Masarykova univerzita
Fakulta přírodovědecká

Bakalářská práce

Systém počítačové algebry YACAS

Havránek Aleš

Vedoucí práce: RNDr. Roman Plch, PhD.

Studijní program: Matematika-ekonomie, bakalářský

Obor: Aplikovaná matematika

květen 2007

Poděkování

Chtěl bych poděkovat RNDr. Romanovi Plchovi, PhD. za vedení bakalářské práce, cenné rady a připomínky při zpracování daného tématu.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval sám, pouze za pomoci RNDr. Romana Plcha, PhD. a uvedené literatury.

V Brně dne 23. května 2007 Aleš Havránek

Obsah

Úvod

Pod pojmem „systém počítačové algebry“(*Computer algebra system*, *CAS*) chápeme počítačový program, který pracuje s matematickými výrazy v symbolické podobě (narozdíl od programů, které umí zpracovávat pouze číselné hodnoty). Tyto programy nám umožňují provádět rozmanité symbolické výpočty, například zjednodušování, expanzi a vyhodnocování algebraických výrazů, výpočet derivací a integrálů, operace s vektory a maticemi atd. Jejich vývoj začal v 60. letech 20. století především za účelem provádění složitých fyzikálních výpočtů. CAS se postupně začaly rozšiřovat i do ostatních oblastí vědy a výzkumu (technika, biologie, ekonomie), jako i do vysokoškolského studia – zde uveďme kurzy ovládání CAS Maple na Přírodovědecké fakultě Masarykovy univerzity.

V zahraničí se dokonce začaly objevovat nové přístupy, kdy je kladen důraz na využívání počítačů ve výuce. „V USA v rámci projektu CALC[1] (Calculus As a Laboratory Course) byla klasická cvičení zrušena úplně, výpočetní operace a metody jsou procvičovány v rámci počítačové výuky.“[2] Použití počítače jako pomůcky při studiu může být velmi užitečné. Žák si může jednak daný problém nechat graficky znázornit, ale také může využít schopnosti CAS příklad vypočítat, a tak se může zbavit dlouhého rutinního postupu, případně ověřit správnost jím dosaženého výsledku. Problémem se ovšem může stát to, že žák zadává příkazy bezmyšlenkovitě bez vazby na teorii, což může vést ke chybnému výsledku nebo jeho nesprávné interpretaci.

Ve své práci se budu zabývat systémem počítačové algebry YACAS. Jejím cílem je popsat prostředí a základní funkce tohoto programu, s důrazem na jeho využití při výuce základních kurzů matematické analýzy.

[1] http://www.math.duke.edu/education/proj_calc/index.html

[2] DOŠLÁ, Z., PLCH, R., SOJKA, P. Diferenciální počet funkcí více proměnných s programem Maple V. Brno: MU 1999, s. 5.

Kapitola 1

Základní informace o YACASu

1.1 Představení

YACAS[1] je akronym pro Yet Another Computer Algebra System (volně přeloženo jako „Ještě Další Systém Počítačové Algebry“, což naráží na dnes velké množství podobně zaměřených programů). Distribuce tohoto softwaru je založena na GNU GPL (GNU General Public Licence neboli „Všeobecná veřejná licence GNU“), což znamená, že může být volně rozšiřován a jeho zdrojové kódy mohou být svobodně upravovány a používány (odvozená díla však musí být též šířena pod touto licencí). Jeho vývoj započal v roce 1999 Ayal Pinkus (který je doposud hlavním autorem) a zatím poslední známá verze (k 1. polovině roku 2007) je 1.0.63 ze 7. 1. 2007. YACAS se skládá z malého jádra a knihovny skriptů napsaných ve vlastním jazyce[2] YACASu; v těchto skriptech jsou vlastně ukryty veškeré funkce YACASu jako CAS. Autoři tvrdí, že vývoj jazyku YACASu je v podstatě ukončen, takže jakékoliv programy v něm napsané by měly být použitelné i v jeho dalších verzích.

1.2 Kompatibilita

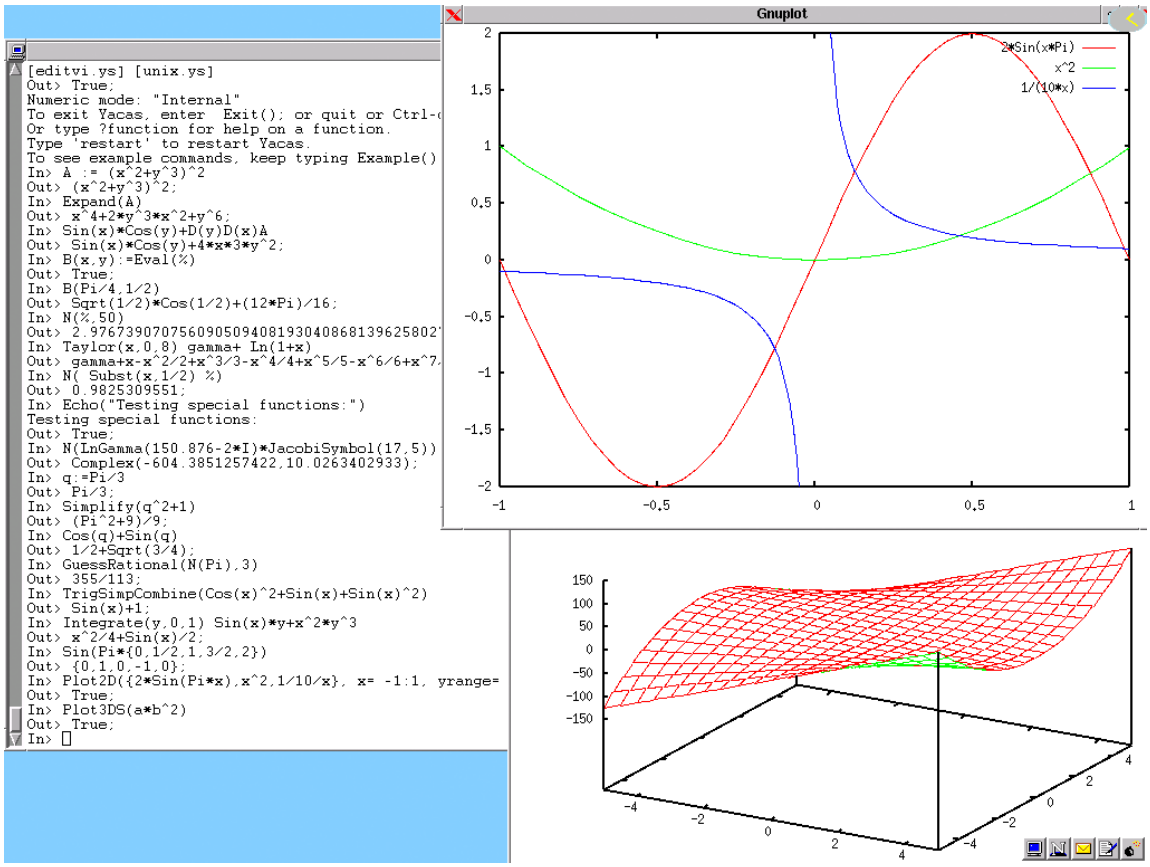
Vzhledem k volně dostupnému zdrojovému kódu (v programovacím jazyku C++) a textovému režimu není problém provozovat jej na prakticky libovolné platformě (Unix/Linux, MAC OS X, 32-bitové Microsoft Windows), která obsahuje překladač C++. Autoři sami podotýkají, že nejvíce funkční je verze pro Linux, nicméně není problém si z internetu[3] stáhnout verzi pro Windows, u které se ale pravděpodobně budete potýkat s některými problémy, jež by ale neměly omezit základní funkce YACASu. Zajímavostí je, že YACAS existuje i ve formě java appletu, tedy může být používán na jakémkoliv počítači s přístupem na internet a internetovým

