Kapitola 3

Užití Maple při řešení kvadrik

3.1 Základy práce s programem Maple

Počítačový algebraický systém Maple^{TM} je produktem kanadské společnosti Maplesoft, Waterloo Maple Inc. Jedná se o špičkový produkt v kategorii programů CAS (Computer Algebra System), který je téměř každý rok aktualizován. V době vydání této knihy byla na trh uvedena verze Maple 14. Uživatel má k dispozici několik typů rozhraní pro komunikaci s programem a pro využívání jeho výpočetních prostředků. Dva základní módy práce s programem nabízejí uživatelská prostředí *Standard worksheet* a *Classic worksheet*. První z nich, *Standard worksheet* (generuje soubory s příponou `mw`), představuje plně grafické uživatelské rozhraní, v jehož pojetí se promítají nejnovější trendy ve vývoji komunikačních rozhraní mezi uživatelem a počítačovým programem. Konkrétně se jedná o snahu zprostředkovat uživateli možnost intuitivního ovládání programu, bez nutnosti znát syntax jeho příkazů. „Konzervativním“ protipólem tohoto grafického uživatelského rozhraní je prostředí *Classic worksheet* (generuje soubory s příponou `mws`). To zachovává design charakteristický pro Maple již od jeho verze 5, zároveň však plně využívá nejnovější výpočetní jádro systému spolu se všemi jeho funkcemi. Zřejmou výhodou prostředí *Classic worksheet* jsou nižší nároky na velikost paměti počítače a rychlost jeho procesoru. Pro běžného uživatele, který nemá důvod pro investování do každoročního upgrade, je ještě významnějším plusem tohoto prostředí nadčasovost jeho vzhledu, stejně jako způsobu práce. Příklady v něm vytvořené jsou tak ve většině případů přenositelné mezi různými verzemi programu. Z tohoto důvodu jsou všechny řešené příklady i ukázky řešení dílčích problémů v této knize realizovány v prostředí *Classic worksheet* Maple 13. Příslušné kódy, uložené v souborech `mws`, jsou k dispozici na CD spolu s touto knihou.
Zevrubné informace o programu Maple, příklady jeho použití, instruktážní videa apod, stejně jako popisy dalších produktů firmy Maplesoft, najde zájemce na webové stránce `http://www.maplesoft.com`.

1. Zadávání příkazů

Na jednom řádku může být uvedeno více příkazů, každý z nich však musí být ukončen středníkem (;) nebo dvojtečkou (:) a řádek potvrzen klávesou `Enter` (bez ohledu na to, kde je v něm kurzor). Například 2 a 3 sečteme příkazem „`2+3;`". Příkaz ukončený středníkem se vykoná a jeho výsledek se zobrazí na následujícím řádku, příkaz ukončený dvojtečkou se rovněž vykoná, výsledek se však nezobrazí.
Nový výpočet s proměnnými je vhodné zahájit příkazem „`restart;`", který vymaže hodnoty všech proměnných. Vyhneme se tak případným komplikacím se starými hodnotami při opakování výpočtu. Pokud potřebujeme vymazat obsah konkrétní proměnné, například proměnné `a`, použijeme příkaz „`a:='a';`".
Nápovědu ke konkrétnímu příkazu Maple vyvoláme zadáním příkazu ve tvaru `?jméno` (zde nemusíme ukončit středníkem, stačí `Enter`). Například, zadáním „`?plot`" získáme kompletní nápovědu k příkazu `plot` i s odkazy na příbuzná témata. Velkým zdrojem informací a inspirace jsou příklady konkrétního použití, které jsou součástí nápovědy ke každému příkazu. Příklady je možno pomocí `Ctrl+C, Ctrl+V` kopírovat do pracovního okna programu Maple, tam je vyzkoušet a následně třeba modifikovat pro potřeby našich výpočtů.

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. Parametr `počet cifer` je nepovinný. Vyzkoušejte „`evalf(sqrt(2));`".

