Matematika na počítači a základy programování

Zuzana Morávková, Radomír Paláček

Katedra matematiky a deskriptivní geometrie
Vysoká škola báňská – Technická Univerzita Ostrava

OBSAH

ÚVOD

Studijní materiály jsou určeny pro studenty prvního ročníku *Hornicko-geologické fakulty* Vysoké školy báňské – Technické Univerzity Ostrava pro předmět *Matematika na počítači a základy programování*.

Pro inženýra i bakaláře je důležité nejen pochopení matematických principů, ale také efektivní zvládnutí programů, které bude v praxi používat pro matematické operace. Během praxe se zajisté setká s potřebou vykreslit graf, vypočítat hodnotu funkce, vyřešit rovnici či nerovnici, spočítat derivaci, nalézt extrém funkce nebo vyřešit soustavu lineárních rovnic. Neméně důležité je umět úlohu v praxi rozebrat a nalézt postup řešení, k čemuž je potřeba ovládat základy algoritmizace.

Studijní text je psán pro studenty kombinovaného i prezenčního studia. Je však vhodný i pro samostudium — tomu odpovídá struktura kapitol. Na začátku každé z nich jsou uvedeny předpokládané znalosti matematiky, případně jsou tyto znalosti stručně zopakovány. Předmět Matematika na počítači a základy programování je vyučován ve stejném semestru jako předmět *Matematika I*, jehož obsahem je *Diferenciální počet (funkcí jedné proměnné)* a základy *Lineární algebry*.

Dále jsou uvedeny předpokládané znalosti softwaru s odkazy na předchozí kapitoly, ve kterých je učivo vysvětleno. Následuje nové učivo doplněné drobnými příklady na osvojení probírané problematiky. V každé kapitole je vždy vysvětlen nový matematický aparát a také nové typy objektu či proměnných, nové příkazy, prvky programu. Důležitou část většiny kapitol tvoří řešené aplikované úlohy, které studenti prezenčního studia řeší nejprve samostatně a poté s pedagogem na cvičeních. Studenti kombinovaného studia mají řešení popsáno krok po kroku a mohou ho sami doma znova nastudovat, neboť časová dotace jejich studia neumožňuje vše podrobně probrat na výuce. Kapitola je zakončena domácími úkoly, což jsou neřešené úlohy navazující na příklady z příslušné kapitoly.

Poděkování

Skriptum vzniklo za finanční podpory projektu FRVŠ 2464/2012 „Inovace počítačových předmětů na Hornicko-geologické fakultě, Vysoké škole báňské – Technické univerzitě Ostrava“. Ráda bych také poděkovala ostatním vyučujícím předmětu Radomíru Paláčkovi, Michaele Tužilové a obzvláště Janě Bělohlávkové za jejich spolupráci při tvorbě domácích úkolů a semestrálních projektů a za jejich postřehy při tvorbě celého předmětu.

Jak pracovat se skripty

Studentům vřele doporučuji vyzkoušet si všechny příklady, které jsou v textu uvedeny. Nejprve si zkuste sami vymyslet řešení a budete-li zcela v úzkých, přečtěte si postup popsaný ve skriptech. Budete občas mile překvapeni, že jste na řešení přišli bez pomoci a o to víc bude vaše práce radostná.
Časová náročnost na jednu kapitolu odpovídá cvičení v délce 3 vyučovacích hodin (asi 120 min). Při samostudii je předpokládaný čast strávený nad jednou kapitolou asi 2–3 hodiny.
Přeji příjemné studium a práci se skripty.

Část I

GeoGebra

KAPITOLA

1

STRUČNÝ ÚVOD DO PROGRAMU GEOGEBRA

NAUČÍME SE

Seznámíme se s uživatelské rozhraní programu Geogebra a s jeho částmi tzv. Náhledy. Vyzkoušíme si práce s Nástroji a Vstupní polem. Představíme si základní objekty (číslo, úhel, bod, přímka) a naučíme se měnit jejich vzhled, jméno nebo hodnotu.

VÝKLAD UČIVA

1.1 První seznámení s programem GeoGebra

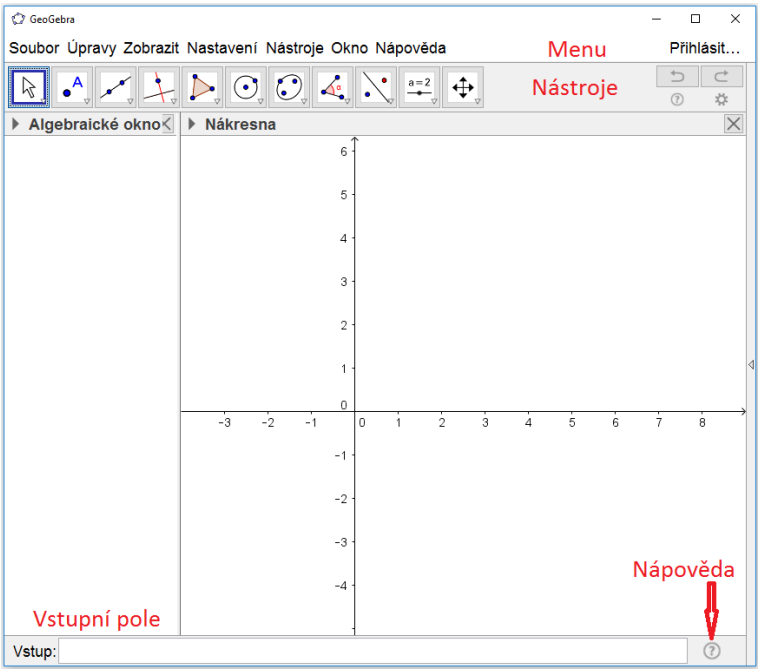
1.1.1 Základní informace

Program je k dispozici na webu http://geogebra.org

GeoGebra je volně šiřitelný matematický software pro studium a výuku, jehož předností je interaktivní grafika, propojená s algebrou a také možnosti tabulkového procesoru. Lze ho použít k řešení úloh z matematiky, geometrie, statistiky. K dispozici je řada volně přístupných výukových materiálů.

1.1.2 Uživatelské rozhraní

Po spuštění programu uvidíte okno, které je zobrazeno na obrázku 1.1.
Okno je rozděleno na jednotlivé Náhledy (části programu). Po spuštění je vždy **Algebraické okno** zobrazeno na levé straně a **Nákresna** vpravo. Nad těmito okny jsou umístěny lišty **Menu** a **Nástroje**. V dolní části je umístněno **Vstupní pole** pro zadávání příkazů. Práci s GeoGebrou usnadní **Nápověda**.



Obrázek 1.1: Program GeoGebra

PŘÍKLAD 1.

Pro první seznámení si vyzkoušíme zadat bod.

1.		V Menu vybereme Nástroj **Nový bod**, klikneme kdekoli do Nákresny a vytvoříme bod A. Jméno bodu A a jeho souřadnice se objeví i v Algebraickém okně.
2.		Vybereme Nástroj **Ukazovátko**. Myší klikneme na bod A a budeme-li držet levé tlačítko myši, můžeme pomoci myši bodem hýbat. V **Algebraickém okně** lze vidět, jak se mění souřadnice bodu A.

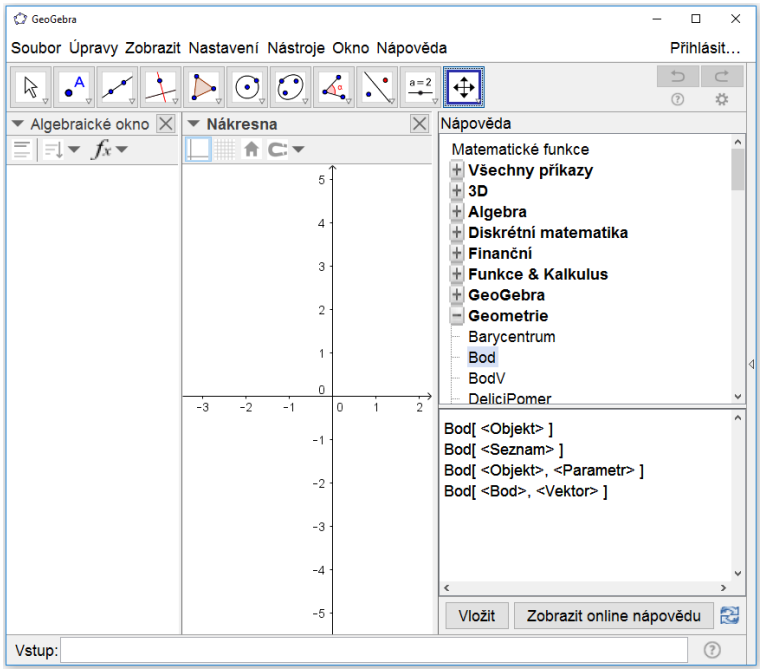
Program má několik částí zvaných Náhledy, teď jsme si vyzkoušeli Nákresnu a Algebraické okno. Každý objekt (např. bod) je uveden v Algebraickém okně a jde-li zobrazit geometricky, vidíme ho i v Nákresně.

1.1.3 Nápověda

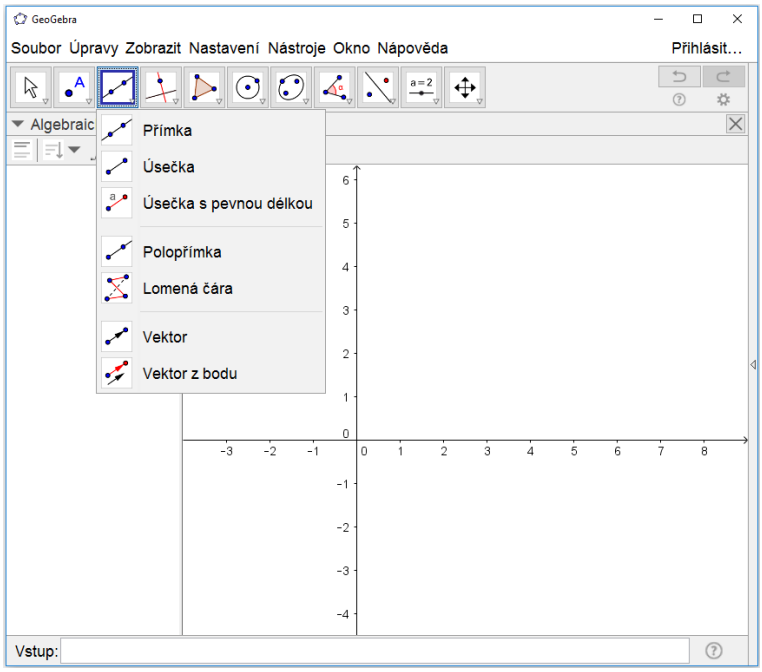
Nápovědu lze zobrazit kliknutím do pravého dolního rohu na symbol  . Rozvine se nabídka příkazů. Na obrázku 1.2 je ukázka nápovědy k příkazu `Bod`.

1.1.4 Nástroje

Další nabídka nástrojů se zobrazí při kliknutí na malý červený trojúhleník na ikoně nástroje, viz obrázek 1.3.



Obrázek 1.2: Otevření nápovědy

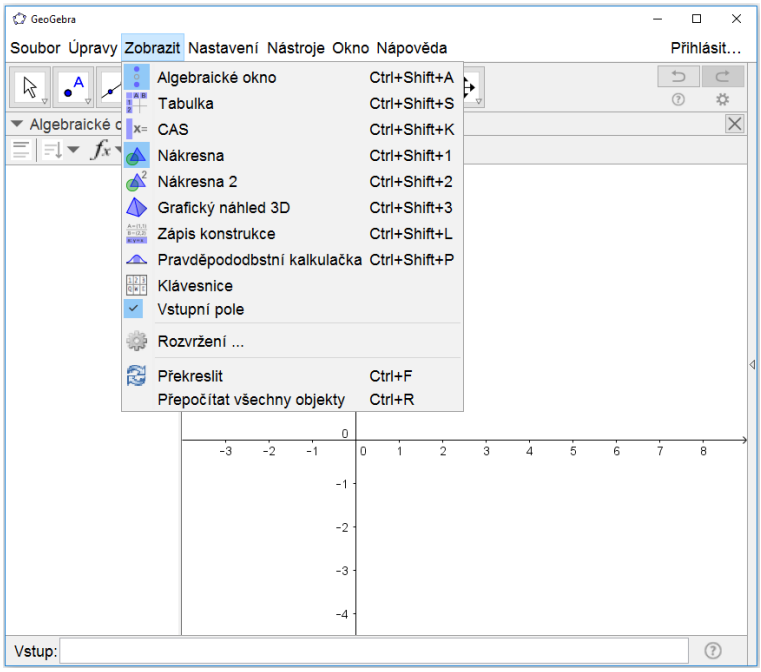


Obrázek 1.3: Nástroje

1.1.5 Náhledy

V Menu v položce **Zobrazit** je možno skrýt nebo zobrazit jednotlivé Náhledy. Na obrázku 1.4 vidíme, že aktuálně zobrazené Náhledy jsou označené zatržítkem.

V předchozím textu jsme se seznámili s Náhledem Algebraické okno a Nákresna. Nyní si popíšeme Vstupní pole.



Obrázek 1.4: Seznam Náhledů

1.1.6 Vstupní pole

Vstupní pole je obvykle umístěné ve spodní části okna GeoGebry, viz obrázek 1.5. Prostřednictvím menu **Zobrazit - Rozvržení** můžeme Vstupní pole umístit také v horní části pod Nástroji.
Do vstupního pole zapisujeme příkazy nebo zadáváme objekty. Vždy nakonec stiskneme klávesu *Enter*.

Obrázek 1.5: Vstupní pole

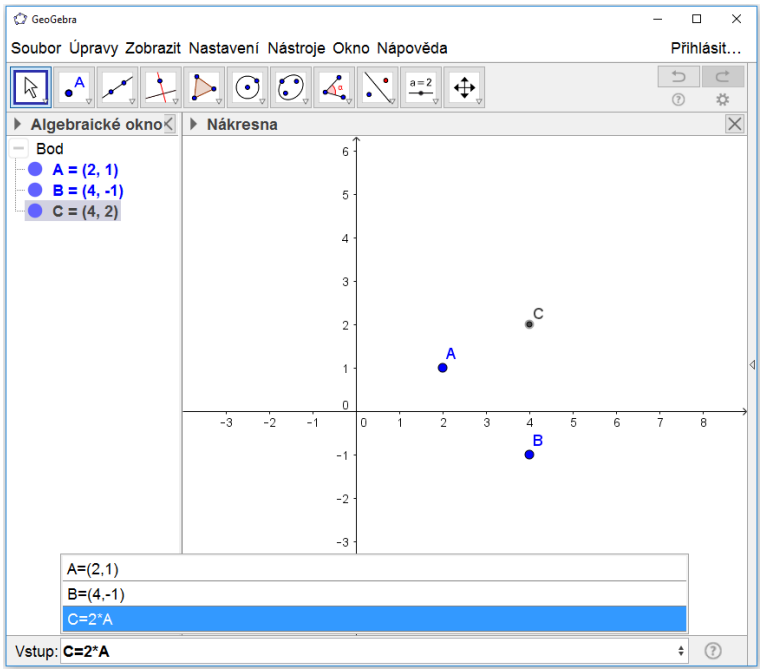
PŘÍKLAD 2.

Opět si zkusíme zadat bod, tentokrát pomocí vstupního pole.

1.	Vstup:	Bod A zadáme ze **Vstupního pole**, zapíšeme do něj `A=(2,1)` a stiskneme klávesu *Enter*.
2.	Vstup:	Bod B zadáme ze **Vstupního pole**, zapíšeme do něj `B=(4,-1)` a stiskneme klávesu *Enter*.
3.	Vstup:	Bod C zadáme ze **Vstupního pole**, zapíšeme do něj `C=2*A` a stiskneme klávesu *Enter*.

Historie vstupů

Na pravém konci Vstupního pole je nyní k dispozici symbol ⇕. Když na něho klikneme, tak můžeme pomocí šipek (nahoru, dolů) procházet historii vstupů, které jsme zadali, viz obrázek 1.6.



Obrázek 1.6: Historie příkazů

1.2 Objekty

V předchozích příkladech jsme si vyzkoušeli práci s objektem bod. GeoGebra umožňuje pracovat i s jinými objekty, ať již geometrickými (např. bod, vektor, přímka, úsečka, kružnice, funkce) nebo obecnými (číslo, úhel, text). GeoGebra obsahuje i tzv. Aktivní objekty (posuvník, tlačítko).

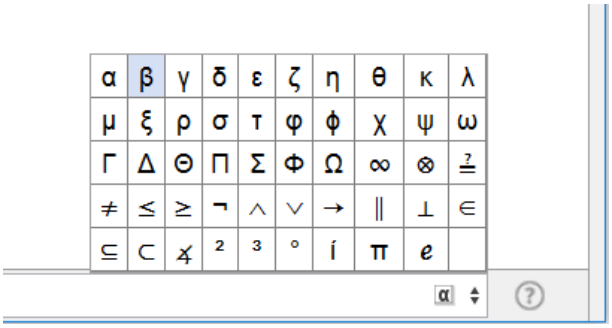
Název objektu

Každý objekt má svůj název, který mu program přidělí sám nebo mu ho sami dáme při zadání příkazu. Jak jsme si ukázali v příkladech 1 a 2.
Název objektu se skládá z písmen, číslic a lze použít i symbol podtržítko pro dolní index, případně řecká písmena. Rozlišují se velká a malá písmena.
Objekt se může jmenovat například `A`, `Objem`, `a`, `pocet`, `v_1`. Jména `x`,`y` jsou již vyhrazeno pro x-ovou a y-ovou souřadnici. Řecká písmena najdeme ve **Výběru**.

Výběr znaků a symbolů

Nabídka Výběr je k dispozice u Vstupního pole na jeho pravém konci, viz obrázek 1.7. Otevře se kliknutím na symbol α (symbol uvidíte, když do Vstupního pole kliknete myší).

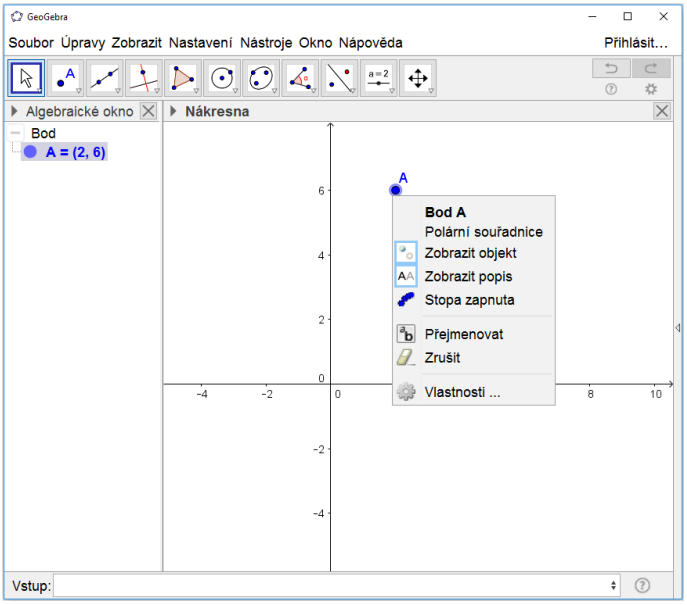
Ve výběru najdeme řecké písmena, konstanty, symboly pro množinové nebo logické operace. Všechny položky výběru můžete vidět na obrázku 1.7. V dalším textu budou znaky z výběru značeny šedě, například řecké písmeno β.



Obrázek 1.7: Otevření výběru

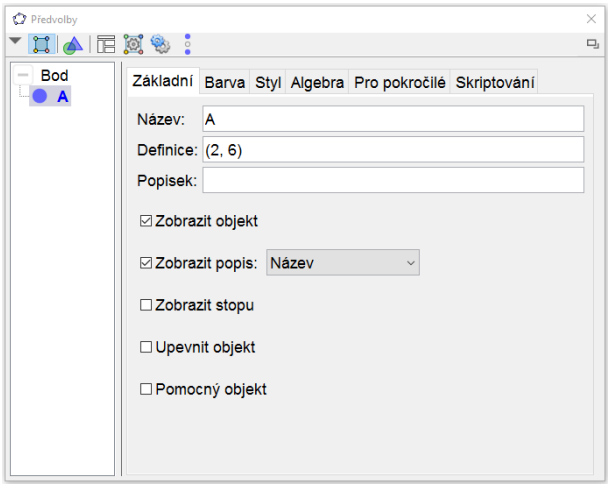
1.2.1 Vlastnosti objektu

U každého objektu (například bodu) lze měnit jeho název, vzhled a řadu jeho vlastností. Použijeme nástroj **Ukazovátko** a v Algebraickém okně nebo v Nákresně klikneme **pravým tlačítkem** myši na příslušný objekt a zobrazí se kontextová nabídka. Každý typ objektu má v nabídce různé položky. Například nabídka pro bod je zobrazena na obrázku 1.8.



Obrázek 1.8: Kontextová nabídka

Každý objekt má v kontextové nabídce položku **Vlastnosti**. Když ji vybereme tak se otevře **Okno Vlastnosti** (obrázek 1.9).
V záložce **Základní** vidíme název objektu v poli **Název**, jeho **Hodnotu** a případně **Popis**, což může být delší text obsahující mezery i diakritické znaménka.



Obrázek 1.9: Okno Vlastnosti

Můžeme zde skrýt objekt odtrhnutí nebo zatrhnutím **Zobrazit objekt**.
Lze změnit text, který se v Nákresně u objektu zobrazuje, a to v položce **Zobrazit popis**. Máme na výběr jeho **Název**, **Název & Hodnotu**, **Hodnotu** nebo **Popis**
V záložce **Barva**, **Styl** lze měnit vzhled objektu.

1.2.2 Základní obecné objekty - číslo, úhel

Číslo zadáme ze vstupního pole například takto `a=125`. Desetinné číslo se zadává s desetinnou tečkou `b=13.45`.
Obdobně zadáme úhel například ve stupních α=30° nebo v radiánech $\alpha = \frac{\pi}{3}$. Symboly pro stupně ° a Ludolfovo číslo π jsou ve výběru.
Poznámka: Počet zobrazených desetinných míst lze změnit v Menu - Nastavení - **Zaokrouhlování**. Při spuštění programu jsou nastavena dvě desetinná místa.

Operace a priorita operací

Seznam operací a jejich priorita jsou uvedeny v následujících tabulkách.

Operace

sčítání	+
odčítání	-
násobení	* nebo mezera
dělení	/
mocnina	^ nebo 2, 3

Priorita operací

priorita	operace
1.	^
2.	* /
3.	+ -

Ostatní užitečné symboly

desetinná tečka	.
závorky	( )

PŘÍKLAD 3.

1.	Vstup:	Zadáme číslo `a=5`
2.	Vstup:	A jeho dvojnásobek spočítáme `b=2*a`
3.	Vstup:	A číslo $c = \frac{a+3}{4}$ zadáme `c=(a+3)/4` Nezapomene celého čitatele dát do závorek. Co by se stalo, kdybychom tak neučinili?

1.2.3 Základní geometrické objekty - bod, přímka

Bod

S objektem Bod jsme se již seznámili a víme, že ho můžeme zadat buď pomoci Nástroje Nový bod  nebo zápisem do Vstupního pole. Například bod (v kartézských souřadnicích) zadáme takto: `A=(1,-8)`.

Operace s body

K bodu můžeme přičíst číslo, které se přičte k oběma souřadnicím bodu. Například zadáme bod bod `A=(1,3)` a vytvoříme bod `B=A+5` a výsledný bod B bude mít souřadnice $(6, 8)$

Dále můžeme bod vynásobit číslem, pak se obě souřadnice tohoto bodu vynásobí daným číslem. Například zadáme bod bod `A=(1,3)` a vytvoříme bod `B=2*A` a výsledný bod bod `B` bude mít souřadnice $(2, 6)$.

Užitečné je také umět pracovat se souřadnicemi bodu, x-ová souřadnice bodu `A` je daná příkazem `x(A)`, analogicky y-ová souřadnice bodu `A` je dána `y(A)`.

Stopa bodu

Ve Vlastnostech bodu lze **Zapnout stopu**, kterou při svém pohybu po nákresně bod zanechává.

Přímka

Přímku lze například vytvořit Nástrojem **Přímka**  .
Je-li přímka dána obecným předpisem, zadáme ji do Vstupního pole `p: 2*x+4*y-8=0` nebo explicitně `p: y=3*x+2`

Souřadné osy x a y jsou pojmenovány jako přímky `OsaX` a `OsaY`.

PŘÍKLAD 4.

1.		Vybereme Nástroj **Přímka**, klikneme kdekoli do Nákresny a vytvoříme postupně dva body A, B a přímku p určenou těmito body. Oba body a přímka se objeví i v Algebraickém okně.
2.		Vybereme Nástroj **Ukazovátko**. Myší klikneme na bod A a budeme-li držet levé tlačítko myši, můžeme bodem hýbat. V **Algebraickém okně** lze vidět, jak se mění souřadnice bodu A i předpis přímky p.

3.		Druhou přímku si vytvoříme pomocí příkazu `Primka`, který dělá totéž co Nástroj Přímka. Do Vstupního pole zadáme definici bodů C, D a přímky procházející těmito body.
		`C=(1,3)`
		`D=(4,-1)`
		`Primka[C,D]`
		Všimněme si, že při zadávání příkazu `Primka` se objeví seznam příkazů začínajících písmeny, které právě píšeme a příkaz si lze šipkami vybrat a není potřeba dopisovat ho celý.

4.		Třetí přímka je určena body A a C a zadáme ji `Primka[A,C]`
5.		Čtvrtou přímku zadáme její obecným předpisem. Přímku $s : 2x - y - 4 = 0$ zadáme příkazem `s:2*x-y-4=0`

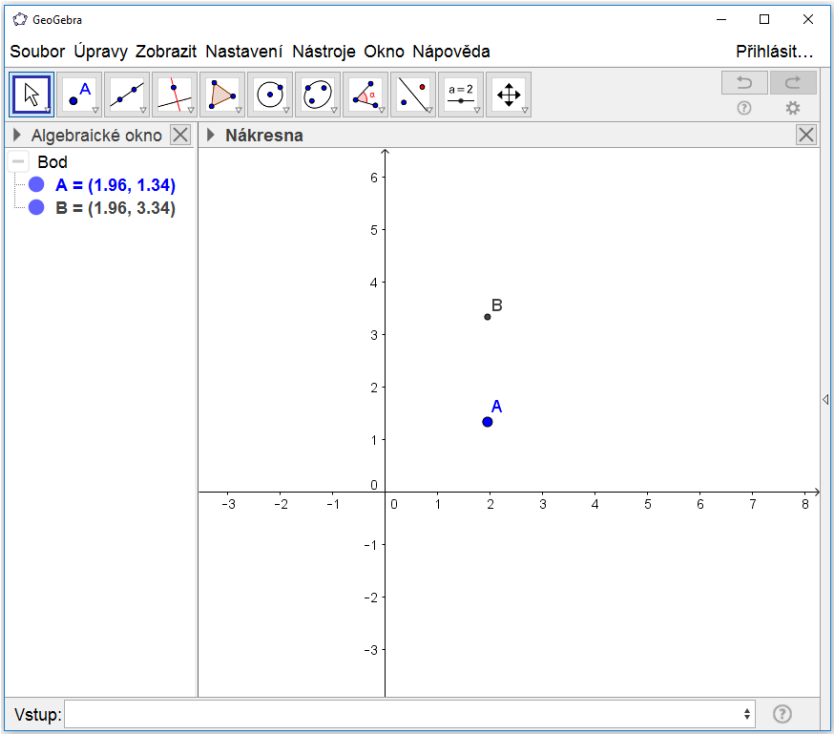
1.2.4 Volné a závislé objekty

PŘÍKLAD 5.

Je dán bod A, vytvoříme bod B tak, že bude mít x-ovou souřadnici stejnou jako bod A a y-ová bude mít o hodnotu 2 větší než je y-ová souřadnice bodu B, viz obrázek 1.10.

1.		Kdekoli do Nákresny (kromě os) zadáme bod A.
2.		Bod B zadáme ze Vstupního pole `B=(x(A),y(A)+2)`
3.		Myší chytneme bod A a pohneme s ním. Co se stane s bodem B? Lze myší chytit i bod B a hýbat s ním?

V příkladě 5 jsme mohli pohnout bodem A, ale nikolik bodem B. Bod A je tzv. **volný objekt**, nezávisí na žádném dalším objektu. Bod B je tzv. **závislý objekt**, neboť závisí na jiném objektu. Rozdíl vidíte i v jejich barvě v Algebraickém okně.



Obrázek 1.10: Příklad volného a závislého bodu

PŘÍKLAD 6.

1.	Vstup:	Zadáme číslo `t=2`
2.	Vstup:	Bod A bude mít x i y souřadnici rovno hodnotě `t`. Tedy ho zadáme `A=(3,t)`
3.		Bodem A teď nelze hýbat myší, neboť je závislý na objektu t.
4.		Můžeme však změnit hodnotu čísla t a tím změnit i souřadnice bodu A. Dvojklikem myší na objekt t v Algebraickém okně můžeme editovat jeho hodnotu. Přepíšeme její hodnotu a stiskneme klávesu *Enter*. Vidíme, že se změnila nejen hodnota čísla t, ale i bodu A.

K PROCVIČENÍ

1. Je dán bod A, vytvořte bod B tak, že

   (a) x-ovou souřadnici bude mít stejnou jako bod A a y-ová bude mít hodnotu 5;

   (b) x-ovou souřadnici bude mít o 3 větší než bod A a y-ová hodnotu -1;

   (c) x-ovou souřadnici bude stejnou jako bod A a bude ležet na ose x.

KAPITOLA

2

MATEMATICKÉ FUNKCE

NAUČÍME SE

Naučíme se zadat matematickou funkci, uvedeme seznam elementárních funkcí a vysvětlíme jak pomoci příkazu `Funkce` zadat matematickou funkci definovanou na intervalu. Seznámíme se s jedním z aktivních prvků GeoGebry posuvníkem a ukážeme si jak pomoci něj spustit animace a vyexportovat ji do formátu *gif*.

OPAKOVÁNÍ MATEMATIKY

Elementární funkce, definiční obor funkce

Definice: Funkce f na množině $D \subset \mathbb{R}$ je zobrazení, které každému číslu $x \in D$ přiřadí právě jedno číslo $y \in \mathbb{R}$. Zapisujeme:

$$f : y = f(x)\,.$$

Poznámka: $y = f(x)$ je **funkční předpis** vyjadřující závislost y na x

x je **nezávislá proměnná** (argument) z definičního oboru

y je **závislá proměnná** z oboru hodnot, vypočítáme ji z funkčního předpisu

Hodnotu funkce f v bodě x_0 označíme $f(x_0) = y_0$ a nazývá se **funkční hodnota funkce** f **v bodě** x_0.

Definice: Množinu D nazveme **definiční obor funkce** f a značíme D_f nebo $D(f)$.

Obor hodnot funkce f je množina všech $y \in \mathbb{R}$, ke kterým existuje aspoň jedno $x \in D(f)$ tak, že $y = f(x)$. Značíme H_f nebo $H(f)$.

Grafem funkce f ve zvolené soustavě souřadnic Oxy je množina všech bodů $[x, f(x)]$, kde $x \in D(f)$.

VÝKLAD UČIVA

2.1 Matematické funkce

2.1.1 Zadání funkce

Matematickou funkci zadáme do vstupního pole pomocí jejího funkčního předpisu. Tedy například funkci $f(x) = sin(x) + x - 1$ zadáme příkazem `sin(x)+x-1`
Program vytvoří objekt funkce a pojmenuje ho v pořadí `f,g,h,...`. Chceme-li funkci pojmenovat sami, zadáme `f(x)=sin(x)+x-1`.
Jak již bylo uvedeno v předchozí kapitole, proměnná `x` je vyhrazena pro x-ovou souřadnici. Funkce může být v jakékoli proměnné, tedy například funkci $s(t) = \frac{1}{t} + 2$ zadáme `s(t)=1/t+2`.
Pokud potřebuje funkci na určitém intervalu, použijem příkaz `Funkce`. Jeho syntaxe je:
`Funkce[ <Funkce>, <Počáteční hodnota>, <Koncová hodnota> ]`

Například funkci $f(x) = x^3$ na intervalu $\langle 0, 5 \rangle$ zadáme příkazem:
`f=Funkce[x^3,0,5]`

2.1.2 Seznam matematických funkcí a konstant

Seznam matematických funkcí

absolutní hodnota $\|x\|$	`abs( )`
druhá odmocnina $\sqrt{x}$	`sqrt( )`
třetí odmocnina $\sqrt[3]{x}$	`cbrt( )`
exponenciální funkce e^x	`exp( )` nebo e `^x`
přirozený logaritmus $\ln(x)$	`ln( )` nebo `log( )`
dekadický logaritmus $\log(x)$	`lg( )` nebo `log(10, )`
logaritmus o základu a $\log_a(x)$	`log(a, )`
sinus $\sin(x)$	`sin( )`
kosinus $\cos(x)$	`cos( )`
tangens $\mathrm{tg}(x)$	`tan( )`
kotangens $\mathrm{cotg}(x)$	`cot( )`
arkussinus $\arcsin(x)$	`asin( )` nebo `arcsin( )`
arkuskosinus $\arccos(x)$	`acos( )` nebo `arccos( )`
arkustangens $\mathrm{arctg}(x)$	`atan( )` nebo `arctan( )`

Konstanty

Ludolfovo číslo $\pi = 3.14\ldots$	π nebo `pi` nebo Alt+p
Eulerovo číslo $e = 2.71\ldots$	e nebo Alt+e
nekonečno ∞	∞ nebo Alt+u
imaginární jednotka $i = \sqrt{-1}$	i nebo Alt+i

PŘÍKLAD 7.

Vytvoříme následující funkce.

1.	Vstup:	$f(x) = \sqrt{x-1}$ `f(x)=sqrt(x-1)`
2.	Vstup:	$nosnost(x) = \cos(x + \frac{\pi}{4})$ `nosnost(x)=cos(x+pi/4)`
3.	Vstup:	$r(x) = \log_3(5x+2)$ `r(x)=log(3,5*x+2)`
4.	Vstup:	$g(t) = 2^t + 7\sin(t)$ `g(t)=2^t+7*sin(t)`
5.	Vstup:	$g(x) = \text{cotg}^2(x-1)$ `g(x)=cot(x-1)^2`
6.	Vstup:	$T(x) = \sqrt{x^2+1}$ $x \in \langle 1, 8 \rangle$ `T=Funkce[sqrt(x^2+1),1,8]`

2.2 Posuvník, animace

2.2.1 Posuvník

Posuvník je grafickou reprezentací čísla a umožní nám pohodlně měnit hodnotu číselné proměnné v daném rozsahu.

Vytvoříme ho pomocí Nástroje **Posuvník** 

.
Nejprve nastavíme jeho rozsah, tj. jeho minimální a maximální hodnotu a krok.
Budeme-li například potřebovat čísla 0, 0.2, 0.4, 0.6, 0.8, ..., 3, nastavíme minimální hodnotu 0, maximální hodnotu 3 a krok 0.2.

PŘÍKLAD 8.

