Základy programování v GNU Octave pro předmět PPAŘ

Introduction to programing in Octave for subject denoted as Computer Aires Automation Control

Jaroslav Popelka

Bakalářská práce
2008

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2007/2008

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jaroslav POPELKA**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Základy programování v GNU Octave pro předmět PPAŘ**

Zásady pro vypracování:

1. **Seznamte se s bakalářskými pracemi vytvořenými na FAI UTB ve Zlíně zabývajícími se Octave.**
2. **Nastudujte český manuál tohoto software a projděte si syntaxi a základy v práce v programu Octave.**
3. **Vytvořte programy v Octave pokrývající obsahově celou výuku základů programování v MATLABu na FAI.**
4. **Dále uveďte rozdíly v syntaxi vámi vytvořených programů v Octave vůči syntaxi MATLABu.**
5. **Vámi vytvořené programy v Octave umístěte na samostatné CD-ROM jako přílohu bakalářské práce.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Perůtka (2005): MATLAB – Základy pro studenty automatizace a informačních technologií. Skriptum, 1. vydání, UTB ve Zlíně, Zlín.**
2. **Just, M. (2006): Český průvodce programem Octave. Bakalářská práce, UTB ve Zlíně, Fakulta aplikované informatiky.**
3. **Prokop, D. (2007): Průvodce grafickými nadstavbami Octave. Bakalářská práce, UTB ve Zlíně, Fakulta aplikované informatiky.**
4. **Heczko, M. (2006): Výukový a zkušební program pro předmět PPAŘ. Bakalářská práce, UTB ve Zlíně, Fakulta aplikované informatiky.**
5. **Selhofer, H., Oliver, M. (2003): Introduction to GNU Octave. International University Bremen, Německo [online]. [cit 2008-01-28]. Dostupný z WWW: (http://math.jacobs-university.de/oliver/teaching/iub/resources/octave/octave-intro.pdf)**

Vedoucí bakalářské práce: **Ing. Karel Perůtka, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **20. února 2008**

Termín odevzdání bakalářské práce: **5. května 2008**

Ve Zlíně dne 20. února 2008

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato bakalářská práce pojednává o možnostech použití software Octave jako alternativní software Matlabu při výuce předmětu Programová podpora automatického řízení .

První část se věnuje samotnému programu. Je popsána jeho historie, instalace a prostředí a také jsou zde uvedeny základní příkazy potřebné pro práci v programu.

Druhá část se věnuje vypracovaným programům. Ty jsou rozděleny na dvě části. První demonstrují příkazy vysvětlené v teoretické části , druhé jsou komplexnější a používají větší množství příkazů. Všechny programy jsou umístěny na přiloženém CD.

Klíčová slova:

Octave , Matlab , Programování

ABSTRACT

This bachelor thesis deals about possibilities with using software Octave as alternative software for Matlab on instruction for subject Computer aid of automatic control .
First part is about program. Them is described his history, installation and environment and also are here presented any basic statements which are needed for work in program.
Second part is dedicated for all elaborated programs. Those are divided on two parts. First demonstrate all commands explained on theoretic parts , second are more complex and use more command. All programs are placed on enclosed CD.

Keywords:

Octave , Matlab , Programming

Rád bych zde na tomto místě poděkoval vedoucímu bakalářské práce Ing. Karlu Perůtkovi za odbornou pomoc , rady , návrhy i připomínky , které mi poskytl během vytváření této bakalářské práce .

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně dne 15.5.2008

…….
Jaroslav Popelka

OBSAH

ÚVOD

GNU Octave je vyšší programovací jazyk určený zejména pro numerické operace . Jeho počátky sahají do roku 1988 , avšak jeho stálý vývoj začal až o čtyři roky později . Za jeho autora je považován John W. Eaton a jeho tým . Dnes je jej možné spustit na všech standardních operačních systémech – Linux (Unix) , Mac OS i MS Windows .

Jeho řádkové rozhraní je zcela vyhovující pro řešení lineárních i nelineárních numerických problémů , přičemž pro řešení těchto početních experimentů používá jazyka podobnému tomu , jenž je znám z programu Matlab .

Program disponuje rozsáhlými nástroji pro řešení běžných numerických problémů algebry , ale ocení jej i ti co potřebují vykonávat složitější výpočty krok po kroku , přičemž nemají ve svém počítači jinou vhodnou kalkulačku . Je velmi lehce rozšiřitelný a přizpůsobitelný pomocí funkcí definovaných uživatelem napsaných přímo v jazyce Octave , popřípadě v jiných jazycích jako C , C++ nebo Fortan . [9]

GNU Octave je volně šiřitelný program a je možné jej tedy používat , distribuovat a modifikovat zcela podle veřejné licence GNU .

I. TEORETICKÁ ČÁST

1 PŘEDMĚT PPAŘ

Hlavním úkolem této práce je vytvoření programů v Octave, které svým obsahem pokrývají výuku předmětu Programová podpora automatického řízení (PPAŘ, A4PAR) , vyučovaném na Univerzitě Tomáše Bati ve Zlíně – Fakultě aplikované informatiky. V současné době je v předmětu používán výhradně software Matlab. Nejedná se však o jediný program, který by mohl sloužit studentům. Vhodnou alternativou programu Matlab je program Octave, protože je šířen narda pod GNU licencí .

1.1 Přehled probírané látky

Předmět A4PAR studenty seznamuje se základy programování, především pak numerických operací. Nynější přehled probírané látky je přibližně tento :

- Úvod do A4PAR, historie Matlabu
- Popis prostředí Matlab : ovládání, nápověda, proměnné
- Vektory, matice, řetězce, práce s daty
- Vykreslování grafů, vizualizace
- Programování, skripty, funkce
- Toolboxy, Simulink, GUI

V následující části, která může posloužit také jako základní učební pomůcka, jsou realizovány body 1 – 5 z hlediska Octave. Poslední bod není uveden, jelikož jej Octave nepodporuje.

Praktická část práce se zabývá demonstračními programy, které byly vytvořeny jednotlivým okruhům výuky programování v Octave. Zároveň tyto programy jsou porovnány s programy vytvořenými v Matlabu a jsou uvedeny rozdíly v jejich syntaxi.

2 OCTAVE

2.1 Historie

Octave bylo původně při svém vzniku v roce 1988 koncipováno jako určitý druh software pro vysokoškolské studenty. James B. Rawling a John G. Ekerdt jej pojmenovali podle svého profesora, kterého uznávali nejen pro jeho znalost chemie a chemických reakcí, ale i pro jeho speciální postupy při matematických výpočtech. Původně program vytvořili jako učební pomůcku vhodnou pro návrh chemických reaktorů. Představovali si jej jako velmi specializovaný nástroj pro řešení problémů spjatých s touto oblastí, avšak později svolili k vybudování mnohem flexibilnějšího nástroje. Zároveň byli autoři z mnoha stran přesvědčováni , aby upustili od svého projektu a přešli raději k použití jazyka Fortran, oba ale byli přesvědčeni, že právě s takovým interaktivním prostředím, jakým Octave je, budou studenti schopni zpracovat všechna důležitá fakta mnohem rychleji. [6]

Trvalý vývoj software začal na jaře roku 1992 pod vedením Johna W. Eatona, první zkušební verze byla poskytnuta o rok později. Verze s označením 1.0 byla vydána 17 února 1994. Od té doby program prodělal několik významnějších oprav, byl zařazen do několika distribucí Debian GNU/Linux či SuSe Linux a roku 1997 byl recenzován v prestižním časopise Linux Magazine . V tomto roce byla uvedena verze s označením 3.0. [9]

Dle samotných autorů je dnes Octave mnohem více, než jen další výukový software. I přes nejasné počáteční cíle však věděli, že chtějí vytvořit program, který by umožňoval studentům řešit jejich reálné problémy související nejen s problémy při návrhu chemických reaktorů. Výsledkem jsou dnes tisíce lidí po celém světě používající tento software nejen ve výuce a výzkumu ale i ke komerci.

2.2 Instalace

2.2.1 Výběr verze

Od prosince roku 2007 je volně k dispozici verze programu označená jako 3.0. Jedná se poslední stabilní formu programu – předchozí stabilní sestavení neslo označení 2.1.5 a bylo vydáno v létě roku 2003. Od té doby vyšlo několik zkušebních verzí, které vždy obsahovaly určité novinky oproti těm předchozím. V březnu roku 2005 jich ve verzi 2.9.X bylo importováno již takové množství, že bylo rozhodnuto k významnější změně číslice v určení verze programu. Všechny novinky verze 3.0 jsou vypsány v souboru ReadMe ve složce s programem.

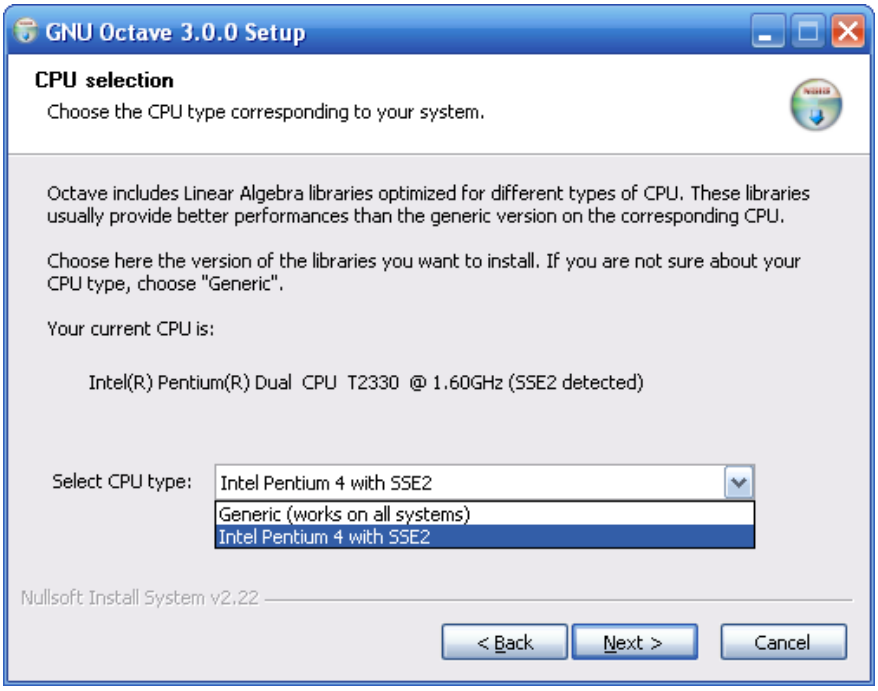
2.2.2 Průběh instalace

Octave je open-source program určený pro všechny běžné operační systémy. V rámci bakalářské práce byl nainstalován pod operační systém Windows XP společnosti Microsoft. Samotný software lze stáhnout například ze stránek *www.octave.org*. Velikost vybraného instalačního souboru je zhruba 35 MB, tedy více jak 4 krát větší než u předchozí verze. I samotná instalace je od staršího sestavení poněkud rozdílná (průběh instalace u verze 2.1.5 je popsána na stránkách *www.octave.cz*) .

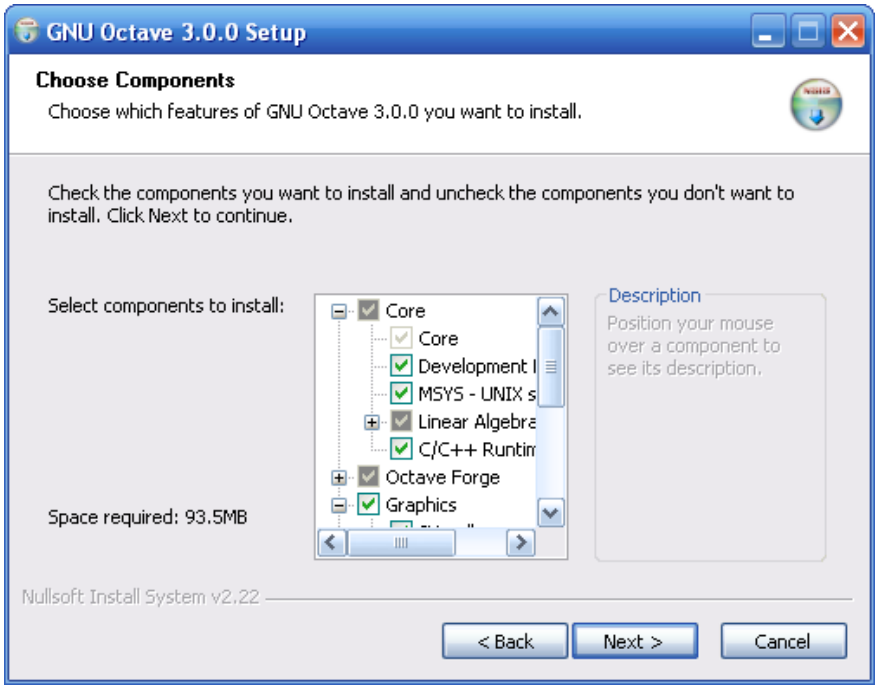
V prvním kroku je uživateli zobrazena licenční smlouva, se kterou je pro postup v instalaci nutné souhlasit. Další okno umožňuje výběr použitého CPU. Octave totiž obsahuje knihovnu lineární algebry, která je optimalizována pro CPU s podporou instrukcí SSE2. Na těch potom tato knihovna obvykle poskytuje mnohem lepší výkon, než na ostatních.

Po vybrání procesoru je uživateli nabídnut seznam komponent, které je možné instalovat společně s programem. Kromě samotných souborů programu je možné určit grafické komponenty (Java Handles nebo standardní Gnuplot) , dokumentaci v HTML či PDF a především doplňky (editor kódu , konzoli) . Uživatel tak jednoduše označí všechny části , které chce společně s programem nainstalovat. Pokud se budou instalovat oba dva typy grafických komponent, vybere si uživatel v další části tu , kterou bude program považovat za primární.

