(-Pi/4)\*z(t));**



> **w(t):=simplify(evalc(w(t)));**



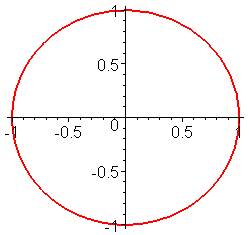
> **wRe(t):=simplify(evalc(Re(w(t))));**



> **wIm(t):=simplify(evalc(Im(w(t))));**



* **plot([wRe(t),wIm(t),t=0..2\*Pi],thickness=2);**



So , the manual is concerned with the packages of functions of the system of analytical calculations Maple, methods of setting tasks to functional dependencies & of drawing their graphs, examples of calculating limits, derivatives & integrals, operations with rows, solving equations, inequalities & their systems, analysis of functions, solving differential equations & their systems, operations with matrices & vectors, transforming complex numbers, expressions & functions of complex variable. For each case there are examples ,illustrating the implementation of function of Maple system. This set of function is sufficient for computer modelling of particular dynamic systems, analyzed by the authors of this manual at different times.

The above mentioned approaches to using the potentialities of the system of analytical calculations Maple for carrying out numerical procedures can be used in working out software in terms of writing term-papers ,Bachelors ‘- & Masters’ papers. Judging by the authors’ experience in this field , the main task for students consists ,as a rule , of elaborating the program , implementing an algorithm on the basis of the received mathematical model. Hence ,it is desirable that identificators in the program should be similar to the ones , given in the program of your scientific adviser. For this reason, the Appendix presents the Greek alphabet & identificators of its letters < small & capital >, used in the system of analytical calculations Maple.

**3. Test assignments for doing laboratory tests on computers**

Assignment N 1.

1. Drawing graphs of functions:

|  |  |  |
| --- | --- | --- |
| а)y = | b) y= | c) z = |
| d) | e) |  |

2. Solving equations & systems of equations:

а)  b) 

3. Vectors:

а)  Define = corner 

b)  Define  – vectorial

c)  Find – mixed

4. Find f(X) & eigen numbers of matrix f(X)

f(X)= X = 

5. Limits

а) b)

6. Find the derivative:

а) y =  b)  =?

c)   d)  =?

e) z=  

7. Integral:

а)  b)  c) 

d)  D:  e) 

8. Complex numbers, polynomials:

а) Draw in (Re(W), Im(W) ) hodograph:

W= ;

б) Calculate the roots & draw them in (Re,Im): 

9. Analyze coincidence of rows:

а)  b)

c) Expand into Tailor row in the field : y=, 

d) Expand into Fourier row in the given interval:  x

10. Solve differential equations & systems of equations:

а) , 

b) 

c) 

Assignment N2.

1. Drawing graphs of functions:

|  |  |  |
| --- | --- | --- |
| а) y= | b) y= | c) z= |
| d) | e) |  |

2. Solving equations & systems of equations:

а)  b) 

3. Vectors:

а)  Find corner =corner 

b)  Find  – vectorial

c)  Find – mixed product4. Find *f*(X) & eigen numbers of matrix *f*(X)

*f*(X)= X=

5. Limits

а) b)

6. Find the derivative:

а) y=  b)  =?

c)   d)  =?

e) z =  

7. Integrals:

а)  b)  c) 

d)  D: 

e) 

8. Complex numbers, polynomials:

а) Draw in (Re(W), Im(W)) hodograph:

W= 

б) Calculate the roots & draw them in (Re, Im): .

9. Analyze coincidence of rows:

а)  б) 

в) Expand into Taylor row in the field : y=, 

г) Expand into Fourier row in the given interval:  

10. Solve differential equations & systems of equations:

а), 

б) 

в) 

Assignment № 3.