1.		Vytvoříme posuvník t od hodnoty -5 do 5 s krokem 0.1
2.	Vstup:	A bod A bude mít x-ovou i y-ovou souřadnici rovny hodnotě `t`. Tedy ho zadáme `A=(t,t)`
3.		Pohneme myší posuvníkem. Co se děje s bodem A?

PŘÍKLAD 9.

Zadáme funkci $f(x) = a \cdot x$ a budeme měnit hodnotu koeficientu a.

1.		Vytvoříme posuvníky a od hodnoty -5 do hodnoty 5 s krokem 0.1
2.	Vstup:	Zadáme funkce `f(x)=a*x`
3.		Pohneme myší posuvníkem. Co se děje s grafem funkce f?

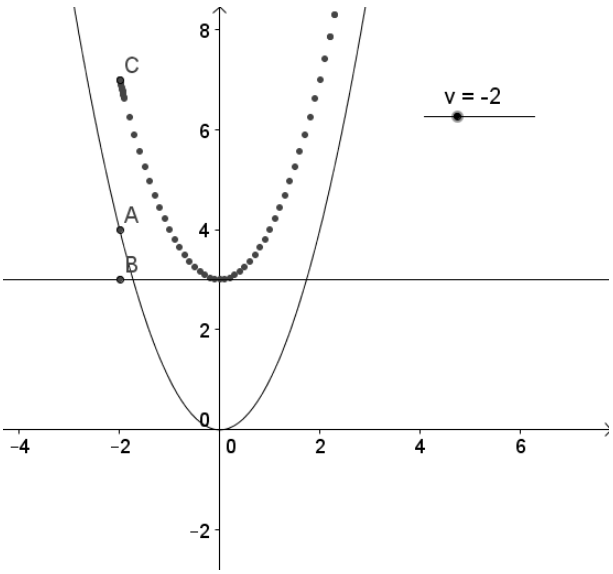
PŘÍKLAD 10.

1.	a = 2	Vytvoříme posuvník v od hodnoty -5 do 5 s krokem 0.1

2.	Vstup:	Zadáme funkci f a bod A, který se bude po grafu funkce pohybovat pomocí posuvníku.
		`f(x)=x^2`
		`A=(v,f(v))`

3.	Vstup:	Zadáme funkci g a bod B, který se bude po grafu funkce pohybovat pomocí posuvníku.
		`g(x)=3`
		`B=(v,g(v))`

4.	Vstup:	Vytvoříme bod C takto `C=(v, f(v)+g(v))`
5.		Ve vlastnostech bodu C zapneme stopu.
6.		Pohneme posuvníkem v. Co vytváří stopa bodu C? Jaký je předpis funkce, kterou tvoří bod C?



Obrázek 2.1: Soucet dvou funkcí.

2.2.2 Animace

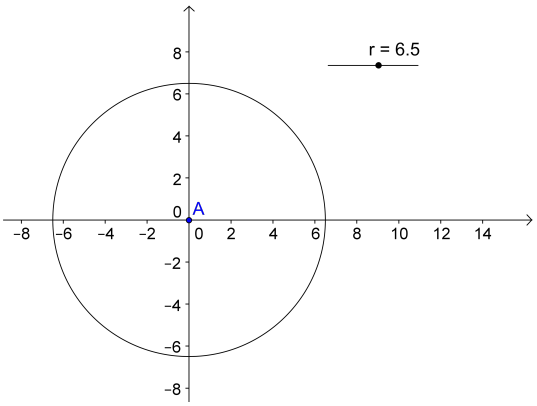
Animaci spustíme ve Vlastnostech posuvníku – **Animace zapnuta**. Postupně se dynamicky mění hodnota posuvníku.
Rychlost animace můžeme nastavit ve Vlastnostech Posuvníku. Například hodnota rychlosti 1 znamená, že celý rozsah proběhne za 10s.

Dále je možno nastavit způsob opakování: **Oscilující** znamené že hodnota posuvníku se mění od minimální po maximální, pak zase zpět k minimální a tak stále dokola. Při nastavení **Rostoucí** běží od minimální po maximální a při volbě **Klesající** naopak. A nakonec lze volbou **Rostoucí (jedenkrát)** zvolit jen jeden cyklus.
Při spuštení animace se objeví v levém dolním rohu nákresny tlačítka na zastavení ▯▯ a spuštění animace ▷.
Animaci lze vyexposrtovat do formátu *gif*. V Menu, položka Soubor, Export, zvolte **Grafický náhled jako animace GIF**.

PŘÍKLAD 11.

Sestavte animaci pro kružnici, jejíž poloměr se bude měnit od hodnoty 2 po hodnotu 10.

1.		Vytvoříme posuvník r od hodnoty 2 do 10 s krokem 0.1
2.		Nástrojem kružnice daná středem a poloměrem vytvoříme kružnici a jako její poloměr zadáme `r`.
3.		Spustíme animaci - pravým tlačítkem na posuvník zvolíme **Animace zapnuta**.



Obrázek 2.2: Animovaná kružnice

2.3 Úloha „Běhající hadi“

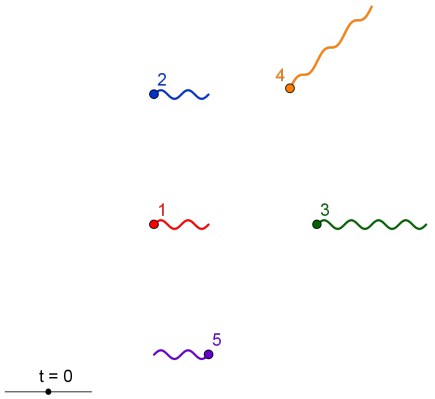
ZADÁNÍ ŘEŠENÉ ÚLOHY

Uděláme animaci s běhajícími hady (obrázek 2.3).

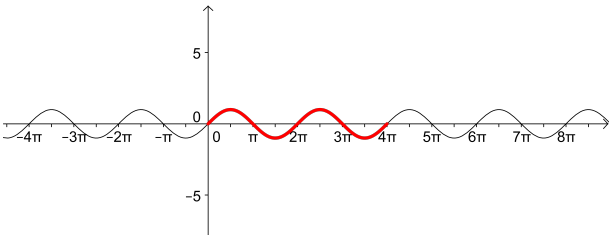
POSTUP ŘEŠENÍ

Rozbor úlohy

Jednotlive křivky (hadi) budeme tvořit pomocí funkce $y = \sin(x)$. První had jsou dvě periody funkce sinus. Zadáme část funkce sinus na intervalu $\langle 0, 4\pi \rangle$ a to příkazem `had=Funkce[sin(x),0,4*pi]`.



Obrázek 2.3: Zadání úlohy „Běhající hadi“

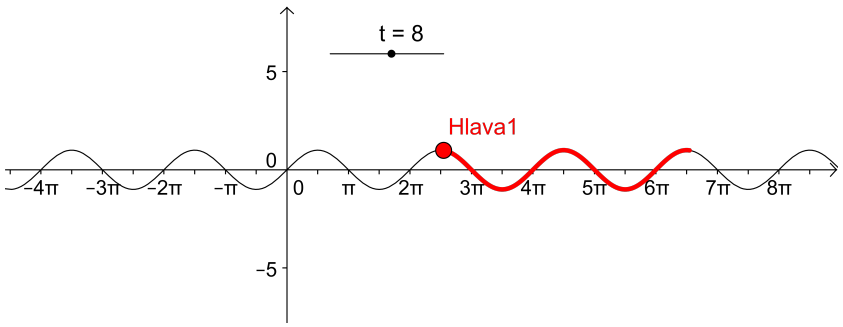


Obrázek 2.4: Příprava na prvního hada

Dále budeme chtít hada „rozhýbat“, tj. budeme měnit jeho definiční obor a tím zobrazovat postupně jiný kus grafu, z čehož vznikne dojem pohybu.

Konstrukce

1.	a = 2	Vytvoříme posuvník t od -100 do 100 s krokem 0.1
2.	Vstup:	Zadáme prvního hada `had1=Funkce[sin(x),t,t+4*pi]` a můžeme mu změnit barvu a zvětšit tloušťku čáry.
3.	Vstup:	Přidáme mu hlavu jako bod na začátku křivky (levým konci) `Hlava1=(t, had1(t))` a můžeme ji změnit na červenou a zvětšit velikost.
4.	Vstup:	Pohneme s posuvníkem *t*
5.		Nastavíme animaci jen jedním směrem, ve Vlastnostech posuvníku zvolíme **Animace** ⇒ **Klesající**.
6.		Animaci spustíme pravým tlačítkem na posuvník – **Animace zapnuta**.



Obrázek 2.5: Had s posuvníkem

Ke kosnstrukci ostatních hadů použijeme stejný posuvník.

7.	Vstup:	Zadáme druhého hada, který leží na grafu funkce $y = \sin(x) + 30$ `had2=Funkce[sin(x)+30,t,t+4*pi]`
8.	Vstup:	Přidáme mu hlavu jako bod na začátku křivky (levým konci) `Hlava2=(t, had2(t))`
9.	Vstup:	Zadáme třetí hada, který leží na grafu funkce $y = \sin(x)$ na intervalu $\langle 12\pi, 20\pi \rangle$. `had3=Funkce[sin(x),t+12*pi,t+20*pi]`
10.	Vstup:	Přidáme mu hlavu jako bod na začátku křivky (levým konci) `Hlava3=(t+12*pi, had3(t+12*pi))`
11.	Vstup:	`had4=Funkce[sin(x)+x,t+10*pi,t+16*pi]` `Hlava4=(t+10*pi,had4(t+10*pi))`
12.	Vstup:	`had5 = Funkce[sin(x)-30,-t,-t+4*pi]` `Hlava5=(-t+4*pi,had5(-t+4*pi))`

K PROCVIČENÍ

1. Zadejte následující funkce:

   (a) $f(x) = \cos^2(x)$

   (b) $g(x) = \cos^2(x)$ na intervalu $\langle 0, 2\pi \rangle$

   (c) $h(x) = e^{x^2}$

   (d) $k(x) = \dfrac{\sqrt{x^2 + x - 2}}{3 - x}$

   (e) $s(t) = \sqrt[3]{4 + \operatorname{cotg}(t)}$

   (f) $h(x) = \ln(x + 4)$ na intervalu $\langle -3, 3 \rangle$

2. Kvadratická funkce má obecný tvar $f(x) = a\,x^2 + b\,x + c$, kde koeficienty $a, b, c \in \mathbb{R}$. V tomto příkladě budeme uvažovat koeficienty z intervalu $\langle -10, 10 \rangle$. Zadejte hodnoty koeficientů a, b, c pomocí posuvníků a funkci $f(x)$.

   Najděte takové hodnoty koeficientů, aby graf kvadratické funkce splňoval:

   (a) Graf je konkávní parabola, která nemá průsečík s osou x.

   (b) Graf je konvexní parabola, která má jeden průsečík s osou x v bodě $x = 2$.

   (c) Graf je parabola, která má průsečík s osou y v bodě $y = 6$ a s osou x v bodech $x = -1$ a $x = 3$.

   (d) Graf je přímka.

3. Sestavte animaci čtverce. Délka jeho strana se bude měnit od hodnoty 1 do hodnoty 5 a zpět, a to tak, že bude:

   (a) jeden roh čtverce pevný (během animace nebude měnit svou polohu);

   (b) střed čtverce pevný;

   (c) střed jedné straný pevný;

4. Zadejte funkci $f(x)$

   (a) $f(x) = \dfrac{5\sin(2x) - \sqrt{3-x}}{1+2^x} + \dfrac{\ln^2(-3x+2)}{-x^3-2}$

   (b) $f(x) = \cos\left(x + \dfrac{\pi}{2}\right)\sin^2(x) - \dfrac{8\left(e^{1-x} + \frac{1}{2}x\right)}{e^x} + 12$

   (c) $f(x) = \operatorname{tg}\left(\dfrac{1}{5}x - 3\right) + \dfrac{2\,|x+1-3^x|}{x^3 - \sqrt{x+3}} + 2$

   (d) $f(x) = \cos(2x - x^2 + 9) + \ln\left(\dfrac{2x+3}{3x+1}\right) + 5x + 10$

   (e) $f(x) = \left(2x^3 - \dfrac{1}{3}x^2 + 14\right) e^{-x+\frac{1}{4}x^2} + \sqrt[3]{x} - 20$

KAPITOLA

3 LOGICKÁ HODNOTA

NAUČÍME SE

Vysvětlíme si, co je to logická (boolovská) hodnota a ukážeme jak se v GeoGebře zadávají relační a logické operace. Seznámíme se s dalšími aktivními prvky: Textovým polem pro vstup a Tlačítkem. Vyzkoušíme si i Nastavení podmínek zobrazení objektu.

OPAKOVÁNÍ MATEMATIKY

Logické operace

A	B	$A \wedge B$	$A \vee B$
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

Konjunkce neboli „a zároveň“ je logická operace, jejíž hodnota je pravda, právě když obě vstupní hodnoty jsou pravda. Označuje se $\wedge$.
Disjunkce neboli „nebo“ je logická operace, jejíž hodnota je pravda, právě když aspoň jedna vstupní hodnota je pravda. Označuje se $\vee$.

VÝKLAD UČIVA

3.1 Logická hodnota

3.1.1 Zadání logické hodnoty

Logická (boolovská) hodnota může mít pouze dvě hodnoty a to buď `true` (pravda, platí) nebo `false` (nepravda, neplatí).

Můžeme ji zadat například `a=true`, tedy do objektu `a` se přiřadí hodnota `true`. Anebo můžeme dostat logickou hodnotu jako výsledek operace. Například `a: x>2`, tedy do `a` se přiřadí `true` pokud je číslo `x` větší než 2 nebo hodnota `false` pokud je číslo `x` menší než 2. Seznam operací, jejíž výsledkem je logická hodnota jsou:

- **rovnost, nerovnost**: pro čísla, body, přímky, kuželosečky a jiné geometrické objekty;
- **porovnání**: je větší, je menší (lze použít pouze pro čísla);
- další operace jsou: množinová operace (je prvkem, používá se pro čísla a seznam čísel) a rovnoběžnost, kolmost (pro dvě přímky).

3.1.2 Relační operátory

Relační operátory jsou operátory, které porovnávají dva objekty a výsledkem je logická hodnota.

Rovnost, nerovnost

operace	výběr	klávesnice	příklad
rovnost	$\stackrel{?}{=}$	==	`a` $\stackrel{?}{=}$ `b` nebo `a == b`
nerovnost	$\neq$	!=	`a` $\neq$ `b` nebo `a != b`

Porovnání hodnot (čísla `a`, `b`)

operace	výběr	klávesnice	příklad
menší než	$<$	$<$	`a` $<$ `b` nebo `a` $<$ `b`
větší než	$>$	$>$	`a` $>$ `b` `a` $>$ `b`
menší nebo roven	$\leq$	$<=$	`a` $\leq$ `b` nebo `a` $<=$ `b`
větší nebo roven	$\geq$	$>=$	`a` $\geq$ `b` nebo `a` $>=$ `b`

PŘÍKLAD 12.

Použití logické hodnoty si zkusíme na jednoduchém příkladě. Zadáme bod A a chceme vědět, zda leží nebo neleží vpravo od osy y, tj. že jeho x-ová souřadnice je kladná. Připomeňme, že v GeoGebře se x-ová souřadnice bodu A zapíše `x(A)`.

1.		Zadáme bod `A` kdekoli v nákresně (kromě os).
2.		Zadáme logickou hodnotu `jevpravo: x(A)>0`
3.		Myší pohneme bodem a sledujeme hodnotu `jevpravo`. Je-li bod A vpravo od osy y má hodnotu `true` a je-li nalevo má hodnotu `false`.

3.1.3 Logické operace

operace	výběr	kláv.	příklad
a (konjunkce)	$\wedge$	&&	`a` $\wedge$ `b` nebo `a && b`
nebo (disjunkce)	$\vee$	||	`a` $\vee$ `b` nebo `a || b`
negace	$\neg$	!	$\neg$ `a` nebo `!a`

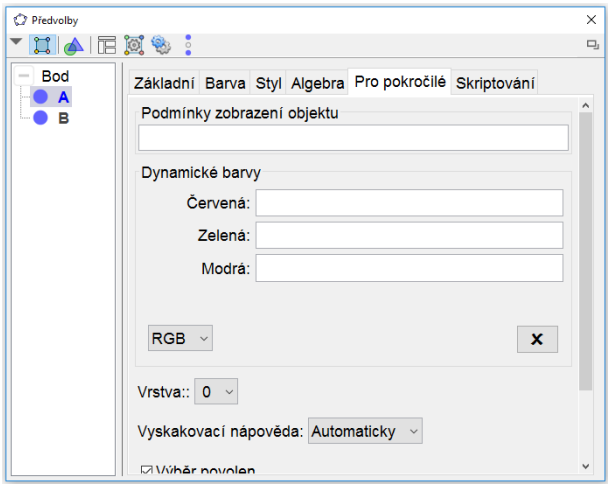
PŘÍKLAD 13.

Zadáme bod B a chceme vědět, zda leží nebo neleží v prvním kvadrantu.

4.		Zadáme bod `B` kdekoli v nákresně (kromě os).
5.		Zadáme logickou hodnotu `jevprvnim: x(B)>0 ∧ y(B)>0`
6.		Myší pohneme bodem a sledujeme hodnotu `jevprvnim`.

3.2 Nastavení podmínek zobrazení

Každému objektu můžeme nastavit podmínky, za kterých bude zobrazen. Ve vlastnostech objektu, záložka **Pro pokročilé** je pole **Podmínky zobrazení objektu**. (viz obrázek 3.1)
Do tohoto pole je nutno zadat logickou hodnotu nebo podmínku, které má logickou hodnotu.



Obrázek 3.1: Podmínky zobrazení objektu

PŘÍKLAD 14.

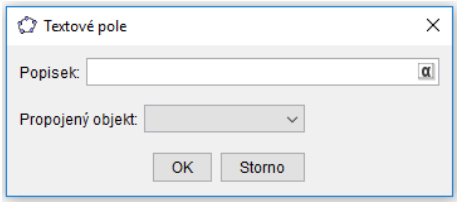
Zadáme bod C, který bude zobrazen pouze leží-li v prvním kvadrantu.

7.		Zadáme bod C kdekoli v nákresně (kromě os).
8.		Ve Vlastnostech bodu C v záložce Pro pokročilé nastavíme Podmínku zobrazení objektu `x(C)>0 ∧ y(C)>0`
9.		Myší pohneme bodem, a sledujeme bod.

3.3 Aktivní objekty

3.3.1 Vložit textové pole

Nástrojem  **Vložit textové pole** můžeme zadávat hodnoty daného objektu pomocí pole. Popisek je jen text, který bude zobrazen u textového pole. Důležitý je výběr **Propojený objekt**, kde z již existujících objektů vybereme ten, jehož hodnotu budeme chtít pomocí pole měnit.



Obrázek 3.2: Vložit textové pole

PŘÍKLAD 15.

Do Textového pole se bude zadávat předpis funkce.

1.		Zadáme funkci f, například: `f(x)=sin(x)`
2.		Vložíme Textové pole s popisem *f(x)*= a propojíme s objektem `f(x)`.
3.		Do textového pole zadáme předpis jiné funkce, například `x^2`.

3.3.2 Tlačítko

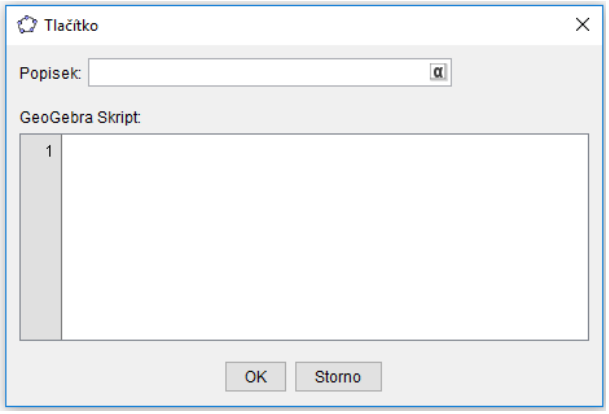
Nástroj Tlačítko  umožní vytvořit tlačítko, pomocí kterého lze spustit tzv. GeoGebraSkript. Tlačítko má také Popisek, což je text, který na něm bude zobrazen.

Skripty

Skript je posloupnost příkazů, které se jeden za druhým vykonají.
Je možno zadat buď GeoGebraSkript nebo JavaSkript.
GeoGebraskript se spustí buď na kliknutí nebo po aktualizace hodnot. Příkazy se zapisují stejně jako do Vstupního pole, po jednom na řadek.

Náhodné číslo

Náhodné celé číslo ležící mezi a a b se vygeneruje příkazem `NahodneMezi[a,b]` Tedy například náhodné celé číslo mezi 3 a 10: `NahodneMezi[3,10]`

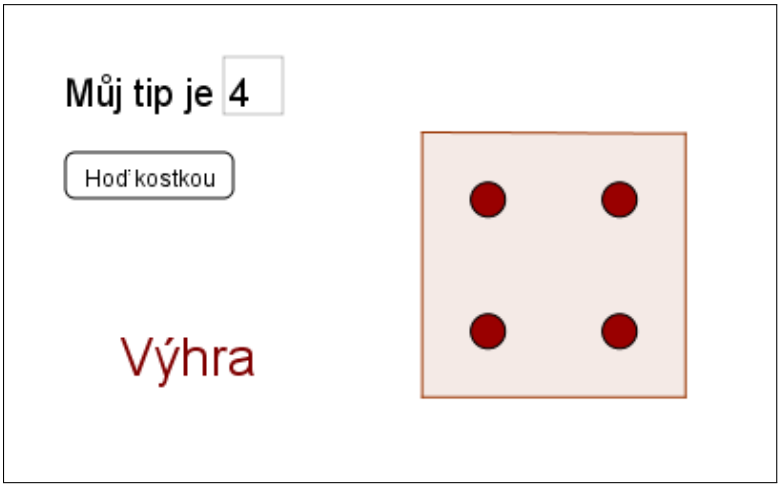


Obrázek 3.3: Tlačítko

3.4 Úloha „Hrací kostka“

Zadání řešené úlohy

Vytvoříme simulaci hrací kostky. Na posuvníku můžeme volit hodnoty $1, 2, \ldots, 6$ a na kostce se obrazí příslušný počet ok. Hráč může zadat do textového pole svůj tip a stisknout tlačítko *Hoď kostkou*. Na obrázku 3.4 je zobrazena výsledná simulace.

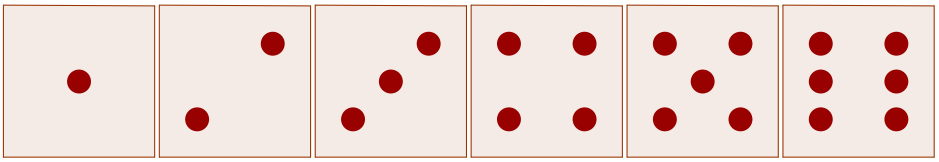


Obrázek 3.4: Hrací kostka

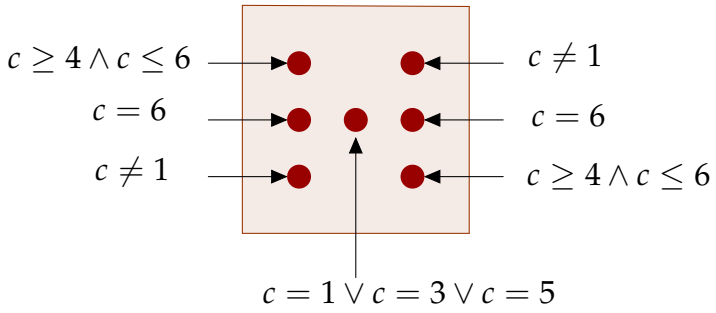
Postup řešení

Rozbor

Nakreslíme si jak vypadají jednotlivá oka na kostkách.
Do jednoho čtverce zakreslíme všechna oka, která budeme potřebovat k zobrazení jednotlivých čísel na kostce.
Simulaci hrací kostky uděláme pomocí sedmi bodů, která reprezentují jednotlivá oka kostky.
Tvar kostky vytvoříme pomocí čtverce okolo těchto bodů.



Obrázek 3.5: Oka na kostce pro všechna čísla



Obrázek 3.6: Podmínky pro zobrazení jednotlivých ok.

Konstrukce

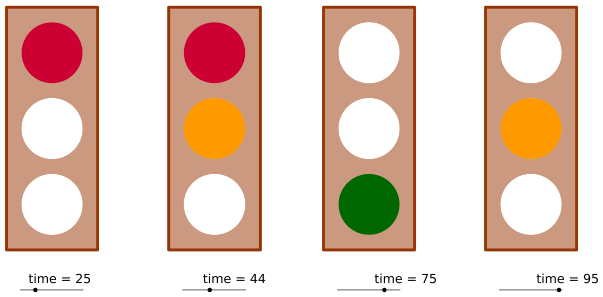
1.		Zadáme všech sedm bodů a ve Vlastnostech zvětšíme jejich velikost.
2.		Okolo bodů vytvoříme čtverec.
3.		Uděláme posuvník c od 1 do 6 s krokem 1.
4.		Jednotlivým bodům ve Vlastnostech nastavíme Podmínky zobrazení podle obrázku 3.6.

Nyní ke hře přidáme tlačítko „Hoď kostkou“, po jehož stisknutí se vygeneruje číslo a to se nastaví na kostce. Dále přidáme textové pole, do kterého hráč napíše svůj tip, jaké číslo padne.

5.	Vstup:	Zadám e svůj tip, například `tip=1`
6.	a = 1	Vložíme textové pole, kterému dáme Popisek `tip=` a propojíme ho s objektem `tip`. Ve Vlastostech - Styl můžeme změnit velikost pole.
7.	OK	Vložíme tlačítko s Popisem „Hoď kostkou“ a skriptem `c=NahodneMezi[1,6]`
8.	ABC	Vložíme do nákresny text *Výhra* a v jeho Vlastnostech, v Podmínkách zobrzení nastavíme podmínku `c==tip`
9.		Skryjeme posuvník *c* a zavřeme Algebraické okno. Hru zkusíme.

K PROCVIČENÍ

1. Semafor má cyklus trvající 100 s, který začíná červenou, která svítí 40 s, pak na 20 s svítí červená s oranžovou. Následuje 30 s zelená a nakonec 10 s oranžová. Vytvoříme tři barevné kružnice jako semafor a pomoci posuvníku nasimulujeme jeho cyklus.



Obrázek 3.7: Semafor.

KAPITOLA

4

ROVNICE A NEROVNICE

NAUČÍME SE

Vysvětlíme si jak se řeší rovnice a nerovnice, dále si ukážeme jak popsat množinu bodů v rovině.

VÝKLAD UČIVA

Vyřešení rovnice $f(x) = 0$ je ekvivalentní nalezení nulových bodů (průsečíků s osou x) funkce $f(x)$. GeoGebra má pro hledání nulových bodů příkaz `NuloveBody`.
Je-li funkce polynom, lze jej použít takto: `NuloveBody[<Funkce>]`.

V případě, že je funkcí složitější je potřeba z grafu vybrat interval, na kterém nulový bod leží a spočítat ho:

`NuloveBody[ <Funkce>, <Počáteční hodnota x>, <Koncová hodnota x> ]`

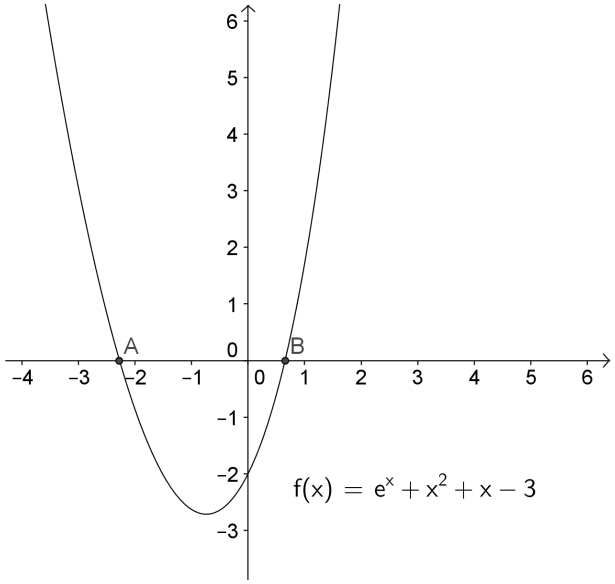
`<Funkce>`	funkce
`<Počáteční hodnota x>`	krajní meze intervalu, ve kterém leží nulový bod
`<Koncová hodnota x>`	

4.1 Řešení rovnic

PŘÍKLAD 16.

Najdeme řešení rovnice $e^x + x^2 + x - 3 = 0$.

1.	Vstup:	Zadáme funkci `f(x)=e^x +x^2 + x-3`.
2.	Vstup:	Jeden nulový bod funkce f najdeme příkazem `NuloveBody[f,-3,-2]`. A vidíme, že výsledkem je bod B, jehož x-ová souřadnice je -2.27 a to přibližná hodnota jednoho z řešení rovnice.
3.	Vstup:	Druhé řešení najdeme na intervalu $\langle 0, 1 \rangle$.



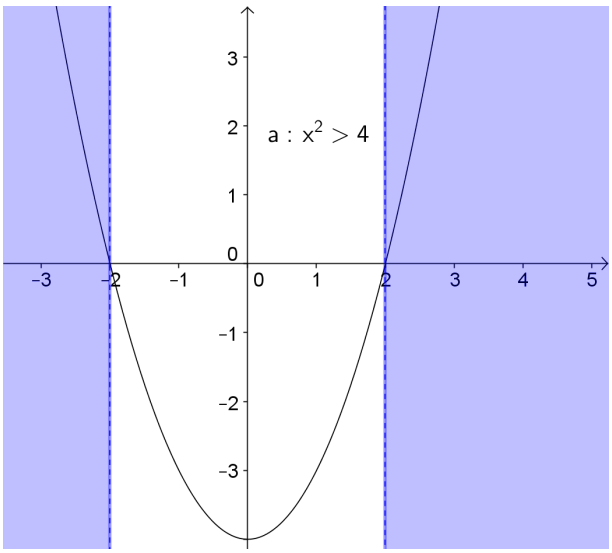
Obrázek 4.1: Řešení rovnice

4.2 Nerovnice

Řešení nerovnic si ukážeme v souvislosti s grafy funkcí a jejich nulovými body.

PŘÍKLAD 17.

1.	Vstup:	Zadáme `x^2>4` což je kvadratická nerovnice, jejíž řešení se vykreslí.
2.	Vstup:	Zadáme funkci `f(x)=x^2-4` což je kvadratická funkce, jejíž graf (parabola) se vykreslí.
3.		Vidíme, že tam kde je parabola nad grafem je rovnice splňena. Tedy na intervalech $(\infty, -2) \cup (2, \infty)$.



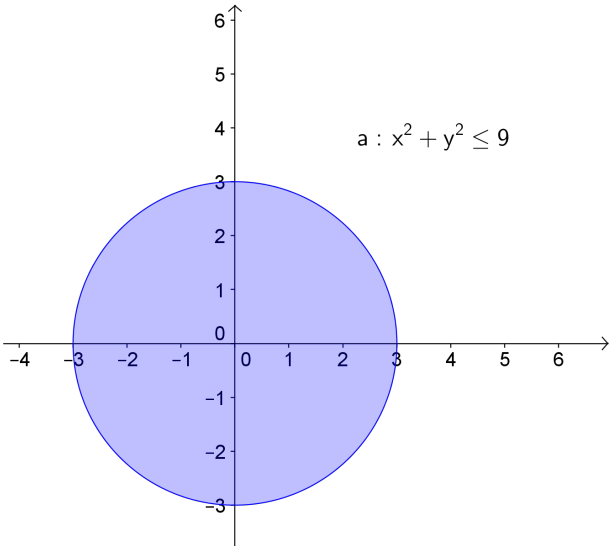
Obrázek 4.2: Nerovnice $x^2 > 4$

4.3 Objekt Nerovnost

V Geogebře lze vytvořit objekt typu **Nerovnost**, který se graficky zobrazí jako množina bodů v rovině.

PŘÍKLAD 18.

1.	Vstup:	Zadáme `x^2+y^2<=9` což je kruh se středem v počátku a poloměrem 3.

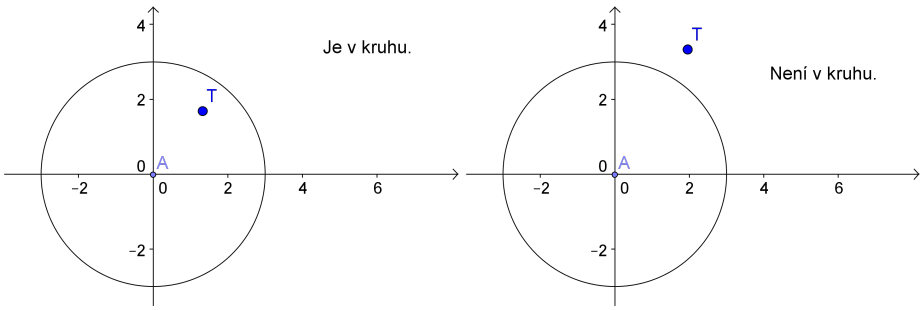


Obrázek 4.3: Nerovnost

PŘÍKLAD 19.

Vytvoříme kružnici, bod T a text, který bude oznamovat, zda bod T leží nebo neleží uvnitř kruhu. Připomeňme, že rovnice kružnice je $(x - x_s)^2 + (y - y_s)^2 = r^2$, kde (x_s, y_s) je střed a r poloměr kružnice.

1.		Vytvoříme kružnici se středem v počátku souřadnic a poloměrem 3 a bod T.
2.		A kdekoli (kromě os) bod T.
3.		Vložíme text `Je v kruhu` a v Pro Pokročilé Podmínky zobrazení zapíšeme `x(T)^2+y(T)^2<=9`
4.		Pohneme bodem T.
5.		Sami udělejte text `Není v kruhu.`, který se zobrazí bude-li bod mimo kruh.



Obrázek 4.4:

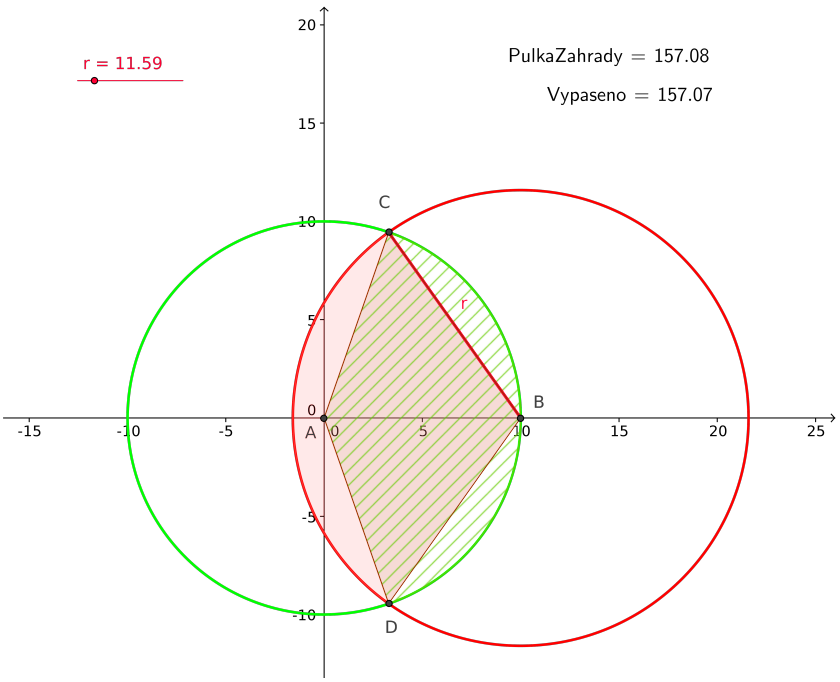
4.4 Nákresna 2

V Geogebře můžeme používat dvě nákresny. Druhou zapneme v menu **Zobrazit - Nákresna 2**. Objekt se zobrazí na právě aktivní nákresnu (vybereme myší). Výběr nákresny, na které má být objekt zobrazen, je možný také ve **Vlastnostech - Pro pokročilé - Umístění**.