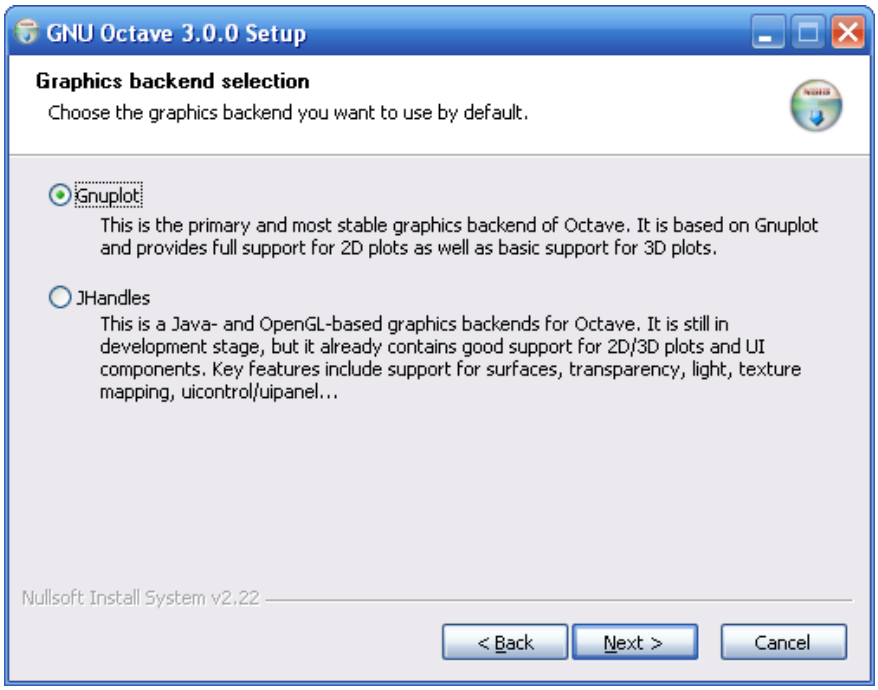
Poslední 2 kroky, které může uživatel ovlivnit je specifikace adresáře kam se program nainstaluje a následně název adresáře s programem ve složce `start`. O samotném průběhu instalace je potom uživatel informován jak graficky, tak i detailním výpisem právě prováděné práce. Posledním oknem je oznámení o úspěšném dokončení instalace s možností zobrazení souboru ReadMe.



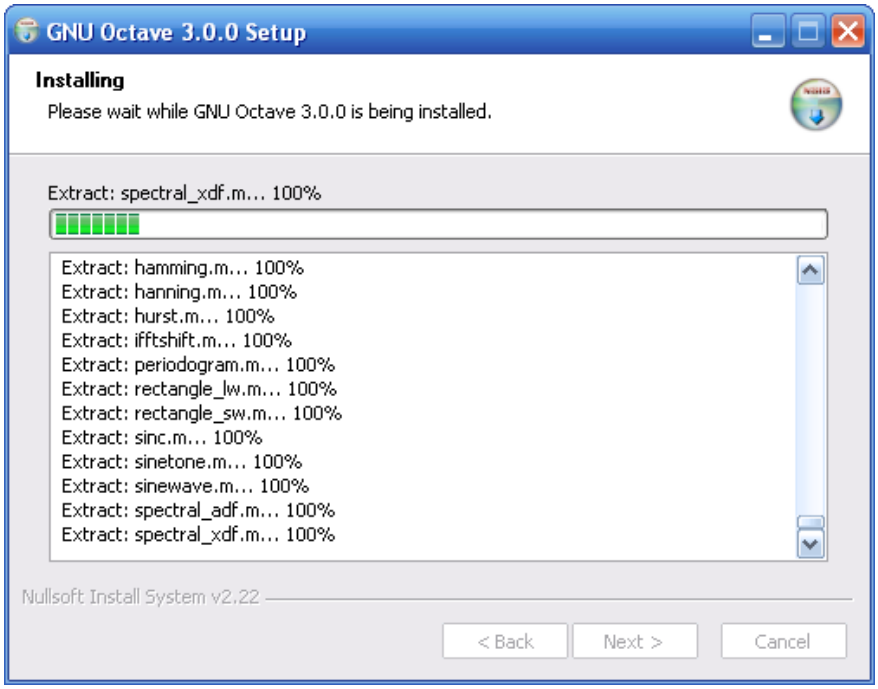
Obr. 1 – Instalace Octave : výběr CPU



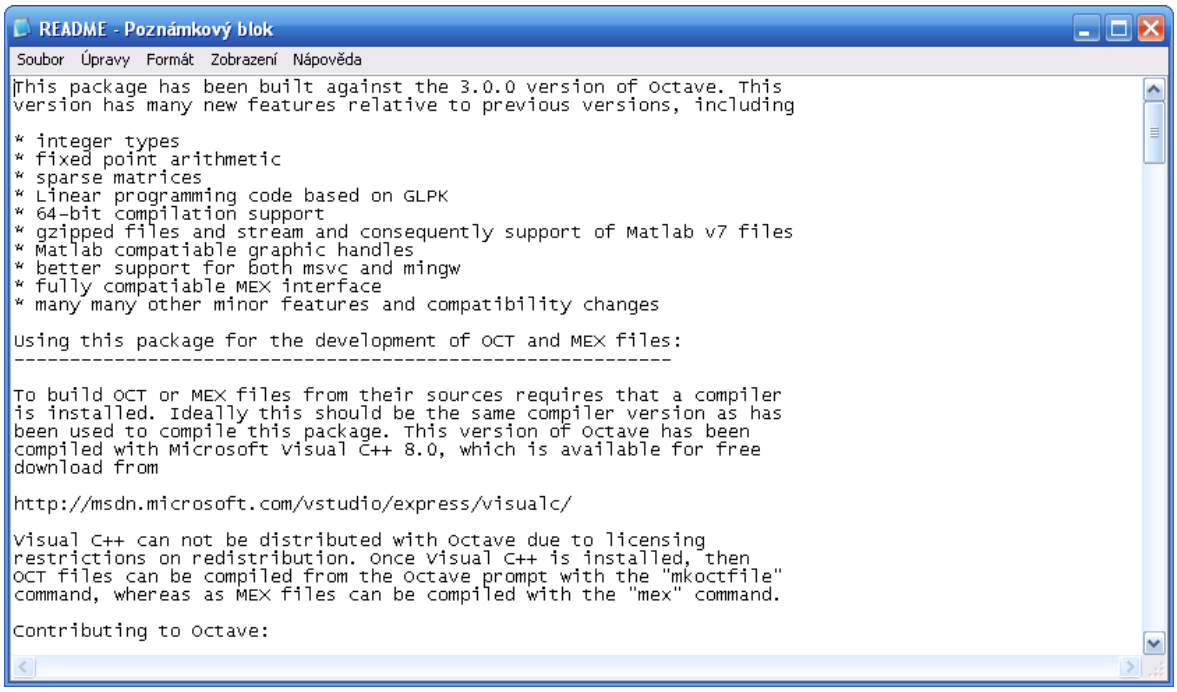
Obr. 2 – Instalace Octave : výběr instalovaných komponent



Obr. 3 – Instalace Octave : výběr primárního grafického balíku



Obr. 4 – Instalace Octave : průběh instalace



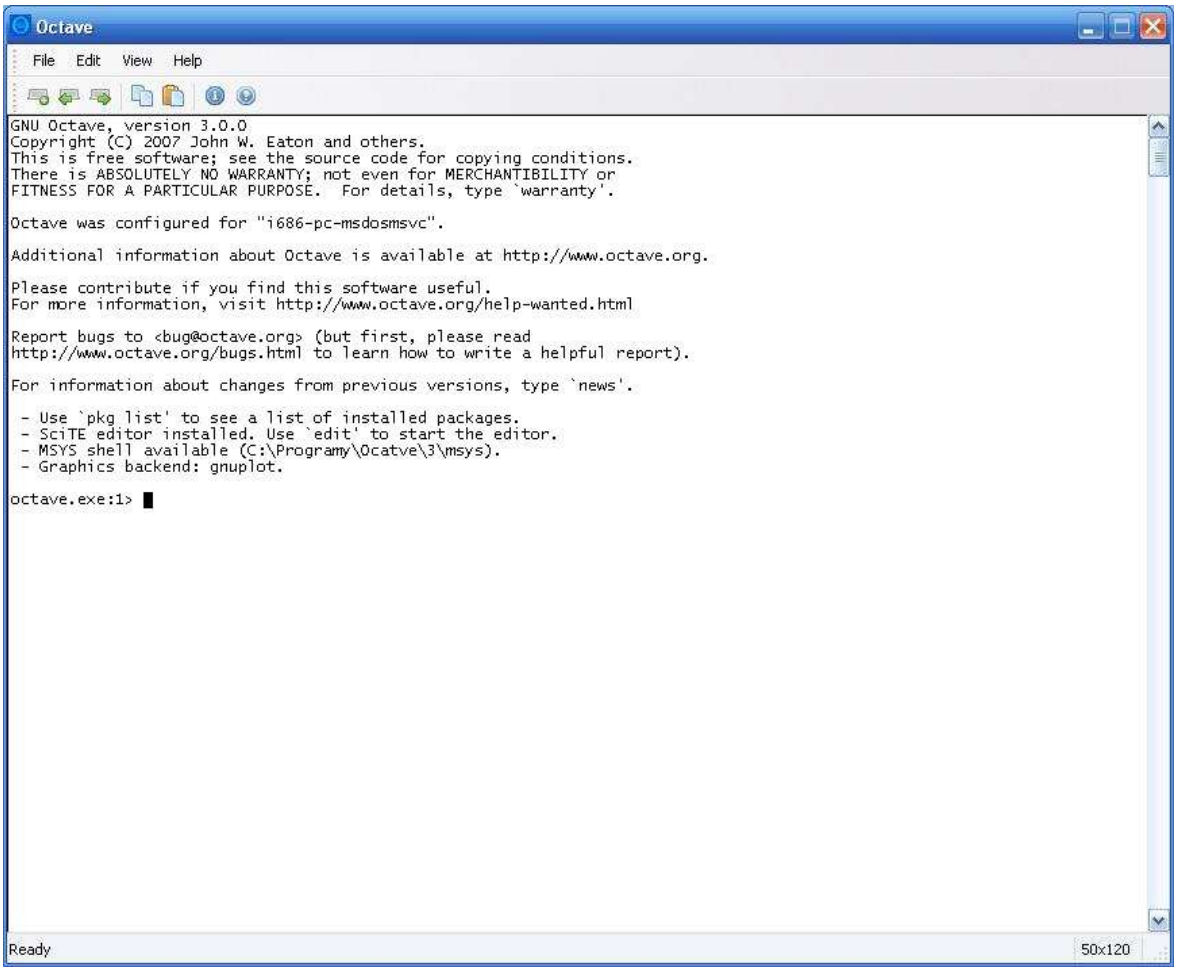
Obr. 5 – Instalace Octave : soubor README

2.2.3 První spuštění

Po úspěšné instalaci se na ploše vytvoří zástupce programu. Dvojitým poklikáním se program spustí. Pokud se tak děje na počítači s právě instalovaným systémem, může se vyskytnout chybové okno oznamující nenalezení knihovny `msvcp71.dll`. V takovém případě stačí tuto knihovnu stáhnou z internetu a uložit ji do složky `system32`, popřípadě instalovat sadu knihoven MS .NET Framework 2.0.

Jak již bylo psáno, software funguje na příkazové řádce. Pokud však nainstalujeme i doplněk console, bude tato řádka umístěna do grafického obalu, který umožňuje vytváření několika na sobě nezávislých oken a také kopírování a vkládání příkazů .

První řádky spuštěného programu obsahují výpis informací. Následuje krátká inicializace, během které jsou zjištěny všechny instalované komponenty, jenž jsou následně také vypsány. Za tímto blokem se nachází první řádek určený pro samotnou práci v programu.



Obr. 6 – První spuštění

2.3 Základy práce

Tato část se zabývá základními funkcemi, které Octave uživateli nabízí a které jsou následně demonstrovány na vytvořených programech. Popisuje především hlavní příkazy a vlastnosti daných funkcí. Kompletní seznam příkazů a podrobný profil jednotlivých funkcí popisuje webový manuál, který lze najít na adrese `http://octave.wz.cz`.

2.3.1 Výchozí příkazy a poznatky

Před samotnou prací je potřeba se s programem seznámit. Prvním příkazem by proto měl být `help`. Po jeho zadání se na obrazovce vypíše několikastránková nápověda obsahující seznam všech operátorů, rezervovaných slov, funkcí a zabudovaných proměnných. Zapsáním jména položky ze seznamu za příkaz help dostaneme podrobnou nápovědu k této položce. V tabulce 1 jsou uvedeny základní příkazy pro práci s Octave (vlevo příkaz , vpravo popis jeho funkce) .

`pwd`	zobrazení aktuálního adresáře
`dir`	zobrazení obsahu aktuálního adresáře
`who`	seznam všech proměnných
`whos`	seznam všech proměnných , včetně velikosti , počtu bajtů a typu proměnné
`clear`	smazání všech proměnných
`clc`	smazání celé obrazovky
`edit`	otevře editor pro editaci externích souborů

Tab. 1 – Základní příkazy pro práci s programem

Velmi zajímavou a často aplikovanou vlastností programu je používání klávesy Tabulátoru. Ta slouží pro doplňování slov. Pokud uživatel napíše první písmena příkazu a stlačí klávesu `Tab`, zobrazí se seznam všech příkazů, které na daná písmena začínají. [8]

Posledním důležitým poznatkem je použití speciálních znaků. Octave podporuje několik různých oddělovačů, které mají různé funkce :

. odděluje celou a desetinnou část reálného čísla , užívá se při násobení prvků matic
, odděluje jednotlivé výrazy na řádku, výrazy ukončené čárkou jsou vypsány
; odděluje jednotlivé výrazy na řádku, výrazy ukončené středníkem nejsou vypsány
\ rozepsání výrazů na více řádků
nebo % uvozují komentáře k programu, slouží k přehlednosti programu

2.3.2 Datové typy a proměnné

Tak jako jiné programovací jazyky, tak i Octave má zabudované určité datové typy. Použitá verze 3 disponuje více jak 40 druhy. Jelikož je Octave určeno především pro numerické výpočty, jsou těmi hlavními ty číselné - reálné a komplexní skaláry a matice. Pokud by však uživatel potřeboval takový typ, jaký program nenabízí, je možné si jej pomocí několika řádků kódu C++ definovat. Pro práci s datovými typy jsou nejdůležitější následující příkazy :

PŘÍKAZ	VÝZNAM
`typeinfo`	zobrazí seznam obsažených datových typů
`typeinfo (a)`	zobrazí datový typ pro proměnnou a
`realmin`	minimální číslo jenž je Octave schopno uložit (2.225e-308)
`realmax`	maximální číslo jenž je Octave schopno uložit (1.797e+308)
`eps`	relativní přesnost s kterou je možné číslo uložit (2.220e-16)

Tab. 2 – Základní příkazy pro datové typy

Základní proměnná označená jako `ans` (answer – odpověď) slouží pro uložení každého posledního výsledku u kterého není specifikována vlastní proměnná. Pokud by však následoval další výpočet, byl by předešlý výsledek přemazán novým. Pro budoucí použití je proto lepší si výsledek výpočtu uchovat ve vlastní proměnné. Tu si můžeme vytvořit za běhu programu bez toho, že bychom museli udávat její typ či velikost :

```
1+2          % výsledek se uloží to základní proměnné ans
ans =3
a = 2+3      % zde se výsledek uloží do vytvořené  proměnné a
a =5
```

Avšak i při vytváření těchto proměnných je nutné dodržovat určitá pravidla. Jméno proměnné musí být sekvence znaků, čísel či podtržítek a zároveň nesmí začínat číslicí. Délka názvu proměnné není nijak omezena, ale nedoporučuje se používat příliš dlouhá pojmenování. Octave také rozlišuje malá a velká písmena – x a X jsou pro něj tedy dvě zcela rozdílné proměnné. Nakonec je nutné brát v úvahu jména již existujících či předdefinovaných proměnných. Jedná se například o již zmíněné `ans`, ale také o konstanty jako `pi` (Ludolfovo číslo) či `e` (Eulerovo číslo) . [8]

V uvedeném příkladu lze také vidět, že společně s výsledkem se zobrazí i název proměnné (`ans , a`) . Pro potlačení výpisu názvu proměnné se použije příkazu `disp` .