[1] http://yacas.sourceforge.net/

[2] ten autoři označují jako easy-to-use, tedy snadno použitelný

[3] http://www.xs4all.nl/~apinkus/backups/

prohlížečem podporujícím Javu, a to na adrese http://yacas.sourceforge.net/yacasconsole.html. Pro YACAS existuje i grafické prostředí pro Windows - GUYACAS.[4]



Obrázek 1.1: Prostředí YACASu

1.3 Stažení a instalace

Na webových stránkách[5] si vyberte verzi podle vašeho operačního systému. Tzn. pro Linux si stáhneme archiv,[6] pro Windows zkompilovaný balíček.[7]

Instalace v Linuxu: Po stažení výše uvedeného souboru spustíme příkaz `tar zxvf yacas-latest.tgz` a přejdeme do vytvořeného adresáře `yacas-<verze>`. Napíšeme `./configure` s volitelnými parametry.[8] Dále napíšeme `make` a poté `make`

[4] http://genetics.agrsci.dk/~sorenh/misc/GUYacas/index.html
[5] http://yacas.sourceforge.net/downloads.html
[6] http://yacas.sourceforge.net/backups/yacas-1.0.63.tar.gz
[7] http://www.xs4all.nl/~apinkus/backups/Yacas-Windows-Console-1.0.61.zip
[8] např. `./configure --with-numlib=gmp --enable-ps-doc --prefix=/home/user/`

`install` (tento příkaz vyžaduje administrátorská práva, pokud nemáte nastaven parametr –prefix pro váš domovský adresář).

Instalace ve Windows: Stažený archív .zip rozbalíme do námi zvoleného adresáře (např. pomocí programu WinZIP), není třeba nic instalovat.

GUYACAS stáhneme z odkazu `http://gbi.agrsci.dk/%7Eshd/Misc/GUYacas/download/GUYacas.exe`, soubor spustíme a provedeme instalaci. Nutno poznamenat, že je potřeba mít nainstalovaný software „Microsoft .NET Framework Version 2.0“.[9]

1.4 Spuštění

V Linuxu spustíme YACAS napsáním příkazu `yacas` v konzoli a ve Windows spuštěním souboru `yacas.exe` z adresáře, kam jsme stažený archiv rozbalili. Při spouštění můžeme k názvu souboru připsat několik parametrů, těmi se ale nebudeme zabývat (jedná se např. o spouštění souborů, designu prostředí atd.), bližší informace se dají nalézt v manuálu [2].

1.5 Nápověda

Veškerá dokumentace je přístupná z promptu YACASu. Napíšete-li `??`, budete mít možnost číst veškeré dostupné manuály. Dokumentace se zobrazí v internetovém prohlížečí dle vaší volby. Standardně je nastaven lynx. Jiný prohlížeč může být zvolen příkazem `SetHelpBrowser`.

Pokud chceme využít nápovědy k dané funkci, použijeme příkaz `?`. Jako parametr použijeme název funkce, jejíž detaily chceme znát. Ukáže se příslušná sekce příručky. Názvy funkcí by měly být zadány bez uvozovek. Všimněme si, že u názvů funkcí v YACASu se rozlišují velká a malá písmena, ale příkaz `?` reaguje také na jména napsaná malými písmeny. Další možnost je stáhnout si manuály z domovské stránky.

1.6 Vypnutí a restart

Zadání příkazu `quit` v promptu YACASu způsobí jeho vypnutí, příkaz `restart` YACAS restartuje. Jejich efekt se projeví pouze tehdy, když jsou napsána ihned na začátku promptu (ovšem mohou před nimi být mezery). Zmáčnutí `Ctrl-C` zastaví právě probíhající operaci; jestliže žádná neprobíhá, ukončí tato zkratka YACAS. Během ukončení se zaznamenává historie příkazů. Starší příkazy vyvoláme šipku nahoru, můžeme v nich listovat šipkami nahoru nebo dolů. Pokud chceme jako argument nějakého příkazu využít výsledek z předchozího řádku, napíšeme místo argumentu znak `%`.

[9] `http://www.stahuj.cz/vyvojove_nastroje/ostatni/net-framework/`

1.7 Forma výstupu

Nelíbí-li se nám standardní forma výstupu, která může být v případě delších či složitějších výrazů velmi nepřehledná, můžeme použít jinou formu výstupu. K tomuto slouží příkaz `PrettyForm(výraz)` - ten vypíše výraz estetičtěji než standardní výpis, a to pomocí ASCII artu.[10] Dále máme k dispozici příkaz `EvalFormula`, který daný výraz vyhodnotí a následně vypíše ve stejném tvaru jako `PrettyForm`.

Pokud chceme, aby se nám všechny výsledky zobrazovaly jinak než standardně, použijeme příkaz `PrettyPrinter(printer)`, kde printer je řetězec (viz níže), který obsahuje jméno funkce pro tisk výrazů. Nás zajímá především funkce `PrettyForm`, dále můžeme použít výstup v TeXovské formě prostřednictvím `TeXForm` a původní formu získáme zadáním funkce `DefaultPrint`. Nutno poznamenat, že název vykreslovací funkce musí být v uvozovkách, jak vyplývá z definice řetězce v YACASu. Pokud nás zajímá aktuálně používaná forma výpisu, zadáme příkaz `GetPrettyPrinter`.

