Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering
Department of Mathematics

The Mathematics 4 course in Wolfram Mathematica

Bachelor's Degree Project

Author: **Eva Lorencová**
Supervisor: **Ing. Matěj Tušek, Ph.D.**
Academic Year: 2013/2014

Acknowledgments:

I would like to express my profound gratitude to my supervisor Ing. Matěj Tušek, Ph.D. for his expert guidance and encouragement throughout my work on the project, and thank to Irena Dvořáková, prom. fil. for language corrections. Finally, I want to express deepest thanks to my family for supporting me during my studies.

Declaration:

I declare that this project is my own work and I have cited all sources I have used in the bibliography. I agree that the Department can use it for non-commercial purposes in the future.

In Prague, July 7, 2014 Eva Lorencová

Název práce:

Kurz Matematika 4 ve Wolfram Mathematica

Autor: Eva Lorencová

Obor: Inženýrská informatika

Zaměření: Praktická informatika

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Matěj Tušek, Ph.D., Katedra matematiky, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze

Abstrakt: Práce se zabývá vybranými částmi kurzu Matematika 4. Tento kurz je určený pro studenty Fakulty jaderné a fyzikálně inženýrské Českého vysokého učení technického v Praze a zahrnuje zejména diferenciální a integrální počet funkcí více proměnných, dále potom základní obyčejné diferenciální rovnice. Všechny problémy jsou řešeny v počítačovém algebraickém systému Wolfram Mathematica. Některé vybrané pojmy a problémy jsou zpracovány jako interaktivní vizualizace. Dále je numericky řešen komplexnější fyzikální problém. Celá práce je napsaná jako sešit ve Wolfram Mathematica, verze 9.0.

Klíčová slova: kurz Matematika 4, vizualizace, Wolfram Mathematica.

Title:

The Mathematics 4 course in Wolfram Mathematica

Author: Eva Lorencová

Abstract: The project deals with selected parts of Mathematics 4 course. This course is intended for students of the Faculty of Nuclear Sciences and Physical Engineering of the Czech Technical University in Prague and primarily includes multi-variable calculus and basic ordinary differential equations. All problems are solved in computer algebra system Wolfram Mathematica. Some selected notions and problems are processed as interactive visualizations. Moreover, a more complex physical problem is solved numerically. The whole project is written as a notebook in Wolfram Mathematica, version 9.0.

Key words: Mathematics 4 course, visualization, Wolfram Mathematica.

Contents

Preface **6**

1 Introduction to Wolfram Mathematica **7**

1.1 Wolfram Mathematica . . . 7

1.2 WolframAlpha . . . 7

2 The Mathematics 4 course in examples **8**

2.1 Differential equations . . . 8

2.2 Multivariable limits . . . 9

2.3 Partial derivative . . . 11

2.4 Directional derivative . . . 14

2.5 Local extrema . . . 18

2.6 Constrained extrema . . . 22

2.7 Double integral . . . 26

3 Interactive visualizations **32**

3.1 Solution to the differential equation . . . 32

3.2 Directional derivative . . . 33

3.3 Constrained extrema . . . 35

3.4 Useful alternative coordinate systems . . . 37

3.4.1 Polar coordinate system . . . 37

3.4.2 Spherical coordinate system . . . 38

3.4.3 Cylindrical coordinate system . . . 39

3.5 Double integral . . . 40

4 Bivariate limits **43**

4.1 Bivariate limits in Mathematica . . . 43

4.2 Function doubleLimit . . . 43

4.2.1 Method of critical paths . . . 45

4.2.2 Possible issues . . . 46

5 Ballistic curves **47**

5.1 Ballistic trajectory without air resistance . . . 47

5.2 Ballistic trajectory with air resistance . . . 47

5.2.1 Drag proportional to velocity . . . 48

5.2.2 Drag proportional to velocity squared . . . 49

5.3 Visualization . . . 49

Conclusion **52**

Preface

All students of our Faculty have to take the courses of mathematics. To go through this process, they use books, university textbooks, materials from lectures, etc. We wanted to help them by creating some interactive study material. For this reason, the project processes one of the courses of mathematics of our Faculty – the Mathematics 4 course.

The project is divided into four main parts. The first part deals with the particular topics of Mathematics 4 course. For each topic, the most important definitions and theorems are given and some specific examples solved. The source code written in Mathematica is included.

In the next part, interactive visualizations of some notions and problems are developed. They could be used not only by students but also by their lecturers as an interactive tool during classes. They are developed in the spirit of demonstrations at [16] but tailored to the students of our Faculty.

As there is no built-in function for calculating bivariate limits in Mathematica, the third part of the project contains a function which can calculate some of them.

Finally, as a practical application of the skills developed during the Mathematics 4 course, a physical problem is solved – namely the problem of ballistic curves.

As has been already said, the main goal of the project is to create an interactive tool for students of our Faculty. They can see that there is another way how to solve a problem and that Mathematica could help them with the task they are not able to solve due to difficult calculations. Furthermore, some more difficult and interesting problems are solved.

The whole project will be available on the Internet and is written as a notebook in Mathematica in order that students could work with it and could change the content in any way they want.

1 Introduction to Wolfram Mathematica

1.1 Wolfram Mathematica

Wolfram Mathematica, or simply Mathematica, is a computational software program based on symbolic mathematics. It was invented by Stephen Wolfram in 1988 and is developed by Wolfram Research. The latest version is available for three major platforms – Microsoft Windows, MacOS X, and Linux (supporting both 32-bit and 64-bit implementations) [7].
Mathematica is divided into two main parts. The code is interpreted by the kernel, whereas the front end provides the graphical user interface. Much of the content can be edited interactively or generated by an algorithm [14].
Programming in Mathematica differs from programming in other languages. The code consists of primitives. For a new user of Mathematica, it can be a problem to find their right combination which would result in the simplest solution. Furthermore, Mathematica has changed a lot from the first version even though the core principles remain the same. Obsolete functions have been replaced or discarded. As many highly specialized functions have been added, the built-in library has increased, and even the most difficult tasks can be very often resolved in a very simple way [1].
Mathematica is not used only in mathematics and physics. Due to its strength, user-friendly interface, and possibility of creating interactive modules, it is often utilized in other fields such as biology or chemistry. Moreover, the source code can be accompanied by the text and the document can be formatted with ease, so it is possible to write whole articles using Mathematica only. Later versions also enable connection with other applications through MathLink protocol, parallel processing, dynamic interactivity, and high-performance computing. This makes Mathematica a complex tool with a wide range of applications [3].

1.2 WolframAlpha

Another project developed by Wolfram Research is a computational knowledge engine WolframAlpha. Unlike search engines, it does not give a list of websites containing the searched information but tries to give the answer directly by searching the web and using some dynamic computations [6].
WolframAlpha is based on Mathematica (its language and built-in functions). Additional data is gathered from various websites. For instance, some reports about users of Facebook are generated. They contain how often they use some specific word, how popular their friends are, etc. [15].
Since WolframAlpha is available on the Internet free of charge, it is very popular among students.

2 The Mathematics 4 course in examples

2.1 Differential equations

The Mathematics 4 course deals with three types of ordinary differential equations:

- separable differential equations,
- linear differential equations of 1st degree,
- linear differential equations of *n*th degree with emphasis on equations with constant coefficients.

In contrast to the hand calculation, Mathematica can handle the above mentioned types of differential equations with ease. Just use the command *DSolve* to find the solution.

Example 2.1.1: Find the solution of the equation $y' - 2\,xy = 3\,x^2\,e^{x^2}$ passing through the point $M = (0, 5)$ [11].

The command *DSolve* is used to search the solution. The initial condition is represented by the second equation.

```
sol = DSolve[{y'[x] - 2 x * y[x] == 3 x^2 * e^(x^2), y[0] == 5}, y[x], x]
```

```
{{y[x] → e^(x^2) (5 + x^3)}}
```

Let us have a look what the function *y* looks like.

```
Plot[y[x] /. sol[[1]], {x, -2, 1}, PlotStyle → Black, ImageSize → 300]
```

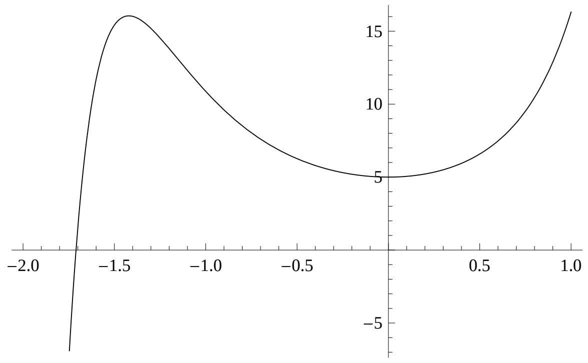


Figure 2.1.1: Solution of the differential equation $y' - 2\,xy = 3\,x^2\,e^{x^2}$.

The domain of *y* is the set of all real numbers.

```
Clear[sol];
```

Example 2.1.2: Find the solution of the equation $y' = \cos(x)$ passing through the point $M = (0, 1)$.

Again, the command *DSolve* is used.

```
DSolve[{y'[x] == Cos[x], y[0] == 1}, y[x], x]
```

```
{{y[x] → 1 + Sin[x]}}
```

The domain of *y* are all real numbers.

Example 2.1.3: Find the general solution of the equation $y'' + 2y' + y = 3 e^{-x}\sqrt{x+1}$ [11].

```
DSolve[y''[x] + 2*y'[x] + y[x] == 3*e^-x*√(x + 1), y[x], x]
```

```
{{y[x] → 4/5 e^-x (1 + x)^5/2 + e^-x C[1] + e^-x x C[2]}}
```

As it can be seen from the solution, $Dom(y) = (-1, +\infty)$.

2.2 Multivariable limits

Definition 2.2.1:

*Let $M \subset \mathbb{R}^n$ and $\vec{a} \in \mathbb{R}^n$. We say that $\vec{a}$ is a **limit point** of M if and only if $U^*_\epsilon(\vec{a}) \cap M \neq \emptyset$ for all $\epsilon > 0$, where $U^*_\epsilon(\vec{a})$ is the punctured ϵ-neighbourhood of $\vec{a}$.*

Definition 2.2.2:

*Let $f : \mathbb{R}^n \to \mathbb{R}$ and $\vec{a}$ be a limit point of the domain of f. The **limit** of f, as $\vec{x}$ tends to $\vec{a}$, is $c \in \mathbb{R}$, written as $\lim_{\vec{x}\to\vec{a}} f(\vec{x}) = c$ if and only if for every $\epsilon > 0$ there exists $\delta > 0$ such that for all $\vec{x} \in D_f$, $0 < \|\vec{x} - \vec{a}\| < \delta$ implies $|f(\vec{x}) - c| < \epsilon$.*

Definition 2.2.3:

*Let $f : \mathbb{R}^n \to \mathbb{R}$, $M \subset \mathbb{R}^n$, and $\vec{a}$ be a limit point of $D_f \cap M$. The **limit** of f, as $\vec{x}$ tends to $\vec{a}$, **with respect to the set M** is $c \in \mathbb{R}$, written as $\lim_{\vec{x}\to\vec{a},\, \vec{x}\in M} f(\vec{x}) = c$, if and only if for every $\epsilon > 0$ there exists $\delta > 0$ such that for all $\vec{x} \in U^*_\delta(\vec{a}) \cap D_f \cap M$ we have $|f(\vec{x}) - c| < \epsilon$.*

Theorem 2.2.1:

Let $f : \mathbb{R}^2 \to \mathbb{R}$, $\vec{a} = (a_x, a_y)$. If the limit of f, as $\vec{x}$ tends to $\vec{a}$, is equal to $c \in \mathbb{R}$ and the sequential limits exist and are finite, then

$$\lim_{x\to a_x}\left(\lim_{y\to a_y} f(\vec{x})\right) = \lim_{y\to a_y}(\lim_{x\to a_x} f(\vec{x})) = c.$$

Corollary 2.2.1:

If both sequential limits exist and are finite but different, the overall limit does not exist.

Theorem 2.2.2:

Let $f : \mathbb{R}^n \to \mathbb{R}$, $\vec{a}$ be a limit point of D_f, and $\lim_{\vec{x}\to\vec{a}} f(\vec{x})$ exists. Then for any $A \subset \mathbb{R}^n$ such that $\vec{a}$ is a limit point of A,

$$\lim_{\vec{x}\to\vec{a},\, \vec{x}\in A} f(\vec{x}) = \lim_{\vec{x}\to\vec{a}} f(\vec{x}).$$

Corollary 2.2.2:

Suppose $A \subset \mathbb{R}^n$, $B \subset \mathbb{R}^n$, and $\vec{a}$ is a limit point of them. If

$$lim_{\vec{x}\to\vec{a},\,\vec{x}\in A}\, f(\vec{x}) \neq lim_{\vec{x}\to\vec{a},\,\vec{x}\in B}\, f(\vec{x}),$$

then the overall limit does not exist.

Example 2.2.4: Let $g\,(x,\ y) = \frac{xy^2}{x^2+y^4}$. Show that $lim_{(x,y)\to(0,0)}\, g(x,\ y)$ does not exists.

```
g[x_, y_] = x*y^2/(x^2 + y^4);
```

The non-existence of the limit could be confirmed using sequential limits. If the sequential limits existed and were different, the limit of g would not exist.

```
Limit[Limit[g[x, y], y → 0], x → 0]

0

Limit[Limit[g[x, y], x → 0], y → 0]

0
```

The sequential limits are the same. Therefore, the decision whether the limit exists cannot be made using sequential limits. Nonetheless, the limit would be equal to zero provided it existed. Another method could be used to prove the non-existence of a limit. For instance, approaching (0, 0) via points which lie on the lines $x = ky$ or $y = kx$.

```
Limit[g[k*y, y], y → 0]

0

Limit[g[x, k*x], x → 0]

0
```

Again, the result provides no information on the non-existence of the limit. Finally, when we choose parabolae $x = ky^2$, we will get a significant result.

```
Limit[g[k*y^2, y], y → 0]

  k
──────
1 + k^2
```

The numerical value of the limits with respect to different parabolae clearly depends on k. Consequently, the overall limit does not exist. The level curves of the function g have the shape of parabolae $x = ky^2$. Along these parabolae, f attains different constant values.

```
Clear[g];
```

Example 2.2.5: Let $f(x,\ y) = \frac{2\,xy}{xy+2\,x-y}$. Show that $lim_{(x,y)\to(0,0)}\, f(x,\ y)$ does not exists [11].

```
f[x_, y_] = 2*x*y/(x*y + 2*x - y);
```

At first, let us have a look what the sequential limits look like.

```
Limit[Limit[f[x, y], y → 0], x → 0]

0

Limit[Limit[f[x, y], x → 0], y → 0]

0
```

Since they are the same, another method have to be used to prove the non-existence. Try to approach the point (0, 0) via lines $y = kx$.

```
Limit[f[x, y] /. y → k*x, x → 0]

0
```

Although the result is that the limit is 0 on these lines, it is not always true.

```
f[x, y] /. y → k*x // Simplify
```

$$\frac{2\,k\,x}{2 + k\,(-1 + x)}$$

It is possible to use L'Hospital's rule for $k = 2$.

$$\frac{\texttt{D[2 k*x, x]}}{\texttt{D[2 + k*(-1 + x), x]}}$$

```
2
```

To summarize, the limit does not exist because we were able to find two different paths on which the limit is not the same.

```
Clear[f];
```

2.3 Partial derivative

Definition 2.3.4:
Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is defined in a neighbourhood of $\vec{a} = (a_1, a_2, \ldots, a_n)$. ***Partial derivative*** *of f at the point $\vec{a}$ with respect to the i-th variable is defined as*

$$\frac{\partial f}{\partial x_i}(\vec{a}) := \lim_{h\to 0} \frac{f(a_1, a_2, \ldots, a_{i-1}, a_i + h, a_{i+1}, \ldots, a_n) - f(a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)}{h}$$

if the right-hand side exists and is finite.

Definition 2.3.5:
By a ***gradient*** *of a function f at the point $\vec{a}$ is understood*

$$grad\ f\,(\vec{a}) := \left(\frac{\partial f}{\partial x_1}(\vec{a}), \ldots, \frac{\partial f}{\partial x_n}(\vec{a})\right)$$

if all partial derivatives of f at $\vec{a}$ exist.

Remark 2.3.1:
The symbol ∇f is used for the gradient of f.