K proměnné se váže ještě jeden příkaz – `input`. Ten od uživatele vyžaduje vstup z klávesnice. Jeho použití se objevuje především ve funkcích a skriptech.

2.3.3 Operátory

Operátory v Octave pracují s jednou a více vstupními proměnnými a vrací pouze jednu výstupní hodnotu. Stejně jako v Matlabu rozlišujeme celkem 3 různé typy operátorů – základní, relační a logické.

Základní operátory jsou všechny nejčastěji používané matematické funkce. Jmenovitě se jedná o sčítání, odčítání, násobení, dělení a umocňování. Při použití více operátorů v jedné funkci se tyto řídí dle platných matematických pravidel o prioritě operátorů. Absolutně prioritní je umocňování, následuje násobení a dělení a jako poslední se provádí sčítání a dělení. Prioritu lze dále ovlivnit použitím kulatých závorek (hranaté a složené závorky mají v Octave jiné využití). Z hlediska programování však do této skupiny patří také operátory inkrementace a dekrementace (++ a --) . Jsou-li použity ještě před proměnnou , bude tato upravena ihned. Jsou-li však zapsány až za ní, bude proměnná upravena až při jejím dalším použití. [5]

Relační operátory slouží pro porovnávání hodnot. Výsledkem těchto operátorů je číslo logického typu dat – logická 0 nebo logická 1. Jejich seznam i příklad použití je uveden v tabulce 3.

OPERÁTOR	VÝZNAM	PŘÍKLAD
<	menší než	1<2
>	větší než	2>1
<=	menší nebo rovno než	1<=2
>=	větší nebo rovno než	2>=1
==	rovno	1==1
~= , != , <>	nerovno	1~=2

Tab. 3 – Relační operátory

Poslední skupinou jsou logické operátory. Ty slouží k aplikaci logických funkcí na pole a skaláry. Stejně jako relační operátory mají na výstupu buď logickou 0, nebo logickou 1. Nejvyšší prioritu z logických operátorů má negace, následují operátory pro pole a nejnižší prioritu mají operátory pro skaláry. Seznam všech logických operátorů uvádí tabulka 4. [2]

OPERÁTOR	VÝZNAM	PŘÍKLAD
`&`	logický součin pro pole	A&B
`&&`	logický součin pro skalár	if (x<2)&&(x>0) …
`|`	logický součet pro pole	A|B
`||`	logický součet pro skalár	if (x<2)||(x>0) …
`xor`	exkluzivní součet pro pole	xor(A,B)
`~`	negace pro pole	~A

Tab. 4 – Logické operátory

2.3.4 Podmínky , cykly a další řídící příkazy

Tak jako u jiných programovacích jazyků i v Octave slouží podmínky a cykly k určení průběhu zpracování programu a tvoří tak většinou jeho páteř. Všechny mají velmi podobnou stavbu – začínají klíčovým slovem (`if , while`) , které následuje podmínka, tělo příkazů a ukončeny jsou dalším klíčovým slovem (`endif , endwhile`) . Podle klíčových slov Octave pozná, kdy podmínka začíná a kdy končí. Umožňuje to její následné rozepsání na více řádků (tak jako v následujících příkladech) , což zpřehledňuje zdrojový soubor.

Nejznámější podmínkou používanou ve všech programovacích jazycích je podmínka `if`. V Octave se s ní můžeme setkat ve 3 různých tvarech – minimálním, základním a rozvětveném . V prvním případě je obsažena pouze jedna podmínka a tělo, které se vykoná je-li podmínka splněna. Ve druhém případě je součástí také příkaz `else`, který následuje tělo jenž se vykoná při nesplnění podmínky za `if`. Ve třetím případě se uvnitř prvního těla pomocí atributu `elseif` nachází ještě další podmínka či podmínky.

```
if(a==0)          % začátek podmínky
disp('nula')      % v případě splnění první podmínky se vypíše text
elseif(a==1)      % při nesplnění je na řadě druhá podmínka
disp('jedna')     % v případě splnění druhé podmínky se vypíše text
else              % pokud neplatí ani druhá podmínka
disp('jine')      % bude zobrazen tento text
endif             % konec podmínky
nula              % hodnota proměnné a byla 0
```

Druhou z podmínek, které dokáže Octave vyhodnocovat, je podmínka `switch`. V tomto případě je porovnávána hodnota výrazu za klíčovým slovem `switch` s hodnotami za slovy `case`. V případě, že jsou hodnoty vyhodnoceny shodné, provede program příkazy, jenž jsou součástí těla za daným `case`. Pokud není ani jedna z nabízených možností vyhodnocena jako pravdivá, provede se tělo za slovem `otherwise`:

```
switch b          % začátek podmínky
case 1            % pokud b=1…
disp('jedna')     % … vypíše se tento text
case 2            % pokud b=2…
disp('dve')       % … vypíše se tento  text
otherwise         % pokud je b = něco jiné
disp('jine')      % … vypíše se tento  text
endswitch         % konec podmínky
dve               % hodnota proměnné b byla 2
```

Cykly umožňují opakovat skupinu daných příkazů vícekrát po sobě. Octave rozeznává všechny 3 základní typy. Prvním z nich je `while`, který provádí příkazy obsažené ve svém těle tak dlouho, dokud je platná podmínka na jeho začátku. `Do-until` se od `while` liší pouze umístěním podmínky na konec. Cyklus u `while` se proto nemusí vykonat ani jedenkrát, kdežto `until` se vykoná minimálně jedenkrát. Třetí typ cyklů známý i z jiných programovacích jazyků je `for`. Tento cyklus se také někdy nazývá jako cyklus s předem daným počtem kroků, jelikož ještě před jeho prvním průchodem víme, kolikrát se bude muset provést. [5]

```
% základní tvar pro cyklus while
while ( podmínka )          % začátek cyklu
tělo                        % tělo které se vykoná při splnění podmínky
endwhile                    % konec cyklu
% základní tvar pro cyklus do-until
do                          % udělej…
tělo                        % … tyto příkazy
until ( podmínka )          % pokud platí podmínka , vrať se zpět k do
% základní tvar pro cyklus for
for výraz                   % pro daný počet kroků
tělo                        % proveď tyto příkazy
endfor                      % konec cyklu
```

K cyklům `for` a `while` se vážou ještě 2 další příkazy, které je potřeba zmínit. První z nich je `break`, který způsobuje okamžité vyskočení z cyklu. Ten je ihned ukončen a program potom pokračuje příkazy uvedenými až za cyklem. Druhý příkaz je `continue`, jenž způsobuje okamžité ukončení dané iterace a start nové. Cyklus tak není ukončen.

```
rada=round(rand(1,5)*10);  % náhodně vygenerovaná čísla
for x=rada                 % začátek cyklu
if (x==5)                  % podmínka - pokud bude x = 5
break;                     % ukonči celý cyklus
endif                      % konec podmínky
printf("%n\d",x);          % pokud nebude podmínka platit, vytiskne se číslo
endfor                     % konec cyklu
2 7                        % vytisknutá čísla

for x=rada                 % začátek cyklu
if (x<=5)                  % podmínka - pokud bude číslo rovno nebo menší 5
continue;                  % ukonči aktuální otočku cyklu a přejdi k další
endif                      % konec podmínky
printf("%n\d",x);endfor    % pokud nebude podmínka platit, vytiskne se číslo
7 8                        % vytisknutá čísla
```

2.3.5 Matematické funkce

Octave lze chápat minimálně jako velmi výkonnou kalkulačku . Kromě základních výpočtů jako je sčítání , odečítání , dělení , násobení a umocňování zvládá i práci s velkým množstvím matematických funkcí . Tyto funkce lze rozdělit do tří základních skupin – na goniometrické , exponenciální a číselné . Stručný seznam těch nejzákladnějších funkcí je uveden v tabulce 5 .

FUNKCE	VÝZNAM
`sin, cos, tan`	goniometrické funkce
`asin, acos, atan`	cyklometrické funkce
`ceil, floor`	vrací nejbližší vyšší / nižší celé číslo
`round`	zaokrouhlení dle matematických pravidel
`exp`	vypočítání exponenciálu : e^X
`log, log10`	vrací logaritmus / přirozený logaritmus
`gcd`	vrací nejvyšší společný dělitel
`lcm`	vrací nejnižší společný násobek
`mod, rem`	vrací modulo / zbytek po celočíselném dělení
`abs`	vrací absolutní hodnotu čísla
`sqrt`	vrací druhou odmocninu čísla
`sign`	fce signum : -1 pro záporná , +1 pro kladná , 0 pro 0

Tab. 5 – Matematické funkce

U velké části těchto funkcí se setkáme s tím , že výsledek výpočtu není celé čísl . V takovém případě je u reálného čísla zobrazeno pět platných desetinných číslic. I to ale uživatel může ovlivnit. Stačí před samotný výpočet zadat příkaz `format` doplněný o klíčové slovo . Pomocí `format bank` tak můžeme kupříkladu nechat vypisovat pouze 2 desetinné číslice.

```
c=3.7;          % uložení čísla do proměnné
d=floor(c)      % určení nejbližšího nižšího čísla
d=3
format bank     % určení nového výstupního formátu na 2 desetinná čísla
e=sqrt(d)       % výpočet odmocniny
e=1.73          % vypsání výsledku v požadovaném tvaru
```

Podobným případem jsou u Octave také příliš velká čísla. Ta nejsou vypisována jako celek, nýbrž jako násobky mocnin 10 s použitím oddělovače `e`. Stejným způsobem můžeme tato čísla i zadávat.

```
f=1e9;          % uložení čísla do proměnné
g=sqrt(f)       % výpočet odmocniny
g = 3.1623e+004 % vypsání velkého čísla ve formě násobku mocniny 10
```

Zároveň je potřeba podotknout, že Octave nemusí na výstup z matematické funkce vypsat pouze číslo. Takovým základním příkladem je dělení nulou. Pokud je například programu nařízeno, aby vydělil číslo 1 číslem 0, bude výsledkem `Inf` značící nekonečno. V případě, že bychom tak ale chtěli učinit s nulou samotnou, dostali bychom na výstupu `NaN`, což značí, že výsledkem není číslo. [8]

Octave podobně jako matlab počítá také s komplexními čísly. Komplexní číslo ve tvaru `a + bi` se skládá z reálné části `a` a imaginární části `b`. Stejně jako Matlab navíc připouští použití 2 různých znaků pro imaginární jednotku – `i` a `j`, přičemž je možné použít i velká písmena. Nejčastěji používané příkazy spojené s komplexními čísli udává tabulka 6.

PŘÍKAZ	VÝZNAM
`real`	vrací reálnou část komplexního čísla
`imag`	vrací imaginární část komplexního čísla
`abs`	vrací hodnotu (modul) komplexního čísla
`angle`	vrací argument komplexního čísla
`conj`	vrací komplexně sdružené číslo

Tab. 6 – Příkazy pro práci s komplexními čísli

2.3.6 Vektory a matice

Vektory a matice jsou v Octave stejně jako v Matlabu základními stavebními prvky. Při jejich zadávání je píšeme do hranatých závorek, přičemž jednotlivé prvky jsou od sebe odděleny pomocí mezer či čárek a v případě matic dělí jednotlivé řádky znaky středníku. Samotný vektor lze vytvořit i pomocí tzv. dvojtečkové metody, popřípadě příkazem `linspace`. Z několika vektorů či menších matic můžeme také složit matici větší, je však nutné dbát platných matematických pravidel tak, aby výsledná matice měla ve výsledku v každém svém řádku stejný počet sloupců. [1]

```
a=[1,2;3 4]          % vytvoření matice
a = 1  2
    3  4
b=1:2:10             % dvojtečková metoda –první prvek:krok:maximální prvek
b=1 3 5 7 9          % vytvořený vektor – další by bylo 11 – vyšší jak maximum
c=linspace(1,4,3)    % příkaz linspace( první prvek , poslední prvek , počet čísel )
c=1 2.5 4            % vytvořený vektor
```

Kromě ručního zadávání můžeme pro vytvoření matice nebo vektoru použít některou z předdefinovaných funkcí . Seznam těchto funkcí je uveden v tabulce 7.

PŘÍKAZ	VÝZNAM
`eye`	vrací jednotkovou matici
`ones`	vrací matici jedniček
`zeros`	vrací matici nul
`diag`	vrací diagonální matici
`rand , randn`	vrací matici s náhodnými čísly od 0 do 1
`randperm`	vrací vektor s náhodnými čísly od 1 do zadaného
`tril`	vrací matici s extrahovanou dolní trojúhelníkovou maticí
`triu`	vrací matici s extrahovanou horní trojúhelníkovou maticí

Tab. 7 – Příkazy pro vytváření speciálních typů matic

Posledním okruhem matic, které Octave podporuje, jsou speciální matice. Jedná se Hankelovu , Hilbertovu , Sylvesterovu či Toeplitzovu matici. Popis a funkce jednotlivých matic je k dispozici v nápovědě.

Často je také potřeba znát o maticích něco více, než jen jejich obsah. K tomuto účelu slouží v Octave příkazy uvedené v tabulce 8 .

PŘÍKAZ	VÝZNAM
`size`	vrací velikost matice : řádky , sloupce
`rows`	vrací počet řádků
`columns`	vrací počet sloupců
`length`	vrací vyšší hodnotu z hodnot vrácených funkcí size

Tab. 8 – Příkazy pro práci s maticemi 1

Dalším okruhem při práci s maticemi je jejich indexace. Indexace slouží k výpisu vybraných prvků matice. Ukázky jednotlivých způsobů indexace jsou demonstrovány na vytvořených programech. Například lze vypisovat pouze jeden prvek matice, celý její řádek či sloupec a také jen určité části matice.

S maticemi lze v Octave také a dále pracovat . U vektorů i matic jsou podporovány následující operace : sčítání a odčítání, násobení a dělení. Pokud budeme tyto operace provádět mezi skalárem (číslem) a maticí, proběhne výpočet s každým prvkem matice zvlášť. Podobně jsou na tom operace sčítání a odečítání při použití dvou matic, kdy jsou spolu vždy počítány ty prvky, které mají stejné indexy. Násobení a dělení při použití dvou matic však již tak jednoduché není. Pokud použijeme mezi maticemi operátoru `*`, provede se násobení dle platných matematických pravidel. Pokud však ještě před operátor vložíme znak tečky, provede se opět násobení po prvcích. V tomto případě je také nutné, aby byla dodržena pravidla pro násobení matic (druhá matice musí mít stejný počet sloupců jako první řádků) . Obdobně je na tom také dělení. Octave podporuje dva způsoby dělení. První je tzv. levostranné, kdy se jako operátor používá znak `\` a které se používá při dělení matic se stejným počtem řádků. Druhé je pravostranné dělení s použitím operátoru `/`, jenž se využívá při dělení matic se stejným počtem sloupců. [8]

Kromě toho lze s maticí pracovat také pomocí několika dalších příkazů, které jsou uvedeny v tabulce 9.