1.8 Import a export dat

YACAS obsahuje několik procedur pro import a export dat, nicméně se jedná o velice jednoduché příkazy, které samy o sobě nestačí např. pro komplexnější zálohu vypracovaných dat. Příkaz `Load("název_souboru")` načte kompletní obsah daného souboru – ten se nezobrazí, pouze se načtou a provedou příkazy a proměnné se zapíšou do paměti. Je třeba dbát na to, aby za příkazy byly středníky, jinak YACAS nebude schopen příkazy ze souboru spustit. Pro zápis daného příkazu použijeme `ToFile("název_souboru") WriteString(příkaz)`. Zde je potřeba dávat pozor, jelikož pokud budeme zapisovat do již existujícího souboru, jeho obsah bude kompletně přepsán.

Pokud nás detailněji zajímá, jak určitá procedura pracuje, nastavíme globální proměnnou `Verbose` (což se dá přeložit jako „ukecaný “) na `True` a příkazy nás budou informovat o probíhajících procesech. Do původního stavu se dostaneme nastavením `Verbose:=False`.

[10] Jedná se o vykreslování obrázků a tvarů pomocí písmen a jiných znaků. Více viz http://en.wikipedia.org/wiki/ASCII_art.)

Kapitola 2

Prostředí YACASu

2.1 Základní výpočty

YACAS můžeme použít jako kalkulačku. Máme k dispozici tyto základní operátory: `+`, `-`, `*`, `/`, `^` neboli sčítání, odečítání, násobení, dělení a umocňování.[1]

Z často používaných elementárních funkcí jsou podporovány např. trigonometrické funkce `Sin`, `Cos`, `Tan` a jejich inverzní funkce `ArcSin`, `ArcCos`, `ArcTan`, `Sqrt` (druhá odmocnina), `Exp` (exponenciální funkce), `Ln` (přirozený logaritmus), `Abs` (absolutní hodnota), `Sign` (znaménko čísla). Do kulatých závorek za danou funkci umístíme její argument. K číslu π můžeme přistupovat přes konstantu `Pi`.

Z operací s přirozenými čísly je podporován například faktoriál, a to operátorem `!` napsaným přímo za číslo, dále výpočet hodnoty kombinačního čísla $\binom{n}{m}$ příkazem `Bin(n,m)`.

Nyní si můžeme předvést jednoduchý příklad:

```
In> 2*Sin(Pi/2)/2^2+1
Out> 3/2
```

Pokud nechceme jako výsledek zlomek, ale desetinné číslo, použijeme funkci `N()`. Ta slouží k vypsání přibližné hodnoty daného výrazu (např. zlomku), jinak zůstane daný zlomek nevyčíslený. Funkce `N()` má druhý nepovinný argument - požadovanou přesnost (počet desetinných míst). Přesnost pro všechny výpočty můžeme nastavit příkazem `Precision(n)`, kde n značí počet číslic, se kterými bude YACAS počítat a které budou následně zobrazeny. Nutno podotknout, že výsledky YACAS nezaokrouhluje, tedy výstup `N(2/3)` bude 0.666 při `Precision(3)`.

Berme na vědomí, že priorita základních operátorů je v YACASu stanovena následovně (seřazeno od nejnižší po nejvyšší): sčítání, odečítání, násobení, dělení,

[1] Je zřejmé, že libovolnou odmocninu z daného čísla vypočítáme jako jeho mocninu na převrácenou hodnotu exponentu.

umocňování. Z toho např. vyplývá, že výraz `16^1/2` YACAS vyhodnotí jako 8, tedy jako `(16^1)/2`. Prioritu operátorů můžeme řídit kulatými závorkami.

YACAS umí také pracovat s komplexními čísly. Komplexní číslo ve tvaru $a+bi$ si zadefinujeme procedurou `Complex(a,b)`, kde `a` je jeho reálná část a `b` je jeho imaginární část. V případě, že chceme vypsat reálnou část nějakého komplexního čísla, použijeme funkci `Re`, pro vypsání imaginární části slouží funkce `Im`. V YACASu je definovaná také imaginární jednotka i, a to pod konstantou `I`. Pro výpočet komplexně sdruženého čísla slouží procedura `Conjugate` a argumentu procedura `Arg`.

2.2 Proměnné

K přiřazení hodnot globálním proměnným se používají funkce `Set()` a operátor `:=`. Pro vymazání spojení k příslušné proměnné a použijte funkci `Clear(a)`. Hodnota proměnné `a` se bude nyní rovnat a. Toto je jedna z vlastností výpočetního modelu YACASu; jestliže nějaký objekt nemůže být vyhodnocen nebo jakkoliv dále transformován, je sám vrácen jako konečný výsledek. V současné verzi YACASu nejsou velké rozdíly mezi přiřazováním proměnných použitím `Set()` nebo operátoru `:=`, nicméně druhý z nich může také přiřazovat seznamy a definovat funkce.

```
In> Set(a,cos(0));
Out> True
In> a:=a+1;
Out> 2
```

2.3 Řetězce a seznamy

Navíc k číslům a proměnným podporuje YACAS řetězce a seznamy. Řetězce jsou jednoduché posloupnosti znaků uzavřené dvojitými uvozovkami.

```
In> a:="Toto je řetězec s \"uvozovkami\" uvnitř sebe"
Out> "Toto je řetězec s "uvozovkami" uvnitř sebe"
```

Obsah vybraného řetězce nebo i jiné struktury YACASu vypíšeme příkazem Echo.

```
In> Echo(a);
Toto je řetězec s "uvozovkami" uvnitř sebe
Out> True
```

Seznamy jsou uspořádané skupiny položek. YACAS znázorňuje seznamy vložením položek mezi závorky a jejich oddělením čárkami. Seznam objektů a, b, a c může být vložen napsáním `{a,b,c}`. V YACASu jsou vektory znázorněny jako seznamy a

matice jako seznamy seznamů. Jakýkoliv výraz YACASu může být převeden na seznam. K položkám seznamu můžeme přistupovat pomocí `[ ]` operátoru nebo pomocí příkazu `Nth`, kde první argument je vybraný seznam a druhý argument je pořadí jeho položky, kterou chceme zobrazit.

```
In> uu:={a,b,c,d,e,f};
Out> {a,b,c,d,e,f}
In> uu[2];
Out> b
In> uu[2 .. 4];
Out> {b,c,d}
```

Rozsah `2 .. 4` se vyhodnotí jako {`2,3,4`}. Všimněte si, že mezery okolo `..` operátoru jsou nutné, jinak si jej parser (syntaktický analyzátor) bude plést s částí čísla. Seznamy vyhodnocují své argumenty, tedy {`1+2,3`} se vyhodnotí jako {`3,3`}.