1. Drawing graphs of functions:

|  |  |  |
| --- | --- | --- |
| а) y= | б) y= | в) z= |
| г)  t | д) |  |

2. Solving equations & systems of equations:

а) б) 

3. Vectors:

а)  Find  = corner 

b)  Find  – vectorial product

c)  Find – mixed product

4. Find *f*(*x*) & eigen numbers of matrix *f*(X)

*f*(X)= X=

5. Limits

а) б)

6. Find the derivative:

а) y=x+  б)  =?

c)   d)  =?

e) z=arcsin()  

7. Integrals

а)  b)  c) 

d)  D:  e) 

8. Complex numbers, polynomials:

а) Draw in (Re(W), Im(W)) hodograph: W=, 

б) Define L.C.D. of polynomials:

9. Analyze coincidence of rows:

а)  б) 

c) Expand into Taylor row in the field : y=, 

d) Expand into Fourier row in the given interval:

x

10. Solve differential equations & systems of equations:

а) , 

б) 

в) 

Assignment № 4.

1. Drawing graphs of functions:

|  |  |  |
| --- | --- | --- |
| а) y= | б) y= | в) z= ln (--) |
| c)  t[-2,2] | d) 5(1+cosφ),  φ[0,2π] |  |

2. Solving equations & systems of equations:

а) (++1)(2+2-3)=-3(1--) б) 

3. Vectors:

а) corner between

б) Find  – vectorial product

c) Find mixed product  –

4. Find *f*(X) & eigen numbers of matrix *f*(X)

**** 

5. Limits

а)  б) 

6. Find the derivative:

а)   б) , 

c) ,  e)  

d) , , 

7. Integrals

а)  б)  c) 

d)  D: ,  e) 

8. Complex numbers , polynomials:

а) Draw in (Re(W), Im(W)) hodograph:

, ,

9. Analyze the coincidence of rows:

а)  б) 

в) Expand into Taylor row in the field : , 

10. Solve differential equations & systems of equations:

а) , , 

б) 

c) 

Assignment № 5.

1. Drawing graphs of functions:

|  |  |  |
| --- | --- | --- |
| а) | б) | c) |
| d) | e) , |  |

2. Solving equations & systems of equations.

а)  б)

3. Vectors.

а) ; ; Find corner between vectors 

б) ; ; Find vectorial product 

c) ; ; ; Find mixed product 

4. Find *F*(*x*) & eigen numbers of matrix *F*(X).

5. Limits.

а)  б) 

6. Derivatives.

а) ,  б) , 

c) ,  d) , 

e)  , , 

7. Integrals  б)  c) 

d)  D: e) 

8. Complex numbers, polynomials.

а) Draw in (Re(W), Im(W)) hodograph:

, , 

б) Define LCD of polynomials.

9. Analyze coincidence of rows:

а)  б) 

в) Expand into Taylor row of function in the field x0:

, 

г) Expand into Fourier row in the given interval:

, 

10. Solve differential equations & systems of equations.

а) , , 

б) 

c) 

Assignment № 6.

1. Draw graphs of functions:

|  |  |  |
| --- | --- | --- |
| а) y= | б) y= | c) z =x2+y2 |
| d) | e) , |  |

2. Solving equations & systems of equations:

а) (x+1)(x+2)(x+3)(x+4)=120; б) 

3. Vectors:

а)  Find = corner 

б)  Find  – vectorial product

c)  Find – mixed product

4. Find *f*(X) =, if X = .

Find eigen numbers of matrix *f*(*x*).

5. Limits

а) б)

6. Find the derivative:

а) y=sin(cos2(x))cos(sin2(x)))  б)  =?

c)   d)  =?

e) z=  

7. Integrals:

а) б) c)

d) D:  e) 

8. Complex numbers, polynomials:

а) Draw in (Re(W), Im(W)) hodograph:

*W*= 

б) Define L.C.D. of polynomials:

*P*6(*x*) = ; *P*5(*x*) = 3*x*5–7*x*3+3*x*2–7;

9. Analyze coincidence of rows:

а)  б) 

c) Expand into Taylor row in the field : y=, 

d) Expand into Fourier row in the given interval:  

10. Solve differentials equations & systems of equations:

а) , 

б) 

c) 

Assignment № 7.