4.5 Úloha „Ovce na pastvě“

ZADÁNÍ ŘEŠENÉ ÚLOHY

Máme kruhovou zahradu o poloměru R a na její okraj přivážeme ovci. Chceme určit takovou délku provazu, aby vypásla jen půlku obsahu zahrady.



Obrázek 4.5: Úloha „Ovce na pastvě“

POSTUP ŘEŠENÍ

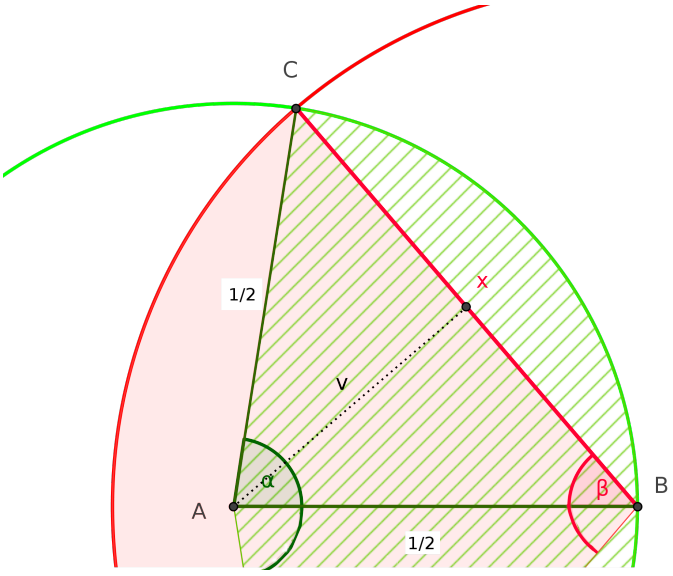
Konstrukce

1.		Vložíme posuvník R (poloměr zahrady) od 0 do 10 s krokem 0.1.
2.		Uděláme kružnici se středem v počátku $(0,0)$ a poloměrem R. Kružnici se přejmenujeme na c.
3.		Vložíme posuvník r (poloměr provazu) od R do $2 * R$ s krokem 0.01.
4.		Vložíme bod na objektu, tedy na kružnici c. Bod se pojmenujeme B.
5.		Uděláme kružnici se středem v bodě B a poloměrem r. Kružnice si pojmenujeme d.
6.		Najdeme průsečíku kružnic. Body si pojmenujeme C, D.
7.		Vytvoříme kruhovou výseč určenou středem A a dvěma body C, D. Body označujeme v pořadí: střed a body na oblouku v kladném směru (proti směru hodinových ručiček). Výseč přejmenuje `VysecZahrada`.
8.		Vytvoříme kruhovou výseč určenou středem B a dvěma body C, D. Výseč přejmenuje `VysecOvce`

9.		Vytvoříme mnohoúhleník určený body A,D,B,C a přejmenujeme na `Prunik`.
10.		Do vstupního pole zadáme obsah vypasené oblasti `Vypaseno=VysecZahrada+VysecOvce-Prunik` a myší lze přetáhnout objekt `Vypaseno` z Algebraického okna do Nákresny, kde se zobrazí jeho název a hodnota.
11.		Do vstupního pole zadáme obsah půlky zahrady: `PulkaZahrady=1/2*pi*R^2` a opět myší přetáhneme objekt `PulkaZahrady`.
12.		Změnou hodnoty posuvníku r vidíme změnu hodnot `Vypaseno` a můžeme najít hodnotu rovnu hodnotě `PulkaZahrady`

Rozbor

Úlohu jsme popsali geometricky a „zkusmo“ jsme našli řešení. Nyní se na problém podíváme podrobněji. Budeme počítat s poloměrem zahrady $R = \frac{1}{2}$.
Půlka obsahu zahrady bude $S = \frac{1}{2}\pi R^2 = \frac{\pi}{8}$
Označme jako x hledanou délku provazu.



Obrázek 4.6: Úloha „Ovce na pastvě“ - rozbor

Podíváme se na trojúhelník ABC a lehce zjistíme, že platí:

$$\sin\frac{\alpha}{4} = x \quad \Rightarrow \quad \alpha = 4\arcsin(x)$$
$$\cos\frac{\beta}{2} = x \quad \Rightarrow \quad \beta = 2\arccos(x)$$

Výška v rovnoramenném trojúhelníku se spočítá z Pythagorovy věty:

$$v = \sqrt{\left(\frac{1}{2}\right)^2 - \left(\frac{x}{2}\right)^2} = \frac{\sqrt{1-x^2}}{2}$$

Obsahy kruhové vyseče se spočítá:

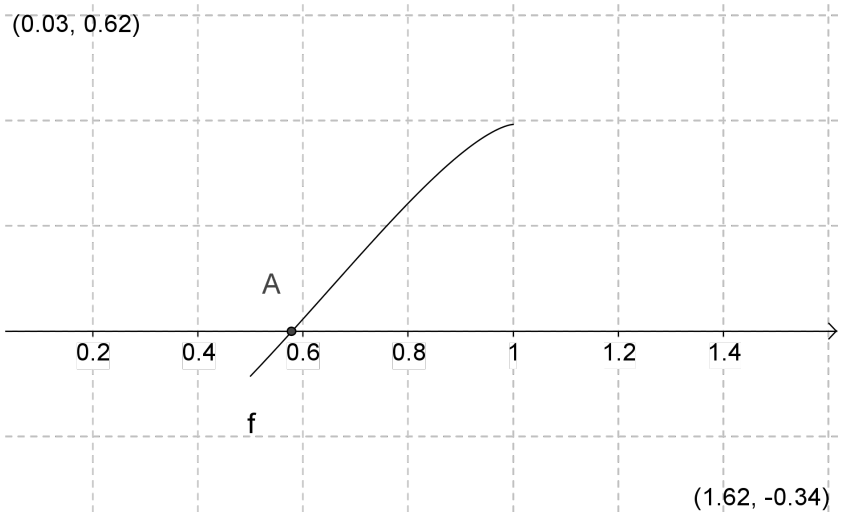
$$\mathrm{VysecZ}(x) = \frac{1}{2}\left(\frac{1}{2}\right)^2 \alpha = \frac{1}{2}\arcsin(x)$$
$$\mathrm{VysecO}(x) = \frac{1}{2}x^2\beta = x^2\arccos(x)$$
$$\mathrm{Prunik}(x) = x\,v = \frac{x\sqrt{1-x^2}}{2}$$

Hledáme $x \in \langle \frac{1}{2}, 1\rangle$ *jako řešení rovnice:*

$$\frac{1}{2}\arcsin(x) + x^2\arccos(x) - \frac{x\sqrt{1-x^2}}{2} = \frac{\pi}{8}$$

Rovnici pro hodnotu $R = \frac{1}{2}$ vyřešíme pomoci příkazu `NuloveBody`.

13.		Nastavme posuvník R na hodnotu 0.5
14.		Zadáme funkci `f(x)=1/2*asin(x)+x^2*acos(x)-1/2*x*sqrt(1-x^2)-pi/8`
15.		A rovnici vyřešíme na příslušném intervalu. `NuloveBody[f,1/2,1]`

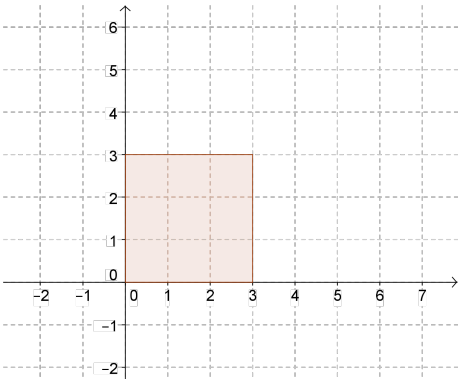


Obrázek 4.7: Úloha „Ovce na pastvě“ – funkce

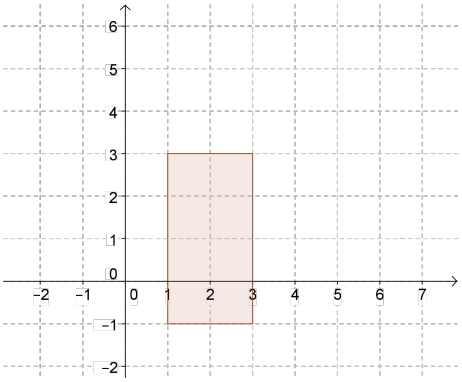
K PROCVIČENÍ

1. Pomoci nerovností popište danou oblast, bod T a text, který bude oznamovat, zda bod T leží nebo neleží uvnitř oblasti.

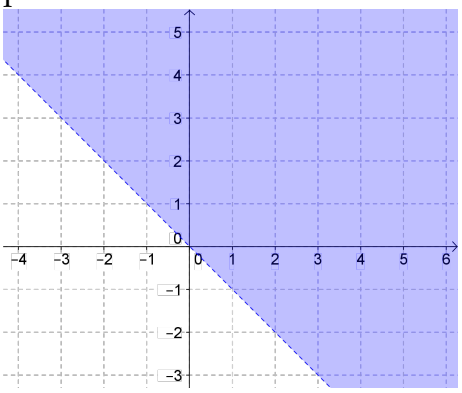
(a) čtverec



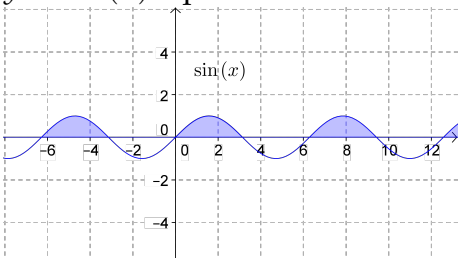
(b) obdélník



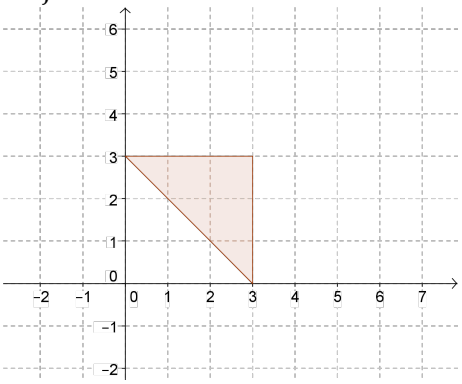
(c) polorovina



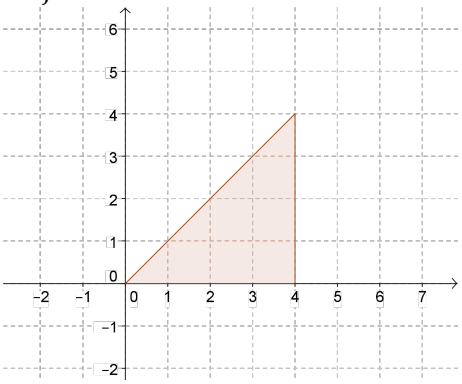
(d) oblast ohraničená $y = \sin(x)$ a přímkami



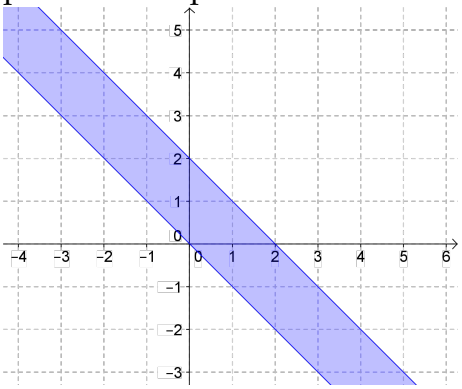
(e) trojúhelník



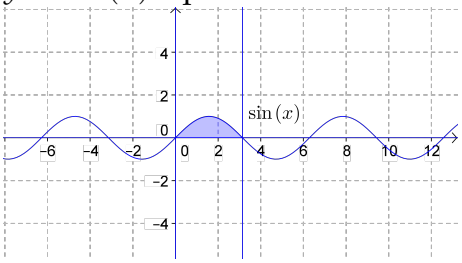
(f) trojúhelník



(g) průnik dvou polorovin



(h) oblast ohraničená $y = \sin(x)$ a přímkami



KAPITOLA

5

TEČNA A DERIVACE

NAUČÍME SE

Ukážeme si jak sestrojit tečnu, spočítat derivaci a druhou derivaci. Vytvoříme Taylorův polynom a naučíme se určovat lokální extrémy a inflexní body funkce.

OPAKOVÁNÍ MATEMATIKY

Derivace funkce

Definice : Je dána funkce f a bod $x_0 \in D_f$. Existuje-li vlastní limita

$$\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

pak ji nazveme **derivace funkce** f **v bodě** x_0 a značíme ji $f'(x_0)$.

Definice : Funkce f je definována v každém bodě intervalu (a, b) a má v každém bodě derivaci $f'(x)$. Pak je na (a, b) definovaná funkce f', která každému $x \in (a, b)$ přiřadí hodnotu $f'(x)$. Tuto funkci nazveme **derivace** funkce f. Značíme $f'(x)$, y' $\frac{df(x)}{dx}$, $\frac{dy}{dx}$.

Derivace vyšších řádů

Definice : Nechť má funkce $f(x)$ derivaci v intervalu I. Pak funkci $[f'(x)]'$ nazveme **druhou derivaci funkce** a značíme $f''(x)$.

Poznámka: Obdobně definujeme derivaci n-tého řádu

$$f^{(n)}(x) = \left[f^{(n-1)}(x)\right]' .$$

Tečna ke grafu funkce

Definice : Nechť má funkce $f(x)$ v bodě x_0 derivaci. Přímku t, procházející bodem $[x_0; f(x_0)]$ a mající směrnici rovnu hodnotě derivace v x_0 nazveme **tečna ke grafu funkce** $f(x)$ **v bodě** x_0.
Přímku n, procházející bodem $[x_0; f(x_0)]$ a kolmou k tečně nazveme **normála ke grafu funkce** $f(x)$ **v bodě** x_0.

Věta : *Tečna ke grafu funkce $f(x)$ v bodě x_0 je daná předpisem:*

$$t: \; y - f(x_0) = f'(x_0) \cdot (x - x_0)$$

Normála ke grafu funkce $f(x)$ v bodě x_0 je daná předpisem:

$$n: \; y - f(x_0) = -\frac{1}{f'(x_0)} \cdot (x - x_0)$$

Poznámka: V bodě ve kterém nemá funkce derivaci tečna neexistuje.

Taylorův polynom

Definice : Nechť je dána funkce $f(x)$, která má v bodě $x_0 \in D_f$ derivace až do řádu $n \in \mathbb{N}$. Pak polynom:

$$T_n(x) = f(x_0) + \frac{f'(x_)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

nazveme **Taylorův polynom funkce** $f(x)$ **stupně** n **v bodě** x_0.

Poznámka: Taylorův polynom prvního stupně je tečna.

Rostoucí a klesající funkce, lokální extrémy funkce

Definice : Funkce f se nazývá **rostoucí na intervalu** I, právě když pro všechna $x_1, x_2 \in I$ platí: je-li $x_1 < x_2$ pak $f(x_1) < f(x_2)$.

Definice : Funkce f se nazývá **klesající na intervalu** I, právě když pro všechna $x_1, x_2 \in I$ platí: je-li $x_1 < x_2$ pak $f(x_1) > f(x_2)$.

Věta : *Nechť je funkce $f(x)$ definována na intervalu I a platí-li na tomto intervalu $f'(x) > 0$, pak je funkce $f(x)$ na tomto intervalu rostoucí. Platí-li $f'(x) < 0$, pak je funkce klesající.*

Definice : Řekneme, že funkce $f(x)$ má **v bodě** x_0 **lokální maximum**, existuje-li takové okolí bodu x_0, že pro všechna $x \neq x_0$ z tohoto okolí platí $f(x) \leq f(x_0)$. Platí-li $f(x) < f(x_0)$, pak řekneme, že má **ostré lokální maximum**.

Definice : Řekneme, že funkce $f(x)$ má **v bodě** x_0 **lokální minimum**, existuje-li takové okolí bodu x_0, že pro všechna $x \neq x_0$ z tohoto okolí platí $f(x) \geq f(x_0)$. Platí-li $f(x) > f(x_0)$, pak řekneme, že má **ostré lokální minimum**.

Věta : ***(nutná podmínka existence lokálního extrému)***
Má-li funkce $f(x)$ v bodě x_0 lokální extrém a existuje-li v tomto bodě derivace. Pak platí:

$$f'(x_0) = 0\,.$$

Poznámka: Funkce může mít lokální extrém pouze v bodech, ve kterých buď neexistuje derivace nebo je derivace rovna nule.

Konvexní a konkávní funkce, inflexní body

Definice : Necht má funkce $f(x)$ derivaci v bodě x_0. řeknem, že **funkce $f(x)$ je v bodě x_0 konvexní (resp. konkávní)**, jestliže existuje okolí bodu takové, že pro všechny x z tohoto okolí je graf funkce nad (resp. pod) tečnou:

$$f(x) > f(x_0) + f'(x_0)(x - x_0)$$

resp.

$$f(x) < f(x_0) + f'(x_0)(x - x_0)$$

Definice : Řeknem, že **funkce $f(x)$ je konvexní (resp. konkávní) v intervalu $I \subset D_f$.** Jestliže je konvexní (resp. konkávní) v každém bodě intervalu I.

Věta : *Nechť $f''(x_0) > 0$, pak je $f(x)$ v bodě x_0 konvexní. Nechť $f''(x_0) < 0$, pak je $f(x)$ v bodě x_0 konkávní.*

Definice : Nechť má funkce $f(x)$ derivaci v bodě x_0. Přechází-li graf funkce v bodě x_0 z polohy pod tečnou do polohy nad tečnou (nebo naopak) nazveme bod x_0 **inflexním bodem** funkce $f(x)$.

Věta : ***(nutná podmínka existence inflexního bodu)***
Je-li x_0 inflexní bod funkce $f(x)$ a má-li $f(x)$ v tomto bodě druhou derivaci, pak

$$f''(x_0) = 0\,.$$

Poznámka: Funkce může mít inflexi pouze v bodech, ve kterých buď neexistuje druhá derivace nebo je druhá derivace rovna nule.

VÝKLAD UČIVA

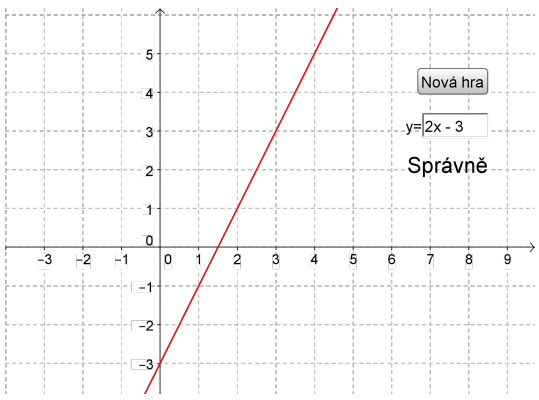
5.1 Směrnice přímky

PŘÍKLAD 20.

Hra, ve které hráč musí poznat předpis lineární funkce $y = kx + q$. Svůj tip napíše do textového pole, a objeví se nápis *Správně* v případě, že předpis určíte správně. Hra také obsahuje tlačítko na vytvoření nové náhodné funkce.

1.		Vytvoříme tlačítko na vygenerování náhodné lineární funkce. Do pole Popis napíšeme *Nová hra* a do pole Skript příkaz: `f(x)=NahodneMezi[-3,3]*x+NahodneMezi[-3,3]`
2.		Několikrát si vyzkoušejte tlačítko. Na kliknutí se vygeneruje lineární funkce. Poznáte její předpis? Pro usnadnění si zapněte Mřížku (Pravým Tlačítkem do Nákresny, zapnout **Mřížku**).
3.		Svůj tip na předpis funkce zapíšeme do Vstupního pole , tedy například `tip(x)=2*x-3`
4.		Pro pohodlnější zápis našeho tipu vložíme Textové pole. Jako popisek napíšeme `y=` a propojíme ho s objektem `tip(x)`.
5.		Vložíme do Nákresny Text *Správně*.
6.		Chceme aby se text objevil pouze v případě, že jsme uhodli, tady ve Vlastnostech textu Správně v záložce Pro pokročilé nastavíme Podmínky zobrazení `f==tip`
7.		Nakonec skryjeme objekt tip a zavřeme Algebraické okno a hru zkusíme.

Funkce má předpis $y = kx + q$ a víme, že hodnota q je rovna hodnotě průsečíku s osou y. Jak poznáme hodnotu směrnice k?



Obrázek 5.1: Poznej lineární funkci

5.2 Tečna funkce

Tečnu ke grafu funkce sestrojíme v GeoGebře pomoci příkazu `Tecna`

`Tecna[<Bod>, <Funkce> ]`

`<Bod>`	bod na funkci
`<Funkce>`	funkce

`Tecna[ <Hodnota x>, <Funkce> ]`

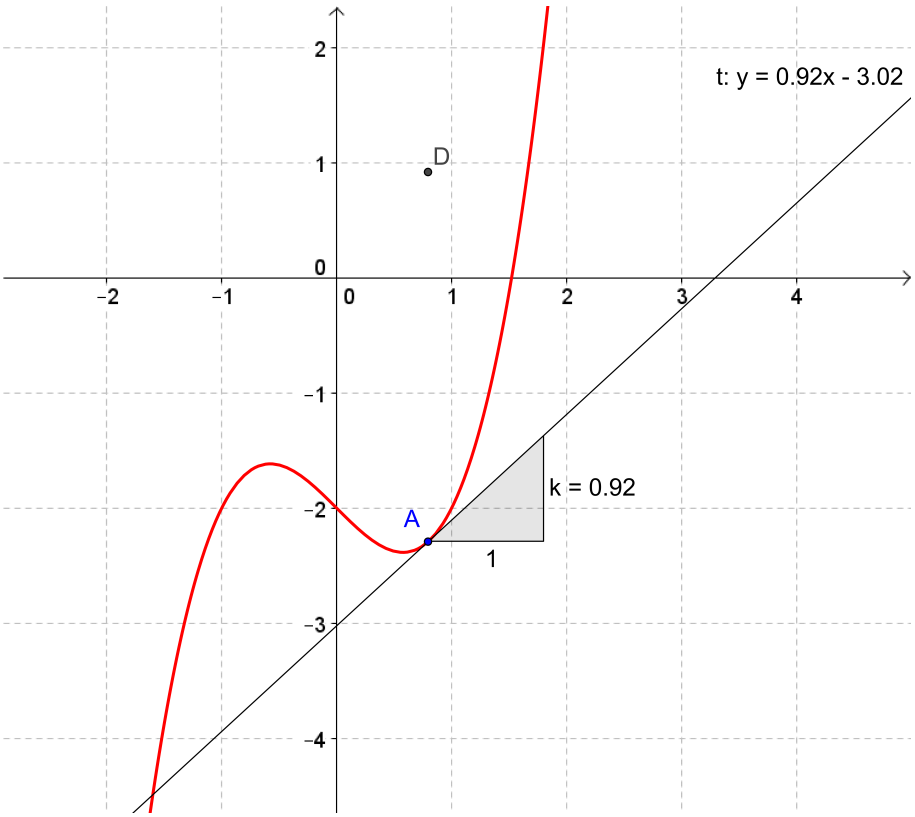
`<Hodnota x>`	hodnota
`<Funkce>`	funkce

PŘÍKLAD 21.

Jaká je směrnice tečny v bodě?
Na graf funkce $f(x) = x^3 - x - 2$ umístíme bod A a v tomto bodě sestrojíme tečnu. Určíme si hodnotu směrnice této tečny a v každém bodě si ji vykreslíme. Tedy vytvoříme bod, jehož x-ová souřadnice bude stejná jako bodu A a y-ová souřadnice bude mít hodnotu směrnice tečny v bodě A.

1.	Vstup:	Zadáme funkci `f(x)=x^3-x-2`
2.		Na graf funkce f umístíme bod A.
3.	Vstup:	Spočítáme tečnu v bodě A `t=Tecna[A,f]`
4.	Vstup:	A spočítáme směrnici tečny `k=Smernice[t]`
5.	Vstup:	Zadáme bod, jehož x-ová souřadnice bude stejná jako bodu A a y-ová bude rovna hodnotě směrnice tečny v bodě A. `D = (x(A), k)` A ve vlastnostech bodu D zapneme stopu **Stopa zapnuta**.
6.		Myší pohneme bodem A.

Stopa bodu D vykreslila graf funkce, která je derivací funkce f.



Obrázek 5.2: Směrnice tečny

5.3 Derivace funkce

Derivaci funkce spočítáme příkazem `Derivace`:

`Derivace[ <Funkce> ]`

Program derivaci označí názvem funkce s apostrofem. Například pro funkce `g(x)=cos(x)` spočítáme derivaci příkazem Derivace[g] a vznikne objekt s názvem `g'`.

Derivace n-tého řádu funkce f se spočítá:

`Derivace[ <Funkce>, <Číslo> ]`

`<Funkce>`	funkce
`<Číslo>`	řád derivace n

5.4 Taylorův polynom

Taylorův polynom k funkci f v bodě x_0 řádu n se spočítá:

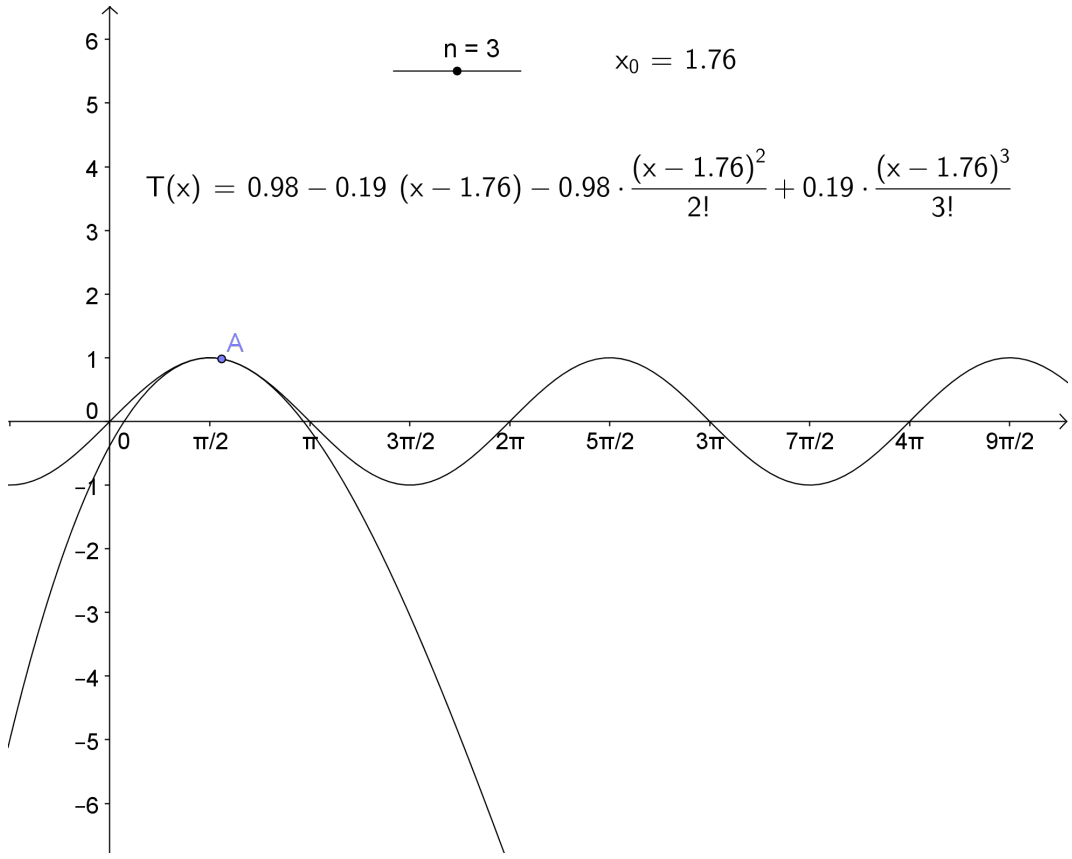
`TaylorovaRada[ <Funkce>, <Hodnota x>, <Číslo> ]`

`<Funkce>`	funkce
`<Hodnota x>`	bod x_0, ve kterém se Taylorův polynom počítá
`<Číslo>`	řád n

PŘÍKLAD 22.

Sestrojte Taylorův polynom k funkci $f(x) = \sin(x)$. Řád lze měnit od 1 do 5 a bod po celém definičním oboru.

1.	Vstup:	Zadáme funkci `f(x)=sin(x)`
2.	A	Na graf funkce umístíme bod A.
3.	Vstup:	K výpočtu Taylorova polynomu potřebujeme x-ovou souřadnici bodu A `x_0=x(A)`
4.	a = 2	Vytvoříme posuvník n od 1 do 5 s krokem 1.
5.	Vstup:	Sestrojíme Taylorův polynom `T=TaylorovaRada[f,x_0,n]` Zkuste měnit řád polynomu n nebo pohnout bodem A.



Obrázek 5.3: Taylorův polynom

5.5 Rostoucí a klesající

Připomeňme, že má-li funkce derivaci v bodě kladnou, pak je tomto bodě rostoucí. A má-li derivaci zápornou, je klesající.

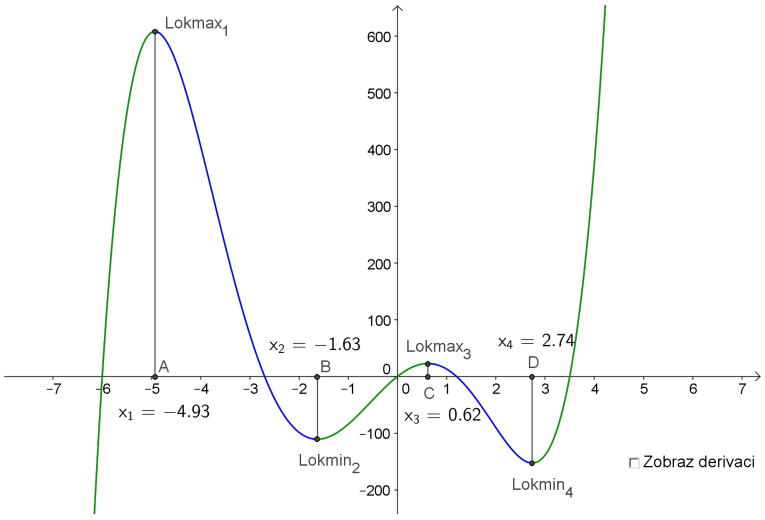
PŘÍKLAD 23.

Graf funkce $f(x) = 100x^5 + 400x^4 - 2049x^3 - 3960x^2 + 6804x$ zobrazte tak, aby rostoucí úseky grafu byly zobrazeny zeleně a klesající úseky modře.

1.	Vstup:	`f(x)=100*x^5+400*x^4-2049*x^3-3960*x^2+6804*x`
2.	Vstup:	`Derivace[f]`
3.	Vstup:	`NuloveBody[f']`
4.	Vstup:	`x_1=x(A)  x_2=x(B)  x_3=x(C)  x_4=x(D)`

5.	Vstup:	`f1rost=Funkce[f,-∞,x_1]`
		`f2kles=Funkce[f,x_1,x_2]`
		`f3rost=Funkce[f,x_2,x_3]`
		`f4kles=Funkce[f,x_3,x_4]`
		`f5rost=Funkce[f,x_4,∞]`

6.		Funkcím `f1rost, f3rost, f5rost` změníme barvu na zelenou a Funkcím `f2kles, f4kles` na modrou.



Obrázek 5.4: Rostoucí a klesající funkce

5.6 Konvexní a konkávní

Připomeňme, že má-li funkce druhou derivaci v bodě kladnou, pak je tomto bodě konvexní. A má-li druhou derivaci zápornou, je konkávní.

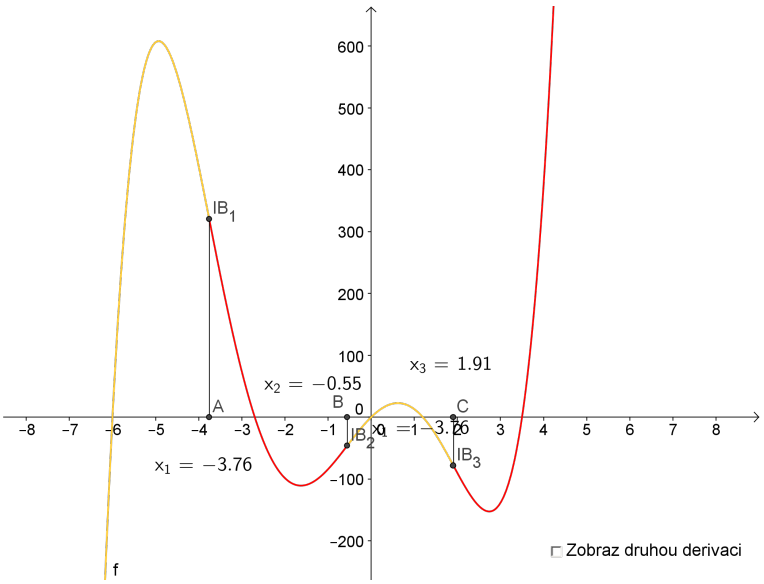
PŘÍKLAD 24.

Graf funkce $f(x) = 100x^5 + 400x^4 - 2049x^3 - 3960x^2 + 6804x$ zobrazte tak, aby konvexní úseky grafu byly zobrazeny červeně a konkávní úseky žlutě.

1.	Vstup:	`f(x)=100*x^5+400*x^4-2049*x^3-3960*x^2+6804*x`
2.	Vstup:	`Derivace[f,2]`
3.	Vstup:	`NuloveBody[f'']`
4.	Vstup:	`x_1=x(A)  x_2=x(B)  x_3=x(C)`

5.	Vstup:	`f1konkav=Funkce[f,-∞,x_1]`
		`f2konvex=Funkce[f,x_1,x_2]`
		`f3konkav=Funkce[f,x_2,x_3]`
		`f4konvex=Funkce[f,x_3,∞]`

6.		Funkcím `f1konkav`, `f3konkav` změníme barvu na žlutou a Funkcím `f2konvex`, `f4konvex` na červenou.



Obrázek 5.5: Konvexní a konkávní

K PROCVIČENÍ

1. Vykreslete graf funkce $f(x) = 100x^5 + 400x^4 - 2049x^3 - 3960x^2 + 6804x$ tak, aby rostoucí konvexní úseku byly červeně, rostoucí konkávní modře, klesající konvexní žlutě a klesající konkávní zeleně.