Se seznamy čísel můžeme provádět následující operace: `Add` sečte všechny prvky seznamu a vrátí toto číslo, `Average` vypočítá průměr prvků seznamu a `Max` (`Min`) vrátí největší (nejmenší) prvek seznamu.

Přiřazení více proměnných je také možné pomocí seznamů, např. výsledkem {`x,y`}:={`2!,3!`} bude to, že x se přiřadí 2 a y se přiřadí 6. Délku seznamu (respektive počet prvků seznamu) zjistíme pomocí příkazu `Length`, převrácený seznam (tedy seznam seřazený postupně od posledního po první prvek) získáme příkazem `Reverse`. Obsah vybraných seznamů sloučíme do jednoho příkazem `Concat`. YACAS obsahuje velkou řádku dalších operací se seznamy, nalezneme je v příručce.

```
In> a:=1 .. 5;
Out> {1,2,3,4,5};
In> Length(a);
Out> 5;
In> a:=Reverse(a);
Out> {5,4,3,2,1};
In> b:={0};
Out> {0};
In> c:=Concat(a,b);
Out> {5,4,3,2,1,0};
```

Kapitola 3

Lineární algebra

3.1 Vektory

Vektor o daných rozměrech je reprezentován jako seznam jeho prvků. Formálně se tedy zapíše jako `v:={prvek_1,prvek_2,...,prvek_n}`. Pokud nás zajímá hodnota i-tého prvku daného vektoru v, použijeme příkaz `v[i]`. Dále můžeme přiřadit i-tému prvku daného vektoru jinou hodnotu h příkazem `v[i]:=h`. Vektor můžeme násobit skalárem. Vektory o stejných rozměrech můžeme sčítat. Skalárně násobit můžeme vektory o stejné délce operátorem `.` (pro upřesnění – tím je míněn znak tečka).

YACAS umí dále generovat nulové vektory, a to příkazem `ZeroVector` (parametrem je délka vektoru neboli počet prvků vektoru), vektory kanonické (standardní) báze, a to příkazem `BaseVector` (první parametr je index bazického vektoru,[1] druhý parametr je délka vektoru). Vektor můžeme normalizovat (upravit tak, aby jeho velikost byla 1) příkazem `Normalize`.

```
In> v:=ZeroVector(3);
Out> True;
In> v;
Out> {0,0,0};
In> v[ 2 ]:=2;
Out> True;
In> v;
Out> {0,2,0};
```

YACAS v sobě obsahuje i Gramm-Schmidtův algoritmus pro ortogonalizaci a ortonormalizaci bází. Toho využijeme, máme-li bázi generující určitý vektorový prostor a chceme najít bázi, která bude generovat stejný vektorový prostor, ale její vektory na sebe budou kolmé (tedy skalární součin vektorů báze bude roven 0) – taková báze se nazývá ortogonální. Je-li navíc velikost všech vektorů báze 1, budeme mluvit

[1] Určuje tedy, na kterém místě vektoru bude jednička.

o ortonormální bázi. K tomu slouží funkce `OrthogonalBasis` a `OrthonormalBasis`. Jejich argumenty budou vektory báze.[2]

3.2 Matice

Matice jsou reprezentovány jako vektor vektorů. Jednotkovou matici rozměru 3×3 tedy zapíšeme jako `{{1,0,0},{0,1,0},{0,0,1}}`. Matice vhodných rozměrů můžeme sčítat, odečítat, násobit. Pokud se na matice stejných rozměrů pokusíme aplikovat operátor `/`, nebude výsledkem násobení první matice inverzí druhé (jako např. v Matlabu), ale dělení po prvcích (tedy $A/B = (a_{ij}/b_{ij})$.

Jednotkovou matici o rozměru $n \times n$ můžeme jednoduše vytvořit příkazem `Identity(n)`, pro nulovou matici můžeme použít příkaz `ZeroMatrix(n,m)`, kde n je počet řádků a m počet sloupců (Pokud vynecháme druhý argument, dostaneme nulovou čtvercovou matici rozměru m.). Chceme-li vytvořit diagonální matici (tedy matici, která obsahuje samé nuly až na prvky na hlavní diagonále), použijeme příkaz `DiagonalMatrix`, kde argumentem bude seznam prvků, které chceme umístit na hlavní diagonálu. Pokud chceme s hlavní diagonálou matice pracovat jako s vektorem, zadáme příkaz `Diagonal`, jehož argumentem bude vybraný vektor. Stopu matice (neboli sumu prvků na hlavní diagonále) získáme příkazem `Trace`. Náhodnou matici, jejíž prvky jsou celá čísla, vygenerujeme příkazem `RandomInteger`, kde první argument je počet řádků, druhý argument počet sloupců a třetí a čtvrtý argument udává dolní a horní mez podmnožiny celých čísel, odkud se budou prvky matice vybírat.

Matici transponujeme příkazem `Transpose`. Příkazem `Determinant` zjistíme determinant čtvercové matice. K dané regulární matici můžeme vypočítat inverzní matici příkazem `Inverse`. Minor (tedy determinant vybrané submatice dané matice vzniklé vypuštěním i-tého řádku a j-tého sloupce) dané čtvercové matice zjistíme po zadání příkazu `Minor(A,i,j)`.

Pro řešení systému lineárních rovnic využijeme příkazu `SolveMatrix(A,b)`, kde `A` je matice koeficientů levých stran rovnic a `b` je vektor pravých stran rovnic. Determinant matice A musí být nenulový, tedy matice A musí být regulární.

Pokud jde o výpočet charakteristického polynomu vhodné matice, použijeme příkaz `CharacteristicEquation(M,k)`, kde `M` je vybraná matice a `k` je zvolená proměnná charakteristického polynomu (běžně se používá λ). Vlastní čísla dostaneme jako výstup funkce `EigenValues`, vlastní vektory jako výstup funkce `EigenVectors` (u ní bude první argument zvolená matice a druhý argument bude výstup funkce `EigenValues`).

[2] Vlastně půjde o matici (viz níže), v jejichž řádcích budou vektory báze.

YACAS obsahuje i funkce pro testování některých vlastností matic. Tyto nabývají hodnot buďto `True` nebo `False`. Některé můžeme vidět v následující tabulce.

