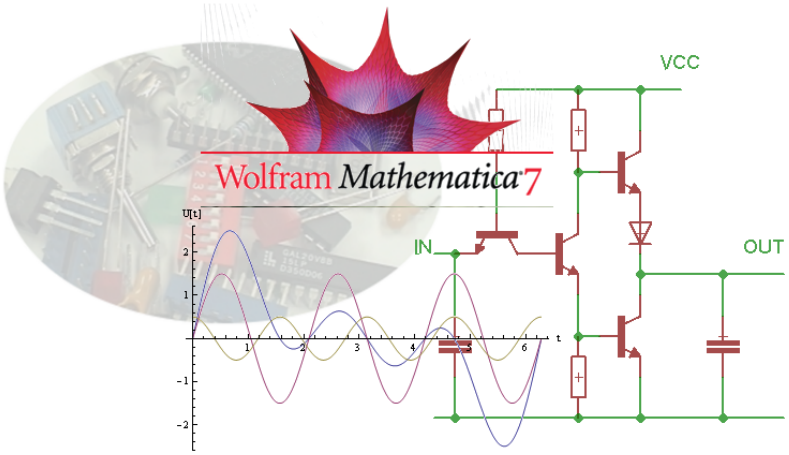
BIK – TZP

vítejte na prvním interaktivním cvičení předmětu

Technologické základy počítačů



Dříve BIK-CAO - jedná se v podstatě o identické předměty s rozdílnými názvy z důvodu nové akreditace,
ohledně Úvodu do Mathematicky není žádný rozdíl.

Vytvořeno na základě materiálů Miroslava Skrbka a Pavla Kubalíka.
Posledni úpravy: Martin Daňhel 2022

Co se naučíte v tomto předmětu ?

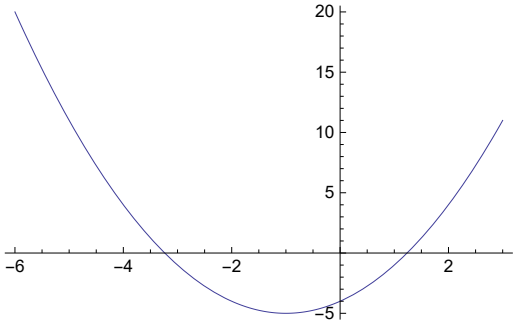
Naučíte se základům analogových a číslicových obvodů, analýza obvodů a simulaci.

Naučíte se pracovat v mocném programu Wolfram *Mathematica*, který budeme používat pro výpočty a simulaci obvodů

```
Solve[x^2 + 2 x - 4 == 0, x] // N
```

```
{{x → -3.23607}, {x → 1.23607}}
```

```
Plot[x^2 + 2 x - 4, {x, -6, 3}]
```



Požadavky pro udělení zápočtu/známky

Nutné podmínky pro získání zápočtu

- získání alespoň 20 bodů ze zápočtového testu

Možné body, které lze v semestru získat

- Zápočtový test (max 40 bodů)
- Aktivita (max 10 bodů)
- Domácí úkoly - celkem 4 sady (max 20 bodů)

Získání známky A resp. B

Pokud student získá zápočet (napíše test na víc jak 20 bodů) a dále pak získá z aktivity a domácích úkoů v součtu 40-44 bodů, má nárok na známku B ze semestru, resp. 45-70 bodů má nárok na známku A ze semestru.

Domácí úkoly a hodnocení

Hodnocení

- lze se dozvědět zde: http://grades.fit.cvut.cz

Domácí úkoly

- Po každém cvičení se zpřístupní 1 sada domácích úkolů
- Celkem jsou naplánovány 4 sady domácích úkolů
- zadání/odevzdání probíhá přes tuto aplikaci: https://ddd.fit.cvut.cz/odevzdavac/ (je potřeba kliknout na odkaz Daňhel v části BIK-TZP (Sobota))

Wolfram Mathematica

Progam Wolfram *Mathematica* je mocný nástroj pro symbolické počítání. Je ho možno použít jako

- kalkulačku
- maticový kalkulátor
- pro řešení lineárních rovnic s více proměnnými
- řešení diferenciálních rovnic
- symbolické derivování a integrování
- vykreslování grafů
- a spoustu dalších funkcí

Kde mohu pracovat s programem *Mathematica* ?

Ve cvičeních, kde je nainstalován na počítačích PC.

Program *Mathematica* si můžete (a my to velmi doporučujeme) nainstalovat na domácí počítač.

Instalační soubor si stáhnete na http://download.cvut.cz po přihlášení hlavním přístupovým heslem.

Pro program potřebujete licenci. Pro získání licence se řiďte pokyny na stránce předmětu BI-TZP.

Pokud budete mít s instalací problémy, sdělte to vašemu cvičícímu a pokuste se problém vyřešit. Ideální je přinést přímo notebook s instalovaným programem (pokud máte).

PC v učebně NTK:PU1 nebo T9:349

Přihlášení os Linux

Pro přihlášení použijte přidělené celoškolské uživatelské jméno a heslo. Pokud jste se již v jiných cvičeních přihlašovali a změnili si heslo, tak musíte použít změněné heslo.
Program Mathematica spusťte tak, že vyhledáte v systémovém menu položku Console a v příkazovém řádku napíšete mathematica.

Soubory ukládáme na /home/stud/<vaše username>
Doporučujeme si vytvořit podadresář “tzp”, aby se vám soubory nepletly se soubory z jiných předmětů. Návod: v terminálu (Console) napíšeme: cd ; mkdir tzp

Přihlášení os Windows

Pro přihlášení použijte přidělené celoškolské uživatelské jméno a heslo. Pokud jste se již v jiných cvičeních přihlašovali a změnili si heslo, tak musíte použít změněné heslo.
Program Mathematica najdete v nabídce Start.

Soubory ukládáme na disk X:/
Doporučujeme si vytvořit podadresář "tzp", aby se vám soubory nepletly se soubory z jiných předmětů.