Note: The partial derivative of f at the point $\vec{a}$ with respect to x has a geometric interpretation of the slope of the tangent at the point $(\vec{a}, f(\vec{a}))$ to the curve given by the intersection of the plot of f with the plane that is perpendicular to the domain of f and contains x-axis. Similarly, the partial derivative of f at $\vec{a}$ with respect to y. These facts can be nicely illustrated on the function $f = \sqrt{1 - x^2 - y^2}$ at some point.

```
f[x_, y_] = √(1 - x² - y²) ;

a = {0.5, 0.4};

a3D = Append[a, f @@ a];

gradient = Grad[f[x, y], {x, y}] /. {x → a3D[[1]], y → a3D[[2]]}

{-0.650945, -0.520756}

hemisphere = Plot3D[f[x, y], {x, -1, 3}, {y, -1, 3},
   Mesh → {{a3D[[1]]}, {a3D[[2]]}}, MeshStyle → Thickness[0.003],
   FaceGrids → {{0, 0, -1}}, FaceGridsStyle → Directive[Dotted],
   BoxRatios → {4, 4, 1}, AxesLabel → {x, y, z},
   ColorFunction → "Aquamarine"];
```

The tangents are defined because we know the starting point and the direction vector.

```
line1[x_] = gradient[[1]] * (x - a3D[[1]]) + a3D[[3]];

line2[y_] = gradient[[2]] * (y - a3D[[2]]) + a3D[[3]];

tangents = {ParametricPlot3D[{x, a3D[[2]], line1[x]}, {x, -1, 3},
    PlotStyle → {Thickness[0.003], Black}],
   ParametricPlot3D[{a3D[[1]], y, line2[y]}, {y, -1, 3},
    PlotStyle → {Thickness[0.003], Black}]};
```

Other objects are defined to make the illustration more lucid.

```
planes = Graphics3D[{FaceForm[GrayLevel[0.5]], EdgeForm[], Opacity[0.3],
    Polygon[{{-1, a3D[[2]], 0}, {-1, a3D[[2]], 1}, {3, a3D[[2]], 1},
      {3, a3D[[2]], 0}}],
    Polygon[{{a3D[[1]], 3, 0}, {a3D[[1]], 3, 1}, {a3D[[1]], -1, 1},
      {a3D[[1]], -1, 0}}]}];

lines = Graphics3D[{Black, Dashed, Thickness[0.003],
    Line[{{-1, a3D[[2]], 0}, {3, a3D[[2]], 0}}],
    Line[{{a3D[[1]], -1, 0}, {a3D[[1]], 3, 0}}]}];

point1 = x /. Solve[line1[x] == 0];

point2 = y /. Solve[line2[y] == 0];

angle1 = ParametricPlot3D[{{x, a3D[[2]], √(0.4 - (x - point1)²)}}, {x, 1, 1.15},
   PlotStyle → {Thickness[0.003], Black}];

angle2 = ParametricPlot3D[{{a3D[[1]], y, √(0.4 - (y - point2)²)}}, {y, 1.2, 1.31},
   PlotStyle → {Thickness[0.003], Black}];
```

```
text1 = Graphics3D[{Style[Text["αx", {1.2, a3D[[2]], 0.15}], 18],
     Style[Text["tx", {1, a3D[[2]], 0.7}], 18]}];
```

```
text2 = Graphics3D[{Style[Text["αy", {a3D[[1]], 1.5, 0.13}], 18],
     Style[Text["ty", {a3D[[1]], 1.2, 0.6}], 18]}];
```

```
Show[{hemisphere, planes, tangents, lines, angle1, angle2, text1, text2},
 ViewPoint → {4, 3, 1.3}, ImageSize → 500]
```

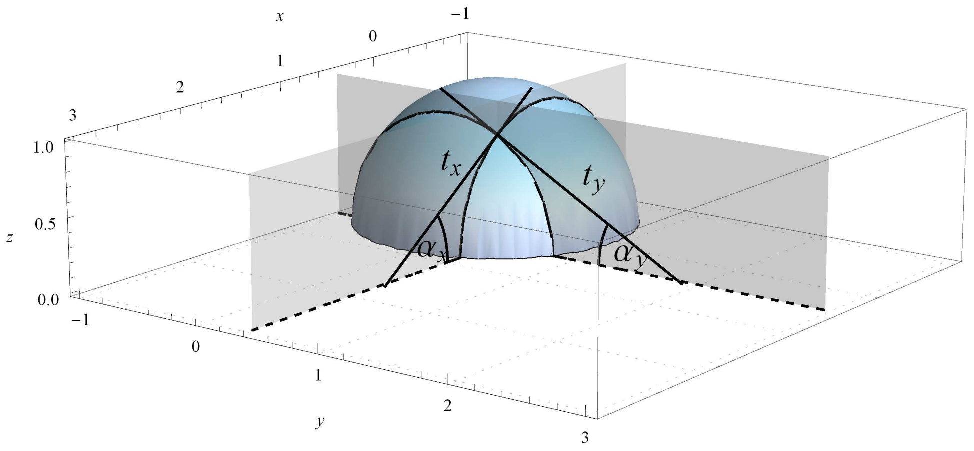


Figure 2.3.2: Geometric interpretation of the partial derivative.

```
Show[{hemisphere, planes, tangents, lines, angle1, text1}, ViewPoint → {0, ∞, 0},
 Ticks → None, AxesLabel → {x, None, z}]
```

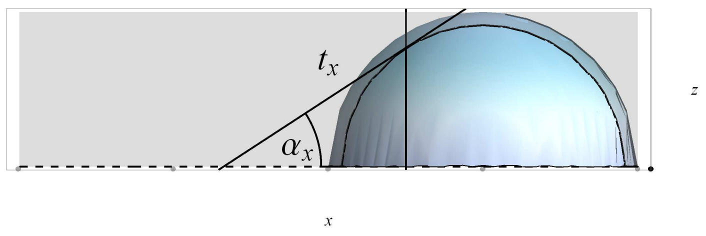


Figure 2.3.3: Geometric interpretation of the partial derivative with respect to *x* when *y* is fixed.

```
Show[{hemisphere, planes, tangents, lines, angle2, text2}, ViewPoint → {∞, 0, 0},
 Ticks → None, AxesLabel → {None, y, z}]
```

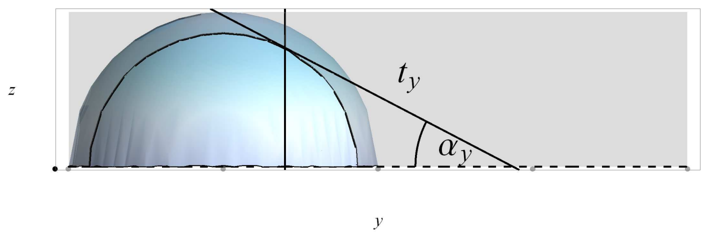


Figure 2.3.4: Geometric interpretation of the partial derivative with respect to *y* when *x* is fixed.

```
Clear[f, a2D, a, gradient, hemisphere, line1, line2, tangents, planes,
   lines, point1, point2, angle1, angle2, text1, text2];
```

Example 2.3.6: Calculate $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ of the function $f = x^3\, y^{y^4+y}$.

```
f[x_, y_] = x^3 * y^(y^4+y);

parcDerX = Simplify[D[f[x, y], x]]

3 x^2 y^(y+y^4)

parcDerY = Simplify[D[f[x, y], y]]

x^3 y^(y+y^4) (1 + y^3 + Log[y] + 4 y^3 Log[y])

Clear[f, parcDerX, parcDerY];
```

2.4 Directional derivative

Definition 2.4.6:
Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is defined in a neighbourhood of $\vec{a}$ and $\vec{s}$ is a non-zero vector in $\mathbb{R}^n$. A ***directional derivative*** *of f along $\vec{s}$ at $\vec{a}$ is defined as*

$$\frac{\partial f}{\partial \vec{s}}(\vec{a}) := \frac{1}{\|\vec{s}\|} lim_{h\to 0} \frac{f(\vec{a} + h\vec{s}) - f(\vec{a})}{h}$$

if the right hand side exists and is finite.

Theorem 2.4.3:
Suppose $f : \mathbb{R}^n \to \mathbb{R}$ and all partial derivatives at $\vec{a}$ are continuous. Then f has derivatives in all directions and

$$\frac{\partial f}{\partial \vec{s}}(\vec{a}) = \frac{1}{\|\vec{s}\|}\langle grad\ f(\vec{a}), \vec{s}\rangle. \qquad (1)$$

Example 2.4.7: Find the directional derivative of $f = \arctan(xy)$ along $\vec{s} = \left(\sqrt{2}, \sqrt{2}\right)$ at the point $\vec{a} = (0, 1)$.

```
f[x_, y_] = ArcTan[x*y];

s = {√2, √2};

a = {0, 1};
```

Definition of the function which can calculate directional derivative of the function of two variables follows:

```
directionalDerivative[f_, s_, a_] :=
  (Grad[f[x, y], {x, y}] /. {x → a[[1]], y → a[[2]]}).s / Norm[s];
```

```
directionalDerivative[f, s, a]
```

$$\frac{1}{\sqrt{2}}$$

```
Clear[f, s, a];
```

Example 2.4.8: Find the directional derivative of $f = \begin{cases} 0 & x = -y \\ (x+y)\sin\left(\frac{1}{(x+y)}\right) & x \neq -y \end{cases}$ along $\vec{s} = (1, -1)$ at the point $\vec{a} = (0, 0)$.

```
f[x_, y_] = If[x == -y, 0, (x + y) * Sin[1/(x + y)]];

s = {1, -1};

a = {0, 0};

Plot3D[f[x, y], {x, -0.1, 0.1}, {y, -0.1, 0.1}, ColorFunction → "Aquamarine",
 ViewPoint → {0, 1, 1}, ImageSize → 300]
```

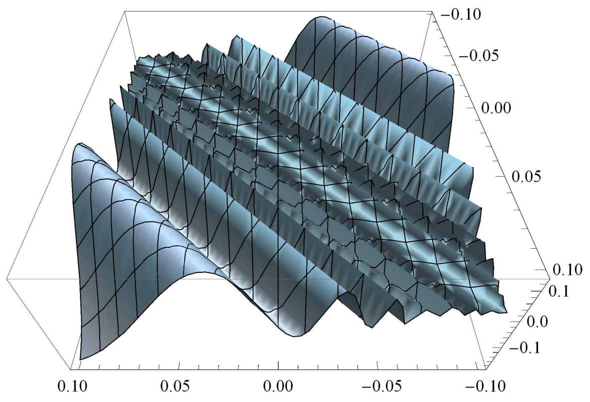


Figure 2.4.5: Function $f = \begin{cases} 0 & x = -y \\ (x+y)\sin\left(\frac{1}{(x+y)}\right) & x \neq -y \end{cases}$.

Equation (1) can be used if partial derivatives of f at the point $\vec{a}$ with respect to x and y exist.

```
derivativeX = D[f[x, y], x] /. {x → a[[1]], y → a[[2]]}

0

derivativeY = D[f[x, y], y] /. {x → a[[1]], y → a[[2]]}

0
```

Mathematica gave us the result that $\frac{\partial f}{\partial x}(\vec{a}) = 0$ and $\frac{\partial f}{\partial y}(\vec{a}) = 0$. Nonetheless, it can be proved that it is incorrect by using the definition of a partial derivative.

```
derX = Limit[1/h
    ((f[x, y] /. {x → a[[1]] + h, y → a[[2]]}) - (f[x, y] /. {x → a[[1]], y → a[[2]]})), h → 0]
```

```
Interval[{-1, 1}]
```

```
derY = Limit[1/h
    ((f[x, y] /. {x → a[[1]], y → a[[2]] + h}) - (f[x, y] /. {x → a[[1]], y → a[[2]]})), h → 0]
```

```
Interval[{-1, 1}]
```

Mathematica returns an interval because the limit does not exist. Thus, the directional derivative must be calculated directly from the definition.

```
dirDer = 1/Norm[s] * Limit[
    1/h ((f[x, y] /. {x → a[[1]] + h*s[[1]], y → a[[2]] + h*s[[2]]}) -
       (f[x, y] /. {x → a[[1]], y → a[[2]]})), h → 0]
```

```
0
```

```
Clear[f, s, a, derivativeX, derivativeY, derX, derY, dirDer];
```

Example 2.4.9: Find the gradient of $g = \left(x^3 + x + y\right) e^{-x^4 - y^2} - 2\left(-x^5 - \frac{1}{5}\, y^2 + x\right) e^{-x^2 - y^2}$. Plot g with several gradient vectors. Find directional derivatives at $\vec{a} = (-1.4,\, 0.4)$ along several directions.

```
g[x_, y_] = (x^3 + x + y) * e^(-x^4-y^2) - 2 * (-x^5 - 1/5 * y^2 + x) * e^(-x^2-y^2);

a = {-1.4, 0.4};

gradient = Simplify[Grad[g[x, y], {x, y}]];

gradientAtA = gradient /. {x → a[[1]], y → a[[2]]};

Plot3D[{g[x, y]}, {x, -2.5, 2.5}, {y, -2.5, 2.5}, ColorFunction → "Aquamarine",
 AxesLabel → {x, y, z}, MeshStyle → Darker[Blue], ImageSize → 250,
 ViewPoint → {-2, -2.5, 1}]
```

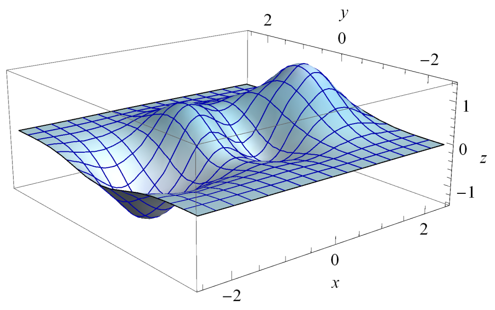


Figure 2.4.6: Function $g = \left(x^3 + x + y\right) e^{-x^4 - y^2} - 2\left(-x^5 - \frac{1}{5}\, y^2 + x\right) e^{-x^2 - y^2}$.

```
contours1 = ContourPlot[g[x, y], {x, -2.5, 2.5}, {y, -2.5, 2.5}, Contours → 14,
   ColorFunction → "Aquamarine"];

contours2 = ContourPlot[g[x, y], {x, -2.5, 0.5}, {y, -1.5, 1.5}, Contours → 11,
   ColorFunction → "Aquamarine"];

gradientField = VectorPlot[gradient, {x, -2.5, 2.5}, {y, -2.5, 2.5},
   VectorPoints → 15, VectorStyle → Arrowheads[0.02]];

Show[{contours1, gradientField}, ImageSize → 250]
```

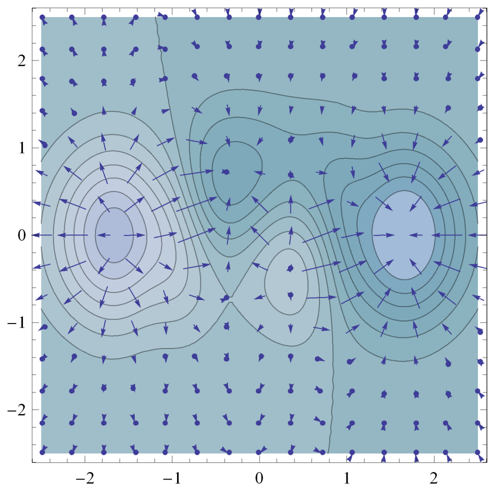


Figure 2.4.7: Level curves of function $g = \left(x^3 + x + y\right) e^{-x^4-y^2} - 2\left(-x^5 - \frac{1}{5}\, y^2 + x\right) e^{-x^2-y^2}$ together with the gradient field.