Příkaz	**Popis**
`IsMatrix`	Je daný objekt matice?
`IsSquareMatrix`	Je daný objekt čtvercová matice?
`IsHermitian`	Je daná čtvercová matice hermitovská (samoadjungovaná)? Tedy platí, že daná matice se rovná její transpozici tvořené komplexně sdruženými čísly?
`IsOrthogonal`	Je daná čtvercová matice ortogonální? Tedy platí, že součin dané matice a její transpozice je roven jednotkové matici neboli $A^{-1} = A^T$?
`IsDiagonal`	Je daná čtvercová matice diagonální? Tedy platí, že kromě hlavní diagonály matice obsahuje samé nuly?
`IsLowerTriangular`	Je daná čtvercová matice dolní trojúhelníková? Tedy platí, že všechny její prvky nad hlavní diagonálou jsou rovny nule?
`IsUpperTriangular`	Je daná čtvercová matice horní trojúhelníková? Tedy platí, že všechny její prvky pod hlavní diagonálou jsou rovny nule?
`IsSymmetric`	Je daná čtvercová matice symetrická, tedy platí $A = A^T$?
`IsSkewSymmetric`	Je daná čtvercová matice antisymetrická, tedy platí $A = -A^T$?
`IsUnitary`	Je daná čtvercová matice unitární? Tedy platí, že součin dané matice a její transpozice tvořené komplexně sdruženými čísly je roven jednotkové matici?
`IsIdempotent`	Je daná čtvercová matice idempotentní? Tedy platí, že $A^n = A$, $n \in N$?

Kapitola 4

Funkce

4.1 Definování funkcí

K definování funkcí je určen operátor `:=`. Např. `f(x):=2*x*x` definuje novou funkci f, která přijme jeden argument a vrátí dvakrát druhou mocninu onoho argumentu. Jedno a též jméno funkce (např. f) může být používáno různými funkcemi, pokud ty používají rozdílný počet argumentů.

```
In> f(x):=x^2;
Out> True;
In> f(x,y):=x*y;
Out> True.
In> f(3)+f(3,2);
Out> 15;
```

V YACASu jsou předdefinovány proměnné `True` a `False` jako Booleovy hodnoty. Funkce vracející Booleovské hodnoty se nazývají predikáty. Např. `IsNumber()` a `IsInteger()` jsou predikáty definované ve standardní knihovně.

```
In> IsNumber(2+x);
Out> False;
In> IsInteger(15/5);
Out> True;
```

Při přiřazování hodnot proměnným je pravá strana vyhodnocena před přiřazením. Tedy `a:=2*2` přiřadí proměnné a číslo 4. Toto však neplatí pro funkce. Když zadáme `f(x):=x+x`, pravá strana před přiřazením není vyhodnocena. Vyhodnocení může být vynuceno použitím příkazu `Eval()`. Příkaz `f(x):=Eval(x+x)` nejdříve vyhodnotí $x + x$ jako $2x$ a pak přiřadí funkci f výraz $2x$. Tento specifický příklad není zrovna velmi užitečný, ale příkaz se hodí, když je operace na pravé straně příliš složitá. Např. vyhodnotíme-li rozvoj Taylorovy řady před jejím přiřazením nějaké

funkci, jádro YACASu nemusí počítat rozvoj Taylorovy řady pokaždé, když je daná funkce zavolána.

4.2 Polynomy

YACAS obsahuje některé funkce pro práci s polynomy. Pokud chceme polynom ve zjednodušené formě (např. $(1+x)^5$) převést na rozšířený tvar (tím je myšlena funkce ve tvaru $P(x) = a_n x^n + \cdots + a_0$), můžeme použít příkaz `Expand`. Pokud námi zadaný polynom obsahuje parametry (např. $(ax+b)^5$), zadáme jako druhý argument název proměnné, podle které chceme daný polynom rozvíjet (v našem případě x). Jestliže chceme převést na rozšířený polynom s více proměnnými, zadáme jako druhý argument vektor s názvy proměnných, přičemž polynom je rozšířen podle první jmenované proměnné, dále jsou všechny koeficienty první proměnné (tedy vlastně polynomy) rozšířeny podle druhé proměnné atd. Další možnost, jak rozšířit daný výraz je použitím příkazu `ExpandBrackets`, který jde ještě dále než `Expand`, protože se zbaví všech závorek.

```
In> ExpandBrackets((a*x+b)^2);
Out> a^2*x^2+2*a*x*b+b^2
In> Expand((a*x+b)^2,x);
Out> a^2*x^2+(b*a+a*b)*x+b^2
In> Simplify(%);
Out> a^2*x^2+2*a*x*b+b^2
In>PrettyForm(%);
Out>
 2    2                    2
a  * x  + 2 * a * x * b + b
```

Stupeň polynomu (neboli nejvyšší mocninu polynomu) získáme zadáním příkazu `Degree`, koeficient i-tého členu polynomu příkazem `Coef`. Zde je prvním argumentem daný polynom, druhým proměnná polynomu a třetím pořadí členů.[1]

Pro zjištění nejvyššího společného dělitele všech členů polynomu použijeme příkaz `Content`, příkaz `PrimitivePart` nám vypíše výsledek po vydělení polynomu výstupem předchozího příkazu.[2] Koeficient členu daného polynomu o nejvyšším stupni získáme příkazem `LeadingCoef`, polynom po vydělení výstupem předchozího příkazu získáme zadáním příkazu `Monic`.[3]

[1] Mějme na paměti, že se členy číslují od nuly.

[2] Z toho vyplývá vztah `polynom=Content(polynom)*PrimitivePart(polynom)`.

[3] Z toho vyplývá vztah `polynom=LeadingCoef(polynom)*Monic(polynom)`.

4.3 Řešení rovnic

YACAS obsahuje několik algoritmů pro řešení rovnic:

Prvním příkazem je `Solve`. Jeho prvním argumentem je sama rovnice (zadává se ve tvaru `L == P`, pokud zadáme jen výraz (levou stranu rovnice), pak algoritmus položí tento výraz roven nule), druhým argumentem je proměnná, pro kterou chceme danou rovnici řešit. Výstupem tohoto příkazu je seznam výsledků ve tvaru `proměnná == výsledek`. Pokud `Solve` nenajde výsledek nebo daná rovnice výsledek nemá, výstupem bude prázdný seznam.

```
In> Solve(x^2-4 == 0,x);
Out> {x==2,x==(-2)}
```

Sami autoří o této funkci píší, že zatím není příliš silná. Je třeba znát několik věcí: `Solve` neumí vyřešit všechny řešitelné rovnice. Pokud však rovnice nelze vyřešit analyticky, můžeme použít funkci `Newton` pro numerické řešení Newtonovou metodou. `Solve` neumí řešit soustavy rovnic, pro lineární soustavy je možno použít funkci `MatrixSolve` a některé nelineární systémy rovnic se dají řešit funkcí `OldSolve`. Periodicita trigonometrických funkcí (a imaginární exponenciály) není brána v potaz, tím pádem `Solve` nevypíše všechny výsledky. `Solve` předpokládá, že všechny jmenovatele jsou nenulové – takže výsledek jeho výstupu nemusí vyhovovat rovnici. Z těchto poznámek si můžeme vyvodit, že není dobré slepě spoléhat na výsledky funkce `Solve`, ale je dobré si je ověřit.

`OldSolve` je starší verzí procedury `Solve`. Tato byla zachována z praktických důvodů – liší se od `Solve`, takže má uživatel možnost využít různé algoritmy. Navíc `OldSolve` zvládá řešit i soustavy rovnic. Jejím prvním argumentem je seznam rovnic (resp. jedna rovnice) a druhým je seznam proměnných (resp. jedna proměnná), pro které chceme systém řešit.

```
In> OldSolve({x+y==0,x-y==1},{x,y});
Out> {{x==1/2,y==(-1)/2}}
```

`PSolve` hledá kořeny polynomu, prvním argumentem je polynom a druhým jeho proměnná. Výstupem bude seznam jeho kořenů. Tato procedura zvládne zpracovat polynomy do čtvrtého stupně.

```
In> PSolve(x^2+1,x);
Out> {Complex(0,1),Complex(0,-1)}
```

Kapitola 5

Matematická analýza

5.1 Limity

Pro výpočet limity funkce jedné proměnné použijeme příkaz `Limit`. Tento příkaz má formu `Limit(var, val, dir) expr`, kde `var` je proměnná dané funkce, `val` značí číslo, v jehož okolí limitu hledáme, `dir` je volitelný parametr pro výpočet jednostrannách limit a `expr` je daná funkce, jejíž limitu hledáme. Chceme-li vypočítat limitu v nevlastním bodě funkce, zadáme jako `val Infinity` pro $+\infty$ a `-Infinity` pro $-\infty$. Pokud chceme znát limitu funkce zleva, bude se hodnota `dir` rovnat `Left`, pro limitu zprava `Right`. U oboustranných limit se tímto parametrem nebudeme zabývat, tedy příkaz `Limit` bude mít v závorce pouze první dva argumenty.

```
In> Limit(x,0) Sin(x)/x;
Out> 1

In> Limit(x,0,Left) 1/x;
Out> -Infinity
```

5.2 Derivace

YACAS umí derivovat elementární funkce a funkce z nich složené. Derivace dané funkce jedné či více proměnných se v YACASu provádí příkazem `D` (D jako differentiation). Příkaz `D` má tvar `D(var, n) expr`, kde `var` je proměnná nebo seznam proměnných, podle které chceme danou funkci (zadanou v argumentu `expr`) derivovat. Pokud zadáme seznam proměnných, pak jako výsledek dostaneme seznam derivací podle daných proměnných. Poslední parametr `n` je řád derivace, pokud není zadán, bude YACAS automaticky vracet první derivaci. Pokud argument `expr` bude zadán jako seznam funkcí, pak musí mít stejnou délku jako seznam proměnných a výpočet bude probíhat tak, že se první funkce v seznamu zderivuje podle první pro-

měnné v seznamu atd. Pokud zadáme `n` rovno nule, pak YACAS vrátí jako výsledek původní funkci (respektive seznam funkcí).

```
In> D(x) Sin(x);
Out> Cos(x)
In> D(x,2) Sin(x);
Out> -Sin(x)
In> D(x) Exp(x^2);
Out> 2*x*Exp(x^2)
```

5.3 Taylorův polynom

Z nástrojů pro aproximaci funkcí jedné proměnné v okolí daného bodu má YACAS implementovaný příkaz `Taylor`, který vytvoří Taylorův polynom dané funkce. Zápis tohoto příkazu vypadá následovně: `Taylor(var, at, order) expr`, kde `var` je nezávislá proměnná dané funkce, `at` je bod (respektive hodnota proměnné), v jehož okolí chceme Taylorův polynom vytvořit, `order` značí stupeň Taylorova polynomu a `expr` označuje funkci, jíž chceme aproximovat.

```
In> Taylor(x,0,5) Exp(x);
Out> x+x^2/2+x^3/6+x^4/24+x^5/120+1
In> PrettyForm(%);
Out>
     2    3    4    5
    x    x    x    x
x + -- + -- + -- + --- + 1
    2    6    24   120
```

5.4 Integrál

Určitý integrál z dané funkce v YACASu vypočítáme použitím příkazu `Integrate`. Tento příkaz má tvar `Integrate(var, x1, x2) expr`, kde `var` je integrační proměnná, `x1` je dolní mez integrálu, `x2` je horní mez integrálu a jako `expr` dosadíme danou funkci, kterou chceme integrovat. Pokud vynecháme horní a dolní mez integrálu, pak jako výsledek dostaneme primitivní funkci zadané funkce. Pro integrování zatím byly implementovány některé jednoduché metody, což znamená, že zatím můžeme integrovat polynomy, některé jejich podíly, trigonometrické funkce a jejich inverze, hyperbolické funkce a jejich inverze, exponenciální a logaritmické funkce a některé složené funkce funkcí předem jmenovaných.

```
In> Integrate(x) Sin(x);
```

```
Out> -Cos(x)
In> Integrate(x,0,Pi/2) Sin(x);
Out> 1
```

5.5 Diferenciální rovnice

YACAS obsahuje funkci `OdeSolve`, která řeší homogenní lineární diferenciální rovnice nejvýše 2. stupně s reálnými koeficienty. Její zápis vypadá následovně:
`OdeSolve( L == 0)`, kde levá strana rovnice obsahuje lineární kombinaci derivací, které zapíšeme jako `y'` (první derivace) a `y''`(druhá derivace). Výpočet probíhá tak, že kořeny charakteristické rovnice jsou zadány jako argumenty exponenciálních funkcí.

```
In> OdeSolve(y''+y' == 0);
Out> C96+C100*Exp(-x);
```

`C96` a `C100` jsou názvy konstant - YACAS obsahuje proceduru pro generování jedinečných konstant. Výsledek můžeme otestovat příkazem `OdeTest`, kde první argument je levá strana dané diferenciální rovnice a druhý je výsledek k otestování.