PŘÍKAZ	VÝZNAM
`inv`	vrací inverzní matici (matice umocněna na -1)
`det`	vrací determinant matice
`rank`	vrací hodnost matice
`rot90`	otočí matici o 90 stupňů (druhý parametr značí směr)

Tab. 9 – Příkazy pro práci s maticemi 2

Důležitou funkci mají v Octave matice při výpočtech nejrůznějších rovnic. Právě pomocí nich lze počítat soustavy lineárních , ale i nelineárních rovnic či kořeny polynomů. S těmi dokáže Octave i další výpočty, jako je jejich násobení a dělení, případně i určení jejich koeficientů podle předem zadaných kořenů. Příklady všech vyjmenovaných možností lze nalézt mezi přiloženými demonstračními programy .

2.3.7 Práce s řetězci

Kromě čísel umí Octave tak jako Matlab pracovat i s řetězci, neboli texty . Lze toho využít například pro vypisování určitých upozornění a povelů. Vše , co budeme chtít, aby Octave považovalo za řetězec, je při zadávání nutné uzavřít do dvojice apostrofů nebo uvozovek. Je pouze na uživateli, kterou formu zadání zvolí. Matlab používá výhradně apostrofy. V dřívějších verzích programu nemohly být některé znaky jako apostrof či uvozovka přímo obsaženy v řetězci. V případě potřeby se tedy reprezentovaly pomocí tzv. „escape sekvencí “ (před daný znak bylo přidáno zpětné lomítko) . Octave verze 3 však některé tyto problémy vyřešilo a dané sekvence nepodporuje.

Řetězce se v Octave stejně jako v Matlabu ukládají ve formě vektorů, kde každý znak odpovídá jednomu prvku tohoto vektoru (lze tedy indexovat) . Díky tomu můžeme také několik takovýchto řetězců spojit – zřetězit. Je-li částí vektoru nějaký řetězec, je potom celý vektor chápán jako řetězec. S touto vlastností nastává v Octave drobný problém, kdy se při určitých formulacích může na výstupu objevit místo požadované hodnoty její ASCII hodnota. Pro tyto případy potom v Octave existuje příkaz `num2str` , který převádí číslo na jeho textovou podobu. Zároveň existuje i opačná funkce `str2num`, která realizuje převod čísla z řetězce do jeho numerické podoby. [8]

Řetězce je možné ukládat nejen pomocí vektorů, ale také pomocí matic. Problémem se může zdát možnost uložení nestejně dlouhých řetězců. Octave však tento problém řeší použitím příkazu `char`, který automaticky doplní znaky za kratší řetězce tak, aby vzniklá matice svými rozměry vyhovovala. Výchozím doplňovacím znakem je mezera, která lze ale nahradit pomocí proměnné `string_fill_char`.

Octave také nabízí funkce pro práci s řetězci. Řetězce lze porovnávat. Pokud použijeme operátoru `==` u dvou stejně dlouhých řetězců, dostaneme vektor logických jedniček a nul, kde 1 značí shodnost znaků na dané pozici. Pokud však budeme chtít zjistit, zda jsou oba řetězce zcela shodné, je na místě použití příkazu `strcmp`. Další nejzákladnější příkazy jsou uvedeny v tabulce 10.

PŘÍKAZ	VÝZNAM
`blanks`	vrací řetězec mezer
`deblank`	odstraní mezery na konci řetězce
`lower , upper`	převede řetězec na malá / velká písmena
`strrep`	nahradí část řetězce jiným
`split`	dělí řetězec na matici podřetězců podle vzoru
`substr`	vrací podřetězec od dané pozice dané délky
`findstr`	vrací vektor vyhovujících pozicí dle zadání
`index , rindex`	vrací první / poslední pozici zadané části v řetězci

Tab. 10 – Příkazy pro práci s řetězci

2.3.8 Práce se soubory , skripty a funkce

Již v první kapitole bakalářské práce jsou uvedeny dva příkazy pro práci s adresářem, ze kterého je Octave spuštěno. Cestu k tomuto adresáři zobrazíme příkazem `pwd`, jeho obsah pomocí `dir`. Příkazem `cd` lze tento pracovní adresář změnit. Dále je možné si v tomto adresáři pomocí `mkdir`, respektive `rmdir` novou složku vytvořit či smazat.

Nejčastějšími příkazy pro práci se soubory jsou příkazy pro přímý přístup k souborům. Pomocí `load` můžeme z textového souboru nahrát matici či vektor hodnot, které můžeme potom dále používat. Obdobně lze také pomocí příkladu `save` proměnné ukládat.

Pokud takto uložený soubor ale otevřeme v nějakém textovém editoru, zjistíme že není příliš čitelný. Octave totiž obsah proměnných v základním nastavení ukládá do binárních souborů. To lze však ovlivnit přidáním vhodného parametru za příkaz (seznam těchto parametrů je obsažen v nápovědě k příkazu `save`) . Vybrat si můžeme také seznam proměnných které se uloží a jméno a typ soubory do kterého se uloží.

Zdrojové soubory, se kterými Octave pracuje, se označují tzv. M-soubory. Pokud v Octave potvrdíme nějaký příkaz, bude tento příkaz v prvé řadě hledat funkce mezi vestavěnými funkcemi. Pokud tam program funkci nenajde, pokusí se hledat v předdefinovaných adresářích zda se zde nenachází stejně pojmenovaný soubor s příponou „.m“ . Pakliže ano, pak provede jeho obsah (očekává se, že bude skript nebo funkce) . Tento princip je převzatý z Matlabu včetně charakterizující přípony *.m* .

V předchozím odstavci jsou zmíněny skripty a funkce. Rozdíl mezi skripty a funkcemi je následující. Skript je sada příkazů uložených v souboru. Tuto sadu příkazů lze také vytvořit přenesením příkazů z příkazové řádky do souboru. Hlavní jeho vlastností je, že může přistupovat k veškerým již zavedeným proměnným, tj. nejenom těm, které jsou vytvořeny přímo v něm, ale také k těm, které byly vytvořeny v příkazové řádce ještě před jeho spuštěním. Zároveň budou veškeré proměnné vytvořené uvnitř něj i nadále přístupné po jeho skončení. Oproti tomu funkce může pracovat pouze s těmi proměnnými, které si zavede ve svém těle nebo deklaruje jako vstupní či výstupní proměnné. Žádné jiné proměnné nejsou ve funkci viditelné a nelze tedy číst jejich hodnoty natož jejich obsah. Funkci od skriptu poznáme také podle toho, že na začátku souboru je uvedeno klíčové slovo `function`, které následuje název samotné funkce, jenž by se měl shodovat s názvem souboru. [1]

2.3.9 Grafický výstup

Mezi podstatné součásti každého programu pro programování matematických výpočtů je grafický výstup. Jedná se především o vykreslování 2D a částečně také 3D grafů. Octave pro vykreslování používá externího programu `gnuplot`. Pro operační systém Linux však existuje několik dalších grafických nadstaveb , které programu dodávají nové vykreslovací funkce. Těmi se však tato práce zabývat nebude, protože je primárně orientována pro uživatele operačního systému Microsoft Windows.

Základní princip kreslení dvourozměrných grafů spočívá v zadání souřadnic bodů v kartézské soustavě, kdy Octave vždy dva sousední body spojí úsečkou. Souřadnice bodů se v takovém případě zadávají jako dva vektory - jeden pro osu x a druhý pro y (oba řádkové nebo oba sloupcové, především stejně dlouhé) . Poté již stačí programu zadat příkaz `plot`, jehož parametry budou oba vektory a nechat si graf vykreslit. Je zcela logické, že čím více prvků budou oba vektory obsahovat, tím přesnější výsledný graf bude.

Někdy je potřeba nechat si do jednoho obrázku vykreslit více grafů na jedné ose. Pokud si necháme vykreslit jeden graf a hned poté druhý, zjistíme že byl první graf překreslen. Jestliže ale před vykreslením druhého grafu uvedeme příkaz `hold` s patřičným parametrem, vykreslí se oba grafy do jednoho obrázku. Stejného výsledku lze dosáhnout i zadáním všech parametrů pro vykreslení do jediného příkazu `plot`. Ten však kromě parametrů vektoru může obsahovat i další parametry, kterými je možné určit typ grafu, jeho barvu, zda se má jednat o bodový graf a také jeho popisek. Jednotlivé možnosti jsou popsány přímo u vzorového příkladu.

Při zobrazení grafu si nelze nevšimnout, že je automaticky nastaveno měřítko a jsou vypsány hodnoty na obou osách. S nimi souvisí příkaz `axis`. Ten může obsahovat více různých parametrů, které přizpůsobují graf a které jsou také všechny popsány ve vzorovém příkladu. Dále lze graf doplnit o titulek (příkaz `title`) , o popis obou os (příkazy `xlabel` a `ylabel`) , o legendu (`legend`) a také o pomocné vodící linky (příkaz `grid`). [2]

Kromě základních 2D grafů jako je například vykreslení goniometrických funkcí Octave ovládá i speciální grafy. První takové jsou schodovité grafy, jenž jsou vykreslovány funkcemi `bar` nebo `stairs`. Určitou obdobou schodovitých grafů jsou histogramy, které je možné zobrazit příkazem `hist` a které slouží například pro zobrazení rozložení náhodných čísel.

Velmi zajímavými funkcemi, které byly převzaty z Matlabu, je možnost zobrazit více grafů (každý na své ose) do jednoho okna, případně vykreslit více grafů najednou, každý ve svém okně. V dřívějších verzích programu bylo vykreslení do jednoho okna učiněno pomocí příkazu `multiplot`. Ten však již ve verzi 3 obsažen není a byl nahrazen „Matlabovským“ příkazem `subplot`, pomocí kterého je možné umístit na stránku až 6 různých grafů. Druhý případ, kdy je pro každý graf otevřeno nové okno, je proveden pomocí standardního příkazu `figure` .

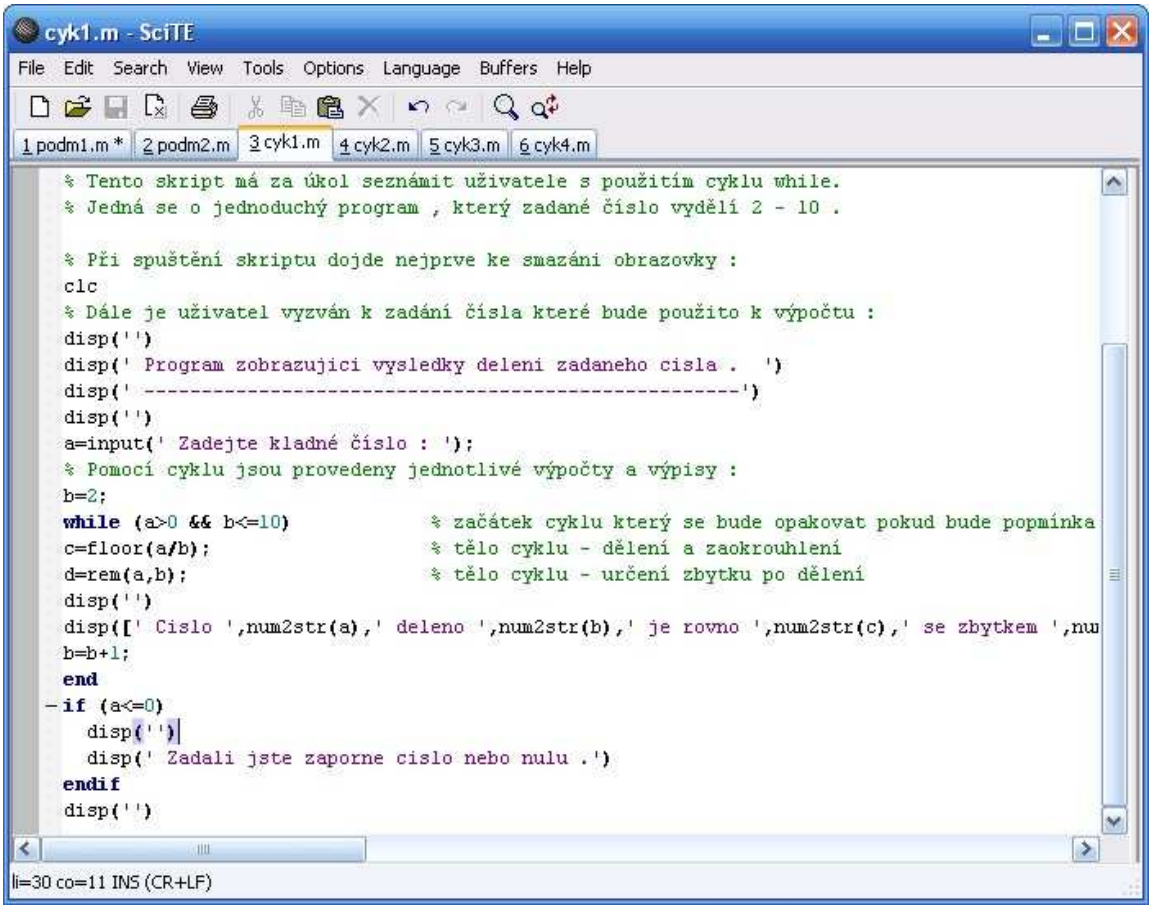
Současné možnosti třírozměrného grafického výstupu v Octave se omezují na jednoduché modely, které mohou reprezentovat plochu funkce dvou proměnných. Zřejmě nejznámějším příkazem je `sombrero`, jenž vytvoří model známý z matematické analýzy. Pro vykreslování v prostoru se dále používá funkce `mesh`. Jejími parametry jsou vektory souřadnic „x“ a „y“ a matice souřadnic „z“ pro všechny průsečíky souřadnic „x“ a „y“. [8]

II. PRAKTICKÁ ČÁST

3 PROGRAMY

První část této práce se věnuje práci v Octave z hlediska teorie. V této části jsou uvedeny vytvořené demonstrační programy, které názorně prezentují použití jednotlivých příkazů. Kromě těchto jednoduchých programů bylo vytvořeno také několik komplexnějších, které používají více příkazů z několika teoretických částí. Jedná se především o příklady podobné těm, které jsou vytvářeny přímo při hodinách předmětu A4PAR .