3. Balíčky příkazů

Velká část příkazů programu Maple je uložena v tzv. balíčcích (packages). Například příkazy pro počítání s vektory a maticemi jsou uloženy v balíčcích `linalg` a `LinearAlgebra`. Při zobrazování křivek a ploch pak využíváme příkazy z balíčků `plots` a `plottools`. Jsou dvě možnosti, jak se k takovým příkazům dostat.
1) Načíst do paměti celý balíček příkazem „`with`". Například balíček `linalg` načteme příkazem „`with(linalg);`". Ukončíme-li příkaz středníkem, je vypsán seznam všech příkazů z balíčku. Pokud o takový přehled nestojíme, ukončíme příkaz dvojtečkou.
2) Aniž bychom balíček otvírali, můžeme konkrétní příkaz v něm obsažený zavolat příkazem ve tvaru „`jméno balíčku[jméno funkce](parametry funkce);`". Viz například volání příkazu `linalg[genmatrix]`, které je uvedeno v partii věnované řešení soustav rovnic na straně 105.

4. Funkce jedné proměnné

Definice funkce. Chceme-li definovat například funkci $f: y = x^2 - 4x$, máme možnost využít těchto dvou příkazů:

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

Základním příkazem pro zobrazení **grafu funkce** je příkaz „`plot(f(x),x);`". Podobu grafu můžeme ovlivnit jeho doplněním o další parametry a volby. Například graf s definovaným rozsahem os zobrazíme příkazem

```
> plot(f(x),x=-5..5,y=-4..4);
```

Všechny volby (options), kterými můžeme modifikovat výsledek příkazu `plot`, zobrazíme zadáním „`?plot,options`". U nespojitých funkcí například oceníme volbu `discont=true`. Porovnejte příkazy:

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

Derivace funkce. První, respektive n-tá derivace funkce (výrazu) $f(x)$ se vypočítá zadáním příkazu

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

Chceme-li s derivací dále pracovat jako s funkcí, je vhodné zadat ji pomocí operátoru `D`. První, respektive n-tá derivace jako funkce proměnné `x` se potom vyjádří příkazem

```
> D(f)(x);   resp.   (D@@n)(f)(x);
```

Neurčitý a určitý integrál. Neurčitý, resp. určitý (s mezemi 2, 4) integrál funkce (výrazu) $f(x)$ vypočítáme příkazem

```
> int(f(x),x);   resp.   int(f(x),x=2..4);
```

Vyzkoušejte příkaz pro výpočet objemu tělesa vzniklého rotací grafu funkce $f(x)$ kolem osy x na intervalu $\langle 0, 5\rangle$ (zobrazení tohoto rotačního tělesa je popsáno ne straně 108):

```
> Int(Pi*f(x)^2,x=0..5)=int(Pi*f(x)^2,x=0..5);
```

Limita funkce. Výpočet limity ve vlastním a nevlastním bodě, stejně jako výpočet jednostranné limity funkce $f(x)$ ilustrují následující příklady:

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

5. Funkce více proměnných

Definice funkce. Opět máme dvě možnosti, jak definovat funkci, např. $g: z = x^2 \sin y$:

```
> g:=(x,y)->x^2*sin(y);   nebo   g:=unapply(x^2*sin(y),x,y);
```

Graf funkce zobrazíme příkazem: „`plot3d(g(x,y),x=-5..5,y=-5..5);`"
Pro zobrazení plochy dané rovnicí $F(x, y, z) = 0$, například $x^2 - y - z^2 = 0$, použijeme příkaz `implicitplot3d` z balíčku `plots`:

```
> plots[implicitplot3d](x^2-y^2-z=0,x=-5..5,y=-5..5,z=-5..5);
```

Zobrazená plocha není příliš „hladká". Vyzkoušejte přidat do výše uvedeného příkazu `implicitplot3d` jako nepovinný parametr volbu `grid=[30,30,30]`.

Více informací o možnostech ovlivnit podobu grafu získáme zadáním „`?plot3d,options`".