2. Najděte lokální extrémy funkce $f(x) = \arcsin \frac{x}{x+1}$. Zobrazte je na grafu funkce a sestrojte v každém lokálním extrému tečnu.

3. Najděte inflexní body funkce $f(x) = \frac{x^2}{x+4}$. Zobrazte je na grafu funkce a sestrojte v každém inflexním bodě bodech tečnu.

4. Vykreste graf funkce a graf derivace funkce $f(x) = |x|$. Na grafu funkce zadejte bod a v tomto bodě sestrojte tečnu.

KAPITOLA

6

EXTREMÁLNÍ ÚLOHY

NAUČÍME SE

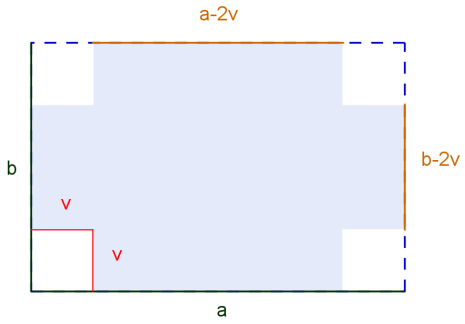
Naučíme se řešit extremální úlohy.

6.1 Úloha „Krabice“

ZADÁNÍ ŘEŠENÉ ÚLOHY

V rozích obdélníkové lepenky o rozměrech a, b vyřízneme čtverce tak, abychom z ní mohli udělat krabici. Určete velikost výřezu tak, aby objem krabice byl co největší.

$$V = (a - 2v)(b - 2v)v$$



Obrázek 6.1: Rozbor úlohy

POSTUP ŘEŠENÍ

Rozbor

1.	a = 2	Posuvníky pro pro rozměr lepenky a, b od hodnoty 0 po hodnotu 10.

2.	Vstup:	Zadáme body v rozích obdélníka.
		`A=(0,0)`
		`B=(a,0)`
		`C=(a,b)`
		`D=(0,b)`

3.		Sestrojíme obdélník $ABCD$.
4.	Vstup:	Jaký bude rozsah výřezu? Od hodnoty 0 po hodnotu $\min(\frac{a}{2}, \frac{b}{2})$. Zadáme tedy `rozsah=Minimum[a/2,b/2]`
5.	a = 2	Vytvoříme Posuvník *vyrez* pro výřez, tedy od 0 do `rozsah`.
6.		V rozích obdélníka uděláme čtverce o délce strany `vyrez`.

Sestavení funkce

Krabice buce mít tvar kvádru, jeho objem se spočítá jako součin délek všech jeho tří rozměrů.

7.	Vstup:	Objem krabice bude `V=(a-2*vyrez)*(b-2*vyrez)*vyrez`
8.		Změnou hodnoty posuvníku `vyrez` zkuste najít největší hodnotu `V`

Nalezení extrému funkce

Označme si nyní hledanou hodnoty výřezu jako x, pak máme úlohu:

Analytické řešení ukážeme pro hodnoty $a = 10$, $b = 6$:

Hledáme maximum funkce $V_f(x)$ *na intervalu* $(0, 3)$*:*

$$V_f(x) = (10 - 2x)(6 - 2x)x$$

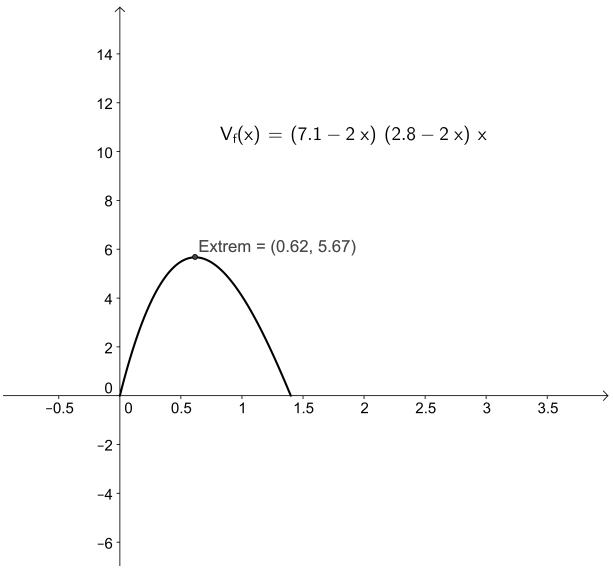
Spočítáme derivaci:

$$V_f'(x) = 12x^2 - 64x + 60$$

Derivaci položíme rovnu nule a dostaneme kvadratickou rovnici, kterou vyřešíme.

$$x^2 - 16x + 15 = 0 \quad \Rightarrow \quad \underline{\underline{x = 1.21}}$$

9.		Otevřeme si druhou Nákresnu a následujíc vypočet provedeme do Nákresny 2.
10.	Vstup:	Zadáme funkci `Vf = Funkce[(a-2*x)*(b-2*x)*x,0,rozsah]`
11.	Vstup:	Spočítám extrém funkce `Extrem[Vf,0,rozsah]`



Obrázek 6.2: Úloha „Krabice“ – funkce

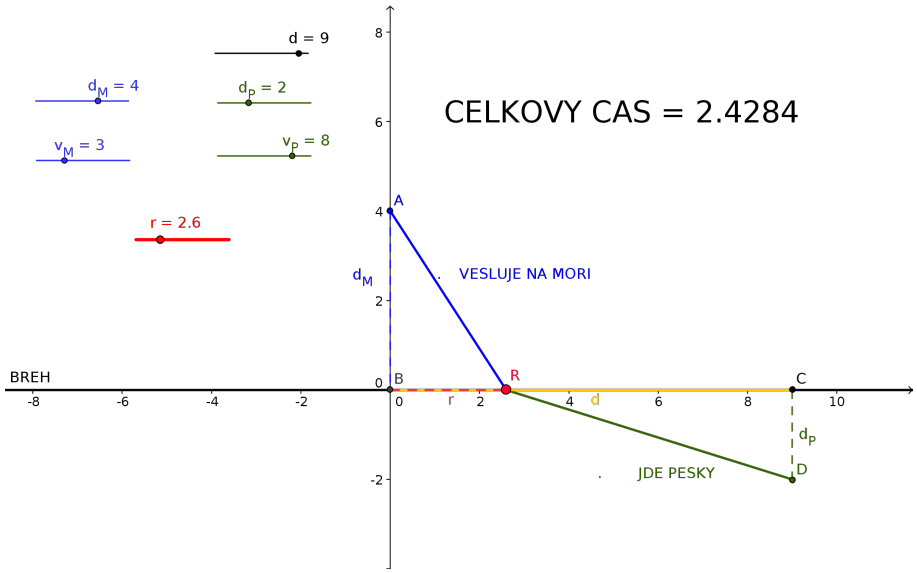
6.2 Úloha „Spěchající rybář“

ZADÁNÍ ŘEŠENÉ ÚLOHY

Rybář se nachází na moři v bodě A, který je od nejbližšího bodu na pobřeží B ve vzdálenosti d_M. Jeho chýše stojí v bodě D ve vnitrozemí, který je od nejbližšího bodu na pobřeží C ve vzdálenosti d_P. Vzdálenost bodů B a C je d. Na moři může veslovat rychlostí v_M a na pevnině jde rychostí v_P.
Do kterého místa R na pobřeží má veslovat, aby se domů dostal co nejrychleji?

POSTUP ŘEŠENÍ

Rozbor



Obrázek 6.3: Úloha „Spěchající rybář“

Konstrukce

1.		Vložíme posuvníky na vzdálenosti `dM`, `dP`, `d` v rozsahu od 0 do 10
2.		Do vstupního pole zadáme postupně body `B=(0,0)` `A=(0,dM)` `C=(d,0)` `D=(d,-dP)`
3.		Vytvoříme úsečky AB, BC, CD
4.		Vložíme posuvník na vzdálenost `r` v rozsahu od 0 do d s krokem 0.01 a změníme mu barvu na červenou.
5.		Do vstupního pole zadáme bod `R=(r,0)`.
6.		Vytvoříme úsečky AR, RD a nakonec úsečku BR, které změníme barvu na červenou.
7.		Do vstupního pole zadáme proměnnou počítající délku dráhy na moři `sM=sqrt(dM^2+r^2)` (jako kontrolu srovnejme hodnotu `sM` s délkou úsečky AR).
8.		Do vstupního pole zadáme proměnnou počítající délku dráhy na pevnině `sP=sqrt(dP^2+(d-r)^2)` (jako kontrolu srovnejme hodnotu `sP` s délkou úsečky DR).
9.		Vložíme posuvníky na rychlosti veslování na moři a chůze na pevnině `vM`, `vP`, rozsah si zvolíme pro libovolné kladné hodnoty.
10.		Zadáme do vstupního pole výpočet celkového času `t=sM/vM+sP/vP`, myší můžeme přetáhnout z algebraického okna na nákresnu.
11.		Změnou hodnoty červeného posuvníku r vidíme změnu hodnoty času, takto „zkusmo“ prozatím najdeme minimum.

Sestavení funkce

Úlohu jsme popsali geometricky a „zkusmo“ jsme našli řešení. Nyní se na problém podíváme podrobněji. Označme jako x hledanou vzdálenosti (bodů AR) na břehu.
Pak celkový čas T bude funkcí x a bude mít předpis:

$$T(x) = \frac{\sqrt{dM^2 + x^2}}{vM} + \frac{\sqrt{dP^2 + (d-x)^2}}{vP}$$

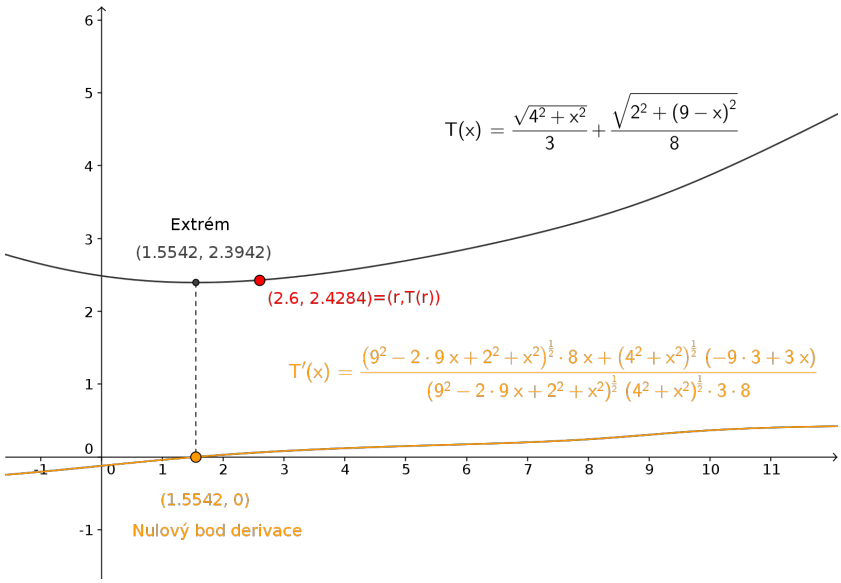
Hledáme lokální mimimum funkce $T(x)$ na intervalu $\langle 0, d \rangle$.

$$T(x) = \frac{\sqrt{dM^2 + x^2}}{vM} + \frac{\sqrt{dP^2 + (d-x)^2}}{vP}$$

Nalezení extrému funkce

V GeoGebře zadáme funkci a najdeme jeho extrém. Jiný způsob je spočítat derivaci funkce a najít její nulový bod na příslušném intervalu.

12.	Vstup:	`T(x)=sqrt(dM^2+x^2)/vM+sqrt(dP^2+(d-x)^2)/vP`
13.	Vstup:	`Extrem[T,0,d]`
14.	Vstup:	`Derivace[T]` `NulovyBod[T',0,d]`



Obrázek 6.4: Úloha „Spěchající rybář“ – funkce

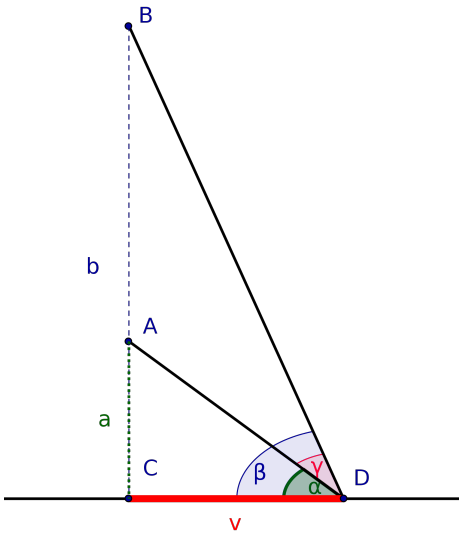
6.3 Úloha „Plakát“

ZADÁNÍ ŘEŠENÉ ÚLOHY

Na stěně je umístěn plakát, jeho spodní okraj je ve výšce a nad pozorovatelem a horní okraje je ve výšce b nad pozorovatelem. Určete vzdálenost v od stěny tak, aby z ní byl vidět plakát v maximálním zorném úhlu.

POSTUP ŘEŠENÍ

Rozbor



Co je dáno:
výška $a = |AC|$ je vzdálenost bodů A a C
výška $b = |BC|$ je vzdálenost bodů B a C

Ostatní:
Plakát je na obrázku úsečka AB.
Pozorovatel je v bodě D.

Co hledáme:
vzdálenost od stěny je $v = |CD|$, což je vzdálenost bodů C a D

Jak hledáme:
tak, aby byl maximální úhel γ

V pravoúhlém trojúhelníku CDA platí:

$$\operatorname{tg}(\alpha) = \frac{a}{v} \quad \Longrightarrow \quad \alpha = \operatorname{arctg}\left(\frac{a}{v}\right)$$

a v pravoúhlém trojúhelníku CDB platí:

$$\operatorname{tg}(\beta) = \frac{b}{v} \quad \Longrightarrow \quad \beta = \operatorname{arctg}\left(\frac{b}{v}\right)$$

Hledaný zorný úhel γ je úhel v trojúhelníku ADB u vrcholu D:

$$\gamma = \beta - \alpha$$

Sestavení funkce

Funkci, jejíž maximum hledáme, je hodnota zorného úhlu γ v závislosti na proměnné v:

$$\underline{\underline{\gamma(v)}} = \beta - \alpha = \underline{\underline{\operatorname{arctg}\left(\frac{b}{v}\right) - \operatorname{arctg}\left(\frac{a}{v}\right)}}$$

Hledáme lokální mimimum funkce $\gamma(x)$ na intervalu $(0, \infty)$.

$$\gamma(x) = \operatorname{arctg}\left(\frac{b}{x}\right) - \operatorname{arctg}\left(\frac{a}{x}\right)$$

Nalezení extrému funkce

Hledáme maximum funkce $\gamma(v)$.

Spočítáme derivaci:

$$\gamma'(v) = \frac{1}{1+\left(\frac{b}{v}\right)^2}\left(\frac{-b}{v^2}\right) - \frac{1}{1+\left(\frac{a}{v}\right)^2}\left(\frac{-a}{v^2}\right) = \frac{1}{\frac{v^2+b^2}{v^2}}\left(\frac{-b}{v^2}\right) - \frac{1}{\frac{v^2+a^2}{v^2}}\left(\frac{-a}{v^2}\right) =$$
$$= \frac{-b}{v^2+b^2} + \frac{a}{v^2+a^2} = \frac{-b(v^2+a^2)+a(v^2+b^2)}{(v^2+b^2)(v^2+a^2)} = \frac{(v^2-ab)(a-b)}{(v^2+b^2)(v^2+a^2)}$$

a najdeme stacionární body, což jsou nulové body derivace:

$$\begin{aligned} \gamma'(v) &= 0 \\ \frac{(v^2-ab)(a-b)}{(v^2+b^2)(v^2+a^2)} &= 0 \\ (v^2-ab)(a-b) &= 0 \\ v &= \sqrt{ab} \end{aligned}$$

Extrém funkce $\gamma(v)$ je pro hodnotu:

$$\underline{\underline{v_* = \sqrt{ab}}}$$

Pro hodnotu v_* má funkce $\gamma(v)$ maximum.
Největší úhel, pod kterým můžeme vidět plakát spočítáme dosazením v_* do funkce $\gamma(v)$:

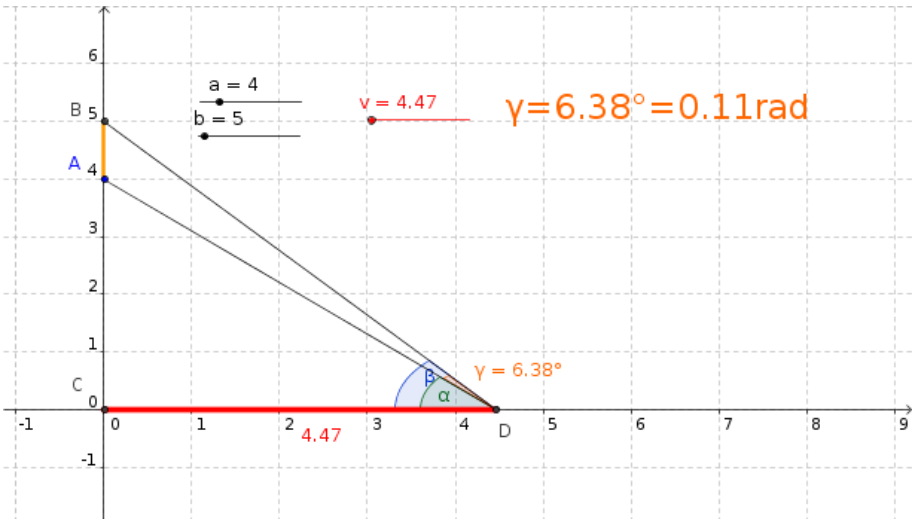
$$\gamma_* = \gamma(v_*) = \operatorname{arctg}\left(\frac{b}{v_*}\right) - \operatorname{arctg}\left(\frac{a}{v_*}\right) = \operatorname{arctg}\left(\frac{b}{\sqrt{ab}}\right) - \operatorname{arctg}\left(\frac{a}{\sqrt{ab}}\right)$$
$$= \operatorname{arctg}\left(\sqrt{\frac{b}{a}}\right) - \operatorname{arctg}\left(\sqrt{\frac{a}{b}}\right)$$

Spočítáme několik konkrétních příkladů.

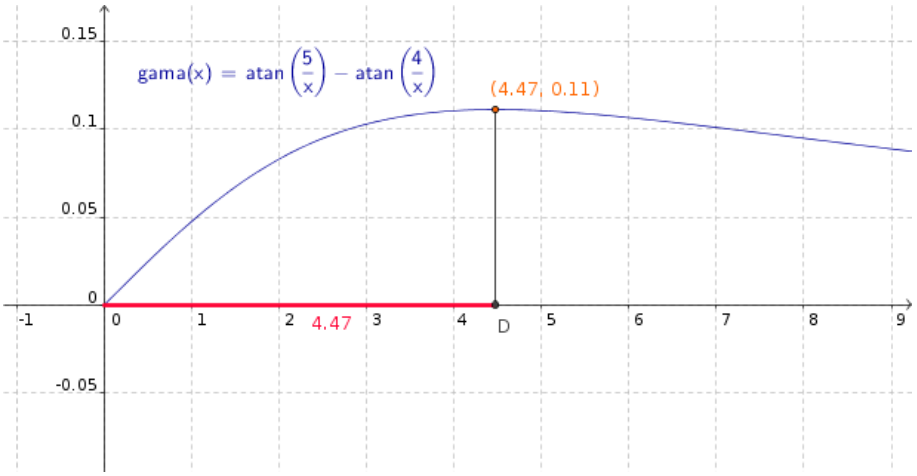
zadané		výsledky	
a	b	γ_*	v_*
4 m	5 m	0.11 rad = 6.38°	4.47 m
3 m	5 m	0.25 rad = 14.48°	3.87 m
10 m	13 m	0.13 rad = 7.49°	11.4 m

Konstrukce

V první nákresně vytvořte geometrickou simulaci (obrázek 6.5) a v druhé vykreste funkci a najděte extrém (obrázek 6.6) .
Zobrazení příkladu pro hodnoty $a = 4\,\text{m}$, $b = 5\,\text{m}$.



Obrázek 6.5: Úloha „ Plakát“



Obrázek 6.6: Úloha „ Plakát“ – funkce

Závěr

Plakát umístěný ve výšce a (spodní okraj) a b (horní okraj) je vidět pod maximálním zorným úhlem $\gamma_* = \operatorname{arctg}\left(\sqrt{\frac{b}{a}}\right) - \operatorname{arctg}\left(\sqrt{\frac{a}{b}}\right)$ ze vzdálenosti $v_* = \sqrt{ab}$.

K PROCVIČENÍ

1. **Odpadní kanál** Dělníci mají vykopat odpadní kanál, jehož průřez má tvar rovnoramenného lichoběžníka. Je zadaná jeho hloubka h a obsah průřezu S. Určete délku kratší základny lichoběžníku a (a delší základny) tak, aby náklady na vyzdění všech stěn kanálu byly co nejmenší.

2. **Dóza na sladkosti** Děti vyrábejí dózu na sladkosti ve tvaru válce s daným povrchem P. Určete polomer podstavy r (a výšku válce) tak, aby se do dózy vešlo co nejvíce sladkostí.

3. **Marcipán** Cukrářka potřebuje z marcipánové koule o poloměru r vyřezat ozdobu ve tvaru válce. Určete poloměr podstavy ozdoby a (a výšku) tak, aby se na výrobu ozdoby použilo co nejvíce marcipánu.

4. **Balík slámy** Farmář má přístřešek ve tvaru pravidelného čtyřbokého jehlanu o výšce v a délce podstavy a. Potřebuje do něj ukrýt balík slámy ve tvaru válce. Určete poloměr podstavy válce r (a jeho výšku) tak, aby mohl být balík co největší.

5. **Aranžování kvetin** Aranžérka potřebuje z polystyrénové koule o poloměru r vyřezat kuželovou podložku pod květiny. Určete poloměr podstavy kužele a (a jeho výšku v) tak, aby se na výrobu podložky použilo co nejvíce polysterénu.

6. **Vzácné plátno** Zloději mají úkryt ve tvaru kužele o poloměru r podstavy a výšce v, a do něj plánují uschovat vzácné plátno stočené do tuby ve tvaru válce. Určete poloměr podstavy válce a (a jeho výšku) tak, aby povrch válce byl co nejvetší.

7. **Skauti** Skauti mají stan ve tvaru pravidelného čtyřbokého jehlanu se stranou podstavy d a výšce v. Do stanu potrebují schovat krabici ve tvaru kvádru s tajným pokladem. Určete rozměry kvádru a, b, c tak, aby mohli schovat co největší poklad.

8. **Mimozemštani** Mimozemská lod má tvar koule o poloměru r a její posádka potřebuje lodí odvézt nasbíraný výzkumný materiál ve válcovém boxu. Určete polomer podstavy válce a (a jeho výšku v) tak, aby byl povrch válce co největší.

9. **Turecký med** Cukrár má kouli tureckého medu o polomeru r a potrebuje z ní vyříznout kvádr. Urcete rozmery kvádru a, b, c tak, aby vzniklý kvádr byl co nejvetší.

10. **Zmrzlinový kornout** Zmrzlinářka chce udělat papírový kornout s víčkem ve tvaru kužele, do které dáme zmrzlinu o objemu V. Určete výšku kornoutu v (a poloměr podstavy) tak, aby byla spotřeba papíru co nejmenší.

11. **Týpí** Indián má šest tyčí délky a, ze kterých chce postavit týpí ve tvaru pravidelného šestibokého jehlanu. Určete výšku v (a základnu) tak, aby v týpí byl co největším prostor.

12. **Stan z kuže** Tramp má kuži ve tvaru kruhu o poloměru r. Z ní chce vyříznout kruhovou výseč na opláštení stanu ve tvary kužele - plášt kužele. Určete výšku stanu v (a poloměr podstavy) tak, aby ve stanu byl co největším prostor.

13. **Zahrada** Zahrada má tvar rovnoramenného lichobežníka, jeho ramena a menší základna mají délku a. Určete délku větší základny b (a výšku lichobežníka) tak, aby obsah zahrady byl největší.

14. **Drát** Drát délky a rozdělíme na dvě části. Z jedné svineme čtverec a z druhé kruh. Určete délku části pro výrobu kruhu b (a pro čtverec) tak, aby součet obsahu čtverce a kruhu byl co nejmenší.

15. **Papírový drak** Papírový drak má tvar kruhové výseče s obvodem l. Určete poloměr oblouku výseče r (a jeho délku) tak, aby drak měl co největší plošný obsah.

16. **Věžička** Ve věžičce tvaru kužele o poloměru podstavy r a výšce v je potřeba zřídit válcovou místnost. Určete poloměr podstavy mísnosti a (a její výšku) tak, aby místnost měla co největší objem.

17. **Tunel** Do tunelu ve tvaru půlkružnice o poloměru r je vloženo bednění ve tvaru lichobežníku, jehož základna je průměr půlkružnice. Určete výšku lichobežníku v (a úhel u jeho základny) tak, aby byl jeho obsah co největší.

18. **Stavitel silnic** Průmyslový závod je umístěn ve vzdálenosti a od staré cesty vedoucí do města. Vzdálenost závodu od města je vzdušnou čarou b. Zjistěte, pod jakým úhlem α je nutné vybudovat přípojku (novou cestu) spojující závod a starou cestu, tak aby doprava materiálu ze závodu do města byla co nejlevnejší. Náklady na přepravu (na 1km) jsou na staré cestě 0.5 Kč a na nové 1.5 Kč.

19. **Střecha** Podkrovní místnost má šířku a a výšku b. Nad mísností se bude stavět sedlová strecha (začínající u země). Určete nejkratší možnou délku střechy s.

20. **Trám** Tesař má kmen stromu ve tvaru válce o poloměru r, ze kterého chce vytesat trám ve tvaru kvádru. Určete výšku průřezu trámu v (a jeho šírku s) tak, aby byla nosnost trámu co největší. Nosnost je dána vztahem $y = ksv^2$, kde k je materiálová konstanta.

KAPITOLA

7

PARAMETRICKY ZADANÁ FUNKCE

NAUČÍME SE

Naučíme se zadat parametrickou funkci, spočítat její derivaci a sestavit tečnu.

OPAKOVÁNÍ MATEMATIKY

Derivace paramatericky zadané funkce

Definice : Jsou dány funkce $x = \varphi(t)$, $y = \psi(t)$, kde $t \in I$ je parametr. Mno6inu bodů o souřadnicích určených těmito rovnicemi nazveme **parametrickou křivkou**.

Definice : Jsou dány funkce $x = \varphi(t)$, $y = \psi(t)$, kde $t \in I$ je parametr. Nechť existuje φ^{-1}. Pak funkci

$$y = f(x) = \psi\left(\varphi^{-1}(x)\right)$$

nazveme **parametricky zadanou funkci**.

Věta : *Funkce f je dána parametricky rovnicemi $x = \varphi(t)$, $y = \psi(t)$, kde $t \in I$. Nechť $\varphi(t)$ a $\psi(t)$ mají derivaci v každém bodě intervalu I. Pak **derivace parametricky zadané funkce** f je dána vztahem:*

$$y' = \frac{\dot{\psi}(t)}{\dot{\varphi}(t)}$$

Poznámka: Derivaci podle t značíme tečkou, abychom ji odlišili od derivace podle x, kterou značíme čárkou.

VÝKLAD UČIVA

Parametricky zadanou funkci zadáme příkazem `Krivka`.
`Krivka[<Výraz1>,<Výraz2>,<Parametr proměnné>,<Počáteční hodnota>,<Koncová hodnota>]`

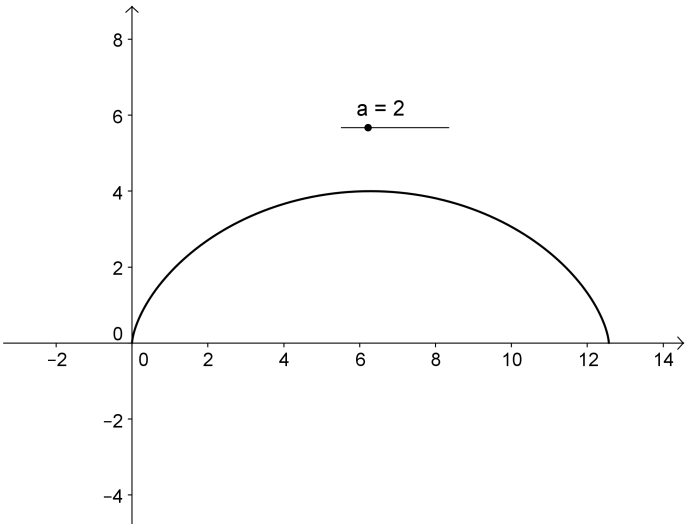
`<Výraz1>`	předpis pro x-ovou souřadnici
`<Výraz2>`	předpis pro y-ovou souřadnici
`<Parametr proměnné>`	název parametru
`<Počáteční hodnota>`	počáteční hodnota parametru
`<Koncová hodnota>`	koncová hodnota parametru

Například funkci danou rovnicemi $x(t) = 5\cos(t)$, $y(t) = 5\sin(t)$, kde $t \in \langle 0, \pi \rangle$ zadáme
`Krivka[5*cos(t),5*sin(t),t,0,pi]`

PŘÍKLAD 25.

Sestavte simulaci pro cykloidu, která je dána parametrickými rovnicemi $x = a(t - \sin(t))$, $y = a(1 - \cos(t))$, parametr $t = \langle 0, 2\pi \rangle$ a hodnotu a budeme volit $1, 2, 3, \ldots, 10$

1.		Vytvoříme posuvník a od 1 do 10 s krokem 1
2.		Zadáme křivku `Krivka[a*(t-sin(t)),a*(1-cos(t)),t,0,2*pi]`



Obrázek 7.1: Cykloida

7.1 Úloha „Střelba na cíl“

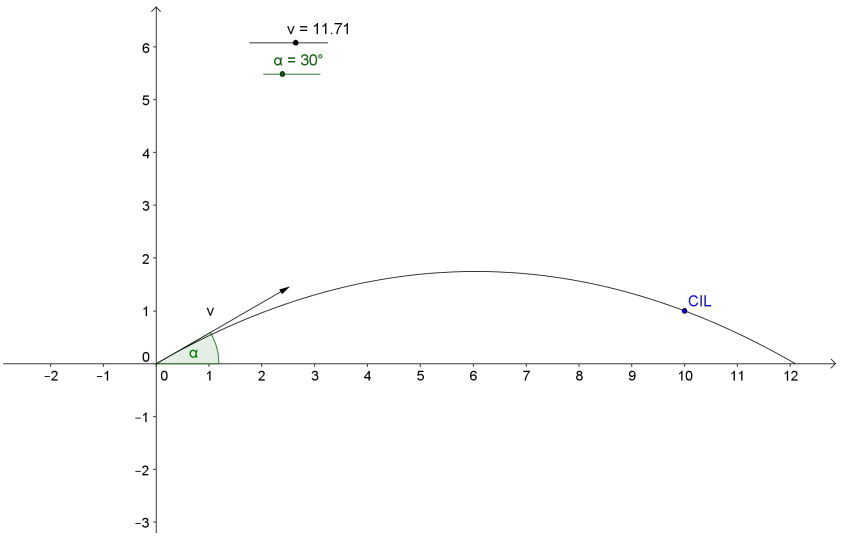
ZADÁNÍ ŘEŠENÉ ÚLOHY

Sestrojte simulaci šikmého vrhu, začínající v počátku souřadnic. Počáteční rychlost a elevační úhel lze měnit. Cíl je umístěn v bodě $(10, 1)$.
Spočítejte:

a) Jakou hodnotu musí mít úhel, je-li pevně dána rychlost a chceme-li se trefit do cíle?

b) Jakou rychlostí je potřeba vystřelit, je-li pevně dán úhel a chceme-li se trefit do cíle?

c) Cílový bod leží na vodorovné desce. Spočítejte úhel dopadu do cíle, pokud je elevační úhel a počáteční rychlost pevně dána podle b).



Obrázek 7.2: Šikmý vrh

POSTUP ŘEŠENÍ

Rozbor

Z fyziky víme, že šikmý vrh je dán parametrickými rovnicemi:

$$\begin{aligned} x(t) &= v\,t\,\cos(\alpha) \\ y(t) &= v\,t\,\sin(\alpha) - \frac{1}{2}g\,t^2 \end{aligned} \tag{7.1}$$

kde t je čas od 0 do doby dopadu na zem, $g = 9.81\,m\,s^{-2}$ je konstanta, v je počáteční rychlost (může být maximálně rovna první kosmické rychlosti $7.9km\,s^{-1}$)a α je elevační úhel (viz obrázek 7.2). Střela má v každém čase t polohu $(x(t), y(t))$.
Nejprve si spočítáme čas dopadu na zem, při kterém má střela y-ovou souřadnici rovnu 0. Do druhé rovnice (7.1) dosadíme za y hodnotu 0:

$$0 = v\,t\,\sin(\alpha) - \frac{1}{2}g\,t^2\,.$$

Vyřešíme tuto kvadratickou rovnici pro neznámou t a dostaneme dvě řešení. Počáteční čas $t = 0$ a čas dopadu na zem:

$$t_{dopad} = \frac{2v\sin(\alpha)}{g} \tag{7.2}$$

Konstrukce

Nejprve si vytvoříme simulaci šikmého vrhu.

1.		Vytvoříme posuvník α pro úhel od 0° do 90° stupňu s krokem 1°. A nastavme jeho hodnotu na 30°.
2.		Vytvoříme posuvník v od 0 do 20 s krokem 0.01 a jeho hodnotu nastavíme na 10.
3.		Zadáme hodnotu konstanty `g=9.81`
4.		Čas dopadu na zem `tdopad=(2*v*sin(α))/g` (podle vzorce (7.2))
5.		Nyní zapíšeme parametricky zadanou křivku: `k=Krivka[v*t*cos(α), v*t*sin(α)-1/2*g*t^2, t, 0, tdopad]`
6.		Zadáme cíl jako bod `CIL=(10,1)` a ve Vlastnostech tohoto bodu zatrhneme **Upevnit objekt**, neboť tímto bodem nebudeme hýbat.
7.		Zkusme nyní najít kombinaci hodnot α a v tak, aby křivka protla cíl.

Výpočet úlohy a)

Jakou hodnotu musí mít úhel, je-li pevně dána rychlost a chceme-li se trefit do cíle?