Všechny tyto programy jsou napsány jako funkce nebo skripty s příponou .m, k jejichž vytvoření bylo použito editoru. Ve starších verzích programu se po zadání příkazu `edit` otevřel poznámkový blok. Ve verzi 3 se však otevře program SciTe, který je přímou součástí instalace. Jedná se o vylepšený textový editor se širokou řadou funkcí a nastavení, který podporuje velké množství programovacích jazyků (kromě Matlab/Octave také Assembler, C, CSS, Java, Pascal či PHP) . Příjemnou vlastností programu je také možnost práce ve více záložkách. [8]



Obr. 7 – Okno editoru SciTE

3.1 Teoretické příklady

3.1.1 Základy

Prvním vytvořeným programem je skript `zaklady` . Ten uživatele seznamuje se základními příkazy, které bude dobré znát při práci s Octave. Jsou v něm shrnuty teoretické kapitoly 2.4.1. až 2.4.3. Jedná se tedy o zobrazování nápovědy a informací o pracovním adresáři, dále o základní práci s proměnnými (určení datového typu, ukládání do souboru, mazání) a také o základní použití podmínek. Všechny operace, které program provádí, jsou okomentovány.

```
Do promenne "a" bude ulozeno cislo , ktere zadate :

Zadejte cislo : 4

Promenna "a" obsahuje cislo 4


Promenna "a" je datovy typ scalar

Zadejte prosim dalsi cislo , ktere bude ulozeno do promenne "b" : 5

Zadana cisla nejsou stejna .

Druhe zadane cislo je vetsi nez prvni .

---------------------------
Pokracujte stisknutim ENTER
```

Obr. 8 – Obrazovka skriptu zaklady

3.1.2 Podmínky a cykly

Další část programů je spjata s teoretickou kapitolou 2.4.4. K podmínkám a cyklům bylo vytvořeno celkem 6 demonstračních příkladů, z nichž 2 shrnují podmínky a 4 popisují funkci cyklů. Všechny ovšem mají společný začátek, kdy je po zadání jména skriptu jenž program obsahuje zcela vymazána obrazovka a uživatel je vyzván k zadání jednoho nebo více čísel. Společným prvkem je také ošetření veškerých podmínek, jenž mohou u jednotlivých úloh nastat.

První skript, který se zabývá cykly, je označen jako `podm1` a seznamuje uživatele s použitím příkazu `if` . Po zadání konkrétního čísla z předem určeného intervalu na obrazovku vypíše, z kterého subintervalu dané číslo je. Druhý program je pojmenován jako `podm2` a představuje použití příkazu `switch`. Uživatel si v něm může z nabídky jazyků vybrat ten, kterým chce být od počítače pozdraven.

Následují programy věnující se jednotlivým cyklům. `Cyk1` názorně ukazuje použití příkazu `while`, neboli cyklu s podmínkou na začátku. Uživatel je vyzván k zadání kladného čísla, které je následně programem děleno čísly 2 až 10. Výstupem programu je 9 řádků oznamujících výsledky jednotlivých početních operací včetně zbytků po dělení. Oproti tomu `cyk2` využívá cyklu s podmínkou na konci – příkazu `do/until`. Funkce programu je jednoduchá – číslo které uživatel zadá vypíše na obrazovku pozpátku. Následuje `cyk3` který pracuje rovnou se třemi ciframi zadanými uživatelem. Program vypisuje čísla ze zadaného intervalu se zvoleným krokem. Jelikož je jejich počet programu znám ještě před samotným výpočtem, jedná se o typické využití příkazu `for` . Posledním ukázkovým programem u cyklů je `cyk4` . Ten vysvětluje použití příkazů `continue` a `break`. Vstupem je u něj číslo z intervalu 0 až 99. Následně jsou vypsána všechna čísla , která jsou větší než zadané číslo a zároveň jsou dělitelná pěti, a to až po cifru 100.

```
Program zobrazujici vysledky deleni zadaneho cisla .
--------------------------------------------------

Zadejte kladné číslo : 273
Cislo 273 deleno 2 je rovno 136 se zbytkem 1
Cislo 273 deleno 3 je rovno 91 se zbytkem 0
Cislo 273 deleno 4 je rovno 68 se zbytkem 1
Cislo 273 deleno 5 je rovno 54 se zbytkem 3
Cislo 273 deleno 6 je rovno 45 se zbytkem 3
Cislo 273 deleno 7 je rovno 39 se zbytkem 0
Cislo 273 deleno 8 je rovno 34 se zbytkem 1
Cislo 273 deleno 9 je rovno 30 se zbytkem 3
Cislo 273 deleno 10 je rovno 27 se zbytkem 3
```

Obr. 9 – Obrazovka skriptu cyk1

3.1.3 Práce s čísly

Pro tuto část byl vytvořen skript pojmenovaný `matematika`. Ten názorně vysvětluje použití všech numerických funkcí, které jsou uvedeny v teoretickém celku 2.4.5. Celkově lze program rozdělit na 3 části – použití základních operátorů, aplikace matematických funkcí a práce s komplexními čísly .

V první je uživatel vyzván k zadání 2 čísel. Ty jsou následně sečteny, odečteny, vynásobeny, vyděleny i umocněny. Dále je pomocí interních příkazů určen jejich nejvyšší společný dělitel i nejmenší společný násobek. Výsledky všech výpočtů jsou vypsány na obrazovku. Další část se objeví stisknutím klávesy `enter`. V té jsou předem připraveny 3 proměnné : ±2 a 0. Nejprve jsou pro kladné číslo vypočteny hodnoty druhé odmocniny, přirozeného a dekadického logaritmu a také exponentu, který je dále použit při demonstraci všech funkcí pro zaokrouhlování (k vyššímu / nižšímu číslu , dle matematických pravidel) . Dále je na záporném čísle naznačeno použití funkce vracející absolutní hodnotu čísla. Poslední tři příklady druhé části se věnují příkazu `sign`, který určuje zda jde o číslo kladné, záporné či nulu. Po dalším stisku klávesy `enter` dojde ke smazání obrazovky a zobrazení poslední části skriptu, která se věnuje práci s komplexními čísly. Ta jsou uvažována ve tvaru `a+bi`, přičemž oba koeficienty si uživatel volí sám. Následně je číslo vypsáno na obrazovku, společně s jeho hodnotou, argumentem i číslem komplexně sdruženým.

```
Do promenne "a" bude ulozeno cislo , ktere zadate :

Zadejte cislo : 4

Promenna "a" obsahuje cislo 4


Promenna "a" je datovy typ scalar


Zadejte prosim dalsi cislo , ktere bude ulozeno do promenne "b" : 5

Zadana cisla nejsou stejna .

Druhe zadane cislo je vetsi nez prvni .

--------------------------
Pokracujte stisknutim ENTER
```

Obr. 10 – Obrazovka skriptu matematika

3.1.4 Matice a vektory

V teoretické části bakalářské práce je uvedeno, že Octave je stejně jako Matlab postaveno především na maticích. Z tohoto důvodu byl vytvořen program, který pokrývá práci s maticemi. Byly vytvořeny celkem 3 skripty s pojmenováním `matice1` , `matice2` a `matice3`. Každý z nich je dále rozdělen na několik obrazovek, jenž se zobrazují stiskem klávesy `enter` .

První ze skriptů popisuje jakými způsoby dochází k plnění matic a vektorů čísly. Nejprve je vysvětleno zapsání samotných čísel do matic, včetně použití mezer, čárek a středníků mezi jednotlivými prvky. Zároveň jsou vytvořeny matice pomocí dvojtečkové metody, příkazem linspace, či složením dvou menších matic do jedné větší. Druhá obrazovka skriptu se věnuje vytváření speciálních typů matic. Jedná se především o ty, které mají čísla pouze na hlavní diagonále (jednotková a diagonální) , které obsahují samé jedničky nebo nuly, a také takové, které mají extrahovanou horní nebo dolní trojúhelníkovou oblast. Třetí a poslední obrazovka prezentuje způsob indexování v maticích (zobrazení jen určitých prvků) a použití základních příkazů, které zobrazují informace o jednotlivých maticích. Je tak možné zjistit jejich celkovou velikost, počet jejich řádků a sloupců, jejich determinant či hodnost.

Druhý skript se věnuje základním operacím s maticemi. Dopředu jsou vytvořeny a naplněny dvě čtvercové, jenž dále slouží k demonstraci jednotlivých operací. Na první obrazovce jsou k nim ještě pomocí apostrofu určeny matice transponované (řádky jsou zaměněny za sloupce a naopak) . Druhá obrazovka se věnuje úkonům mezi skalárem a maticí (násobení a umocnění matice číslem po prvcích) a také sčítání a odečítání dvou matic. Třetí, zabývající se násobením matic, je rozdělena do dvou skupin – na násobení podle matematických pravidel a po prvcích. V závěrečné části jsou popsány možnosti u dělení matic. To je možné rozčlenit na 4 kategorie. Matice je totiž možné dělit levostranně i pravostranně a navíc podle matematických pravidel či po prvcích .

Poslední skript je zaměřen na použití matic a vektorů při počítání s různými typy rovnic. V prvním případě se jedná o soustavu lineárních rovnic, jejichž obecný tvar je `ax + by + cz = d`. Úkolem je určení kořenů `x,y,z` těchto rovnic. Při řešení se zapíší všechny koeficienty z levé strany rovnice do matice a všechny koeficienty z pravé strany do sloupcového vektoru. Výsledkem je taktéž sloupcový vektor hodnot, který lze

určit použitím několika podmínek a Crammerova pravidla, popřípadě pomocí levostranného dělení. Druhá obrazovka se věnuje práci s polynomy, jejichž obecný tvar je $ax^N + bx^{N-1} + cx^{N-2} + \ldots. + dx^0 = 0$. Základem výpočtu je přepsání všech koeficientů rovnice (včetně těch nulových) do vektoru, z něhož lze příkazem `roots` jednotlivé kořeny rovnice určit. Třetí část skriptu se taktéž věnuje práci s polynomy, avšak tentokrát je postup opačný – ze známých kořenů je potřeba vytvořit rovnici. K tomu slouží příkaz `poly`, který z kořenů zapsaných ve vektoru určí všechny koeficienty hledané rovnice. Posledním demonstračním programem skriptu je roznásobení dvou polynomů pomocí příkazu `conv`.

Je nutné dodat, že ukázky uvedené ve třetím skriptu čerpají z příkladů , které jsou uvedeny na webové stránce `www.abclinuxu.cz` a částečně také ze starších testových příkladů používaných při výuce v předmětu A4PAR.

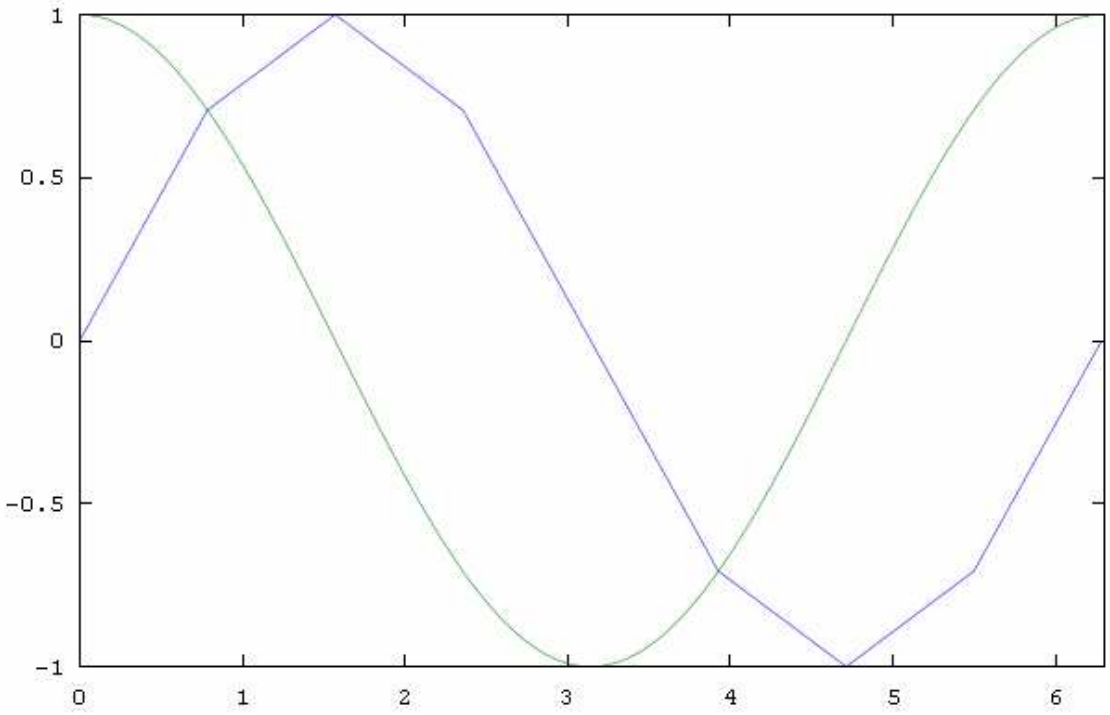
```
Reseni linearnich rovnic

----------------------------------------
Zadani :
----------------------------------------
Urcete hodnoty x,y,z v nasledujicich rovnicich
6x + 2y + 8z = 10
28x + 4y + 2z = 20
x + 6y + 10z = 30
----------------------------------------
Vypracovani :
----------------------------------------

Nejprve je potreba si rovnice prepsat do 2 matic :

Matice "a" obsahuje koeficienty rovnic :

  6.00   2.00   8.00
 28.00   4.00   2.00
  1.00   6.00  10.00

Matice "b" obsahuje vysledky rovnic :

 10.00
 20.00
 30.00

----------------------------------------

K vysledkum se lze dostat pouzitim Cramerova pravidla :

 0.00  5.00  0.00

Nebo take pomoci levostranneho deleni matic :

 -0.00
 5.00
 0.00
```

Obr. 11 – Obrazovka skriptu matice3

3.1.5 Grafické zobrazení

Doposud byly všechny programy napsány tak, že jejich výstupem byla čísla nebo text. Někdy však toto nestačí a je potřeba, aby výstupem byl graf zobrazující kupříkladu průběh funkce. Za tímto účelem vznikly čtyři skripty zabývající se právě prací s grafy, a to jak dvourozměrnými, tak i trojrozměrnými .

Prvním ze skriptů je `graf1`, který seznamuje uživatele s tvorbou jednoduchých i složitějších grafů. Základem je vykreslení z hodnot, jenž jsou uloženy ve vektoru. Následují zobrazení průchodů goniometrických funkcí. Ve skriptu jsou uvedeny funkce sinus a cosinus, pro které byly vstupní data vytvořena dvojtečkovou metodou. Pro první funkci však bylo určeno pouze 9 čísel, zatímco pro druhou celých 200. Tímto způsobem je demonstrována přesnost vykreslení. Pro obě funkce jsou vytvořeny nejprve samostatné grafy a poté jeden společný, na kterém lze oba průchody porovnat. Poslední částí je seznámení s tvorbou speciálních grafů . Jedná se o grafy sloupcovitý a schodový, o histogram, o graf vytvořený z polárních souřadnic a také o graf zobrazující chybové úsečky.