The gradient of g is perpendicular to level curves at all points. Its length depends on the growth of the function at the specific point.

```
scaleParameter = 1.5;

arrow = Graphics[Line[{{-0.5, 1/4}, {0, 0}, {-0.5, -1/4}}]];
```

As we demonstrated in the previous figure, the gradient is perpendicular to the level curves of the function. We can also show that the gradient at $\vec{a}$ is greater than all directional derivatives at the same point.

```
gradientEndPoint = {gradientAtA[[1]] / scaleParameter + a[[1]],
   gradientAtA[[1]] / scaleParameter + a[[2]]};

gradientGraphics =
  Graphics[{Red, Arrowheads[{{.05, 1, arrow}}], Arrow[{a, gradientEndPoint}]}];

                          n
angles = Table[2 * π * ——, {n, 12}];
                          12

directions = Table[{Cos[angles[[n]]], Sin[angles[[n]]]}, {n, 12}];

dirDerivatives = Table[directionalDerivative[g, directions[[n]], a], {n, 12}];

dirDerPoints1 =
  Table[{Cos[angles[[n]]] * dirDerivatives[[n]] / scaleParameter + a[[1]],
    Sin[angles[[n]]] * dirDerivatives[[n]] / scaleParameter + a[[2]]}, {n, 6}];
```

```
dirDerPoints2 =
  Table[{(-1) * Cos[angles[[n]]] * dirDerivatives[[n]] / scaleParameter + a[[1]],
     (-1) * Sin[angles[[n]]] * dirDerivatives[[n]] / scaleParameter + a[[2]]}, {n, 6}];

dirDerGraphics = {Graphics[{{Arrowheads[{{.05, 1, arrow}}],
       Table[Arrow[{a, dirDerPoints1[[n]]}], {n, 6}]}}],
    Graphics[{{Arrowheads[{{.05, 0, arrow}}],
       Table[Arrow[{dirDerPoints2[[n]], a}], {n, 6}]}}]};

Labeled[Show[{contours2, gradientGraphics, dirDerGraphics}, ImageSize → 250],
 PointLegend[{Red, Black}, {"gradient", "directional derivatives"},
  LegendMarkers → "⟶" ], Right]
```

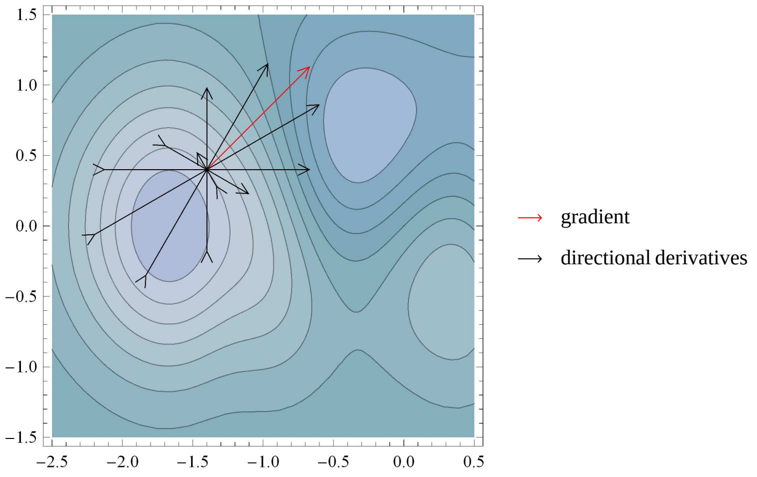


Figure 2.4.8: Level curves, gradient, and several directional derivatives of the function $g = \left(x^3 + x + y\right) e^{-x^4-y^2} - 2\left(-x^5 - \frac{1}{5} y^2 + x\right) e^{-x^2-y^2}$ at $\vec{a} = (-1.4, 0.4)$.

The graph shows directional derivatives of g at $\vec{a}$ along several directions. The length of each arrow corresponds to the numerical value of the directional derivative along the arrow. The greatest growth of the function is in the direction of the gradient.

```
Clear[g, a, gradient, gradientAtA, contours1, contours2, gradientField,
  scaleParameter, arrow, gradientEndPoint, gradientGraphics, angles,
  directions, dirDerivatives, dirDerPoints1, dirDerPoints2, dirDerGraphics];
```

2.5 Local extrema

Definition 2.5.7:

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is defined on $M \subset \mathbb{R}^n$ and $\vec{a} \in M$. Then f attains a ***local minimum*** *or a* ***local maximum*** *at $\vec{a}$ with respect to M if and only if there is $U(\vec{a})$ such that for all $\vec{x} \in M \cap U(\vec{a})$, $f(\vec{x}) \geq f(\vec{a})$ or $f\left(\vec{x}\right) \leq f(\vec{a})$, respectively.*

Similarly, f attains a ***sharp local minimum*** *or a* ***sharp local maximum*** *at $\vec{a}$ if and only if for all $\vec{x} \in M \cap U^*(\vec{a})$, $f(\vec{x}) > f(\vec{a})$ or $f(\vec{x}) < f(\vec{a})$, respectively.*

A point which is either a local minimum or a local maximum is called a ***local extremum****.*

Theorem 2.5.4 (The first derivative test):

Let $\vec{a}$ be a local extremum of f and all partial derivatives of f at $\vec{a}$ exist. Then

$$\frac{\partial f}{\partial x_i}(\vec{a}) = 0$$

*and $\vec{a}$ is called a **critical point**.*

Definition 2.5.8:

*Suppose $f : \mathbb{R}^n \to \mathbb{R}$ and all the second partial derivatives of f at $\vec{a}$ exist and are continuous. A **Hessian matrix** of f at $\vec{a}$ is defined as a square matrix of second partial derivatives*

$$H(f)_{ij}(\vec{a}) := \frac{\partial^2 f}{\partial x_i \partial x_j}(\vec{a}).$$

Theorem 2.5.5 (The second derivative test):

Let $H(f)(\vec{a})$ be the Hessian matrix of f at $\vec{a}$ and $\vec{a}$ be a critical point.
1. If $H(f)(\vec{a})$ is a positive definite matrix, then $\vec{a}$ is a local minimum point.
2. If $H(f)(\vec{a})$ is a negative definite matrix, then $\vec{a}$ is a local maximum point.
3. If $H(f)(\vec{a})$ is indefinite matrix, then $\vec{a}$ is a saddle point.

Remark 2.5.2:

1. $H(f)(\vec{a})$ is a positive definite matrix if and only if all its eigenvalues are positive.
2. $H(f)(\vec{a})$ is a negative definite matrix if and only if all its eigenvalues are negative.
3. $H(f)(\vec{a})$ is indefinite matrix if and only if some of its eigenvalues are positive and some negative.

Example 2.5.10: Find the local extrema and the saddle points of $f = \left(x^2 - y^2\right) e^{\frac{-x^2-y^2}{2}}$ with respect to $M = \left\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 < 9\right\}$.

```
f[x_, y_] = (x^2 - y^2) * e^((-x^2-y^2)/2);
```

The local extremum points can be roughly estimated from the figure.

```
Plot3D[{f[x, y]}, {x, -3, 3}, {y, -3, 3}, ColorFunction → "BlueGreenYellow",
  ImageSize → 250, AxesLabel → {x, y, z}, ViewPoint → {1, -1.5, 1.5}]
```

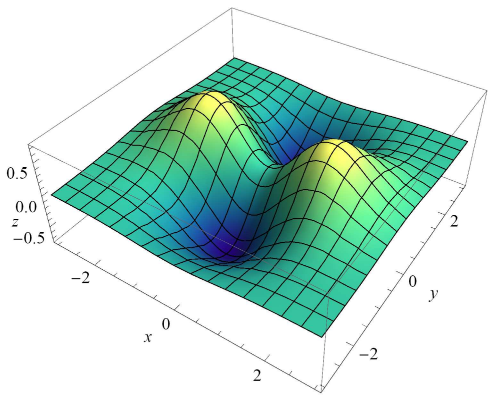


Figure 2.5.9: Function $f = \left(x^2 - y^2\right) e^{\frac{-x^2-y^2}{2}}$.

The first derivative test is used in order to find the critical points.

```
a = Solve[D[f[x, y], x] == 0 && D[f[x, y], y] == 0 && x^2 + y^2 < 9, {x, y}]
```

```
{{x → 0, y → 0}, {x → 0, y → -√2}, {x → 0, y → √2}, {x → -√2, y → 0}, {x → √2, y → 0}}
```

We found five critical points ($\vec{a}_1$, ..., $\vec{a}_5$). Lets us try to use the second derivative test. We calculate Hessian matrices and analyze whether $\vec{a}_i$, i ∈ {1, ..., 5} is a local extremum point or not.

```
hessianMatrix[f_, a_] := {{D[f[x, y], x, x] /. a, D[f[x, y], x, y] /. a},
  {D[f[x, y], x, y] /. a, D[f[x, y], y, y] /. a}}

secondDerivativeTest[mat_] :=
  Module[{eigenvs = Eigenvalues[mat]},
   If[And @@ Positive[eigenvs], Print["Local minimum."],
    If[And @@ Negative[eigenvs], Print["Local maximum."],
     If[And @@ NonPositive[eigenvs],
      Print["The second derivative test cannot be used."],
      If[And @@ NonNegative[eigenvs],
       Print["The second derivative test cannot be used."],
       Print["Saddle point."]]]]]];

secondDerivativeTest[hessianMatrix[f, a[[1]]]]
```

```
Saddle point.
```

```
secondDerivativeTest[hessianMatrix[f, a[[2]]]]
```

```
Local minimum.
```

```
secondDerivativeTest[hessianMatrix[f, a[[3]]]]
```

```
Local minimum.
```

```
secondDerivativeTest[hessianMatrix[f, a[[4]]]]
```

```
Local maximum.
```

```
secondDerivativeTest[hessianMatrix[f, a[[5]]]]
```

```
Local maximum.
```

Now the result can be demonstrated graphically using the *ContourPlot* command.

```
contours = ContourPlot[f[x, y], {x, -3, 3}, {y, -3, 3}, Contours → 10,
   ColorFunction → "BlueGreenYellow", PlotLegends → Automatic, AxesLabel → {x, y}];

criticalPoints =
  Graphics[Table[{Text[Style["a"_i, Black, 17], {(x /. a)[[i]], (y /. a)[[i]] + 0.25}],
     PointSize[Large], Black, Point[{(x /. a)[[i]], (y /. a)[[i]]}]}, {i, 5}]];
```

```
Labeled[Show[{contours , criticalPoints}, ImageSize → 250],
 PointLegend[{Black}, {"critical points"},
  LegendMarkers → {Graphics[Point[{0, 0}]]}], Right]
```

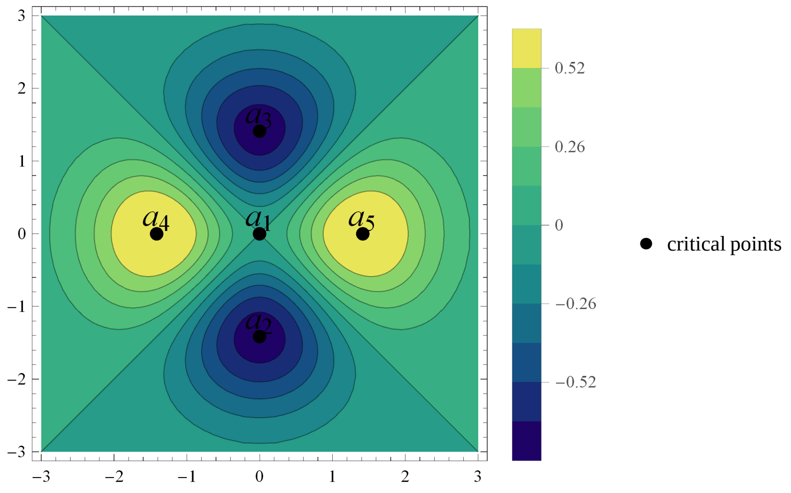


Figure 2.5.10: Level curves of $f = \left(x^2 - y^2\right) e^{\frac{-x^2-y^2}{2}}$ with the critical points.

There are also built-in functions *FindMinimum* and *FindMaximum* in Mathematica which search for local extrema numerically. However, these functions return only one local extremum point and sometimes are not able to find any solution.

```
Clear[f, a, contours, criticalPoints, hessianMatrix];
```

Example 2.5.11: Find the local extrema and the saddle points of $f = x^3 + y^2 + \frac{1}{2} z^2 - 3\,xz - 2\,y + 2\,z$ [11].

```
f[x_, y_, z_] = x^3 + y^2 + 1/2 * z^2 - 3 * x * z - 2 * y + 2 * z;
```

We will use the same procedure as in the example 2.5.10.

```
a = Solve[D[f[x, y, z], x] == 0 && D[f[x, y, z], y] == 0 && D[f[x, y, z], z] == 0 , {x, y, z}]

{{x → 1, y → 1, z → 1}, {x → 2, y → 1, z → 4}}
```

```
hessianMatrix[f_, a_] :=
 {{D[f[x, y, z], x, x] /. a, D[f[x, y, z], x, y] /. a, D[f[x, y, z], x, z] /. a},
  {D[f[x, y, z], x, y] /. a, D[f[x, y, z], y, y] /. a, D[f[x, y, z], y, z] /. a},
  {D[f[x, y, z], z, x] /. a, D[f[x, y, z], y, z] /. a, D[f[x, y, z], z, z] /. a}}
```

The function *secondDerivativeTest* from the previous example can be used again.

```
  secondDerivativeTest[hessianMatrix[f, a[[1]]]]

Saddle point.

  secondDerivativeTest[hessianMatrix[f, a[[2]]]]

Local minimum.

  Clear[f, a, hessianMatrix, secondDerivativeTest];
```

2.6 Constrained extrema

Definition 2.6.9:

*Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is defined on $M \subset \mathbb{R}^n$ and $\vec{a} \in M$. Then f attains a **local minimum** or a **local maximum** at $\vec{a}$ with respect to M if and only if there is $U(\vec{a})$ such that for all $\vec{x} \in M \bigcap U(\vec{a})$, $f(\vec{x}) \geq f(\vec{a})$ or $f\left(\vec{x}\right) \leq f(\vec{a})$, respectively.*

*Similarly, f attains a **sharp local minimum** or a **sharp local maximum** at $\vec{a}$ if and only if for all $\vec{x} \in M \bigcap U^*(\vec{a})$, $f(\vec{x}) > f(\vec{a})$ or $f(\vec{x}) < f(\vec{a})$, respectively.*

*A point which is either a local minimum or a local maximum is called a **local extremum**.*

Theorem 2.6.6 (The first derivative test):

Suppose $f, g_1, g_2, \ldots, g_m$ are functions of n variables and have continuous partial derivatives. Suppose that f has a local extremum with constraints $g_s = 0$, $s \in \{1, \ldots, m\}$, at the point $\vec{a}$ and $\nabla g_s(\vec{a}) \neq 0$ for all $s \in \{1, \ldots, m\}$. Then there are numbers $\lambda_1, \lambda_2, \ldots, \lambda_m$ such that

$$\nabla f(\vec{a}) = \lambda_1 \nabla g_1(\vec{a}) + \lambda_2 \nabla g_2(\vec{a}) + \ldots + \lambda_m \nabla g_m(\vec{a}). \qquad (2)$$

*The numbers $\lambda_1, \lambda_2, \ldots, \lambda_m$ are called **Lagrange multipliers**.*

Remark 2.6.3:

It is convenient to rewrite (2) as follows,

$$\nabla L(\vec{a}) = 0,$$

*where $L := f - \lambda_1 g_1 - \lambda_2 g_2 - \ldots - \lambda_m g_m$ is the so-called **Lagrange function**. The second derivative test, which is used for finding (non-constrained) local extrema, can now be also applied on L (instead of f itself) to find constrained extrema. However, in general it is no longer true that it cannot be decided about the type of extrema if the Hessian matrix is not positively (or negatively) definite. One can still restrict the Hessian matrix to the tangent subspace to all constraints and then investigate the resulting matrix that is effectively lower dimensional. If this matrix is positively or negatively definite, then $\vec{a}$ attains its constrained minimum or maximum, respectively [4].*

Example 2.6.12: Find the local extrema of $f = \frac{1}{x} + \frac{1}{y}$ subjected to the constraint $\frac{1}{x^2} + \frac{1}{y^2} - \frac{1}{4} = 0$ [11].

```
         1   1
f[x_, y_] = — + —;
         x   y

         1    1    1
g[x_, y_] = —— + —— - —;
         x²   y²   4
```

At first, we try the built-in functions *Minimize* and *Maximize*.

```
Minimize[{f[x, y], g[x, y] == 0}, {x, y}]
```

$$\left\{-\frac{1}{\sqrt{2}}, \left\{x \to -2\sqrt{2}, y \to -2\sqrt{2}\right\}\right\}$$

```
Maximize[{f[x, y], g[x, y] == 0}, {x, y}]
```

$\left\{\frac{1}{\sqrt{2}}, \left\{x \to 2\sqrt{2}, y \to 2\sqrt{2}\right\}\right\}$

It can be seen that these functions work just fine in our example. Nevertheless, let us try to solve the same problem mimicking a hand calculation. The first derivative test helps us to find the critical points together with corresponding values of the Lagrange multiplier.

```
a = Solve[D[f[x, y] - λ*g[x, y], x] == 0 && D[f[x, y] - λ*g[x, y], y] == 0 && g[x, y] == 0 ,
  {x, y, λ}]
```

$\left\{\left\{x \to -2\sqrt{2}, y \to -2\sqrt{2}, \lambda \to -\sqrt{2}\right\}, \left\{x \to 2\sqrt{2}, y \to 2\sqrt{2}, \lambda \to \sqrt{2}\right\}\right\}$

In the next step, the Hessian matrix of the Lagrange function is constructed at the critical points.

```
hessianMatrix[f_, a_] := {{D[f, x, x] /. a, D[f, x, y] /. a},
   {D[f, x, y] /. a, D[f, y, y] /. a}};
```