Mathematica - vytvoření notebooku, práce s buňkami

Notebook připomíná textový editor, který nám slouží jako rozhraní k výpočetnímu jádru programu *Mathematica*. Do notebooku píšeme matematické výrazy, komentáře v podobě formátovaného textu, zobrazují se nám zde výsledky výpočtů a grafy.

Notebook vytvoříme výběrem položky menu: **Menu→File→New→Notebook**

Notebook se skládá z buněk (Cell), které připomínají řádky (bloky) textu s proměnlivou výškou. Buňka může obsahovat více standardních řádků. Mezi buňkami přecházíme **kurzorovými šipkami nahoru/dolů**. Pozor! použití klávesy **Enter** znamená přidání řádku v rámci buňky, nikoliv přechod na další buňku.

Buňka je označena na pravé straně notebooku modrou skobičkou, na kterou můžeme kliknout myší a svinout buňku, rozvinout buňku, označit ji a pak klávesou delete vymazat.

Každá buňka má styl. Například: *input* - vstup pro výpočty, *out* - výstup výpočtu, *text* - text (dokumentace výpočtu), *title* - nadpis, ...

Buňky lze spojovat a rozdělovat.

Při práci s notebookem je vhodné si otevřít panel matematických vstupů. **Palettes → Other → Basic Math Input**

Vzhledem k tomu, že je program *Mathematica* napsán jako **Client ⟷ Server**, jsou veškeré výpočty prováděny až po odeslání buňky na server s pomocí **SHIFT+ENTER**.

Čísla v programu *Mathematica*

Celá čísla (přesná)

```
{1, 2, 3, 4, -5, 1000};
```

Racionální čísla (přesná)

$$\left\{\frac{4}{3}, -\frac{1}{3}, \frac{100}{330}\right\};$$

Iracionální čísla (přesná)

$$\left\{\sqrt{6}, e^5, \pi\right\};$$

Komplexní (přesná, nepřesná pokud obsahují reálná čísla)

$$\left\{1 + 3\,i, 6 - 2\,i, 8 - \frac{6}{5}\,i\right\};$$

Reálná čísla (nepřesná)

```
{3.14, 1.41, 2.6545, -32.567};
```

Jednoduché výpočty

S programem *Mathematica* lze provádět i jednoduché výpočty.

3+4

5*8

2*π

Program *Mathematica* provádí všechny vypočty s ohledem na zachování přesnosti. Pokud chceme přesné čislo převést do nepřesné podoby (jako reálné číslo), použijeme funkci **N[2π]**. Pokud chceme počítat "nepřesně" (např. pro zrychlení výpočtu) můžeme čísla zadávat jako reálná. Např.: místo 2*π zadáme 2.*π

Různé způsoby volání funkce

- Infixová forma 3 * 5
- Postfixová forma 2π//N
- Prefixová formaN@ 2π

Vstupy lze zadávat s pomocí panelu Basic Math Input, popřípadě je možné využít klávesovou zkratku

- Děleno $\frac{2}{3}$ lze zadat jako 2 CTRL +/ a 3
- Mocnina 2^3 lze zadat jako 2 CTRL +6 a 3
- PI π lze zadat jako ESC p ESC
- Odmocnina $\sqrt{3}$ lze zadat jako CTRL +2 a 3

Proměnné

Proměnné jsou například x, y, k, L, ale také xx, x1, kocka, pes23, ...
Proměnná začíná písmenem (snažte se používat malá písmena), za kterým následuje žádný, jeden nebo více znaků nebo číslic.
Proměnná nesmí začínat číslem, např. 2x má význam 2 krát x tak, jak je užíváno v matematice (oboru).

Nepoužívat podtržítka, jako je to zvykem v jazyce C, např. x_1, podtržítko má v programu *Mathematica* zvláštní význam.
Pro jistotu **nepoužívat** ani **indexy**, mohou vést k nepředpokládanému chování programu.

```
{x, xx, kocka, pes, x1, xy3, a1, b787, abc4};
```

Pozor ! xy neznamená x krát y, ale jméno proměnné xy. Násobení zapisujeme x * y. Jako násobení se také chápe x y (mezera mezi x a y), to ale nepoužívejte, je to zdroj nepříjemných chyb.

Jména proměnných volte rozumně, spíše více písmen dávajících smysl (rovnice1, vstup, ...), zvyšuje to čitelnost programu.

Operátor =

Pro pochopení operátoru si představte, že program *Mathematica* obsahuje tabulku jmen proměnných a jejich hodnot (tabulku symbolů). Každá položka (řádek) obsahuje jméno proměnné a její hodnotu.

```
x = 1;(* do proměnné x přiřaď jedničku *)
```

Např. po provedení x=1 obsahuje tabulka řádek pro x (na začátku předpokládáme prázdnou tabulku symbolů).

proměnná	hodnota
x	1

```
y = 2;(* do proměnné y přiřaď dvojku *)
```

proměnná	hodnota
x	1
y	2

Nyní jsou v tabulce symbolů dvě proměnné x a y s přiřazenými hodnotami.

Změna obsahu proměnné

```
x = 1;
```

```
y = 2;
```

Tabulka symbolů po provedení výše uvedených přiřazení

proměnná	hodnota
x	1
y	2

```
x = 5;
```

Nyní jsme změnili obsah proměnné x.

proměnná	hodnota
x	5
y	2

Vyhodnocení jednoduchých výrazů

```
x = 1;
y = 2;
```

Tabulka symbolů po provedení výše uvedených přiřazení

proměnná	hodnota
x	1
y	2

```
z = x + y;
```

Výraz x+y na pravé straně operátoru přiřazení se vyhodnotí tak, že se *Mathematica* pro x podívá do tabulky symbolů a nahradí ho jeho hodnotou (1 v našem případě), totéž udělá pro y a pak teprve provede součet. Pak bude tabulka symbolů vypadat takto:

proměnná	hodnota
x	1
y	2
z	3

Funkční program

```
x = 1;
y = 2;
z = x + y;
```

```
z (* vytiskni obsah z *)
```

```
3
```

Zadání k vlastnímu řešení:

Vypočtěte hodnotu výrazu $x^2 + 5\,y - 6\,z$ pro hodnoty x=10, y=25, z=8
Výsledek bude uložen v proměnné r.