1. Draw the graphs of functions.

|  |  |  |
| --- | --- | --- |
| а)  y= | б)  y= | c)  z = cos (x)\*cos(y) |
| d)  t | e) |  |

2. Solving equations & systems of equations.

а) ( 3–*x*)4+(2–*x*)4=(5–2*x*)4 б) 

3. Vectors.

а) , . Find  .

b) , . Find  – vectorial product.

c) , , . Find – mixed product.

4. Find *f*(*X* )= X2+2X+3 , if

*X*= , Find eigen numbers of matrix f(X).

5. Limits.

a)  b)

6. Derivatives.

а) y = 1/cos5 (x), , b) , , c), 

d) x2/4+y2/9=1,  e) z =,  

7. Integrals.

а) b) c)

d) D: e)

8. Complex numbers, polynomials.

a) Draw in (Re(w),Im(w)) hodograph :

*w*= , ,

б) Define L.C.D of polynomials: *p*5(*x*) = *x*5–10*x*4–*x*1

9. Analyze coincidence of rows.

а)  b) 

c) Expand into Taylor row in the field x0: y=

d) Expand into Fourier row in the given interval: , 

10. Solve differential equations & systems of equations.

a)   b) 

c) 

Assignment № 8.

1. Draw graphs of functions:

а) y=;

b) y=;

c) z =x3+;

d) *t*

e) , 

2. Solving equations & systems of equations:

а) x4+1=2(1+x)4; b) 

3. Vectors:

а)  Find  = corner

b)  Find  – vectorial product

c)  Find – mixed product

4. Find *f*(X) & eigen numbers of matrix *f*(X)

*f*(X)=, X=

5. Limits

а) b)

6. Find the derivative:

а) y=tg()  b)  =?

c)   d) =?

e) *Z*=  

7. Integrals:

а)  b)  c) 

d)  D:  e) 

8. Complex numbers , polynomials:

а) Draw in (Re(W), Im(W)) hodograph: *W*= 

b) Define L.C.D. of polynomials

*P*4(*x*) = ; *P*3(*x*) = *x*3+*x*2–*x*;

9. Analyze coincidence of rows:

а)  b) 

c) Expand into Taylor row in the field : *y*= *e*2*x*, 

d) Expand into Fourier row in the given interval:  

10. Solve differential equations & systems of equations:

а)  b) 

c) 

Assignment № 9.

1. Draw graphs of functions:

а) y= b) y= c) z = d)

t

e) ; 

2. Solving equations & systems of equations:

а) 

b) 

3. Vectors:

а) Find  =  corner 

b) Find  – vectorial

c) Find – mixed product

4. Find *f*(X)= , if

X = Find eigen numbers of matrix *f*(X).

5. Limits

а)

b)

6. Find the derivative :

а) y=  b)  =?

c)   d)  =?

д) *z* =  

7. Integrals:

а)  b)  c) 

d)  D:  e) 

8. Complex numbers , polynomials:

а) Draw in (Re(W), Im(W)) hodograph:

W= 

b) Solve equation:



9. Analyze coincidence of rows:

а)  б) 

c) Expand into Taylor row in the field : *y*=, 

d) Expand into Fourier row in the given interval:  

10. Solve differential equations & systems of equations:

а) , 

b) 

c) 

Assignment № 10.

1. Draw graphs of functions:

а) y= б) y= c) z =

d)  t

e)  

2. Solving equations & systems of equations:

а) 

б) 

3. Vectors :

а) Find  = corner 

б) Find  – vectorial product

c) Find – mixed product

4. Find f(X)= if

X=  Find eigen numbers of matrix *f*(X).

5. Limits

а)

б)

6. Find the derivative:

а) *y*=  б)  =?

c)   d)  =?

e) *z*=  

7. Integrals:

а)  б)  c) 

d)  D:  e)  D: 

8. Complex numbers, polynomials:

а) Draw in (Re(W), Im(W)) hodograph:

W= 

б) Expand into factors:



9. Analyze coincidence of rows:

а)  б) 

10. Expand in Taylor row in the field : y=, 

11. Solve differential equations & systems of equations:

а) , 

б) 

c) 

Assignment № 11.