Nejprve si spočítáme úlohu pro konkrétní hodnotu počáteční rychlosti $v = 20$.
Do rovnic (7.1) popisující šikmý vrh dosadíme souřadnice cíle $(10, 1)$ a $v = 20$, $g = 9.81$:

$$\begin{aligned} x &= v\,t\,\cos(\alpha) \\ y &= v\,t\,\sin(\alpha) - \frac{1}{2}g\,t^2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} 10 &= 20t\,\cos(\alpha) \\ 1 &= 20t\,\sin(\alpha) - \frac{1}{2}9.81\,t^2 \end{aligned}$$

Hledáme $t \in \langle 0, t_{dopad}\rangle$ *a* $\alpha \in (0°, 90°)$, *jako řešení soustavy rovnic:*

$$\begin{aligned} 10 &= 20t\,\cos(\alpha) \\ 1 &= 20t\,\sin(\alpha) - \frac{1}{2}9.81\,t^2 \end{aligned}$$

Z první rovnice vyjádříme $\sin(\alpha)$ a z druhé $\cos(\alpha)$:

$$\begin{aligned} 10 &= 20t\,\cos(\alpha) && \Rightarrow \quad \cos(\alpha) = \frac{1}{2t} \qquad (\star) \\ 1 &= 20t\,\sin(\alpha) - \frac{1}{2}\,9.81\,t^2 && \Rightarrow \quad \sin(\alpha) = \frac{2 + 9.81t^2}{40t} \end{aligned}$$

Použijeme známý vztah $\sin^2(\alpha) + \cos^2(\alpha) = 1$ a dosadíme do něj:

$$\begin{aligned} \sin^2(\alpha) + \cos^2(\alpha) &= 1 \\ \left(\frac{2 + 9.81t^2}{40t}\right)^2 + \left(\frac{1}{2t}\right)^2 &= 1 \\ \frac{\left(2 + 9.81t^2\right)^2}{1600t^2} + \frac{1}{4t^2} &= 1 \\ \left(2 + 9.81t^2\right)^2 + 400 &= 1600t^2 \\ 96.2361t^4 - 1560.76t^2 + 404 &= 0 \qquad (\star\star) \end{aligned}$$

Máme polynomickou rovnici, kterou bychom mohli vyřešit substitucí za t^2 (a vyřešit kvadratickou rovnici) nebo použijeme k jejímu vyřešení GeoGebru.

8.		Nastavme hodnotu posuvníku v na hodnotu 20.
9.		V Menu Zobrazit si otevřeme **Nákresnu 2** a použijeme ji k pomocnému výpočtu této rovnice. A následující kroky zobrazujeme v druhé nákresně.
10.		Zadáme rovnici `pom(t)=96.2361*t^4-1560.76*t^2+404` a vyřešíme ji `NuloveBody[pom]` A vidíme, že se na ose x objevily průsečíky, jejichž x-ové souřadnice jsou kořeny naší rovnice.
11.		Neboť t je kladné, podíváme se na kladné kořeny. Předpokládám, že je Geogebra pojmenovala C, D. `t1=x(C)  t2=x(D)`
12.		Nakonec si potřebujeme spočítat úhel, tedy `alfa1=acos(1/(2*t1))  alfa2=acos(1/(2*t2))`
13.		Chceme-li úhly ve stupních, tak `alfa1st=alfa1*180°/pi  alfa2st=alfa2*180°/pi`

Pokud jsou vaším výsledkem hodnoty $\alpha_1 = 12.9^\circ$ a $\alpha_2 = 82.81^\circ$, počítali jste správně.

Závěr: Pro hodnotu počáteční rychlosti $v = 20$ se do cíle v bodě $(10, 1)$ trefíme pro hodnoty elevačního úhlu $\alpha_1 = 12.9^\circ$ nebo $\alpha_2 = 82.81^\circ$.

Výpočet úlohy b)

Jakou rychlostí je potřeba vystřelit, je-li pevně dán úhel a chceme-li se trefit do cíle?

Nejprve si spočítáme úlohu pro konkrétní hodnotu elevačního úhlu $\alpha = 30°$.

14.		Nastavte hodnotu posuvníku α na $30°$ a zkuste najít hodnotu v tak, aby dráha střely procházela cílem.

Do rovnic (7.1) popisující šikmý vrh dosadíme souřadnice cíle $(10, 1)$ a $\alpha = 30°$, $g = 9.81$:

$$\begin{aligned} x &= v\,t\,\cos(\alpha) \\ y &= v\,t\,\sin(\alpha) - \frac{1}{2}\,g\,t^2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} 10 &= v\,t\,\frac{\sqrt{3}}{2} \\ 1 &= v\,t\,\frac{1}{2} - \frac{9.81}{2}\,t^2 \end{aligned}$$

Hledáme $t \in \langle 0, t_{dopad} \rangle$ a $v \in \langle 0, 7900 \rangle$ jako řešení soustavy rovnic:

$$\begin{aligned} 10 &= v\,t\,\frac{\sqrt{3}}{2} \\ 1 &= v\,t\,\frac{1}{2} - \frac{9.81}{2}\,t^2 \end{aligned}$$

Z první rovnice vyjádříme $v\,t$:

$$\begin{aligned} 10 = v\,t\,\frac{\sqrt{3}}{2} \quad &\Rightarrow \quad \frac{20}{\sqrt{3}} = v\,t \\ 1 = v\,t\,\frac{1}{2} - \frac{9.81}{2}\,t^2 \quad &\Rightarrow \quad 2 = v\,t\, - 9.81\,t^2 \end{aligned}$$

A z první rovnice dosadíme do druhé a dostaneme :

$$2 = \frac{20}{\sqrt{3}} - 9.81\,t^2$$

Výsledek je:

$$t_* = \sqrt{\frac{20 - 2\sqrt{3}}{9.81\sqrt{3}}} \approx \underline{\underline{0.9865}}$$

Dopočítáme rychlost

$$v_* = \frac{20}{t\sqrt{3}} = \frac{20}{0.9865\sqrt{3}} = \underline{\underline{11.705}}$$

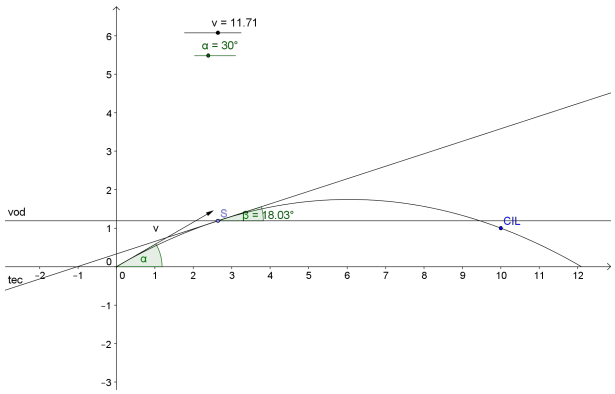
15.		Nastavíme hodnotu `v=11.705` a vidíme, zda křivka protne bod `CIL`

Závěr: Pro hodnotu elevačního úhlu $\alpha = 30°$ se do cíle v bodě $(10, 1)$ trefíme pro hodnotu počáteční rychlosti $v = 11.705$.
A pro hodnotu elevačního úhlu α se do cíle v bodě $(10, 1)$ trefíme pro hodnotu počáteční rychlosti

$$v_* = \frac{10}{t_*\cos(\alpha)}, \text{ kde } \quad t_* = \sqrt{\frac{20\,\mathrm{tg}(\alpha) - 2}{g}} \tag{7.3}$$

Výpočet úlohy c)

Cílový bod leží na vodorovné desce. Spočítejte úhel dopadu do cíle, pokud je elevační úhel a počáteční rychlost pevně dána podle b).



Obrázek 7.3: Šikmý vrh - úhel dopadu

16.		Na křivku k umístíme bod S.
17.		Zadáme tečnu tec v bodě S ke křivce k. `tec=Tecna[S,k]`
18.		Vytvoříme přímku vod, která prochází bodem S a je rovnoběžná s osou x.
19.		Spočítáme úhel β mezi tečnou teca přímkou vod.

Nejprve si spočítáme úhel v jakémkoli bodě na křivce. Připomeňme, že tangens úhlu mezi tečnou a osou x je roven derivaci v tomto bodě

$$\operatorname{tg}(\beta) = \frac{\dot{y}}{\dot{x}} = \frac{v\,\sin(\alpha) - g\,t}{v\,\cos(\alpha)} = \operatorname{tg}(\alpha) - \frac{g\,t}{v\,\cos(\alpha)}$$

V cíli jsme v předchozím bodě spočítali hodnoty $\alpha = 30°,\ t_{30} = 0.9865,\ v_{30} = 11.705$

$$\operatorname{tg}(\beta_{30}) = \operatorname{tg}(\alpha) - \frac{g\,t_{30}}{v_{30}\,\cos(\alpha)} = \operatorname{tg}(30°) - \frac{9.81 \cdot 0.9865}{11.705 \cdot \cos(30°)} = -0.3773$$

$$\beta_{30} = \underline{\underline{-20.67°}} = 339.33°$$

Pro libovolně zvolený úhel dosadíme v_*, t_* z (7.3)

$$\begin{aligned}
\operatorname{tg}(\beta_*) &= \operatorname{tg}(\alpha) - \frac{g\,t_*}{v_*\,\cos(\alpha)} = \operatorname{tg}(\alpha) - \frac{g\,t_*}{\frac{10}{t_*\cos(\alpha)},\,\cos(\alpha)} = \\
&= \operatorname{tg}(\alpha) - \frac{g\,t_*^2}{10} = \operatorname{tg}(\alpha) - \frac{g}{10}\frac{20\operatorname{tg}(\alpha) - 2}{g} = \\
&= -\operatorname{tg}(\alpha) + \frac{1}{5} \\
\beta_* &= \underline{\underline{\operatorname{arctg}\left(-\operatorname{tg}(\alpha) + \frac{1}{5}\right)}}
\end{aligned}$$

20.	Vstup:	`betavcili=atan(-tan(α)+1/5)*180°/pi`

K PROCVIČENÍ

1. Vykreslete graf parametricky zadaná funkce, vytvořte na ni bod a v tomto bodě tečnu. V bodě, kde je tečna rovnoběžná s osu y neexistuje derivace. Mají tyto funkce takové body?

   (a)
   $$x = 2 + \cos(t) \quad y = 1 + \sin(t) \quad t \in \langle 0, \frac{\pi}{2} \rangle$$

   (b)
   $$x = 2\cos(t) \quad y = 4\sin(t) \quad t \in \langle 0, \pi \rangle$$

   (c)
   $$x = \frac{\cos(t)}{t} \quad y = \frac{\sin(t)}{t} \quad t \in (0\frac{3\pi}{4}\rangle$$

KAPITOLA

8

LINEÁRNÍ ALGEBRA

NAUČÍME SE

Ukážeme, jak v GeoGebře spočítat hodnost matice, determinant a jak najít inverzní matici. Dále popíšeme několik způsobů jak vyřešit soustavu lineárních rovnic.

OPAKOVÁNÍ MATEMATIKY

Definice : **Soustavou lineárních rovnic** rozumíme m rovnic o n neznámých, tedy

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \quad \cdots \quad + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \quad \cdots \quad + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \quad \cdots \quad + a_{mn}x_n &= b_m, \end{aligned}$$

kde a_{ij} se nazývají **koeficienty** soustavy, x_j jsou **neznámé** a b_i jsou **pravé strany** rovnic soustavy pro $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$.

Definice : Matici A, jejíž prvky tvoří koeficienty soustavy a_{ij}, nazýváme **maticí soustavy**, tedy

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Definice : Matici R, která vznikne z matice A připojením sloupce pravých stran, nazýváme **rozšířenou maticí soustavy**, tedy

$$R = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{pmatrix}$$

Poznámka: Soustavu m lineárních rovnic o n neznámých můžeme zapsat také maticově:

$$A \cdot \vec{x} = \vec{b},$$

kde A je matice soustavy, $\vec{x}$ je sloupcový vektor neznámých a $\vec{b}$ je sloupcový vektor pravých stran soustavy, tedy

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Věta : ***Frobeniova věta:***
*Soustava m lineárních rovnic o n neznámých má **alespoň jedno řešení**, právě když se hodnost matice soustavy (označení: h_A) rovná hodnosti matice rozšířené (označení: h_R), tedy*

$$h_A = h_R = h. \tag{8.1}$$

*Pokud $h_A \neq h_R$, pak řešení soustavy **neexistuje**.*

*Má-li soustava řešení, pak pro $h = n$ má soustava **právě jedno řešení**, jinak, tj. pro $h < n$ má soustava **nekonečně mnoho řešení** závislých na $n - h$ parametrech.*

8.1 Lineární algebra

8.1.1 Řešení soustav lineárních rovnic

PŘÍKLAD 26.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} 2x_1 + 3x_2 &= 5 \\ x_1 - x_2 &= -5 \\ x_1 + 2x_2 &= 4 \end{aligned}$$

Nejprve vytvoříme z koeficientů, které se nacházejí na levých stranách rovnic matici soustavy

$$A = \begin{pmatrix} 2 & 3 \\ 1 & -1 \\ 1 & 2 \end{pmatrix}.$$

Přidáním vektoru pravých stran poté získáme rozšířenou matici soustavy

$$R = \begin{pmatrix} 2 & 3 & 5 \\ 1 & -1 & -5 \\ 1 & 2 & 4 \end{pmatrix}.$$

Je zřejmé, že matice A je vždy součástí matice R. Nyní nás bude zajímat kolik a zda vůbec má daná soustava řešení. K tomu nám poslouží Frobeniova věta. Pomoci elementárních úprav převedeme matici R do schodovitého tvaru, ze kterého určíme hodnosti h_A a h_R.

$$R = \begin{pmatrix} 2 & 3 & 5 \\ 1 & -1 & -5 \\ 1 & 2 & 4 \end{pmatrix} \sim \quad \cdots \quad \sim \begin{pmatrix} 2 & 3 & 5 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{pmatrix}.$$

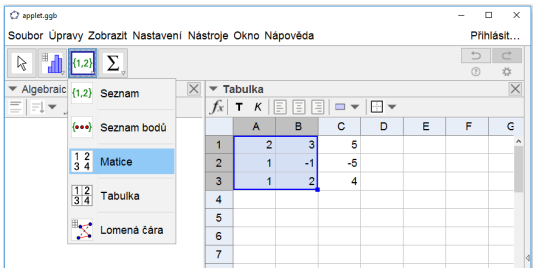
Vidíme, že hodnost $h_A = 2$ a také $h_R = 2$. Podle Frobeniovy věty má tedy soustava alespoň jedno řešení. Jestliže se podíváme na počet proměnných, tak zjistíme, že se rovná hodnotám obou hodností a proto má soustava jediné řešení.

Konstrukce

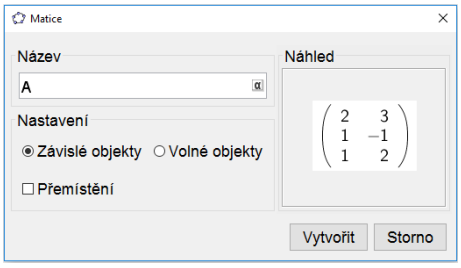
Nejprve zobrazíme tabulku (Zobrazit → Tabulka).

1.		Do buněk `A1:C3` zapíšeme koeficienty rozšířené matice soustavy.
2.		Označíme buňky `A1:B3` a vytvoříme matici soustavy `A` (obrázky 8.1 a 8.2).
3.		Označíme buňky `A1:C3` a vytvoříme rozšířenou matici soustavy `R`.

Jak je vidět na obrázku 8.2, při vytváření matice z tabulky máme možnost zvolit, zda jednotlivé prvky v tabulce budou závislými nebo volnými objekty. První volba znamená, že změníme-li některou z hodnot v tabulce, tak dojde současně ke změně tého hodnoty v matici. Pokud zvolíme druhou možnost, tak se matice bude chovat jako statický objekt a ke změně v matici nedojde. Tyto skutečnosti můžeme pozorovat v algebrajickém okně.



Obrázek 8.1: Nástroj pro vytvoření matice.



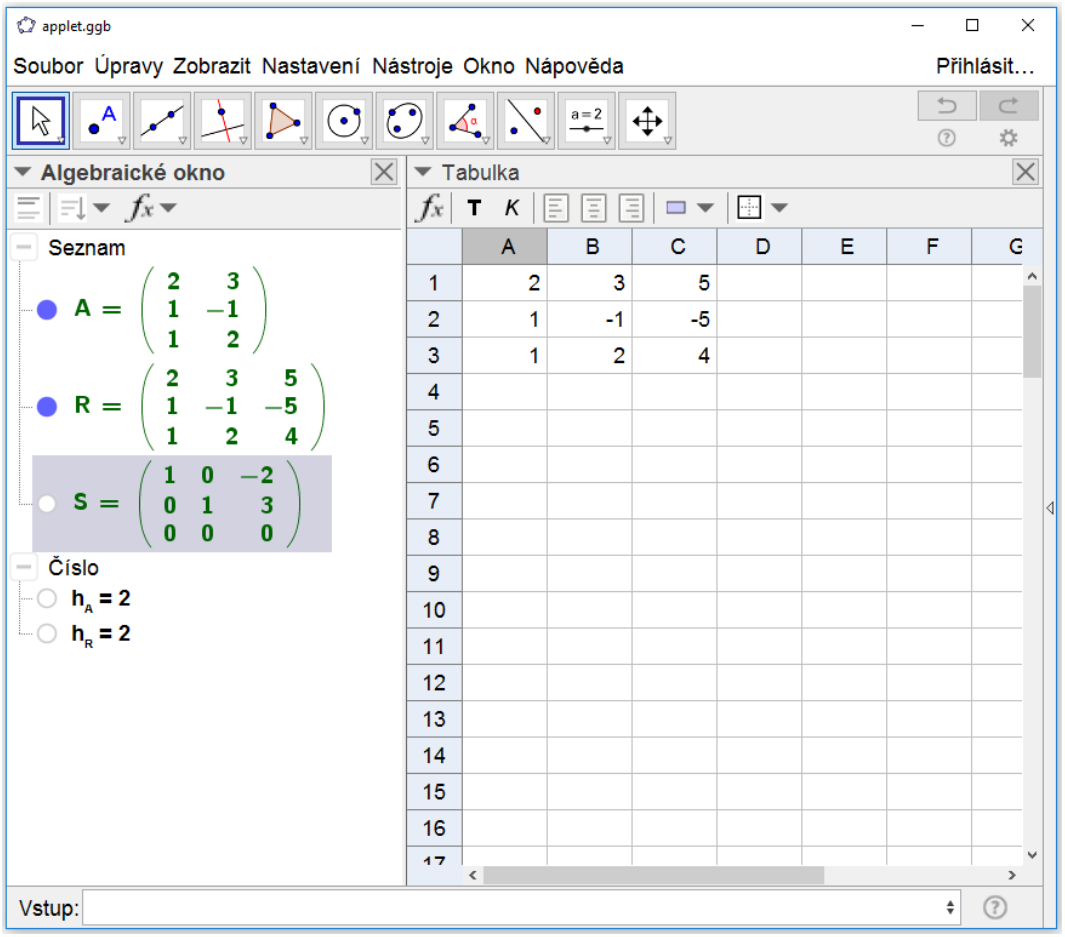
Obrázek 8.2: Vytvoření matice A.

1.		Do vstupního pole zapíšeme `h_A=Hodnost[A]`
2.		Do vstupního pole zapíšeme `h_R=Hodnost[R]`

Rozborem podle Frobeniovy věty zjistíme, zda má soustava lineární rovnic řešení a stanovíme jejich počet. Pokud soustava má řešení, tak se pustíme do jeho hledání.

3.	Vstup:	`S=SchodovityTvar[R]`

Příkaz `SchodovityTvar` převede rozšířenou matici soustavy do tvaru, který označujeme jako **Gauss-Jordanův**. Matice, kterou jsme získali, představuje ekvivalentní soustavu lineárních rovnic. Tato soustava má stejné řešení, jako ta zadaná. Můžeme si jí proto přepsat zpět do rovnic.



Obrázek 8.3: Řešení soustavy lineárních rovnic (matice ve schodovitém stvaru)

$$\begin{aligned} x_1 &= -2 \\ x_2 &= 3 \end{aligned}$$

Odtud již přímo vidíme řešení soustavy

$$\vec{x} = \begin{pmatrix} -2 \\ 3 \end{pmatrix}.$$

PŘÍKLAD 27.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 + x_5 &= 7 \\ 5x_1 + 4x_2 + 3x_3 + 3x_4 - x_5 &= 12 \\ x_1 + 2x_2 + 3x_3 + 3x_4 + 7x_5 &= 30 \\ 3x_1 + 2x_2 + x_3 + x_4 - 3x_5 &= -2 \\ 2x_1 + x_2 - 4x_5 &= -9 \end{aligned}$$

Nejprve vytvoříme z koeficientů, které se nacházejí na levých stranách rovnic matici soustavy

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 5 & 4 & 3 & 3 & -1 \\ 1 & 2 & 3 & 3 & 7 \\ 3 & 2 & 1 & 1 & -3 \\ 2 & 1 & 0 & 0 & -4 \end{pmatrix}.$$

Přidáním vektoru pravých stran poté získáme rozšířenou matici soustavy

$$R = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 7 \\ 5 & 4 & 3 & 3 & -1 & 12 \\ 1 & 2 & 3 & 3 & 7 & 30 \\ 3 & 2 & 1 & 1 & -3 & -2 \\ 2 & 1 & 0 & 0 & -4 & -9 \end{pmatrix}.$$

Je zřejmé, že matice A je vždy součástí v matici R. Nyní nás bude zajímat kolik a zda vůbec má daná soustava řešení. K tomu nám poslouží Frobeniova věta. Pomoci elementárních úprav převedeme matici R do schodovitého tvaru, ze kterého určíme hodnosti h_A a h_R.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 7 \\ 5 & 4 & 3 & 3 & -1 & 12 \\ 1 & 2 & 3 & 3 & 7 & 30 \\ 3 & 2 & 1 & 1 & -3 & -2 \\ 2 & 1 & 0 & 0 & -4 & -9 \end{pmatrix} \sim \quad \cdots \quad \sim \begin{pmatrix} 1 & 0 & -1 & -1 & -5 & -16 \\ 0 & 1 & 2 & 2 & 6 & 23 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Vidíme, že hodnost $h_A = 2$ a také $h_R = 2$. Podle Frobeniovy věty má soustava alespoň jedno řešení. Když se podíváme na počet proměnných, tak zjistíme, že je jich větší počet než-li jsou hodnoty hodností h_A a h_R. Tedy

$$n = 5 > 2 = h = h_A = h_R\,.$$

Zadaná soustava má tedy nekonečně mnoho řešení závislých na $n - h = 5 - 2 = 3$ parametrech. Proměnné x_3, x_4 a x_5 označíme po řadě parametry t, p a r. Hodnoty zbývajících dvou proměnných dopočítáme ze zbývajících rovnic. Ekvivalentní soustavu lineárních rovnic přepíšeme zpět do rovnic:

$$\begin{aligned} x_1 - x_3 - x_4 - 5x_5 &= -16 \\ x_2 + 2x_3 + 2x_4 + 6x_5 &= 23. \end{aligned}$$

Metodou zpětného dosazování a na základě Frobéniovy věty obdržíme všechna řešení soustavy

$$\vec{x} = \begin{pmatrix} -16 + t + p + 5r \\ 23 - 2t - 2p - 6r \\ t \\ p \\ r \end{pmatrix}, \; t, p, r \in \mathbb{R}.$$

PŘÍKLAD 28.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 - x_2 + x_3 &= \frac{1}{6} \\ 2x_1 - x_2 + x_3 &= \frac{1}{6} \\ 2x_1 - 4x_2 + x_3 &= -\frac{4}{3} \end{aligned}$$

Řešení je $\begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{2}{3} \end{pmatrix}$.

PŘÍKLAD 29.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 - x_2 + 5x_4 &= -13 \\ -x_1 - 2x_2 + 3x_3 &= -5 \\ -x_1 + 2x_2 - x_3 &= -1 \\ x_1 + x_3 + x_4 &= 2 \end{aligned}$$

Řešení je $\begin{pmatrix} 4 \\ 2 \\ 1 \\ -3 \end{pmatrix}$.

PŘÍKLAD 30.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 + 2x_2 + 5x_3 &= 7 \\ -x_1 - x_2 - 3x_3 - x_4 &= -6 \\ -x_1 + 2x_2 + 3x_3 - x_4 &= 0 \end{aligned}$$

Řešení je $\begin{pmatrix} 3 - t \\ 2 - 2t \\ t \\ 1 \end{pmatrix} \; t \in \mathbb{R}.$

PŘÍKLAD 31.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 + 2x_2 - x_3 - x_4 &= 6 \\ -x_1 - 2x_2 + 3x_3 + 3x_4 &= -2 \\ x_1 - 2x_2 + x_3 + 5x_4 &= 2 \\ x_1 + 2x_2 - 5x_3 - 5x_4 &= -2 \end{aligned}$$

Řešení je $\begin{pmatrix} 4-2t \\ 2+t \\ 2-t \\ t \end{pmatrix}$ $t \in \mathbb{R}$.

PŘÍKLAD 32.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 &= 7 \\ x_2 + 2x_3 + x_4 + 2x_5 &= 2 \\ -x_1 + 2x_2 + 2x_3 + x_4 &= 1 \\ x_1 + 3x_2 + 6x_3 + 3x_4 + 9x_5 &= 9 \end{aligned}$$

Řešení je $\begin{pmatrix} 3-3t \\ 2-t \\ -\frac{1}{2}t - \frac{1}{2}p \\ p \\ t \end{pmatrix}$ $t, p \in \mathbb{R}$.

PŘÍKLAD 33.

Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} 2x_1 + 3x_2 + 6x_3 + 4x_4 + 9x_5 &= -1 \\ 4x_1 - 2x_2 - 7x_3 - 3x_4 + 2x_5 &= -10 \\ 4x_1 - 4x_2 - 9x_3 - 3x_4 - 2x_5 &= -12 \\ -10x_1 + x_2 + 8x_3 + 2x_4 - 13x_5 &= 21 \end{aligned}$$

Řešení je $\begin{pmatrix} -2 - \frac{3}{2}t - \frac{1}{2}p \\ 1 - 2t + p \\ -p \\ p \\ t \end{pmatrix}$ $t, p \in \mathbb{R}$.

K PROCVIČENÍ

1. Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 + 2x_2 + x_4 + 3x_5 - 2x_6 &= -4 \\ 3x_2 - 4x_3 + 2x_4 + x_5 + 5x_6 &= 21 \\ x_1 + x_2 - 3x_5 - 2x_6 &= -8 \\ -x_1 + 2x_2 + 3x_3 + 5x_5 + 6x_6 &= 0 \\ 3x_1 + 4x_2 + x_4 - 3x_5 - 6x_6 &= -20 \end{aligned}$$

Část II

Matlab

KAPITOLA

9

STRUČNÝ ÚVOD DO PROGRAMU MATLAB

NAUČÍME SE

Seznámíme se s uživatelským rozhraním programu Matlab. Vysvětlíme základní příkazy pro práci s proměnnými a popíšeme proměnnou typu číslo a vektor. Také si ukážeme práci s logickou proměnou, logické operace a relační operátory.

VÝKLAD UČIVA

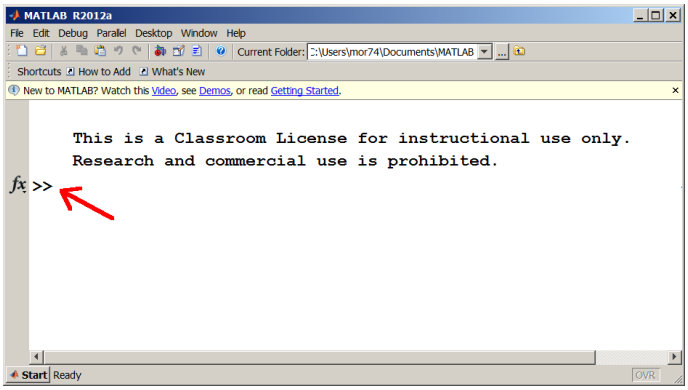
9.1 Základy práce s Matlabem

9.1.1 Základní informace

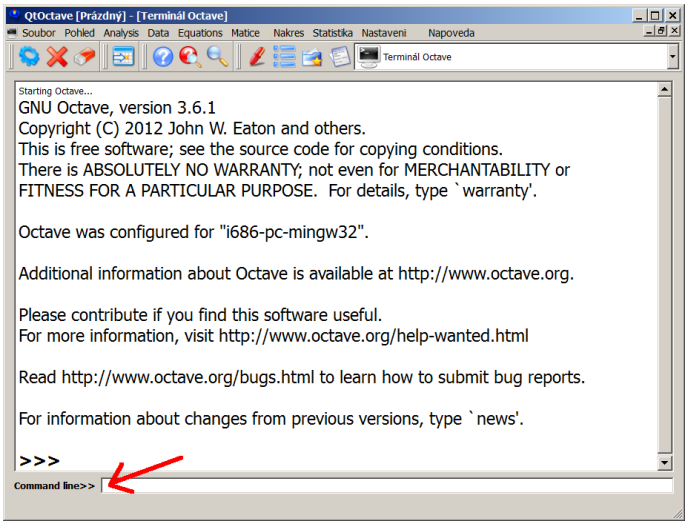
Matlab je programové prostředí a skriptovací programovací jazyk pro technické výpočty. Program není volně k dispozici, ale lze použít program Octave, který je volně dostupnou alternativou k programu Matlab.
Podrobný popis instalace a odkazy na instalační soubory najdete na stránkách předmětu http://mdg.vsb.cz/wiki/index.php/0584_MNPAZP.

9.1.2 Uživatelské rozhraní

Po spuštění programu uvidíte následující okno (obrázky 9.1, 9.2).



Obrázek 9.1: Obrazovka programu Matlab



Obrázek 9.2: Obrazovka programu Octave

9.2 První seznámení s Matlabem

PŘÍKLAD 34.

Př. Zkusíme spočítat 5+2. Klikneme do místa označeného červenou šipkou, zapšeme 5+2 a stiskneme klávesu *Enter*.

```
>> 5+2
ans =
    7
```

9.2.1 Nápověda

Podrobnější nápovědu lze najít v menu Help (Nápověda) nebo pomocí příkazů.

`lookfor text`	nalezení zadaného řetězce `text` v nápovědě
`help text`	nápověda k příkazu `text`

PŘÍKLAD 35.

Př. Najdeme příkazy v jejichž popisu se vyskytuje slovo `product`:

```
>> lookfor product
KRON   Kronecker tensor product.
CROSS  Vector cross product.
DOT  Vector dot product.
CUMPROD Cumulative product of elements.
PROD Product of elements.
```

Př. Vypíšeme nápovědu pro příkaz `length`:

```
>> help length
 LENGTH   Length of vector.
    LENGTH(X) returns the length of vector X. It is equivalent
    to MAX(SIZE(X)) for non-empty arrays and 0 for empty ones.
 Overloaded methods
    help serial/length.m
```

9.3 Proměnné

9.3.1 Jména proměnných

V Matlabu se ve jménech proměnných rozlišují velká a malá písmena (tj. `Abc, abc, abC, ABC` jsou různé proměnné), jméno proměnné může obsahovat písmena, číslice a podtržítka (_), maximálně však 31 znaků. První znak musí být písmeno. Pokud chybí přiřazení, zavede se automaticky proměnná `ans`. Pro přiřazení se používá symbol rovnítko (=).
Příkazy můžeme psát po jednom na řádek, nebo více příkazů na řádek oddělené čárkou (,). Symbol středník (;) slouží k potlačení výstupu, tj. příkaz se vykoná, ale nezobrazí se výsledek. Chceme-li znát hodnotu dané proměnné, zjistíme ji napsání jména proměnné.

PŘÍKLAD 36.

Př. Spočítáme hodnotu $55 + 17$, výsledek se automaticky uloží do proměnné `ans`:

```
>> 55+17
ans =
    72
```

Př. Do proměnné a přiřadíme hodnotu 12 a do proměnné b druhou mocninu a^2. Příkazy oddělíme čárkou:

```
>> a=12, b=a^2
a =
    12
b =
   144
```

Př. Do proměnné x přiřadíme hodnotu 11, do proměnné y druhou mocninu x^2 a do proměnné z rozdíl $100 - y$. Příkazy (kromě posledního) ukončíme středníky a tím potlačíme výstup na obrazovku:

```
>>  x=11; y=x^2; z=100-y
z =
    -21
```

Př. Do proměnné *velkeC* přiřadíme hodnotu 129.043 a do proměnné *Vysledek* hodnotu $\frac{1}{velkeC-1000}$. Všechny příkazy ukončíme středníky a tím potlačíme výstup na obrazovku. Chceme-li znát hodnotu proměnné *Vysledek*, zjistíme ji napsání jména proměnné:

```
>>  velkeC=129.043; Vysledek=1/(velkeC-1000);
>>  Vysledek
Vysledek =
    -0.0011
```

9.3.2 Příkazy pro práci s proměnnými

Pro různé formáty výpisu, vypsání informací o proměnných nebo jejich smazání nám poslouží tyto příkazy.

`format`	formátování výstupu (`long`), `short`, `rat`
`who`	výpis seznamu proměnných
`whos`	výpis podrobných informací o proměnných
`clear`	smazání proměnné

PŘÍKLAD 37.

Př. Příkazem `format long` nastavíme výpis čísel na 14 desetinných míst, příkazem `format short` na 4 místa. Mění se pouze formát výstupu na obrazovku, přesnost vnitřních výpočtů se nemění.

```
>> a=3.4598392134
a =
    3.4598
>> format long
>> a
a =
   3.45983921340000
>> format short
>> a
```

```
a =
    3.4598
```

Př. Příkazem `format rat` zvolíme výpis ve tvaru zlomků:

```
>> format rat
>> 100/30
ans =
    10/3
>> format short
>> 100/30
ans =
    3.3333
```

Př. Nadefinujeme proměnné a, x, b, r a příkazem `who` si vypíšeme seznam existujících proměnných:

```
>> a=[ 2 4 5; 2 3 4]; x=34; b=[ 4 5  8 89 9 9 9]; r='ahoj'
>> who
Your variables are:
a  b  x
```

Pomocí příkazu `whos` zjistíme podrobné informace o proměnných, jejich jméno (`name`), rozměr (`size`), velikost v bytech a typ proměnné (`class`):

```
>> whos
  Name       Size             Bytes  Class
  a          2x3                 48  double array
  b          1x7                 56  double array
  r          1x4                  8  char array
  x          1x1                  8  double array
Grand total is 18 elements using 120 bytes
```

Př. Příkazem `clear` smažeme všechny existující proměnné nebo můžeme smazat pouze vybrané proměnné.
Máme-li nadefinované proměnné stejně jako v předchozím příkladě, pak smažeme proměnnou `x` příkazem `clear x`:

```
>> clear x
>> who
Your variables are:
a  b  r
```

9.4 Práce s čísly

Matlab pracuje s typem *obdélníková matice*. Vektory jsou matice typu $1 \times n$ nebo $n \times 1$. Číslo je čtvercová matice matice typu 1. Pro začátek se naučíme pracovat s čísly a vektory.

9.4.1 Reálná čísla

Zadávání čísel

Reálná čísla zadáváme s desetinnou tečkou (`.`), čísla lze také zadávat v exponenciálním tvaru například číslo 0.000014 zadáme takto `1.4e-5`, číslo 532300 takto `53.23E4`. Pro exponent můžeme používat symbol `E` nebo `e`. Při zadávání nesmíme dělat v čísle mezeru.

PŘÍKLAD 38.

Př. Do proměnné x přiřadíme hodnotu 2.34 a do r hodnotu 0.0000532:

```
>> x=2.34
x =
    2.3400
>> r=0.532e-4
r =
  5.3200e-005
```

Př. Třemi způsoby přiřadíme hodnotu 123 000 do proměnné a:

```
>> a=123000
a =
       123000
>> a=123e3
a =
       123000
>> a=1.23e5
a =
       123000
```

Př. Při zadávání nesmíme dělat v čísle mezeru. Do proměnné a chybně přiřadíme hodnotu 12 540, poté ji zadáme správně:

```
>> a=12 540
??? a=12 540
         |
Error: Missing operator, comma, or semicolon.
>> a=12540
a =
        12540
```

Operace s čísly

Pro operaci s čísly používáme následující symboly.