Finally, the function *secondDerivativeTest* is defined to identify the type of an extremum from the eigenvalues of the Hessian matrix.

```
secondDerivativeTest[mat_] := Module[{eigenvs = Eigenvalues[mat]},
   If[And @@ Positive[eigenvs], Print["Local minimum."],
    If[And @@ Negative[eigenvs], Print["Local maximum."],
     Print[
      "Our basic second derivative test is not sufficient. One has to
        reduce the Hessian matrix to the subspace that is tangent
        to the constraints and then use the second derivative
        test on the resulting matrix that is effectively
        lower-dimensional."]]]];

secondDerivativeTest[hessianMatrix[f[x, y] - λ*g[x, y], a[[1]]]]
```

```
Local minimum.
```

```
secondDerivativeTest[hessianMatrix[f[x, y] - λ*g[x, y], a[[2]]]]
```

```
Local maximum.
```

Next, it will be graphically demonstrated that at the points where local constrained extrema are attained, the gradient of *f* is just a scalar multiple of the gradient of *g*, i.e., (2) holds. To this purpose, denote the critical points as follows:

```
a1 = {x /. a[[1]], y /. a[[1]]};

a2 = {x /. a[[2]], y /. a[[2]]};
```

Moreover, introduce a function that returns the gradient vector of a function *f* at a point *startPoint* shifted to this *startPoint*.

```
arrows[f_, startPoint_, scaleParameter_, colour_] :=
 Module[{gradient, endPoint},
  gradient = Grad[f[x, y], {x, y}] /. {x → startPoint[[1]], y → startPoint[[2]]};
  endPoint = {gradient[[1]] * scaleParameter + startPoint[[1]],
    gradient[[2]] * scaleParameter + startPoint[[2]]};
  Graphics[{colour, Arrow[{startPoint, endPoint}]}]]
```

Finally, define all needed graphical objects and combine all of them into an illustrative figure.

```
contoursf = ContourPlot[f[x, y], {x, -5, 5}, {y, -5, 5},
    Contours → {-2, -1, -1/√2, -0.5, 0, 0.5, 1/√2, 1, 2}, ContourShading → False,
    ContourStyle → Blue, MaxRecursion → 3];

contoursg = ContourPlot[g[x, y] == 0, {x, -5, 5}, {y, -5, 5}, ContourShading → False,
    ContourStyle → Darker[Green]];

b = {-2 √2, 2 √2};

text =
  Graphics[
   {Table[{Text[Style["a"ᵢ, Black, 12], {(x /. a)[[i]] + 0.1, (y /. a)[[i]] - 0.5}],
      PointSize[Large], Point[{(x /. a)[[i]], (y /. a)[[i]]}]}, {i, 2}],
    Text[Style["b", Black, 12], {b[[1]], b[[2]] - 0.5}], PointSize[Large],
    Point[{b[[1]], b[[2]]}],
    Text[Style["∇f(a₁)", Blue, 10], {a1[[1]] - 1.2, a1[[2]] - 0.5}],
    Text[Style["∇g(a₁)", Darker[Green], 10], {a1[[1]], a1[[2]] + 0.7}],
    Text[Style["∇f(a₂)", Blue, 10], {a2[[1]] - 0.6, a2[[2]] - 1.5}],
    Text[Style["∇g(a₂)", Darker[Green], 10], {a2[[1]] - 0.65, a2[[2]] - 0.02}],
    Text[Style["∇f(b)", Blue, 10], {b[[1]] - 0.3, b[[2]] - 1}],
    Text[Style["∇g(b)", Darker[Green], 10], {b[[1]] + 1, b[[2]] - 0.1}],
   }];

Show[{contoursf , contoursg, text, arrows[f, a1, 10, Blue],
  arrows[g, a1, 10, Darker[Green]], arrows[f, a2, 10, Blue],
  arrows[g, a2, 10, Darker[Green]], arrows[f, b, 10, Blue],
  arrows[g, b, 10, Darker[Green]]}, ImageSize → 250]
```

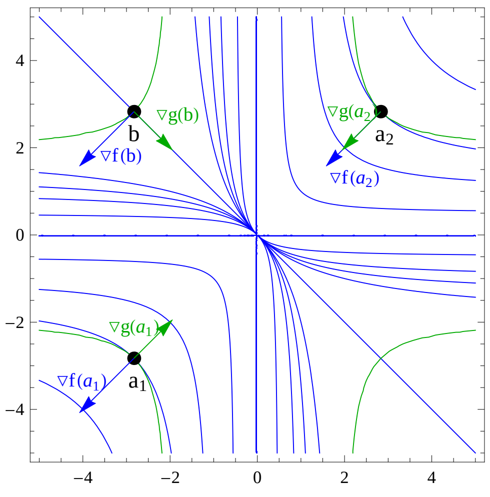


Figure 2.6.11: Level curves of the function $f = \frac{1}{x} + \frac{1}{y}$ and the constraint $\frac{1}{x^2} + \frac{1}{y^2} - \frac{1}{4} = 0$ together with the gradients of f and the constraint at the critical points.

Gradients of f and g at a_1 (local minimum point) are proportional. The same holds true at a_2 (local maximum point). On the other hand, this is not true at the point b which is not a critical point.

```
Clear[f, g, a, a1, a2, b, contoursf, contoursg, text];
```

Example 2.6.13: Find the local extrema of $f(x, y) = \left(x^2 - y^2\right) e^{\frac{-x^2-y^2}{2}}$ subjected to the constraint $(x-1)^2 + (y-1)^2 = \frac{1}{2}$.

```
f[x_, y_] = (x^2 - y^2) * e^((-x^2-y^2)/2);

g[x_, y_] = (x - 1)^2 + (y - 1)^2 - 1/2;
```

It can be seen that the constraint is the equation of the circle with the following parameters:

```
ax = 1;

ay = 1;

radius = Sqrt[1/2];
```

Again, let us start with built-in function.

```
Minimize[{f[x, y], g[x, y] == 0}, {x, y}]

Minimize[{e^(1/2 (-x^2-y^2)) (x^2 - y^2), -1/2 + (-1 + x)^2 + (-1 + y)^2 == 0}, {x, y}]
```

The function *f* is too complicated and so is its gradient.Therefore, it is not possible to solve the equation for the critical points symbolically. So try the numerical version of the above used functions.

```
NMinimize[{f[x, y], g[x, y] == 0}, {x, y}]

{-0.628413, {x → 0.357463, y → 1.2952}}

NMaximize[{f[x, y], g[x, y] == 0}, {x, y}]

{0.628413, {x → 1.29521, y → 0.357463}}
```

This works pretty well, both for the minimum and the maximum. Nevertheless, we could also try to use the first derivative test.

```
a = NSolve[g[x, y] == 0 && D[f[x, y] - λ*g[x, y], x] == 0 && D[f[x, y] - λ*g[x, y], y] == 0 ,
   {x, y, λ}, Reals]

{{x → 0.357474, y → 1.29523, λ → -0.400399}, {x → 1.29523, y → 0.357474, λ → 0.400399}}
```

Again, we use the functions *hessianMatrix* and *secondDerivativeTest* from the previous example in order to find the Hessian matrix of the Lagrange function at the critical points and identify the type of an extremum.

```
  secondDerivativeTest[hessianMatrix[f[x, y] - λ*g[x, y], a[[1]]]]

Local minimum.

  secondDerivativeTest[hessianMatrix[f[x, y] - λ*g[x, y], a[[1]]]]

Local minimum.

  Clear[f, g, ax, ay, radius, a, arrows, hessianMatrix, secondDerivativeTest];
```

Example 2.6.14: The Baraboo, Wisconsin, plant of International Widget Co. uses aluminum, iron, and magnesium to produce high-quality widgets. The quantity of widgets which may be produced using x tons of aluminum, y tons of iron, and z tons of magnesium is $Q(x, y, z) = xyz$. The cost of raw materials is: aluminum, \$6 per ton; iron, \$4 per ton; and magnesium, \$8 per ton. How many tons of aluminum, iron, and magnesium should be used to manufacture 1000 widgets at the lowest possible cost? [4]

Considering the cost of raw materials, the function *f* can be defined.

```
f[x_, y_, z_] = 6*x + 4*y + 8*z;

q[x_, y_, z_] = x*y*z - 1000;
```

We want to minimize the cost, i.e., the function *f*. *Q* is the constraint. The Lagrange function can be defined as:

```
lagrange[x_, y_, z_] = f[x, y, z] - λ*q[x, y, z];
```

The first derivative test is used to find the solution. The fourth equation is given by the constraint.

```
a = Solve[D[lagrange[x, y, z], x] == 0 && D[lagrange[x, y, z], y] == 0
    && D[lagrange[x, y, z], z] == 0 && q[x, y, z] == 0
   , {x, y, z, λ}, Reals] // N

{{x → 9.615, y → 14.4225, z → 7.21125, λ → 0.05769}}

Clear[f, q, lagrange, a];
```

2.7 Double integral

In this section, we will consider only double integrals. The transition to higher dimensions can be made in a similar way.

Definition 2.7.10:
A finite sequence $a = x_0 < x_1 < ... < x_n = b$, where $a < b$, is called a ***partition of the interval $\langle a, b\rangle$****.*

Definition 2.7.11:
Let $\{x_0, x_1, ..., x_n\}$ be a partition of the interval $\langle a, b\rangle$ and $\{y_0, y_1, ..., y_m\}$ be a partition of the interval $\langle c, d\rangle$. Then $\left\{S_{i,j} := \langle x_{i-1}, x_i\rangle \times \langle y_{i-1}, y_i\rangle\right\}_{i=1, j=1}^{n,m}$ is called a ***partition of the interval $\langle a, b\rangle \times \langle c, d\rangle$****. Additionally, measure of the interval $I = \langle a, b\rangle \times \langle c, d\rangle$ is defined as $\mu(I) = (b-a)(d-c)$.*

Definition 2.7.12:
Suppose $f : \mathbb{R}^2 \to \mathbb{R}$ is bounded on the interval $I = \langle a, b\rangle \times \langle c, d\rangle$ and D is a partition of this interval. Let $v_{i,j} = inf_{x\in S_{i,j}} f(x)$ and $V_{i,j} = sup_{x\in S_{i,j}} f(x)$. Then $L(D, f) := \sum_{i=1, j=1}^{n,m} v_{i,j}\, \mu\left(S_{i,j}\right)$ is called a ***lower sum*** *for f on D. Similarly, $U(D, f) := \sum_{i=1, j=1}^{n,m} V_{i,j}\, \mu\left(S_{i,j}\right)$ is called an* ***upper sum*** *for f on D.*

Definition 2.7.13:
Suppose $f : \mathbb{R}^2 \to \mathbb{R}$ is bounded on the interval $I = \langle a, b\rangle \times \langle c, d\rangle$, $M \subset \mathbb{R}^2$, and D is a partition of this interval. Then $sup_M\, L(D, f)$ or $inf_M\, U(D, f)$ is called a ***lower Riemann integral*** *or an* ***upper Riemann integral****, respectively. If $sup_M\, L(D, f) = inf_M\, U(D, f)$, the function f is said to be* ***Riemann integrable*** *over I and $\int_I f = sup_M\, L(D, f)$.*

Definition 2.7.14:
Let $I = \langle a, b\rangle \times \langle c, d\rangle$ and $D = \left\{S_{ij}\right\}$ be a partition of this interval. Then a ***norm*** *of D is defined as $\|D\| := max\, \mu\left(S_{ij}\right)$, where $0 \le i \le n$ and $0 \le j \le m$.*

Theorem 2.7.7:
Suppose $f : \mathbb{R}^2 \to \mathbb{R}$ is continuous on the interval $I = \langle a, b\rangle \times \langle c, d\rangle$ Then $\int_I f$ exists and if $\{D_n\}_{n=1}^{\infty}$ is a

partition sequence such that $lim_{n\to\infty} \|D_n\| = 0$, then

$$\int_I f = lim_{n\to\infty} L(D_n, f) = lim_{n\to\infty} U(D_n, f).$$

Theorem 2.7.8 (The Fubini theorem):
Suppose $f : \mathbb{R}^2 \to \mathbb{R}$ is continuous on the interval $I = \langle a, b\rangle \times \langle c, d\rangle$ Then

$$\int_I f(x, y)\, dx\, dy = \int_a^b \left(\int_c^d f(x, y)\, dy\right) dx = \int_c^d \left(\int_a^b f(x, y)\, dx\right) dy.$$

Definition 2.7.15:
*Let $\vec{F} : \mathbb{R}^n \to \mathbb{R}^s$ and $\vec{a}$ be a point of its domain such that for all $i \in \{1, ..., s\}$ and $j \in \{1, ..., n\}$ there exists $\frac{\partial F_i}{\partial x_j}(\vec{a})$. Then **Jacobian matrix** is, by definition,*

$$\frac{\partial \vec{F}}{\partial \vec{x}}(\vec{a}) \equiv \frac{D(F_1, ..., F_s)}{D(x_1, ..., x_n)} := \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \\ \frac{\partial F_s}{\partial x_1} & & \frac{\partial F_s}{\partial x_n} \end{pmatrix} (\vec{a}).$$

Definition 2.7.16:
*A function $\vec{F} : \mathbb{R}^n \to \mathbb{R}^n$ is called **regular** over an open set $M \in \mathbb{R}^n$ if and only if it has a continuous Jacobian matrix and det $\frac{D(\phi_1,...,\phi_s)}{D(\xi_1,...,\xi_n)}(\vec{x}) \neq 0$ for all $\vec{x} \in M$.*

Theorem 2.7.9 (The substitution theorem):
Suppose $\vec{\phi} : \mathbb{R}^n \to \mathbb{R}^n$, $\vec{\phi} : (\xi_1, ..., \xi_n) \to (x_1, ..., x_n)$ is an injective and regular vector function from $P \subset \mathbb{R}^n$ to $Q \subset \mathbb{R}^n$. Then for arbitrary set $M \subset Q$

$$\int_M f(x_1, ..., x_n)\, dx_1, ..., dx_n = \int_{\phi^{-1}(M)} f\left(\vec{\phi}(\xi_1, ..., \xi_n)\right) \left| det \frac{D(\phi_1, ..., \phi_s)}{D(\xi_1, ..., \xi_n)} \right| d\xi_1, ..., d\xi_n.$$

Note: In the substitution theorem, the absolute value of Jacobian determinant appears. Substitutions to the polar, spherical, and cylindrical coordinates systems are used very often. Therefore, here are Jacobian determinants of them:

```
JacobianDeterminant[f_List, vars_List] :=
 If[Equal @@ Map[Dimensions, {f, vars}], Simplify[Det[Outer[D, f, vars]]],
  "The dimensions of input parameters are not the same."]
```

```
polar = JacobianDeterminant[{ρ*Cos[ϕ], ρ*Sin[ϕ]}, {ρ, ϕ}]
```

```
ρ
```

```
spherical = JacobianDeterminant[{ρ*Cos[θ]*Cos[ϕ], ρ*Cos[θ]*Sin[ϕ], ρ*Sin[θ]},
  {ρ, θ, ϕ}]
```

$-\rho^2$ Cos[θ]

As θ runs from $-\frac{\pi}{2}$to $\frac{\pi}{2}$, $\cos(\theta)$ is non-negative. Thus, the absolute value of the determinant is ρ^2 $\cos(\theta)$ for every ρ and θ.

```
cylindrical = JacobianDeterminant[{ρ*Cos[ϕ], ρ*Sin[ϕ], h}, {ρ, ϕ, h}]
```

```
ρ
```

```
Clear[ϕ, θ, h, polar, spherical, cylindrical];
```

Example 2.7.15: Find the surface area bounded by the curves $y = 2\,x^2$ and $y = 1 - x$.

```
f[x_] = 2*x^2;
g[x_] = 1 - x;
```

To find the limits of integration, we need to locate intersections of the plots of *f* and *g*.

```
sol = Solve[f[x] == g[x], x]
```

$\{\{x \to -1\}, \{x \to \frac{1}{2}\}\}$

```
Show[
 {Graphics[{Style[Text[x /. sol[[1]], {(x /. sol[[1]]) + 0.25, -0.25}], 12],
    Style[Text[x /. sol[[2]], {(x /. sol[[2]]) + 0.25, -0.25}], 12]}],
  Plot[{f[x] /. a→1, g[x] /. a→1}, {x, -1.1, 1.1}, PlotStyle→{Blue, Green},
   PlotLegends→{"f[x]", "g[x]"}]}, Axes→True, Ticks→None,
 GridLinesStyle→Dashed, ImageSize→200,
 GridLines→{{x /. sol[[1]], x /. sol[[2]]}, {}}]
```

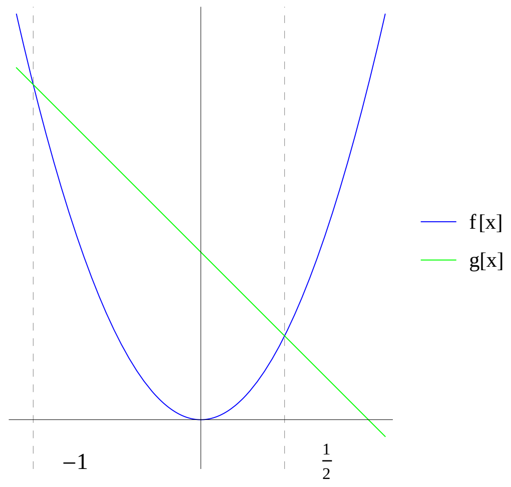


Figure 2.7.12: Functions $f = 2\,x^2$ and $g = 1 - x$.