1. Draw the graph of function.

а)  б)  c) 

d)  t[0,2]

e)  , 

2. Solving equations & systems of equations.

a) 

б) 

3. Vectors.

a)  (0,1,3) find 

б) (-3,0,2) (0,4,7) find – vector of product

c) (1,2,3)(4,3,2)(-1,-2,-4) find  – mixed product

4. Find:

*f*(X)= , если X=

Find eigen numbers of matrix *f*(*x*)

5. Limits

a) б)

6. Derivatives.

a) *y*=  б) 

c)  

d)  

e)   

7. Integrals.

а)  б) 

c)  г)  

d) 

8. Complex numbers , polynomials.

а) Draw in (Re(W), Im(W)) hodograph

W=  

б) Expand into factors



9. Analyze coincidence of rows & expand:

а)  б) 

c) Expand into Taylor row in the field :

, 

г) Expand into Fourier row in the given interval:

, 

10. Solve differential equations & systems of equations:

а)  , 

б) 

c) 

Assignment № 12.

1. Draw graphs of functions.

a) y =  б) y = e) ρ =2sin2φ

φ

c) *z* =arсcos(x+y)

d)  t



2. Solving equations & systems of equations.

а) *x*4+5*x*3+2*x*2+5*x*+1=0

б) 



3. Vectors.

а) , . Find  .

б) , . Find  – vectorial product.

c) , , . Find  – mixed product.

4. Find *f*(*X* )= X2+4X+2 , if *X*= ,

Find eigen numbers of matrix *f*(*X*).

5. Limits.

а)  б)

6. Derivatives.

а) *y* =*℮x*(*x*2-2*x*+2) ,  б), 

c),  d) xexy +cos(y2)=0, 

e) z =,  

7. Integrals.

а) б) c)

d) д :

e) д : 

8. Complex numbers, polynomials.

а) Draw in (Re(w),Im(w)) hodograph :

*w*= , ,

б) Expand into factors:

*p*4(*x*) = *x*4–10*x*2+1

9. Analyze the coincidence of rows.

а)  б) 

c) Expand into Taylor row in the field x0:

*y*=

d) Expand into Fourier row in the given interval:

10. Solve differential equations &systems of equations .

a)  

б) 

c)

Assignment № 13.

1. Draw graphs of functions:

а) *y* = б) *y* = c) *z* =

d) t

e)  

2. Solving equations & systems of equations:

а) 

б) 

3. Vectors:

а) Find =corner 

б) Find  – vectorial product

c) Find mixed product –

4. Find *f*(X)= , if

X= Find eigen numbers of matrix *f*(X).

5. Limits

а)

б)

6. Find the derivative:

а) y=  б)  =?

c)   d)  =?

e) *z*=  

7. Integrals:

а)  б)  c) 

d)  D:  e) , D: 

8. Complex numbers, polynomials:

а) Draw in (Re(W), Im(W)) hodograph

*W*= 

б) Expand into factors:



9. Analyze coincidence of rows:

а)  б) 

c) Expand into Taylor row in the field : y=, 

d) Expand into Fourier row in the given interval:  

10. Solve differential equations & systems of equations:

а) , 

б) 

c) 

Assignment №14.

1. Draw graph of function

а)  б)  c) z=e

d)  t[0,2]

e) =3(1-сos())

2. Solving equations & systems of equations.

a) 

б)



3. Vectors

a)  (0,1,3) find 

б) (-3,0,2) (0,4,7) find  – vector of factor) (1,2,3)(4,3,2)(-1,-2,-4) Find  – mixed product

4. Find :

*f*(X)= , if X=

Find eigen numbers of matrix *f*(X)

5. Limits

a) б)

6. Derivatives.

a) *y*= 

б) 

c)  

d)  

e)   

7. Integrals.

а)  б) 

c)  d)  

e) 

8. Complex numbers, polynomials.

а) Draw inв (Re(W), Im(W)) hodograph

*W*= ,  

б) Expand into factors



9. Analyze coincidence of rows & expand:

а)  б) 

c) Expand into Taylor row in the field :

, 

d) Expand into Fourier row in the given interval:

, 

10. Solve differential equations & systems of equations:

а)  , 

б) 

c) 

Assignment № 15.

1. Draw graphs of functions:

а) y= б) y= c) z = d)

t

e)  

2. Solving equations & systems of equations:

а) 

б) 

3. VECTORS:

а) find = corner 

б) find  – vectorial

c) find – mixed

4. Find *f*(X)= if

X= Find eigen numbers of matrix *f*(X).

5. Limits

а)

б)

6. Find the derivative:

а) *y* =  б)  =?

c)   d)  =?

e) *z*=  

7. Integrals:

а)  б)  c) 

d)  D:  e) 

8. Complex numbers , polynomials:

а) Draw in (Re(W), Im(W)) hodograph:

W = 

б) Calculate the roots & draw them in (Re,Im):



9. Analyze coincidence of rows:

а)  б) 

c) Expand into Taylor row in the field : y =, 

d) Expand into Fourier row in the given interval:  

10. Solve differential equations & systems of equations:

а) , 

б) 

c) 

**The list of used sources:**

1. Дьяконов В. Maple 7: учебный курс. – СПб.: Питер, 2002. 672 с.: ил.
2. Дьяконов В.П. MAPLE 8 в математике, физике и образовании. – М.: СОЛОН-Пресс, 2003. 656 с. (Серия «Полное руководство пользователя»).
3. Дьяконов В.П. MAPLE 9 в математике, физике и образовании. – М.: СОЛОН-Пресс, 2004. 688 с.
4. Говорухин В., Цибулин В. Компьютер в математическом исследовании. Учебный курс. – СПб.: Питер, 2001. 624 с.: ил.
5. Васильев А.Н. MAPLE 8. Самоучитель. – М.: Издательский дом «Вильямс», 2003. 352 с.: ил.

APPENDIX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Names of letters |  |  |  |  |
| *Альфа* | alpha | **α** | Alpha | **Α** |
| *Бетта* | beta | **β** | Beta | **Β** |
| *Гамма* | gamma | **γ** | Gamma | **Γ** |
| *Дельта* | delta | **δ** | Delta | **Δ** |
| *Эпсилон* | epsilon | **ε** | Epsilon | **Ε** |
| *Дзэта* | zeta | **ζ** | Zeta | **Ζ** |
| *Эта* | eta | **η** | Eta | **Η** |
| *Тхэта* | theta | **θ** | Theta | **Θ** |
| *Йота* | iota | **ι** | Iota | **Ι** |
| *Каппа* | kappa | **κ** | Kappa | **Κ** |
| *Ламбда* | lambda | **λ** | Lambda | **Λ** |
| *Мю* | mu | **μ** | Mu | **Μ** |
| *Ню* | nu | **ν** | Nu | **Ν** |
| *Кси* | xi | **ξ** | Xi | **Ξ** |
| *Омикрон* | omicron | **ο** | Omicron | **Ο** |
| *Пи* | pi | **π** | Pi | **Π** |
| *Ро* | rho | **ρ** | Rho | **Ρ** |
| *Сигма* | sigma | **σ** | Sigma | **Σ** |
| *Тау* | tau | **τ** | Tau | **Τ** |
| *Ипсилон* | upsilon | **υ** | Upsilon | **Υ** |
| *Фи* | phi | **ϕ** | Phi | **Φ** |
| *Хи* | chi | **χ** | Chi | **Χ** |
| *Пси* | psi | **ψ** | Psi | **Ψ** |
| *Омега* | omega | **ω** | Omega | **Ω** |
|  |  |  |  |  |

**Greek alphabet**

Сергей Николаевич **Стребуляев**

Надежда Львовна **Орлова**

**КОМПЬЮТЕРНАЯ АЛГЕБРА «MAPLE»**

**В ИНЖЕНЕРИИ**

Учебно-методическое пособие

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского».

603950, Нижний Новгород, пр. Гагарина, 23.