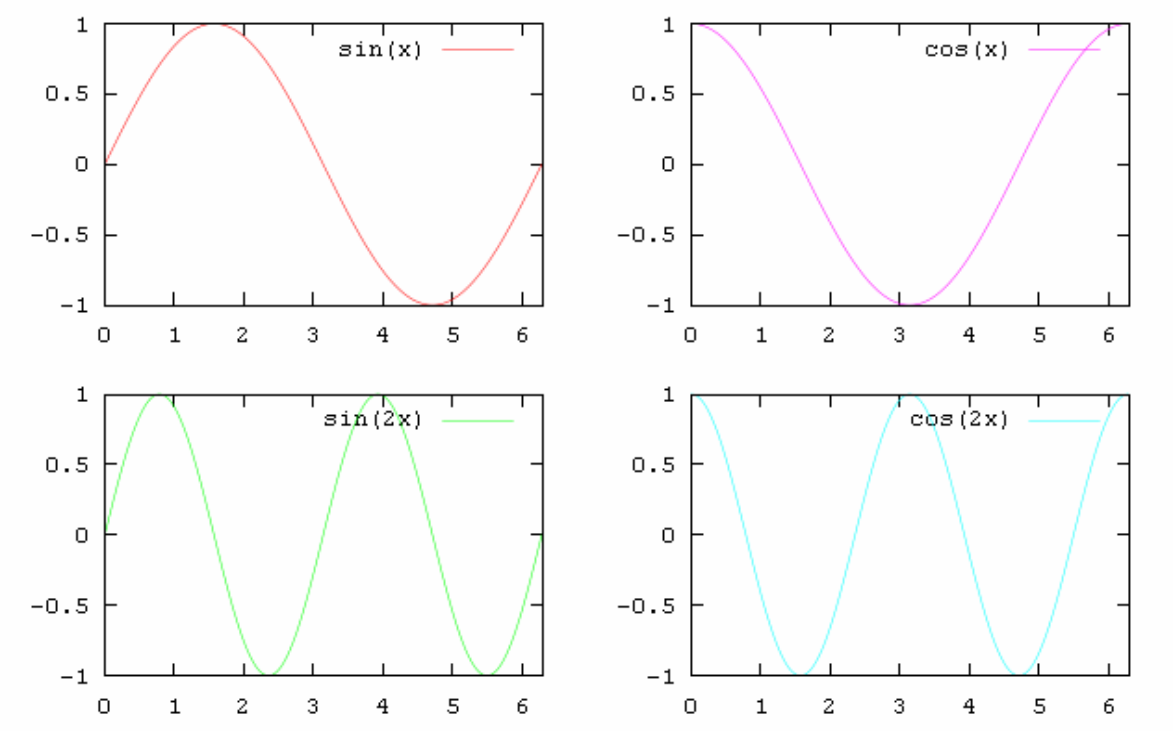


Obr. 12 – Porovnání průběhů funkcí při různém počtu vykreslovaných bodů

Všechny grafy , které jsou vytvořeny v prvním programu lze dále upravit pomocí příkazů uvedených ve skriptu `graf2`. Jeho první část se věnuje samotnému grafu. Na jednotlivých příkladech ukazuje jak lze ovlivnit typ grafu (zda má být čárový či pouze tečkovaný) , jak nastavit jeho barvu, nebo jak vytvořit graf bodový (včetně možnosti zobrazení linek mezi jednotlivými body) . Taktéž je zde uvedeno, jakými způsoby lze ke grafu vytvořit jeho legendu. Ve druhé části je popsána práce s osami grafu. Je možno určit jejich rozmezí i jejich přizpůsobení samotnému grafu. Ovlivnit lze také zobrazení jednotlivých bodů a hodnot na osách a dále také umístění vyšších hodnot na ose y.

`Graf3` se věnuje poměrně časté situaci u grafů a to vykreslení jejich většího množství současně. Tak jako v Matlabu, tak i v Octave jsou zde jsou dvě možnosti. První z nich je vykreslení všech grafů formou matice umístěné v jednom okně pomocí příkazu `subplot`. Ve druhém příkladě je pomocí příkazu `figure` vykreslen každý graf zvlášť v samostatném okně.

Poslední skript `graf4` se věnuje základům vytváření grafů v prostoru. Jsou uvedeny 2 příklady, které demonstrují použití příkazů `mesh` / `meshgrid` a jeden příklad, jehož výstupem je graf znázorňující `sombrero`, což je graf funkce používané v matematické analýze.



Obr. 13 – Zobrazení grafů příkazem subplot

3.1.6 Porovnání syntaxe s Matlabem

Všechny výše vytvořené a uvedené skripty byly spuštěny také v prostředí programu Matlab. Vzhledem k velké kompatibilnosti obou programů nebylo problémem většinu skriptů pustit. V několika případech však bylo nutné ke správné funkci syntaxi kódu lehce změnit. Hlavní rozdíly vytvořených programů oproti programům spustitelným v Matlabu tedy jsou :

- použité ukončování podmínek a cyklů

  - Octave umožňuje ukončování pomocí `end`, ale i `endif`, `endwhile`, `endfor` …

  - Matlab používá výhradně `end`

- příkaz `typeinfo`

  - Octave umožňuje vypsání všech podporovaných datových stylů

- použití inkrementace a dekrementace

  - Octave po vzoru C používá zkrácený operátor `++` a `--`

- příkazy `rows` a `columns`

  - v Octave lze samostatně určit počet sloupců a řádků matice

  - Matlab používá pouze příkaz `size` , ze kterého lze hodnoty extrahovat

- histogram

  - v Octave lze použít více parametrů při vytváření tohoto grafu

- zobrazení legendy u grafů

  - Octave umožňuje legendu zapsat také jako třetí parametr

  - Matlab používá pouze příkaz `legend`

- zadání barvy grafu pomocí čísel

  - v Octave lze pro zadání použít kromě písmen i čísla

  - Matlab podporuje pouze zadání prvními písmeny z anglických názvů

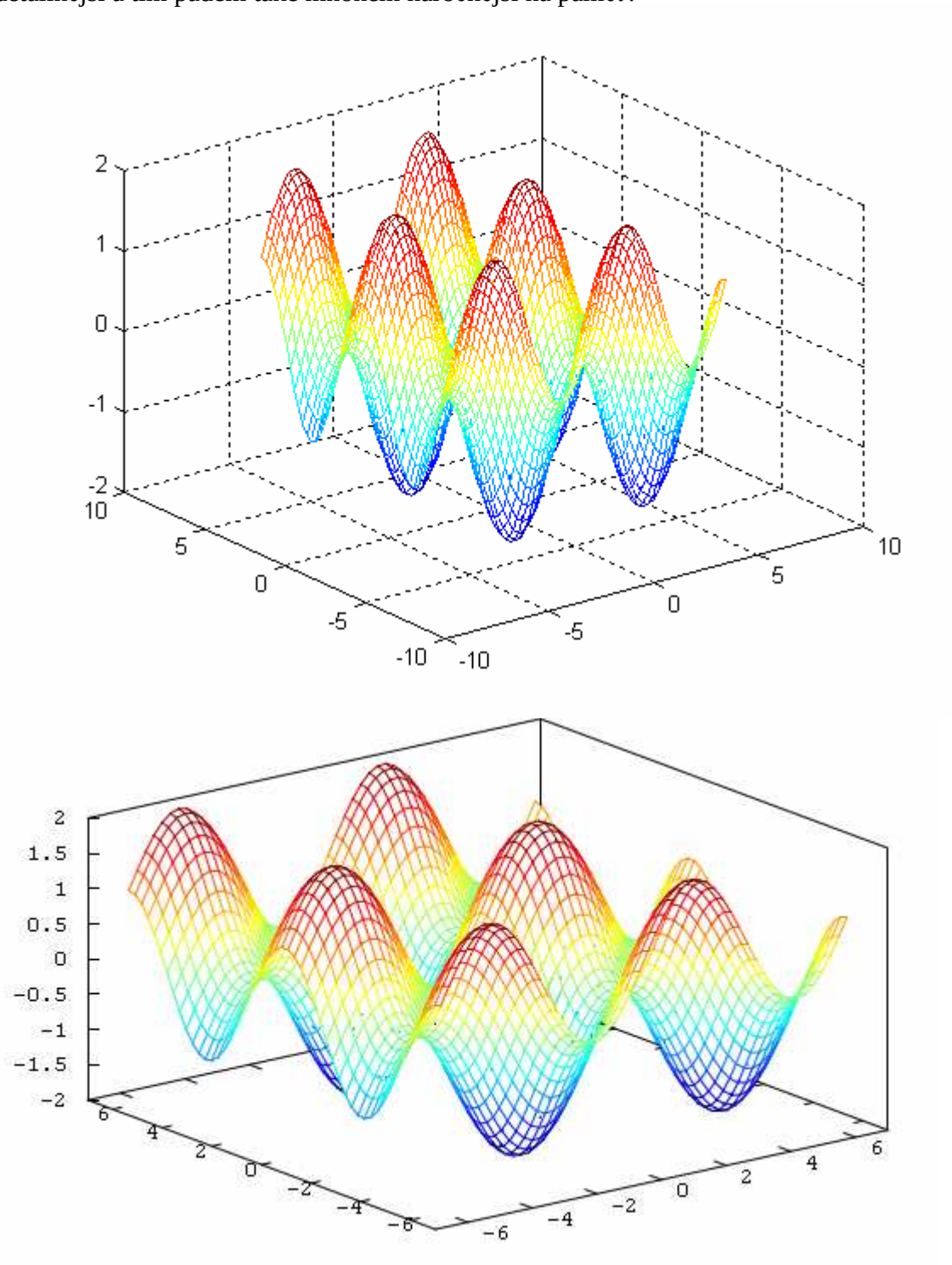
- parametry `ticx/y` a `labelx/y` u definic os grafu

  - v Octave je možnost nezobrazit body a hodnoty na obou osách

- graf sombrero

  - Matlab nemá integrovány podobné vykreslovací funkce

Kromě těchto odlišností v syntaxi byly zjištěny také drobné rozdíly při vykreslování grafů. V Octave jsou 2D grafy vykreslovány v jasnějších barvách a pro vykreslení používají antialiasingu, avšak grafy v Matlabu nabízejí již v základním tvaru lepší popsání obou os a možnost přesouvání legendy do vhodnější polohy. Prostorové grafy z Octave jsou mnohem detailnější a tím pádem také mnohem náročnější na paměť.



Obr. 14 – Porovnání stejných 3D grafů v Matlabu (vrchní) a Octave (spodní)

3.2 Komplexní příklady

3.2.1 Práce s čísly

První dva komplexní programy se ještě zcela zaměřují na základní počítání s čísly. Oba byly vytvořeny jak ve formě funkce, tak skriptu. `Soucet` (`soucets`) je tedy jednoduchý program počítající pomocí podmínky `while` součet všech čísel v zadaném intervalu. Druhý program – `tri` (`tris`) , používá ve svém těle několik podmínek `if` a jeho úkolem je určení minima a maxima ze tří zadaných čísel.

```
Zadejte 3 cisla :
----------------------------------------
Cislo 1 : 3
Cislo 2 : 4
Cislo 3 : 5

Nejmensi cislo ze zadanych je 3
Nejvetsi cislo ze zadanych je 5
```

Obr. 15 – Obrazovka skriptu tris

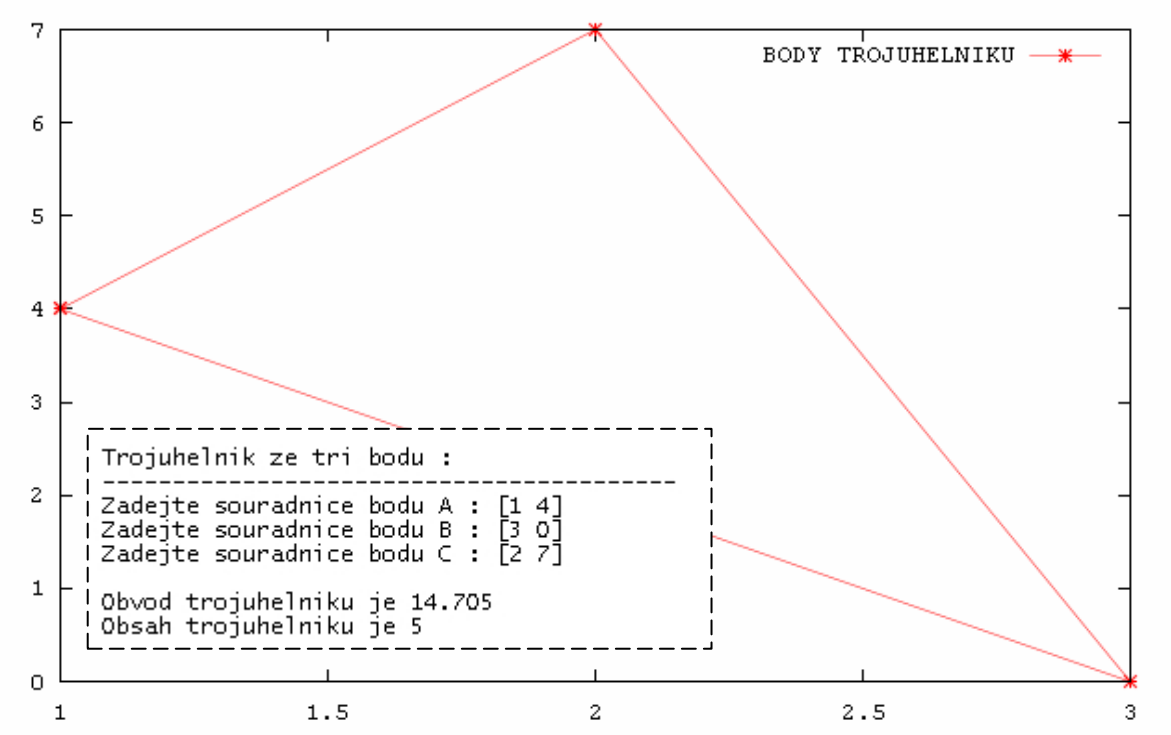
3.2.2 Planimetrie

Planimetrie je část geometrie, která se studuje rovinné útvary. Ve skriptu `telesa`, který počítá obvod a obsah jsou nabídnuty čtyři takové útvary – trojúhelník, čtverec, obdélník a kruh. Po vybrání následuje zadání všech potřebných údajů k výpočtu, jehož výsledky jsou ihned zobrazeny. Pro jednotlivé útvary je možné také použít vytvořené funkce, která vždy nese jméno daného útvaru.

```
Ktere teleso Vas zajima ?
---------------------------------
[1] - Trojuhelnik
[2] - Ctverec
[3] - Obdelnik
[4] - Kruh
---------------------------------
Vase volba : 4
---------------------------------
Polomer kruhu r : 12

Obvod kruhu je 75.398 jednotek .
Obsah kruhu je 452.39 jednotek ctverecnych .
```

Obr. 16 – Obrazovka skriptu telesa

S tělesy souvisí ještě jeden program. Funkce `troj` (respektive skript `trojs`) pracuje s trojúhelníkem, který je zadán pomocí tří bodů. Každý takový bod je tvořen z dvouprvkového vektoru, jenž obsahuje hodnoty na osách x a y. Program pomocí Pythagorovy věty z těchto hodnot vypočte velikosti jednotlivých stran a v případě, že se skutečně jedná o trojúhelník určí také jeho obsah a obvod. Nakonec celý trojúhelník vykreslí v bodovém grafu.