Derivace funkce. Parciální derivace funkce $g(x, y)$ podle x, respektive podle y, se určí příkazem

```
> diff(g(x,y),x);   resp.   diff(g(x,y),y);
```

Při použití operátoru `D`, který umožňuje nakládat s derivací jako s funkcí proměnných x, y, pak vypočítáme uvedené parciální derivace takto:

```
> D[1](g)(x,y);   resp.   D[2](g)(x,y);
```

Smíšená parciální derivace $\dfrac{\partial^5 g}{\partial x^2 \partial y^3}$ se potom zadá příkazem

```
> diff(g(x,y),x$2,y$3);   nebo   > D[1$2,2$3](g)(x,y);
```

6. Řešení rovnice

Symbolické řešení. K symbolickému řešení rovnice, např. $x^2 - 4x - 5 = 0$, použijeme příkaz

```
> Res:=solve(x^2-4*x-5,x);   resp.   Res:=solve(x^2-4*x-5,{x});
```

Výsledek potom dostaneme ve tvaru `Res:=5,-1`, resp. `Res:={x=5},{x=-1}`. Na jednotlivé kořeny rovnice se odkazujeme pomocí indexů, které odpovídají jejich pořadí ve výpisu výsledku příkazu „`solve`", tj. příkazy: `Res[1];` a `Res[2];`

Příkaz `solve` řeší rovnici v oboru komplexních čísel. Pokud nás zajímají **jenom reálné kořeny**, můžeme použít alternativní příkaz `RealDomain[solve]` z balíčku `RealDomain`. Pro ilustraci porovnejte výstupy následujících příkazů:

```
> solve(x^3+2*x+1,x);   a   > RealDomain[solve](x^3+2*x+1,x);
```

Při grafickém řešení rovnice oceníme příkazy „`lhs(rov)`" a „`rhs(rov)`" pro „uchopení" levé a pravé strany rovnice `rov`. Například rovnici $x^x = 0.75^{0.75}$ bychom graficky řešili touto posloupností příkazů:

```
> rov:=x^x=0.75^0.75;   plot({lhs(rov),rhs(rov)},x=-2..2);
```

Důležitou součástí řešení rovnice je ověření jeho správnosti dosazením, tj. **zkouška**. Pro postupné dosazení jednotlivých řešení do rovnice `r` můžeme použít příkaz `subs` nebo příkaz `eval`. Podmínkou použití příkazu `eval`, které ilustruje následující příklad, je uzavření neznámé `x` v příkazu `solve` do složených závorek:

```
> r:=x^3-2*x+1=0; solve(r,{x}); eval(r,Res[1]); eval(r,Res[2]);
```

Někdy potřebujeme provést rozklad mnohočlenu na jedné (např. levé) straně rovnice. Použijeme-li příkaz `factor(lhs(r))`, záhy zjistíme, že má své limity (viz `?factor`). Lepší službu vykoná následující série příkazů:

```
> polytools[split](lhs(r),x);
> convert(%,radical);
```

Symbol `%` představuje jméno (systémové) proměnné, v níž je uložen výsledek naposledy vykonaného (tj. potvrzeného klávesou `Enter`) příkazu. Podobně symbol `%%` odkazuje na výsledek předposledního vykonaného příkazu.

Numerické řešení. K numerickému řešení rovnice je určen příkaz `fsolve`. Pokud má rovnice více nulových bodů, je možné v tomto příkazu specifikovat bod, v jehož okolí chceme řešení hledat. Například výše uvedenou rovnici $x^x = 0.75^{0.75}$ bychom, po jejím grafickém řešení, kompletně numericky vyřešili následující posloupností příkazů:

```
> rov:=x^x=0.75^0.75;  fsolve(rov,x=0);  fsolve(rov,x=1);
```

7. Řešení soustavy lineárních rovnic. Operace s maticemi a s vektory.