Záludnosti při vyhodnocení buněk v notebooku

Buňky v notebooku se vyhodnocují na vyžádání, a to buď

- jednotlivě (v každé buňce stiskneme klávesy Shift-Enter)

pořadí vyhodnocení je dáno pořadím výběru buněk k vyhodnocení. Pokud vyhodnocujeme buňky na přeskáčku, může se stát, že dostaneme nesmyslné výsledky, protože si jádro vytváří tabulku symbolů v pořadí tak, jak vyhodnocujeme buňky a ne tak, jak jsou zapsány v notebooku. Navíc je možné některé pravidlo vyhodnotit vícekrát. Pokud děláme navíc v notebooku zásadnější změny, tak i přes to, že veškeré změny odesíláme do jádra (Shift-Enter), tak stav tabulky symbolů před změnami nám může způsobit nesmyslný výsledek.

Proveďte:

Menu→Evaluation→QuitKernel→Local (zajistí vynulování tabulky symbolů, začínáme tedy s "čistým stolem")

Shift+Enter na buňce z=x+y

výsledek je x+y, protože ani x ani y nebylo zatím definováno (x=1 a y=2 nebylo ještě posláno do jádra).

Shift+Enter na buňce y=2; a potom na z = x+y

výsledek je x+2, protože y má hodnotu 2

Shift+Enter na buňce x=1; a potom na z = x+y

výsledek je 3, který jsme očekávali

- hromadně (**Menu→Evaluation→EvaluateNotebook**)

Vyhodnotí všechny buňky v notebooku směrem od začátku do konce. Toto je významné, ale stále jen částečné řešení předchozího problému. Stále ještě hrozí, že před vyvoláním **Menu→Evaluation→EvaluateNotebook** nebyla tabulka symbolů prázdná a zvláště pokud používáme jednu proměnnou (jedno jméno) vícenásobně pro různé a nesouvisející výrazy, můžeme se stejně dobrat chybných výsledků. **Pokud chceme mít jistotu, pak musíme inicializovat kernel (Menu→Evaluation→QuitKernel→Local) a vyhodnotit všechny buňky v notebooku Menu→Evaluation→EvaluateNotebook**.

Pozor! Pokud máme otevřeno více notebooků, pak všechny využívají jeden kernel. Tzn. proměnná nadefinovaná v jednom notebooku je viditelná v jiném notebooku.
Není to však pravidlem - v některých operačních systémech se při otevření uloženého notebooku spustí i nový samostatný kernel.

Řešení problémů s vyhodnocováním buněk

Před každou skupinou buněk, která řeší nějaký matematický problém, vymažeme funkcí `ClearAll["Global`*"];` hodnoty všech proměnných v tabulce symbolů.

```
ClearAll["Global`*"];
x = 1;
y = 2;
z = x + y
```

```
3
```

Pak pokud použijeme funkci **Menu→Evaluation→EvaluateNotebook**, máme jistotu, že se nám všechny buňky vyhodnotí ve správném pořadí a navíc budou správně inicializovány.
Toto si zvykneme psát do našich notebooků, abychom stále nenaráželi na problémy s vyhodnocováním buněk.
V případě, že je v notebooku více různých menších příkladů (typicky testy), umístíme každý z nich do jedné samostatné buňky a na její začátek vložíme funkci `ClearAll[“Global`*”];`

ClearAll můžeme použít i v případě, že potřebujeme smazat hodnoty pouze u vybraných promměnných:

```
ClearAll["Global`*"];
x = 1;
y = 1;
z = x + y;
ClearAll[x, y];
x
y
z (*z jsme nevymazali, tudiz si svou hodnotu pamatuje*)
```

```
x
```

```
y
```

```
2
```

Funkce (vestavěné)

Funkce v programu *Mathematica* hrají stejnou roli jako funkce v matematice.

Jsou zde pouze odlišnosti v zápisu. Např. sin(x) se zapíše jako `Sin[x]`.

Vestavěné funkce začínají vždy velkým písmenem a argumenty (parametry) jsou uzavřeny v hranatých závorkách.

Funkce má určitý počet pevně určených parametrů a ostatní jsou volitelné a nezávisí na jejich pořadí. Volitelné parametry se zapisují jako pravidla, např. `PlotRange→All`.

Např. funkce `Sin[x]` má jeden pevný parametr. Funkce `Plot[funkce, rozsah, PlotRange->All, ...]` má dva pevné parametry funkce a rozsah, další parametry jsou volitené.

Význam jednotlivých parametrů funkcí nalezneme v helpu.

```
? Sin
```

Sin[z] gives the sine of z. ≫

```
? Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.

Plot[{f_1, f_2, …}, {x, x_{min}, x_{max}}] plots several functions f_i. ≫

Pokud neznáme plně jméno funkce, stačí naznačit (napsat jen část) např. Arc a pak Ctrl-k. *Mathematica* napoví (objeví se pop-up menu s možnostmi).
V aktuálních verzích Mathamaticy se toto menu ukazuje během psaní názvu funkce už automaticky.

Funkce definované uživatelem

Uživatel programu *Mathematica* si může definovat vlastní funkci.

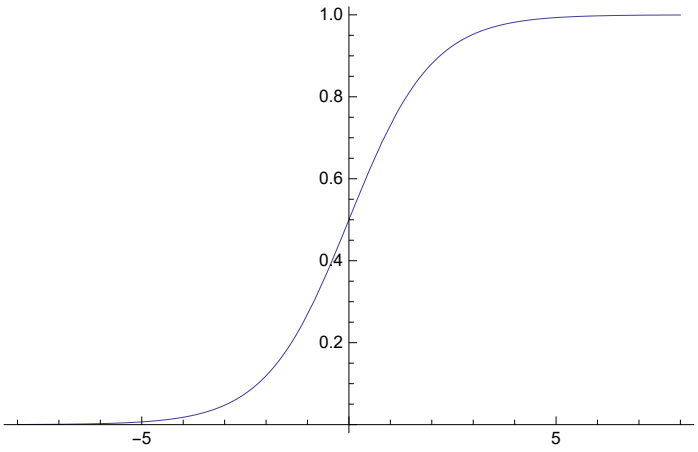
Parametry na levé straně přiřazení musí končit podtržítkem x_, yy_, případně x:_ nebo yy:_. Jedná se tzv. formální parametry, za které se v místě užití dosadí parametry skutečné. Formální parametry na pravé straně přiřazení podtržítko nemají.

Užívá se zde jiného operátoru přiřazení (:=). Ten je interpretován odlišně oproti operátoru =. U operátoru = je k levé straně (jméno proměnné, funkce, ...) v tabulce symbolů přiřazena hodnota (vyhodnocený výraz na pravé straně) okamžitě. U operátoru := dojde k nahrazení levé strany pravou až v době užití, kdy je znám ke každému formálnímu parametru odpovídající parametr skutečný. Např. `sigmoida[a+3,1]` se nahradí $\frac{1}{1+e^{-1*(a+3)}}$, kde výraz a+3 a konstanta 1 jsou skutečnými parametry.

Definujme logistickou funkci zvanou Sigmoida.