Pro řešení lineárních diferenciálních rovnic se může hodit Wronského matice. Tuto matici získáme příkazem `WronskianMatrix(func,var)`, kde `func` je seznam funkcí y_1 až y_n a `var` nezávislá proměnná. Wronského matice se používá pro zjištění, zda-li nejsou rovnice lineárně závislé. To platí, pokud je determinant Wronského matice neboli wronskián – `Determinant(WronskianMatrix(func,var))` roven nule.

Kapitola 6

Grafy funkcí

Další z možností využití YACASu je tvorba grafů. YACAS neobsahuje program pro vykreslování (tedy vlastní zobrazení) grafů, proto je třeba mít nainstalovaný nějaký vykreslovací software. Implicitně je používán gnuplot. Ten můžeme nalézt na stránkách http://www.gnuplot.info/ (konkrétně na ftp://ftp.gnuplot.info/pub/gnuplot/) – jedná se o multiplatformní software, na výše zmíněných stránkách tedy nalezneme verze pro různé operační systémy (Windows, DOS, UNIX(Linux),...). Ve Windows je třeba rozbalit archiv se soubory gnuplotu do adresáře, kde máme YACAS.

Prostřednictvím YACASu můžeme vykreslovat v kartézské soustavě souřadnic funkce jedné a dvou proměnných, a to ve tvaru $y = f(x)$ (respektive $z = f(x, y)$).

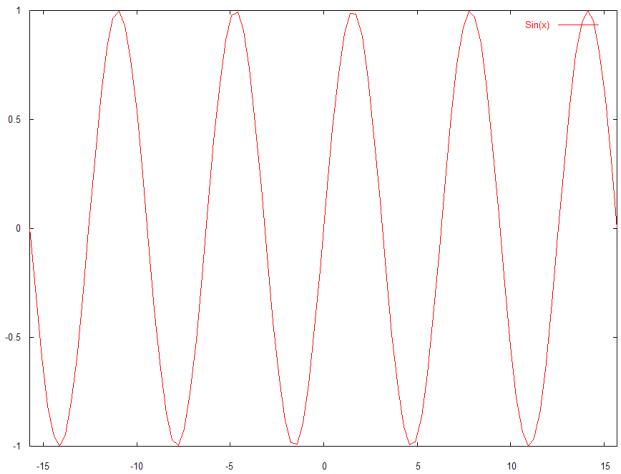
6.1 Grafy funkcí jedné proměnné

`Plot2D(f(x), a:b, ...)`

Procedura `Plot2D` vykresluje jednu nebo zároveň více funkcí jedné proměnné $f(x)$ v daném intervalu. Můžeme specifikovat více parametrů této procedury, o kterých bude řeč dále. Výstup může být poslán do vykreslovacímu programu (standardně je nastaven gnuplot), do datového souboru nebo do seznamu bodů. Funkci $f(x)$ jako parametr této procedury musí YACAS vyhodnotit jako výraz s nejvýše jednou proměnnou (tato se nemusí jmenovat x), také `N(f(x))` musí YACAS vyhodnotit jako reálnou (ne komplexní) hodnotu při zadání číselné hodnoty argumentu x. Pokud daná funkce nesplňuje tyto podmínky, YACAS nahlásí chybu. Můžeme vložit více funkcí jako seznam a ty nemusí být závislé na stejné proměnné, např. {`f(x), g(y)`}. Funkce budou vykresleny do stejného grafu se stejnými rozsahy souřadnic. Standardní rozsah souřadnic je interval $(-5, 5)$, toto můžeme změnit definováním vlastního rozsahu `a:b`, kde `a` je levá hranice intervalu a `b` pravá. Parametr `yrange` udává rozsah druhé souřadnice pro vykreslování, např. `yrange=0:10`. Pokud není tento parametr zadefinován, ponechává se volba rozsahu na vykreslovacím programu. Měli bychom si dát pozor na to, že vykreslovací program `gnuplot` vybere rozsah na základě zadaných dat,

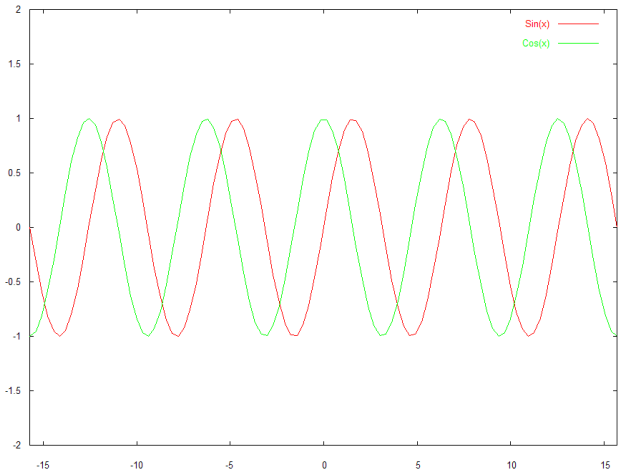
což může vést k vizuálně špatným výsledkům, pokud má funkce singularitu. Můžeme si nastavit další parametry (přesnost vykreslování, výstup do souboru, atd.), o nich se více dozvíme v manuálu.

```
In> Plot2D(Sin(x),-5*Pi:5*Pi)
```



Obrázek 6.1: Graf funkce $sin(x)$

```
In> Plot2D(Sin(x),Cos(x),-5*Pi:5*Pi,yrange=-2:2)
```



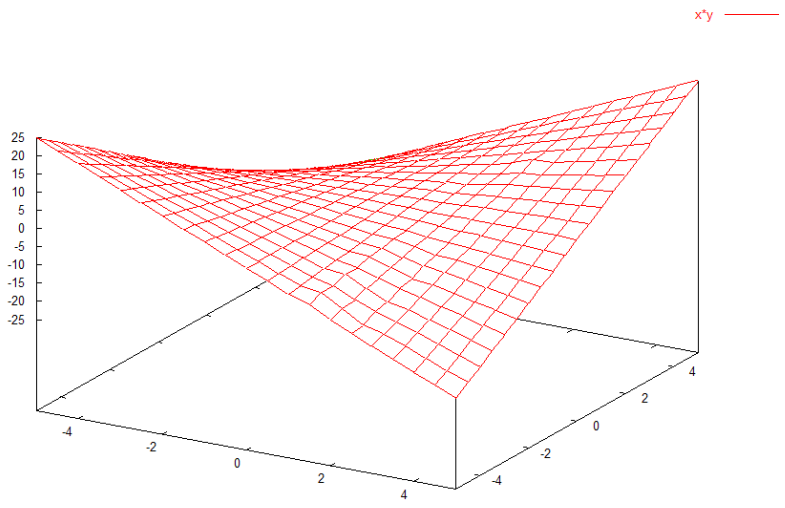
Obrázek 6.2: Graf funkcí $sin(x)$ a $cos(x)$