Řešme následující (regulární) soustavu lineárních rovnic:

$$x + y + 2z = 1,\ 3x - y - z = -4,\ 2x + 3y - z = -6$$

Přímé řešení provedeme příkazem

```
> 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` vytvoříme následujícími 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 příkaz `geneqns`.

Hodnost matice `A` zjistíme příkazem „`linalg[rank](A);`", **Gaussovu eliminaci** provedeme příkazem „`linalg[gausselim](A);`", **Gauss-Jordanovu eliminaci** pak realizujeme 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`. Na tomto místě je třeba uvést, že balíček příkazů `linalg` je již staršího data a jeho obsah není nijak aktualizován. V novějších verzích Maple je sice nadále trpěn (s přívlastkem „deprecated"), avšak jeho funkci přebírá balíček moderněji naprogramovaných příkazů `LinearAlgebra` (viz `?LinearAlgebra`).

Řešení regulární soustavy užitím inverzní matice. Řešení soustavy $AX = b$ můžeme vyjádřit vztahem $X = A^{-1}b$, kde A^{-1} je inverzní matice k matici A. **Inverzní matici** k matici `A` získáme zadáním příkazu „`inverse(A);`". Součin matic A^{-1} a b můžeme provést jedním z následujících dvou příkazů:

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

Cramerovo pravidlo. Matice A_1, A_2, A_3 vytvoříme například pomocí příkazů `submatrix` a `augment` knihovny `linalg`. **Determinant matice** A potom vypočítáme příkazem `linalg[det](A);`

Poznámka: Pokud používáme více příkazů z nějaké knihovny, vyplatí se jí otevřít příkazem „`with`", v našem případě „`with(linalg):`".

Potom bude výpočet, např. pro neznámou x_2, vypadat následovně. Nejprve vytvoříme matici A_2 příkazem

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

Potom vypočítáme hodnotu x_2:

```
> x2:=det(A2)/det(A);
```

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]]);
```

Operace s vektory. Vektory, např. $\vec{u} = (1,-2,3)$ a $\vec{v} = (0,-5,4)$, zadáme příkazy „`u:=[1,-2,3];`" a „`v:=[0,-5,4];`"
Skalární, respektive vektorový součin vektorů $\vec{u}$, $\vec{v}$ provedeme následující aplikací příkazu `dotprod`, resp. `crossprod`:

```
> linalg[dotprod](u,v,'orthogonal');
> linalg[crossprod](u,v);
```

Normu (eukleidovskou) vektoru $\vec{u}$ spočítáme příkazem „`linalg[norm](u,2);`". Pozor na záměnu s výpočtem absolutní hodnoty výrazu x. Ta se určí příkazem „`abs(x);`". Úhel vektorů $\vec{u}$, $\vec{v}$ spočítáme následující posloupností příkazů. První z nich otevře balíček příkazů `linalg`, druhý uvede velikost úhlu α v radiánech a pomocí třetího příkazu potom tento údaj převedeme na velikost úhlu ve stupních:

```
> with(linalg):
> alpha:=arccos(dotprod(u,v)/(norm(u,2)*norm(v,2)));
> evalf(convert(alpha,degrees));
```

3.2 3D grafy v Maple

V této příloze se budeme nejprve podrobně věnovat možnostem znázornění trojrozměrných křivek a ploch pomocí prostředků rozhraní *Classic worksheet* programu Maple. V závěru přílohy potom zmíníme dvě konkrétní aplikace programu Maple, které dovolují uživatelsky nenáročným způsobem, bez znalosti jakéhokoliv příkazu, konstruovat grafické znázornění ploch a provádět analýzu ploch druhého stupně. Jedná se o takzvané *maplety*. První z těchto mapletů, *Interactive Plot Builder*, je asistentem pro kreslení 3D grafů dodávaným spolu s programem. Takovýchto asistentů má uživatel programu Maple k dispozici více, pouze však v prostředí *Standard worksheet* (pro více informací stačí zadat „`?assistants`"). Druhý z představených mapletů je příkladem uživatelsky naprogramované aplikace. Byl vytvořen v roce 2008 studentem Padagogické fakulty JU Markem Dvorožňákem. Pojetí této aplikace přesně odpovídá jejímu účelu, kterým je provedení kompletní analýzy zadané křivky spolu s jejím grafickým znázorněním.

1. Křivka
Křivka zvaná **šroubovice** je dána parametrickými rovnicemi $x = r\cos\omega,\ y = r\sin\omega,\ z = v_0\omega,$ kde r je poloměr příslušného otáčení, ω je úhel tohoto otáčení