```
sigmoida[x_, γ_] := 1/(1 + e^(-x*γ));
```

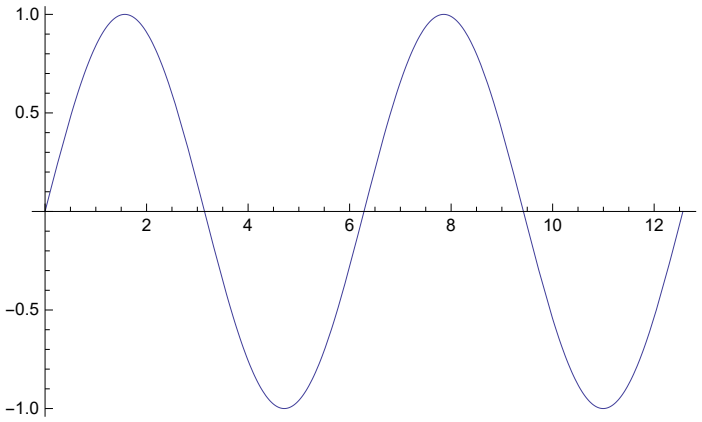
```
Plot[sigmoida[a, 1], {a, -8, 8}]
```



Kreslení grafů

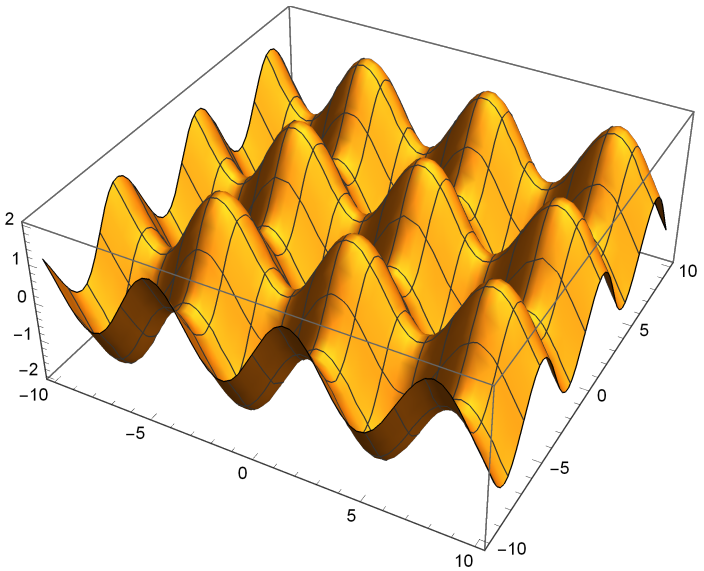
Pro kreslení grafů (2D) používáme funkci Plot. První parametr je funkce, kterou chceme vykreslit. Druhý parametr je obor hodnot funkce pro zvolenou nezávislou proměnnou. Druhý parametr je vždy seznam obsahující po řadě proměnnou, minimální hodnotu a maximální hodnotu.

```
Plot[Sin[x], {x, 0, 4 π}]
```



Pro kreslení grafů (3D) používáme funkci Plot3D. První parametr je funkce, kterou chceme vykreslit. Druhý a třetí parametr jsou obory hodnot funkcí pro zvolené nezávislé proměnné. Druhý i třetí parametr jsou vždy seznamy obsahující po řadě proměnnou, minimální hodnotu a maximální hodnotu.

```
Plot3D[Sin[x] + Sin[y], {x, -10, 10}, {y, -10, 10}]
```



Existují i další typy grafů (např. LogPlot, ParametricPlot, ListPlot, ListPlot3D, apod.) Detaily o těchto typech grafů lze nalézt v nápovědě.

Seznamy (List)

Seznam je datová struktura, která dovoluje sdružit více položek a definuje nad nimi množinu specifických operací.

Příklad seznamu:

```
ClearAll["Global`*"]
```

```
basaPiv = {Gambrinus, Plzen, Staropramen}
basaPiv2 = {Bernard, Svijany, Rohozec}
hromadaPiv = {basaPiv, basaPiv, basaPiv2, Starobrno, Radegast}
skladPiv = {hromadaPiv, basaPiv2}
```

```
{Gambrinus, Plzen, Staropramen}
```

```
{Bernard, Svijany, Rohozec}
```

```
{{Gambrinus, Plzen, Staropramen}, {Gambrinus, Plzen, Staropramen},
 {Bernard, Svijany, Rohozec}, Starobrno, Radegast}
```

```
{{{Gambrinus, Plzen, Staropramen}, {Gambrinus, Plzen, Staropramen},
  {Bernard, Svijany, Rohozec}, Starobrno, Radegast}, {Bernard, Svijany, Rohozec}}
```

Vybrané operace se seznamem:

```
(* první pivo v base *)
First[basaPiv]
```

```
Gambrinus
```

```
(* přidej nové pivo do basy *)
novaBasaPiv = Append[basaPiv, jesteJednaPlzen]
```