Jak jsme zvyklí z matematiky, provádějí se operace v předepsaném pořadí, nejprve operace umocnění, pak násobení a dělení a nakonec sčítání a odčítání. Například, když napíšeme $5 + 7 \cdot 2$, tak se nejprve provede násobení $7 \cdot 2 = 14$ a poté se provede sčítání $5 + 14 = 19$.

Tabulka 9.1: Operace a priorita operací

sčítání	+
odčítání	-
násobení	*
dělení	/
mocnina	^

operace	priorita
1.	^
2.	* /
3.	+ -

Pokud chceme změnit pořadí v jakém se bude výraz počítat, například chceme-li spočítat $(5+7)\cdot 2$, použijeme závorky. Takže se nejprve provede sčítání $5+7=12$ a pak násobení $12\cdot 2=24$.
Stejně tak v Matlabu platí stejná priorita operací a používáme kulaté závorky (žádné jiné pro tento účel nelze použít).

PŘÍKLAD 39.

Př. Vyčíslíme výraz $4^2-7\cdot 12$:

```
>> 4^2-7*12
ans =
   -68
```

Př. Spočítáme hodnotu $\frac{111+171}{57-10}$. Ručně budeme počítat takto: $\frac{111+171}{57-10}=\frac{282}{47}=6$. Na počítači zadáme:

```
>> (111+171)/(57-10)
ans =
     6
```

Vidíme, že jak čitatele $111+171$ tak jmenovatele $57-10$ jsme dali do závorek, aby se nejprve provedlo sčítání, odčítání a pak dělení. Ukážeme si, co by se stalo, kdybychom tak neučinili:

```
>>  111+171/57-10
ans =
   104
```

a vidíme, že je výsledek špatně. Matlab spočítal `111+171/57-10=111+3-10=104`.

Př. Vypočítáme výraz $\frac{1+a^5}{3(a-b)}-\sqrt[3]{b}$ pro $a=2$, $b=13$:

```
>> a=2;b=13;x=(1+a^5)/(3*(a-b))-b^(1/3)
x =
   -3.3513
```

Př. Spočítáme hodnotu 2^{3^2}. Ručně budeme počítat takto: $2^{3^2}=2^9=512$. Na počítači zadáme:

```
>> 2^3^2
ans =
    64
```

a vidíme, že se výsledky liší. Kde je chyba? Chyba je ve špatné prioritě, Matlab spočítal `2^3^2=8^2=64`.
Správně výpočet zadáme takto:

```
>> 2^(3^2)
ans =
   512
```

9.4.2 Zaokrouhlování

Čísla můžeme zaokrouhlit několika způsoby.

`round(x)`	zaokrouhlí číslo x na celé číslo
`floor(x)`	zaokrouhlí číslo x na nejbližší menší celé číslo (dolů)
`ceil(x)`	zaokrouhlí číslo x na nejbližší větší celé číslo (nahoru)
`fix(x)`	zaokrouhlí číslo x na nejbližší celé číslo směrem k nule

Funkce pro zaokrouhlování můžeme použít i pro matice, funkce se pak vyčíslí pro jednotlivé prvky matice.

PŘÍKLAD 40.

Př. Zaokrouhlíme číslo 12.567:

```
>> round(12.567)
ans =
    13
```

Př. Všimněme si, že `fix` odřízne desetinnou část čísla, tato funkce se nazývá *celá část čísla*. Chceme-li získat i *desetinnou část čísla*, odečteme od daného čísla jeho celou část:

```
>> a=162.3409; desetinna=a-fix(a), cela=fix(a)
desetinna =
    0.3409
cela =
   162
```

9.5 Vektory

Vektory zadáváme do hranatých závorek a jednotlivé prvky oddělujeme čárkou nebo mezerou. K jednotlivým prvkům přistupujeme pomocí kulatých závorek.

`length(v)`	počet prvků vektoru v

PŘÍKLAD 41.

Př. Zadáme vektor $\vec{u} = (1, 8, -3.2, -1.3, 0.4)$, spočítáme počet jeho prvků a vypíšeme jeho čtvrtý prvek:

```
>> u=[1 8 -3.2 -1.3 0.4]
u =
    1.0000    8.0000   -3.2000   -1.3000    0.4000

>> length(u)
ans =
     5

>> u(4)
ans =
   -1.3000
```

9.5.1 Vygenerování aritmetické posloupnosti

Potřebujeme-li vygenerovat vektor čísel, která jsou prvky aritmetické posloupnosti, máme v Matlabu k dispozici příkaz `linspace` nebo použijeme „dvojtečku“.

`prvni_clen:diference:posledni_clen`
`linspace(prvni_clen,posledni_clen,pocet_clenu)`

Př. Potřebujeme vektor (4, 7, 10, 13, 16, 19, 22) tedy čísla od 4 do 22 s krokem 3.

```
>> 4:3:22
ans =
     4     7    10    13    16    19    22
```

Př. Pokud se diference vynechá, Matlab ji bere jako rovnu 1.

```
>> 2:10
ans =
     2     3     4     5     6     7     8     9    10
```

Př. Diference může být i záporné číslo.

```
>> 13:-3:0
ans =
    13    10     7     4     1
```

Př. Diference může být i desetinné číslo.

```
>> 5:0.2:6
ans =
    5.0000    5.2000    5.4000    5.6000    5.8000    6.0000
```

Př. Příkazem `linspace` vytvoříme 9 čísel od 3 do 6.

```
>> linspace(3,6,9)
ans =
    3.0000    3.3750    3.7500    4.1250    4.5000    4.8750

    5.2500    5.6250    6.0000
```

9.6 Práce s daty

Pro výpočet součtu, součinu nebo nalezení největšího, nejmenšího čísla mezi prvky vektoru máme k dispozici funkce.

`max(v)`	maximum mezi prvky vektoru v
`min(v)`	minimum mezi prvky vektoru v
`sum(v)`	součty prvků vektoru v
`prod(v)`	součiny prvků vektoru v
`sort(v)`	setřídí prvky vektoru v vzestupně

Tyto funkce lze použít i pro matice, jak si ukážeme v kapitole 11.

PŘÍKLAD 42.

Př. Spočítáme maximum :

```
>> max([2 5 6 3 0 9 1 8 -1 4 9 2])
ans =
     9
```

9.7 Logická proměnná

9.7.1 Pravda, nepravda

V Matlabu neni speciální typ pro logické proměnné, *pravda* má hodnotu `1` a *nepravda* hodnotu `0`.

9.7.2 Relační operátory

Relační operátory lze použít na čísla, vektory i matice, kde se pak srovnávají prvek po prvku. Výsledkem je jedna (platí) nebo nula (neplatí).

rovnost $=$	`==`
nerovnost $\neq$	`~=`
menší než $<$	`<`
větší než $>$	`>`
menší nebo roven $\leq$	`<=`
větší nebo roven $\geq$	`>=`

PŘÍKLAD 43.

Př. Porovnáme prvky vektoru a, b:

```
>> a=[3 9 4 1 -5 3], b=[4 4 0 9 3 0]
a =
     3     9     4     1    -5     3
b =
     4     4     0     9     3     0

>> a>b
ans =
     0     1     1     0     0     1
```

9.7.3 Logické operátory

Připomeňme si logické operátory negace ($\neg$), konjunkce ($\wedge$), disjunkce ($\vee$) a exkuzivní disjunkce (xor) a jejich zápis v Matlabu.

Vstup		`not(A)` nebo `~A`	`and(A,B)` nebo `A&B`	`or(A,B)` nebo `A|B`	`xor(A,B)` nebo `xor(A,B)`
`A`	`B`				
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

9.7.4 Funkce pro zjišťování platnosti podmínek

Pro zjištění platnosti podmínek máme k dispozici několik příkazů.

`all(podminka)` Platí `podminka` pro všechny prvky?
`any(podminka)` Platí `podminka` aspoň pro jeden prvek?
`find(podminka)` Pro které prvky `podminka` platí?

PŘÍKLAD 44.

Př. Zadáme vektor čísel:

```
>> r=[-1 4 -3 4 -5 -2 7]

r =

    -1     4    -3     4    -5    -2     7
```

Zadáme podmínku na kladnost a dostaneme vektor jedniček (platí) a nul (neplatí):

```
>> r>0

ans =
```

```
     0     1     0     1     0     0     1
```

Zeptáme se, zda jsou všechna čísla kladná.

```
>> all(r>0)

ans =

     0
```

Zeptáme se, zda je některé z čísel kladné.

```
>> any(r>0)

ans =

     1
```

Zeptáme se, která čísla jsou kladná.

```
>> find(r>0)

ans =

     2     4     7
```

K PROCVIČENÍ

1. Najděte čtyři způsoby jak zadat číslo 15 miliónů.
2. Najděte tři způsoby jak zadat číslo 0.007.
3. Vytvořte posloupnost čísel pomocí dvojtečky a pomocí příkazu `linspace`.
   (a) 2, 5, 8, 11, 14, 17, 20
   (b) 15.5, 15.4, 15.3, 15.2, 15.1, 15, 14.9, 14.8
   (c) 220, 205, 190, 175, 160, 145, 130, 115, 100, 85
4. Spočítejte průměr prvků daného vektoru.
5. Je dán vektor. Zjistěte, které jeho prvky jsou menší než jejich průměr.

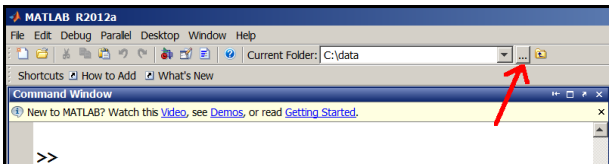
KAPITOLA

10

PROGRAMOVÁNÍ V MATLABU

10.1 Než začneme

První a velmi důležitý krok je nastavit pracovní adresář (složku). Budeme například pracovat ve složce **c:\data**. V Matlabu klikneme v Menu na ikonu ... (viz obrázek 10.1) a vybereme naši složku.



Obrázek 10.1: Změna pracovní složky v Matlabu

Poznámka: V Octave změníme pracovní složku v Menu **Soubor - Změnit adresář** a vybereme naši složku.

10.2 Skripty

S pojmem skript jsme se seznámili při práci s GeoGebrou. Je to posloupnost příkazů, které se vykonají.
V Matlabu se skript vytvoří tak, že v Menu **File - New - Skript** otevřeme Editor a do něj napíšeme příkazy. Poté soubor uložíme pod jménem (neobsahující české znaky nebo mezery) a s příponou `m`.
Skript se pustí kliknutím na ikonu v editoru Matlabu.
Poznámka: V Octave se editor otevře v Menu **Pohled - Nástroje panelu - Editor**. A skript se pustí kliknutím na ikonu v Menu editoru.

PŘÍKLAD 45.

Napíšeme do editoru následující jednoduché příkazy:

```
1 a=15
2 b=45
3 c=a+b
```

Skript uložíme do adresáře, například **c:\data** pod jménem `priklad.m`.
Spustí ho kliknutím na ikonu v editoru Matlabu. Příkazy se vykonají.

10.3 Vstupy a výstupy

Práci se vstupy a výstupy nám usnadí tyto příkazy:

`disp('text')`	vypíše text
`disp(promenna)`	vypíše hodnotu proměnné
`promenna = input('text')`	vstup z klávesnice, který se načte do proměnné
`error('text')`	chybová hláška

10.4 Programovací struktury

rozhodovací blok

```
if podmínka 1
    blok příkazů 1
end
```

rozhodovací blok

```
if podmínka 1
    blok příkazů 1
elseif podmínka 2
    blok příkazů 2
...
else
     blok příkazů 3
end
```

PŘÍKLAD 46.

Př. Zjistíme, zda je číslo x kladné nebo není:

```
>> x=3

x =

     3
```

```
>> if x>0 disp('je kladne'), else disp('neni kladne'), end
 je kladne

>> x=-5

x =

    -5

>> if x>0 disp('je kladne'), else disp('neni kladne'), end
 neni kladne
```

cyklus se známým počtem opakování

`for` *řídící proměnná=rozsah hodnot*
    blok příkazů
`end`

cyklus s podmínkou

`while` *podmínka*
    blok příkazů
`end`

PŘÍKLAD 47.

Př. Spočítáme faktoriál čísla 8:

```
>> fakt=1
fakt =

     1

>> for i=2:8, fakt=fakt*i, end
fakt =
     2

fakt =
     6

fakt =
    24

fakt =
```

```
   120

fakt =
   720

fakt =
        5040

fakt =
       40320
```

Př. Budeme postupně sčítat čísla z posloupnosti 1 3 2 4 5 6 3 4 5 3 6 7 8, dokud nebude součet vetší než 20.

```
>> a=[1 3 2 4 5 6 3 4 5 3 6 7 8 ];

>> soucet=0
soucet =
     0

>> i=1
i =
     1

>>  while soucet<20, soucet=soucet+a(i), i=i+1; end
soucet =
     1

soucet =
     4

soucet =
     6

soucet =
    10

soucet =
    15

soucet =
    21
```

10.5 Funkční M–soubory

Nejprve si zkusíme jednoduchý příklad.

PŘÍKLAD 48.

Napíšeme do editoru následující řádky:

```
1 function [c]=secti(a,b)
2 % funkce secte dve cisla
3 c=a+b
```

Funkci uložíme do adresáře, například **c:\data** pod jménem `secti.m`.
Poté v hlavním okně (Command Windows) napíšeme

```
1 >> secti(10,14)
2 ans =
3      24
4
5 >> r=secti(10,14)
6 r  =
7      24
```

Podrobný popis funkce

Funkce je posloupnost příkazů, která má hlavičku obsahující jméno, vstupní a výstupní parametry:

`function[vystupni parametry]=jmeno(vstupni parametry)`

Jméno volíme jednoznačně, stručně a výstižně. Pod tímto jménem je pak nutné funkci uložit do souboru s příponou `m`).
`vstupni parametry` je seznam vstupních parametrů, v kulatých závorkách, oddělujeme je čárkou.
`vystupni parametry` je seznam výstupních parametrů, v hranatých závorkách, oddělujeme je čárkou.
Seznamy parametrů mohou být i prázdné.

Funkci je vhodné opatřit komentářem, ten začíná znakem % na začátku řádku. První komentářový řádek obsahuje stručný popis funkce, v tomto řádku hledá příkaz `lookfor`. Druhý komentářový řádek je tvar volání funkce, tj. vstupní a výstupní parametry ve správném pořadí. Další komentářové řádky obsahují popis všech parametrů funkce a podrobnější informace.

Volání funkce

Funkci voláme jejím jménem a vstupními parametry, které mají v okamžiku volání hodnotu.

PŘÍKLAD 49.

Př. Sestavte funkci, která spočítá minimum, průměr a maximum ze vstupního vektoru.

```
function [minimum,prumer,maximum]=vlastnosti_vektoru(v);
% Vypocet minima, prumeru a maxima
%            [minimum,prumer,maximum]=vlastnosti_vektoru(v);
%  VSTUP
%       v ... vektor cisel
%  VYSTUPY
%  minimum... minimalni hodnota mezi prvky v
%  maximum... maximalni hodnota mezi prvky v
%  prumer ... prumerna hodnota mezi prvky v
%
n=length(v);
minimum=min(v);
maximum=max(v);
prumer=sum(v)/n;
```

Funkci zavoláme pro vektor (1,4,3):

```
>> vlastnosti_vektoru([1,4,3])
```

10.6 Úloha „Vedoucí prodejny“

ZADÁNÍ ŘEŠENÉ ÚLOHY

Vedoucí prodejny (která má 8 pokladen) si zapisuje čísla pokladen, které obslouží zákazníka. Zajímá ho, která pokladna neobsoužila žádného zákazníka a která obsloužila nejvíce zákazníků.
Poznámka: Upravte program, neznáte-li počet pokladen.

POSTUP ŘEŠENÍ

Zadáme vstupní data

```
>> x=[1 2 1 3 3 5 6 1 2 5 8 8 2 2 1 3 2 1]
```

Zjistíme nejprve kolik obsloužila jedna pokladna, například pátá. Dostaneme vektor jedniček a nul, podle toho, zda na příslušné pozici je nebo není 5.

```
>> x==5
ans =

   0   0   0   0   0   1   0   0   0   1   0   0   0   0   0   0
0   0
```

Sečteme počty jedniček, tedy spočítáme kolik zakazníků obsloužial pátá pokladna.

```
>>sum(x==5)
ans =
     2
```

Pomocí cyklu zjistíme počtu u všech osmi pokladen.

```
>> for i=1:8, p(i)=sum(x==i), end
p =
   5
p =

   5   5

p =

   5   5   3

p =

   5   5   3   0

p =

   5   5   3   0   2

p =

   5   5   3   0   2   1

p =

   5   5   3   0   2   1   0

p =

   5   5   3   0   2   1   0   2
```

Vypíšeme si udaje do tabulky.

```
>> tabulka=[1:8; p]
tabulka =

   1   2   3   4   5   6   7   8
   5   5   3   0   2   1   0   2
```

Zjistíme, které pokladny obsloužily nejvíce zakaníků.

```
>> nejvice=find(p==max(p))
```

```
nejvice =

   1   2
```

Z které neobsloužily žádného.

```
>> zadny=find(p==0)

zadny =

   4   7
```

Příkazy zapíšeme jako funkci. Vstupním parametrem bude vektar dat `x` a výstupy budou: `tabulka`, čísla pokladen, které obsloužily nejvíce zákazníků `nejvice` a čísla pokladen, které neobsloužily žádného `zadny`.

```
function [tabulka,nejvice,zadny]=pokladny(x);

for i=1:8
        p(i)=sum(x==i)
end

tabulka=[1:8; p]

nejvice=find(p==max(p))

zadny=find(p==0)
```

Nyní upravíme program tak, aby mohl zpracovat data nejen pro 8 pokladen, ale pro libovolný počet. Počet pokladen bude roven největšímu z čísel v datech `x`, a označíme ho `n`.

```
function [tabulka,nejvice,zadny]=pokladny(x);
n=max(x)
for i=1:n
        p(i)=sum(x==i)
end

tabulka=[1:n; p]

nejvice=find(p==max(p))

zadny=find(p==0)
```

K PROCVIČENÍ

1. Vrátný má dva seznamy, v jednom má hodinu příchodu a v druhém hodinu odchodu zaměstance. Spočítejte počty hodin, které jednotlicí zaměstanci strávili v práci. A dále

zjistěte kolik zaměstnanců bylo v práci déle než 8 hodin a který (kteří) byli v práci nejdéle.

joooo... jsem dobrý

KAPITOLA

11 MATICE

NAUČÍME SE

Naučíme se vytvořit matici a pracovat s jejími částmi. Ukážeme si maticové operace a operace „prvek po prvku“.

VÝKLAD UČIVA

11.1 Matice a vektory

11.1.1 Zadávání matic a vektorů

Matici zadáváme v hranatých závorkách, prvky na řádku oddělujeme mezerou nebo čárkou. Jednotlivé řádky pak středníkem nebo klávesou *Enter*.

PŘÍKLAD 50.

Př. Zadáme matici $A = \begin{pmatrix} 3 & 5 \\ 0 & 9 \\ -3 & 2 \end{pmatrix}$, prvky na řádku oddělíme mezerou a jednotlivé řádky středníkem:

```
>> A=[3 5; 0 9; -3 2]
A =
     3     5
     0     9
    -3     2
```

Matici lze vytvořit spojením jiných matic nebo vektorů, se kterými zacházíme jako s prvky. Matice „vedle sebe“ jsou jako prvky na řádku a matice „nad sebou“ jsou jako řádky v matici.

Př. Pomocí matic $\begin{pmatrix} 4 & 5 \\ 6 & 2 \end{pmatrix}$ a $\begin{pmatrix} 8 \\ 0 \end{pmatrix}$ vytvoříme matici $\begin{pmatrix} 4 & 5 & 8 \\ 6 & 2 & 0 \end{pmatrix}$. Umístíme tedy matice vedle sebe, oddělíme je mezerou jako prvky na řádku:

```
>> a=[4 5; 6 2], b=[8;0]
a =
     4     5
     6     2
b =
     8
     0
>> c=[a b]
c =
     4     5     8
     6     2     0
```

Př.

Pomocí vektorů $\vec{v} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ a $\vec{u} = \begin{pmatrix} 2 & 3 & 4 \end{pmatrix}$ vytvoříme matici $\begin{pmatrix} 0 & 0 & 1 \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix}$. Umístíme tedy vektory nad sebe, oddělím je středníkem jako řádky v matici, a to v pořadí $\vec{v}, \vec{u}, \vec{v}$:

```
>> u=[2 3 4]
u =
     2     3     4
>> v=[0 0 1]
v =
     0     0     1
>> [v;u;v]
ans =
     0     0     1
     2     3     4
     0     0     1
```

Př.

K matici $A = \begin{pmatrix} 3 & -8 & 67 & 1 \\ 1 & 0 & 0 & 23 \end{pmatrix}$ přidáme jako poslední řádek čísla $\begin{pmatrix} 1 & 2 & -9 & 12 \end{pmatrix}$. Umístíme tedy vektor pod matici, oddělíme je od sebe středníkem:

```
>> A=[3 -8 67 1; 1 0 0 23]
A =
     3    -8    67     1
     1     0     0    23
>> A=[A; 1 2 -9 12]
A =
     3    -8    67     1
     1     0     0    23
     1     2    -9    12
```

11.1.2 Funkce pro tvorbu matic

Matici se speciálními prvky lze vytvořit i pomocí následujících příkazů.

`zeros(m,n)`	nulová matice typu $m \times n$
`ones(m,n)`	matice jedniček typu $m \times n$
`eye(m,n)`	jednotková matice typu $m \times n$
`rand(m,n)`	matice náhodných čísel z intervalu $(0,1)$ typu $m \times n$

Pokud se použijí funkce `zeros`, `ones`, `eye`, `rand` pouze s jedním parametrem (například `zeros(3)`), vznikne čtvercová matice příslušného řádu.

PŘÍKLAD 51.

Př. Vygenerujeme čtvercovou matici řádu 2 náhodných čísel z intervalu $(0,1)$:

```
>> rand(2)
ans =
    0.5252    0.6721
    0.2026    0.8381
```

Př. Vygenerujeme matici typu 2×5 náhodných čísel z intervalu $(0,100)$.
Příkaz `rand(2,5)` vygeneruje matici typu 2×5 čísel z intervalu $(0,1)$ a když tuto matici vynásobíme číslem 100, získáme požadovanou matici:

```
>> 100*rand(2,5)
ans =
    1.9640   37.9481   50.2813   42.8892   18.9654
   68.1277   83.1796   70.9471   30.4617   19.3431
```

Př. Vygenerujeme čtvercovou matici řádu 3 náhodných celých čísel z intervalu $(0,10)$. Příkaz `rand(3)` vygeneruje čtvercovou matici řádu 3 čísel z intervalu $(0,1)$ a když tuto matici vynásobíme číslem 10, získáme matici čísel z intervalu $(0,10)$, po zaokrouhlení příkazem `round` získáme požadovanou matici celých čísel:

```
>> round(10*rand(3))
ans =
     0    10     5
     8     8     2
    10     4     6
```

Př. Do proměnné `hody` vygenerujeme 10 náhodných čísel, které reprezentují hody kostkou, tj. jsou to čísla 1, 2 ... 6. Příkaz `rand(1,10)` vygeneruje vektor 10 čísel z intervalu $(0,1)$ a když jej vynásobíme číslem 6, získáme matici čísel z intervalu $(0,6)$, po zokrouhlení nahoru (neboť nechceme mít mezi výslednými čísly nulu) příkazem `ceil` získáme požadovaný vektor:

```
>> hody=ceil(6*rand(1,10))
hody =
     4     2     2     1     5     3     6     3     3     6
```

11.1.3 Práce s částmi matic a vektorů

Matlab umí pracovat s jednotlivými prvky, řádky, sloupci matice nebo se submaticemi.

PŘÍKLAD 52.

Př. V tomto příkladě si ukážeme práci s jednotlivými častmi (jsou označeny šedě) této matice:

$$\begin{pmatrix} 9 & 4 & 9 & 4 & 1 & 0 & 5 & 2 \\ 2 & 0 & 7 & 8 & 2 & 7 & 5 & 1 \\ 6 & 8 & 1 & 0 & 1 & 4 & 8 & 0 \\ 4 & 4 & 4 & 3 & 6 & 9 & 3 & 7 \\ 8 & 6 & 9 & 8 & 2 & 4 & 1 & 8 \\ 7 & 7 & 9 & 0 & 1 & 4 & 3 & 8 \\ 4 & 6 & 1 & 0 & 0 & 3 & 2 & 1 \end{pmatrix}$$

Nejprve si matici zadáme:

```
>> A=[9 4 9 4 1 0 5 2;2 0 7 8 2 7 5 1;6 8 1 0 1 4 8 0;
4 4 4 3 6 9 3 7;8 6 9 8 2 4 1 8;7 7 9 0 1 4 3 8;4 6 1 0 0 3 2 1]
```

Vypíšeme první řádek matice A:

```
>> A(1,:)
ans =
     9     4     9     4     1     0     5     2
```

A sedmý sloupec:

```
>> A(:,7)
ans =
     5
     5
     8
     3
     1
     3
     2
```

Prvek $a_{3,3}$:

```
>> A(3,3)
ans =
     1
```

A vyznačené submatice:

```
>> A(3:6,1)
ans =
     6
     4
     8
     7
>> A(4:7,3:5)
ans =
     4     3     6
     9     8     2
     9     0     1
```

Celou matici A:

```
>> A
A    =

     9     4     9     4     1     0     5     2
     2     0     7     8     2     7     5     1
     6     8     1     0     1     4     8     0
     4     4     4     3     6     9     3     7
     8     6     9     8     2     4     1     8
     7     7     9     0     1     4     3     8
     4     6     1     0     0     3     2     1
```

PŘÍKLAD 53.

Př. V matici *B* přepíšeme prvky v druhém sloupci na čísla 10:

```
>> B=[2 3 1 6;5 0 7 9;3  2 1 4]
B =
     2     3     1     6
     5     0     7     9
     3     2     1     4
>> B(:,2)=10
B =
     2    10     1     6
     5    10     7     9
     3    10     1     4
```

Smažeme první řádek, tak že ho nahradíme prázdnou matici `[ ]`:

```
>> B(1,:)=[ ]
B =
     5    10     7     9
     3    10     1     4
```

Př. Z matice C vypíšeme jen sudé sloupce:

```
>> C=[0.5 2 0.45 2.4 9 0 4.9 ;9 3.4 5.2 8 9 0 1]
C =
 0.5000    2.0000    0.4500    2.4000    9.0000         0    4.9000
 9.0000    3.4000    5.2000    8.0000    9.0000         0    1.0000
>> C(:,1:2:end)
ans =
    0.5000    0.4500    9.0000    4.9000
    9.0000    5.2000    9.0000    1.0000
```

11.1.4 Operace s maticemi a vektory

Pro operaci s maticemi a vektory používáme následující symboly:

`+`	součet matic (např. `A+B` je matice s prvky $a_{ij} + b_{ij}$)
`-`	rozdíl matic (např. `A-B` je matice s prvky $a_{ij} - b_{ij}$)
`*`	součin matic „řadek krát sloupec“
`/`	pravé maticové dělení (např. `A/B` je matice $A \cdot B^{-1}$)
`\`	levé maticové dělení (např. `A\B` je matice $A^{-1} \cdot B$)
`^`	mocnina matic (např. `A^k` je $A \cdot A \cdot \ldots \cdot A$ (k-krát))
`'`	transponovaná matice (např. `A'` je matice s prvky a_{ji})

Z matematiky známe součin matic, který se provádí tzv. „řádek krát sloupec“(například $A \cdot B$). Dále známe součin čísla a matice (například $c \cdot A$). Matlab používá symbol hvězdička $*$ pro tyto operace. Připomeňme, že násobit můžeme pouze matici typu $m \times n$ krát matici typu $n \times p$ a výsledkem násobení je matice typu $m \times p$.

$$A_{m\times n} * B_{n\times p} = C_{m\times p}$$

`.*`	součin matic „prvek po prvku“ (např. `A.*B` je matice s prvky $a_{ij}b_{ij}$)
`./`	pravé dělení „prvek po prvku“ (např. `A./B` je matice s prvky a_{ij}/b_{ij})
`.\`	levé dělení „prvek po prvku“ (např. `A.\B` je matice s prvky b_{ij}/a_{ij})
`.^`	mocnina (např. `A.^k` je matice s prvky $(a_{ij})^k$)

Symbolem tečka a hvězdička `.*` označuje Matlab tzv. součin „prvek po prvku“, který se počítá tak, že se vynásobí prvky obou matic na stejných pozicích. Takto můžeme násobit pouze matice stejných typů.

$$A_{m\times n} .* B_{m\times n} = C_{m\times n}$$

PŘÍKLAD 54.

Př. Spočítáme $A + B$, $A - B$, $-4A$ $A \cdot A = A^2$

```
>> A=[2 3; 0 9],B=[1 0; -1 5]
A =
     2     3
     0     9
```

```
B =
      1      0
     -1      5

>> A+B
ans =
      3      3
     -1     14

>> A-B
ans =
      1      3
      1      4
>> -4*A
ans =
     -8    -12
      0    -36
>> A^2
ans =
      4     33
      0     81
```

Př. Na maticích A, B z předchozího příkladu si ukážeme násobení matic `A*B` a násobení „prvek po prvku“ `.*`, kdy se spolu vynásobí prvky na stejných pozicích. Porovnejte výsledky.

```
>> A*B
ans =
     -1     15
     -9     45
>> A.*B
ans =
      2      0
      0     45
```

11.1.5 Počet prvků, rozměr matice

Občas je potřeba zjistit rozměr matice nebo počet jejich prvků.

`size(A)`	typ matice A (počet řádků, počet sloupců)
`numel(A)`	počet prvků A

PŘÍKLAD 55.

Př. Určíme typ matice a počet jejich prvků:

```
>> D=[1  2  5;  0  4  3]
```

```
D =
     1     2     5
     0     4     3
>> [radky,sloupce]=size(D)
radky =
     2
sloupce =
     3
>> numel(D)
ans =
     6
```

11.1.6 Manipulace s maticemi

`reshape(A,m,n)`	změna typu matice A na typ $m \times n$
`diag(A)`	hlavní diagonála matice A
`diag(v)`	matice, která má na hlavní diagonále vektor $\vec{v}$, a jinde nuly

PŘÍKLAD 56.

Př. Vytvoříme matici typu 3×2 z vektoru $1, 4, 3, 8, 0, 2$:

```
>> b=reshape([1 4 3 8 0 2],3,2)
b =
     1     8
     4     0
     3     2
```

Př. Přetransformujeme zadanou matici na typ 2×6, vidíme že Matlab matici tvoří po sloupcích:

```
>> mat
mat =
     1     6     4
     5     3     2
     7     5     0
     4     0    -2
>> reshape(mat,2,6)
ans =
     1     7     6     5     4     0
     5     4     3     0     2    -2
```

Př. Pro funkci `reshape` je potřeba zachovávat počet prvků původní matice, zkusíme matici `mat` z předchozího příkladu přetransformovat na typ 3×3:

```
>> reshape(mat,3,3)
??? Error using ==> reshape
To RESHAPE the number of elements must not change.
```

Př. Z matice a do proměnné v vybereme hlavní diagonalu:

```
>> a=[3 5 7; 2 1 23; -5 9 1]
a =
     3     5     7
     2     1    23
    -5     9     1
>> v=diag(a)
v =
     3
     1
     1
```

Př. Vytvoříme matici, která ma na diagonále čísla $2, -1, 4$ a jinde nuly:

```
>> diag([2 -1 4])
ans =
     2     0     0
     0    -1     0
     0     0     4
```

11.2 Úloha „Zemědělec“

Zadání řešené úlohy

Zemědělec má v tabulce uvedeny sklizně (v tunách) jednotlivých plodin během pěti let. Sestavte funkci, která spočítá celkové množství sklizně jednotlivých plodin během pěti let a největší z těchto sklizní. Dále vypíše, která plodina měla v součtu za pět let největší sklizeň a informaci, zda největší sklizeň měla nebo neměla pšenice.
Příklad:

	2007	2008	2009	2010	2011
1 - ječmen	1	3	5	1	5
2 - pšenice	2	1	3	2	6
3 - brambory	1	2	11	12	5
4 - řepka olejka	1	2	11	5	7
5 - pohanka	3	5	9	10	8
6 - hrách	12	2	2	9	10

Výpis programu: `Sklizne plodin (soucet za 2007-2010): 15 14 31 26 35 35`

```
Nejvetsi sklizen: 35
Maji plodiny: 5 6
Nejvetsi sklizen nemela psenice.
```

Postup řešení

Nejprve zadáme do matice `data` vstupní data z tabulky.

```
>> data=[1 3 5 1 5; 2 1 3 2 6;1 2 11 12 5;
       1 2 11 5 7;3 5 9 10 8; 12 2 2 9 10]

data =

     1     3     5     1     5
     2     1     3     2     6
     1     2    11    12     5
     1     2    11     5     7
     3     5     9    10     8
    12     2     2     9    10
```

Do proměnné `sklizen` spočítáme součet hodnot v jednotlivých řádcích matice:

```
>> sklizne=sum(data')

sklizne =

    15    14    31    26    35    35
```

A největší hodnotu z těchto čísel si uložíme do proměnné `nejvetsi`:

```
>> nejvetsi=max(sklizne)

nejvetsi =

    35
```

Do proměnné `pozice` uložíme pozici let s největší sklizní:

```
>> pozice=find(nejvetsi==sklizne)

pozice =

     5     6
```

A nakonec zjistíme zda pšenice měla největší sklizeň:

```
>> psenice=any(pozice==2)

psenice =

     0
```

Sestavíme funkci a uložíme do souboru **zemedelec.m**

```
1 function zemedelec(data)
2 % funkce na zpracovani dat sklizne
3 sklizne=sum(data');
4 disp('Sklizne plodin')
```

```
5  disp(sklizne)
6
7  nejvetsi=max(sklizne);
8  disp('Nejvetsi sklizen')
9  disp(nejvetsi)
10
11 pozice=find(nejvetsi==sklizne);
12 disp('Maji plodiny')
13 disp(pozice)
14
15 if any(pozice==2)
16     disp('Nejvetsi sklizen mela psenice')
17 else
18     disp('Nejvetsi sklizen nemela psenice')
19 end
```

Funkci zavoláme:

```
>> data=[1 3 5 1 5; 2 1 3 2 6;1 2 11 12 5;
       1 2 11 5 7;3 5 9 10 8; 12 2 2 9 10];

>> zemedelec(data)
Sklizne plodin
    15    14    31    26    35    35

Nejvetsi sklizen
    35

Maji plodiny
     5     6

Nejvetsi sklizen nemela psenice
```

K PROCVIČENÍ

1. Provozní restaurace má v tabulce uvedeny tržby (v tisicích korun) v jednotlivých směnách během celého týdne.

   Sestavte funkci, která najde největší tržbu v jednotlivých dnech a největší tržbu během celého týdne. Dále vypíše, ve kterých dnech a na kterých směnách byla největší tržba a informaci, zda největší tržba byla nebo nebyla na některé páteční směně.