The limits of integration are -1 and $\frac{1}{2}$ (with respect to *x*) and the functions *f* and *g* (with respect to *y*). Since the surface area is searched, the functions which is to be integrated is just 1.

```
Integrate[Integrate[1, {y, 2*x^2, 1 - x}], {x, -1, 1/2}]
```

$\frac{9}{8}$

This example could be also solved without any knowledge of double integrals. We can calculate the integral of both g and f from -1 to $\frac{1}{2}$, and the result would be the difference between them.

```
Clear[f, g, sol];
```

Example 2.7.16: Calculate the integral $\iint_M \frac{1}{\sqrt{2x-x^2}}\, dx\, dy$, where M is an area bounded by the curves $x = 0,\ y^2 = -x+2$.

```
f[x_] = 1/Sqrt[2*x - x^2];
```

Let us have a look at M to find the limits of integration:

```
Show[
 {Graphics[{Style[Text["2", {2.5, -0.5}], 12], Style[Text["0", {0.5, -0.5}], 12]}],
  ContourPlot[{x == 0, y^2 == -x + 2}, {x, -3, 3}, {y, -3, 3},
   ContourStyle → {Blue, Green}, PlotLegends → {"x = 0", "y^2 = - x + 2"}]},
 Axes → {True, False}, Frame → False, Ticks → None, ImageSize → 200,
 GridLinesStyle → Dashed, GridLines → {{2}, {}}]
```

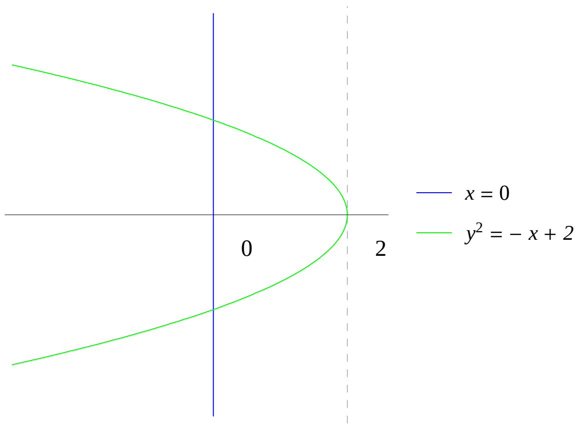


Figure 2.7.13: Curves $x = 0$ and $y^2 = -x+2$.

Considering the x-axis symmetry, only the area above the x-axis can be integrated and then multiplied by 2.

```
2*Integrate[Integrate[f[x], {y, 0, Sqrt[-x + 2]}], {x, 0, 2}]
```

$4\sqrt{2}$

```
Clear[f];
```

Example 2.7.17: Find the integral $\iint_M \ln(x^2+y^2)\, dx\, dy$, where M is an area in the first quadrant bounded by the curves $x^2+y^2 = 1$ and $x^2+y^2 = 4$ [11].

```
f[x_, y_] = Log[x^2 + y^2];
```

```
Labeled[Show[{Graphics[{Style[Text["0", {0.25, -0.25}], 12],
     Style[Text["1", {1.25, -0.25}], 12], Style[Text["2", {2.25, -0.25}], 12]}],
   Graphics[{Green, Disk[{0, 0}, 2, {0, Pi/2}], White, Disk[{0, 0}, 1, {0, Pi/2}],
     Blue, Circle[{0, 0}], Circle[{0, 0}, 2]}]},
  Axes → True, Frame → False, Ticks → None, ImageSize → 200],
 PointLegend[{Green}, {"integrated area"}, LegendMarkers → "■"], Right]
```

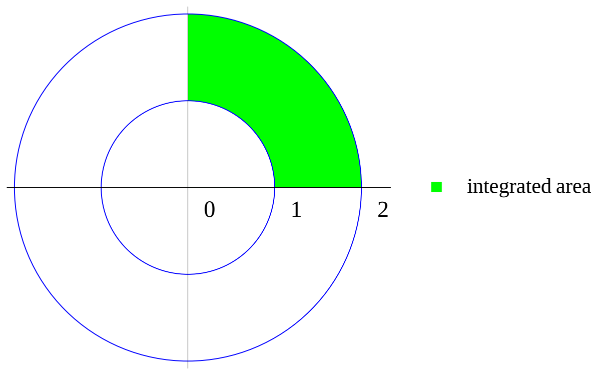


Figure 2.7.14: Curves $x^2 + y^2 = 1$ and $x^2 + y^2 = 4$.

Due to the polar symmetry, it is convenient to use the polar coordinates via the substitution $x = \rho\cos(\phi)$ and $x = \rho\sin(\phi)$.

```
fPolar[ρ_, ϕ_] = TransformedField["Cartesian" → "Polar", f[x, y], {x, y} → {ρ, ϕ}] //
  Simplify
```

$\text{Log}[\rho^2]$

The integration is much easier now. As it can be deduced from the figure, ρ runs from 1 to 2 and ϕ from 0 to 2π. The Jacobian determinant for polar substitution is ρ.

```
Integrate[
  Integrate[fPolar[ρ, ϕ] * JacobianDeterminant[{ρ * Cos[ϕ], ρ * Sin[ϕ]}, {ρ, ϕ}],
   {ρ, 1, 2}], {ϕ, 0, π/2}] // Simplify
```

$\pi\left(-\frac{3}{4} + \text{Log}[4]\right)$

```
Clear[f, fPolar];
```

Example 2.7.18: Find the integral $\iiint_B \left(1 + x^2 + y^2 + z^2\right)^{-\frac{1}{2}} dx\,dy\,dz$, where B is an sphere with radius 1 and centre at the origin [11].

```
g[x_, y_, z_] = (1 + x^2 + y^2 + z^2)^(-1/2);
```

The integration domain is a sphere so we use spherical coordinates system.

```
gSpherical[ρ_, θ_, ϕ_] =
 TransformedField["Cartesian" → "Spherical", g[x, y, z], {x, y, z} → {ρ, θ, ϕ}] //
  Simplify
```

$\frac{1}{\sqrt{1 + \rho^2}}$

The absolute value of Jacobian determinant is, in this case, $\rho^2 \cos(\theta)$. Since the integration domain is a sphere of radius 1, ρ goes from 0 to 1, ϕ from 0 to 2π, and θ from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$.

```
Integrate[Integrate[Integrate[gSpherical[ρ, θ, ϕ] *
    Abs[JacobianDeterminant[{ρ * Cos[θ] * Cos[ϕ], ρ * Cos[θ] * Sin[ϕ], ρ * Sin[θ]},
      {ρ, θ, ϕ}]], {ρ, 0, 1}], {ϕ, 0, 2 * π}], {θ, -π/2, π/2}]
```

```
2 π (√2 - ArcSinh[1])
```

```
Clear[g, gSpherical];
```

Example 2.7.19: Find the integral $\iiint_M z e^{x^2+y^2}\, dx\, dy\, dz$, where $M = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 \le 4,\ 2 \le z \le 3\}$ [11].

```
f[x_, y_, z_] = z * e^(x^2+y^2);
```

We convert the function *f* to the cylindrical coordinates.

```
fCylindrical[ρ_, ϕ_, h_] =
 TransformedField["Cartesian" → "Cylindrical", f[x, y, z], {x, y, z} → {ρ, ϕ, h}] //
  Simplify
```

```
e^(ρ^2) h
```

The integration domain *M* is a cylinder with radius 2. Thus, ρ runs from 0 to 2. Its height *h* goes from 2 to 3 and ϕ must run around full circle. The Jacobian determinant is ρ.

```
Integrate[Integrate[Integrate[fCylindrical[ρ, ϕ, h] *
     JacobianDeterminant[{ρ * Cos[ϕ], ρ * Sin[ϕ], h}, {ρ, ϕ, h}], {ρ, 0, 2}], {ϕ, 0, 2 * π}]
  {h, 2, 3}] // Simplify
```

$$\frac{5}{2}\left(-1 + e^4\right)\pi$$

```
Clear[f, fCylindrical, JacobianDeterminant];
```

3 Interactive visualizations

3.1 Solution to the differential equation

The demonstration depicts the plot of the solution of a chosen differential equation passing through a point $\vec{a}$ that is determined by dragging the locator. The equations are of the type $y' = f(x, y)$, where $f : \mathbb{R}^2 \to \mathbb{R}$ is a continuous function. The vector field in the background represents the direction of the tangent line to the plot of the solution at each point. This field is given just by the right-hand side of the equation. The particular solution is fixed by the initial condition.

```
Manipulate[
 equation = Switch[function,
   "y", y'[x] == y[x],
   "cos(x)", y'[x] == Cos[x],
   "3x^2ⅇ^x^2+2xy", y'[x] == 3*x^2*ⅇ^x^2 + 2 x*y[x]
  ];
 fun = Switch[function,
   "y", y,
   "cos(x)", Cos[x],
   "3x^2ⅇ^x^2+2xy", 3*x^2*ⅇ^x^2 + 2 x*y
  ];
 sol = DSolve[{equation, y[a[[1]]] == a[[2]]}, y[x], x];
 Show[{Plot[y[x] /. sol[[1]], {x, -5, 5}, PlotRange → 5, PlotStyle → Black],
   VectorPlot[{1, fun}/Sqrt[1 + fun^2], {x, -5, 5}, {y, -5, 5}, Axes → True, Frame → None,
    VectorPoints → 15, VectorScale → Small,
    VectorStyle → {Opacity[0.3], Arrowheads[0.02], Thickness[0.001]}]},
  ImageSize → 220, PlotRange → 5, AspectRatio → 1],
 Row[{Control[{{a, {1, 1}}, Locator}],
   Row[{Labeled[Control[{{function, "y", ""}, {"y", "cos(x)", "3x^2ⅇ^x^2+2xy"}}],
      {"y' ="}, {Left}], Spacer[60],
     Pane[Row[{" a⃗ = (", Dynamic[Round[a[[1]], 0.1]], ", ",
        Dynamic[Round[a[[2]], 0.1]], ")"}], ImageSize → 80]}]}],
 ControlType → PopupMenu]
```

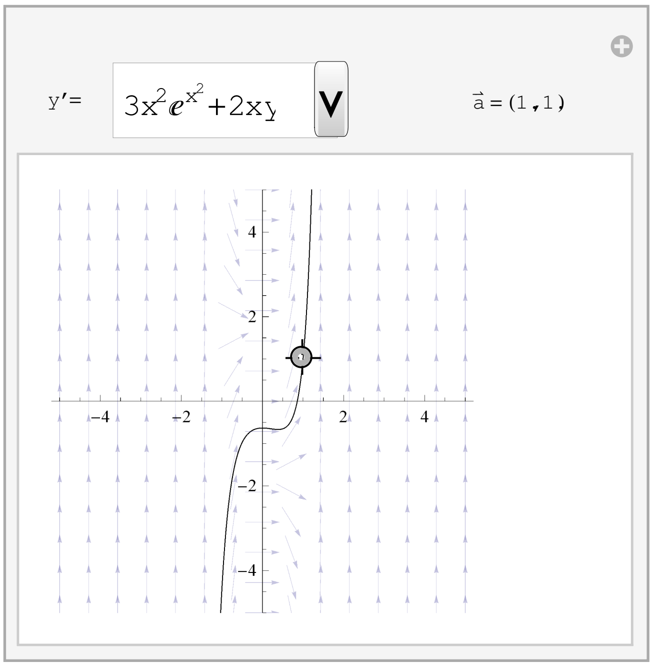


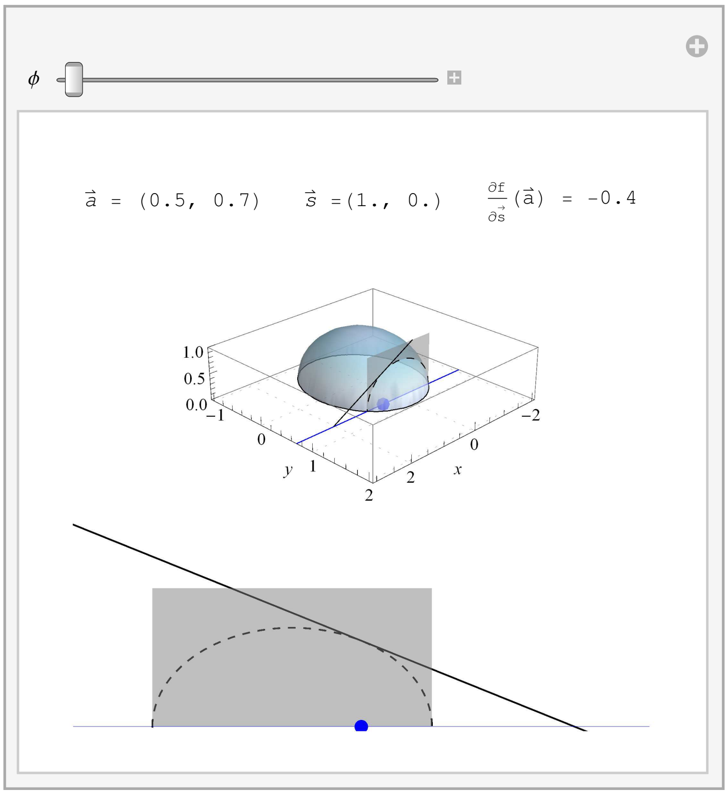
Figure 3.1.15: Particular solution to chosen differential equation fixed by selected initial condition. The vector field represents the direction of the tangent line to the plot at each point.

```
Clear[equation, fun, sol, a, function, y];
```