```
{Gambrinus, Plzen, Staropramen, jesteJednaPlzen}
```

```
? List
```

{e_1, e_2, …} is a list of elements. ≫

Přístup k prvkům seznamu (indexace)

K jednotlivým položkám listu lze přistupovat pomocí [[]] (po programátorsku: indexy polí se píšou do dvojitých hranatých závorek)

```
basaPiv[[2]]
```

```
Plzen
```

```
hromadaPiv[[3]]
```

```
{Bernard, Svijany, Rohozec}
```

```
hromadaPiv[[3, 2]]
```

```
Svijany
```

```
skladPiv[[1]]
```

```
{{Gambrinus, Plzen, Staropramen}, {Gambrinus, Plzen, Staropramen},
 {Bernard, Svijany, Rohozec}, Starobrno, Radegast}
```

```
skladPiv[[1, 2, 3]]
```

```
Staropramen
```

Vícerozměrné seznamy

Příkladem vícerozměrného seznamu může být matice $\begin{pmatrix} 3 & 12 & 5 \\ 7 & 9 & 1 \\ 22 & 11 & 10 \end{pmatrix}$

Seznam odpovídající matici je sestaven po řádcích

```
matice = {{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
```

```
{{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
```

```
prvekTretiRadekDruhySloupec = matice[[3, 2]]
```

```
11
```

```
druhyRadek = matice[[2]]
```

```
{7, 9, 1}
```

```
prvniSloupec = matice[[All, 1]]
```

```
{3, 7, 22}
```

Matice (i vektory) lze také zapsat pomocí palety Basic Math Assistant - sekce Basic Commands - ikona matice $\begin{pmatrix} \square & \square \\ \square & \square \end{pmatrix}$

Paleta také obsahuje tlačítka pro přidávaní řádků (CTRL + ENTER) a sloupců (CTRL + ,).

```
matice2 = ⎛ 3  12  5  ⎞
          ⎜ 7   9  1  ⎟
          ⎝ 22 11 10 ⎠
```

```
matice2[[3, 2]]
```

```
{{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
```

```
11
```

Funkce operující nad listy

```
? Flatten
```

Flatten[*list*] flattens out nested lists.

Flatten[*list*, *n*] flattens to level *n*.

Flatten[*list*, *n*, *h*] flattens subexpressions with head *h*.

Flatten[*list*, {{s_{11}, s_{12}, …}, {s_{21}, s_{22}, …}, …}]

flattens *list* by combining all levels s_{ij} to make each level *i* in the result. ≫

```
velkaHromadaPiv = Flatten[skladPiv]
```

```
{Gambrinus, Plzen, Staropramen, Gambrinus, Plzen, Staropramen,
 Bernard, Svijany, Rohozec, Starobrno, Radegast, Bernard, Svijany, Rohozec}
```

```
? Sort
```

Sort[*list*] sorts the elements of *list* into canonical order.

Sort[*list*, *p*] sorts using the ordering function *p*. ≫

```
srovnanaHromadaPiv = Sort[velkaHromadaPiv]
```

```
{Bernard, Bernard, Gambrinus, Gambrinus, Plzen, Plzen, Radegast,
 Rohozec, Rohozec, Starobrno, Staropramen, Staropramen, Svijany, Svijany}
```

```
? Union
```

Union[$list_1$, $list_2$, …] gives a sorted list of all the distinct elements that appear in any of the $list_i$.

Union[*list*] gives a sorted version of a list, in which all duplicated elements have been dropped. ≫

```
prebranaHromadaPiv = Union[velkaHromadaPiv]
```

```
{Bernard, Gambrinus, Plzen, Radegast, Rohozec, Starobrno, Staropramen, Svijany}
```

Další funkce lze najít v nápovědě (List Manipulation).

◄ | ►

Funkce generující listy

? Range

Range[i_{max}] generates the list {1, 2, …, i_{max}}.

Range[i_{min}, i_{max}] generates the list {i_{min}, …, i_{max}}.

Range[i_{min}, i_{max}, di] uses step di. ≫

```
Range[3, 30]
Range[3, 30, 6]
```

```
{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30}
```

```
{3, 9, 15, 21, 27}
```

? Table

Table[$expr$, n] generates a list of n copies of $expr$.

Table[$expr$, {i, i_{max}}] generates a list of the values of $expr$ when i runs from 1 to i_{max}.

Table[$expr$, {i, i_{min}, i_{max}}] starts with i = i_{min}.

Table[$expr$, {i, i_{min}, i_{max}, di}] uses steps di.

Table[$expr$, {i, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….

Table[$expr$, {i, i_{min}, i_{max}}, {j, j_{min}, j_{max}}, …] gives a nested list. The list associated with i is outermost. ≫

```
Table[x^2, {x, 0, 10}]
```

```
{0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

Další funkce lze najít v nápovědě (Constructing Lists).

◄ | ►

Řešení soustavy lineárních rovnic

S řešením soustavy lineárních rovnic o dvou neznámých jste se setkali na střední škole.

Řešte následující soustavu rovnic

$3x+2y=0$

$6x-4y=1$

Použijeme funkci Solve

```
ClearAll["Global`*"]
Solve[{3 x + 2 y == 0, 6 x - 4 y == 1}, {x, y}]
```

$$\left\{\left\{x \to \frac{1}{12}, y \to -\frac{1}{8}\right\}\right\}$$

Matematické rovná se (=) se v programu *Mathematica* zapisuje jako == (dvě rovná se za sebou). Výsledkem výrazu je True (pravda) nebo False (nepravda). Například 5==6 dává False, ale 5==(6-1) dává True. Operátor == má prakticky stejný význam jako v jazycích C nebo Java.

Druhým parametrem funkce Solve je seznam neznámých proměnných.

Řešení soustavy diferenciálních rovnic

Diferenciální rovnice a jejich soustavy lze řešit v Mathematice buď analyticky pomocí funkce DSolve (výsledkem je funkce) nebo numericky pomocí NDSolve (výsledkem je popis grafu funkce pomocí interpolační funkce).
Hlavním rozdílem oproti klasickým lineárním rovnicím je nahrazení neznámých (x, y) za funkce (x[t], y[t]) a jejich derivace.