6.2 Grafy funkcí dvou proměnných

`Plot3DS(f(x,y), a:b, c:d, ...)`

Procedura Plot3DS vykresluje jednu nebo zároveň více funkcí dvou proměnných $f(x, y)$ v daném intervalu. Výstup bude ukázán jako plocha daná rovnicí $z = f(x, y)$. Podobně jako u dvourozměrného vykreslování můžeme u Plot3DS specifikovat více parametrů. Výstup může být poslán do vykreslovacímu programu (standardně je nastaven gnuplot), do datového souboru nebo do seznamu. Podobně jako u Plot2D musí funkce $f(x, y)$ z argumentu splňovat určité podmínky: tuto funkci musí YACAS vyhodnotit jako výraz s nejvýše dvěma proměnnými (ty se nutně nemusí jmenovat `x` a `y`), `N(f(x,y))` musí YACAS vyhodnotit jako reálné číslo při zadání číselných hodnot. Můžeme vložit více funkcí jako seznam, tyto funkce musí být závislé na stejných proměnných – vyhovuje {`f(x,y), g(y,x)`}, nevyhovuje {`f(x,y), g(a,b)`}. Funkce budou vykresleny do stejného grafu se stejnými rozsahy souřadnic. Standardní rozsah x-ové a y-ové souřadnice je interval $(-5, 5)$ u obou. Pomocí xrange, yrange, zrange volíme rozsah příslušných souřadnic (např. `xrange=-10:10`). Podobně jako u Plot2D můžeme nastavit další parametry, o kterých se dočteme v manuálu.

```
In> Plot3DS(x*y,-5:5,-5:5)
```



Obrázek 6.3: Graf funkce xy

Kapitola 7

Větvení a cykly

Přímo v příkazovém řádku YACASu můžeme používat některé programátorské struktury. Pro větvení použijeme příkaz `If(pred, then, else)`, kde první argument je podmínka, jejíž pravdivost se bude ověřovat, druhý argument je příkaz, který se spustí v případě, že podmínka bude vyhodnocena jako `True`, a třetí (nepovinný) argument je příkaz, který se spustí v případě, že podmínka bude vyhodnocena jako `False`. Příkaz `If` můžeme použít i při definování funkcí.

```
In> f(x):=If(x>0,Echo("x je kladné"),Echo("x je záporné"));
Out> True;
In> x:=5;
Out> 5;
In> f(x);
Out> x je kladné
```

Z funkcí YACASu pro použití cyklů si představíme `While` a `Until`. `While(pred) body` vykonává příkazy uvedené v `body`, dokud je podmínka `pred` vyhodnocována jako `True` a `Until(pred) body` vykonává příkazy uvedené v `body` tak dlouho, dokud nebude podmínka `pred` vyhodnocena jako `True`. Pokud v `body` chceme použít více příkazů, je třeba je uzavřít do hranatých závorek.

```
In> x:=1;
Out> 1;
In> While(x<6) [Echo({x,x^2});x++;];
Out> 1 1
2 4
3 9
4 16
5 25
```

Kapitola 8

Ostatní volně šiřitelné CAS

Kromě YACASu existuje řada dalších volně šiřitelných systémů počítačové algebry. Obrázek si můžeme udělat na http://en.wikipedia.org/wiki/Category:Free_computer_algebra_systems. Ze známějších můžeme jmenovat např.:

- Maxima (http://sourceforge.net/projects/maxima) je založena na verzi programu Macsyma z roku 1982, který byl vyvinut MIT pro Ministerstvo energetiky USA. Je multiplatformní a existuje pro ni mnoho grafických prostředí. Obsahuje vlastní programovací jazyk. Dalo by se říct, že je všeobecně zaměřená.

- Axiom (http://wiki.axiom-developer.org/) je vyvíjen od roku 1973, původně společností IBM jako komerční software pod názvem Scratchpad. Nyní jde o otevřený a volně šiřitelný software. Používá se zejména pro výzkum a vývoj matematických algoritmů. Obsahuje vlastní programovací jazyk.

- PARI/GP (http://pari.math.u-bordeaux.fr/) se začal vyvíjet v roce 1985 a jeho vývoj dnes stále pokračuje pod taktovkou bordeauxské univerzity. Je primárně určen pro teorii čísel, jeho hlavním kladem je velká rychlost výpočtů. Na druhou stranu je méně obratný v symbolické manipulaci.

Závěr

Na předchozích stránkách jsem nastínil využití programu YACAS při studiu matematiky. Zůstává ovšem otázkou, do jaké míry tento software naplňuje potřeby uživatelů. Nejprve pár vět k rozhraní YACASu. Pracoval jsem s ním v operačním systému Windows XP, a to jak s jeho textovou verzí, tak s grafickým rozhraním GUYACAS. GUYACAS je nedoladěný a často se stává, že se sám ukončí, případně nezobrazuje zadaná data. Jeho jedinou výhodou je možnost jednoduše zkopírovat pomocí myši zadané příkazy do schránky. Textová verze YACASu je o poznání stabilnější a dalo by se říci, že i přehlednější. U obou ovšem zůstává problém s voláním externích příkazů, což je vidět zejména u nápovědy. Ta je standardně navázána na prohlížeč lynx a ve Windows nefunguje ani s ním, ani při snaze nastavit jiný internetový prohlížeč.

Z hlediska funkčnosti by se YACASu dalo vytknout malé množství procedur, což je ale dáno jeho koncepcí jako malého, ale účinného nástroje. Kdo chce využít některou speciální funkci, může si ji doprogramovat. Některé procedury jsou ve své funkčnosti omezeny (jako například metody integrování v proceduře `Integrate`). YACAS nemůže konkurovat drahým komerčním CAS, ale pro účely začínajícího studenta matematiky se jeví být dostatečným a v tomto případě zůstává rozhodujícím faktorem dostupnost YACASu zdarma.

Literatura

[1] Došlá, Z., Plch, R., Sojka, P.: *Diferenciální počet funkcí více proměnných s programem Maple V*, Masarykova univerzita, Brno 1999

[2] *Yacas user's function reference*,
URL: http://yacas.sourceforge.net/ref.html [cit. 2007-05-05]

[3] *Introduction to Yacas: tutorial and examples*,
URL: http://yacas.sourceforge.net/intromanual.html [cit. 2007-05-01]

[4] *Essays on Yacas*,
URL: http://yacas.sourceforge.net/essaysmanual.html [cit. 2007-05-02]

[5] Rybička, J.: *LaTeX pro začátečníky*, Konvoj, Brno 2003