Obr. 17 – Obrazovka a výsledný graf skriptu trojs

3.2.3 Kvadratické rovnice

Častým úkolem při výuce programování bývá vytvoření programu pro výpočet kořenů kvadratické rovnice. Vstupem do funkce `kvadraticka` (`kvadratickas`) tak jsou tři čísla symbolizující koeficienty takové rovnice . V programu je pomocí několika podmínek `if` určeno které ze zadaných koeficientů jsou rovny nule a kolik kořenů bude zadaná rovnice mít. Ty jsou následně vypočteny a vypsány na obrazovku.

```
octave.exe:10> kvadraticka(-1,-4,2)

 Prvni koren - x1 = -4.4495
 Druhy koren - x2 = 0.44949
```

Obr. 18 – Obrazovka funkce kvadraticka

3.2.4 Fyzikální výpočty

Numerické výpočty se však neobjevují pouze v matematice. Velmi podobným odvětvím je fyzika a proto i zde lze Octave uplatnit pro počítání.

První funkce a skript nesou pojmenování `sekundy` (`sekundys`) . Jejich hlavním úkolem je demonstrace použití několika podmínek použitých při přepočítávání času. Vstupem programu je doba zapsaná v sekundách, přičemž na výstupu uživatel zjistí kolik dní, hodin, minut a sekund daná doba vlastně je.

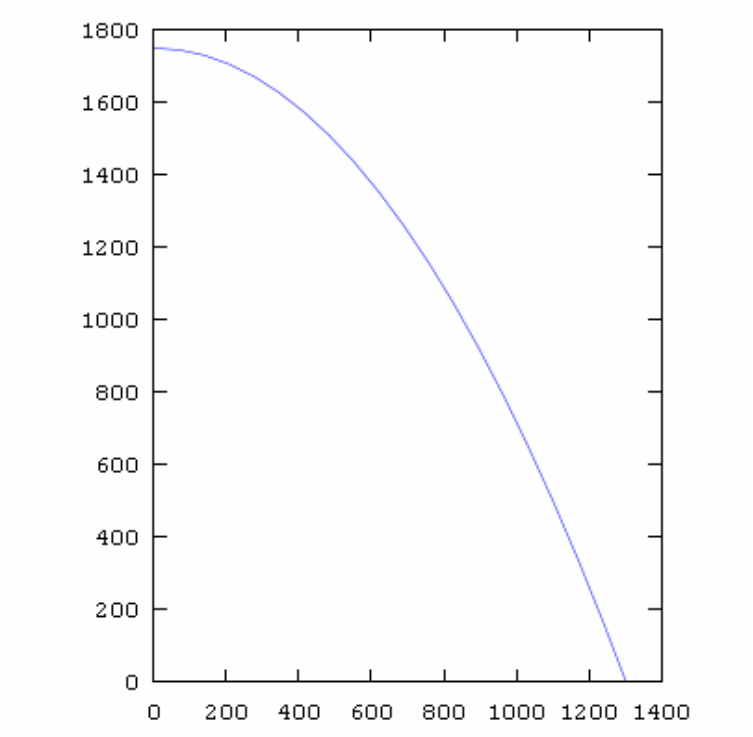
```
Zadejte cas v sekundach :
----------------------------------------
Cas : 34525

Zadany cas je : 9 hodin , 35 minut a 25 sekund .
```

Obr. 19 – Obrazovka skriptu sekundy

Další dva programy se věnují výpočtům s gravitačním polem Země. Jedná se o skripty `vrhvodorovny` a `vrhsikmy`, jejichž názvy popisují i jejich funkce. Pro vodorovný vrh je vstupem výška, ze které je těleso hozeno a rychlost se kterou je hozeno. Následně jsou pomocí několika výpočtů určeny dva vektory obsahující souřadnice bodů v osách x a y v jednotlivých časech t. Výstupem je délka a doba vrhu. Z vektoru hodnot je dále vykreslen graf znázorňující křivku hodu, u kterého jsou pro větší názornost nastaveny na obou osách stejná měřítka :

```
while(y(i)>=0)
  x(i+1)=v*t;
  y(i+1)=vyska-0.5*g*t*t;
  i++;t++;
endwhile
y(i)=0;
cas=sqrt(2*vyska/g);
x(i)=v*cas;
disp([' Teleso dopadne do vzdalenosti ',num2str(x(i)),' metru za dobu ',num2str(cas),' sekund .']),disp('')
plot(x,y)
axis('equal')
```

```
Vypocet delky vodorovneho vrhu :
----------------------------------------
Zadejte vysku odkud je teleso vrzeno [m] : 1749
Zadejte rychlost se kterou je teleso vrzeno [m/s] : 69

----------------------------------------

Teleso dopadne do vzdalenosti 1303.2 metru za dobu 18.886 sekund .
```



Obr. 20 – Skript vrhvodorovny

Vstupem druhého programu je úhel, pod jakým bylo těleso vyhozeno a také rychlost s jakou bylo vyhozeno. Jelikož je úhel zadáván ve stupních, je nutné jej nejprve převést na radiány, se kterými je v Octave počítáno u goniometrických funkcí :

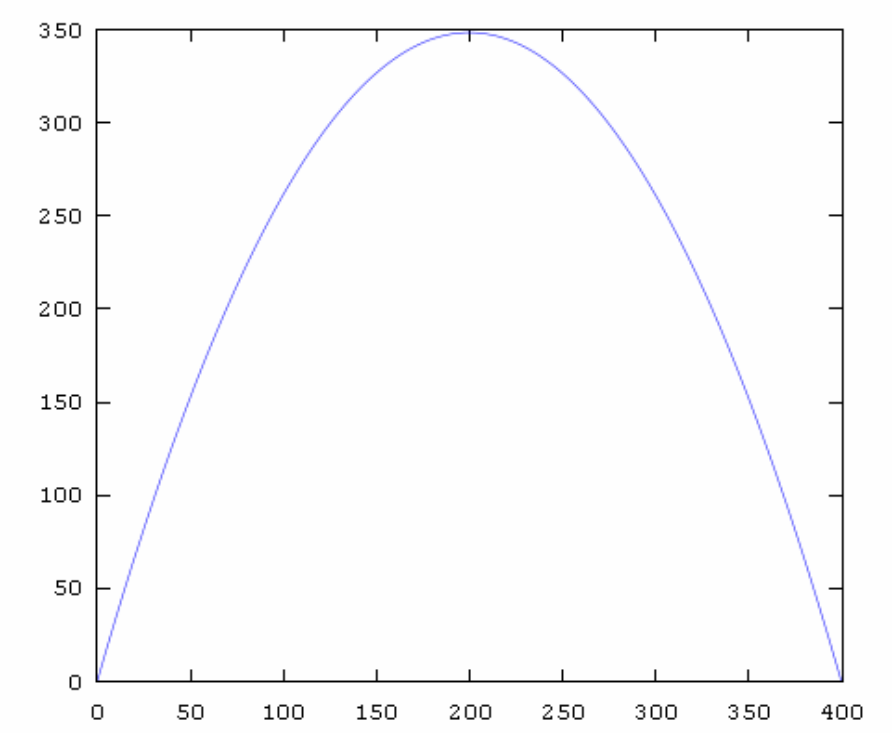
*radian=uhel*pi/180;*

Následně jsou opět vypočteny dva vektory hodnot udávající souřadnice tělesa v danou dobu hodu .

```
while(y(i)>=0)
  x(i+1)=v*t*cos(radian);
  y(i+1)=v*t*sin(radian)-0.5*g*t*t;
  i++;
  t=t+0.1;
endwhile
```

Posledním krokem je jako u předešlého příkladu výpis na obrazovku. Tentokrát je vypsána délka vrhu a také maximální výška, které bylo během vrhu dosaženo. Zároveň je vykreslen graf znázorňující celý průběh vrhu.

```
Vypocet delky sikmeho vrhu vzhuru :
-----------------------------------------
Zadejte uhel pod jakym je teleso vrzeno [1°-89°] : 74
Zadejte rychlost se kterou je teleso vrzeno [m/s] : 86

-----------------------------------------

Teleso dopadne do vzdalenosti 399.66 metru , pricemz dosahne maximalni vysky 348.44 metru .
```



Obr. 21 – Skript vrhsikmy

..

3.2.5 Aproximace hodnot

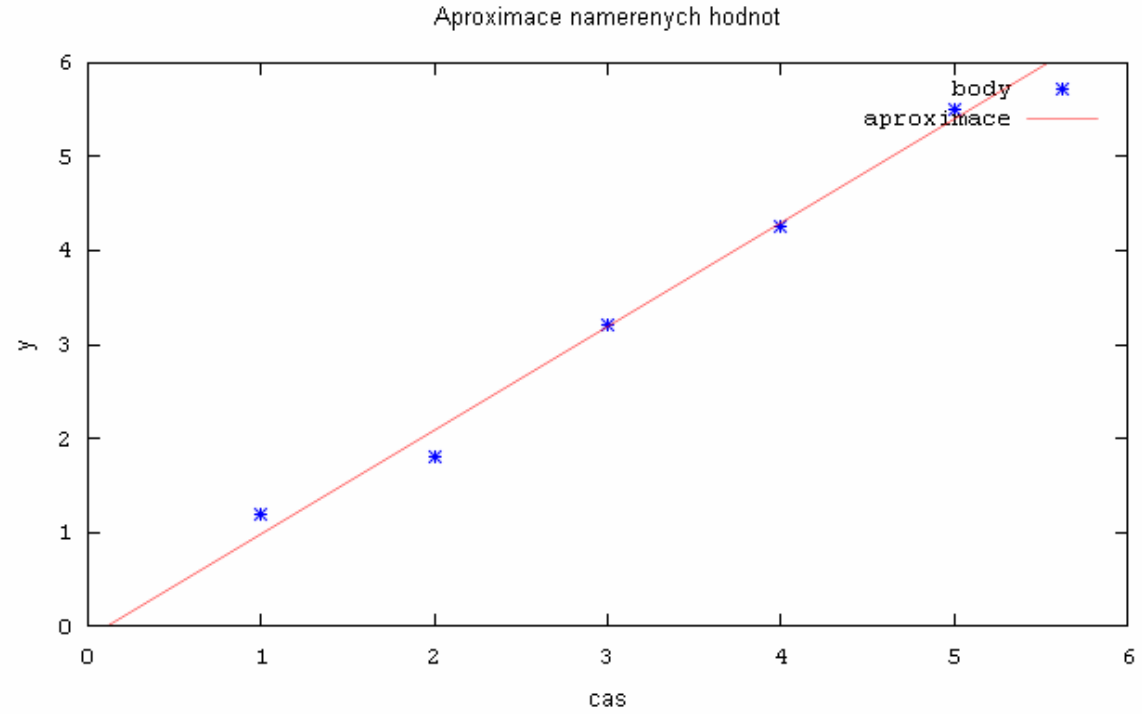
Součástí některých procvičovacích testů z předmětu A4PAR jsou také příklady s podobným zadáním, jako je toto :

> Je dána tabulka naměřených hodnot. Proložte naměřené body polynomem 1.stupně. Do grafu vykreslete naměřené body a přímku, která je aproximuje ve smyslu nejmenších čtverců.
>
> - naměřené hodnoty budou v grafu zobrazeny modrou hvězdičkou
> - aproximace bude zobrazena červenou čárkovanou čárou
> - do grafu také uveďte legendu , popis os a nadpis grafu

ČAS [S]	1	2	3	4	5
y	1.20	1.80	3.20	4.25	5.50

Tab. 11 – Hodnoty pro ukázkový příklad

Řešení je popsáno ve skriptu `aproximace`. Zadané hodnoty je nejprve nutno přepsat do dvou samostatných vektorů. Ty jsou následně použity při regresi dat polynomem prvního stupně příkazem `polyfit`. Pomocí příkazu `polyval` lze posléze vzniklý polynom vyhodnotit a vykreslit z něj graf.



Obr. 22 – Vyhodnocení skriptu aproximace

3.2.6 Uživatelské funkce

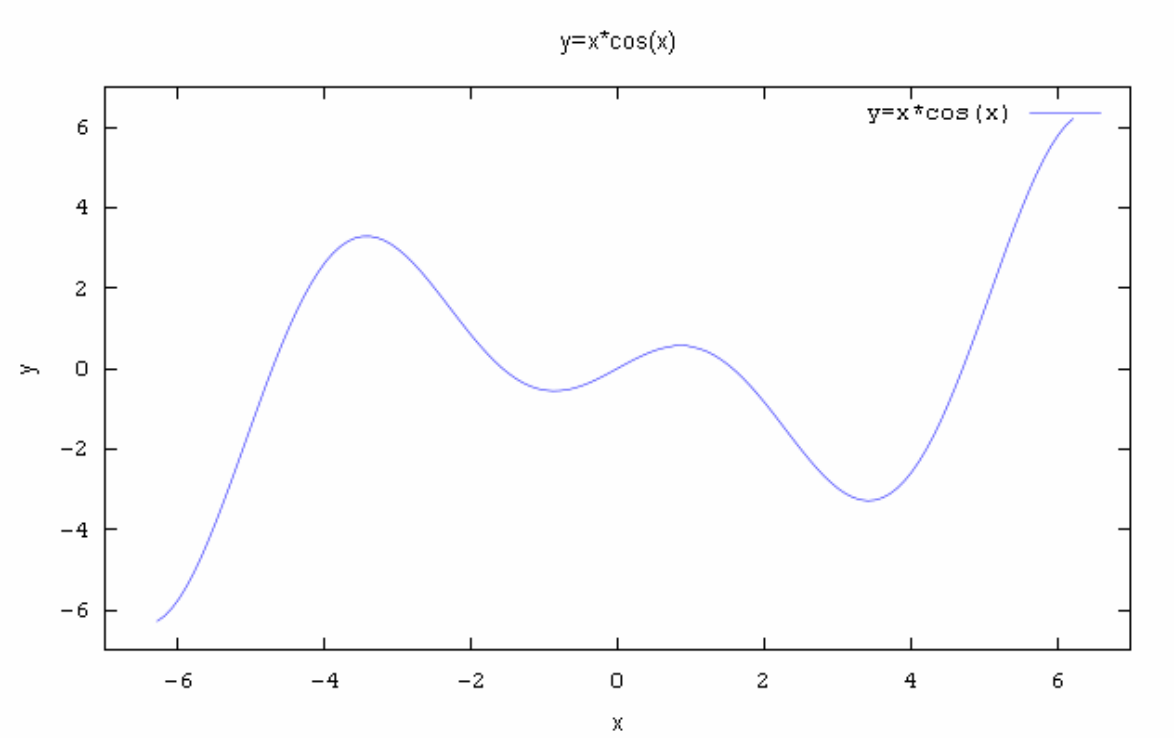
Dalšími příklady z hodin A4PAR jsou ty, které mají za úkol vypočítat hodnotu určité funkce a následně její průběh vykreslit na nějakém intervalu. Ukázkou takových příkladů je skript `funkce`, který obsahuje celkem 4 různé funkce :