Řešte následující soustavu rovnic

```
3x[t]+2y[t]=x'[t]
6x[t]-4y[t]=1
```

Použijeme funkci DSolve

```
ClearAll["Global`*"]
DSolve[{3 x[t] + 2 y[t] == x'[t], 6 x[t] - 4 y[t] == 1}, {x[t], y[t]}, t]
```

$$\left\{\left\{x[t] \to \frac{1}{25}\left(\frac{25}{12} + e^{6t} C[1]\right), y[t] \to -\frac{1}{4} + \frac{3}{50}\left(\frac{25}{12} + e^{6t} C[1]\right)\right\}\right\}$$

V rovnicích je opět nutno použít == (dvě rovná se za sebou).

Druhým parametrem funkce DSolve je seznam neznámých funkcí, třetí parametr je tzv. nezávisle proměnná, tj. ta proměnná, která se vyskytuje uvnitř závorek všech neznámých funkcí.

Ve výsledku řešení diferenciálních rovnic se vyskytuje konstanta C[1], kterou lze odstranit zadáním tzv. počáteční podmínky do soustavy. Soustava diferenciálních rovnic sice vypočítá tvar výsledné funkce, ale její přesné umístění je určeno právě počáteční podmínkou.

In[•]:=
```
ClearAll["Global`*"]
resDS = DSolve[{3 x[t] + 2 y[t] == x'[t], 6 x[t] - 4 y[t] == 1, x[0] == 0}, {x[t], y[t]}, t]
```

Out[•]=
$$\left\{\left\{x[t] \to \frac{1}{12}\left(1 - e^{6t}\right), y[t] \to \frac{1}{8}\left(-1 - e^{6t}\right)\right\}\right\}$$

◄ | ►

Numerické řešení soustavy diferenciálních rovnic

Parametry numerického řešení pomocí NDSolve jsou velmi podobné jako u analytického řešení pomocí DSolve, pouze u třetího parametru je třeba zadat rozsah odkud kam bude výpočet probíhat. **Výsledek je pak možno použít jen v tomto rozsahu!**

```
In[•]:= resNDS =
         NDSolve[{3 x[t] + 2 y[t] == x'[t], 6 x[t] - 4 y[t] == 1, x[0] == 0}, {x[t], y[t]}, {t, 0, 1}]
```

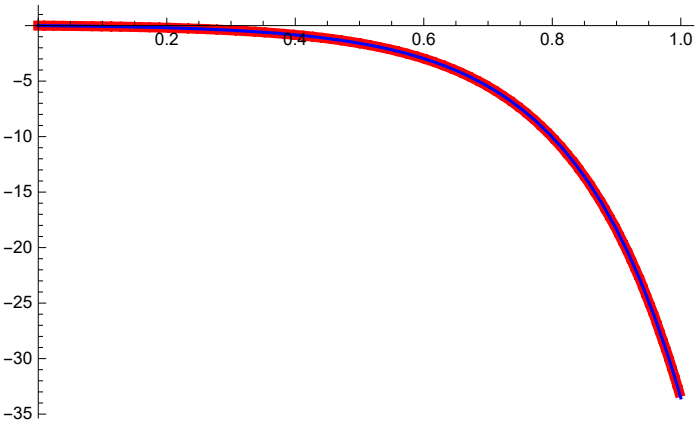
```
Out[•]= {{x[t] → InterpolatingFunction[ Domain: {{0., 1.}}
                                        Output: scalar ][t],

          y[t] → InterpolatingFunction[ Domain: {{0., 1.}}
                                        Output: scalar ][t]}}
```

Ve výsledku numerického řešení jsou vidět náhledy tvarů grafů všech neznámých funkcí.

Porovnání analytického a numerického řešení:

```
In[•]:= Plot[{x[t] /. resNDS[[1]], x[t] /. resDS[[1]]},
         {t, 0, 1}, PlotStyle → {{Red, Dotted, Thickness[0.015]}, {Blue}}]
```

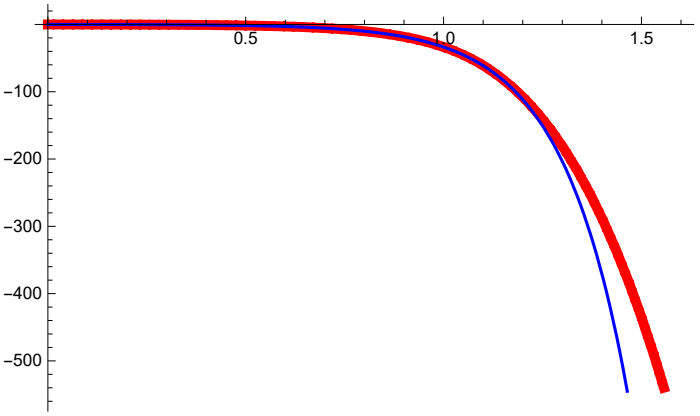
Out[•]=



Porovnání analytického a numerického řešení mimo rozsah NDSolve:

```
In[•]:= Plot[{x[t] /. resNDS[[1]], x[t] /. resDS[[1]]},
         {t, 0, 1.6}, PlotStyle → {{Red, Dotted, Thickness[0.015]}, {Blue}}]
```

Out[•]=



Proměnné - rozšířené možnosti

Hodnotou proměnné může být nejenom číslo, ale prakticky jakýkoliv výraz či řetězec. Například:

```
x = 5
budova = NTK
```

```
5
```

```
NTK
```

Proměnné lze proto využít například pro pojmenování složitých výrazů, například rovnic. Soustavu lineárních rovnic