   Příklad:

	1.směna (7-12 hod)	2.směna (13-18 hod)	3.směna (19-24 hod)	4.směna (1-6 hod)
po	1	3	5	1
út	2	1	3	2
st	1	2	11	12
čt	1	2	11	5
pá	0	5	9	10
so	12	2	2	9
ne	1	5	10	8

```
Nejvetsi trzby v jednotlivych dnech: 5 3 12 11 10 12 10
Nejvetsi trzba behem celeho tydne: 12
Byla ve dnech: 3 6
A na smenach: 1 4
Nejvetsi trzba nebyla v patek.
```

2. Dispečerka má v tabulce uvedeny počty výjezdů požárního vozu v jednotlivých čtvrtletích podle situací, ke kterým se vyjíždělo.

   Sestavte funkci, která spočítá počet výjezdu za jednotlivá čtvrtletí a najde nejmenší počet výjezdů za čtvrtletí. Dále vypíše, ve kterém čtvrtletí bylo nejméně výjezdů a informaci, zda celkový počet výjezdů za rok byl nebo nebyl větší než 100.

   Příklad:

	1.čtvrtletí	2.čtvrtletí	3.čtvrtletí	4.čtvrtletí
dopravní nehoda	1	3	5	1
požár budovy	2	1	3	2
požár lesa	1	2	11	12
povodeň	1	2	11	5
technická pomoc	0	5	9	10
cvičení	12	2	2	9
planý poplach	1	3	10	8

```
Pocet vsech vyjezdu za jednotliva ctvrtleti: 18 18 51 47
Nejmensi pocet vyjezdu ve ctvrtleti: 18
Bylo to ve ctvrtletich: 1 2
Celkovy pocet vyjezdu za rok byl vetsi nez 100.
```

KAPITOLA

12

PROGRAMOVÁNÍ V MATLABU – ŘEŠENÉ PŘÍKLADY

Kapitola obsahuje množství řešených příkladů.

12.1 Základní algoritmy

1. záměna hodnot v proměnných
2. součet čísel
3. největší číslo
4. největší číslo a jeho pozice
5. celočíselné dělení
6. řadící algoritmus

12.1.1 Záměna hodnot v proměnných

Sestavte algoritmus na záměnu hodnot v proměnných.

```
a=7
b=12
pom=b
b=a
a=pom
```

12.1.2 Součet čísel

Sestavte algoritmus na výpočet součtu čísel např. $2 + 8 + 1 + 4 = 15$

```
a=[2 8 1 4]
s=0
for i=1:length(a)
  s=s+a(i)
end
```

12.1.3 Největší číslo

Sestavte algoritmus na nalezení největšího čísla např. $\max(1, 8, 3, 6, 8, 4, 6, 7, 5) = 8$

```
a=[1 8 3 6 8 4 6 7 5]
m=a(1)
for i=2:length(a)
  if a(i)>m
    m=a(i)
  end
end
```

12.1.4 Největší číslo a jeho pozice

Sestavte algoritmus na nalezení největšího čísla a jeho pozici např. $\max(1, 2, 3, 6, 8, 4, 6, 7, 5) = 8$, maximální hodnota je na pozici 5

```
a=[1 2 3 6 8 4 6 7 5]
m=a(1)
p=1
for i=2:length(a)
  if a(i)>m
    m=a(i)
    p=i
  end
end
```

12.1.5 Celočíselné dělení

Sestavte algoritmus na výpočet výsledku a zbytku po celočíselném dělení $\frac{a}{b}$ např. $\frac{14}{4} = 3 + \frac{2}{4}$ (tj. výsledek je 3 a zbytek je 2)

```
a=14
b=4
zbytek=a
vysledek=0
while zbytek>=b
```

```
  zbytek=zbytek-b
  vysledek=vysledek+1
end
```

12.1.6 Řadící algoritmus

Bubble sort Algoritmus na vzestupné seřazení čísel.

```
pocet=length(a)
for i=1:pocet-1
  for j=1:pocet-i
    if a(j)>a(j+1)
      pom=a(j)
      a(j)=a(j+1)
      a(j+1)=pom
    end
  end
end
```

12.2 Funkce - jednoduché výpočty

1. Sestavte funkci, která spočítá obsah a obvod čtverce.
2. Sestavte funkci, která spočítá obsah a obvod obdélníka. V případě, že se bude jednat o čtverec, pak tuto informace vypíše.
3. Sestavte funkci, která z poloměru kružnice a délek stran obdélníka zjistí, zda lze kružnici vepsat do obdélníka.

12.2.1 Obsah a obvod čtverce

Sestavte funkci, která spočítá obsah a obvod čtverce.

```
function [S,o]=ctverec(a)
%
S=a^2
o=4*a
```

12.2.2 Obsah a obvod obdélníka

Sestavte funkci, která spočítá obsah a obvod obdélníka. V případě, že se bude jednat o čtverec, pak tuto informace vypíše.

```
function [S,o]=obdelnik(a,b)
%
S=a*b
```

```
o=2*(a+b)
if a==b
  disp('je to ctverec')
end
```

12.2.3 Kružnice vepsaná do obdelníka

Sestavte funkci, která z poloměru kružnice a délek stran obdélníka zjistí, zda lze kružnici vepsat do obdélníka.

```
function kruznice_vepsana_obdelnik(r,a,b)
%
if a<b
  mensi=a
else mensi=b
end
if r<=mensi/2
  disp('lze')
else disp('nelze')
end
```

12.3 Funkce - výpočty s vektorem čísel

1. Sestavte funkci, která v zadaném vektoru **a** zjistí počet kladných čísel.
2. Sestavte funkci, která v zadaném vektoru **a** spočítá součet všech kladných čísel.
3. Sestavte funkci, která v zadaném vektoru **a** spočítá součin všech čísel z intervalu $(-4, 0)$.
4. Sestavte funkci, která v zadaném vektoru **a** zjistí pozice všech čísel z intervalu $(5, 10\rangle$.
5. Sestavte funkci, která v zadaném vektoru **a** zjistí počet sudých čísel větších než 6.
6. Sestavte funkci, která v zadaném vektoru **a** přepíše všechny kladná na číslo 100.
7. Sestavte funkci, která ze zadaného vektoru **a** vybere do jiného vektoru **k** všechna kladná čísla a do vektoru **z** záporná čísla.
8. Sestavte funkci, která v zadaném vektoru **a** změní znaménko u všech čísel záporných čísel.

12.3.1 Počet kladných čísel ve vektoru

Sestavte funkci, která v zadaném vektoru **a** zjistí počet kladných čísel.

```
function [pocet]=pocet_kladnych(a)
%
pocet=0
for i=1:length(a)
  if a(i)>0
    pocet=pocet+1
  end
end
```

12.3.2 Součet kladných čísel ve vektoru

Sestavte funkci, která v zadaném vektoru **a** spočítá součet všech kladných čísel

```
 function [soucet]=soucet_kladnych(a)
%
soucet=0
for i=1:length(a)
  if a(i)>0
  soucet=soucet+a(i)
  end
end
```

12.3.3 Součin čísel

Sestavte funkci, která v zadaném vektoru **a** spočítá součin všech čísel z intervalu $(-4, 0)$.

```
 function [soucin]=soucin_z_intervalu(a)
%
soucin=1
for i=1:length(a)
  if (a(i)>-4 & a(i)<0)
    soucin=soucin*a(i)
  end
end
```

12.3.4 Pozice čísel ve vektoru

Sestavte funkci, která v zadaném vektoru **a** zjistí pozice všech čísel z intervalu $(5, 10\rangle$.

```
function [pozice]=pozice_z_intervalu(a)
%
pozice=[ ]
for i=1:length(a)
  if (a(i)>5 & a(i)<=10)
    pozice=[pozice,i]
  end
```

```
end
```

12.3.5 Počet sudých čísel

Sestavte funkci, která v zadaném vektoru **a** zjistí počet sudých čísel větších než 6.

```
function [pocet]=pocet_sudych(a)
%
pocet=0
for i=1:length(a)
    if (a(i)>6 & rem(a(i),2)==0)
      pocet=pocet+1
    end
end
```

12.3.6 Přepsání čísel

Sestavte funkci, která v zadaném vektoru **a** přepíše všechny kladná na číslo 100.

```
 function [a]=prepise_kladna(a)
%
for i=1:length(a)
  if a(i)>0
    a(i)=100
  end
end
```

12.3.7 Výběr kladných a záporných čísel

Sestavte funkci, která ze zadaného vektoru **a** vybere do jiného vektoru **k** všechna kladná čísla a do vektoru **z** záporná čísla.

```
 function [k,z]=vybere(a)
%
k=[ ]
z=[ ]
for i=1:length(a)
  if a(i)>0
      k=[k a(i)]
    elseif a(i)<0
      z=[z a(i)]
  end
end
```

12.3.8 Změna znamínek

Sestavte funkci, která v zadaném vektoru **a** změní znaménko u všech čísel záporných čísel.

```
 function [a]=zmena_znamenka(a)
%
for i=1:length(a)
  if a(i)<0
    a(i)=-a(i)
  end
end
```

12.4 Funkce - výpočty s matici čísel

1. Sestavte funkci, která v dané matici spočítá součty ve sloupcích.
2. Sestavte funkci, která v dané matici na zadaném řádku najde největší prvek.
3. Sestavte funkci, která v dané matici zjistí počet kladných čísel.
4. Sestavte funkci, která v dané matici zjistí pozice kladných čísel mezi prvky na diagonále čtvercové matice.
5. Sestavte funkci, která v dané matici spočítá průměrnou hodnoty prvků na diagonále čtvercové matice.

12.4.1 Součty ve sloupcích matice

Sestavte funkci, která v dané matici spočítá součty ve sloupcích.

```
function [soucty]=soucet_sloupce(A)
%
[pocetR,pocetS]=size(A)
soucty=zeros(pocetS,1)
for j=1:pocetS
  for i=1:pocetR
    soucty(j)=soucty(j)+A(i,j)
  end
end
```

12.4.2 Největší prvek v řádku matice

Sestavte funkci, která v dané matici na zadaném řádku najde největší prvek.

```
function [m]=nejvetsi_v_radku(A,radek)
%
[pocetR,pocetS]=size(A)
m=A(radek,1)
```

```
for j=2:pocetS
  if A(radek,j)>m
    m=A(radek,j)
   end
end
```

12.4.3 Počet kladných čísel v matici

Sestavte funkci, která v dané matici zjistí počet kladných čísel.

```
function [pocet]=pocet_kladnych(A)
%
[pocetR,pocetS]=size(A)
pocet=0;
for j=1:pocetS
  for j=1:pocetR
    if A(i,j)>0
      pocet=pocet+1
    end
  end
end
```

12.4.4 Pozice kladných čísel na diagonále matice

Sestavte funkci, která v dané matici zjistí pozice kladných čísel mezi prvky na diagonále čtvercové matice.

```
function [pozice]=pozice_kladnych_diagonala(A)
%
[pocetR,pocetS]=size(A)
pozice=[ ]
for i=1:pocetR
  if A(i,i)>0
    pozice=[pozice,i]
  end
end
```

12.4.5 Průměr na diagonále matice

Sestavte funkci, která v dané matici spočítá průměrnou hodnoty prvků na diagonále čtvercové matice.

```
 function [m]=prumer_diagonala(A)
%
[pocetR,pocetS]=size(A)
soucet=0
```

```
for j=1:pocetS
  soucet=soucet+A(i,i)
end
m=soucet./pocetS
```

KAPITOLA

13

LINEÁRNÍ ALGEBRA

NAUČÍME SE

Ukážeme si jak v Matlabu spočítat hodnost matice, determinant a jak najít inverzní matici. Dále si popíšeme několik způsobů jak vyřešit soustavu lineárních rovnic.

OPAKOVÁNÍ MATEMATIKY

Definice : **Soustavou lineárních rovnic** rozumíme m rovnic o n neznámých, tedy

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \quad \cdots \quad + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \quad \cdots \quad + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \quad \cdots \quad + a_{mn}x_n &= b_m, \end{aligned}$$

kde a_{ij} se nazývají **koeficienty** soustavy, x_j jsou **neznámé** a b_i jsou **pravé strany** rovnic soustavy pro $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$.

Definice : Matici A, jejíž prvky tvoří koeficienty soustavy a_{ij}, nazýváme **maticí soustavy**, tedy

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Definice : Matici R, která vznikne z matice A připojením sloupce pravých stran, nazýváme **rozšířenou maticí soustavy**, tedy

$$R = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{pmatrix}$$

Poznámka: Soustavu m lineárních rovnic o n neznámých můžeme zapsat také maticově:

$$A \cdot \vec{x} = \vec{b},$$

kde A je matice soustavy, $\vec{x}$ je sloupcový vektor neznámých a $\vec{b}$ je sloupcový vektor pravých stran soustavy, tedy

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Věta : ***Frobeniova věta:***
Soustava m lineárních rovnic o n neznámých má ***alespoň jedno řešení****, právě když se hodnost matice soustavy (označení: h_s) rovná hodnosti matice roz?ířené (označení: h_r), tedy*

$$h_s = h_r = h. \tag{13.1}$$

Pokud $h_s \neq h_r$, pak řešení soustavy ***neexistuje****.*

Má-li soustava řešení, pak pro $h = n$ má soustava ***právě jedno řešení****, jinak, tj. pro $h < n$ má soustava* ***nekonečně mnoho řešení*** *závislých na $n - h$ parametrech.*

VÝKLAD UČIVA

13.1 Lineární algebra

13.1.1 Funkce lineární algebry pro matice

`det(A)`	determinant matice A
`rank(A)`	hodnost matice A
`inv(A)`	inverzní matice A
`A'`	transponovaná matice

PŘÍKLAD 57.

Př. Spočítáme determinant, hodnost a transponovanou k matici A:

```
>> A=[2 3 4; 3 5 0; 2 3 9]
A =
     2     3     4
     3     5     0
     2     3     9
>> det(A)
ans =
     5
>> hodnost=rank(A)
hodnost =
```

```
     3
>> transponovana=A'
transponovana =
     2     3     2
     3     5     3
     4     0     9
```

Př. K matici z předchozího příkladu spočítáme matici inverzní a zkotrolujeme, zda platí $A^{-1} \cdot A = I$

```
>> inverzni=inv(A)
inverzni =
    9.0000   -3.0000   -4.0000
   -5.4000    2.0000    2.4000
   -0.2000         0    0.2000
>> inverzni*A
ans =
    1.0000   -0.0000    0.0000
         0    1.0000         0
         0         0    1.0000
```

13.1.2 Funkce lineární algebry pro vektory

Nyní uvedeme některé funkce pro vektory:

`cross(v,u)`	vektorový součin vektorů $\vec{v}$ a $\vec{u}$
`dot(v,u)`	skalární součin vektorů $\vec{v}$ a $\vec{u}$

PŘÍKLAD 58.

Př. Spočítáme skalární a vektorový součin vektorů $\vec{v}$, $\vec{u}$:

```
>> v=[2 3 4],u=[0 2 1]
v =
     2     3     4
u =
     0     2     1
>> length(v), length(u)
ans =
     3
ans =
     3
>> skalarni=dot(v,u)
skalarni =
    10
>> vektorovy=cross(v,u)
vektorovy =
    -5     2     4
```

13.1.3 Řešení soustav lineárních rovnic

Soustavu lineárních rovnic lze vyřešit několika způsoby. Jedním z nich je použít dělení zprava, tedy pro soustavu $Ax = b$ najdeme řešení příkazem `x=A\b`. A to jak v případě, že má soustava jedno řešní, tak v případě, že má nekonečně mnoho řešení závislých na parametru či parametrech. Pak tímto dostaneme jedno z těchto řešení.
Chceme-li najít všechan řešení soustavy, která má nekonečně mnoho řešení použijeme příkaz pro Gauss-Jordanova eliminaci.

`rref` Gauss-Jordanova eliminace

PŘÍKLAD 59.

Př. Najděte řešení soustavy rovnic

$$\begin{aligned} 3x_1 + 4x_2 &= 7 \\ x_1 - 2x_2 &= -6 \end{aligned}$$

Nejprve zadáme matici a vektor:

```
>> A=[3  4;1  -2],b=[7;  -6]
A =
      3      4
      1     -2
b =
      7
     -6
```

Spočítáme hodnost matice a hodnost matice rozšířené:

```
>> rank(A),rank([A  b])
ans =
      2
ans =
      2
```

Jelikož se hodnosti rovnají, má soustava řešení. Jelikož jsou zároveň rovny počtu neznámých, je toto řešení jediné.

```
>> x=A\b
x =
   -1.0000
    2.5000
```

Soustava má řešení

$$x_1 = -1 \quad x_2 = 2.5$$

PŘÍKLAD 60.

Najděte řešení soustavy rovnic

$$\begin{aligned} 3x_1 + 7x_2 - 2x_3 &= 2 \\ 4x_1 + 3x_2 + 2x_3 &= 5 \\ 2x_1 + 11x_2 - 6x_3 &= -1 \\ -x_1 + 7x_2 + 3x_3 &= -1 \end{aligned}$$

Nejprve zadáme matici a vektor:

```
>> A=[ 3  7 -2; 4 3 2; 2 11 -6; -1 7 3]
A =
       3             7            -2
       4             3             2
       2            11            -6
      -1             7             3
>> b=[2;5;-1;-1]
b =
       2
       5
      -1
      -1
```

Spočítáme hodnost matice a hodnost matice rozšířené:

```
>> rank(A)
ans =
       3
>> rank([A,b])
ans =
       3
>> size(A)
ans =
       4             3
```

Jelikož se hodnosti rovnají, má soustava řešení. Jelikož jsou zároveň rovny počtu neznámých (počet sloupců matice A), je toto řešení jediné.

```
>> x=A\b
x =
    1.1714
   -0.1200
    0.3371
```

Zkontrolujeme chybu vypočtu:

```
>> A*x-b
ans =
```

```
  1.0e-014 *

   -0.0444
   -0.1776
    0.0888
    0.2220
```

Řešení lze nalézt i převedením rozšířené matice na trojúhelníkový tvar:

```
>> rref([A,b])
ans =
    1.0000         0         0    1.1714
         0    1.0000         0   -0.1200
         0         0    1.0000    0.3371
         0         0         0         0
```

Soustava má řešení

$$x_1 = 1.1714 \quad x_2 = -0.1200 \quad x_3 = 0.3371$$

PŘÍKLAD 61.

Najděte řešení soustavy rovnic

$$\begin{aligned} 3x_1 + 7x_2 - 2x_3 &= 2 \\ 4x_1 + 3x_2 + 2x_3 &= 5 \\ 2x_1 + 11x_2 - 6x_3 &= 6 \end{aligned}$$

Nejprve zadáme matici a vektor:

```
>> A=[ 3 7 -2; 4 3 2; 2 11 -6]
A =
     3     7    -2
     4     3     2
     2    11    -6
>> b=[2; 5; 6]
b =
     2
     5
     6
```

Spočítáme hodnost matice a hodnost matice rozšířené:

```
>> rank(A)
ans =
     2
>> rank([A,b])
ans =
     3
```

Jelikož se hodnosti nerovnají, nemá soustava řešení. Podíváme se ještě na trojúhleníkový tvar rozšířené matice:

```
>> ans=rref([A b])
ans =
    1.0000         0    1.0526         0
         0    1.0000   -0.7368         0
         0         0         0    1.0000
```

PŘÍKLAD 62.

Př. Najděte řešení soustavy rovnic

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1 \\ -x_1 + 3x_2 + 2x_3 &= 0 \\ 3x_1 - 4x_2 - x_3 &= 1 \end{aligned}$$

Nejprve zadáme matici a vektor:

```
>> A=[-1 2 3;-1 3 2;3 -4 -1], b=[1;0;1]

A =

     1     2     3
    -1     3     2
     3    -4    -1


b =

     1
     0
     1
```

Spočítáme hodnost matice a hodnost matice rozšířené:

```
>> rank(A),rank([A b])
ans =
     2
ans =
     2
>> size(A)

ans =

     3     3
```

Jelikož se hodnosti rovnají, má soustava řešení. Máme však 3 neznámé a tak má soustava nekonečně mnoho řešení závislých na jednom parametru.

```
>> rref([A b])

ans =

    1.0000         0    1.0000    0.6000
         0    1.0000    1.0000    0.2000
         0         0         0         0
```

Rozšířenou matici po eliminaci přepíšeme jako soustavu lineárních rovnic: Zavedeme si parametr $x_3 = t$ a dosadíme do rovnic

$$\begin{aligned} x_1 + x_3 &= 0.6 \quad \Rightarrow \quad x_1 = 0.6 - t \\ x_2 + x_3 &= 0.2 \quad \Rightarrow \quad x_2 = 0.2 - t \end{aligned} \tag{13.2}$$

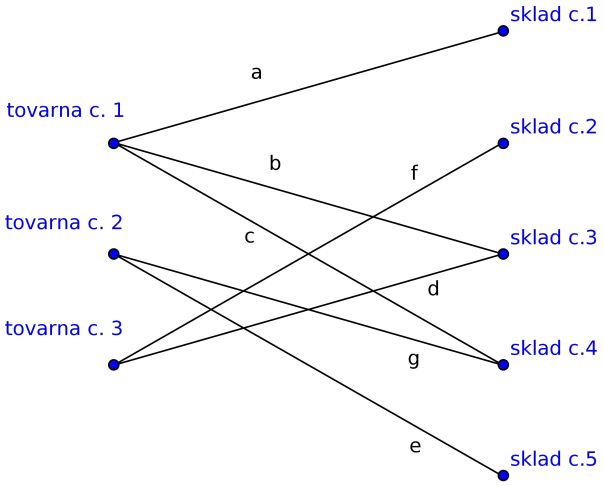
Řešení je

$$x_1 = 0.6 - t \quad x_2 = 0.2 - t \quad x_3 = t \qquad \text{kde } t \in \mathbb{R}$$

13.2 Úloha „Dopravní úloha“

ZADÁNÍ ŘEŠENÉ ÚLOHY

Ze tří továren potřebujeme rozvést do pěti skladů zboží cestami označenými a, b, c, d, e, f, g (viz. obrázek 13.1).



Obrázek 13.1: Dopravní úloha – zadání

Množství zboží, které vyrobí jednotlivé továrny a kapacity skladů jsou uvedeny v následujících tabulkách.

továrna č.	vyrobí
1	100
2	150
3	100

sklad č.	má kapacitu
1	20
2	80
3	60
4	100
5	90

POSTUP ŘEŠENÍ

Součet zboží odvezeného z první továrny musí být roven množství zboží, které se tam vyrobí. Tedy $a + b + c = 100$. Obdobně sestavíme rovnici i u druhé a třetí továrny. Další rovnice dostaneme s úvahy, že součet množství zboží přivezeného do skladu se musí rovnat jeho kapacitě.
Pak dostáváme soustavu lineárních rovnic:

$$\begin{aligned} a+b+c &= 100 \\ e+g &= 150 \\ d+f &= 100 \\ a &= 20 \\ f &= 80 \\ b+d &= 60 \\ c+g &= 100 \\ e &= 90 \end{aligned}$$

Zadáme matici a pravou stranu.

```
>> A=[1     1     1     0     0     0     0     0
      0     0     0     0     1     0     1     0
      0     0     0     1     0     1     0     0
      1     0     0     0     0     0     0     0
      0     0     0     0     0     1     0     0
      0     1     0     1     0     0     0     0
      0     0     1     0     0     0     0     1
      0     0     0     0     1     0     0     0];

>> b=[100; 150; 100; 20; 80; 60; 100; 90];
```

Vyřešíme soustavu lineárních rovnic.

```
>> A\b

ans =
```

```
20
40
40
20
90
80
60
60
```

Řešení naší úlohy je:

$$a = 20,\ b = 40,\ c = 40,\ d = 20,\ e = 90,\ f = 80,\ g = 60$$

K PROCVIČENÍ

1. Najděte všechna řešení soustavy lineárních rovnic:

$$\begin{aligned} x_1 + 2x_2 + x_4 + 3x_5 - 2x_6 &= -4 \\ 3x_2 - 4x_3 + 2x_4 + x_5 + 5x_6 &= 21 \\ x_1 + x_2 - 3x_5 - 2x_6 &= -8 \\ -x_1 + 2x_2 + 3x_3 + 5x_5 + 6x_6 &= 0 \\ 3x_1 + 4x_2 + x_4 - 3x_5 - 6x_6 &= -20 \end{aligned}$$

KAPITOLA

14 FUNKCE A GRAFY

NAUČÍME SE

Ukážeme si jak se zadávají matematické funkce a konstanty. Naučíme vykreslit graf funkce a graf dat.

VÝKLAD UČIVA

14.1 Funkce

14.1.1 Elementární matematické funkce

Nyní uvedeme seznam elementárních funkcí.

`sin(x)`	sinus: $\sin(x)$
`cos(x)`	kosinus: $\cos(x)$
`tan(x)`	tangens: $\mathrm{tg}(x)$
`cot(x)`	kotangens: $\mathrm{cotg}(x)$
`asin(x)`	arkussinus: $\arcsin(x)$
`acos(x)`	arkuskosinus: $\arccos(x)$
`atan(x)`	arkustangens: $\mathrm{arctg}(x)$
`acot(x)`	arkuskotangens: $\mathrm{arccotg}(x)$
`log(x)`	přirozený logaritmus: $\ln(x)$
`log10(x)`	dekadický logaritmus: $\log(x)$
`log2(x)`	logaritmus při základu 2: $\log_2(x)$
`exp(x)`	exponeciální funkce: e^x
`pow2(x)`	mocninná funkce při základu 2: 2^x
`sqrt(x)`	druhá odmocnina: $\sqrt{x}$
`abs(x)`	abolutní hodnota: $\lvert x \rvert$
`sign(x)`	funkce signum

Funkce můžeme použít i pro matice, funkce se pak vyčíslí pro jednotlivé prvky matice.

PŘÍKLAD 63.

Př. Spočítáme hodnotu $\log_2(8)$:

```
>> log2(8)
ans =
      3
```

Př. Spočítáme hodnotu funkce $y = \frac{1}{x} + e^{-x}$ v bodě $x = 4$:

```
>> x=4
x =
     4
>> 1/x+exp(-x)
ans =
    0.2683
```

Př. Spočítáme hodnotu funkce $y = \dfrac{\mathrm{tg}(x) + 2\pi}{\sin(x - \pi/2)}$ v bodě $x = 0$:

```
>> x=0;y=(tan(x)+2*pi)/sin(x-pi/2)
y =
   -6.2832
```

Př. Spočítáme hodnotu funkce $y = \sqrt[3]{x-2}$ v bodě $x = 241.5$:

```
x=241.5; y=(x-2)^(1/3)
y =
    6.2101
```

Př. Spočítáme absolutní hodnotu čísel $-5,\ 9,\ 0,\ 3,\ 4,\ -8$:

```
>> posl=[-5 9 0 3 4 -8]
posl =
    -5     9     0     3     4    -8
>> vysledek=abs(posl)
vysledek =
     5     9     0     3     4     8
```

Př. Spočítáme hodnotu $\log_3(81)$. Matlab nemá k dispozici funkci pro logaritmus se základem 3, avšak z matematiky známe vzorec $\log_a(x) = \frac{\ln(x)}{\ln(a)}$, který použijeme:

```
>> log(81)/log(3)
ans =
     4
```

a provedeme kontrolu:

```
>>3^4
ans =
    81
```

14.1.2 Definice vlastní funkce

Vlastní matematickou funkci si nadefinujeme příkazem `inline`, jehož parametrem je funkční předpis v apostrofech:

```
jméno=inline('předpis')
```

PŘÍKLAD 64.

Př. Funkci $f = \sin(x^2) - \sqrt{3-x}$ nadefinujeme:

```
>> f=inline('sin(x.^2)-sqrt(3-x)')

f =

     Inline function:
     f(x) = sin(x.^2)-sqrt(3-x)

>> f(1)

ans =

   -0.5727
```

Př. V definice funkce v předchozím příkladě jsme mocninu zapsali pomoci operace `.^`. Je-li potřeba vytabelovat funkci pro více hodnot, je nezbytné použít operace „prvek po prvku".

```
>> f=inline('1./x')
f =
     Inline function:
     f(x) = 1./x

>> x=1:5
x =
     1     2     3     4     5

>> f(x)
ans =
    1.0000    0.5000    0.3333    0.2500    0.2000

>> [x',f(x)']
ans =
    1.0000    1.0000
    2.0000    0.5000
    3.0000    0.3333
    4.0000    0.2500
    5.0000    0.2000
```

14.1.3 Konstanty a speciální proměnné

Matlab má k dispozici několik konstant a speciálních proměnných.

`pi`	Ludolfovo číslo $\pi = 3.141592\ldots$
`inf`	nekonečno ∞
`NaN`	neurčitý výraz
`eps`	relativní přesnost $2^{-52} \approx 2.22e-16$
`realmax`	největší kladné číslo $2^{1024} \approx 1.7977e+308$
`realmin`	nejmenší kladné číslo $2^{-1022} \approx 2.2251e-308$

PŘÍKLAD 65.

Př. Budeme-li potřebovat Eulerovo číslo (základ přirozeného logaritmu) získáme ho jako hodnotu e[1] tedy:

```
>> e=exp(1)
e =
    2.7183
```

Př. Čísla větší než `realmax` (resp. menší než `-realmax`) jsou považována za ∞ (resp. $-\infty$):

```
>> realmax
ans =
  1.7977e+308
>> 1e400
ans =
   Inf
```

Př. Čísla jejichž absolutní hodnota je menší než `realmin` jsou považována za 0:

```
>> realmin
ans =
  2.2251e-308
>> 1e-400
ans =
     0
```

Př. Neurčitý výraz je například $\infty - \infty$ nebo $0 \cdot \infty$:

```
>> inf-inf
ans =
   NaN
>> 0*inf
ans =
   NaN
```

14.2 Grafy

14.2.1 Vykreslení grafu

`plot(x,y)`	graf bodů o souřadnicích (x, y)
`fplot('f',[a,b])`	graf funkce f na intervalu $\langle a, b \rangle$

Vstupem příkazu `plot` jsou souřadnice bodů, které se pak spojí čarou. Chceme-li vykreslit graf funkce, lze použít i příkaz `fplot`, první parametrem je funkční předpis v apostrofech a druhým je interval, na kterém chceme graf vykreslit, třetím nepovinným parametrem je specifikace.

PŘÍKLAD 66.

Př. Vykreslíme graf funkce $y = x^2$ na intervalu $\langle -4, 4 \rangle$ pomocí příkazu `plot`. Vstupem příkazu `plot` jsou souřadnice bodů, které se pak spojí čarou, viz obrázek 14.1. Nejprve si do proměnné `x` uložíme souřadnice x, které vygenerujeme jako 100 bodů mezi -4 a 4:

```
>> x=linspace(-4,4);
```

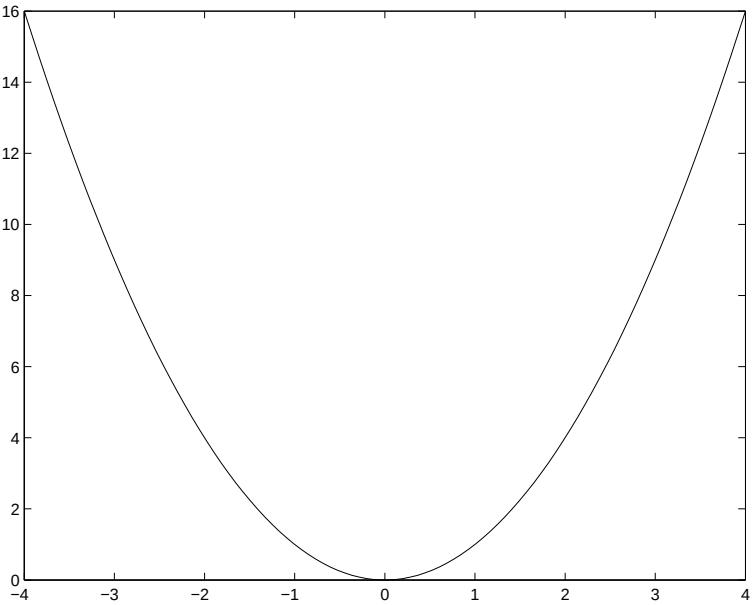
Pak vytvoříme souřadnice y jako funkční hodnoty funkce $y = x^2$ v bodech `x`:

```
>> y=x^2;
```

A vykreslíme graf:

```
>> plot(x,y)
```

Po vypsání příkazu se zobrazí nové okno *Figure No. 1*, které se stane aktivním.

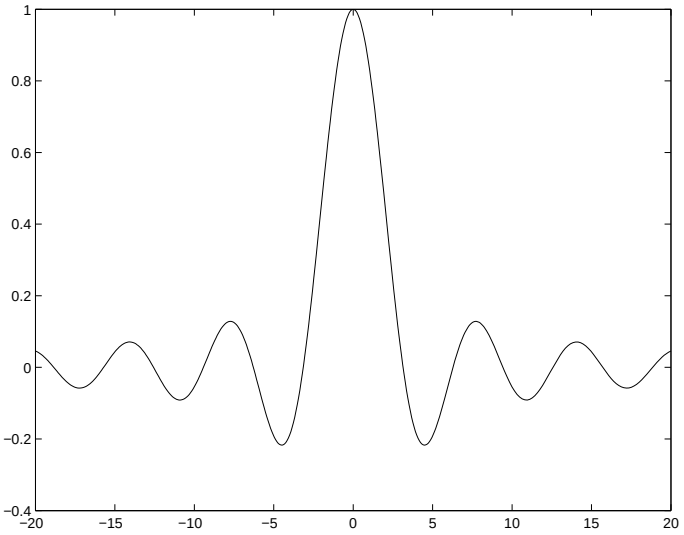


Obrázek 14.1: Graf funkce $y = x^2$

Pokud je toto okno otevřené, graf se překreslí, ale okno se nestane aktivním. V tomto případě okno učiníme aktivní kliknutím na tlačítko *Figure No. 1* na hlavním panelu ve *Windows*.

Př. Chceme-li vykreslit graf funkce, lze použít i příkaz `fplot`, první parametrem je funkční předpis v apostrofech a druhým je interval, na kterém chceme graf vykreslit, viz obrázek 14.2. Zobrazíme graf funkce $y = \frac{\sin(x)}{x}$ na intervalu $\langle -20, 20 \rangle$:

```
>> fplot('sin(x)/x',[-20,20])
```



Obrázek 14.2: Graf funkce $y = \frac{\sin(x)}{x}$

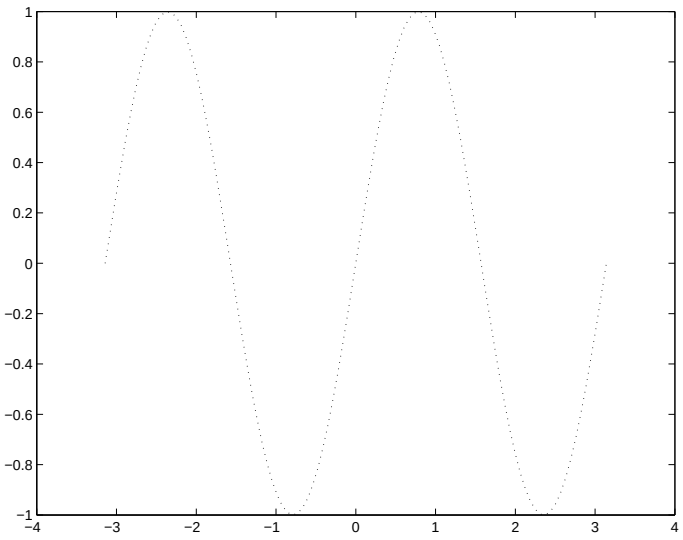
Chceme-li graf vykreslit jinak než modrou plnou čárou, použijeme tzv. specifikaci grafu. Pro specifikaci můžeme použít libovolnou kombinaci voleb z prvního, druhého a třetího sloupce tabulky 14.1 (v uvedeném pořadí), přičemž některou lze vynechat. Specifikaci uvádíme jako řetězec, tj. do apostrofů a je to třetí nepovinný parametr příkazu `plot` a `fplot`. Například červenou čárkovanou čáru vytvoříme specifikací `'r-'`, zelené kroužky `'go'`, žlutou čáru s trojúhelníčky `'yv-'`.