3.2 Directional derivative

This applet demonstrates geometric interpretation of the directional derivative of the function f along a given vector $\vec{s}$ at the point $\vec{a}$. The upper inset in the figure shows the graph of f; the point $\vec{a}$ is depicted in blue, and the blue line lies in the direction of $\vec{s}$. The intersection of the graph with the plane, which is perpendicular to the xy-plane and contains the blue line, is displayed as a dashed line. The lower inset shows a two-dimensional cut together with the tangent to this intersection. The value of the directional derivative of f along $\vec{s}$ at $\vec{a}$ is just the tangent function of the angle between the tangent and the blue line. As one varies $\vec{s}$ (by choosing the value of ϕ, which denotes the angle between $\vec{s}$ and the x-axis), the directional derivative changes correspondingly.

```
Manipulate[
 Module[{f, a, a3D, directionalDerivative, plotf, s, horizontalLine,
   tangent, intersections, intersection1, intersection2, cut, plane, point,
   tangent2, cut2, plane2, point2},
  f[x_, y_] = Sqrt[(1 - 1/2 * x^2 - y^2)];
  a = {0.5, 0.7};
  a3D = Append[a, f @@ a];
  s = {Cos[ϕ], Sin[ϕ]};
  directionalDerivative[f_, s_, a_] :=
   (Grad[f[x, y], {x, y}] /. {x → a[[1]], y → a[[2]]}).s;
  plotf = Plot3D[f[x, y], {x, -2, 3}, {y, -1, 2},
    PlotRange → {0, 1}, Mesh → None, AxesLabel → Automatic,
    FaceGrids → {{0, 0, -1}}, FaceGridsStyle → Directive[Dotted],
    BoxRatios → {4, 4, 1}, ColorFunction → "Aquamarine", PlotStyle → Opacity[0.7]];
  horizontalLine = ParametricPlot3D[{a3D[[1]] + s[[1]] * t, a3D[[2]] + s[[2]] * t, 0},
     {t, -10, 10}, PlotStyle → {Thickness[0.003], Blue}];
  tangent = ParametricPlot3D[{a3D[[1]] + s[[1]] * t, a3D[[2]] + s[[2]] * t,
     a3D[[3]] + directionalDerivative[f, s, a3D] * t},
     {t, -10, 10}, PlotStyle → {Thickness[0.003], Black}];
  intersections = t /. NSolve[f[a3D[[1]] + s[[1]] * t, a3D[[2]] + s[[2]] * t] == 0, t];
  intersection1 = {a3D[[1]] + s[[1]] * intersections[[1]],
    a3D[[2]] + s[[2]] * intersections[[1]]};
  intersection2 = {a3D[[1]] + s[[1]] * intersections[[2]],
    a3D[[2]] + s[[2]] * intersections[[2]]};
  cut = ParametricPlot3D[{a3D[[1]] + s[[1]] * t, a3D[[2]] + s[[2]] * t,
     f[a3D[[1]] + s[[1]] * t, a3D[[2]] + s[[2]] * t]},
     {t, intersections[[1]], intersections[[2]]},
    PlotStyle → {Thickness[0.003], Dashed}];
  plane = Graphics3D[{FaceForm[GrayLevel[0.5]], EdgeForm[], Opacity[0.5],
     Polygon[{{intersection1[[1]], intersection1[[2]], 0},
       {intersection1[[1]], intersection1[[2]], 1},
       {intersection2[[1]], intersection2[[2]], 1},
       {intersection2[[1]], intersection2[[2]], 0}}]}];
  point = Graphics3D[{PointSize → Large, Blue, Point[Append[a, 0]]}];
  tangent2 = ParametricPlot[{t, a3D[[3]] + directionalDerivative[f, s, a3D] * t},
     {t, -10, 10}, PlotStyle → {Thickness[0.003], Black}];
  cut2 = ParametricPlot[{t, f[a3D[[1]] + s[[1]] * t, a3D[[2]] + s[[2]] * t]},
     {t, intersections[[1]], intersections[[2]]},
    PlotStyle → {Thickness[0.003], Black, Dashed}];
  plane2 = Graphics[{FaceForm[GrayLevel[0.5]], EdgeForm[], Opacity[0.5],
     Polygon[{{intersections[[1]], 0}, {intersections[[1]], 1},
       {intersections[[2]], 1}, {intersections[[2]], 0}}]}];
  point2 = Graphics[{PointSize → Large, Blue, Point[{0, 0}]}];
  Labeled[
   GraphicsGrid[{{Show[{plotf, tangent, horizontalLine, cut, plane, point},
       ViewPoint → {3, 3, 2}]},
     {Show[{tangent2, cut2, plane2, point2}, PlotRange → {{-2, 2}, {0, 1.5}},
       AxesStyle → Blue, ImageSize → 300, Axes → {True, False}, Ticks → False]}},
    Spacings → {0, 5}],
   Pane[Row[{Pane[Row[{"a⃗ = (", a3D[[1]], ", ", a3D[[2]], ")"}]],
      Spacer[20],
      Pane[Row[{"s⃗ =(", N[Round[s[[1]], 0.1]], ", ", N[Round[s[[2]], 0.1]], ")"}]],
      Spacer[20],
      Pane[Row[{"∂f/∂s⃗(a⃗) = ", N[Round[directionalDerivative[f, s, a3D], 0.01]]}]]}],
    ImageMargins → 20], Top]], {ϕ, 0, 2 * π}, ContinuousAction → False]
```



ϕ
$\vec{a}$ = (0.5, 0.7) $\vec{s}$ =(1., 0.) $\frac{\partial f}{\partial \vec{s}}(\vec{a})$ = -0.4
1.0
0.5
0.0
-1
0
y
1
2
-2
0
2
x

Figure 3.2.16: Geometric interpretation of directional derivative.

3.3 Constrained extrema

The animation illustrates how the gradient of the function $f(x, y) = (x^2 - y^2)\, e^{\frac{-x^2-y^2}{2}}$ and the gradient of the constraint $(x-1)^2 + (y-1)^2 - \frac{1}{2} = 0$ look at the points of the constraint. The function f has two local extremum points subjected to the constraint g. Only at these points, the gradients of f and g are proportional.
The gradient of f is displayed as a blue arrow, whereas the gradient of g is displayed as a red arrow. The critical points are represented by pink points. The length and the direction of the arrows correspond to the length and the direction of the gradient.

```
f[x_, y_] = (x² - y²) * e^((-x²-y²)/2);

g[x_, y_] = (x - 1)² + (y - 1)² - 1/2;

ax = 1; ay = 1; radius = √0.5 ;

a = NSolve[g[x, y] == 0 && D[f[x, y] - λ*g[x, y], x] == 0 && D[f[x, y] - λ*g[x, y], y] == 0 ,
   {x, y, λ}, Reals]

{{x → 0.357474, y → 1.29523, λ → -0.400399}, {x → 1.29523, y → 0.357474, λ → 0.400399}}

a1 = {x /. a[[1]], y /. a[[1]]};

a2 = {x /. a[[2]], y /. a[[2]]};
```

```
arrows[f_, startPoint_, colour_] :=
 Module[{gradient, endPoint, scaleParameter},
  scaleParameter = 1;
  gradient = Grad[f[x, y], {x, y}] /. {x → startPoint[[1]], y → startPoint[[2]]};
  endPoint = {gradient[[1]] * scaleParameter + startPoint[[1]],
    gradient[[2]] * scaleParameter + startPoint[[2]]};
  Graphics[{colour, Arrow[{startPoint, endPoint}]}]]

backGround = {ContourPlot[f[x, y], {x, -2, 4}, {y, -2, 4},
    Contours → 10, ColorFunction → "BlueGreenYellow",
    MaxRecursion → 5, ImageSize → 250, AxesLabel → {x, y}],
   Graphics[Circle[{ax, ay}, radius]],
   Graphics[{PointSize[Large], Pink, Point[{a1[[1]], a1[[2]]}]}],
   Graphics[{PointSize[Large], Pink, Point[{a2[[1]], a2[[2]]}]}]};

frames = Table[Show[Append[backGround,
     {arrows[f, {Cos[angle] * radius + ax, Sin[angle] * radius + ay}, Blue],
      arrows[g, {Cos[angle] * radius + ax, Sin[angle] * radius + ay}, Red]}],
    PlotLabel → Pane[Row[{"f[", N[Round[Cos[angle] * radius + ax, 0.01]], ", ",
        N[Round[Sin[angle] * radius + ay, 0.01]], "] = ",
        N[Round[f[Cos[angle] * radius + ax, Sin[angle] * radius + ay], 0.01]]}]]],
   {angle, 0, 2 * π, π/50}];

Labeled[ListAnimate[Rasterize[#, "Image"] & /@ frames, AnimationRunning → False],
 PointLegend[{Pink, Blue, Red},
  {"critical points", "gradient of f", "gradient of the constaint"},
  LegendMarkers → {Graphics[Point[{0, 0}]], "⟶" , "⟶" }], Right]
```

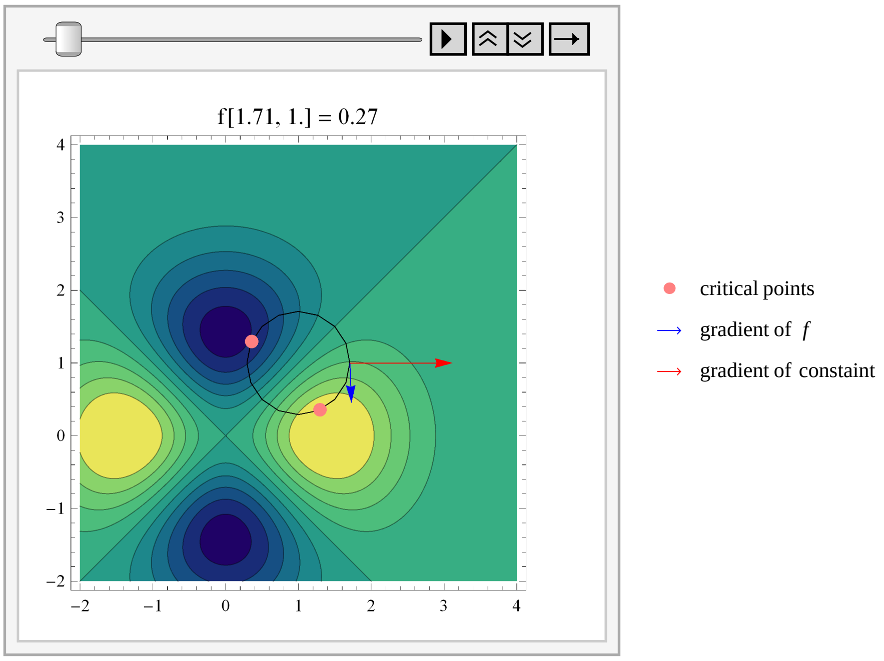


Figure 3.3.17: Gradient of the function $\left(x^2 - y^2\right) e^{\frac{-x^2-y^2}{2}}$ and the constraint $(x-1)^2 + (y-1)^2 - \frac{1}{2} = 0$ at the points of the constraint.

```
Clear[f, g, ax, ay, radius, a, a1, a2, arrows, backGround, frames];
```

3.4 Useful alternative coordinate systems

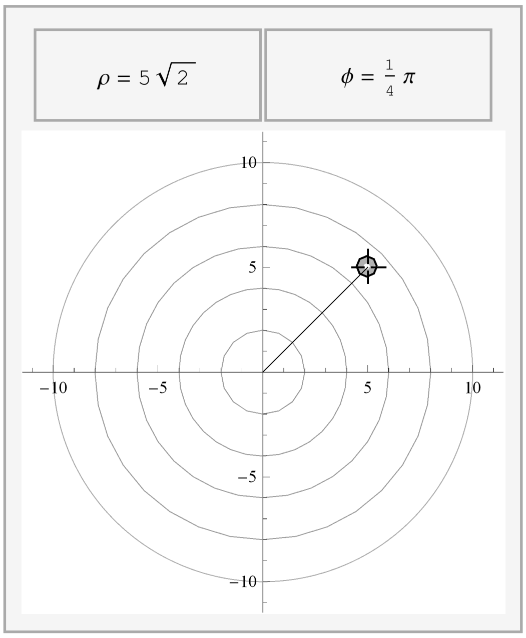
Transformation from cartesian to one of the following coordinate systems can help us with calculating integrals when integrated function and/or integration domain have radial, spherical or cylindrical symmetry.

3.4.1 Polar coordinate system

This is a two-dimensional coordinate system in which each point is determined by the distance from the origin and the angle from the polar axis. The relationship between these coordinate systems is given by

$$x = \rho\, cos(\phi),$$
$$y = \rho\, sin(\phi).$$

```
Panel[DynamicModule[{p = {5, 5}}, Column[{Row[{Panel[Row[{"ρ = ",
        Dynamic[EuclideanDistance[{0, 0}, p]]}], ImageSize → {120, 50},
      Alignment → {Center, Center}], Panel[Row[{"ϕ = ", Dynamic[If[p[[2]] > 0,
         VectorAngle[{1, 0}, p]   2 * π - VectorAngle[{1, 0}, p]
         ---------------------- , -------------------------------]], " π"}],
                   π                             π
      ImageSize → {120, 50}, Alignment → {Center, Center}]}],
   Show[PolarPlot[{10}, {t, 0, 2 * π}, PlotStyle → Lighter[Gray],
     PolarGridLines → {{0, Pi / 2, Pi, 3 Pi / 2}, {2, 4, 6, 8}}], Graphics[{Locator[
       Dynamic[p]], Dynamic[Line[{{0, 0}, p}]], Opacity[0.2]}, Axes → True],
    PlotRange → {{-11, 11}, {-11, 11}}, ImageSize → 250]}, Alignment → Center]]]
```



$\rho = 5\sqrt{2}$ $\phi = \frac{1}{4}\pi$

Figure 3.4.18: Polar coordinate system.

3.4.2 Spherical coordinate system

This coordinate system is three-dimensional. The position of a point is specified by the distance from the pole, the polar angle and the azimuth angle. The cartesian coordinates are determined by

$$\begin{aligned} x &= \rho\, cos(\theta)\, cos(\phi), \\ y &= \rho\, cos(\theta)\, sin(\phi), \\ z &= \rho\, sin(\theta). \end{aligned}$$

```
Manipulate[
 point = {ρ * Cos[angles[[2]]] * Cos[angles[[1]]],
   ρ * Cos[angles[[2]]] * Sin[angles[[1]]], ρ * Sin[angles[[2]]]};
 Show[Graphics3D[{PointSize[0.015], Point[point],
    Thickness[0.003], Line[{{0, 0, 0}, point}], Opacity[0],
    EdgeForm[{Lighter[Gray], Thickness[0.001]}],
    Cylinder[{{0, 0, 0}, {0, 0, 0.01}}, ρ],
    EdgeForm[{Black}],
                            ρ * Cos[angles[[2]]] * Cos[angles[[1]] + π/2]
    Cylinder[{{0, 0, 0}, {―――――――――――――――――――――――――――――――――――――――――――――,
                                              100
       ρ * Cos[angles[[2]]] * Sin[angles[[1]] + π/2]
       ―――――――――――――――――――――――――――――――――――――――――――――, 0}}, ρ],
                        100
    EdgeForm[{Black}],
    Cylinder[{{0, 0, point[[3]]}, {0, 0, point[[3]] + 0.01}},
     Abs[ρ * Cos[angles[[2]]]]],
    Lighter[Gray], Opacity[0.15], Sphere[{0, 0, 0}, ρ]}, ViewPoint → {1, 0.5, 1}],
  PlotRange → {{-11, 11}, {-11, 11}, {-11, 11}}, ImageSize → {250},
  AxesOrigin → {0, 0, 0}, Axes → True, Ticks → All, Boxed → False,
  BoxRatios → Automatic, SphericalRegion → True],
 Column[{Row[{Control[{ρ, 5, 10}], Spacer[20], "ρ = ", Dynamic[ρ]}],
   Row[
                                                π               π
    {Labeled[Control[{{angles, {0, 0}, ""}, {0, -―}, {2 * π, ―}, ImageSize → Small}],
                                                2               2
      {"ϕ", "θ"}, {Bottom, Left}], Spacer[160],
                                   angles[[1]]
     Column[{Row[{"ϕ = ", Dynamic[―――――――――――], " π"}],
                                        π
                                 angles[[2]]
       Row[{"θ = ", Dynamic[―――――――――――], " π"}]}]}]}]]
                                      π
```

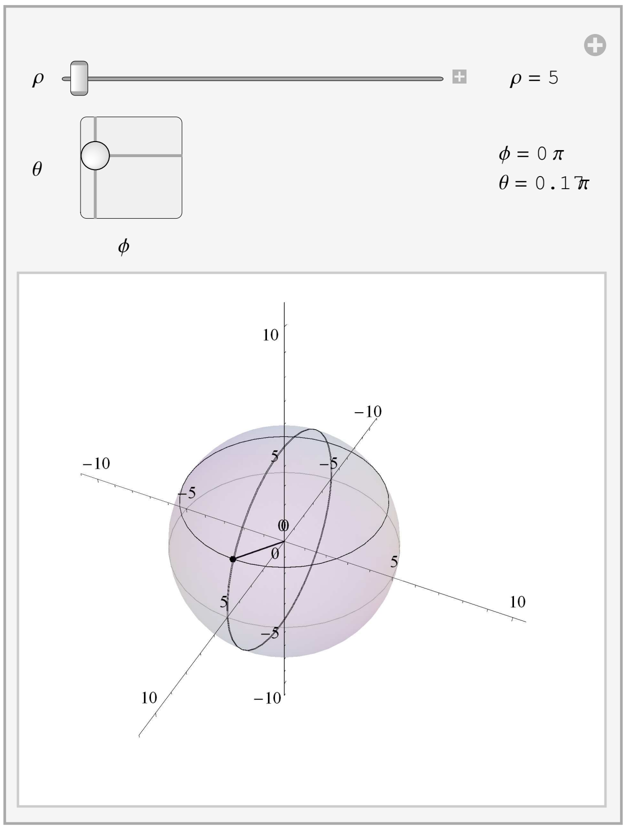


Figure 3.4.19: Spherical coordinate system.

```
Clear[point];
```

3.4.3 Cylindrical coordinate system

The cylindrical coordinate system is another three-dimensional coordinate system which is obtained by replacing *x* and *y* with polar coordinates *ρ* and *ϕ* and leaving the z coordinate the same, i.e.

$$x = \rho \, cos(\phi)$$
$$y = \rho \, sin(\phi)$$
$$z = h$$

```
Manipulate[
 point = {ρ*Cos[ϕ], ρ*Sin[ϕ], h};
 Show[Graphics3D[{PointSize[0.015], Point[point],
    Thickness[0.003], Line[{{0, 0, h}, point}],
    Lighter[Gray], Opacity[0.15], Cylinder[{{0, 0, 0}, {0, 0, h}}, ρ]},
   ViewPoint → {1, 1, 0.5}], PlotRange → {{-11, 11}, {-11, 11}, {-11, 11}},
  ImageSize → {250}, AxesOrigin → {0, 0, 0}, Axes → True, Ticks → All,
  Boxed → False, BoxRatios → Automatic, SphericalRegion → True],
 Column[{Row[{Control[{ρ, 1, 7}], Spacer[40], "ρ = ", Dynamic[ρ]}],
   Row[{Control[{ϕ, 0, 2*π}], Spacer[40], "ϕ = ", Dynamic[ϕ/π], " π"}],
   Row[{Control[{h, -7, 7}], Spacer[40], "h = ", Dynamic[h]}]}]]
```

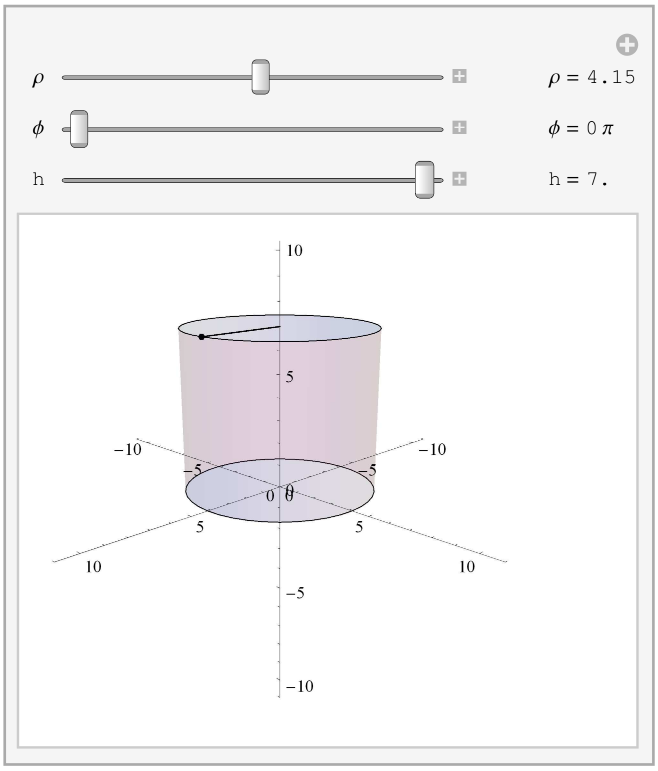


Figure 3.4.20: Cylindrical coordinate system.

```
Clear[point];
```

3.5 Double integral

This illustration shows how a definite integral of $f(x, y) = 2 - 2\sin(x^2 + y^2)$ can be calculated using different numerical methods and the convergence of Riemann sums.

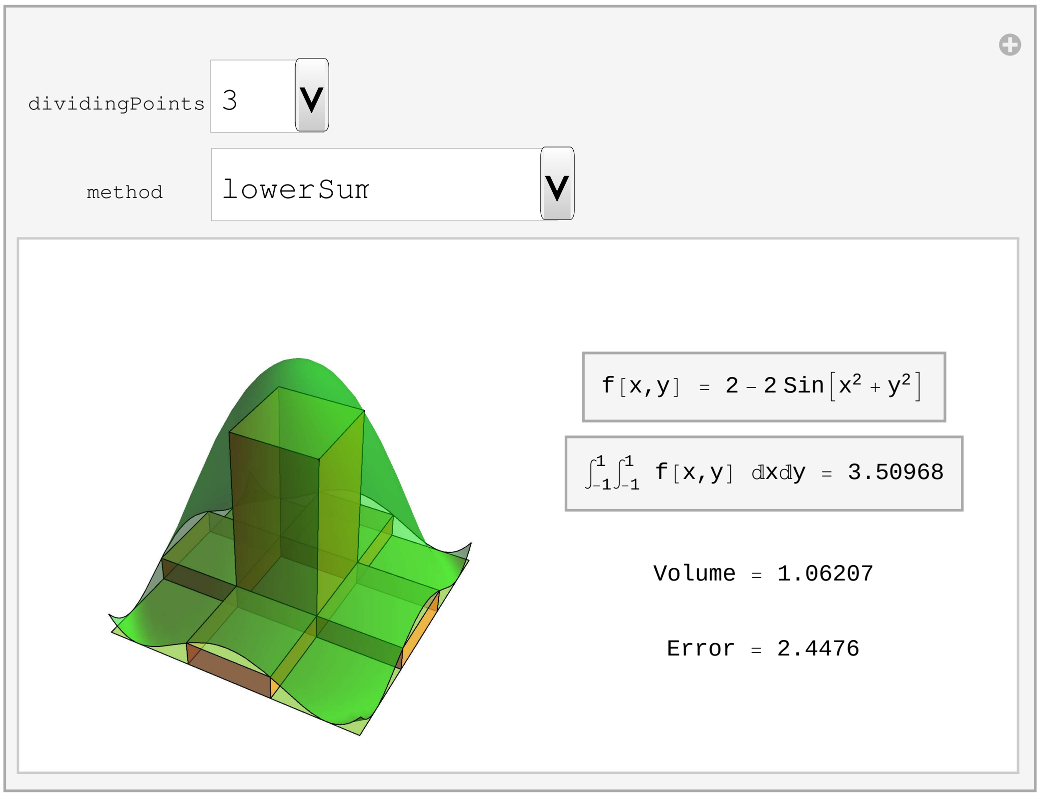
The interval of integration is divided into subintervals according to the number of dividing points. The approximation is performed by finding the volume of the collection of cuboids. The lengths and widths of these cuboids are defined by the size of subintervals. Their heights are determined by the approximation method:

- Lower sum – the height is equal to the infimum of f on the specific subinterval.
- Upper sum – the height is equal to the supremum of f on the specific subinterval.

- Midpoint rule – the height is equal to the value of *f* in the middle of the subinterval [9].
- Trapezoidal rule – the height is equal to the average of endpoints of subinterval [9].

The error is increasing or decreasing depending on the number of dividing points and the technique of approximation.

```
Manipulate[
 Module[{f, plotf, plotInterval, plotSize, intervalSize, partition,
   intervals, intervals2, fvalue, sum},
  f[x_, y_] = 2 - 2 * Sin[x^2 + y^2];
  plotf = Plot3D[f[x, y], {x, -1, 1}, {y, -1, 1}, PlotStyle → {Opacity[0.5], Green},
    Mesh → None, Boxed → False, Axes → None];
  plotInterval = {-1, 1};
  plotSize = plotInterval[[2]] - plotInterval[[1]];
  intervalSize = plotSize / dividingPoints;
  partition =
   Partition[Table[i, {i, plotInterval[[1]], plotInterval[[2]], intervalSize}],
    2, 1];
  intervals = Flatten[Table[{partition[[i]][[1]], partition[[j]][[2]]},
     {i, 1, dividingPoints}, {j, 1, dividingPoints}], 1];
  intervals2 =
   Flatten[Table[partition[[i]][[1]] < x < partition[[i]][[2]] &&
      partition[[j]][[1]] < y < partition[[j]][[2]], {i, 1, dividingPoints},
     {j, 1, dividingPoints}]];
  fvalue = Switch[method,
    lowerSum, Table[NMinimize[{f[x, y], intervals2[[i]]}, {x, y}, PrecisionGoal → 2][[
      1]], {i, 1, dividingPoints^2}],
    upperSum,
    Table[NMaximize[{f[x, y], intervals2[[i]]}, {x, y}, PrecisionGoal → 2][[1]],
     {i, 1, dividingPoints^2}],
    midpointRule,
    Table[f[intervals[[i]][[1]] + intervalSize / 2,
      intervals[[i]][[2]] - intervalSize / 2], {i, 1, dividingPoints^2}],
    trapezoidalRule, Table[
     1/4 * (f[intervals[[i]][[1]], intervals[[i]][[2]]] +
        f[intervals[[i]][[1]] + intervalSize, intervals[[i]][[2]]] +
        f[intervals[[i]][[1]], intervals[[i]][[2]] - intervalSize] +
        f[intervals[[i]][[1]] + intervalSize, intervals[[i]][[2]] - intervalSize]),
     {i, 1, dividingPoints^2}]];
  sum = N[Total[Table[intervalSize^2 * fvalue[[i]], {i, 1, dividingPoints^2}]]];
  Labeled[Show[{plotf,
      Table[Graphics3D[{Opacity[0.7], Lighter[Yellow],
         Cuboid[{intervals[[i]][[1]], intervals[[i]][[2]], fvalue[[i]]},
          {intervals[[i]][[1]] + intervalSize, intervals[[i]][[2]] - intervalSize,
           0}]}], {i, 1, dividingPoints^2}]},
    ImageSize → {250, 250}, BoxRatios → {1, 1, 1}, Boxed → False],
  Pane[Column[{StandardForm[Panel[Row[{"f[x,y] = ", f[x, y]}]]],
     StandardForm[
      Panel[Row[{"∫_{-1}^{1}∫_{-1}^{1} f[x,y] dxdy = ",
          N[Integrate[f[x, y], {x, -1, 1}, {y, -1, 1}]]}]]],
     Pane[Row[{"Volume = ", sum}], ImageMargins → 20],
     Pane[
      Row[{"Error = ",
        Abs[sum - N[Integrate[f[x, y], {x, -1, 1}, {y, -1, 1}]]]}]]},
    Center], ImageMargins → 10], Right]],
{{dividingPoints, 3}, {3, 4, 5, 6, 7, 8, 9, 10}},
{{method, lowerSum}, {lowerSum, upperSum, midpointRule , trapezoidalRule }},
ControlType → {PopupMenu, PopupMenu}]
```



dividingPoints 3

method lowerSum

$f[x,y] = 2 - 2\,\mathrm{Sin}[x^2 + y^2]$

$\int_{-1}^{1}\int_{-1}^{1} f[x,y]\ dxdy = 3.50968$

Volume = 1.06207

Error = 2.4476

Figure 3.5.21: Approximation of the definite integral of the function $2 - 2\sin(x^2 + y^2)$ using selected methods.

4 Bivariate limits

4.1 Bivariate limits in Mathematica

Mathematica does not provide any function for calculating limits of functions of two variables. The only way is to use the command *WolframAlpha* which sends the input to WolframAlpha and imports the output. However, it requires the Internet access.

```
WolframAlpha["limit (x^2*y^2)/(x^2+y^2) as x→0, y→0", "PodCells"]
```

$$\left\{ \lim_{\{x,y\}\to\{0,0\}} \frac{x^2 y^2}{x^2+y^2},\ 0 \ \text{(assuming variableare real-valued)} \right\}$$

Although it looks like it works pretty well, the results of some computations can be incorrect. For instance, the limit of the function $\frac{2xy}{xy+2x-y}$, where both x and y tend to zero.

```
WolframAlpha["limit (2*x*y)/(x*y + 2*x - y) as x→0, y→0", "PodCells"]
```

$$\left\{ \lim_{\{x,y\}\to\{0,0\}} \frac{2\,x\,y}{x\,y+2\,x-y},\ 0 \ \text{(assuming variableare real-valued)} \right\}$$

WolframAlpha returns 0 but the limit does not exist as we proved in example 2.2.5

4.2 Function doubleLimit

Due to the insufficient performance of the built-in function *limit* of WolframAlpha, we have decided to define our own function named *doubleLimit*. It consists of three modules, each of which uses a different method.
At first, we use the method of sequential limits. If both are finite, this module can determine that either the limit does not exist or the limit would equal some specific value if it existed.

```
sequentialLimits[f_, {x_, y_} → {ax_, ay_}] :=
  Module[{seqLim2},
   If[Head[f] === If || Head[f] === Piecewise || Head[f] === Which ||
     Head[f] === Switch || Head[f] === Boole, "The limit may still exist.",
    seqLim1 = Limit[Limit[f, y → ay], x → ax];
    seqLim2 = Limit[Limit[f, x → ax], y → ay];
    If[NumericQ[seqLim1] && NumericQ[seqLim2] && seqLim1 ≠ seqLim2,
     "The limit does not exist.",
     If[NumericQ[seqLim1] && NumericQ[seqLim2] && seqLim1 == seqLim2,
      Row[{"The limit may exist and if it does then it has to be ",
        seqLim1, "."}], "The limit may still exist."]]]];
```

The second method how the nonexistence of a limit can be proven is approaching the point at which the limit is searched via various curves of the form $x = ky^{\frac{i}{j}}$. If the function is a polynomial, the curves are produced for i and j run from 1 to the highest exponent from all of the numerators and denominators of the variables. In case the function is not polynomial, the highest exponent is increased by 1 in order to construct lines at least. The output of this module can be that the limit does not exist or what the limit would be provided it existed.

```
curves[f_, {x_, y_} → {ax_, ay_}] :=
  Module[{g, nonexistence, exponents, highestExp, k},
   If[Head[f] === If || Head[f] === Piecewise || Head[f] === Which ||
     Head[f] === Switch || Head[f] === Boole, "The limit may still exist.",
    g = Factor[f];
    nonexistence = False;
    exponents = {Numerator[Exponent[Numerator[g], {x, y}]],
      Denominator[Exponent[Numerator[g], {x, y}]],
      Numerator[Exponent[Denominator[g], {x, y}]],
      Denominator[Exponent[Denominator[g], {x, y}]]};
    highestExp = Max[exponents] + 1;
    For[i = 1, i ≤ highestExp, i++,
     For[j = 1, j ≤ highestExp, j++,
      lim = Limit[g /. x → k * Abs[y - ay]^(i/j) + ax, y → ay];
      If[((Exponent[Numerator[lim], k] ≠ 0 || Exponent[Denominator[lim], k] ≠ 0) &&
         Exponent[lim, k] ≠ -Infinity), {nonexistence = True, Break[]},
       Unevaluated[Sequence[]]]]];
    If[nonexistence, "The limit does not exist.",
     If[NumericQ[lim],
      Row[{"The limit may exist and if it does then it has to be ", lim, "."}],
      "The limit may still exist."]]]];
```

The purpose of the last module is to find the limit employing the definition. Nonetheless, it can also state whether or not it exists. The function is maximized and minimized on two-dimensional interval of the size $2d \times 2d$, where the point at which the limit should be found is in the middle of this interval. The result of this maximization and minimization are one-variable functions dependent on d. If they have the same limit as d tends to zero, the searched limit exists and is equal to the their common value. If they differ, the limit does not exist. This method, which is described at [12], works fine typically for rational functions.

```
maxMinMethod[f_, {x_, y_} → {ax_, ay_}] :=
  Module[{d, max, min, limMax, limMin},
   max = Maximize[{f, 0 < d < 1 && -d ≤ x - ax ≤ d && -d ≤ y - ay ≤ d}, {x, y}];
   min = Minimize[{f, 0 < d < 1 && -d ≤ x - ax ≤ d && -d ≤ y - ay ≤ d}, {x, y}];
   limMax = Limit[max[[1]], d → 0];
   limMin = Limit[min[[1]], d → 0];
   If[(limMax == Infinity && limMin == -Infinity) ||
     (limMax == -Infinity && limMin == Infinity), "The limit does not exist.",
    If[! NumericQ[limMax] || ! NumericQ[limMin], "Function cannot find the limit.",
     If[limMax == limMin, limMax, "The limit does not exist."]]]];
```

Note that this method can be easily transformed to search limits of more than two-variable functions. All three methods are used in the following complex module termed *doubleLimit*.

```
doubleLimit[f_, {x_, y_} → {ax_, ay_}] :=
  Module[{possibleLimit, limit, seqLim1, lim},
   possibleLimit = False;
   limit = sequentialLimits[f, {x, y} → {ax, ay}];
   If[limit === "The limit does not exist.", limit,
    If[limit === Row[{"The limit may exist and if it does then it has to be ",
        seqLim1, "."}], possibleLimit = limit];
    limit = curves[f, {x, y} → {ax, ay}];
    If[limit === "The limit does not exist.", limit,
     If[
      limit === Row[{"The limit may exist and if it does then it has to be ",
          lim, "."}]
       && possibleLimit === False, possibleLimit = limit];
     limit = maxMinMethod[f, {x, y} → {ax, ay}];
     If[limit === "The limit does not exist." || NumericQ[limit], limit,
      If[possibleLimit =!= False, possibleLimit,
       "Function cannot find the limit."]]]]];
```

Here are some examples of applications:

```
doubleLimit[(x^2 * y^2)/(x^2 + y^2), {x, y} → {0, 0}]
```

```
0
```

```
doubleLimit[(2 * x * y)/(x * y + 2 * x - y), {x, y} → {0, 0}]
```

```
The limit does not exist.
```

```
doubleLimit[(x^2 - y^2)/(x^2 + y^2), {x, y} → {0, 0}]
```

```
The limit does not exist.
```

```
doubleLimit[Cos[π * x^2/(4 * x^2 + y^4)], {x, y} → {0, 0}]
```

```
The limit does not exist.
```

```
Clear[sequentialLimits, seqLim1, curves, lim, maxMinMethod, doubleLimit];
```

4.2.1 Method of critical paths

There is one more method published at [8] which we intended to employ for calculating bivariate limits. It is used and delineated by Maplesoft in Maple version 17. It is based on so called critical paths. The function attains its local maximum or minimum subjected to the constraint C, which is a circle with radius ρ and centre $\vec{a} = (a_x, a_y)$, if the gradient of the function and the gradient of the constraint are parallel. Thus, the following equation can be derived:

$$\frac{\partial f}{\partial x}\frac{\partial C}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial C}{\partial x} = 0.$$

Together with the equation of the function, it determines the critical paths. If all limits along these paths are the same, the bivariate limit exists and equals this common value. On the other hand, if at least one of them differs, it can be said that the bivariate limit does not exist.
Although it works in theory, the reality is much worse. Function *Solve* has only limited power how to find the solution, especially of more complicated equations. Moreover, this method solves the same kind of examples as the module *maxMinMethod*.
Taking into consideration these facts, we have decided not to use this method.