2a + 3b + 4.2c == 5
3a + 15.2b + 8.7c == 80.1
4a + 8.9b + 17.2c == 11

lze proto řešit buď takto:

```
Solve[{2 a + 3 b + 4.2 c == 5, 3 a + 15.2 b + 8.7 c == 80.1, 4 a + 8.9 b + 17.2 c == 11}]
```

```
{{a → -3.63596, b → 7.29908, c → -2.29174}}
```

nebo takto:

```
rce1 = 2 a + 3 b + 4.2 c == 5;
rce2 = 3 a + 15.2 b + 8.7 c == 80.1;
rce3 = 4 a + 8.9 b + 17.2 c == 11;
Solve[{rce1, rce2, rce3}]
```

```
{{a → -3.63596, b → 7.29908, c → -2.29174}}
```

proměnná	hodnota
rce1	2 a + 3 b + 4.2 c == 5
rce2	3 a + 15.2 b + 8.7 c == 80.1
rce3	4 a + 8.9 b + 17.2 c == 11

Pravidla

```
ClearAll[x]
```

```
def = {x → 2, y → 1, z → 4}
moje = {x → 5}
{x, y, z} /. moje
({x, y, z} /. moje) /. def
```

```
{x → 2, y → 1, z → 4}
```

```
{x → 5}
```

```
{5, y, z}
```

```
{5, 1, 4}
```

"→" se napíše tak, že zadáte - následované > a pak pokračujete v psaní. Stejný znak se používá i nepovinných parametrů (např.: Plot[Sin[x],{x,0,1},PlotStyle→Red])

Pozor při kopírování z pdf souborů, textových polí a ze stránek Courses! Často zkopírují netisknutelné a neviditelné znaky - problémy jsou zejména právě se znakem →, ale i s uvozovkami, indexy apod.

In[•]:=
```
spatne = x → 2; (*RightArrow*)
dobre = x → 2; (*Rule*)
x /. spatne
x /. dobre
```

ReplaceAll: {x → 2} is neither a list of replacement rules nor a valid dispatch table, and so cannot be used for replacing.

Out[•]=
```
x /. x → 2
```

Out[•]=
```
2
```

In[•]:=
```
FullForm[spatne]
FullForm[dobre]
```

Out[•]//FullForm=
```
RightArrow[x, 2]
```

Out[•]//FullForm=
```
Rule[x, 2]
```

◄ | ►

Použití pravidel

S pravidly se setkáme nejčastěji jako s výsledkem funkce Solve (DSolve i NDSolve):

```
rce1 = 2 a + 3 b + 4.2 c == 5;
rce2 = 3 a + 15.2 b + 8.7 c == 80.1;
rce3 = 4 a + 8.9 b + 17.2 c == 11;
reseni = Solve[{rce1, rce2, rce3}]
a /. reseni
a + 2 * b /. reseni[[1]]
```

```
{{a → -3.63596, b → 7.29908, c → -2.29174}}
```

```
{-3.63596}
```

```
10.9622
```

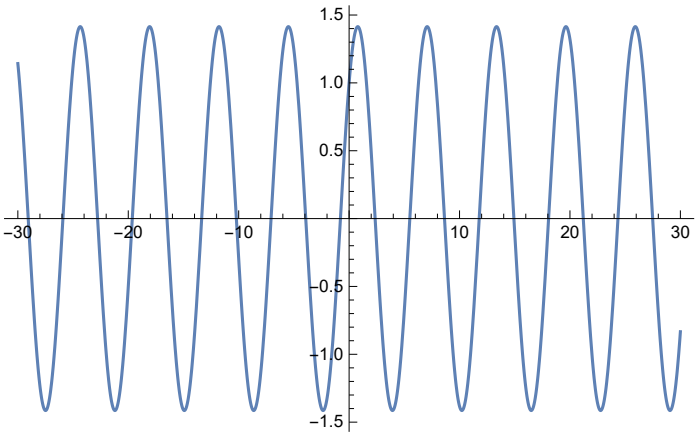
Je ale možné je použít i jako jednorázové dosazení v případě, že nechceme nedopatřením přepsat původní výraz

```
ClearAll["Global`*"]
```

Při prvním odpálení následující buňky je výsledek v pořádku, ale při druhém se vykreslí už jen rovná čára - do původního výrazu se při druhém odpálení uložila již dosazená konstanta x.

```
mujVyraz = Sin[x] + Cos[x]
Plot[mujVyraz, {x, -30, 30}]
x = π;
mujVyraz
```

```
Cos[x] + Sin[x]
```



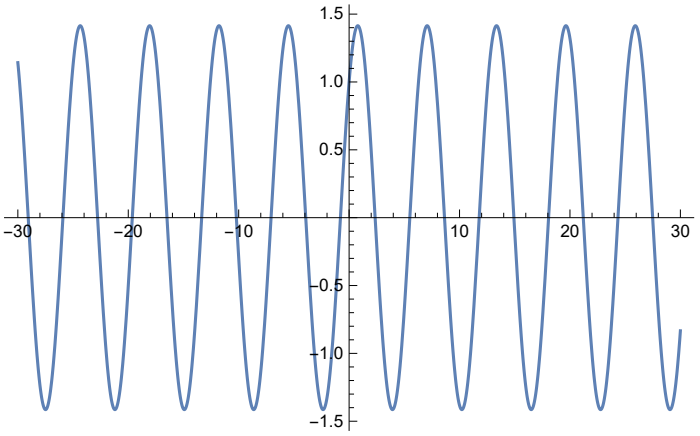

```
-1
```

```
ClearAll["Global`*"]
```

Následující buňku lze odpalovat opakovaně beze změny výsledku - aplikací pravidla se do x ani do výrazu nic neuložilo

```
mujVyraz = Sin[x] + Cos[x]
Plot[mujVyraz, {x, -30, 30}]
mujVyraz /. x → π
```

```
Cos[x] + Sin[x]
```




```
-1
```

Práce s maticemi (vícerozměrnými seznamy)

Zobrazení matice v matematickém tvaru pomocí MatrixForm:

```
ClearAll["Global`*"];
matice = {{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
MatrixForm[matice]
```

```
{{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
```

$$\begin{pmatrix} 3 & 12 & 5 \\ 7 & 9 & 1 \\ 22 & 11 & 10 \end{pmatrix}$$

Pozor: Neukládejte MatrixForm[matice] do proměnné - Mathematica by pak už takovou matici neuměla dále zpracovat.
MatrixForm (a ostatní příkazy typu *Form - viz nápověda) slouží pouze pro zobrazení výsledku v čitelnější formě.