Barvy		Symboly		Typy čar	
`b`	modrá	`.`	tečky	`-`	plná
`g`	zelená	`o`	kroužky	`:`	tečkované
`r`	červená	`x`	křížky ×	`-.`	čerchovaná
`c`	světlé modrá	`+`	křížky +	`--`	čárkovaná
`m`	fialová	`*`	hvězdičky		
`y`	žlutá	`s`	čtverce		
`k`	černá	`d`	kosočtverce		
		`v`	trojúhelníky (dolu)		
		`^`	trojúhelníky (nahoru)		
		`<`	trojúhelníky (vlevo)		
		`>`	trojúhelníky (vpravo)		
		`p`	pěticípé hvězdy		
		`h`	šesticípé hvězdy		

Tabulka 14.1: Specifikace grafu

Př. Ukážeme si graf funkce $y = \sin(2x)$ na intervalu $\langle -\pi, \pi \rangle$ černou tečkovanou čarou, viz obrázek 14.3.

```
>> x=linspace(-pi,pi);
>> y=sin(2*x);
>> plot(x,y,'k:')
```



Obrázek 14.3: Graf funkce $y = \sin(2x)$

Př. Specifikace pomocí symbolů se často používá chceme-li zobrazit diskrétní data. Máme zadané hodnoty v tabulce, a zobrazíme je jako hvězdičky, viz obrázek 14.4.

t	1	3	5	7	10	14	15
s	19	17	18	20	16	22	19

```
>> t=[1 3 5 7 10 14 15];
>> s=[19 17 18 20 16 22 19];
>> plot(t,s,'h')
```

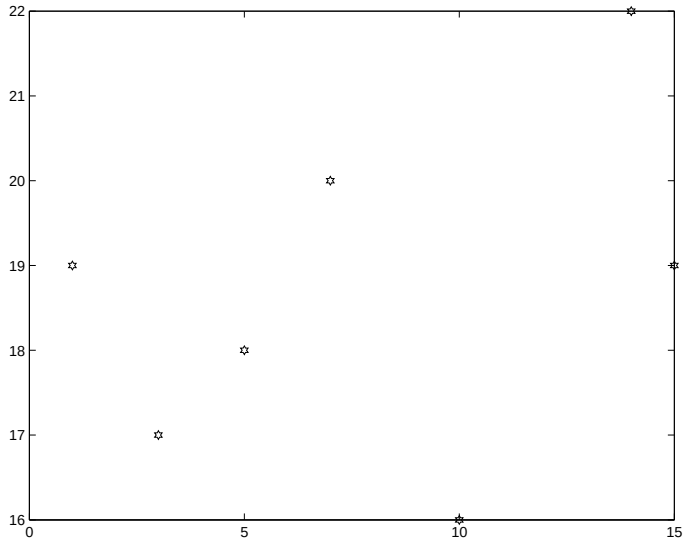
Př. Jsou-li x-ové souřadnice čísla $1, 2, 3, \ldots$ můžeme je v příkazu `plot` vynechat ,viz obrázek 14.5.

den	1	2	3	4	5	6	7	8	9	10	11
teplota	17	21	19	23	16	21	19	19	15	23	20

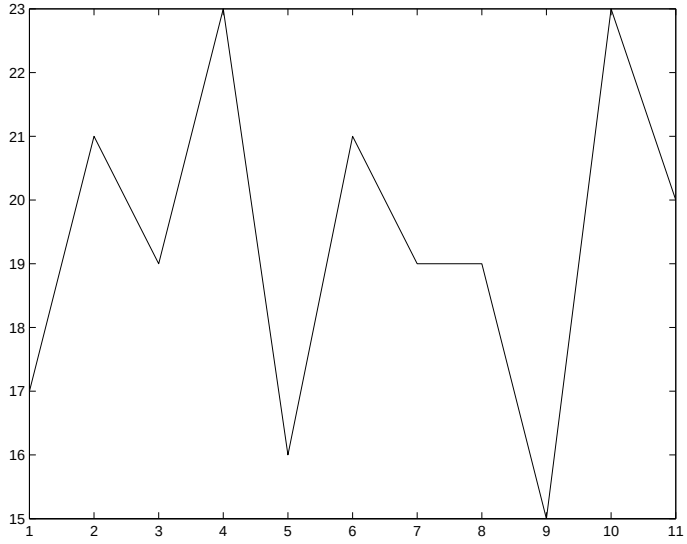
```
>> teplota=[17 21 19 23 16 21 19 19 15 23 20];
>> plot(teplota)
```

Př. Při volbě intervalu, na kterém vykreslujeme graf musíme být obezřetní a brát v úvahu definiční obor funkce. Zobrazíme si graf funkce $y = \ln(x)$, víme, že definiční obor je $(0, \infty)$, zobrazíme si tedy graf na intervalu, který leží v definičním oboru, například $\langle 0.0001, 3 \rangle$.

```
>> fplot('log(x)',[0.0001,3])
```



Obrázek 14.4: Graf diskrétních dat



Obrázek 14.5: Graf teplot

14.2.2 Více grafů najednou

`hold`	vykreslení grafů do jednoho obrázku, nastavením `on` tuto vlastnost zapneme, `off` vypneme
`subplot(m,n,k)`	víceobrázků do jednoho okna

PŘÍKLAD 67.

Př. Máme-li naměřené hodnoty v tabulce:

	-4	2	3	7	10
	9	6	0	32	89

A víme že předpokládané chování této fyzikální veličiny je dáno funkcí:

$$y = x^2 - xe^{\frac{x}{10}} + 1\,.$$

Chceme do jednoho grafu zobrazit naměřené i teoretické hodnoty. Nejprve zadáme data z tabulky do proměnných `vel1, vel2` a příkazem `hold on` otevřeme okno (zatím prázdné), do kterého se budou zobrazovat všechny grafy, které budeme vykreslovat:

```
>> vel1=[1 3 7 9 10];
>> vel2=[9 6 0 32 89];
>> hold on
```

Vykreslíme data z tabulky jako červené hvězdičky:

```
>> plot(vel1,vel2,'r*')
```

a graf teoretických hodnot:

```
>> fplot('x^2-x*exp(x/10)+1',[-5,10])
```

14.2.3 Nastavení grafu

Do grafu můžeme přidat popisy os, nadpis, text na libovolné místo, také lze měnit rozsah os.

`axis([x1,x2,y1,y2])`	rozsah os pro první proměnnou od x1 do x2, pro druhou od y1 do y2
`axis equal`	osy v poměru 1:1
`grid`	zobrazení mřížky do grafu
`title('text')`	titulek obrázku
`xlabel('text')`	popis x-ové osy
`ylabel('text')`	popis y-ové osy
`text(x1,y1,'text')`	umístění textu na pozici $x1, y1$
`legend('text1','text2','text3')`	legenda ke grafům

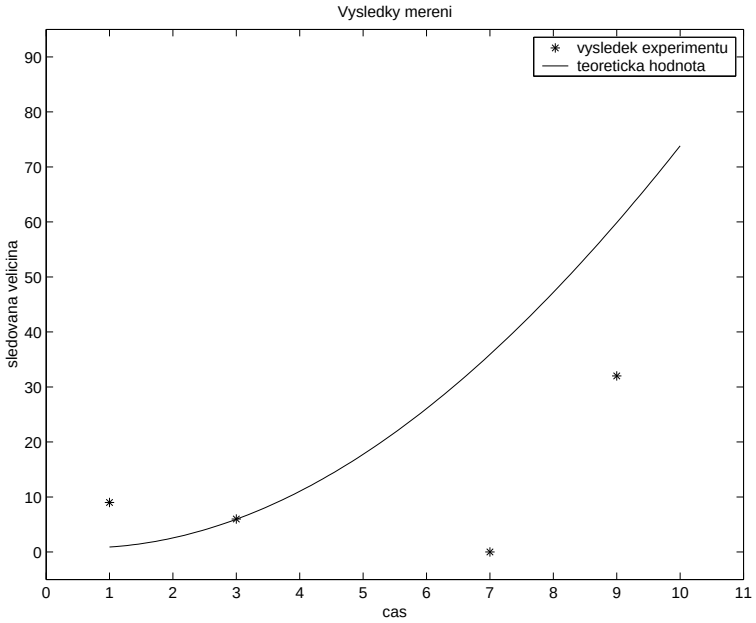
PŘÍKLAD 68.

Př. Nejprve si vykreslíme graf stejně jako v předchozím příkladě:

```
>> t=[ 1 3 7 9 10];
>> s=[9 6 0 32 89];
>> x=linspace(1,10);
>> y=x.^2-x.*exp(x/10)+1;
>> plot(t,s,'*',x,y,'-')
```

A grafu změníme rozsah os, přidáme legendu, nadpis a popis obou os.

```
>> axis([0,11,-5,95])
>> legend('vysledek experimentu','teoreticka hodnota')
>> title('Vysledky mereni')
>> xlabel('cas')
>> ylabel('sledovana velicina')
```



Obrázek 14.6: Graf funkce

14.3 Úloha „Orientační běžec“

Zadání řešené úlohy

Orientační běžec má souřadnice stanovišť, ke kterým se musí dostat. Nakreslete modře dráhu, po které poběží a nejdelší úsek červeně. Spočítejte celkovou délku trasy a také délku nejdelšího a nejkratšího úseku.

x_i	1	2	4	5	6	8	9	11	12	14
y_i	3	4	1	6	9	7	2	4	7	9

Pozn. Upravte program, je-li nejdelších úseků více.

Postup řešení

Nejprve si zadáme data.

```
>> x=[1  2  4  5  5  8  9  11  12  14]

x =

     1     2     4     5     6     8     9    11    12    14

>> y=[3  4  1  5  9  7  2  4  7  9]

y =
     3     4     1     5     9     7     2     4     7     9
```

Vykreslíme zadaná data jako plnou čáru a kolečka.

```
>> hold on
>> plot(x,y,'-o')
```

Určíme počet bodů

```
>> n=length(x)

n =

    10
```

A spočítáme délky úseček.

```
>> for i=1:n-1, d(i)=sqrt((x(i)-x(i+1))^2+(y(i)-y(i+1))^2); end
>> d

d =

    1.4142    3.6056    4.1231    4.1231    2.8284    5.0990
2.8284    3.1623    2.8284
```

Najdeme nejdelší délku.

```
>> celkem=sum(d)

celkem =

    30.0125
```

Najdeme nejdelší délku.

```
>> nej=max(d)

nej =

    5.0990
```

A zjistíme které z úseček je nejdelší.

```
>> d==nej

ans =

     0     0     0     0     0     1     0     0     0

>> p=find(d==nej)

p =

     6
```

Do proměnné `nejx`, `nejy` uložíme souřadnice krajních bodů nejdelší úsečky a vykreslíme ji červeně.

```
>> nejx=x(p:p+1)

nejx =

     8     9

>> nejy=y(p:p+1)

nejy =

     7     2

>> plot(nejx,nejy,'r')
```

Předchozí program upravíme pro případ, že bude v lomené čáře více nejdelších úseček.

```
>> x=[1 2 4 5 5 8 9 11 12 14]

x =

     1     2     4     5     6     8     9    11    12    14

>> y=[3 4 1 6 9 7 2 4 7 9]

y =
     3     4     1     6     9     7     2     4     7     9

>> hold on
>> plot(x,y)
>> plot(x,y,' o ')

>> n=length(x)

n =

    10
>> for i=1:n-1, d(i)=sqrt((x(i)-x(i+1))^2+(y(i)-y(i+1))^2); end
>> d

d =

         1.4142    3.6056    5.0990    3.1623    2.8284    5.0990
2.8284    3.1623    2.8284

>> celkem=sum(d)
```

```
>> celkem=

    30.0276

>> nej=max(d)

nej =

    5.0990

>> d==nej

ans =

     0     0     1     0     0     1     0     0     0

>> p=find(d==nej)

p =

     3     6
```

Vidíme, že nejdelší úsečky jsou dvě. Musíme tedy v cyklu vybrat do `nejx` a `nejy` krajní bodů jednotlivých nejdelších useček.

```
>> for i=1:length(p),
nejx=x(p(i):p(i)+1),
nejy=y(p(i):p(i)+1),
plot(nejx,nejy,'g'),
end
```

Vytvoříme funkci. Vstupem budou vektory `x` a `y`. Výstupem bude celková délka lomené čáry a délka nejdelší úsečky.

```
1  function [celkem, nej]=bezec(x,y)
2  % funkce na  zpracoani lomene cary
3  if length(x)~=length(y)
4      error('Neni stejny pocet x jako y')
5  end
6
7
8  n=length(x);
9
10 for i=1:n-1
11     d(i)=sqrt((x(i)-x(i+1))^2+(y(i)-y(i+1))^2)
12 end
13
```

```
14 celkem=sum(d)
15 nej=max(d)
16 p=find(d==nej)
17
18 hold on
19
20 plot(x,y,'-o')
21
22 for i=1:length(p)
23     nejx=x(p(i):p(i)+1)
24     nejy=y(p(i):p(i)+1)
25     plot(nejx,nejy,'g')
26 end
```

K PROCVIČENÍ

1. Nakreslete grafy funkcí, s ohledem na jejich definiční obor
   (a) $f(x) = \dfrac{1}{x}$
   (b) $f(x) = \mathrm{tg}(x)$
   (c) $f(x) = \log_4(x)$
   (d) $f(x) = \arcsin(x)$

2. Je dána lomená čára (obdobně jako v úloze „Orientační běžec“). Spočítejte průměrnou délku úseku. Vykreslete čáru modře a úseky, které jsou delší než průměrná délka červeně.

3. Vypočítejte hodnoty funkce f v bodech $x_1 = 1$, $x_2 = 1.7$, $x_3 = 3.5$.
   Vykreslete graf funkce f na intervalu $\langle 1, 4 \rangle$.

$$f : y = \sin^2\left(2x + \frac{1}{3}\right) + \sqrt{x^3 + 4}$$

KAPITOLA

15

SYMBOLICKÉ POČÍTÁNÍ

NAUČÍME SE

Ukážeme si práci s *Symbolic Math Toolbox*. Naučíme e řešit rovnici, počítat limitu, derivovat funkci a řešit soustavu lineárních rovnic s parametrem.

15.1 Symbolické počítání

Pro symbolické počítání má Matlab *Symbolic Math Toolbox*, které není standartní součástí Matlabu, lze jej dokoupit.
Poznámka: Postupy a příkazy uvedené v této kapitole nejsou v Octave k dispozici.

Označení symbolické proměnné

Na začátky každého výpočtu je potřeba příslušné proměnné, které budeme ve výpočtech požíva označit jako symbolické proměnné příkazem `syms`.

15.2 Řešení rovnic

Rovnici nebo soustavu rovnic řešíme příkazem `solve`.

PŘÍKLAD 69.

Př. Vyřešíme rovnici $x^3 - 4 * x^2 - x = 0$. Nejprve vytvoříme symbolickou proměnnou x.

```
>> syms x
```

A příkazem `solve` rovnici vyřešíme.

```
>> solve(x^3-4*x^2-x)
ans =
            0
 5^(1/2) + 2
 2 - 5^(1/2)
```

Výsledkem jsou tři kořeny $x_1 = 0$, $x_2 = \sqrt{5}+2$, $x_3 = 2-\sqrt{5}$.

15.3 Limity

Limita se spočítá příkazem `limit`.

PŘÍKLAD 70.

Př. Spočítáme limitu $\lim_{x\to 0} \frac{\sin(x)}{x}$.
Nejprve vytvoříme symbolickou proměnnou x.

```
>> syms x
```

A příkazem `limit` spočítáme limitu.

```
>> limit(sin(x)/x,0)
ans =
1
```

Výsledek je $\lim_{x\to 0} \frac{\sin(x)}{x} = 1$.

15.4 Derivace

Pro výpočet derivace máme k dispozici příkaz `diff`.

PŘÍKLAD 71.

Př. Vypočítáme derivaci funkce $y = \sin(2x)$.
Nejprve vytvoříme symbolickou proměnnou x.

```
>> syms x
```

A příkazem `diff` spočítáme derivaci.

```
>> diff(sin(2*x))
ans =
2*cos(2*x)
```

Derivace je $y' = 2\cos(2x)$.

15.5 Soustava lineárních rovnic

PŘÍKLAD 72.

Př. Najdeme řešení soustavy lineárních rovnic.

$$\begin{aligned} 3a + 4b + 7c &= -11 \\ 5a + 3b + 3c &= -4 \end{aligned}$$

Nejprve vytvoříme symbolické proměnné a, b, c.

```
>> syms a b c
```

Zadáme rovnice.

```
>> rovnice = [3*a + 4*b + 7*c + 11; 5*a + 3*b + 3*c + 4]
rovnice =
 3*a + 4*b + 7*c + 11
 5*a + 3*b + 3*c + 4
```

A rovnice vyřešíme.

```
>> reseni=solve(rovnice)
reseni =
    b: [1x1 sym]
    c: [1x1 sym]
```

Vidíme, že řešení má dvě složky, vypíšeme si je.

```
>> reseni.b
ans =
5/9 - (26*a)/9
>> reseni.c
ans =
(11*a)/9 - 17/9
```

Soustav lineárních rovnic má řešení

$$a = t,\ b = \frac{5}{9} - \frac{26t}{9},\ c = \frac{11t}{9} - \frac{17}{9}$$

K PROCVIČENÍ

1. Spočítejte limity:

   (a) $\lim\limits_{x\to\infty} \dfrac{x^3 + x^2 - 3x + 1}{x^2 - 1}$

   (b) $\lim\limits_{x\to\infty} \left(\dfrac{2x+3}{2x}\right)^x$

   (c) $\lim\limits_{x\to 2} \dfrac{x}{x-2}$

(d) $\lim_{x \to 0} \frac{\mathrm{tg}(3x)}{4x}$

2. Spočítejte derivaci funkce:

(a) $y = \mathrm{e}^{3x+1}$

(b) $y = \arcsin(3x)$

(c) $y = x^3 + 4x^2 - x + 3$

(d) $y = 2^{3x-1}$

(e) $y = \log(x + 1)$

(f) $y = \sin(x)\cos(3x)$

GEOGEBRA – PŘEHLED PŘÍKAZŮ

Množinové a logické operace, relační operátory a vybrané typy objektů

Množinové operace

operace	výběr	příklad
je prvkem	$\in$	a $\in$ seznam
je podmnožinou	$\subseteq$	seznam1 $\subseteq$ seznam2
je vlastní podmnožinou	$\subset$	seznam1 $\subset$ seznam2
rozdíl množin	$\setminus$	seznam1 $\setminus$ seznam2

Logické operace (boolovské hodnoty a, b)

operace	výběr	kláv.	příklad
a (konjunkce)	$\wedge$	&&	a $\wedge$ b nebo a && b
nebo (disjunkce)	$\vee$	||	a $\vee$ b nebo a || b
negace	$\neg$	!	$\neg$a nebo !a

Relační operátory

Rovnost, nerovnost

operace	výběr	kláv.	příklad
rovnost	$\stackrel{?}{=}$	==	a $\stackrel{?}{=}$ b nebo a == b
nerovnost	$\neq$	!=	a $\neq$ b nebo a != b

Porovnání hodnot (čísla a, b)

operace	výběr	kláv.	příklad
menší než	$<$	$<$	a $<$ b
větší než	$>$	$>$	a $>$ b
menší nebo roven	$\leq$	$<=$	a $\leq$ b nebo a $<=$ b
větší nebo roven	$\geq$	$>=$	a $\geq$ b nebo a $>=$ b

Objekty

Číslo a úhel

```
a=10.5
```
α=30°
```
α=pi/6
```

Bod a vektor

```
A=(2,3)
v=(5,9)
```

Přímka

```
p: 2*x+3*y-1=0
p: y=5*x-1
p: X = (-5, 5) + t*(4, -3)
```

Funkce

```
f(x)=2*sin(x)
f(x)=Funkce[ln(x),2,10]
```

Boolovská hodnota

`a=true` pravda, platí
`a=false` nepravda, neplatí
`a: x(A)>0`

Operace, konstanty a matematické funkce

Operace

sčítání	`+`
odčítání	`-`
násbení	`*` nebo mezera
dělení	`/`
mocnina	`^` nebo $\boxed{2}$, $\boxed{3}$
závorky	`( )`

Priorita operací

priorita	operace
1.	`^`
2.	`* /`
3.	`+ -`

Matematické funkce

absolutní hodnota $\lvert x \rvert$	`abs( )`
druhá odmocnina $\sqrt{x}$	`sqrt( )`
třetí odmocnina $\sqrt[3]{x}$	`cbrt( )`
exponenciální funkce e^x	`exp( )` nebo $\boxed{e}$`^x`
přirozený logaritmus $\ln(x)$	`ln( )` nebo `log( )`
dekadický logaritmus $\log(x)$	`lg( )` nebo `log(10, )`
logaritmus o základu a $\log_a(x)$	`log(a, )`
sinus $\sin(x)$	`sin( )`
kosinus $\cos(x)$	`cos( )`
tangens $\mathrm{tg}(x)$	`tan( )`
kotangens $\mathrm{cotg}(x)$	`cot()`
arkussinus $\arcsin(x)$	`asin( )` nebo `arcsin( )`
arkuskosinus $\arccos(x)$	`acos( )` nebo `arccos( )`
arkustangens $\mathrm{arctg}(x)$	`atan( )` nebo `arctan( )`

Konstanty

Ludolfovo číslo $\pi = 3.14\ldots$	$\boxed{\pi}$ nebo `pi` nebo Alt+p
Eulerovo číslo $e = 2.71\ldots$	$\boxed{e}$ nebo Alt+e
nekonečno ∞	$\boxed{\infty}$ nebo Alt+u
imaginární jednotka $i = \sqrt{-1}$	$\boxed{i}$ nebo Alt+i

Ostatní

x-souřadnice	`x( )`
y-souřadnice	`y( )`
zaokrouhlení	`round( )`
zaokrouhlení dolů	`floor( )`
zaokrouhlení nahoru	`ceil( )`
faktoriál	`!`
náhodné číslo mezi 0 a 1	`random( )`

Operace pro vektory

skalární součin	`*` nebo mezera
vektorový součin	$\boxed{\otimes}$

Méně používané funkce

signum	`sgn( )`
logaritmus o základu 2	`ld( )`
hyperbolický sinus	`sinh( )`
hyperbolický kosinus	`cosh( )`
hyperbolický tangens	`tanh( )`
hyperbolický kotangens	`coth( )`
argument hyperbolického sinu	`asinh( )` nebo `arcsinh( )`
argument hyperbolického kosinu	`acosh( )` nebo `arccosh( )`
argument hyperbolického tangens	`atanh( )` nebo `arctanh( )`

Vybranrané příkazy (diferenciální počet)

`Derivace[ <Funkce> ]`
`Derivace[f]` Derivace funkce f.

`Derivace[ <Funkce>, < slo> ]`
`Derivace[f,n]` Derivace n-tá funkce f.

`Funkce[ <Funkce>, <Počáteční hodnota>, <Koncová hodnota> ]`
`Funkce[f,a,b]` Funkce daná funkčním předpisem f definovaná a intervalu $\langle a, b \rangle$.

`Limita[ <Funkce>, <Hodnota x> ]`
`Limita[f,a]` Limita funkce f v bodě a tj. $\lim_{x \to a} f(x)$.

`LimitaZleva[ <Funkce>, <Hodnota x> ]`
`LimitaZleva[f,a]` Limita zprava funkce f v bodě a tj. $\lim_{x \to a^-} f(x)$.

`LimitaZprava[ <Funkce>, <Hodnota x> ]`
`LimitaZprava[f,a]` Limita zleva funkce f v bodě a tj. $\lim_{x \to a^+} f(x)$.

`NuloveBody[ <Funkce>, <Počáteční hodnota x>, <Koncová hodnota x> ]`
`NuloveBody[f,a,b]` Nulový bod funkce f ležící v intervalu $\langle a, b \rangle$.

`NulovyBod[ <Funkce>, <Původní hodnota x> ]`
`NuloveBody[f,a]` Nulový bod funkce f ležící poblíž hodnoty a.

`Extrem[ <Funkce>, <Počáteční hodnota x>, <Koncová hodnota x> ]`
`Extrem[f,a,b]` Extrém funkce f na intervalu $\langle a, b \rangle$.

`TaylorovaRada[ <Funkce>, <Hodnota x>, < slo> ]`
`TaylorovaRada[f,a,n]` Taylorův polynom funkce f v bodě a řádu n.

`Tecna[ <Bod>, <Funkce> ]`
`Tecna[T,f]` Tečna ke grafu funkce f v bodě T.

`Tecna[ <Hodnota x>, <Funkce> ]`
`Tecna[a,f]` Tečna ke grafu funkce f pro hodnotu $x = a$.

MATLAB – PŘEHLED PŘÍKAZŮ

Operace, konstanty a matematické funkce

Operace

sčítání	`+`
odčítání	`-`
násobení	`*`
dělení	`/`
mocnina	`^`
závorky	`( )`

priorita	operace
1.	`^`
2.	`* /`
3.	`+ -`

Matematické funkce

absolutní hodnota $\|x\|$	`abs( )`
druhá odmocnina $\sqrt{x}$	`sqrt( )`
exponenciální funkce e^x	`exp( )`
přirozený logaritmus $\ln(x)$	`log( )`
dekadický logaritmus $\log(x)$	`log10( )`
sinus $\sin(x)$	`sin( )`
kosinus $\cos(x)$	`cos( )`
tangens $\mathrm{tg}(x)$	`tan( )`
kotangens $\mathrm{cotg}(x)$	`cot()`
arkussinus $\arcsin(x)$	`asin( )`
arkuskosinus $\arccos(x)$	`acos( )`
arkustangens $\mathrm{arctg}(x)$	`atan( )`
arkuskotangens $\mathrm{arctg}(x)$	`acot( )`

Konstanty

Ludolfovo číslo $\pi = 3.14\ldots$	`pi`
nekonečno ∞	`inf`
neurčitý výraz	`NaN`
imaginární jednotka $i = \sqrt{-1}$	`i`

Méně používané funkce

signum	`sign( )`
logaritmus o základu 2	`log2( )`
hyperbolický sinus	`sinh( )`
hyperbolický kosinus	`cosh( )`
hyperbolický tangens	`tanh( )`
hyperbolický kotangens	`coth( )`
argument hyperbolického sinu	`asinh( )`
argument hyperbolického kosinu	`acosh( )`
argument hyperbolického tangens	`atanh( )`
argument hyperbolického kotangens	`acoth( )`

Definice vlastní matematické funkce

f=inline(*'předpis funkce'*)

Zaokrouhlování

zaokrouhlování na celé číslo	`round( )`
zaokrouhlování na nejbližší nižší celé číslo (dolů)	`floor( )`
zaokrouhlování na nejbližší vyšší celé číslo (nahoru)	`ceil( )`
zaokrouhlování na nejbližší celé číslo směrem k nule	`fix( )`

Diskrétní matematika

největší společný dělitel čísel n, k	`gcd(n,k)`
nejmenší společný násobek čísel n, k	`lcm(n,k)`
zbytek po celočíselném dělení n/k	`rem(n,k)`
faktoriál $n!$	`factorial(n)`
kombinařní číslo $\binom{n}{k}$	`nchoosek(n,k)`

Vektory a matice

`A(3,2)` prvek $a_{3,2}$ matice A
`A(3,:)` třetí řádek matice A
`A(:,2)` druh sloupec matice A

Základní informace o matici, vektoru

rozměr matice A (počet řádků , počet sloupců	`size(A)`
počet prvků matice A	`numel(A`
počet prvků vektoru $\vec{v}$	`length(v)`

Příkazy lineární algebry

determinant matice A	`det(A)`
hodnost matice A	`rank(A)`
inverzní matice A^{-1}	`inv(A)`
převedení matice A na horní trojúhelníkový tvar pomocí eliminace	`rref(A)`
transponovaná atice k matici A	`A'`

Další operace pro matice

matice typu $m \times n$ náhodných čísel z intervalu $\langle 0, 1 \rangle$	`rand(m,n)`
matice nul typu $m \times n$	`zeros(m,n)`
matice jedniček typu $m \times n$	`ones(m,n)`
jednotková matice typu $m \times n$	`eyes(m,n)`
součet prvků ve sloupcích matice A	`sum(A)`
součin prvků ve sloupcích matice A	`prod(A)`
největší hodnota ve sloupcích matice A	`max(A)`
nejmenší hodnota ve sloupcích matice A	`min(A)`
z matice A vytvoříme (po sloupcích) matici typu $m \times n$	`reshape(A,m,n)`

Maticové operace

součet matic	`+`
(např. `A+B` je matice s prvky $a_{ij} + b_{ij}$)	
rozdíl matic	`-`
(např. `A-B` je matice s prvky $a_{ij} - b_{ij}$)	
součin matic	`*`
„ řádek krát sloupec“	
pravé maticové dělení	`/`
(např. `A/B` je matice $A \cdot B^{-1}$)	
levé maticové dělení	`\`
(např. `A\B` je matice $A^{-1} \cdot B$)	
mocnina matic	`^`
(např. `A^k` je $A \cdot A \cdot \ldots \cdot A$ (k-kr)	

Operace „prvek po prvku“

součin matic „prvek po prvku“	`.*`
(např. `A.*B` je matice s prvky $a_{ij} b_{ij}$)	
pravé dělení ‚prvek po prvku“	`./`
(např. `A./B` je matice s prvky a_{ij} / b_{ij})	
levé dělení ‚prvek po prvku“	`.\`
(např. `A.\B` je matice s prvky b_{ij} / a_{ij})	
mocnina	`.^`
(např. `A.^k` je matice s prvky $(a_{ij})^k$)	

Operace pro vektory

vektorový součin vektorů $\vec{v}$ a $\vec{u}$	`cross(v,u)`
skalární součin vektorů $\vec{v}$ a $\vec{u}$	`dot(v,u)`

Programování

záhlaví funkce

`function [`*výstupy*`] =` *jméno* `(`*vstupy*`)`

rozhodovací blok

`if` *podmínka 1*
    blok příkazů
`end`

rozhodovací blok

`if` *podmínka 1*
    blok příkazů
`elseif` *podmínka 2*
    blok příkazů
`...`
`else`
    blok příkazů `end`

přepínač

`switch` *proměnná*
`case` *hodnota1*
    blok příkazů
`case` *hodnota2*
    blok příkazů
`...`
`otherwise`
    blok příkazů
`end`

cyklus se známým počtem opakování

`for` *rom ozsah hodnot*
    blok příkazů `end`

cyklus s podmínkou

`while` *podmínka*
    blok příkazů `end`

Komunikace s programem

výpis hodnoty proměnné	`disp(`*proměnná*`)`
výpis textu	`disp('`*text*`')`
načtení z klávesnice	*prom input('text')*

Relační operátory

je rovno $=$	`==`
není ovno $\neq$	`~=`
je menší $<$	`<`
je větší $>$	`>`
je menší nebo rovno $\leq$	`<=`
je větší nebo rovno $\geq$	`>=`

Logické operátory

a (konjunkce) $\wedge$	`and(a,b)` nebo `a & b`
nebo (disjunkce) $\vee$	`or(a,b)` nebo `a | b`
negace $\neg$	`not(a)` nebo `~a`

Grafy, ostatní příkazy

```
plot(x,y)
plot(x,y,'specifikace')
fplot(funkce,[a,b])
```

Specifikace grafu

Barvy	Symboly	Typy čar
`b` modrá	`.` tečky	`-` plná
`g` zelená	`o` kroužky	`:` tečkovaná
`r` červená	`x` křížky $\times$	`-.` čerchovaná
`c` světle modrá	`+` křížky +	`--` tečkovaná
`m` fialová	`*` hvězdy	
`y` žlutá	`s` čtverce	
`k` černá	`d` kosočtverce	
	`v` trojúhelníky (dolu)	
	`^` trojúhelníky (nahoru)	
	`<` trojúhelníky (vlevo)	
	`>` trojúhelníky (vpravo)	
	`p` pěticípé hvězdy	
	`h` šesticípé hvězdy	

Úprava grafu

rozsah os	`axis([ , , , ])`
poměr os 1:1	`axis equal`
nadpis obrázku	`title('text')`
popis x-ové osy	`xlabel('text')`
popis y-ové osy	`ylabel('text')`
legenda	`legend('text1','text2',...)`
zobrazení mřížky do grafu	`grid on`
více grafů jednoho obrázku	`hold on`

Konverze typů

konverze čísla na řetězec	`num2str( )`
konverze matice na řetězec	`mat2str( )`
konverze celého čísla na řetězec	`int2str( )`
konverze řetězce na číslo	`str2num( )`

Zjišťování podmínek

Pro které pozice je splněna podmínka?	`find('podmínka')`
Platí podmínka pro všechny prvky?	`all('podmínka')`
Platí odmínka pro některý prvek?	`any('podmínka')`

Příkazy pro práci s proměnnými, programem

smazání proměnných	`clear`
zavření okna s obrázkem	`close`
smazání obrazovky	`clc`
uložení textu z command window	`diary 'soubor.txt'`
seznam proměnných	`who`
seznam proměnných s informacemi	`whos`
formát výpisu dlouhý	`format long`
formát výpisu krátký	`format short`
formát výpisu ve zlomku	`format rat`

LITERATURA

[1] Rektorys K.: Přehled užité matematiky I. díl: Prometheus, 2009 , ISBN 978-80-7196-180-2

[2] Rektorys K.: Co je a k čemu je vyšší matematika. Academia Praha 2001, ISBN 8020008837

[3] Hohenwarter J., Hohenwarter M: Introduction to GeoGebra, version 4.2, 2012, http://www.geogebra.org/book/intro-en.pdf

[4] Manuály a návody k programu GeoGebra v českém jazyce http://http://wiki.geogebra.org/cs/

[5] Dušek, F.: MATLAB a SIMULINK – úvod do používání Univerzita Pardubice, 2000, ISBN 80-7194-273-1

[6] Doňar B., Zaplatílek K.: MATLAB pro začátečníky 1. díl BEN – technická literatura, 2003, ISBN 80-7300-175-6

[7] Doňar B., Zaplatílek K.: MATLAB - tvorba uživatelských aplikací 2. díl BEN - technická literatura, 2004, ISBN 80-7300-133-0