Základy práce s programem Maple I

1. Zadávání příkazů

Na jednom řádku může být i více příkazů, každý příkaz musí být však ukončen středníkem (;) nebo dvojtečkou (:) a řádek potvrzen klávesou `Enter`. Například **2 a 3 sečteme** příkazem „`2+3;`"
Nový výpočet s proměnnými doporučuji inicializovat příkazem „`restart;`", který vymaže všechny proměnné. Tím se vyhneme případným komplikacím se starými hodnotami při opakování výpočtu.
Nápovědu ke konkrétnímu příkazu získáme zadáním příkazu ve tvaru `?jméno` (zde nemusíme psát středník, pouze potvrdíme klávesou `Enter`). Například, zadáním „`?plot`" získáme kompletní nápovědu k příkazu `plot` i s odkazy na příbuzná témata.
Nápovědu (`Help`) doporučuji hojně využívat. Velkým zdrojem informací a inspirace jsou konkrétní příklady použití, které jsou součástí nápovědy ke každému příkazu. Příklady je možno pomocí `Ctrl+C, Ctrl+V` kopírovat do pracovního okna programu Maple.

2. Přibližná hodnota výrazu

Maple pracuje v symbolickém režimu. Pokud potřebujeme přibližné vyjádření hodnoty nějakého výrazu desetinným rozvojem, můžeme použít příkaz „`evalf(výraz, počet cifer);`". Například po zadání „`evalf(Pi,20);`" dostaneme hodnotu π na 19 desetinných míst.

3. Knihovny funkcí

Velká část příkazů programu Maple je uložena v tzv. knihovnách funkcí. Například příkazy pro počítání s vektory a maticemi jsou uloženy v knihovně `linalg`. Jsou dvě možnosti, jak se k takovým příkazům dostat.
1) Načíst do paměti celou knihovnu příkazem „`with`". Například uvedenou knihovnu `linalg` načteme příkazem „`with(linalg);`". Ukončíme-li příkaz středníkem, je vypsán seznam všech funkcí knihovny. Pokud o něj nestojíme, oceníme dvojtečku.
2) Aniž bychom knihovnu otvírali, můžeme její konkrétní funkci zavolat příkazem ve tvaru „`jméno knihovny[jméno funkce](parametry funkce);`".
Viz například volání funkce `linalg[genmatrix]`, které je uvedeno v partii věnované řešení soustav rovnic.

4. Funkce jedné proměnné

Definice funkce (například $f\colon y = x^2 - 4x$):

`> f:=x->x^2-4*x;` nebo `f:=unapply(x^2-4*x,x);`

Graf funkce:

`> plot(f(x),x);`

Graf funkce s definovaným rozsahem os dostaneme příkazem „`plot(f(x),x=-5..5,y=-4..4);`"
Příkaz `plot` může mít i další parametry. U nespojitých funkcí například oceníme parametr `discont=true` (více viz nápověda, `Options`). Porovnejte příkazy

`> plot(tan(x),x=-2*Pi..2*Pi,y=-4..4);` a

`> plot(tan(x),x=-2*Pi..2*Pi,y=-4..4,discont=true);`

Poznámka: `discont` je také funkce Maple pro výpočet bodů nespojitosti dané funkce, více viz nápověda.

Derivace funkce:
První, respektive n-tá derivace funkce $f(x)$ se určí příkazy

`> diff(f(x),x);   diff(f(x),x$n);`

Chceme-li s derivací dále pracovat jako s funkcí, je vhodné zadat ji pomocí operátoru `D`:

> D(f)(x);   (D@@n)(f)(x);

Limita funkce:

> limit(f(x),x=4);   limit(f(x),x=infinity);   limit(f(x),x=4,right);

5. Řešení rovnice

Výsledek příkazu pro řešení kvadratické rovnice $x^2 - 4x - 5 = 0$

> S:=solve(x^2-4*x-5,{x});

dostaneme ve tvaru `S := {x = 5}, {x = -1}`. Pro další práci s řešením je výhodné změnit uvedené rovnosti na přiřazovací příkazy, aby se `x` stalo proměnnou, v níž je uloženo řešení. To provedeme příkazem `assign(S[1]);` (pro druhý kořen je příkaz analogický, akorát `S[1]` změníme na `S[2]`)
Příkaz `solve` řeší rovnici v oboru komplexních čísel. Pokud nás zajímají **jenom reálné kořeny**, můžeme použít alternativní příkaz `RealDomain[solve]`. Ze zápisu vidíme, že je součástí knihovny `RealDomain`.

6. Řešení soustavy lineárních rovnic

ÚKOL: Řešte soustavu lineárních rovnic: $x + y + 2z = 1,\ 3x - y - z = -4,\ 2x + 3y - z = -6$

Přímé řešení:

> solve({x+y+2*z=1,3*x-y-z=-4,2*x+3*y-z=-6},{x,y,z});

Ověření řešitelnosti (Frobeniova podmínka):
Rozšířenou matici `Aroz`, matici soustavy `A` i vektor pravých stran `b` vygenerujeme příkazy:

> Aroz:=linalg[genmatrix]({x+y+2*z=1,3*x-y-z=-4,2*x+3*y-z=-6},[x,y,z],flag);
> A:=linalg[genmatrix]({x+y+2*z=1,3*x-y-z=-4,2*x+3*y-z=-6},[x,y,z],b);

Poznámka: Opakem příkazu `genmatrix` je `geneqns`.
Hodnost matice `A` zjistíme příkazem `linalg[rank](A);`
Gaussovu eliminaci provedeme příkazem `linalg[gausselim](A);`, **Gauss-Jordanovu eliminaci** příkazem `linalg[gaussjord](A);`. Eliminaci můžeme provádět i krok za krokem, například užitím příkazu „`pivot`". Lineární soustavu můžeme řešit i užitím speciálního příkazu „`linsolve`" z knihovny `linalg`. Pro získání přehledu o všech možnostech **doporučuji prostudovat nápovědu**, konkrétně ke knihovnám `linalg` a `LinearAlgebra`.

Řešení regulární soustavy užitím inverzní matice:
Řešení soustavy $AX = b$ je rovno součinu $X = A^{-1}b$.
Inverzní matice k matici `A` získáme příkazem `inverse(A);`. Součin matic A^{-1}, b můžeme zadat dvojím způsobem:

> evalm(inverse(A)&*b);   nebo   linalg[multiply](inverse(A),b);

Cramerovo pravidlo
Determinant matice A určíme příkazem `linalg[det](A);`
Matice A_1, A_2, A_3 vytvoříme pomocí příkazů `submatrix`, `augment` knihovny `linalg`.
Poznámka: Pokud používáme více příkazů z nějaké knihovny, vyplatí se jí otevřít příkazem „`with`", v našem případě `with(linalg);`.
Potom bude výpočet matice A_2 vypadat takto:

> A2:=augment(submatrix(A,1..3,[1]),b,submatrix(A,1..3,[3]));

Zadání matice
Matici M typu $(2,3)$ zadáme jedním z příkazů:

> M:=linalg[matrix](2,3,[1,2,3,4,5,6]);
> M:=linalg[matrix]([[1,2,3],[4,5,6]]);

Doporučení: Součástí programu Maple verze 9.5 je balíček (spíše balík) funkcí `Student`, který je určen speciálně pro účely výuky kalkulu a lineární algebry v základních kurzech na vysokých školách. Obsahuje i materiály určené k samostudiu.