4.2.2 Possible issues

The success of finding the bivariate limit depends on the built-in function *Limit* which does not have to be so trustworthy as one would expect. For example, the following limit should not exist as the one-sided limits differ.

```
Limit[Abs[x]/x, x → 0]

1
```

However, Mathematica declares that it exists because, by default, it computes the limit from the right.

5 Ballistic curves

A ballistic curve is a path of a projectile in the gravity field. Besides the gravitational force, there are other forces acting on the projectile. One of the them is air resistance (also called drag). The earth's rotation is neglected, and the height between the projectile and the earth is assumed to be insignificant in comparison with the earth's dimensions in order that the gravitational constant would not change.

5.1 Ballistic trajectory without air resistance

The projectile is launched at a velocity of

$$\vec{v} = \left(v_{x,}\ v_y\right).$$

The *x*- and *y*-component can be expressed as

$$v_x = v\,cos(\phi)\,,$$
$$v_y = v\,sin(\phi),$$

where ϕ is the angle between *x*-axis and vector $\vec{v}$.
Let us assume that the projectile is launched from the point (0, 0). The gravitational force influences only the vertical component. The trajectory is given by the parametric equations

$$x(t) = v_x\,t, \tag{3}$$

$$y(t) = v_y\,t - \frac{1}{2}\,gt^2. \tag{4}$$

5.2 Ballistic trajectory with air resistance

Air resistance is a kind of force which acts in the opposite way to the motion of an object with respect to the surrounding fluid [13].
We will consider two kinds of drag – linear and quadratic. The former is usually used for lower velocity, the latter usually for higher velocity [10]. They are given by the equations

$$\vec{F}_L = -c\,\vec{v},\ \ [5]$$
$$\vec{F}_Q = -c\,\vec{v}^2,\ \ [5]$$

where *c* is the drag coefficient that is used to quantify the drag of an object. It is associated with the surface area of the object and is obtained from laboratory experiments. The value usually ranges between 0.4 and 1 [2].

5.2.1 Drag proportional to velocity

The total force in *x*- and *y*-direction should be expressed as

$$F_x = ma_x = -cv_x \tag{5}$$

and

$$F_y = ma_y = -cv_y - mg. \tag{6}$$

Note that only F_y is influenced by the gravitational force. Since the trajectory should be found, the following differential equations are derived by alteration of (5) and (6):

$$\frac{d^2 x}{d t^2} = -\frac{c}{m}\frac{d x}{d t}, \tag{7}$$

$$\frac{d^2 y}{d t^2} = -\frac{c}{m}\frac{d y}{d t} - g. \tag{8}$$

As equations (7) and (8) are independent, they can be solved separately. The particular solutions are fixed by the initial conditions $x'(0) = v_x$, $y'(0) = v_y$, $x(0) = 0$, and $y(0) = 0$.

```
DSolve[{x''[t] == -c* x'[t]/m, x'[0] == vx, x[0] == 0}, x[t], t] // FullSimplify
```

$$\left\{\left\{x[t] \to \frac{\left(1 - e^{-\frac{c t}{m}}\right) m\, vx}{c}\right\}\right\}$$

```
DSolve[{y''[t] == -c* y'[t]/m - g, y'[0] == vy, y[0] == 0}, y[t], t] // FullSimplify
```

$$\left\{\left\{y[t] \to \frac{m\left(g\, m - c\, g\, t + c\, vy - e^{-\frac{c t}{m}}(g\, m + c\, vy)\right)}{c^2}\right\}\right\}$$

The equations of the trajectory are

$$x(t) = \frac{\left(1 - e^{-\frac{c t}{m}}\right) m\, v_x}{c}, \tag{9}$$

$$y(t) = \frac{1}{c^2}\, m\left(g\, m - c\, g\, t + c\, v_y - e^{-\frac{c t}{m}}\left(g\, m + c\, v_y\right)\right). \tag{10}$$

5.2.2 Drag proportional to velocity squared

The total force in *x*- and *y*-direction is

$$F_x = ma_x = -cv_x^2,$$
$$F_y = ma_y = -cv_y^2 - mg.$$

Thus, the trajectory can be expressed as the following differential equations:

$$\frac{d^2x}{dt^2} = -\frac{c}{m}\frac{dx}{dt}\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}, \qquad (11)$$

$$\frac{d^2y}{dt^2} = -\frac{c}{m}\frac{dy}{dt}\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} - g. \qquad (12)$$

Unlike (7) and (8), equations (11) and (12) are dependent and must be solved as a system of differential equations. However, Mathematica is not able to solve them analytically. As a consequence, *NDSolve* is used instead of *DSolve* to find numerical solution. Hence, *c*, *m*, and *g* must be equal to some specific value. The initial conditions are $x'(0) = v_x$, $y'(0) = v_y$, $x(0) = 0$, and $y(0) = 0$.

```
NDSolve[{x''[t] == -c/m*x'[t]*Sqrt[x'[t]^2 + y'[t]^2] /. {c→1, m→1, g→9.81},
  y''[t] == -g - c/m*y'[t]*Sqrt[x'[t]^2 + y'[t]^2] /. {c→1, m→1, g→9.81}, x'[0] == 1,
  y'[0] == 1, x[0] == 0, y[0] == 0}, {x[t], y[t]}, {t, 0, 3}]

{{x[t] → InterpolatingFunction[{{0., 3.}}, <>][t],
  y[t] → InterpolatingFunction[{{0., 3.}}, <>][t]}}
```

NDSolve returns only an approximate function, which differs according to the specific initial conditions.

5.3 Visualization

The trajectory of a projectile is now visualized using the equations (3) and (4) for trajectory without drag, and (9) and (10) for trajectory with drag proportional to the velocity. Drag proportional to the velocity squared is determined numerically according to the selected initial conditions. The mass *m* and constant *c* must be fixed for the purposes of visualization (we chose $m = 1$ and $c = 1$).

```
Manipulate[Module[{x1, y1, x2, y2, numSol},
  x1[t_] = vx*t;
                   1
  y1[t_] = vy*t -  - *g*t^2;
                   2
  If[f1 == 1, plot1 = ParametricPlot[
      {{x1[t] /. {vx -> v0*Cos[ϕ0], ϕ -> ϕ0, m -> 1, g -> 9.81, c -> 1},
        y1[t] /. {vy -> v0*Sin[ϕ0], ϕ -> ϕ0, m -> 1, g -> 9.81, c -> 1}}}, {t, 0, 3},
      PlotStyle -> Blue], plot1 = ParametricPlot[0, {t, 0, 3}]];
                     c*t
                   - ---
                      m
            (1 - ⅇ      ) *m*vx
  x2[t_] = ----------------------;
                    c
                                                  c*t
                                                - ---
                                                   m
            m* (g*m - c*g*t + c*vy - ⅇ             * (g*m + c*vy))
  y2[t_] = ----------------------------------------------------------;
                                      c^2
  If[f2 == 1, plot2 = ParametricPlot[
      {{x2[t] /. {vx -> v0*Cos[ϕ0], ϕ -> ϕ0, m -> 1, g -> 9.81, c -> 1},
        y2[t] /. {vy -> v0*Sin[ϕ0], ϕ -> ϕ0, m -> 1, g -> 9.81, c -> 1}}}, {t, 0, 3},
      PlotStyle -> Red], plot2 = ParametricPlot[0, {t, 0, 3}]];
  numSol =
                               c
   NDSolve[{x3''[t] == - - *x3'[t]*Sqrt[x3'[t]^2 + y3'[t]^2] /. {c -> 1, m -> 1, g -> 9.81},
                               m
                          c
      y3''[t] == -g - - *y3'[t]*Sqrt[x3'[t]^2 + y3'[t]^2] /. {c -> 1, m -> 1, g -> 9.81},
                          m
      x3[0] == 0, y3[0] == 0, x3'[0] == v0*Cos[ϕ0], y3'[0] == v0*Sin[ϕ0]},
    {x3[t], y3[t]}, {t, 0, 3}];
  If[f3 == 1, plot3 = ParametricPlot[{x3[t], y3[t]} /. numSol, {t, 0, 3},
      PlotStyle -> Darker[Green]], plot3 = ParametricPlot[0, {t, 0, 3}]];
  Show[{plot1, plot2, plot3}, PlotRange -> {{0, 11}, {0.1, 6}}, ImageSize -> 400]],
                                                              π                π
 Column[{Control[{{v0, 7.5, "v"}, 5, 10}], Control[{{ϕ0, -, "ϕ"}, 0, -}],
                                                              4                2
   Row[{Control[{{f1, 1, ""}, {0, 1}}],
      Text[Style[" without drag", Medium, Blue]]}],
   Row[{Control[{{f2, 1, ""}, {0, 1}}],
      Text[Style[" drag proportional to velocity", Medium, Red]]}],
   Row[{Control[{{f3, 1, ""}, {0, 1}}],
      Text[Style[" drag proportional to velocity squared", Medium,
        Darker[Green]]]}]}]]
```

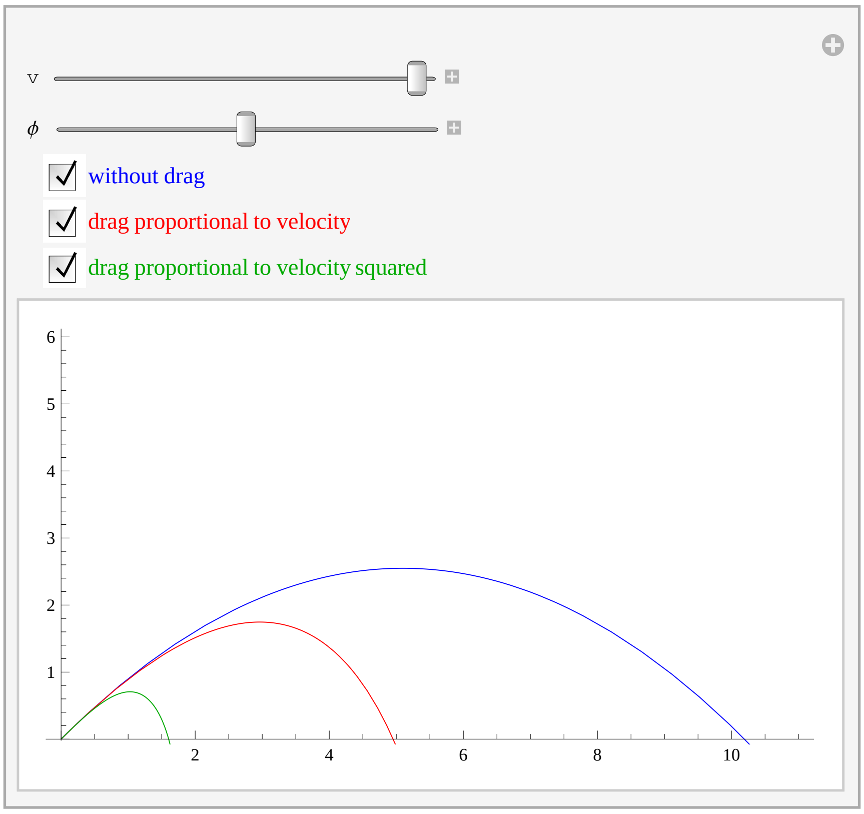

v
ϕ
without drag
drag proportional to velocity
drag proportional to velocity squared
6
5
4
3
2
1
2
4
6
8
10

Figure 5.3.22: Trajectory of a projectile without drag and with drag proportional to the velocity and velocity squared.

Conclusion

An interactive tool for students of the Mathematics 4 course has been developed. This project could help them to understand the problems on many solved examples. The advantages of computer algebra system Wolfram Mathematica have been demonstrated on particular examples. On the other hand, we were also able to show that Mathematica is not unfailing and sometimes gives bad results without any warning.

Since the source code is included for all examples, this project can also help learning the basics of Mathematica, although it is not the main purpose.

The question of bivariate limits has been discussed. Our own function has been created to find them or disprove their existence using various methods. It works very good; especially when it is applied to a rational function, and covers the needs of the Mathematics 4 course.

Furthermore, the problem of ballistic curves has been solved and visualized.

Several interactive visualizations have been developed to help students understand some notions and problems. Moreover, they could be also used as an interactive tool by lecturers.

In the future, the function for calculating bivariate limits could be upgraded to operate on more difficult examples. Furthermore, other courses of mathematics could be processed in a similar way to create complex materials for mathematics courses at our Faculty.

Bibliography

[1] Abell, Martha L., and James P. Braselton. *Mathematica By Example*. 3rd ed. Boston: Academic, 2004. Print.

[2] Halliday, David, Robert Resnick, and Jearl Walker. *Fyzika: Část 1 Mechanika*. Trans. Petr Dub and Jan Obdržálek. Brno: VUTIUM, 2000. Print.

[3] Mangano, Sal. *Mathematica Cookbook*. Sebastopol, CA: O'Reilly, 2010. Print.

[4] Marsden, Jerrold and Alan Weinstein. *Calculus III*. New York: Springer-Verlag, 1985. Print.

[5] Štoll, Ivan. *Mechanika*. Prague: Česká Technika, 2010. Print.

[6] About WolframAlpha. *Making the World's Knowledge Computable*. N.p., [2014]. Web. 24 June 2014. <http://www.wolframalpha.com/about.html>.

[7] About Wolfram Research. *Wolfram Research Company Background*. N.p., [2014]. Web. 24 June 2014. <http://www.wolfram.com/company/background.html>.

[8] Bivariate Limits. *Updates/Maple17/BivariateLimits – Help*. N.p., [2014]. Web. 25 June 2014. <http://www.maplesoft.com/support/help/Maple/view.aspx?path=updates/Maple17/BivariateLimits>.

[9] Dawkins, Paul. *Pauls Online Notes: Calculus II – Approximating Definite Integrals*. N.p., [2014]. Web. 24 June 2014.
<http://tutorial.math.lamar.edu/Classes/CalcII/ApproximatingDefIntegrals.aspx>.

[10] Erlichson, Herman. *Maximum Projectile Range with Drag and Lift, with Particular Application to Golf*(n.d.): n. pag. 16 June 1982. Web. 24 June 2014.
<http://http://ww2.odu.edu/∼agodunov/teaching/phys420/files/Erlichson.pdf>.

[11] Franc, Jiří, and Matěj Tušek. *Sbírka úloh k předmětu Matematika 4 (MAT4)* (n.d.): n. pag. 25 Feb. 2014. Web. 24 June 2014.
<http://kmlinux.fjfi.cvut.cz/∼tusekmat/download/MAT4_sbirka.pdf>.

[12] Michael E2. Finding Limits in Several Variables. *Functions*. N.p., 18 Mar. 2013. Web. 24 June 2014. <http://mathematica.stackexchange.com/questions/21544/finding-limits-in-several-variables>.

[13] Wikipedia Contributors. *Drag (physics)*. Wikipedia. Wikimedia Foundation, 18 June 2014. Web. 24 June 2014. <http://en.wikipedia.org/wiki/Drag_(physics)>.

[14] Wikipedia Contributors. *Mathematica*. Wikipedia. Wikimedia Foundation, 22 June 2014. Web. 25 June 2014. <https://en.wikipedia.org/wiki/Mathematica>.

[15] WolframAlpha. *Personal Analytics*. N.p., [2014]. Web. 24 June 2014.
<http://www.wolframalpha.com/facebook/>.

[16] Wolfram Demonstrations Project. *Demonstrations RSS*. N.p., [2014]. Web. 24 June 2014.
<http://demonstrations.wolfram.com/>.