Tip: Pro příkazy typu *Form je výhodná postfixová notace zápisu.

```
matice // MatrixForm
```

$$\begin{pmatrix} 3 & 12 & 5 \\ 7 & 9 & 1 \\ 22 & 11 & 10 \end{pmatrix}$$

Maticové operátory a výpočty (viz nápověda Matrix Operations):

```
matice = {{3, 12, 5}, {7, 9, 1}, {22, 11, 10}};
matice2 = {{1, 2, 5}, {7, 8, 1}, {1, 4, 3}};
matice + matice2 // MatrixForm (*součet*)
Transpose[matice] // MatrixForm (*tranzpozice*)
{a, b, c}.{x, y, z} (*skalární součin*)
{{a, b}, {c, d}} * {x, y} // MatrixForm (*maticový součin*)
```

$$\begin{pmatrix} 4 & 14 & 10 \\ 14 & 17 & 2 \\ 23 & 15 & 13 \end{pmatrix}$$

$$\begin{pmatrix} 3 & 7 & 22 \\ 12 & 9 & 11 \\ 5 & 1 & 10 \end{pmatrix}$$

$$a\,x + b\,y + c\,z$$

$$\begin{pmatrix} a\,x & b\,x \\ c\,y & d\,y \end{pmatrix}$$

Práce s maticemi (vícerozměrnými seznamy)

Zobrazení matice v matematickém tvaru pomocí MatrixForm:

```
ClearAll["Global`*"];
matice = {{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
MatrixForm[matice]
```

```
{{3, 12, 5}, {7, 9, 1}, {22, 11, 10}}
```

$$\begin{pmatrix} 3 & 12 & 5 \\ 7 & 9 & 1 \\ 22 & 11 & 10 \end{pmatrix}$$

Pozor: Neukládejte MatrixForm[matice] do proměnné - Mathematica by pak už takovou matici neuměla dále zpracovat.
MatrixForm (a ostatní příkazy typu *Form - viz nápověda) slouží pouze pro zobrazení výsledku v čitelnější formě.

Tip: Pro příkazy typu *Form je výhodná postfixová notace zápisu.

```
matice // MatrixForm
```

$$\begin{pmatrix} 3 & 12 & 5 \\ 7 & 9 & 1 \\ 22 & 11 & 10 \end{pmatrix}$$

Maticové operátory a výpočty (viz nápověda Matrix Operations):

```
matice = {{3, 12, 5}, {7, 9, 1}, {22, 11, 10}};
matice2 = {{1, 2, 5}, {7, 8, 1}, {1, 4, 3}};
matice + matice2 // MatrixForm (*součet*)
Transpose[matice] // MatrixForm (*tranzpozice*)
{a, b, c}.{x, y, z} (*skalární součin*)
{{a, b}, {c, d}} * {x, y} // MatrixForm (*maticový součin*)
```

$$\begin{pmatrix} 4 & 14 & 10 \\ 14 & 17 & 2 \\ 23 & 15 & 13 \end{pmatrix}$$

$$\begin{pmatrix} 3 & 7 & 22 \\ 12 & 9 & 11 \\ 5 & 1 & 10 \end{pmatrix}$$

$$a\,x + b\,y + c\,z$$

$$\begin{pmatrix} a\,x & b\,x \\ c\,y & d\,y \end{pmatrix}$$

Derivace a integrály

Můžeme je napsat pomocí funkcí, nebo pomocí palety BasicMath Assistant v sekci Basic Commands:

```
D[x^n, x]
```

$$\partial_x x^n$$

$$n\, x^{-1+n}$$

$$n\, x^{-1+n}$$

Derivaci uživatelsky definované funkce můžeme napsat i pomocí apostrofu:

```
mojeFce[x_] := x^n
mojeFce'[x]
```

$$n\, x^{-1+n}$$

```
Integrate[n x^(-1+n), x]
```

$$\int n\, x^{-1+n}\, dx$$

$$x^n$$

$$x^n$$

Určitý integrál:

```
Integrate[e^-x, {x, 1, Infinity}]
```

$$\int_1^\infty e^{-x}\, dx$$

$$\frac{1}{e}$$

$$\frac{1}{e}$$

Boolova algebra

Výrazy můžeme psát v programátorském i matematickém tvaru:

```
a || b && ! c
boolVyraz = a ∨ b ∧ ¬ c
```

```
a || (b && ! c)
```

```
a || (b && ! c)
```

```
boolVyraz // TraditionalForm
```

$a \vee (b \wedge \neg c)$

Generování tabulky a mapy z funkce:

```
BooleanTable[boolVyraz, {a, b, c}] (*začíná od samých 1 (True)*)
```

```
{True, True, True, True, False, True, False, False}
```

```
BooleanTable[boolVyraz, {a}, {b}, {c}] // TableForm
```

```
True    True
True    True
False   False
True    False
```

Minimalizace výrazu a test tautologie:

```
BooleanMinimize[a ∧ b ∨ ¬ a ∧ b]
```

```
b
```

```
TautologyQ[(a || b) || (! a && ! b), {a, b}]
```

```
True
```

Další funkce lze najít v nápovědě (Boolean Computation).

Wolfram Alpha

K dispozici na adrese http://www.wolframalpha.com/ i bez nainstalované Mathematicy.
Výrazy můžeme zadávat v syntaxi Mathematicy i v běžné angličtině.
Z Mathematicy lze zadat dotaz pro Wolfram Alpha pomocí znaků ==, které zadáme na začátku nové buňky.

ursa major

iss today

Answer to the Ultimate Question of Life, the Universe, and Everything

Doctor John Zoidberg-like curve

Wolfram Alpha

K dispozici na adrese http://www.wolframalpha.com/ i bez nainstalované Mathematicy.
Výrazy můžeme zadávat v syntaxi Mathematicy i v běžné angličtině.
Z Mathematicy lze zadat dotaz pro Wolfram Alpha pomocí znaků ==, které zadáme na začátku nové buňky.

ursa major

iss today

Answer to the Ultimate Question of Life, the Universe, and Everything

Doctor John Zoidberg-like curve