- $y = (\exp(x) + \exp(-x)) / 2$
- $y = x \cdot \cos(x)$
- $y = \sin(x)^2$
- $y = (\sin(x) + \cos(x))^2$

Pomocí příkazu `switch` si uživatel může sám některou z funkcí vybrat. Ta je následně interně vyhodnocena a za získaných hodnot je vykreslen graf :

```
disp(' Kterou funkci chcete vykreslit ?'),
disp(' [1] - y = ( exp(x) + exp(-x) ) / 2')
disp(' [2] - y = x * cos(x)')
disp(' [3] - y = sin(x)^2')
disp(' [4] - y = ( sin(x) + cos(x) )^2')
X=input(' Vase volba : ');
switch X
case 1
     x=0:0.1:10;
    plot(x,(exp(x)+exp(-x))/2)
    title('y=(exp(x)+exp(-x))/2;');
    legend('y=(exp(x)+exp(-x))/2;');
     xlabel('x');ylabel('y');
case 2
    x=-2*pi:0.1:2*pi;
    fce=x.*cos(x);
    plot(x,fce);
     title('y=x*cos(x)');
     legend('y=x*cos(x)');
    xlabel('x');ylabel('y');
```

```
case 3
        x=0:0.1:2*pi;
        plot(x,sin(x).^2);
        title('y=sin(x)^2');
        legend('y=sin(x)^2');
        xlabel('x');ylabel('x');
case 4
        x=0:0.1:2*pi;
        plot(x,(sin(x)+cos(x)).^2);
        title('y=(sin(x)+cos(x))^2');
        legend('y=(sin(x)+cos(x))^2');
        xlabel('x');ylabel('y');
otherwise
        disp(' Volba mimo !')
end
```



Obr. 23 – Graf funkce x · cos(x) ze skriptu funkce

3.2.7 Funkce funkcí

Taktéž příklady uvedené v této sekci jsou původem z hodin A4PAR. Nejprve se jedná o hledání řešení zadaných rovnic v následujícím tvaru :

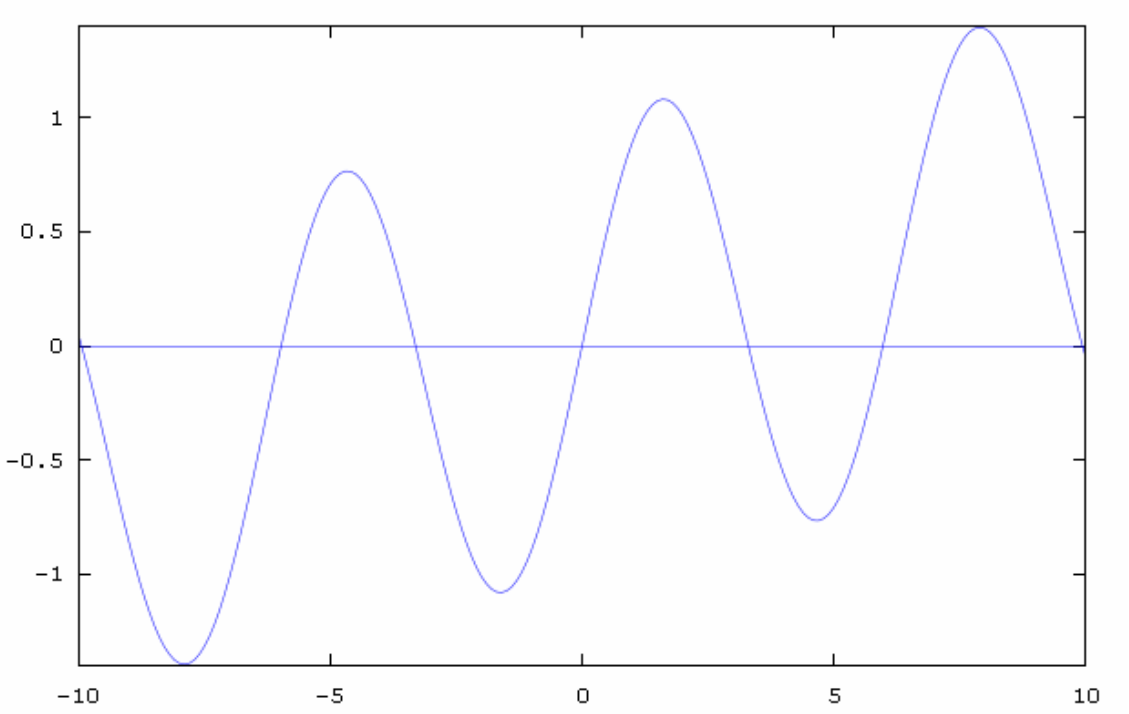
- sin(x) + x/20 = 0

Stejně jako v Matlabu je potřeba si vytvořit vždy 2 samostatné soubory – jeden který obsahuje zadanou funkci a druhý obsahující její výpočet, případně její vykreslení. První soubor pro druhou zadanou funkci proto vypadá následovně :

```
function y=fce(x)
y=sin(x)+x/20;
```

Druhý soubor `fce2`, potom jméno toho prvního používá jako parametr při vyhodnocování i vykreslování . Pro určení řešení je použito příkazu `fzero`, jehož druhým parametrem může být hrubý odhad výsledku :

```
x=-10:0.1:10;
plot(x,fce(x));
x1=fzero('fce',-9)
x2=fzero('fce',-6)
x3=fzero('fce',-4)
```

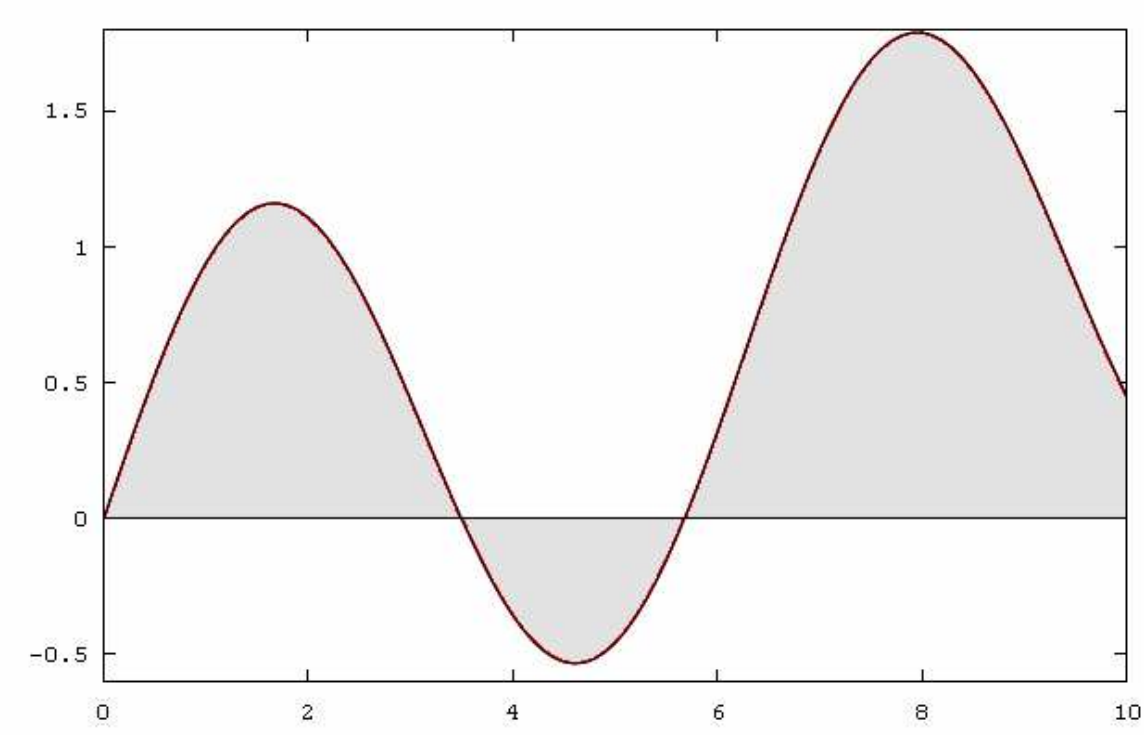


Obr. 24 – Graf skriptu fce2

Funkce funkcí lze také využít při výpočtech určitých integrálů, kdy první ze souborů opět obsahuje zadanou funkci. Jeho jméno je posléze použito ve druhém souboru pro vykreslení, ale také pro výpočet samotného určitého integrálu funkcí `quad`, jejíž další parametry integrál ohraničují :

```
quad('fcei',0,10)
x=0:0.1:10;
plot(x,fcei(x));
hold on;
area(x,fcei(x))
```

Poslední řádek kódu obsahuje příkaz `area`, který zajišťuje zvýraznění ohraničeného určitého integrálu ve grafu.



Obr. 25 – Graf skriptu fcei2

ZÁVĚR

Hlavním cílem této práce bylo vytvoření programů v Octave, které pokrývají celou výuka základů programování předmětu Programová podpora automatického řízení vyučovaného na Fakultě aplikované informatiky ve Zlíně. Tento předmět je prozatím zaměřen pouze na integrované prostředí Matlab.

Jedním z programů, které by jej ve výuce mohly doplnit je Octave, jehož poslední verze nese označení 3. Ta oproti verzím minulým doznala mnoha změn a vylepšení. Především se nejedná o software určený pouze pro Linux, jak se mnoho uživatelů domnívá. Zcela bez problémů jej lze používat také na počítačích vybavených operačním systémem MacOS či MS Windows. Verze 3 je více stabilnější, obsahuje několik nových příkazů a funkcí, umožňuje instalovat více doplňků. Také díky tomu se opět o něco více Octave přiblížilo Matlabu.

V první části této bakalářské práce jsou nejprve popsány okruhy výuky předmětu Programová podpora automatického řízení, u kterých je možné kromě Matlabu použít také programu Octave . Tomu se také věnuje zbytek první části . Nejprve je popsána jeho historie a jednotlivé části jeho instalace . Teorii uzavírá seznam a popis základních příkazů potřebných pro práci v programu .

Druhá část práce je věnována vytvořeným programům, které jsou rozděleny do dvou skupin. První demonstrují příkazy vysvětlené v teoretické části, druhé jsou více komplexnější a používají větší množství příkazů. Uvedené je také porovnání mezi programy vytvořenými v Octave a v Matlabu, při kterém bylo zjištěno několik menších rozdílů v syntaxi jednotlivých programů.

Celkově je v teoretické části vysvětleno použití více jak sta příkazů , funkcí a operátorů. Většina z nich je také názorně použita v některém ze 40 vytvořených programů. I přes to je v této práci uvedena jen část všech příkazů, které Octave umožňuje používat.

ZÁVĚR V ANGLIČTINĚ

General object those work was creation programs in Octave, which cover all education bases of programming on subject Computer aid of automatic control that is taught on Faculty of applied informatics at Zlin. This subject is for the present specialized only on integrated environment Matlab.

One from programs which could him complete in education is Octave, whose latest version takes mark 3. That in comparison with last versions confess to many changes and improvement. Firstly it is not only software destined for Linux, how many users assume. Totally without any difficulty is possible use him also on computers equipped operating systems MacOS or MS Windows. Version 3 is more stable, includes several new orders and functions, and make possible install more supplements. Also for that is Octave again by something more near Matlab.

At first part those bachelor thesis are firstly described rounds of education of subject Computer aid of automatic control, at which is possible use Matlab and also program Octave . That is also subject of residue first parts . Firstly is described his history and individual parts of his installation . Theory is closed by list and description of basic orders needed for work in program .

Second part of this work is devoted for created programs, that are divided to the two groups. First illustrated commands which are explained in theoretic part, second are more complex and use more commands. Adduced is also contrast between programs created in Octave and in Matlab, whereat was developed several smaller differences in syntax individual programs.

Globally is in theoretic part explained using over most hundred commands , functions and operators. Most of these is also clearly used in some from 40 created programs. Through that is in this work mentioned only part of all orders that the Octave makes it possible to use.

SEZNAM POUŽITÉ LITERATURY

Monografie :

[1] PERŮTKA, K. : MATLAB – Základy pro studenty automatizace a informačních technologií , Zlín : Univerzita Tomáše Bati ve Zlíně , 2005 , ISBN 80-7318-355-2

[2] JUST, M. : Český průvodce programem Octave, Zlín : Univerzita Tomáše Bati ve Zlíně , Bakalářská práce, 2006

[3] PROKOP, D. : Průvodce grafickými nadstavbami Octave, Zlín: Univerzita Tomáše Bati ve Zlíně, Bakalářská práce, 2007

[4] HECZKO, M. : Výukový a zkušební program pro předmět PPAŘ, Zlín: Univerzita Tomáše Bati ve Zlíně, Bakalářská práce, 2006

[5] KADLEC, V. : Učíme se programovat v jazyce C, Brno: Computer Press, 2005, ISBN 80-7226-715-9

Internetové zdroje :

[6] Domovská stránka Octave [online].[cit 2008-05-12]. Dostupný z WWW:

<http://www.gnu.org/software/octave/>

[7] SELHOFER, H., OLIVER, M. : Introduction to GNU Octave , International University Bremen , Německo [online].[cit 2008-05-12]. Dostupný z WWW:

<http://math.jacobs-university.de/oliver/teaching/iub/resources/octave/octave-intro.pdf>

[8] Seriál o programování v Octave [online].[cit 2008-05-12]. Dostupný z WWW:

<http://www.abclinuxu.cz/clanky/programovani/octave >

[9] Octave – základy programování [online].[cit 2008-05-12]. Dostupný z WWW:

<http://www.root.cz/clanky/octave>

[10] Sylabus předmětu A4PAR [online].[cit 2008-05-12]. Dostupný z WWW:

<http://stag.utb.cz>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

C++	Objektově orientovaný programovací jazyk
CPU	Control processing unit – Procesor
CSS	Cascade style sheets
GNU	GNU is no Linux – Projekt svobodného software
GUI	Graphical user interface – Grafické uživatelské rozhraní
HTML	Hypertext Markup Language
MS	Microsoft
OS	Operační systém
PDF	Portable document format
PHP	Hypertext Preprocesor
SSE2	Sada instrukcí pro procesory

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH