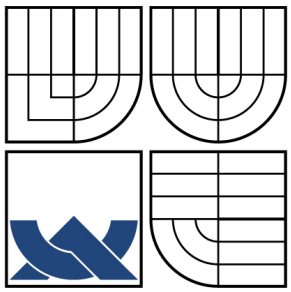
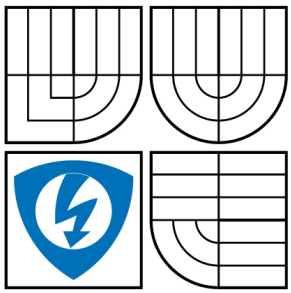


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

MATHSCRIPT V LABVIEW

MATHSCRIPT IN LABVIEW

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. MICHAL SELINGER

AUTHOR

VEDOUCÍ PRÁCE — Ing. MILOSLAV ČEJKA, CSc.

SUPERVISOR

BRNO 2008

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Selinger Michal, Bc.
Ročník: 2

ID: 47432
Akademický rok: 2007/08

NÁZEV TÉMATU:

MathScript v LabVIEW

POKYNY PRO VYPRACOVÁNÍ:

Seznamte s možnostmi textového programování v rámci systému LabVIEW 8. Porovnejte možnosti LabVIEW s možnostmi systému MATLAB fy MathWorks.

1/ Seznamte se s jazykem MathScript prostředí LabVIEW a popište jeho vlastnosti
2/ Vyhledejte rozdíly mezi obdobným popisem v prostředí MATLAB
3/ Stanovte, které funkce z obou prostředí jsou obdobné a na nich ověřte vlastnosti (přesnost, rychlost, syntaktické rozdíly)
4/ Zpracujte stručný návod pro použití MathScriptu a zhodnoťte výsledky.

DOPORUČENÁ LITERATURA:

Firemní literatura National Instruments, MathWorks.

Termín zadání: 3.12.2007 ***Termín odevzdání:*** 26.5.2008

Vedoucí projektu: Ing. Miloslav Čejka, CSc.

prof. Ing. Pavel Jura, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Vysoké učení technické Brno

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

MathScript v LabVIEW

Diplomová práce

Studijní obor: Kybernetika, automatizace, měření

Student: Bc. Michal Selinger

Vedoucí: Ing. Miloslav Čejka, CSc.

Abstrakt:

Hlavním cílem práce je zjistit přenositelnost programů (tzv. m-skriptů) mezi prostředím LabVIEW MathScript společnosti National Instruments a prostředím MATLAB společnosti MathWorks. Četné konzultace fór National Instruments, testy u nejasných příkladů, ale především nápověda obou prostředí vedla ke zpracování tohoto požadavku.

Další částí je zpracování návodu k prostředí LabVIEW MathScript. Jsou popsány základy spolu s omezeními a rozdíly oproti prostředí MATLAB.

Klíčová slova:

MathScript, MATLAB, LabVIEW

Brno University of Technology

The Faculty of Electrical Engineering and Communication

Department od Control, Measurement and Instrumentation

MathScript in LabVIEW

Master's Thesis

Specialization of study: Cybernetics, Control and Measurement

Student: Bc. Michal Selinger

Supervisor: Ing. Miloslav Čejka, CSc.

Abstract:

The aim of this thesis is to examine the portability of m-scripts between National Instruments LabVIEW MathScript and The MathWorks MATLAB developing environment. The differences between textual functions in both programs were also part of examination. This was done through consults of NI forums, numerous of individual tests and mainly through programme help.

The second goal is to elaborate a brief manual helping the user to start working with LabVIEW MathScript. The restrictions and differences between both developing environments are described also.

Keywords:

MathScript, MATLAB, LabVIEW

Bibliografická citace

SELINGER, M. MathScript v LabVIEW. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 92 s. Vedoucí diplomové práce Ing. Miloslav Čejka, CSc.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma "MathScript v LabVIEW" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o d ě k o v á n í

Na tomto místě bych rád poděkoval Ing. Miloslavu Čejkovi, CSc. za četné konzultace, cenné připomínky a rady při vypracování diplomové práce a dále své rodině, za neutuchající podporu po celou dobu studia.

V Brně dne : Podpis:

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

Zkratka / Symbol	Jednotka	Popis
CAD	-	návrhové programy Computer-aided Design
CAM	-	programy pro řízení výroby Computer-aided Manufacturing
DSP	-	digitální signálový procesor Digital Signal Processor
IEEE	-	asociace elektroinženýrů a informatiků The Institute of Electrical and Electronics Engineers
LabVIEW	-	počítačový program Laboratory Virtual Instrumentation Engineering Workbench
LTI	-	lineární, časově invariantní systém Linear Time-invariant
MATLAB	-	počítačový program Matrix Laboratory
UNICODE	-	univerzální znaková sada všech existujících abeced
ZV	-	zpětná vazba
$\bar{t}$	[s]	střední hodnota výběrového souboru hodnot

1. ÚVOD

V dnešní době se čím dál více uplatňují osobní a jiné druhy počítačů v domácnostech stejně tak jako v současném průmyslu. Hardware počítače nabývá na výkonu každým dnem avšak tím co dokáže tento výkon využít je programové vybavení počítače.

Z principu je možné programové vybavení počítače rozdělit do několika kategorií specifických pro předem určenou činnost. Kancelářské aplikace, grafika, CAD/CAM systémy, simulace, výpočty a spousty dalších. Stejně jako v jakémkoliv jiném odvětví v běžném životě i mezi programy, přesněji řečeno mezi jejich výrobci, existuje konkurenční prostředí. Každý se snaží nabídnout program co nejjednodušší, nejpřehlednější, nejkomplexnější a přitom doufá, že to bude právě ten jeho, který si uživatel vybere. Spousta programů tak může nabízet podobné, či stejné funkce a služby jako programy konkurenční.

Společnost National Instruments, výrobce software a hardware sloužící pro analýzu a sběr dat, měřicí, řídicí a další činnosti, nedávno doplnila funkce svého vývojového prostředí LabVIEW o možnost textového programování jazykem m-skript, jež nazvala MathScript. M-skriptovací jazyk je dnes používám ve stále více a více prostředích a pro svou strukturovanost a jednoduchost, je čím dál více populární.

Cílem této Diplomové práce je vypracování návodu k užívání jazyka MathScript prostředí LabVIEW a porovnání s obdobnými funkcemi v prostředí MATLAB společnosti MathWorks.

2. MATHSCRIPT

2.1 ÚVOD

MathScript je textový programovací jazyk, který byl vytvořen společností National Instruments. Je často spojován s prostředím LabVIEW, kde zvyšuje celkovou flexibilitu, použitelnost, ale i uživatelskou přitažlivost tohoto vývojového nástroje.

MathScript je datován od verze LabVIEW 8.0, kdy vznikl jako inovace tohoto vývojového prostředí. Díky novému matematicky orientovanému jazyku si může uživatel libovolně zvolit, zdali použije textový, grafický nebo kombinaci obou programovacích jazyků k řešení výpočetních algoritmů, zpracování, řízení, analýze signálů.

Na tomto místě je nutné podotknout, že LabVIEW již od svých raných verzí poskytovalo možnost textového programování. Prvním příkladem je blok „Formula Node“, který má syntaxi velmi podobnou jazyku C. Má navíc i omezené množství použitelných funkcí. Druhým příkladem je blok „Expression Node“, který disponuje opět omezeným množstvím funkcí a hodí se především pro výpočet jednoduchých rovnic. MathScript však jako jediný má vlastní vývojové prostředí, popsané dále, což ani jeden z právě zmíněných bloků neměl a nemá. MathScript je silný matematický nástroj a v případě výše zmíněných bloků je možné mluvit spíše jen o jednoduchých textově orientovaných programovacích nástrojích, jež mají usnadnit přechod na plně grafické programování. Oba nástroje, jak Formula Node, tak Expression Node zůstávají i nadále v distribucích LabVIEW kvůli zachování kompatibility se staršími, různým způsobem psanými programy.

Syntaxe jazyka MathScript byla záměrně volena tak, aby se příliš nelišila od syntaxe a příkazů používaných v prostředích MATLAB nebo COMSOL, čili tzv. m-skriptů. Až na drobné výjimky je tedy možná v drtivé většině běžných algoritmů

přenositelnost kódu mezi těmito a jinými vývojovými prostředími užívajícími skriptovací jazyk m-skript.

Od verze LabVIEW 8.0, kdy byl již MathScript plně použitelný pro vývoj různorodých algoritmů a obsahoval více, jak 500 textových příkazů, došel spousty změn. Největších změn dosáhl MathScript ve verzi vývojového prostředí LabVIEW 8.2, kdy došlo až ke ztrojnásobení rychlosti kompilace m-skriptů a možnosti rozdělení dlouhého kódu na funkční bloky (a tím zrychlení doby kompilace). Další novinkou bylo až zpětinásobení rychlosti práce s maticemi. Přidáno nebo opraveno bylo přes 200 funkcí a celkově tak obsahoval přes 600 použitelných funkcí pro práci s DSP, k vykreslování, pro práci se soubory a jiné. Byly přidány funkce pro efektivní řízení algoritmů a ještě efektivnější práci s maticemi, filtrování, modelování, analýza a transformace. Jedním z posledních, nikoliv však nejméně významným zlepšením bylo, že od této verze již mohly být MathScript algoritmy zcela volně spouštěny v samostatných spustitelných souborech, což v předchozí verzi nebylo možné. Aplikace využívající MathScript byly odkázány při spouštění na vývojové prostředí LabVIEW. Od teď již mohl MathScript zcela volně pracovat i s knihovnami.
V současné verzi 8.5 je MathScript schopen pracovat ještě efektivněji, než v předchozích verzích a obsahuje celkově již přes 650 textových funkcí.

2.2 PROSTŘEDÍ LABVIEW MATHSCRIPT

Toto prostředí slouží k editaci a vykonávání textových příkazů jazyka MathScript, vytváření tzv. m-skriptů, jejich ukládání, otevírání a k zobrazování textových i grafických výsledků vytvářených algoritmů.

Prostředí obsahuje několik oken. Výstupní okno (Output Window) zobrazující vstupní příkazy a odezvu na ně z prostředí MathScript. Výstupní okno je standardně umístěno v levé části prostředí.

Příkazové okno (Command Window) sloužící k uživatelskému vstupu příkazů. Zde je možno listovat historií příkazů pomocí šipek nahoru a dolů nebo

vkládat vícenásobné příkazy pomocí kombinace kláves „SHIFT-ENTER“ pro odskok na další řádek příkazového okna. Umístěno je vlevo dole.

Stavové okno (Status) umístěné hned pod příkazovým oknem slouží k informaci ohledně průběhu příkazů.

V pravé části prostředí LabVIEW MathScript jsou umístěny tři záložky. Jedna z nich, historie příkazů (History) slouží k prohlížení všech již provedených příkazů a umožňuje jejich vkládání pomocí myši opět do příkazového okna.

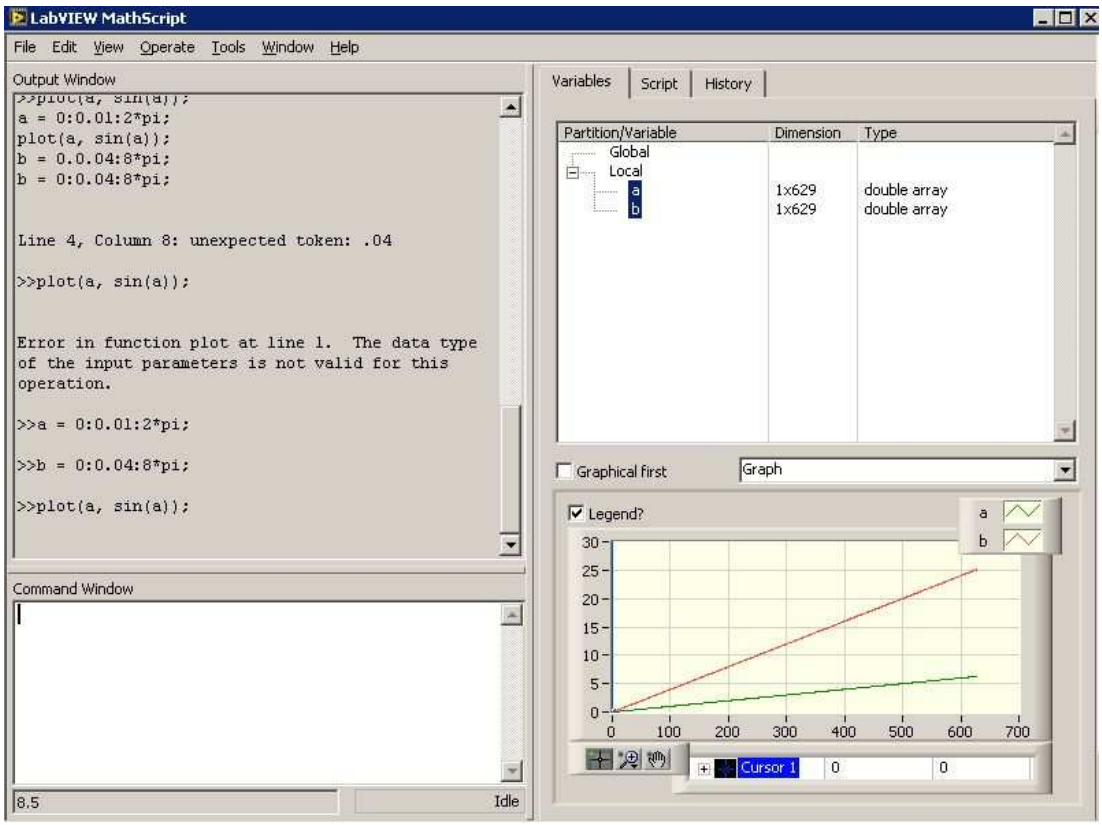
Další záložkou je záložka skript (Script). Tato slouží k vytváření, načítání, ukládání a provádění jednotlivých m-skriptů. Jednotlivé skripty však trpí jistými neduhy. Především horší čitelností díky tomu, že nejsou barevně značeny klíčové názvy a příkazy. Písmo není patkové a tudíž je rozdílná šířka písmene „m“ a písmene „i“. Při dlouhých a složitějších algoritmech tento fakt způsobuje různý odskok stejně dlouhých slov a tím přispívá ke zhoršené čitelnosti textu. Stejně tak nefungují automatické odrážky vpravo nebo vlevo pro editaci skriptu.

Třetí záložka je záložka proměnných (Variables). V této má uživatel přehled o použitých nebo předem zadaných proměnných. Zdali jsou lokální, globální, jejich název, rozměr a datový typ. Jsou řazeny abecedně. Po kliknutí na název libovolné proměnné se ve spodní části pravé oblasti prostředí LabVIEW MathScript zobrazí obsah této proměnné. Proměnná lze interpretovat jako numerický výsledek, řetězec, časový graf, XY graf, 3D graf, obrázek, dokonce i zvukový záznam. Zvolit můžeme samozřejmě i více proměnných současně pomocí kombinací kláves „CTRL“ a „SHIFT“ a ty potom zobrazit např. v XY grafu. Způsob reprezentace výsledku se zobrazuje pro každý datový typ různě. Výsledek je potom možné zobrazit v samostatném okně, přizpůsobit jak velikostně, tak obsahově a exportovat.

V nastavení prostředí LabVIEW MathScript je možné přizpůsobit uživateli samozřejmě spoustu dalších věcí, jako jsou fonty hlavních oken (nikoliv záložky skript), standardní číselnou reprezentaci výsledku, velikost vyrovnávací paměti nebo umístění, kde se mají hledat m-skripty, automatické ukládání a jiné, které umožní prostředí lépe přizpůsobit jednotlivému uživateli.

Uživateli, který již pracoval s prostředím MATLAB společnosti MathWorks jistě neunikne, že jsou si tyto dvě prostředí velice podobná jak vzhledem, tak

obsahem. Dochází pouze k drobným změnám v uspořádání jednotlivých oken. Společnost National Instruments tak pravděpodobně vzala v potaz jistou zaběhlost a tradici prostředí MATLAB a usnadnili tak uživatelům orientaci a práci v jimi vytvořeném prostředí LabVIEW MathScript.



Obrázek 1 Prostředí LabVIEW MathScript

2.3 ZÁKLADY JAZYKA

Jazyk MathScript, jak bylo řečeno, je maticově orientován. Veškerá jeho síla tkví v neuvěřitelně efektivní práci s maticemi a jako k takovému se k němu musí přistupovat.

Matice je obdélníkové nebo čtvercové uspořádání prvků. Její rozměr udávají velikosti jednotlivých rozměrů, což je rozměr1 x rozměr2 x rozměr3 x… a tak

bychom mohli pokračovat dle libosti. Jak je vidět matice může teoreticky dosahovat libovolných rozměrů, nicméně v jazyku MathScript je maximální dimenze matice 2, čili počet řádků x počet sloupců (ŘxS). Speciálním typem matice je vektor, což je matice s jedním rozměrem rovným 1. Podle uspořádání mluvíme o řádkovém (1xS), či sloupcovém vektoru (Řx1). Posledním speciálním typem matice je skalár, což je matice rozměru 1x1. Ke každému prvku jazyk MathScript přistupuje jako k matici.

2.3.1 Základní operace s maticemi

Mezi základní operace s maticemi bezpochyby patří jejich vytvoření, zápis a adresování. Matici deklarujeme v prostředí LabVIEW MathScript několika možnými způsoby. Jedním z nich je vypsáním přímo prvků matice v hranatých závorkách. Sloupcové prvky jsou od sebe odděleny čárkou nebo mezerou. Jednotlivé řádky matice jsou pak oddělovány středníkem. Matice je zapisována po řádcích. Tímto definujeme matici A rozměru 2x2 `A = [0 0 ; 0 5]`. Dalším způsobem je například definice rozměrů matice v kulatých závorkách za názvem symbolické proměnné reprezentující matici. `A(2, 2) = 5`, tímto inicializujeme prvek na druhém řádku a ve druhém sloupci na hodnotu 5, všechny předchozí prvky jsou přitom doplněny automaticky a inicializovány na 0 samotným programem. Oba příkazy tedy udělají totéž.

K prvkům matice můžeme přistupovat pomocí zápisu jež obsahuje symbolickou proměnnou reprezentující matici a v kulatých závorkách umístění prvku v matici nebo ve vektoru takto `A(1, 3)`. Příkaz vrací hodnotu prvku uloženého v prvním řádku a třetím sloupci matice A. K prvkům je možné přistupovat i uvedením pouze jedné pozice, např. `A(5)`. Tato potom značí vzdálenost hledaného prvku od prvního prvku v matici, přičemž program postupuje po řádcích (první sloupec, první řádek – první sloupec, druhý řádek - … - n-tý sloupec, n-tý řádek) až se posune na požadovaný prvek matice. Indexace začíná číslem 1. Můžeme samozřejmě přistupovat jen k určité části matice, kterou specifikujeme rozsahem. Rozsah je ohraničen mezemi dolní a horní, mezi nimiž je dvojtečka. Budeme-li chtít

zobrazit submatici obsahující 2, 3 řádek a 2, 3 sloupec v matici A rozměru 3x3 uděláme to příkazem `A(2:3, 2:3)`. Jednoduchou náhradou samostatné dvojtečky místo rozměru určení řádku nebo sloupce vezmeme v úvahu celý jeden rozměr. Tedy celý řádek nebo celý sloupec. Takto zobrazíme všechny sloupce druhého řádku matice A

`A(2, :)`. Existuje zde ještě jeden symbolický název pro operace s prvky matic a tím je příkaz „end“. Pomocí něj programu říkáme, že chceme nějakým způsobem operovat s posledním prvkem matice. Máme-li např. matici A rozměrů 10x10
a budeme chtít označit 4 až 10 řádek a všechny sloupce můžeme to udělat např. takto `A(4:10, :)`, anebo pomocí symbolického příkazu „end“ takto `A(4:end, :)`. Výhoda symbolické proměnné je, že můžeme poměrně lehce operovat s oběma stranami matice nebo vektoru bez dalšího zjišťování rozsahu některého z rozměrů.

2.3.2 Operátor rozsahu

Operátor rozsahu, tedy dvojtečku, jsme již použili. Ve skutečnosti je možné pomocí tohoto operátoru zvládnout daleko více věcí, než jen označit určité prvky matice nebo vektoru. Jak např. definovat číselnou posloupnost od 1 až do 1000? Velmi jednoduše a efektivně pomocí operátoru rozsahu a to následovně
`t = 1:1000;`. Takto jsme definovali vektor od 1 až do 1000 s krokem 1 v zanedbatelném čase oproti indexaci v cyklu. Rozsah je možné specifikovat dle požadavků uživatele, je možné zvolit krok, nebo změnit posloupnost na klesající a to takto `t = 1000:-2:0;`. Tímto definujeme klesající posloupnost s krokem 2. V případě klesající posloupnosti musí být vždy uvedena první mez větší, druhá mez menší a krok se záporným znaménkem, jinak se příkaz neprovede. Můžeme vytvořit i posloupnost znakovou `znaky = 'A':'Z';`. MathScript však inicializuje prvky posloupnosti jako ASCII hodnoty. Definovaný vektor jazykem MathScript může mít jen omezenou délku.

2.3.3 Potlačení výstupu

Jedním z důležitých příkazů je potlačení výstupu. Při spuštění jakéhokoliv příkazu bez potlačení výstupu se nám ve výstupním okně objeví odpověď o provedeném příkazu. Tento fakt brzdí výkon prostředí a dost často i obtěžuje samotného uživatele. Má se totiž za to, že jestliže uživatel ví, co dělá, nepotřebuje být nadále informován, že to již udělal. V tzv. m-skriptu plném příkazů algoritmu je to vyloženě medvědí služba, neboť čas běhu programu potom bývá daleko delší. V prostředí MathScript je proto k dispozici příkaz potlačení výstupu, který se může zapsat za jakýkoliv příkaz a tím omezit jeho odezvu do výstupního okna, příkaz se přitom vykoná zcela regulérně bez jakýchkoliv dalších omezení v paměti počítače. Pro potlačení výstupu se používá středník za příkazem. Tímto např. definujeme matici A rozměrů 5x5 a inicializujeme její hodnoty na 0, přičemž se nám obsah této matice již nevypíše, jako tomu bylo v předchozích případech `A(5, 5) = 0;`.

Použít potlačení výstupu lze u libovolného příkazu a není to chybou. Jsou ovšem příkazy, které toto ignorují. Mezi takové příkazy patří např. příkaz „plot“, „disp“ a jiné. Jsou to příkazy z třídy zobrazování, u kterých by bylo nelogické potlačovat jejich výstup, když jejich jediná funkce je takto graficky nebo textově informovat uživatele.

2.3.4 Nápověda

Nápověda je klíčovou částí jakéhokoliv programu. Nechybí ani v prostředí MathScript. Nápovědu je možné jednoduše vyvolat syntaxí `help jmenofunkce;`. Přístup k nápovědě je možný také pomocí tlačítka „F1“, či přes nabídku z lišty „Help >> Search the LabVIEW Help“ v prostředí MathScript.

2.3.5 Rozdíly v prostředí MATLAB

V těchto základních příkazech prostředí MATLAB a MathScript je pouze jeden závažný rozdíl a tím je fakt, že v prostředí MATLAB lze definovat matice libovolných dimenzí. V prostředí MathScript jsou to matice maximálně dvourozměrné. Dále prostředí MathScript nedokáže definovat tzv. řídké matice (Sparse Matrix). To jsou matice, kde ne všechny prvky obsahují hodnotu. Takové matice pak zabírají jen zlomek velikosti paměti oproti matici plné (Full Matrix). Matice se v prostředí MATLAB konvertuje příkazem „sparse“. Musí se však jednat o matici maximálně dvou dimenzí datového typu „double“ nebo „logical”. Dalším rozdílem je snad jen to, že znakovou posloupnost prostředí MATLAB neuloží jako vektor ASCII hodnot, ale jako vektor znakových proměnných. Zároveň umožňuje rozlišování klíčových slov v textu barvami, patkové písmo pro lepší čitelnost a automatické odskoky v textu.

2.4 DATOVÉ TYPY

Každá proměnná používaná v prostředí MathScript musí mít jasně specifikováno jakého datového typu jsou uložená data a jak k nim tedy přistupovat. Programátor nemusí definovat datový typ používané proměnné jako by tomu bylo v jiných jazycích (např. C). Prostředí MathScript používá jazyk, jež se označuje jako „volně psaný“. V tomto případě si vývojové prostředí zvolí vlastní, implicitní, datový typ, do kterého lze proměnnou uložit. Tento způsob však může být nežádoucí, neboť se algoritmus může dopustit varovných hlášení, hrubých chyb nebo nefungovat zcela podle představ programátora. V datové třídě „support“ je přiložena celá řada příkazů na konverzi datového typu. Patří mezi ně příkazy „int8“, „int16“, „int32“, „int64“, „uint8“, „uint16“, „uint32“, „uint64“, „single“ a „double“. Naneštěstí matematické operace jakékoliv mutace znaménkového celočíselného datového typu („int“) nebo neznaménkového celočíselného datového typu („uint“) nejsou prostředím MathScript

podporovány. Na výběr z datových typů tedy zůstává číselný typ „double“, znakový typ „char” a logický datový typ „logical” se kterými dokáže prostředí MathScript pracovat.

Jakýkoliv z výše uvedených typů může být reprezentován jako skalár (1x1), vektor (Řx1, 1xS) nebo jako matice (ŘxS).

2.4.1 Rozdíly v prostředí MATLAB

Oproti prostředí MathScript poskytuje prostředí MATLAB daleko více základních datových typů. Mezi hlavní patří logický typ „logical”, znakový typ „char”, celočíselné znaménkové datové typy „int8“, „int16“, „int32“, „int64“, a celočíselné neznaménkové datové typy „uint8“, „uint16“, „uint32“, „uint64“, číselný datový typ „single“ a „double“. Mimo uvedené podporuje i struktury „structure“, buňky „cell“ a ukazatele na funkci „function handle“. Buňky (Cells) mohou obsahovat jakýkoliv typ, i navzájem různé, stejně tak struktura (Structure). Ukazatel na funkci (Function Handle) slouží k uchování ukazatele pro volání funkce. Existují i rozšířené datové typy, mezi něž patří uživatelem definované datové typy a datové typy pro práci s Java objekty. Nutno však podotknout, že s celočíselnými datovými typy „int64“ a „uint64“ není možné provádět žádné matematické operace stejně jako v prostředí MathScript.

2.5 FORMÁTOVÁNÍ VÝSTUPU

Výstupem z prostředí MathScript může být textová, stejně tak i grafická informace pro uživatele. Obě tyto možnosti se dají v určitých mezích formátovat a upravovat tak, jak je to pro uživatele nejvhodnější.

2.5.1 Textový výstup

Číselné formáty se týkají pouze zobrazení výsledků. Výpočty se vždy provádí v tzv. dvojité přesnosti (Double Precision) číselné datové třídě „double“. Zobrazení je možné modifikovat následujícími způsoby. Zobrazovat můžeme v pevné nebo pohyblivé řádové čárce a to na 5 nebo 15 platných číslic. Na 5 platných číslic zobrazujeme modifikátorem „short“ za příkazem „format“, na 15 číslic modifikátorem „long“. Posledním modifikátorem může, ale nemusí být „e“, jež značí zobrazení v pohyblivé řádové čárce. Chci-li např. zobrazit výsledek na 15 platných číslic v pohyblivé řádové čárce napíši příkaz `format long e;`. Implicitní nastavení je 5 platných číslic v pevné řádové čárce. Samostatným příkazem „format“ resetuji nastavení na implicitní.

2.5.2 Grafický výstup

Typickým grafickým výstupem je graf. V prostředí LabVIEW MathScript je k dispozici velké množství grafů: sloupcové, koláčové, sférické, prostorové nebo klasické XY s lineárními, logaritmickými osami. Každý takový graf jako celek se skládá ze čtyř samostatných instancí, které se dají formátovat. Je to samotná grafická křivka, grafická oblast (podklad na kterém je křivka vykreslena), grafické okno (vyskakující okno s grafem) a textový popisek.

Křivka slouží pro zobrazení průběhu funkce. Příkazy vykreslující křivku a zároveň umožňující nastavovat její parametry jsou „line“, „loglog“, „plot“, „semilogx“, „semilogy“. Standardní hodnoty (velikost bodu, tvar, tloušťka čáry, barva) nemusí uživateli z nějakého důvodu vyhovovat a proto je možné je modifikovat.

Tabulka 1 Vlastnosti grafické křivky

Vlastnosti grafické křivky		
označení vlastnosti	popis	přípustné hodnoty
Color	barva křivky, standardní nastavení je ‘b’ (modrá), či maticově [0, 0, 1]	‘r‘ (červená), ‚g‘ (zelená), ‚b‘ (modrá), ‚c‘ (azurová), ‚m‘ (purpurová), ‚y‘ (žlutá), ‚k‘ (černá), ‚w‘ (bílá) je možno zadat i pomocí RGB palety maticově takto [r, g, b], každá barva nabývá hodnot <0;1>
LineStyle	typ křivky, standardně je nastavena na ‘-’	‘-, (pevná křivka), ‚--, (čárkovaná), ‚:‘ (tečkovaná), ‚-.‘ (čerchovaná)
LineWidth	určuje tloušťku křivky, standardně je 1	číslo v rozmezí <0;5>
Marker	nastavuje značku, standardně je ‘none’ (žádná)	‘none‘ (žádná), ‚.‘, ‚o‘, ‚x‘, ‚+‘, ‚*‘, ‚s‘ (čtverec), ‚d‘ (kosočtverec)
MarkerSize	určuje velikost značek, standardně je 1	číslo v rozmezí <0;5>
Type	určuje typ grafického prvku, tato hodnota se nedá nastavit, lze jen číst	‘line‘ (grafická křivka), ‘plot area‘ (grafická oblast), ‚plot window‘ (grafické okno) ‚text‘ (text)
Visible	nastavuje grafickou křivku viditelnou, či ne, standardně je ‘on’ (viditelná)	‘on‘ (viditelná), ‚off‘ (neviditelná)
XData	nastavuje či vrací soubor hodnot na nezávislé ose	
YData	nastavuje, či vrací soubor hodnot na závislé ose	

Až na vlastnost „Type“ lze všechny ostatní libovolně modifikovat a to dvěma způsoby. Prvním z nich je jejich specifikace přímo ve vytváření grafu a to následujícím způsobem.

```
t = 0:pi/100:2*pi;
plot(t, sin(t), 'Color', 'r', 'LineStyle', ':', 'Marker', 'o',
'LineWidth', 4);
```

Takto jsem vytvořil graf funkce sin(t) pro t = <0;2π> s červenou tečkovanou křivkou tloušťky 4 a kolečky jako značkami. Jde to i jednodušeji, např. `plot(t, sin(t), 'r:o', 'LineWidth', 4);`. Za povšimnutí stojí především zkrácená verze zápisu barvy křivky, stylu čáry a značky. Druhý způsob je použití funkcí „get“, „set“ u již vytvořených grafů. Každý graf vrací jako návratovou hodnotu referenci sám na sebe, čili ukazatel na místo v paměti, kde se nachází. Funkcí „set“ potom můžeme přes tuto referenci nastavovat libovolný z výše uvedených dovolených parametrů. Stejně tak funkcí „get“ pomocí reference na objekt lze získat libovolný z výše uvedených parametrů. Příklad, jak to udělat je uveden zde.

```
t = 0:pi/100:2*pi;
reference = plot(t, sin(t));
tlouska_cary = get(reference, 'LineWidth');
set(reference, 'LineWidth', tloustka_cary+1);
```

Přes referenci jsem získal tloušťku grafické křivky a následně ji rozšířil o 1. V jednom grafu se může vyskytovat i více křivek, např. pomocí příkazu `hold on;`. Tento zastaví vytváření nových grafických oken a vykresluje do toho samého (naposledy vytvořeného), dokud není příkaz zrušen následovně `hold off;`. Každá z takto vykreslených křivek pak vrací vlastní referenci, takže je možné nastavit v rámci možností libovolnou křivku v grafu, jak uživatel uzná za vhodné.

Dalším grafickým prvkem je grafická oblast. Jejím nastavením se dosahuje změny rozsahů závislé, nezávislé osy, barvy značek na osách, barvy pozadí grafu, atd. Tyto parametry je možné nastavit spolu s příkazy „axes“, „gca“ a „subplot“. Nastavení může proběhnout opět dvěma způsoby, tak jako v případě předchozím. Zadáním přímo nebo přes referenci na objekt příkazy „get“ a „set“.

Tabulka 2 Vlastnosti grafické oblasti

Vlastnosti grafické oblasti		
označení vlastnosti	popis	přípustné hodnoty
Color	barva křivky, standardní nastavení je ‘w’ (bílá), či maticově [1, 1, 1]	‘r‘ (červená), ‚g‘ (zelená), ‚b‘ (modrá), ‚c‘ (azurová), ‚m‘ (purpurová), ‚y‘ (žlutá), ‚k‘ (černá), ‚w‘ (bílá) je možno zadat i pomocí RGB palety maticově takto [r, g, b], každá barva nabývá hodnot <0;1>
ColorOrder	vrátí pole barev v které jsou postupně automaticky přidělovány dalším křivkám v grafu, lze pouze číst	vrací maticově pomocí RGB palety takto [r, g, b], každá barva nabývá hodnot <0;1>
FontAngle	nastavuje text v grafu na kurzívu, či normální styl (standardně)	‘italic’ (kurzíva), ‘normal’ (normální písmo)
FontName	pojmenovává textový objekt, standardně „App Font“	název ve formátu string
FontSize	specifikuje standardní velikost textu v grafu, implicitně 13	číselný formát
FontWeight	určuje, zdali je text tučně, či nikoliv (standardně)	‘bold’ (tučné), ’normal’ (normální písmo)
NextPlot	jakým způsobem se má přidat nový graf do oblasti, implicitně nastaveno „replace“	‘add’ (přidá zcela nový), ’replace’ (přepíše starý)
OuterPosition	vrátí relativní umístění os vzhledem k oknu a šířku os, lze pouze číst	ve formátu [polohax, polohay, sirkax, sirkay], vše nabývající hodnot <0;1>

Pokračování Tabulka 2
Vlastnosti grafické oblasti

označení vlastnosti	popis	přípustné hodnoty
Position	vrátí relativní umístění oblasti vzhledem k oknu a lze pouze číst	ve formátu [polohax, polohay, sirkax, sirkay], vše nabývající hodnot <0;1>
Title	Vrací název oblasti, jen pro čtení	
Type	určuje typ grafického prvku, tato hodnota se nedá nastavit, lze jen číst	‘line‘ (grafická křivka), ‘plot area‘ (grafická oblast), ‚plot window‘ (grafické okno) ‚text‘ (text)
Visible	nastavuje grafickou oblast viditelnou, či ne, standardně je ‘on’ (viditelná)	‘on‘ (viditelná), ‚off‘ (neviditelná)
XColor YColor	barva osy, standardní nastavení je ‘k’ (černá), či maticově [0, 0, 0]	‘r‘ (červená), ‚g‘ (zelená), ‚b‘ (modrá), ‚c‘ (azurová), ‚m‘ (purpurová), ‚y‘ (žlutá), ‚k‘ (černá), ‚w‘ (bílá) je možno zadat i pomocí RGB palety maticově takto [r, g, b], každá barva nabývá hodnot <0;1>
XDir YDir	určuje, zda jsou hodnoty na ose vzestupně (standardně), či sestupně	‘normal’ (vzestupně), ’reverse’ (sestupně)
XGrid YGrid	určuje viditelnost mřížky os, standardně není viditelná	‘on’ (viditelná), ’off’ (neviditelná)
XLabel YLabel	vrací popisek osy, jen pro čtení	
XLim YLim	nastavuje minimální a maximální značku na ose, standardně je [0, 10]	[minimum, maximum]
XLimMode YLimMode	nastavuje rozsah os podle vstupních dat (standardně)	‘auto’ (automaticky), ’manual’ (ručně)

Pokračování Tabulka 2 Vlastnosti grafické oblasti		
označení vlastnosti	popis	přípustné hodnoty
XMinorGrid YMinorGrid	určuje viditelnost podružné mřížky os, standardně není viditelná	‘on’ (viditelná), ’off’ (neviditelná)
XMinorTick YMinorTick	určuje, zdali zobrazit značky podružné mřížky osy, standardně vypnuto	‘on’ (viditelné), ’off’ (neviditelné)
XScale YScale	nastavuje typ osy, standardně lineární	‘linear’ (lineární), ’log’ (logaritmická)
XTick YTick	nastavuje značky na hlavní mřížce osy, standardně []	[znacka1, znacka2, znacka3…]
XTickMode YTickMode	nastavuje, zdali zvolit značky na ose automaticky (standardně)	‘auto’ (automaticky), ’manual’ (ručně)

Dalším prvkem je grafické okno. Grafické okno se objeví pokaždé, když chceme vykreslit jakýkoliv graf. Je to vlastně plocha pro grafickou oblast a křivku v ní. Parametry grafického okna můžeme upravovat ve spojitosti s funkcemi „clf“, „figure“ a „gcf“. Jejich nastavení může proběhnout opět buď přímo při vytváření nebo pomocí funkcí „get“ a „set“. Seznam parametrů následuje v tabulce.

Tabulka 3 Vlastnosti grafického okna

Vlastnosti grafického okna		
označení vlastnosti	popis	přípustné hodnoty
Color	barva pozadí grafického okna, standardní nastavení je nastavení systémové	‘r‘ (červená), ‚g‘ (zelená), ‚b‘ (modrá), ‚c‘ (azurová), ‚m‘ (purpurová), ‚y‘ (žlutá), ‚k‘ (černá), ‚w‘ (bílá) je možno zadat i pomocí RGB palety maticově takto [r, g, b], každá barva nabývá hodnot <0;1>

pokračování Tabulka 3
Vlastnosti grafického okna

označení vlastnosti	popis	přípustné hodnoty
CurrentAxes	vrací identifikátor současného grafu, jen pro čtení	číslo
Name	umožňuje pojmenovat grafické okno v systémové liště	
NextPlot	jakým způsobem se má přidat nová oblast do okna, implicitně nastaveno „replace“	‘add’ (přidá zcela novou), ’replace’ (přepíše starou) ’new’ (vytvoří nové okno)
NumberTitle	nastavuje, zdali zobrazovat pořadí grafického okna v systémové liště, standardně ano	‘on’ (zobrazovat), ’off’ (nezobrazovat)
Position	vrátí relativní umístění a šířku okna vzhledem k obrazovce	ve formátu [polohax, polohay, sirkax, sirkay], vše nabývající hodnot <0;1>
Resize	nastavuje, zdali může uživatel měnit myší velikost grafického okna, standardně může	‘on’ (povoleno), ’off’ (nepovoleno)
Type	určuje typ grafického prvku, tato hodnota se nedá nastavit, lze jen číst	‘line‘ (grafická křivka), ‘plot area‘ (grafická oblast), ‚plot window‘ (grafické okno) ‚text‘ (text)
Visible	nastavuje grafické okno viditelné, či ne, standardně je ‘on’	‘on‘ (viditelné), ‚off‘ (neviditelné)

Posledním prvkem jsou textové popisky. I tyto je možno podobným způsobem, jako předchozí parametry grafického vyjádření výsledků, nastavit. Jedná se především o příkazy „text“, „title“, „xlabel“ a „ylabel“ jež umožňují upravovat parametry textových popisků. Opět je tu dvojí způsob a to buď při vytváření popisku

nebo přes referenci pomocí funkcí „set“ a „get“. Parametry jsou vysvětleny v tabulce níže

Tabulka 4 Vlastnosti textových popisků

Vlastnosti textových popisků		
označení vlastnosti	popis	přípustné hodnoty
BackgroundColor	barva pozadí textové oblasti, standardní nastavení je ‘w’ (bílá), neboli [1, 1, 1]	‘r‘ (červená), ‚g‘ (zelená), ‚b‘ (modrá), ‚c‘ (azurová), ‚m‘ (purpurová), ‚y‘ (žlutá), ‚k‘ (černá), ‚w‘ (bílá) je možno zadat i pomocí RGB palety maticově takto [r, g, b], každá barva nabývá hodnot <0;1>
Color	barva textu, standardně ‘k’ (černá), neboli [0, 0, 0]	
EdgeColor	barva obrysu textové oblasti, standardně ‘w’ (bílá), neboli [1, 1, 1]	
FontAngle	nastavuje text na kurzívu, či normální styl (standardně)	‘italic’ (kurzíva), ‘normal’ (normální písmo)
FontName	pojmenovává textový objekt, standardně „App Font“	název ve formátu string
FontSize	specifikuje standardní velikost textu, implicitně 13	číselný formát
FontWeight	určuje, zdali je text tučně, či nikoliv (standardně)	‘bold’ (tučné), ’normal’ (normální písmo)
Horizontal-Alignment	vodorovné zarovnání textu, standardně vlevo	‘center’ (na střed), ’left’ (vlevo), ’right’ (vpravo)
Position	nastavuje umístění textu	[polohax, polohay], vše v rozsahu hodnot <0;1>
String	obsahuje samotný text	

Pokračování Tabulka 4 Vlastnosti textových popisků		
označení vlastnosti	popis	přípustné hodnoty
Type	určuje typ grafického prvku, tato hodnota se nedá nastavit, lze jen číst	‘line‘ (grafická křivka), ‘plot area‘ (grafická oblast), ’plot window‘ (grafické okno) ‘text‘ (text)
Vertical-Alignment	svislé zarovnání textu, standardně na střed	‘baseline’ (spodní hrana textové oblasti), ’bottom’ (dolů), ’cap’ (nahoru), ’middle’ (na střed), ’top’ (horní hrana textové oblasti)
Visible	nastavuje text viditelný, či nikoliv, standardně je ‘on’ (viditelný)	‘on‘ (viditelný), ’off‘ (neviditelný)

Jak je vidět parametrů přístupných uživatelskému nastavení je mnoho a co víc, jsou pro každý grafický prvek pokaždé trochu odlišné. Jeden způsob je tyto parametry získat z nápovědy přes některý z podporovaných příkazů, jiný způsob poskytuje přímo příkaz „get“. Tento, zadaný pouze s referenčním parametrem na grafický objekt, vrací řetězcovou matici se všemi dostupnými parametry k tomuto objektu. Obsahuje symbolické názvy s aktuálními nastaveními.

Doposud jsem uvedl, že změna parametrů grafické reprezentace výsledků je možná dvěma způsoby, což je pravda. K těmto dvou způsobům specifickým pro každý grafický prvek se teď přidává ještě třetí. Parametry uvedené výše se dají měnit v již vytvořeném grafu nejen pomocí referencí přes funkce „get“ a „set“, ale i myší. Pravým kliknutím na vykresleném grafu a vyvoláním menu lze editovat drtivou většinu parametrů všech grafických prvků uvedených výše. Lze zapnout měřítko os, posuvníky, legendu, přejmenovávat grafické prvky, měnit typ grafu, barvu křivky a značek, typ značky, tloušťky a typy čar, zvětšovat, zmenšovat a měnit zobrazení, exportovat buď jako bitmapu nebo vektorovou grafiku.

2.5.3 Rozdíly v prostředí MATLAB

Prostředí MATLAB poskytuje řadu dalších voleb pro zobrazování výsledků mimo těch, které obsahuje prostředí MathScript. Dalšími modifikátory mohou být za volbami „short“, či „long“ mimo „e“ ještě „g“, jež poskytuje nejlepší z volby mezi pevnou a pohyblivou řádovou čárkou. Modifikátorem „eng“ můžeme navíc zobrazit číslo v inženýrské notaci. Pro číselný datový typ „single“ má formát „long“ poloviční počet číslic.

Dalšími formáty čísel jsou „+“ zobrazující pouze znaménko čísla. Formát „bank“ zobrazující číslo v bankovním formátu (dolary.centy zaokrouhleno na dvě desetinná místa). Formát „hex“ zobrazující číslo hexadecimálně a formát „rat“ zobrazující jakékoliv číslo jako podíl dvou celých čísel.

Poslední modifikací typu je formát „compact“, jež těsná výsledek a nenechává volné řádky při výpisu. Opakem je formát „loose“, jež naopak vkládá volné řádky při výpisu výsledku, aby tak zlepšil čitelnost. Formát „loose“ je nastaven jako implicitní.

Ohledně parametrů grafických objektů poskytuje prostředí MATLAB navíc hned několik odlišných nastavení. V prostředí MATLAB se vyskytují stejným způsobem nastavovaná grafická schémata i jednotlivé barvy symbolickými značkami. Styl čáry je rovněž totožný, nicméně tloušťka čáry není omezena jako je tomu v prostředí MathScript a její standardní hodnota je rovněž jiná. Pokud jde o značky je možno navíc umístit trojúhelníkové, všemi čtyřmi směry směřující, pentagon i hexagon jako další prvek. Barvu těchto značek, stejně tak i případných výplní je možné upravovat libovolně a nezávisle. V prostředí MathScript se barva značek nezávisle upravovat nedá, je totožná s barvou křivky. Velikost značek je v prostředí MATLAB opět neomezena, oproti omezení v prostředí MathScript. Barevná schémata jsou v obou prostředích shodná. Parametr volby kurzívou psaného textu je v prostředí MATLAB rozšířen o volbu 'oblique'. Velikost textu lze možno nastavovat ve více jednotkách, v prostředí MathScript lze pouze v pixelech. Tloušťka textu umožňuje navíc i volby 'light' a 'demi'. Nastavení viditelnosti jsou totožná

a další změna se týká jen typu vyjádření grafického prvku (line x line, plot area x axes, plot window x figure, text x text). Prostředí MathScript nenabízí zdaleka všechny vlastnosti grafických objektů jako prostředí MATLAB.

V prostředí MATLAB je dále možné nastavovat typ křivky zobrazující mřížku, značky na osách, vně, uvnitř, velikosti. Dále umožňuje zapisovat textové popisky ve značkovacím jazyce TeX a LaTeX, volba šířky okrajů textových popisků, tloušťka čáry a její typ, volba jednotek, ve kterých se velikost textu vyjadřuje. Navíc umožňuje spravovat objekty přes potomky a rodiče, umožňuje detekovat kliknutí na objekt a dále s ním pracovat. Umožňuje volat externí funkce, jakmile se s objektem něco děje, změna velikosti, snaha o zavření. Nasvícení scény, úhel pohledu a to zdaleka není konec výčtu seznamu vlastností. Prostředí MATLAB poskytuje několik desítek vlastností k jednotlivým objektům navíc oproti prostředí MathScript. Pro práci s nimi je vždy vhodné využít nápovědy.

I v prostředí MATLAB je možné měnit nastavení pomoci myši u již vytvořeného zobrazení má však daleko více možností nastavení. Pro pohodlné nastavování je v prostředí MATLAB přístupný „Object Inspector“, jež je graficky pěkně vyvedené prostředí pro editaci parametrů objektů.

2.6 PROMĚNNÉ

Proměnná je symbolické označení pro místo v paměti, kde je uložena hodnota přesně definovaného datového typu. Název proměnné definuje uživatel, musí se však držet jistých pravidel.

Proměnné se dělí na lokální a globální. Globální jsou z hlediska použitelnosti přístupné i funkcím mimo právě běžící funkci nebo cyklus. Např. definice globální proměnné A probíhá příkazem `global A;`. Podobným modifikátorem je příkaz „persistent“. Jeho použití je možné pouze uvnitř funkce a pro definovanou proměnnou vytvoří místo v paměti, podobně jako u příkazu „global“. Rozdíl je pouze v tom, že k této proměnné potom může přistupovat pouze funkce, ve které byla

proměnná definována. Ta si při každém opětovném volání funkce uchovává svoji původní hodnotu.

Název proměnné může mít neomezenou délku. Existuje zde sice příkaz „namelengthmax“, který vrací hodnotu maximální délky proměnné, ale při nadefinování proměnné delší než je tato hodnota se žádná varovná, ani chybová hlášení nevypíší a s proměnnou se mohou provádět zcela běžné operace. Název se však musí skládat ze znaků UNICODE a nelze definovat proměnné začínající podtržítkem, mezerou nebo číslicí. Zároveň se doporučuje, aby se uživatel vyvaroval názvu již definovaných funkcí. Prostředí MathScript sice takovou proměnnou nadefinuje (splňuje-li předpoklady o povolených znacích), ale pak již nebude možno používat původní funkci, neboť při užití takového názvu se bude přistupovat k definované proměnné. Abych se vyhnul tomuto problému mohu si například zjistit, kde všude se mnou zvolený název proměnné vyskytuje takto `which jmenozkoumanepromenne;`. Prostředí MathScript je „Case-Sensitive“, tzn., že rozlišuje mezi velkými a malými znaky. Navíc pro kontrolu, zda lze použít název proměnné, je implementována funkce „isvarname“, kterou mohu logicky otestovat, zda mnou zvolená proměnná může existovat.

2.6.1 Speciální proměnné

Patří sem proměnné, které jsou součástí jádra prostředí MathScript. MathScript počítá veškeré výpočty s tzv. dvojitou přesností (Double Precision), dle IEEE 754, jak bylo řečeno výše a od toho se odvíjí hodnoty těchto speciálních proměnných.

Tabulka 5 Speciální proměnné v MathScript, dle IEEE

Speciální proměnné v MathScript, dle IEEE	
symbolická proměnná	vysvětlení
pi	Ludolfovo číslo, obvod kruhu dělený jeho průměrem
i, j	imaginární jednotka, $\sqrt{-1}$
eps	nejmenší kladná hodnota, 2^{-52}
realmin	nejmenší číslo v pohyblivé řádové čárce, 2^{-1022}

pokračování Tabulka 5 Speciální proměnné v MathScript, dle IEEE	
symbolická proměnná	vysvětlení
realmax	největší číslo v pohyblivé řádové čárce, $(2\text{-eps})^{1024}$
inf	větší číslo než realmax
nan	výsledek nesmyslné operace, zpravidla dělení nulou

2.6.2 Naplnění proměnných

Pro přístup k proměnné je nutné, aby existovala v pracovní oblasti (Workspace). Proměnnou nadefinuji např. `global A;`, nebo `P = [];`. Takto definovaná proměnná ovšem nemá žádnou velikost (rozměr je 0x0), neukazuje tedy na žádné místo v paměti a nedá se s ní operovat.

Proměnnou naplníme uvedením jejího názvu, operátoru přiřazení, což je znak „=“ a hodnotou, např. `A = 5;`. Takto nadefinuji jednoduchou skalární proměnnou. Definice matice může proběhnout různými způsoby probranými na začátku. K definici textové proměnné se používá apostrofy uvozený text, např. `text = 'Ahoj.';`. V případě zapsání apostrofu do textového řetězce je nutné takto provést dvěma apostrofy za sebou. Pro uložení číselné hodnoty v pohyblivé řádové čárce se jako oddělovač používá desetinná tečka. Definice komplexní proměnné se provádí zapsáním složeného čísla. U imaginární části může být uvedeno „j“ nebo „i“.

2.6.3 Operace s proměnnými

Patří sem základní operace s proměnnými jednotlivých datových typů.

Tabulka 6 Aritmetické operátory v MathScript

Aritmetické operátory v MathScript		
operátor	význam	datový typ
+	součet	double
-	rozdíl	double
* .*	součin, součin po prvcích	double
/ ./	podíl, podíl po prvcích	double
\ .\	podíl, podíl po prvcích	double
^ .^	mocnina, mocnina po prvcích	double

Tabulka 7 Relační operátory v MathScript

Relační operátory v MathScript		
operátor	význam	datový typ
=	rovná se	double, char, logical
~=	nerovná se	double, char, logical
<	Je menší než	double, char, logical
>	je větší než	double, char, logical
<=	je menší než, nebo rovno	double, char
>=	je větší než, nebo rovno	double, char

Tabulka 8 Logické operátory v MathScript

Logické operátory v MathScript		
operátor	význam	datový typ
&&	logické vyhodnocení podmínky AND	logical
||	logické vyhodnocení podmínky OR	logical
&	logický součin argumentů	logical
|	logický součet argumentů	logical
~	negace argumentu (zapisuje se před proměnnou)	logical

2.6.4 Rozdíly v prostředí MATLAB

Pro znaky použité v názvu platí stejné podmínky jako v prostředí MathScript. Stejně tak lze nadefinovat jako proměnnou, název již existující funkce. V prostředí MATLAB však není možné, aby měla proměnná neomezenou délku. Není chyba ji takto nadefinovat, nicméně prostředí vezme jen prvních N znaků, přičemž N je hodnota, kterou vrací funkce „namelengthmax“ a je na různých systémech různá. Tato funkce existuje i v prostředí MathScript a funguje naprosto stejně, nicméně toto ji nepoužívá ke zkrácení délky názvu proměnné. I zde je k dispozici příkaz „isvarname“ ke kontrole názvu proměnné a stejně tak je prostředí MATLAB „Case-Sensitive“. Vše ostatní ohledně definování proměnných a jejich naplnění je totožné. Vzhledem k tomu, že prostředí MATLAB oplývá vícero datovými typy jsou některé z operátorů použitelné i pro ně.

2.7 FUNKCE

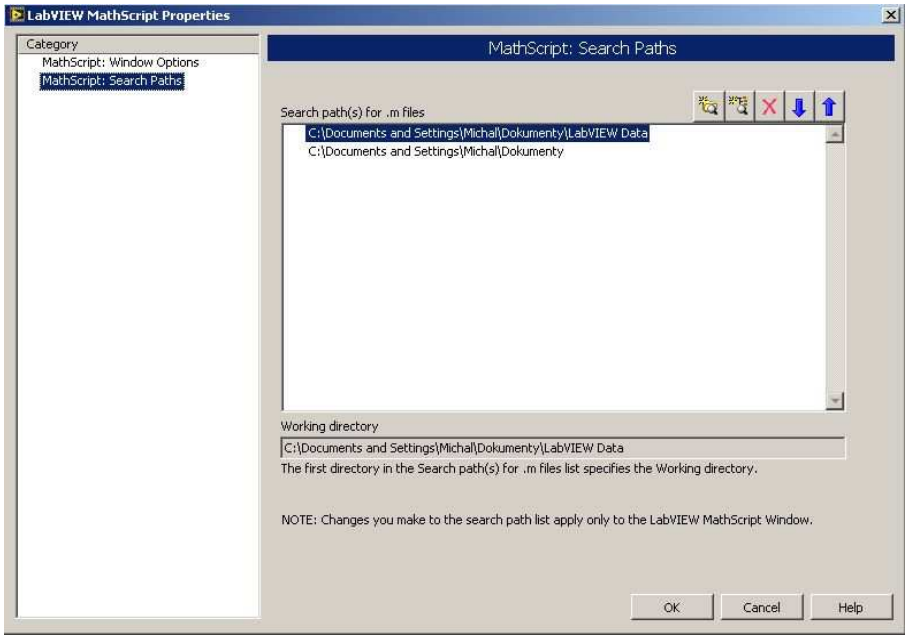
V prostředí MathScript se funkce dělí na tři druhy. Prvním druhem jsou tzv. „Core Functions“ implementovány v jádru programu poskytující nejvyšší efektivitu. Druhou skupinou jsou „Add-On Functions“ a poslední skupinou jsou „UD Functions“ (User-Defined). Příkazem `which -all;` jednoduše zjistíme, jaká funkce patří do jaké skupiny. Funkce je posloupnost příkazů, které vykonají požadovanou činnost. Jsou to speciálně zapsané m-skripty, kterým mohou být předávány vstupní parametry. Tyto m-skripty mohou zároveň vracet výstupní parametry. Uživatel může definovat funkce, které bude chtít použít v prostředí MathScript nebo v objektu MathScript Node. Měl by se však vyvarovat jistým stavům. Syntaxe uživatelem definované funkce má následující notaci

```
function [vystup1, …] = nazevfunkce(vstup1, …)
% Dokumentace k funkci
Příkazy…
```

K tomu, aby mohla být funkce spuštěna se musí předně napsat a uložit do souboru. Název souboru musí být shodný s názvem funkce a musí být uložen ve formátu „nazevfunkce.m“. Pro název funkce platí stejná pravidla jako pro název proměnné, tedy nesmí začínat podtržítkem, číslicí, mezerou a musí se skládat ze znaků UNICODE. Ke kontrole nám může pomoci implementovaná funkce „isvarname“ vracející logickou hodnotu. Syntaxe funkce uvedená výše se skládá z klíčového slova „function“, jež jazyku říká, že se jedná o funkci. Následuje výstup z funkce. Pokud má funkce pouze jediný výstup, nemusí být umístěn v závorkách. V případě více výstupů musí být uvedeny v hranatých závorkách a symbolické názvy proměnných odděleny čárkou nebo mezerou. Dalším znakem v syntaxi je znaménko „=“, název námi definované funkce a v kulatých závorkách jednotlivé symbolické proměnné jako vstupy oddělené čárkou. Samotná funkce nemusí mít žádný vstup, ani výstup. V tom případě je ji možno nadefinovat jako „function nazevpromenne()“. Pro řízení počtu vstupů a výstupů ve funkci je možno používat již definované funkce „nargin“ a „nargout“, jež vrací množství vstupů, či výstupů do/z právě probíhající funkce. Pod definicí funkce spolu s jejími vstupy a výstupy se dále umísťuje nápověda k funkci ve formě komentáře v textu. Komentáře v textu se značí znakem „%“ na začátku řádku. Dále pak již následuje kód, který chceme, aby se provedl při zavolání funkce. Je třeba však vzít na vědomí, že není možné volat funkce rekurzivně, či cyklicky (přes funkci, která zavolá opět tuto funkci). Ačkoliv prostředí MathScript umožní uživateli definovat, jako název funkce již funkci definovanou implicitně samotným programem, nedoporučuje se to. Uživatel totiž již nebude mít možnost přistupovat ke „staré“ funkci a veškeré operace, jako je např. volání nápovědy se bude vztahovat také již pouze k nové funkci.

Volání funkce se provádí uvedením jejího přesného názvu spolu s přesným počtem vstupů a výstupů do příkazového okna. Volání je ve své podstatě identické se syntaxí, vyjma klíčového slova „function“ na začátku. K tomu aby mohlo prostředí MathScript danou funkci provést, musí se uživatel postarat o to, aby byla uložena do pracovního adresáře. Seznam pracovních adresářů se nastavuje v sekci „FILE >> LabVIEW MathScript Properties“ v prostředí MathScript. Vybereme cestu k novému adresáři a vložíme ji do seznamu. Zbývá jen určit pořadí v jakém

prohledávat jednotlivé adresáře. Pak již nic nebrání námi volanou funkci provést. Při volání funkcí se postupuje následovně. Nejdříve program zjistí, zdali není volaná funkce umístěná v souboru „nazevfunkce.m“ uložena v paměti. Jestliže je v tomto hledání neúspěšný, zkontroluje adresář, kde byl naposledy uložen m-skript použitý ve VI souboru obsahujícím blok MathScript Node. Pokud ani v tomto případě není úspěšný, prohledá své pracovní adresáře (Search Paths) od prvního až po poslední uvedený. Jestliže ani v tomto případě není úspěšný, zobrazí uživateli do výstupního okna chybové hlášení o nenalezeném názvu funkce. Zjištění nám dává odpověď na otázku, co dělat v případě nadefinování vlastní funkce se shodným názvem již funkce definované. Můžeme odebrat adresář, v němž je uložena ze seznamu pracovních adresářů, odstranit ji nebo jednoduše přejmenovat. Potom již nic nebrání prostředí MathScript volat pod tímto názvem svoji původní funkci, kterou jsme mu přejmenovali.



Obrázek 2 Pracovní adresáře Search Paths

2.7.1 Rozdíly v prostředí MATLAB

Ve vývojovém prostředí MATLAB nacházíme několik rozdílů. Je zde především definice trošku jiná a to v tom, že funkce se tu dělí na dvě skupiny, tzv. funkce a skripty. Skript je jen posloupnost příkazů zapsaný do m-skriptu. Funkce je pak regulérně nadefinovaná funkce, jak je uvedeno výše. Dále se mluví o vestavěných funkcích tzv. „Built- In Functions“(podobné „Core Functions“) jejíž vykonávání je velmi efektivní, nicméně není možné prohlížet kód uživatelem, ani jej měnit. Druhou skupinou jsou „Implemented Functions“(analogie s „Add-On Functions“ a „User-Defined Functions“) u kterých je možno prohlížet a měnit kód funkce. Do této skupiny spadají i uživatelem definované funkce. V prostředí MathScript však není možnost nahlížet do jiných, než uživatelem definovaných funkcí. Při zadání jména takové funkce dojde k vytvoření nového skriptu s tímto názvem. Je možné definovat názvy funkcí podobě jako názvy proměnných, tedy nesmí začínat číslicí, nesmí začínat podtržítkem nebo mezerou. Délka proměnné je však omezena na délku, kterou vrací implementovaná funkce „namelenghtmax“. Syntaxe funkce je totožná se syntaxí v prostředí MathScript, avšak v prostředí MATLAB se může konec bloku příkazů funkce označit symbolickým označením „end“, což v prostředí MathScript vrací chybu. Opět je možno k řízení průběhu funkce použít implementované funkce „nargin“ a „nargout“. V prostředí MATLAB je dále možno definovat více typů funkcí. Prvním typem jsou tzv. anonymní funkce (Anonymous Functions). Tyto funkce není třeba vytvářet v m-skriptech, ale jednoduchou syntaxí přímo v příkazovém okně. Definice je následující

```
nazevfunkce = @(argumenty) operace;
```

Pak již můžeme tuto funkci zavolat, tak jak jsme zvyklí s požadovaným počtem argumentů. Je určena především pro jednoduché výrazy, kdy by bylo zbytečně složité definovat celý m-skript. Takto definovat funkce v prostředí MathScript nelze. Dále lze vytvářet ukazatele na funkce (Function Handle) a těmi je pak volat následujícím způsobem.

```
handle_na_funkci_sinus = @sin;
sin_45_stupnu = handle_na_funkci_sinus(pi/4);
```

Takto volat funkce pomocí ukazatele na funkci rovněž není v prostředí MathScript možné.

2.8 ŘÍZENÍ BĚHU PROGRAMŮ A ŘÍZENÍ CYKLŮ

Pro řízení cyklů a běhu programu v prostředí MathScript je na výběr hned z několika možností. Každá je vhodná pro různé aplikace různým způsobem.

2.8.1 SWITCH-CASE

Prvním takovým příkazem pro řízení běhu programu je příkaz „switch-case“. Příkaz vyhodnocuje zvolenou proměnnou a na základě její hodnoty provádí definované příkazy. Syntaxe je následující

```
switch testovanapromenna
      case 'A'
      Příkazy…
      case 'B'
      Příkazy2…
      otherwise
      Příkazy3…
end
```

Jednotlivé příkazy „case“ nemusí být zakončeny příkazem „break“ pro vyskočení z těla cyklu, jako je tomu například v programovacím jazyku C. Program totiž při první shodě testované proměnné vykoná přikazy uvnitř daného „case“
a zbytek přeskočí. Další příkazy „case“ nejsou povinné, stejně tak není povinná volba „otherwise“, která se vykoná, když žádný z předchozích testovaných „case“ nebyl úspěšně vykonán. Je možné porovnávat pouze stejný datový typ, tzn., jestliže

si vyberu k porovnávání znakový typ „char“, musí být všechny volby označeny 'testovanavolbajechar' (v apostrofech) a nelze je kombinovat s dalšími datovými typy, např. číselný typ „double“. Program jinak vrací chybové hlášení.

2.8.2 Cyklus FOR

Dalším příkazem na řízení cyklu je příkaz „for“. Tento se používá, známe-li přesný počet opakování části programu, který tento příkaz cyklicky vykonává. Jeho syntaxe je následující

```
for v = x : y : z
Příkazy…
end
```

Řídící proměnná v tomto případě „v“ se od proměnné „x“ znakového typu „char“ nebo číselného typu „double“ až do proměnné „z“ stejného datového typu po krocích „y“ inkrementuje, případně dekrementuje v závislosti na tom, jak jsou uvedeny meze. Tento způsob definice vektoru již byl uveden v základech výše.
V případě, že se proměnná dekrementuje, musí být uveden krok se záporným znaménkem. V případě inkrementace není potřeba kladné znaménko uvádět. Krok může nabývat pouze číselných hodnot a jeho implicitní hodnota je 1.

2.8.3 IF-ELSE

Další řídícím algoritmem jsou příkazy „if-else“, často používané pro jednoduché rozhodování. Zápis má následující syntaxi

```
if vyraz == 3
      q = 44100;
elseif vyraz == 4
      q = 11800;
else
      q = 0;
end
```

Příkaz se používá k logickému vyhodnocení nějakého konkrétního zápisu. Je možno použít libovolný z logických vyhodnocovacích operátorů použitelných v jazyce MathScript (~=, ==, <, >, <=, >=). Je možno vyhodnocovat argumenty znakových, číselných nebo logických proměnných. Další příkazy jako „elseif“ nebo „else“ nejsou povinné, často se ale používají pro zvýšení čitelnosti a komfortu výrazu.

2.8.4 WHILE

Dalším příkazem, jímž můžeme řídit nějaký cyklus je příkaz „while“. Tento příkaz probíhá cyklicky do té doby, dokud jeho testovaná proměnná nabývá pravdivých hodnot. Syntaxe je následující

```
while vyraz ~= 'A'
Příkazy…
end
```

Příkaz „while“ je dalším, který vyhodnocuje logickou hodnotu uvedeného výrazu a pokud je pravdivá provede příkazy ve svém těle. Tento příkaz se explicitně používá všude tam, kde není přesně znám počet opakování cyklu na rozdíl od cyklu typu „for“.

2.8.5 CONTINUE, BREAK, RETURN

Speciálními příkazy řídícími cykly jsou příkazy „continue“, „break“ a „return“. První dva příkazy se používají jako doplňky příkazů „for“ a „while“. Příkaz „continue“ přeskočí všechny příkazy následující za ním a skočí před nejbližší strukturovaný konec cyklu (příkaz „end“). Může se takto začít další iterace cyklu. Příkaz „break“ také přeskočí všechny příkazy následované za ním, ale skočí až za nejbližší strukturovaný konec cyklu a tím přeruší jeho vykonávání. Příkaz „return“ vrátí řízení zpět volajícímu procesu funkce, dojde tak k předčasnému ukončení celé funkce. Příkaz „return“ se nemusí používat pouze s příkazy „for“ a „while“. Tyto příkazy jsou často využívány.

2.8.6 Rozdíly v prostředí MATLAB

Oproti příkazům použitých v prostředí MathScript tu nejsou v podstatě žádné změny a základní struktura a syntaxe příkazů řídících běh programu a cyklů zůstává zachována. Pokud jde o nějakou závažnější změnu týká se především příkazu „switch-case“. V prostředí MathScript není možno porovnávat proměnnou s různými datovými typy v jednom „switch“ příkazu. V prostředí MATLAB toto možné je. Můžeme porovnávat jednu vstupní proměnnou se znakovou proměnnou i číselnou v jednom příkazu „switch“.

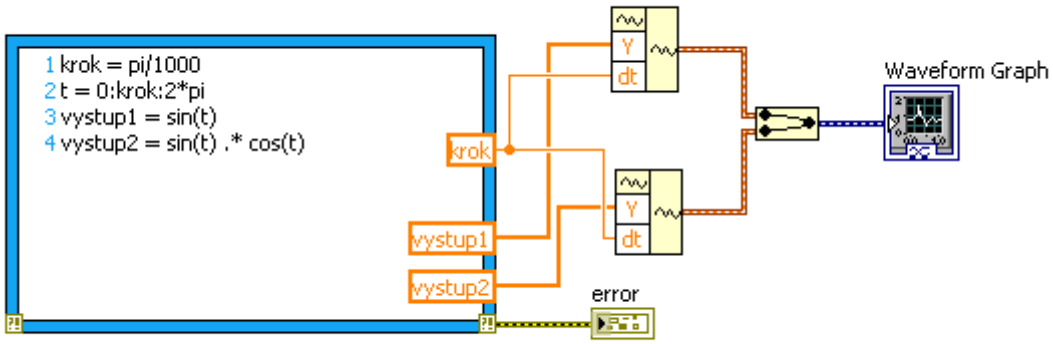
Další změnou je výskyt příkazu řídícího cyklus programu. Je jím příkaz „trycatch“. Ve skutečnosti by se tento dal nahradit kombinací „while“ a „if“ příkazů, takže se nejedná o unikátní způsob řízení cyklu, ale prostředí MATLAB jej uvádí a proto jej zde uvedu také. Syntaxe cyklu je následující.

```
try
Příkazy…
catch
Příkazy2…
end
```

Jeho funkcí je provést příkazy mezi symbolickými názvy „try“ a „catch“. Pokud se v této oblasti vyskytne jakákoliv chyba provedou se příkazy v bloku „catch-end“. Pokud se i v těchto vyskytne chyba program se ukončí s daným chybovým hlášením. Generovanou chybu lze získat pomocí příkazu „lasterror“ a případně ji opravit v části bloku „catch-end“.

2.9 MATHSCRIPT NODE

Již několikrát byl v textu letmo zmíněn pojem MathScript Node, nebyl však prozatím objasněn. MathScript Node je struktura vývojového prostředí LabVIEW určená k vykonání příkazů jazyka MathScript v tomto prostředí.



Obrázek 3 Propojení prvku MathScript Node v prostředí LabVIEW

MathScript Node může a nemusí obsahovat vstupy, výstupy. Tyto však potom musí uživatel zadat ručně, stejně tak jako zvolit jejich datový typ kvůli přechodu MathScript-LabVIEW, LabVIEW-MathScript. Do těla MathScript Node potom umístí příkazy v souladu s konvencí a syntaxí příkazů jazyka MathScript. Příkazy může uživatel importovat z již napsaného m-skriptu nebo napsat ručně. Ty lze zpětně exportovat do souboru. Z MathScript Node se dají volat i funkce napsané v prostředí MathScript, musí však být umístěny ve stejném projektu. Datové typy, ze

kterých může uživatel volit jsou číselný typ „double“, „complex double“, logický typ „boolean“ a řetězcový typ „string“ buď jako skalár nebo jako vektor. Výstupní vektor je implicitně vždy sloupcový. V případě matic se rozlišuje jen reálná (Real) nebo komplexní (Complex) matice. Dále umožňuje MathScript Node zvolit a operovat i se systémem vyjádřeném ve stavovém prostoru, ve tvaru operátorového přenosu nebo s vyjádřením pomocí nul a pólů. To však platí pouze v případě, že je instalován „Control Design Toolkit“, což je rozšiřující sada pro řízení a operace se systémy. Některé funkce a příkazy však nejsou v MathScript Node zcela nebo vůbec podporovány. Jedná se především o příkazy pro práci s adresáři, cestami a okny. Tyto jsou charakteristické čistě pro prostředí MathScript, kde mají své opodstatnění. V případě těchto funkcí může dojít k chybě při překladu, značnému zpomalení běhu programu nebo odmítnutí vytvořit samostatnou aplikaci. Naprostá většina běžně využívaných příkazů však funguje bez problémů.

2.10 DOSTUPNOST MATHSCRIPT

Prostředí MathScript není plně funkční ve všech vývojových verzích LabVIEW. Plně funkční je ve verzích „LabVIEW Developer Suite“ a „LabVIEW Student Edition“. Plně funkční, avšak bez „Control Design Toolkit“ pro řízení a operace se systémy je dále ve verzích „LabVIEW Full Development“, „LabVIEW Professional Development“ a „LabVIEW Evaluation“.

Jazyk MathScript je navíc dostupný i online přes webové rozhraní jako volně použitelná beta verze s omezenými funkcemi a to na adrese <http://netdemo.ni.com/mathscript/Default.aspx>. Takto si může každý volně odzkoušet některé funkce tohoto nástroje.

Přes webové rozhraní je navíc volně dostupné i samotné prostředí LabVIEW ve verzi „Professional Development“. Velká nevýhoda ovšem je, že z jedné IP adresy takto dovolí systém pracovat uživateli ve vývojovém prostředí pouhé tři hodiny. Navíc v prostředí MathScript nelze ukládat a spouštět vlastní funkce. Online

přístupné vývojové prostředí LabVIEW je na adrese <http://www.ni.com/trylabview>.

2.11 MATLAB SCRIPT NODE

Vývojové prostředí LabVIEW od svých předchozích verzí disponuje nástrojem jménem „MATLAB Script Node“. Pomocí něj se dají provádět jakékoliv příkazy prostředí MATLAB a předávat výsledky do prostředí LabVIEW k dalšímu zpracování.

Nevýhoda tohoto způsobu ovšem je ta, že je třeba mít na stejné stanici, z jaké se tento blok volá licencováno i prostředí MATLAB a to ve verzi minimálně 6.5. Prostředí LabVIEW pracuje tak, že předá příkazy interpretované v tomto bloku skriptovacímu serveru prostředí MATLAB pomocí ActiveX spojení, ten příkazy vykoná a výsledky odešle zpět volajícímu prostředí LabVIEW. Díky technologii ActiveX je ovšem možné používat jej pouze ve verzích LabVIEW určených pro operační systém Windows.

Vytvářet MATLAB Script Node lze pouze ve verzích „LabVIEW Professional Development“ a „LabVIEW Full Development“. V případě, že je program s touto vnitřní strukturou spuštěn na jiné než Full nebo Professional verzi, je možné jej úspěšně vykonat, splňuje-li pracovní stanice předpoklad ohledně verze a licence prostředí MATLAB, jak je uvedeno výše.

Jinak se MATLAB Script Node blok chová stejně jako blok MathScript Node. Stejným způsobem se mu určují potřebné vstupy, výstupy, volí stejný datový typ kvůli přechodu LabVIEW-MATLAB, MATLAB-LabVIEW. Stejným způsobem se i píší, importují a exportují již jednou napsané m-skripty.

Celkově se však tento způsob spouštění m-skriptů v prostředí LabVIEW nedá považovat za optimální z důvodu toho, že je závislý na dalším licencovaném softwaru. Řešení je ve stále větší podpoře a zdokonalování prostředí MathScript, které si dokáže též s m-skripty poradit.

2.12 MODULARITA PROSTŘEDÍ

Prostředí LabVIEW je prostředím modulárním. Je možné jej libovolně rozšiřovat o specializované sady funkcí a nástrojů dle individuálních požadavků každého uživatele. Jedná se o tzv. Toolboxy nebo Toolkity.

Placenými rozšiřujícími Toolkity přímo od výrobce pro prostředí MathScript jsou např. LabVIEW Control Design and Simulation Module a Digital Filter Design Toolkit, které přidávají nové textové funkce a možnosti pro práci se systémy a soustavami, pro práci s filtry a jejich návrhy. LabVIEW Control Design and Simulation Module přidává mimo nových funkcí do prostředí MathScript i nástroj pro simulace podobné prostředí Simulink, od společnosti MathWorks. Tento nástroj umožňuje zcela autonomní simulační funkce v prostředí LabVIEW podobné jako v prostředí Simulink. Zároveň umožňuje konverzi modelů a soustav vytvořených v prostředí Simulink do LabVIEW. Dalším simulačním rozšířením je LabVIEW Simulation Interface Toolkit. Tento poskytuje propojení mezi prostředím Simulink a Real-Time Workshop vytvořené společností MathWorks a prostředím LabVIEW. Umístí do knihovny bloků Simulink nový blok, který poskytuje virtuální propojení přes knihovny s prostředím LabVIEW. Na základě toho pak může Simulink poskytovat své prvky a rozhraní prostředí LabVIEW. Nevýhodou tohoto uspořádání může být opět fakt, že je třeba licencované verze dalšího programu.

Výrobce poskytuje desítky dalších Toolkitů k zakoupení, šité na míru různým potřebám uživatelů. Výrobce sice umožňuje otestovat tyto Toolkity vzdáleně přes internet na svých počítačích, ale pouze na dobu tří hodin z jedné IP adresy, kdy je dovoleno uživateli připojení.

Na internetu však existuje i řada uživatelů, kteří poskytují své funkce a rutiny volně ke stažení. Příkladem může být portál OpenG.org [11]. Přes jednoduchého klienta VIPM (VI Package Manager) lze z internetu stáhnout dostupné rozšiřující funkce doslova jedním kliknutím myši a to vše zdarma. Příspěvky jsou samozřejmě ze strany autorů vítány. Tento přístup je společností National Instruments tolerován a klient VIPM doporučují i na svých stránkách.

3. MATHSCRIPT A MATLAB

3.1 SROVNÁNÍ FUNKCÍ

3.1.1 Přehled tříd funkcí v prostředí MathScript

V prostředí LabVIEW MathScript je v současné době více jak 650 funkcí použitelných pro nejrůznější konstrukce uživatelských programů. Tyto funkce jsou přehledně rozděleny do tříd. V každé takové třídě se nacházejí funkce, jež mají z hlediska použití nebo určení něco společné. K zobrazení podrobností o těchto třídách slouží příkaz `help classes` spuštěný v prostředí LabVIEW MathScript. Obsah funkcí se však odvíjí i od rozšiřujících Toolboxů. Příkladem mohou být LabVIEW Control Design and Simulation Module nebo Digital Filter Design Toolkit, které přidávají funkce prostředí MathScript, ale i prostředí LabVIEW spolu s dalšími možnostmi.

Tabulka 9 Třídy funkcí v prostředí LabVIEW MathScript

Třídy funkcí v prostředí LabVIEW MathScript	
třída	popis, funkce
advanced	obsahuje především pokročilé matematické funkce a funkce pro řešení Besselových rovnic
approximation	obsahuje funkce pro aproximaci a interpolaci hodnot
audio	funkce sloužící pro práci se zvukovými soubory
basic	funkce pro běžně užívané matematické výpočty a transformace, trigonometrické funkce mají svoji vlastní třídu
bitwise	třída sdružující funkce pro bitové operace
boolean	třída s funkcemi sloužícími pro logické vyhodnocování výrazů
commands	funkce umožňující uživateli přizpůsobovat vzhled okna LabVIEW MathScript Window

pokračování Tabulka 9
Třídy funkcí v prostředí LabVIEW MathScript

třída		popis, funkce
comparison		relační operátory sloužící pro vyhodnocování podmínek
constants		konstanty a speciální proměnné dle IEEE využívané prostředím
daq		funkce sloužící pro sběr dat a práce se zařízením NI-DAQmx
dsp	filter design	funkce sloužící pro návrh a tvorbu filtrů pro práci s digitálními signálními procesory
	filter implementation	použití a aplikace s navrženými filtry
	linear systems	práce s lineárními systémy, stavový prostor, přenosové funkce
	modelling and prediction	funkce pro modelování a identifikaci systémů
	resampling	funkce pro úpravu signálů
	spectral analysis	funkce pro spektrální analýzu signálů
	transforms	funkce pro transformaci signálů
	waveform generation	funkce sloužící pro vytváření signálů
	windows	sdružuje funkce pro práci s časovými okny
geometry		funkce pro tvorbu geometrických obrazců
integration		obsahuje funkce pro integrální počet
libraries		funkce pro práci s knihovnami
linalgebra		obsahuje funkce lineární algebry pro práci s maticemi
matrix		funkce pro tvorbu speciálních typů matic
matrixops		funkce pro unární a logické vyhodnocování matic
membership		obsahuje funkce pro logické vyhodnocování příslušnosti nebo podmínek argumentů jednotlivým skupinám
ode		sdružuje funkce pro řešení obyčejných diferenciálních rovnic
optimization		funkce pro minimalizaci polynomů
plots		funkce pro vykreslování grafů a objektů
polynomials		funkce pro práci s polynomy
programming		funkce pro řízení běhu a cyklů uživatelských programů
sets		funkce pro práci s množinami
statistics		sdružuje funkce pro práci se statistikou a interpretaci jejích výsledků
string		funkce pro práci s řetězci

pokračování Tabulka 9 Třídy funkcí v prostředí LabVIEW MathScript	
třída	popis, funkce
support	obsahuje funkce pro práci s adresáři a soubory, úprava a konverze datových typů, práce s proměnnými a prostředím LabVIEW MathScript Window
time	funkce pro práci s datem a časem
timing	obsahuje funkce monitorující především běh programu a časování
trigonometric	obsahuje funkce pro trigonometrii
vector	obsahuje funkce pro práci s vektory
zerofinder	funkce pro řešení nelineárních rovnic

3.1.2 Odlišnost funkcí v prostředí MathScript

Dle tříd má prostředí LabVIEW MathScript funkce rozděleny viz výše. Nicméně je třeba zjistit, zdali všechny funkce obsažené v jednotlivých třídách mají stejné vlastnosti, co se odlišnosti týká, zkrátka jestliže vykonávají to samé jako v prostředí MATLAB.

Z tohoto důvodu jsem dále funkce v prostředí LabVIEW MathScript rozdělil na tři jednoduché skupiny.

- funkce, které jsou stejné jako funkce v prostředí MATLAB, zkráceně „totožné funkce“
- funkce, které mají omezené použití oproti funkcím v prostředí MATLAB, zkráceně „odlišné funkce“
- funkce, které nejsou v prostředí MATLAB standardně obsaženy, zkráceně „chybějící funkce“

3.1.3 Totožné funkce

Jsou to funkce, které jsou zcela identické v obou prostředích, jak svým významem a početním algoritmem, tak svojí syntaxí. Tyto jsou umístěny jako Příloha 1. Z hlediska použití těchto funkcí v přenášených programech jedním

(MATLAB → MathScript) nebo druhým (MathScript → MATLAB) směrem, nemusí mít uživatel žádné starosti s kompatibilitou příkazů z hlediska jejich významu nebo užité syntaxe. Těchto funkcí je v současnosti více jak 500 z uvedených 650. Přesto je však dobré vždy dbát pokynů výrobce. National Instruments má v tomto případě výhrady k funkci „expm1“ a doporučuje místo ní raději používat funkci „expm“. Důvody nejsou dále nijak vysvětleny.

3.1.4 Odlišné funkce

Mezi tzv. „odlišné funkce“ patří funkce, jež nejsou z hlediska významu v obou prostředích nijak rozdílné, avšak jejich syntaxe je různá. Jsou odlišná v různých rozšiřujících přepínačích a přídavných možnostech, které funkce jednoho nebo druhého prostředí poskytuje. Jen ve výjimečných případech se jedná o kompletně zcela odlišnou syntaxi funkce, základní volání zůstává totožné v obou prostředích. Význam jednotlivých funkcí zůstává ve všech případech zachován. Příkladem funkce se zcela odlišnou syntaxí může být např. příkaz „funm“. Prostředí MATLAB nabízí volání pomocí ukazatele na funkci (Function Handle). Jak bylo zmíněno v základech výše, prostředí MathScript takovéto odkazování na funkce nepodporuje a jeho volání externích funkcí je čistě přes řetězcovou proměnnou.

U všech těchto funkcí je nutná obezřetnost při přenositelnosti programů mezi jedním či druhým prostředím. Nejlepší způsob jak dosáhnout informací o konkrétním omezení té, či oné funkce je vyhledáním v nápovědě příkazem `help jmenofunkce` jak v jednom, tak v druhém prostředí. Jak již bylo zmíněno výše odlišnosti jsou především v přidaných přepínačích funkcí. Základní syntaxe zůstává zpravidla stejná. Tyto funkce jsou umístěny v Příloha 2.

3.1.5 Chybějící funkce

Mezi poslední skupinu příkazů, patří ty, které nejsou standardně obsaženy v prostředí MATLAB. Standardně píši proto, že není problém si takovou funkci napsat

jako uživatelskou a používat ji. V standardním prostředí MATLAB používané verze však funkce chybějí. Jedná se především o specifické funkce prostředí LabVIEW. Zde doporučuje National Instruments používat raději funkci „expm“ místo funkcí „expm2“ a „expm3“. Důvody jsou stejné jako v případě funkce „expm1“ hlouběji nejsou vysvětleny. Chybějící funkce jsou obsaženy v Příloha 3.

3.1.6 Zastaralé funkce

V prostředí MATLAB se stále vyskytují funkce, které jsou funkční, avšak považují se za zastaralé a zůstávají pouze z důvodu kompatibility. Není vůbec vyloučeno, že v některé z příštích verzí prostředí MATLAB budou vypuštěny úplně. Funkce, které se považují za zastaralé jsou vypsány do tabulky.

Tabulka 10 Zastaralé funkce prostředí MATLAB

Zastaralé funkce prostředí MATLAB	
původní funkce	bude nahrazena
clg	clf
cohere	mscohere
csd	cpsd
firgauss	gaussfir
path2rc	savepath
psd	pwelch
remez	firpm
remezord	firpmord
setstr	char
tfe	tfestimate

Mezi zastaralé funkce patřila např. i funkce „quad8“, která je již vypuštěna úplně a nahrazena funkcí „quadl“.

3.1.7 Funkce omezené v bloku MathScript Node

Všechny funkce v sekcích výše uvedených jsou dle svých specifikací plně funkční v prostředí LabVIEW MathScript. Mají však jistá omezení vzhledem k bloku MathScript Node sloužící k vykonávání výše zmíněných textových příkazů v grafickém programovacím jazyku vývojového prostředí LabVIEW. Samostatně spustitelné aplikace nebo sdílené knihovny obsahující MathScript Node s nějakou z těchto funkcí potom nelze spustit či použít. Nejsou totiž podporovány v tzv. „Run-Time Engine“, což je soubor knihoven a ostatních prvků potřebných ke spuštění samostatné aplikace na počítači. Jsou však podporovány v grafickém vývojovém prostředí LabVIEW. Tyto funkce jsou obsaženy v Příloha 4.

Některé z těchto funkcí ovlivňují jen ten MathScript Node, ve kterém jsou spuštěny, jiné naopak i všechny okolní v projektu. Důležité informace a doporučení, stejně tak jako příklady poskytne nápověda `help jmenofunkce`.

3.2 MODULÁRNÍ PRVKY

3.2.1 Control Design and Simulation Module

Tento rozšiřující modul přidává další funkce a možnosti nejen samotnému prostředí LabVIEW, ale také prostředí MathScript. Ačkoliv jsem jej neměl k dispozici, tak díky bohaté nápovědě umístěné na stránkách výrobce jsem byl schopen vypracovat seznam nových tříd a funkcí, které modul ve verzi 8.5 přidává.

Tento modul se zaměřuje, jak již jeho název sám napovídá, především na operace s regulační smyčkou. Návrhy regulátorů, testování systémů a soustav, atd.

Tabulka 11 Třídy funkcí MathScript v Control Design and Simulation Module

Třídy funkcí MathScript v Control Design and Simulation Module	
třída	popis, funkce
cdsolvers	pro řešení spojitých i diskrétních Ljapunovských a Riccatiho rovnic
cdutil	třídění, změna a tvorba parametrů filtrů
connect	pro vzájemné propojení systémů
construct	pro tvorbu LTI modelů a převod mezi vyjádřeními modelů
convert	převod mezi spojitými a diskrétními systémy
dynchar	výpočet dynamických parametrů systémů
frqrsp	analýza systémů ve frekvenční oblasti
info	informace o systémech
reduce	eliminace stavů systémů a zjednodušení systémů
ssanals	výpočet vlastností systému ve stavovém prostoru
ssdesign	nastavování a testování parametrů systémů ve stavovém prostoru
timeresp	generování signálů v časové oblasti

Nových funkcí je přibližně 100. Jsou rozdělené do kategorií shodných, rozdílných a chybějících oproti prostředí MATLAB a jsou uvedené jako Příloha 5. Funkce „set“ a „get“ zde navíc definují nové vlastnosti pro systémy.

Tabulka 12 Vlastnosti soustav

Vlastnosti soustav		
označení vlastnosti	popis	přípustné hodnoty
num	čitatel přenosu soustavy	pouze pro použití s přenosovou funkcí (tf), může být skalár, či vektor reálných hodnot
den	jmenovatel přenosu soustavy	
z	nuly systému	pouze ve vyjádření pomocí nul a pólů soustavy (zpk) z, p jsou vektory komplexních čísel k je skalár, či matice reálných čísel
p	póly systému	
k	zesílení ZV smyčky	

pokračování Tabulka 12 Vlastnosti soustav		
označení vlastnosti	popis	přípustné hodnoty
a	matice ZV A	pouze pro vyjádření systému ve stavovém prostoru (ss) a může být skalár, či matice reálných hodnot b, c, d skalár či vektor reálných hodnot
b	matice vstupů B	
c	matice výstupů C	
d	dopředná matice D	
ts	perioda vzorkování	skalár reálných hodnot
inputdelay	vstupní zpoždění	vektor reálných hodnot
outputdelay	výstupní zpoždění	
iodelay	zpoždění soustavy	matice reálných hodnot
transdelay		

3.2.2 Rozdíly v prostředí MATLAB

Rozdílů v prostředí MATLAB není mnoho, avšak patří mezi ně možnost zobrazit a měnit operátor přenosu, pojmenování vstupů/výstupů, jmenovka soustavy, ale třeba i poznámky, jež je možné do popisu soustavy taktéž umístit.

3.2.3 Digital Filter Design Toolkit

Stejně jako předchozí rozšiřující modul, ani tento jsem neměl k dispozici, ale podobně jako v předchozím případě není závažnější problém využít bohatou nápovědu na stránkách výrobce.

Tento rozšiřující Toolkit, ve verzi 8.2, slouží pro návrhy filtrů a operace s nimi.

Tabulka 13 Třídy funkcí pro MathScript v Digital Filter Design Toolkit

Třídy funkcí pro MathScript v Digital Filter Design Toolkit	
třída	popis, funkce
multirate	pro návrh Nyquistova filtru
singlerate	pro návrhy FIR a IIR filtrů

Nových funkcí oproti předchozímu rozšíření není mnoho, ale i tak mohou uživateli usnadnit práci. Opět jsou rozděleny na totožné a rozdílné funkce oproti prostředí MATLAB a jsou uvedeny jako Příloha 6.

3.3 SROVNÁNÍ POČETNÍCH OPERACÍ S MATICEMI

3.3.1 Testovací program

Vzhledem k tomu, že prostředí MathScript stejně tak jako prostředí MATLAB, jsou programy orientovány na operace s maticemi, je záhodno podrobit zkoušce právě tuto tezi. V těle tohoto programu se nejedná o nic jiného, než o vytvoření čtvercové matice A rozměrů NxN, inicializované náhodnými hodnotami, přičemž N je každou iteraci zvětšováno. S touto maticí A jsou každou iteraci provedeny úkony A^2 a změřena doba vykonání příkazu. Následují již jen informace pro operátora, které nejsou v časovém měření již zahrnuty. Později jsem zvolil i operace s maticemi A^4 a A^{16} pro lepší ilustraci rozdílu. Program je obsažen jako Příloha 7.

3.3.2 Výsledky

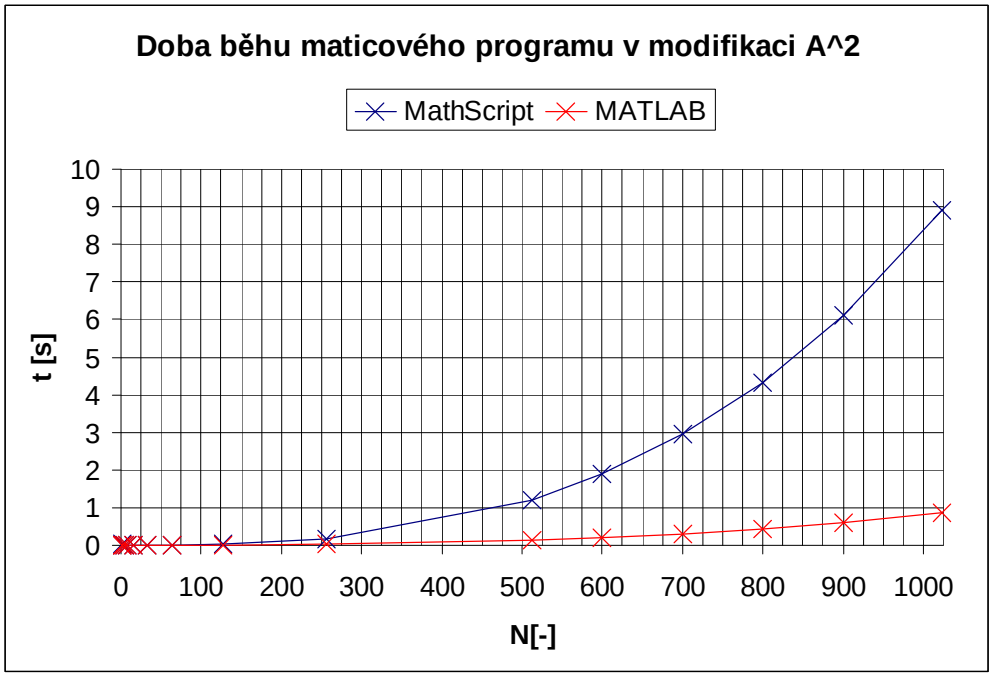
Výsledky testovacího programu jsou zpracovány jak tabelárně, tak graficky. Operuje se střední hodnotou výběrového souboru hodnot $\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i$.

Tabulka 14 Doba výpočtu v prostředí MathScript, modifikace A^2

Doba výpočtu v prostředí MathScript, modifikace A^2						
N [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1	0,026	0,038	0,039	0,038	0,037	0,036
2	0,002	0,002	0,001	0,002	0,002	0,002
4	0	0	0	0,001	0,001	0
8	0,001	0,001	0	0	0,001	0,001
16	0	0	0,001	0,001	0	0
32	0,002	0,002	0,001	0	0,001	0,001
64	0,004	0,006	0,005	0,005	0,005	0,005
128	0,028	0,03	0,029	0,029	0,028	0,029
256	0,172	0,172	0,172	0,172	0,173	0,172
512	1,208	1,209	1,21	1,207	1,209	1,209
600	1,954	1,893	1,895	1,886	1,885	1,903
700	2,975	2,959	2,949	2,95	2,946	2,956
800	4,363	4,327	4,313	4,307	4,311	4,324
900	6,152	6,103	6,085	6,095	6,097	6,106
1024	9,018	8,896	8,888	8,88	8,903	8,917

Tabulka 15 Doba výpočtu v prostředí MATLAB, modifikace A^2

Doba výpočtu v prostředí MATLAB, modifikace A^2						
N [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1	0	0	0	0	0	0
2	0	0	0	0	0	0
4	0	0	0	0	0	0
8	0	0	0	0	0	0
16	0	0	0	0	0	0
32	0	0	0	0	0	0
64	0	0	0	0	0	0
128	0,002	0,002	0,002	0,002	0,002	0,002
256	0,016	0,017	0,016	0,019	0,016	0,017
512	0,125	0,12	0,125	0,118	0,118	0,121
600	0,195	0,186	0,188	0,186	0,185	0,188
700	0,297	0,288	0,288	0,288	0,288	0,290
800	0,504	0,422	0,421	0,422	0,421	0,438
900	0,611	0,596	0,598	0,596	0,596	0,599
1024	0,872	0,876	0,868	0,869	0,868	0,871



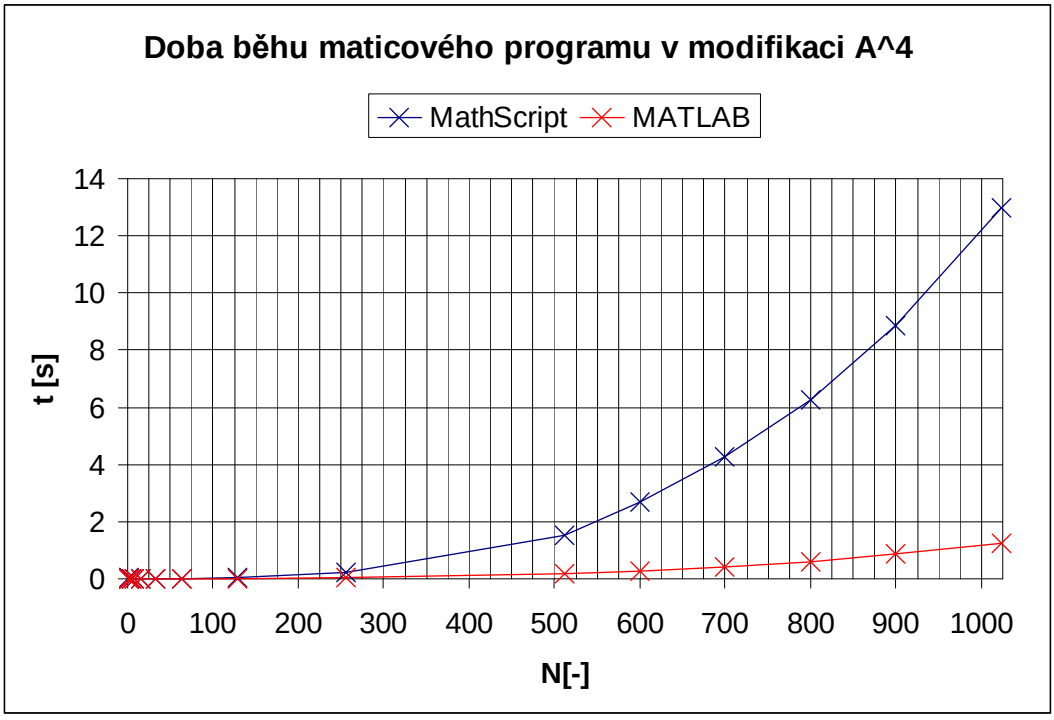
Obrázek 4 Doba běhu maticového programu v modifikaci A^2

Tabulka 16 Doba výpočtu v prostředí MathScript, modifikace A^4

Doba výpočtu v prostředí MathScript, modifikace A^4						
N [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1	0,038	0,039	0,039	0,039	0,038	0,039
2	0,002	0,002	0,003	0,002	0,002	0,002
4	0	0,001	0	0,001	0,001	0,001
8	0,001	0	0,001	0	0	0
16	0	0,001	0	0,001	0	0
32	0,002	0,001	0,002	0,001	0,002	0,002
64	0,005	0,006	0,005	0,007	0,006	0,006
128	0,037	0,036	0,037	0,036	0,036	0,036
256	0,236	0,234	0,235	0,234	0,236	0,235
512	1,718	1,717	0,719	1,719	1,719	1,518
600	2,754	2,697	2,695	2,697	2,694	2,707
700	4,268	4,238	4,233	4,234	4,237	4,242
800	6,259	6,229	6,236	6,245	6,245	6,243
900	8,874	8,835	8,833	8,854	8,841	8,847
1024	12,98	12,97	12,97	12,97	12,97	12,972

Tabulka 17 Doba výpočtu v prostředí MATLAB, modifikace A^4

Doba výpočtu v prostředí MATLAB, modifikace A^4						
N [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1	0	0	0	0	0	0
2	0	0	0	0	0	0
4	0	0	0	0	0	0
8	0	0	0	0	0	0
16	0	0	0	0	0	0
32	0	0	0,002	0	0	0
64	0,001	0,001	0,001	0,001	0,001	0,001
128	0,003	0,003	0,003	0,003	0,003	0,003
256	0,022	0,022	0,022	0,025	0,025	0,023
512	0,175	0,175	0,175	0,171	0,17	0,173
600	0,293	0,271	0,269	0,269	0,269	0,274
700	0,433	0,42	0,42	0,42	0,42	0,423
800	0,631	0,618	0,617	0,617	0,621	0,621
900	0,889	0,875	0,875	0,877	0,875	0,878
1024	1,27	1,273	1,278	1,278	1,275	1,275



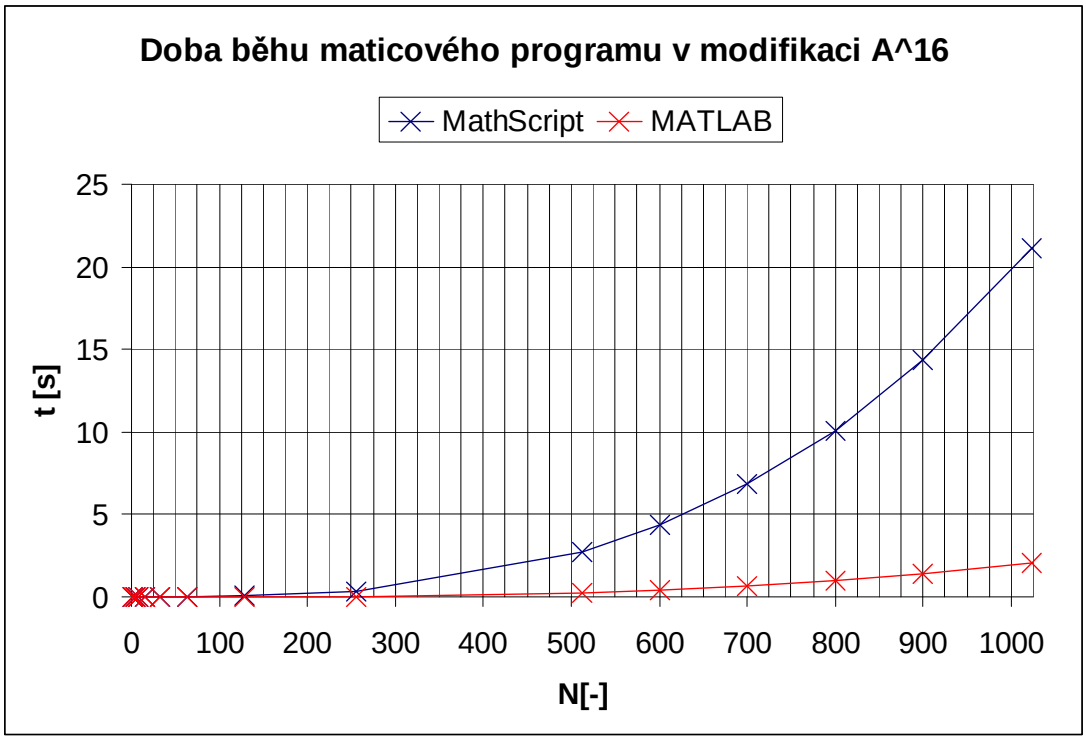
Obrázek 5 Doba běhu maticového programu v modifikaci A^4

Tabulka 18 Doba výpočtu v prostředí MathScript, modifikace A^{16}

Doba výpočtu v prostředí MathScript, modifikace A^{16}						
N [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1	0,04	0,038	0,038	0,039	0,041	0,039
2	0,002	0,001	0,002	0,002	0,002	0,002
4	0,001	0,001	0	0,002	0	0,001
8	0	0	0	0,001	0,001	0
16	0,001	0,001	0	0	0	0
32	0,001	0,001	0,002	0,002	0,002	0,002
64	0,008	0,008	0,008	0,008	0,008	0,008
128	0,051	0,053	0,053	0,053	0,052	0,052
256	0,361	0,364	0,361	0,362	0,362	0,362
512	2,731	2,732	2,729	2,734	2,73	2,731
600	4,376	4,331	4,328	4,33	4,325	4,338
700	6,866	6,842	6,841	6,841	6,839	6,846
800	10,148	10,098	10,095	10,095	10,091	10,105
900	14,389	14,367	14,365	14,363	14,385	14,374
1024	21,09	21,09	21,08	21,08	21,09	21,086

Tabulka 19 Doba výpočtu v prostředí MATLAB, modifikace A^{16}

Doba výpočtu v prostředí MATLAB, modifikace A^{16}						
N [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1	0	0	0	0	0	0
2	0	0	0	0	0	0
4	0	0	0	0	0	0
8	0	0	0	0	0	0
16	0	0	0	0	0	0
32	0	0	0	0	0	0
64	0,001	0,001	0,001	0,001	0,001	0,001
128	0,005	0,005	0,005	0,005	0,005	0,005
256	0,036	0,035	0,037	0,035	0,035	0,036
512	0,277	0,281	0,276	0,278	0,282	0,279
600	0,437	0,435	0,435	0,435	0,438	0,436
700	0,685	0,684	0,687	0,685	0,684	0,685
800	1,02	1,01	1,01	1,01	1,01	1,012
900	1,445	1,433	1,434	1,432	1,447	1,438
1024	2,095	2,09	2,09	2,088	2,091	2,091



Obrázek 6 Doba běhu maticového programu v modifikaci A^{16}

Z grafických i tabelárních výsledků vyplývá, že obě prostředí dokážou velmi efektivně operovat i s rozměrnými maticemi. Z grafů je dále patrno, že délka výpočtu kódu se v jednotlivých prostředích začíná lišit až při operacích s maticemi většími jak 100x100. U takto velkých matic je časový rozdíl zanedbatelný, zvětšuje se však přibližně exponenciálně se vzrůstajícími rozměry matice a to v neprospěch prostředí MathScript. V prostředí MATLAB dochází k přibližně lineárnímu nárůstu zanedbatelného sklonu oproti nárůstu v prostředí MathScript. Nicméně nutno podotknout, že tento program obsahuje pouze velmi jednoduchý výpočet. Tato teze se tedy nemusí potvrdit zcela např. u složitějších nebo rozsáhlejších programů. Za povšimnutí stojí i velmi zajímavý časový rozdíl mezi jednotlivými výpočty. Pomineme-li počáteční výpočty pro $N < 100$, které se dají jen velmi těžko porovnávat kvůli časové hodnotě výpočtu v prostředí MATLAB, stojí za povšimnutí výpočty pro hodnoty ostatní, tedy $N \geq 100$. Zajímavým faktem je, že při takto zvoleném testovacím programu došlo přibližně k desetinásobnému běhu programu v prostředí MathScript, oproti prostředí MATLAB.

Pro matice vyšších řádů 2048x2048 začínalo mít prostředí MathScript problémy s pamětí a výpočet byl odepřen. V prostředí MATLAB byl výpočet odepřen až u matic 8192x8192 a to ze stejného důvodu.

Původní zkoumané hodnoty velikosti matic byly mocniny čísla 2. Pro lepší grafickou ilustraci jsem však doplnil do tabulek i grafů hodnoty rozměrů matic N = (600, 700, 800, 900) a to z důvodu, že mezi hodnotami N = 512 a N = 1024 byla až příliš propastná mezera znesnadňující odhad grafické křivky.

3.4 SROVNÁNÍ OPERACÍ S CYKLY

3.4.1 Testovací program

V téměř každém programu se vyskytuje cyklus pro řízení běhu. Maticově orientované jazyky, jako je i jazyk MathScript, doporučuje se cyklům vyhýbat a kde je to možné používat raději čistě maticové operace. Tento požadavek však nelze vždy splnit. Z tohoto důvodu jsem pro další vzájemné srovnání volil program, jenž ve svém těle obsahuje smyčku, která je předmětem zkoumání.

Program v hlavním cyklu funguje tak, že naplní pole určité délky náhodnými hodnotami. Tato délka se s každou další iterací o 1000 prvků prodlouží. Tyto se poté sčítají do samostatné proměnné, se kterou již dále není operováno. Posledním příkazem v těle měřeného cyklu se vypočte druhá mocnina této náhodné veličiny. Kombinací příkazů v cyklu vznikly celkem tři modifikace testovacího kódu:

- modifikace 0 čistě s vygenerováním náhodné hodnoty
- modifikace 1 s vygenerováním hodnoty a kumulativním součtem těchto hodnot
- modifikace 2 s vygenerováním náhodné hodnoty, kumulativním součtem náhodně vygenerovaných hodnot a kvadrátem náhodné hodnoty, se kterým se již dále nepočítá

Hlavní cyklus je změřen časovacími funkcemi „tic-toc“ a uložen do výstupního pole, které je na konci předáno zpět uživateli. Program je uveden jako Příloha 8.

3.4.2 Výsledky

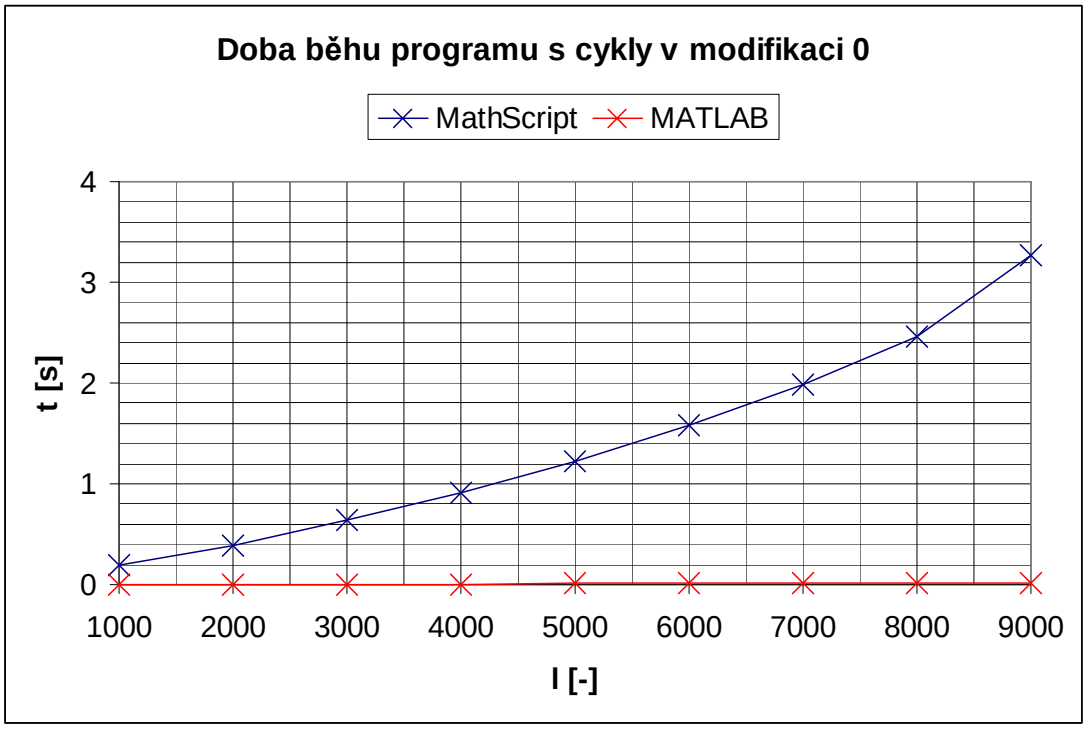
Výsledky jsou reprezentovány tabelárně i graficky. Ve výpočtech se opět operuje se střední hodnotou výběrového souboru hodnot $\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i$.

Tabulka 20 Doba výpočtu v prostředí MathScript, modifikace 0

Doba výpočtu v prostředí MathScript, modifikace 0						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1000	0,187	0,187	0,188	0,186	0,19	0,188
2000	0,388	0,395	0,395	0,393	0,392	0,393
3000	0,628	0,637	0,636	0,637	0,637	0,635
4000	0,903	0,929	0,918	0,917	0,92	0,917
5000	1,209	1,236	1,227	1,239	1,239	1,230
6000	1,562	1,587	1,588	1,583	1,607	1,585
7000	1,972	2,001	1,965	1,991	2,002	1,986
8000	2,453	2,482	2,426	2,481	2,487	2,466
9000	3,271	3,278	3,313	3,234	3,238	3,267

Tabulka 21 Doba výpočtu v prostředí MATLAB, modifikace 0

Doba výpočtu v prostředí MATLAB, modifikace 0						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1000	0,001	0,002	0,002	0,002	0,002	0,002
2000	0,003	0,003	0,003	0,003	0,003	0,003
3000	0,004	0,004	0,004	0,004	0,004	0,004
4000	0,006	0,007	0,006	0,006	0,006	0,006
5000	0,008	0,018	0,008	0,008	0,008	0,010
6000	0,01	0,01	0,01	0,01	0,001	0,008
7000	0,016	0,011	0,011	0,011	0,011	0,012
8000	0,012	0,013	0,013	0,013	0,013	0,013
9000	0,014	0,014	0,014	0,014	0,014	0,014



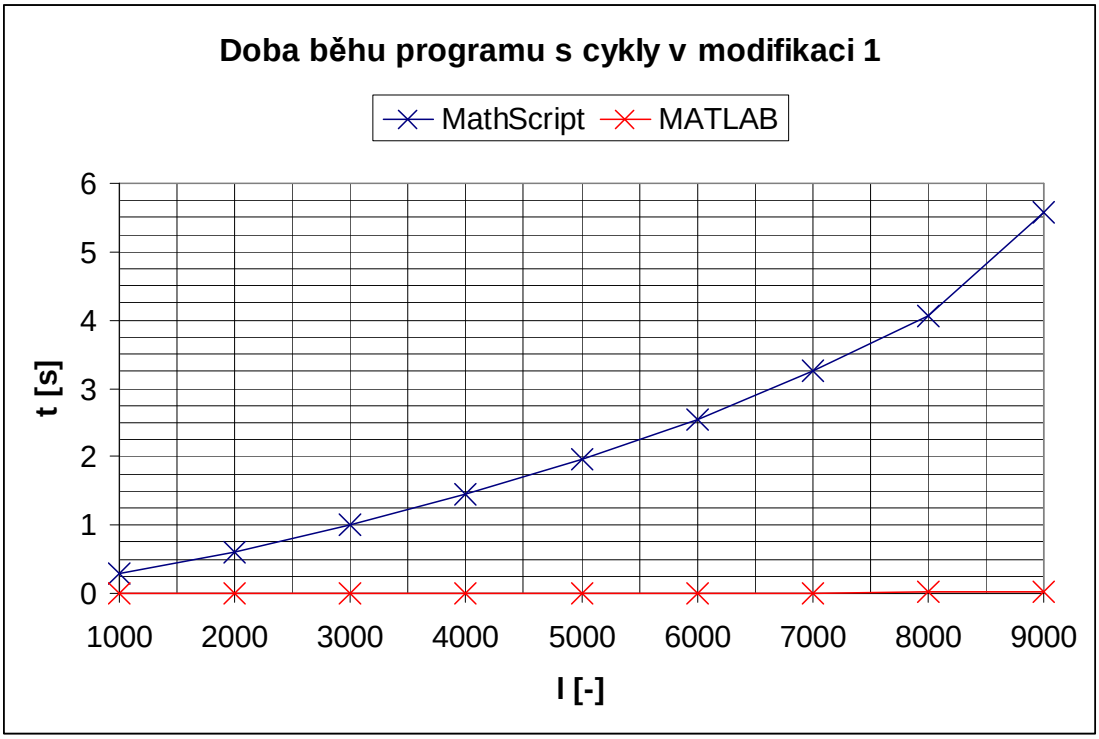
Obrázek 7 Doba běhu programu s cykly v modifikaci 0

Tabulka 22 Doba výpočtu v prostředí MathScript, modifikace 1

Doba výpočtu v prostředí MathScript, modifikace 1						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1000	0,286	0,285	0,286	0,286	0,286	0,286
2000	0,605	0,607	0,608	0,609	0,606	0,607
3000	0,994	1	0,995	0,996	0,995	0,996
4000	1,458	1,443	1,463	1,447	1,472	1,457
5000	1,939	1,941	1,953	1,951	1,986	1,954
6000	2,514	2,553	2,572	2,551	2,574	2,553
7000	3,179	3,246	3,207	3,314	3,367	3,263
8000	3,986	4,051	3,968	4,052	4,223	4,056
9000	5,662	5,727	5,521	5,624	5,386	5,584

Tabulka 23 Doba výpočtu v prostředí MATLAB, modifikace 1

Doba výpočtu v prostředí MATLAB, modifikace 1						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1000	0,002	0,002	0,002	0,002	0,002	0,002
2000	0,003	0,003	0,003	0,003	0,003	0,003
3000	0,004	0,004	0,004	0,004	0,004	0,004
4000	0,007	0,006	0,006	0,006	0,006	0,006
5000	0,008	0,008	0,008	0,008	0,008	0,008
6000	0,01	0,01	0,01	0,01	0,01	0,010
7000	0,011	0,011	0,011	0,011	0,011	0,011
8000	0,012	0,013	0,022	0,022	0,02	0,018
9000	0,014	0,022	0,014	0,014	0,014	0,016



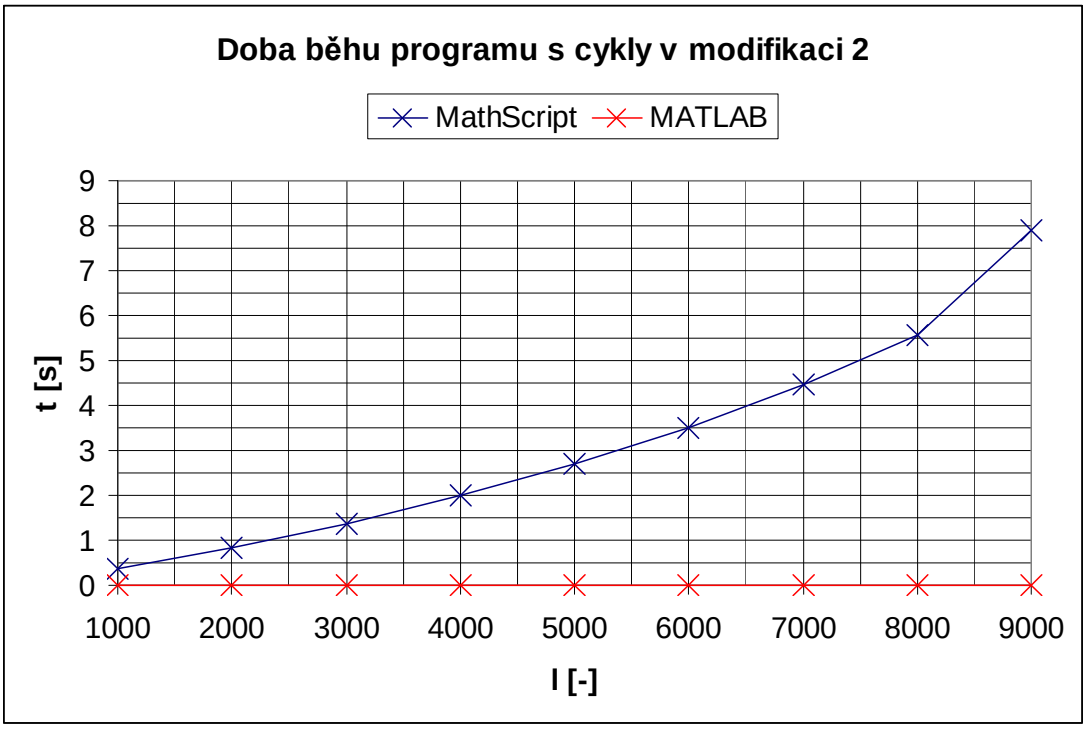
Obrázek 8 Doba běhu programu s cykly v modifikaci 1

Tabulka 24 Doba výpočtu v prostředí MathScript, modifikace 2

Doba výpočtu v prostředí MathScript, modifikace 2						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1000	0,383	0,383	0,383	0,383	0,384	0,383
2000	0,817	0,811	0,818	0,82	0,818	0,817
3000	1,361	1,369	1,35	1,342	1,362	1,357
4000	2,026	1,987	2	1,995	2,04	2,010
5000	2,703	2,672	2,678	2,691	2,693	2,687
6000	3,527	3,52	3,514	3,51	3,497	3,514
7000	4,536	4,501	4,443	4,474	4,433	4,477
8000	5,53	5,497	5,643	5,544	5,699	5,583
9000	7,769	7,83	7,952	7,893	8,033	7,895

Tabulka 25 Doba výpočtu v prostředí MATLAB, modifikace 2

Doba výpočtu v prostředí MATLAB, modifikace 2						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
1000	0,002	0,002	0,002	0,002	0,002	0,002
2000	0,003	0,003	0,003	0,003	0,003	0,003
3000	0,004	0,004	0,004	0,004	0,004	0,004
4000	0,006	0,006	0,006	0,006	0,006	0,006
5000	0,008	0,008	0,008	0,008	0,008	0,008
6000	0,01	0,01	0,01	0,01	0,01	0,010
7000	0,011	0,011	0,011	0,011	0,011	0,011
8000	0,013	0,013	0,013	0,013	0,013	0,013
9000	0,014	0,014	0,014	0,014	0,014	0,014



Obrázek 9 Doba běhu programu s cykly v modifikaci 2

Z tabulek i grafů je vidět, že testovaný cyklus v prostředí MathScript je výrazně pomalejší, nežli ten samý v prostředí MATLAB. Je logické, že čím více příkazů daná smyčka obsahuje tím více jí zabere času, než se provede. V prostředí MathScript však čas potřebný ke zpracování příkazů v cyklu byl s přibližně lineárním nárůstem v závislosti na délce zpracovávaného vektoru oproti prostředí MATLAB. V prostředí MATLAB kupodivu čas potřebný pro zpracování smyčky trval přibližně tisíciny sekundy a na testovaném programu se neprojevila závislost doby zpracování na množství příkazů ve smyčce, jako tomu bylo v prostředí MathScript. Ačkoliv je porovnávaný program identický v obou prostředích je nutné podotknout, že velký problém v tomto případě je rostoucí vektor v nejvnitřnější smyčce. Takovémuto zápisu by se měl uživatel vyvarovat a např. i prostředí MATLAB na něj upozorňuje jako na závažnou příčinu možného zpomalení programu, nicméně i přesto si s ní dokáže velmi dobře poradit.

3.5 SROVNÁNÍ VYKRESLOVACÍ FUNKCE

3.5.1 Testovací program a výsledky

Dalším srovnáním bylo vykreslení křivky pomocí funkce „plot“. Taktéž jde o vzájemné porovnání, ale i o zjištění, zdali záleží při vykreslování na vykreslované funkci nebo délce datového vektoru. Testovací programy byly v tomto případě dva. Prvním testovacím programem byla vykreslována ta samá funkce, ale pro různě dlouhé časové vektory, uvedená jako Příloha 9. V druhém testovacím programu šlo naopak o datový vektor konstantní délky, ale vykreslována byla pokaždé jiná funkce, uvedeno jako Příloha 10.

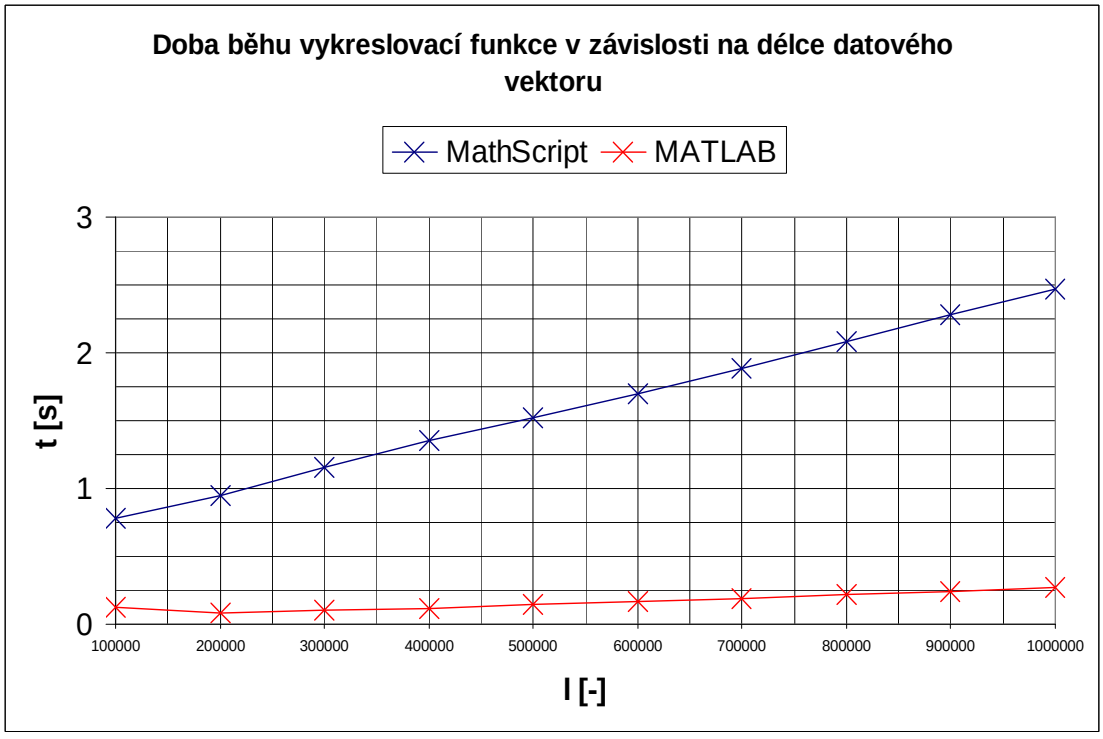
Ve výpočtech se stejně jako v předchozích případech operuje se střední hodnotou výběrového souboru hodnot $\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i$.

Tabulka 26 Doba výpočtu v prostředí MathScript, proměnná délka vektoru

Doba výpočtu v prostředí MathScript, proměnná délka vektoru						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
100000	0,75	0,791	0,791	0,789	0,789	0,782
200000	0,944	0,952	0,949	0,949	0,949	0,949
300000	1,154	1,158	1,152	1,151	1,156	1,154
400000	1,328	1,354	1,352	1,359	1,355	1,350
500000	1,538	1,521	1,521	1,507	1,512	1,520
600000	1,711	1,701	1,71	1,69	1,693	1,701
700000	1,897	1,89	1,896	1,857	1,896	1,887
800000	2,076	2,085	2,089	2,092	2,067	2,082
900000	2,278	2,298	2,29	2,262	2,291	2,284
1000000	2,462	2,453	2,481	2,462	2,463	2,464

Tabulka 27 Doba výpočtu v prostředí MATLAB, proměnná délka vektoru

Doba výpočtu v prostředí MATLAB, proměnná délka vektoru						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
100000	0,127	0,118	0,129	0,123	0,125	0,124
200000	0,078	0,08	0,084	0,077	0,077	0,079
300000	0,097	0,093	0,097	0,112	0,098	0,099
400000	0,123	0,125	0,114	0,114	0,118	0,119
500000	0,146	0,146	0,156	0,142	0,145	0,147
600000	0,17	0,169	0,164	0,169	0,162	0,167
700000	0,193	0,193	0,195	0,186	0,189	0,191
800000	0,216	0,221	0,214	0,219	0,213	0,217
900000	0,239	0,241	0,234	0,236	0,241	0,238
1000000	0,269	0,26	0,267	0,263	0,271	0,266



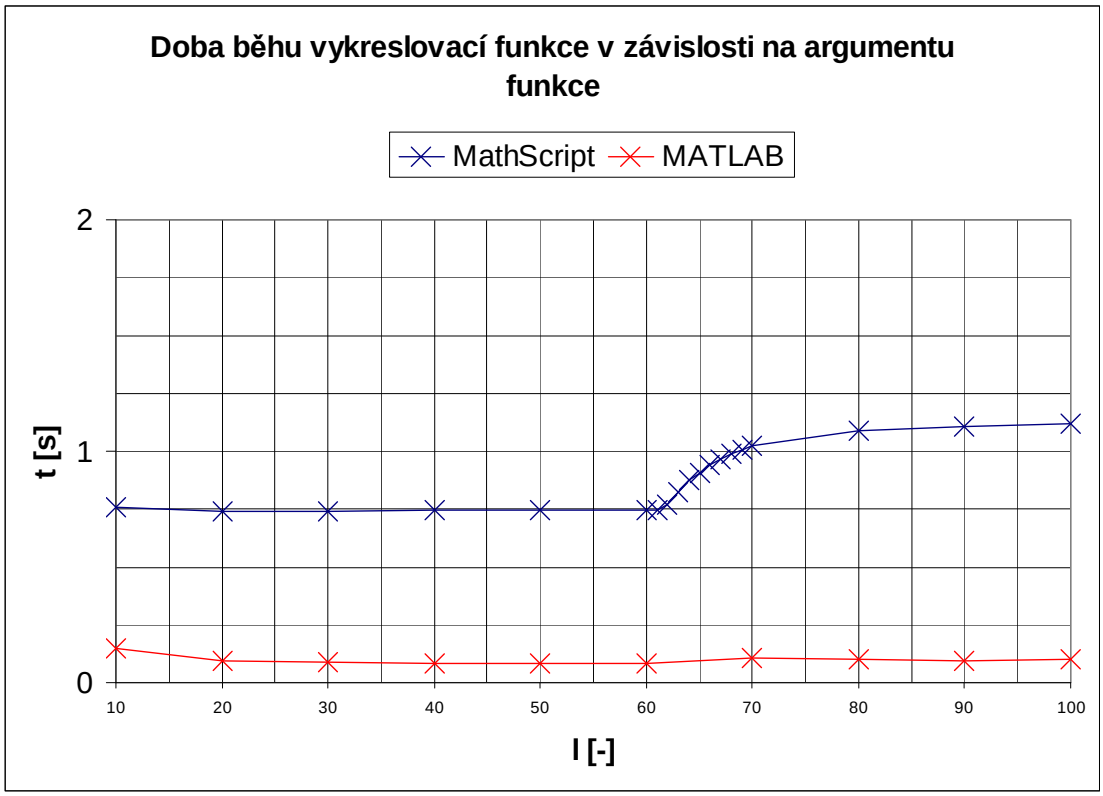
Obrázek 10 Vykreslení funkce v závislosti na délce datového vektoru

Tabulka 28 Doba výpočtu v prostředí MathScript, proměnná funkce

Doba výpočtu v prostředí MathScript, proměnná funkce						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
10	0,75	0,759	0,761	0,761	0,761	0,758
20	0,741	0,741	0,741	0,741	0,743	0,741
30	0,742	0,745	0,743	0,741	0,742	0,743
40	0,743	0,745	0,745	0,742	0,747	0,744
50	0,745	0,745	0,746	0,745	0,743	0,745
60	0,746	0,746	0,743	0,742	0,746	0,745
61	0,739	0,745	0,746	0,747	0,746	0,745
62	0,77	0,768	0,768	0,77	0,77	0,769
63	0,824	0,825	0,824	0,826	0,825	0,825
64	0,882	0,871	0,87	0,87	0,871	0,873
65	0,909	0,909	0,905	0,907	0,906	0,907
66	0,939	0,939	0,938	0,938	0,938	0,938
67	0,967	0,964	0,965	0,966	0,966	0,966
68	0,988	0,986	0,986	0,987	0,984	0,986
69	1,004	1,006	1,005	1,006	1,006	1,005
70	1,021	1,021	1,023	1,022	1,037	1,025
80	1,089	1,091	1,089	1,09	1,087	1,089
90	1,105	1,114	1,108	1,102	1,107	1,107
100	1,118	1,129	1,122	1,12	1,115	1,121

Tabulka 29 Doba výpočtu v prostředí MATLAB, proměnná funkce

Doba výpočtu v prostředí MATLAB, proměnná funkce						
l [-]	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
10	0,139	0,15	0,147	0,141	0,151	0,146
20	0,102	0,092	0,098	0,095	0,1	0,097
30	0,084	0,095	0,079	0,091	0,087	0,087
40	0,081	0,085	0,088	0,087	0,08	0,084
50	0,084	0,085	0,087	0,083	0,089	0,086
60	0,087	0,082	0,084	0,086	0,083	0,084
70	0,102	0,099	0,1	0,11	0,108	0,104
80	0,099	0,1	0,101	0,1	0,098	0,100
90	0,092	0,092	0,105	0,097	0,1	0,097
100	0,091	0,096	0,118	0,105	0,092	0,100



Obrázek 11 Doba běhu vykreslovací funkce v závislosti na argumentu funkce

Z tabelárních a grafických výsledků je opět vidět rozdílná doba zpracování příkazu mezi oběma prostředími, ačkoliv se jedná o příkazy, které v obou prostředích fungují stejným způsobem. V případě časové závislosti vykreslení požadované neměnné funkce, ale při proměnné délce datového vektoru je v obou případech patrný lineární nárůst. V případě prostředí MATLAB se jedná o nárůst s velmi zanedbatelnou strmostí, vezmeme-li v úvahu počet dat vstupního vektoru. V případě prostředí MathScript se jedná o nárůst s již o poznání větším koeficientem, avšak stále je patrný lineárně narůstající trend.

Druhý zkoumaný algoritmus, tedy s konstantní délkou datového vektoru a proměnnou vykreslovanou funkcí, má o poznání zajímavější graf. V případě prostředí MATLAB se opět jedná o velmi nízké hodnoty a celá křivka je prakticky konstantní. V případě prostředí MathScript se dá křivka popsat jako po částech konstantní. Zlom nastává v rozmezí $f = x^{60}$ až $f = x^{70}$ kde dochází k exponenciálnímu

nárůstu a ustálení na nové konstantní hodnotě. Doba zpracování algoritmu je oproti prostředí MATLAB opět vyšší.

3.6 OBECNÝ PŘÍKLAD

3.6.1 Testovací program

Jako testovací program obecného příkladu byl zvolen algoritmus Back-Propagation s pevným krokem učení neuronové sítě. Tento příklad v sobě obsahuje všechny předchozí testované atributy. Je uveden jako Příloha 11.

3.6.2 Výsledky

Výsledky jsou zpracovány tabelárně. Ve výpočtu se operuje se střední hodnotou výběrového souboru hodnot $\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i$.

Tabulka 30 Obecný program Back-Propagation s pevným krokem učení

Obecný program Back-Propagation s pevným krokem učení						
prostředí	t_1[s]	t_2[s]	t_3[s]	t_4[s]	t_5[s]	$\bar{t}$ [s]
MathScript	1,569	1,478	1,498	1,496	1,495	1,507
MATLAB	0,142	0,139	0,141	0,138	0,138	0,140

V tomto příkladu jsou měřeny kombinace maticových operací, cyklů i vykreslování. Z výsledků je patrný velký rozdíl doby výpočtu algoritmu v obou prostředích. V prostředí MathScript dosahuje až desetinásobku času prostředí MATLAB.

3.7 PRŮBĚHY TESTŮ

Pro všechny uvedené výpočty bylo použito PC následující konfigurace.

Procesor: Intel Core2Duo E4500 (2,20GHz),
paměť: 2x 1GB 667MHz DDR2 RAM,
operační systém: Windows XP Professional x86 platforma.

Vzhledem k dvou jádrovému procesoru byly veškeré procesy ve správci úloh přesunuty ke zpracovávání jednomu procesoru a výpočty ať v prostředí MathScript nebo v prostředí MATLAB probíhaly vždy střídavě procesorem druhým. Tímto se dosáhlo eliminace jevů (v rámci možností operátora) ovlivňujících dobu výpočtu.

Hlavní součástí testů bylo vývojové prostředí LabVIEW Professional ve verzi 8.5. Rozšiřující Toolkity, se kterými jsem pracoval pouze na internetu, byly ve verzích Control Design and Simulation Module 8.5 a Digital Filter Design Toolkit 8.2.

Vývojové prostředí MATLAB bylo ve verzi 7.4 s rozšiřujícím Filter Design Toolbox verze 4.1.

4. LABVIEW

4.1 HISTORIE NI A HISTORIE LABVIEW

Americkou společnost National Instruments(NI) založili roku 1976 společníci Dr. James Truchard, Jeff Kodosky a Bill Nowlin. Cílem bylo uplatnění počítače ve vývoji, simulaci a měření průmyslových zadání. Díky orientaci na spokojenost koncového zákazníka, investicím do moderních technologií a spoustě dalších inovací se společnost vyvinula od garážové záležitosti až po mezinárodní gigant, jakým je dnes.

Roku 1986 společnost představila revoluční vědecký nástroj. Grafické vývojové prostředí LabVIEW, jež za dobu celých dvaceti let dosáhlo obrovských změn, až do podoby s jakou je možnost pracovat v současné verzi 8.5. Tímto společnost dala vzniknout fenoménu tzv. „virtuální instrumentace“, jež umožňuje nezbytné simulace průmyslových procesů před jejich uvedením v provoz.

4.2 VÝVOJOVÉ PROSTŘEDÍ LABVIEW

LabVIEW je, jak již bylo řečeno výše, grafické vývojové prostředí, které zahrnuje flexibilitu programovacího jazyka, stejně tak jako vysokou úroveň funkčního použití především v automatizaci a měření. Vychází ze základní myšlenky, že ten, kdo ví co a jak měřit a jak to prezentovat je technik a ten a priori nemusí vědět jak danou aplikaci nebo proces, naprogramovat. Tyto data musí předat programátorovi, který se o to postará. Tento fakt má ovšem velmi nepříznivé následky na tzv. „informační šum“. Ujali se tedy myšlenky o vývoji programovacího jazyka, který se v rukou technika promění ve všehoschopnou zbraň.

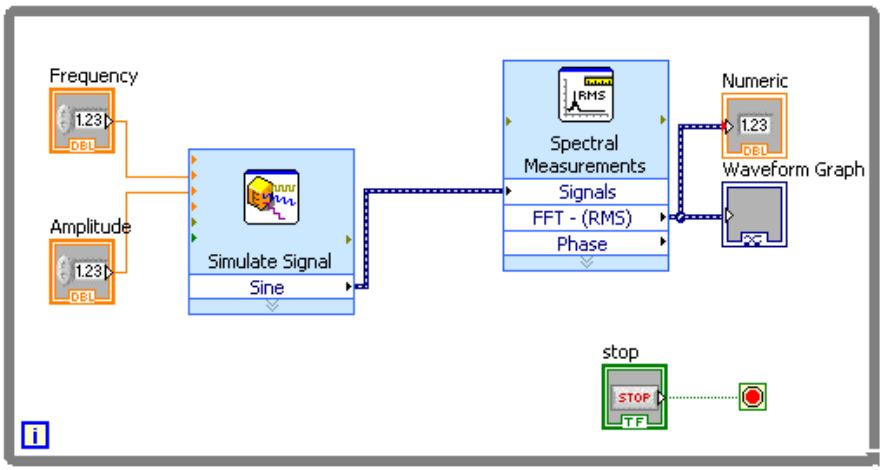
Laboratory Virtual Instruments Engineering Workbench neboli LabVIEW, se stal grafickým vývojovým prostředím, jehož hlavním atributem není textové programování, ale programování pomocí jednoduchých bloků i složitých komplexních soustav těchto bloků. Obsahuje bohaté knihovny pro vytváření různorodých aplikací zaměřených především do oblasti měření a automatizace, tedy sběr, analýza a prezentace dat. Jedná se o plnohodnotný programovací jazyk se všemi odpovídajícími datovými i programovými strukturami, ovšem v grafické podobě. Často bývá označován také jako jazyk G (grafický).

Výsledkem vývojového prostředí je tzv. virtuální přístroj (VI – Virtual Instrument), jež si v ničem nezadá s přístrojem skutečným. Každý VI se skládá ze dvou základních částí. Čelního panelu (Front Panel), jež určuje, jak bude přístroj vypadat pro koncového uživatele, umístění indikačních i ovládacích prvků, velikosti a podobně. Dále pak z blokového diagramu (Block Diagram), ve kterém je jednoznačně určena jeho základní funkce. Virtuální přístroj se sám o sobě může skládat z dalších mnohem jednodušších virtuálních „pod-přístrojů“. Tyto vykonávají jednodušší základní funkce přístroje, jako je např. získávání dat, přepočet, analýza, atd. Takovýmto VI říkáme podřízené virtuální přístroje (Substitute Virtual Instruments), neboli zkráceně Sub-VI's. Aby se námi vytvořený virtuální přístroj mohl stát „pod-přístrojem“ (Sub-VI), musíme mu říci, co budou jeho vstupy/výstupy z/do nadřízené aplikace. K tomu slouží tzv. konektor (Connector), jež definuje počet vstupů, výstupů a jejich organizaci do každého jednoho Sub-VI.

Celý takto modulárně poskládaný VI se dá díky bohaté škále ladících prostředků odzkoušet a vyladit k plné spokojenosti koncového uživatele a nakonec i přeložit do spustitelného tvaru. Tedy samostatné aplikace nezávislé na vývojovém prostředí LabVIEW.

4.3 POJEM VIRTUÁLNÍ INSTRUMENTACE

Virtuální instrumentace kombinuje výhody lehce dostupných zařízení, jako je PC s flexibilním softwarem spolu s velkým množstvím měřicího a řídicího hardwaru. Takto může koncový uživatel lehce vytvořit požadované aplikace nebo je v průběhu času měnit, dle požadavků. Díky virtuální instrumentaci dochází ke snížení času potřebného k vývoji, ke zvýšení kvality vyvíjených aplikací a současnému snížení jejich nákladů. Bez virtuální instrumentace se v současné době neobejde žádná aplikace počínaje spotřební elektronikou a rafinérskými společnostmi konče. Její rozmach zapříčinil především obrovský technický vývoj a rovněž tak příznivý cenový trend na poli osobních počítačů.



Obrázek 12 Blokový diagram v grafickém jazyku G prostředí LabVIEW

4.4 NATIVNÍ PROSTŘEDÍ LABVIEW

Jak již bylo výše řečeno, prostředí MathScript v prostředí LabVIEW slouží ke zvýšení komfortu uživatele a samotných aplikací. Rozhodně nemá sloužit k nahrazení grafického jazyka jiným jazykem.

Často proto nalezneme různá poučení a doporučení na stránkách výrobce, že nejlepší volbou pro komplikované a rozsáhlé aplikace je čisté nativní prostředí LabVIEW. Je to způsobeno tím, že prostředí MathScript je implementováno „nad LabVIEW“ a používá tzv. „volný jazyk“ (není třeba definovat datový typ proměnné, prostředí jej určí samo). Kombinace těchto faktorů způsobuje, že jsou algoritmy vytvořené v prostředí MathScript pomalejší, než ekvivalentní algoritmy v prostředí LabVIEW. Tuto informaci je možné nalézt v [8] např. zde: < http://forums.ni.com/ni/board/message?board.id=MathScript&view=by_date_ascending&message.id=105#M105>.

Z tohoto důvodu jsem se po předchozích úvahách rozhodl testovací programy, uvedené v přílohách, napsat i v čistém nativním prostředí LabVIEW a podrobit je stejnému testu rychlosti čistě pro utvoření představy. Rozsáhlost výsledků však neumožňuje všechny umístit do této práce, proto jsou jako přílohy na doprovodném CD. Měření a grafy jsou uloženy v pracovním souboru vytvořeném v tabulkovém procesoru. Blokové diagramy z grafického prostředí LabVIEW jsou umístěny jako Příloha 12 až Příloha 16.

Dle očekávání byly ekvivalentní programy napsané v nativním prostředí LabVIEW výrazně rychlejší, než programy v prostředí MathScript. Toto potvrzuje tezi od vývojářů uváděnou na fórech výrobce. V prvních dvou (maticové operace, operace s cykly) testovacích programech byly grafické programy jen mírně pomalejší oproti jejich ekvivalentům z prostředí MATLAB. V dalších třech testovacích případech (proměnná délka vykreslovaného vektoru, proměnná funkce a obecný program) byly programy v grafickém jazyce dokonce mírně rychlejší oproti prostředí MATLAB.

5. ZÁVĚR

V této práci jsou popsány základy vývojového prostředí LabVIEW MathScript společnosti National Instruments. Od samotného grafického rozložení, přes práci ve vývojovém prostředí a základy programovacího jazyka spolu s příklady jeho příkazů. Každý takový blok je na konci posuzován a porovnáván s vývojovým prostředím MATLAB společnosti MathWorks, které jsou si navzájem velmi podobné. Jsou porovnány příkazy obsažené v prostředí MathScript s příkazy obsaženými v prostředí MATLAB z hlediska jejich funkce a odlišností. V neposlední řadě je porovnáván i čas potřebný na vykonání experimentálně napsaných zdrojových kódů v obou prostředích.

Dle jednotlivých shrnutí lze usoudit, že prostředí LabVIEW MathScript nemá všechny možnosti prostředí MATLAB. Jde především o možnosti volby jednotlivých příkazů, jejich použití a možnosti nastavení, které je u velkého množství příkazů v prostředí MATLAB bohatší. Přesto je však do jisté míry zajištěna přenositelnost zdrojových programů mezi jednotlivými prostředími. Drtivá většina základních příkazů včetně použití zůstává identické.

Pokud jde o časové zpracování experimentálních programů lze z výše uvedeného vyvodit následující závěr. Při operacích s maticemi rozměrů větších než 100x100 se projevuje exponenciální závislost doby zpracování programu v neprospěch prostředí MathScript. Použití cyklů v m-skriptech určených pro prostředí MathScript se nedoporučuje z důvodu dalšího výrazného zpomalení běhu programu a celkově je lepší používat maticové operace. Větší počet příkazů uvedených v těle cyklu dalším způsobem zpomaluje čas vykonání programu. V prostředí MATLAB se v testovaném rozsahu nic takového neprojevilo. Při vykreslování funkcí se v prostředí MathScript projevuje výrazná lineární rostoucí závislost na objemu vykreslovaných dat. Tato závislost se v testovaném rozsahu v prostředí MATLAB projevila jen minimálně, avšak i zde byla lineárně rostoucí. Při konstantní délce vykreslovaného vektoru se v testovaném programu neprojevila závislost prostředí MATLAB na době vykreslování funkce a byla přibližně

konstantní. V prostředí MathScript se projevila poměrně zvláštní závislost. Zprvu konstantní, avšak v určitém bodě došlo k exponenciálnímu nárůstu a dále se závislost ustálila opět na konstantní, avšak odlišné hodnotě. Tento jev mohl být důsledek práce prostředí s pamětí. U obecného testovacího programu, kde byl předmětem testování skutečně používaný program se vyskytovaly všechny předchozí testované atributy, jako jsou operace s maticemi, cykly a vykreslování. I zde doba vykonání programu v prostředí MathScript trvala déle než v prostředí MATLAB a to přibližně desetkrát déle.

Z výsledků je patrný rozdíl doby vykonání programů v jednotlivých prostředích, ačkoliv podmínky využití a zachování výpočetního prostředí byly identické v rámci možností. Rozdíl je pravděpodobně způsoben tím, že prostředí MathScript je implementováno „nad LabVIEW“, jak bylo psáno a jak na něj poukazují aktivní členové MathScript fóra, zaměstnanci National Instruments. Obecně proto vývojáři doporučují používat čisté nativní prostředí LabVIEW (tedy grafický jazyk G), protože je zdaleka nejrychlejší. Čistě textové nebo kombinace textového a grafického programování však není zavrženo a jak je řečeno i v této práci, zvyšuje pohodlí uživatele spolu s čitelností a přehledností kódu, především pro jednoduché příkazy.

Jedním z mnoha dalších omezení je, že prostředí MathScript je použitelné pouze na systémech PC, nikoliv v systémech reálného času. I na PC však oplývá mnoha dalšími omezeními. Jedná se o omezení použití v prvku MathScript Node a omezení použití v samostatně spustitelných aplikacích. Výčet funkcí takto omezených je uveden v této práci. Konkrétní omezení každého příkazu však nemohla být z důvodů rozsáhlosti uvedena. Prvek MathScript Node je součástí prostředí LabVIEW, v prostředí MathScript neexistuje a proto zde nejsou funkce tímto způsobem dále omezeny.

Vývojáři National Instruments však pokračují ve vývoji a zdokonalování tohoto jazyka a dokazuje to i skok mezi verzemi 8.0, 8.2 a 8.5. Je možné tedy říci, že tento jazyk je zaveden do prostředí LabVIEW poměrně nedávno a vývojáři tak pomalu dostávají zpětnou vazbu na požadavky od uživatelů.

V současné době se objevují na webových stránkách společnosti National Instruments výzvy všem, kteří se chtějí podílet na testování nové beta verze prostředí LabVIEW 8.6. Toto prostředí zahrnuje další změny, opravy a doplnění. Už nyní odpovídají vývojáři na fórech výrobce o desítkách a stovkách opravených algoritmů a vylepšení nové verze jazyka MathScript, ačkoliv nebyl vydán žádný oficiální seznam těchto vylepšení.

Na závěr snad jen připomenutí, že účelem prováděných testů bylo zobrazení rozdílu. V drtivé většině případů může uživateli prostředí MathScript postačit, jak z hlediska přenositelnosti kódu, tak z hlediska doby vykonání programu. Celkově lze označit prostředí LabVIEW MathScript za funkční, nicméně v porovnání s prostředím MATLAB ne až tak vyspělé. Situace se ovšem mění k lepšímu s každou novou verzí prostředí LabVIEW.

6. LITERATURA

[1] ČEJKA, M.: Teorie měření. Přednáška VUT Brno 2006.

[2] JKI SOFTWARE: VI Package Manager. Počítačový program VIPM. Dostupný z: <http://jkisoft.com/vipm>.

[3] MATHWORKS: Internetové stránky společnosti. Dostupné z: <http://www.mathworks.com>.

[4] MATHWORKS: Nápověda MATLAB verze 7.4.

[5] NATIONAL INSTRUMENTS: Internetové stránky společnosti. Dostupné z: <http://www.ni.com>.

[6] NATIONAL INSTRUMENTS: LabVIEW Online Evaluation. Dostupné z: <http://www.ni.com/trylabview>.

[7] NATIONAL INSTRUMENTS: Nápověda LabVIEW MathScript verze 8.5.

[8] NATIONAL INSTRUMENTS: NI Discussion Forums. Diskusní fóra. Dostupná z: <http://forums.ni.com/ni>.

[9] NATIONAL INSTRUMENTS: NI LabVIEW MathScript Online Evaluation Beta. Dostupné z: <http://netdemo.ni.com/mathscript/Default.aspx>.

[10] NATIONAL INSTRUMENTS: Začínáme s LabVIEW. Příručka. Leden 2004. 64 stran.

[11] OPENG.ORG: Home of the LabVIEW open source community. Dostupné z: <http://wiki.openg.org>.

[12] SELINGER, M.: MathScript v LabVIEW. Semestrální projekt 2 VUT Brno, leden 2008.

[13] SELINGER, M.: Propojení měřicího systému PULSE a vývojového prostředí LabVIEW. Semestrální projekt 1 VUT Brno, květen 2007.

[14] WIKIPEDIA: IEEE 754-1985 floating-point standard. Dostupné z: <http://en.wikipedia.org/wiki/IEEE_floating-point_standard>.

[15] ŽÍDEK, J.: Grafické programování ve vývojovém prostředí LabVIEW. Skripta VŠB-TU FEI Ostrava. Říjen 2002. 215 stran.

SEZNAM PŘÍLOH

Příloha 1

Totožné funkce v prostředí LabVIEW MathScript
abs, ac2poly, ac2rc, acos, acosh, acot, acoth, acsc, acsch, airy, and, angle, arburg, arcov, armcov, aryule, asec, asech, asin, asinh, atan, atan2, atanh, barthannwin, bartlett, base2dec, bessel, besselap, besselh, besseli, besselj, besselk, bessely, besschk, beta, betainc, betaln, bilinear, bin2dec, bitand, bitcmp, bitget, bitmax, bitor, bitrevorder, bitset, bitshift, bitxor, blackman, blackmanharris, blanks, blkdiag, bohmanwin, break, buttap, buttord, calendar, calllib, cart2pol, cart2sph, case, cat, cd, ceil, circshift, clc, clock, close, colormap, compan, complex, cond, condeig, conj, continue, contourc, conv, conv2, convmtx, corrcoef, corrmtx, cos, cosh, cot, coth, cov, cputime, csc, csch, csvread, csvwrite, ctanspose, cumprod, cumsum, cylinder, czt, date, dct, deal, deblank, decimate, deconv, del2, delete, det, detrend, dftmtx, diag, diary, digitrevorder, diric, disp, display, divergence, dlmread, dos, dot, double, downsample, dst, edit, ellipap, ellipj, ellipke, ellipord, ellipsoid, else, elseif, end, eomday, eq, eqtflength, erf, erfc, erfcinv, erfcx, erfinv, errorbar, etime, eval, evalc, exit, exp, expint, expm, expm1, ezcontour, ezcontourf, ezmesh, ezmeshc, ezplot, ezplot3, ezpolar, ezsurf, ezsurfc, factor, factorial, false, fclose, feof, fft, fft2, fftfilt, fftshift, fgetl, fgets, figure, fileparts, filter, filter2, filternorm, filtfilt, filtic, findstr, fir1, fir2, firls, fix, flattopwin, flipdim, fliplr, flipud, floor, for, fprintf, freqs, freqspace, freqz, frewind, fscanf, fseek, fullfile, gamma, gammaln, gausswin, gca, gcd, gcf, ge, genpath, gensig, ginput, global, gmonopuls, goertzel, gplot, gradient, griddata, grpdelay, gsvd, gt, hamming, hankel, hann, help, hex2dec, hex2num, hilbert, hist, histc, hold, home, horzcat, humps, char, cheb1ap, cheb1ord, cheb2ap, cheb2ord, chebwin, chirp, cholupdate, icceps, idct, idst, if, ifftshift, imag, impinvar, impz, ind2sub, inpolygon, input, int16, int2str, int32, int64, int8, interp, interpft, intersect, inv, invfreqs, invfreqz, ipermute, is2rc, isa, isdir, isempty, isequal, equalwithequalnans, isfinite, isglobal, ischar, isinf, iskeyword, isletter, islogical, ismember, isnan, isnumeric, isprime, isreal, isscalar, issorted, isspace, isstr, isvarname, kaiser, kron, lar2rc, latc2tf, latcfilt, lcm, ldivide, le, legendre, length, levinson, libfunctionsview, libisloaded, line, linspace, log, log10, logical, logspace, lower, lp2bp, lp2bs, lp2hp, lp2lp, lpc, ls, lsf2poly, lt, magic, mean, median, minus, mldivide, mod, mpower, mrdivide, mtimes, namelengthmax, ndims, ne, nextpow2, nchoosek, nnz, nonzeros, norm, normest, not, now, null, nuttallwin, or, orth, parzenwin, path, path2rc, pathdef, pathsep, peaks, perms, permute, persistent, phasedelay, phasez, pchip, pi, pinv, planerot, plus, poaly2lsf, pol2cart, poly, poly2ac, poly2rc, polyder, polyeig, polyfit, polyint, polyscale, polystab, power, poyvalm, primes, prod, prony, psi, pwd, qr, qz, randperm, rank, rat, rc2ac, rc2is, rc2lar, rc2poly, rceps, rcond, rdivide, real, reallog, realpow, realsqrt, rectint, rectpuls, rectwin, regexptranslate, rem, remez, remezord, repmat, resample, residue, residuez, return, rgbplot, rlevinson, rmpath, roots, rot90, round, rref, run, sawtooth, sec, sech, seqperiod, setdiff, setstr, setxor, sgolay, shg, schur, schurrc, sign, sin, sinc, single, sinh, sortrows, sos2ss, sos2tf, sos2zp, sosfilt, sound, sph2cart, sphere, spline, sqrt, square, squeeze, sscanf, stepz, stmcb, str2double, str2mat, str2num, str2rng, strcat, strcmp, strcmpi, strfind, strips, strjust, strmatch, strncmp, strncmpi, strread, strrep, strtok, strtrim, strvcat, sub2ind, subspace, switch, system, tan, tanh, tempdir, tempname, tf2latc, tf2sos, tf2ss, tf2zp, tf2zpk, tic, times, toc, toeplitz, trace, transpose, triang, tril, tripuls, triu, true, tsearch, tukeywin, uencode, uint16, uint32, uint64, uint8, uminus, union, unique, unloadlibrary, unwrap, upfirdn, uplus, upper, upsample, userpath, vander, vco, vectorize, ver, vertcat, waitforbuttonpress, wavwrite, while, xcorr, xcorr2, xcov, xor, zerophase, zp2ss, zp2tf

Příloha 2

Odlišné funkce v prostředí LabVIEW MathScript
addpath, all, any, area, axes, axis, balance, bar, bar3, bar3h, barh, beep, besself, butter, cceps, class, clear, clf, clg, cohere, condest, contour, contour3, contourf, convhull, cplxpair, cross, csd, cumtrapz, curl, datenum, datestr, datetick, datevec, dblquad, dec2base, dec2bin, dec2hex, delaunay, diff, dir, dlmwrite, eig, eigs, ellip, eps, error, exist, eye, feather, feval, fill, find, firgauss, firpm, firrcos, fit, fminbnd, fmincon, fminsearch, fminunc, fopen, format, fplot, fsolve, funm, fwrite, fzero, gammainc, gauspuls, get, grid, gtext, hadamard, hess, hilb, cheby1, cheby2, chol, ifft, ifft2, image, imread, imwrite, inf, interp1, interp2, intfilt, invhilb, kaiserord, lasterror, legend, linprog, linsolve, load, loadlibrary, log2, loglog, logm, lookfor, lu, mat2str, max, maxflat, medfilt1, mesh, meshc, meshgrid, mfilename, min, mkpp, nan, nargchk, nargin, nargout, nargoutchk, num2str, numel, ode113, ode15s, ode23, ode23s, ode23tb, ode45, ones, pareto, pascal, pause, pie, plot, plot3, plotmatrix, plotyy, polar, polyarea, polyval, pow2, pulstran, quadl, quadprog, quit, quiver, rand, randn, random, realmax, realmin, regexp, regexpi, regexprep, reshape, rosser, save, scatter, scatter3, semilogx, semilogy, set, sgolayfilt, size, sort, soundsc, sprintf, sqrtm, ss2sos, ss2tf, ss2zp, stairs, std, stem, stem3, subplot, sum, surf, surfc, surfnorm, svd, text, textread, tfe, title, trapz, treelayout, treeplot, triplequad, type, udecode, uiload, var, view, voronoi, waterfall, wavplay, wavread, wavrecord, weekday, what, which, who, whos, wilkinson, xlabel, ylabel, zeros, zlabel, zoom, zp2sos, zplane

Příloha 3

Chybějící funkce v prostředí MATLAB
aich, aiwf, alltitle, allxlabel, allylabel, aoch, aowf, deferdraw, dioread, diowrite, expm2, expm3, getfileproperty, halton, iczt, isieee, labviewroot, quad8, readbmp, richtmeyer, setfileproperty

Příloha 4

Funkce, které nejsou podporovány v LabVIEW Run-Time Engine
Totožné funkce
calendar, cd, clc, close, colormap, cylinder, diary, edit, ellipsoid, errorbar, eval, evalc, ezcontour, ezcontourf, ezmesh, ezmeshc, ezplot, ezplot3, ezpolar, ezsurf, ezsurfc ,figure, fprintf, frewind, fscanf, fseek, gca, gcf, ginput, global, gplot, grpdelay, help, hist, hold, home, line, ls, path, path2rc, pathdef, pathsep, peaks, phasedelay, phasez, rgbplot, rmpath, shg, sphere, strips, tempdir, tempname, waitforbuttonpress, wavwrite, zerophase
Odlišné funkce
addpath, area, axes, axis, bar, bar3, bar3h, barh, clear, clf, clg, cohere, contour, contour3, contourf, csd, datetick, dblquad, dir, exist, feather, feval, fill, fminbnd, fmincon, fminsearch, fminunc, format, fplot, fsolve, funm, fzero, get, grid, gtext, image, imread, imwrite, legend, load, loglog, lookfor, mesh, meshc, ode113, ode15s, ode23, ode23s, ode23tb, ode45, pareto, pause, pie, plot, plot3, plotmatrix, plotyy, polar, psd, quadl, quit, quiver, save, scatter, scatter3, semilogx, semilogy, set, stairs, stem, stem3, subplot, surf, surfc, surfnorm, text, textread, tfe, title, treeplot, triplequad, type, uiload, view, waterfall, wavread, what, which, who, whos, xlabel, ylabel, zlabel, zoom, zplane
Chybějící funkce
alltitle, allxlabel, allylabel, deferdraw, getfileproperty, labviewroot, quad8, setfileproperty

Příloha 5

Funkce prostředí MathScript v rozšiřujícím Control Design and Simulation Module
Totožné funkce
append, augstate, bandwidth, bode, bodemag, covar, ctrb, ctrbf, damp, dcgain, dlqr, dlyap, dsort, esort, estim, evalfr, filt, get, gram, hasdelay, impulse, initial, iopzmap, isct, isdt, isempty, issiso, isstable, lqr, lsim, margin, minreal, ndims, nichols, nyquist, obsv, obsvf, ord2, pole, pzmap, rlocfind, set, sigma, size, sminreal, ss2ss, ssbal, ssdata, step, totaldelay, transpose
Odlišné funkce
acker, allmargin, balreal, c2d, canon, care, d2c, d2d, dare, delay2z, diag, dlqry, drss, feedback, isproper, kalman, lqrd, lqry, lyap, modred, norm, pade, parallel, place, reg, remove, rlocus, rss, select, series, ss, tf, tfdata, zero, zpk, zpkdata
Chybějící funkce
areequal, aresimilar, distdelay, drtf, drzpk, hconcat, isctrb, isobsv, lqrdy, ord1, pid, randvec, rtf, rzpk, vconcat

Příloha 6

Funkce prostředí MathScript v rozšiřujícím Digital Filter Design Toolkit
Totožné funkce
firhalfband, firminphase, iircomb, iirnotch, iirpeak
Odlišné funkce
fircband, firgr, firlpnorm, firnyquist, iirgrpdelay, iirlpnorm, iirlpnormc

Příloha 7

Testovací program pro operaci s maticemi – „Prog_DP_1.m“

```
function Prog_DP_1()
clear all;
close all;
clc;
a = [1 2 4 8 16 32 64 128 256 512 600 700 800 900 1024];

for i = 1:15

    tic
    A = rand(a(i));
    C = A^2; % Pozdeji postupne C = A^4; a C = A^16;
    cas = double(toc);

    disp(['Pro rad matice ' num2str(a(i)) ' x ' num2str(a(i))]);
    disp(['ukon trva ' num2str(cas) ' [s].']);

    V(i) = cas;

end
```

Příloha 8

Testovací program pro operaci s cykly – „Prog_DP_2.m“

```
function Prog_DP_2()
soucet = 0;

for m = 1:9
    delka = m * 1000;

    disp(['Zacinam cyklus pro ' num2str(delka) ' prvku.']);

    tic;

    for n = 1:delka

        V(n) = rand;                % pro vsechny modifikace
        soucet = soucet + V(n);     % pro modifikaci 1 a 2
        kvadrat = V(n)^2;           % pro modifikaci 2

    end

    cas = toc;
    disp(['Cyklus trval ' num2str(cas) ' s.']);

end
```

Příloha 9

Testovací program závislosti doby vykreslení na délce vykreslovaného vektoru – „Prog_DP_3a.m“

```
function Prog_DP_3a()
clear all;
close all;
clc;

for k = 1:10

    t = 1:k*100000;
    tic;
    figure;
    plot(t, t.^2, 'r--', 'LineWidth', 2);
    cas = toc;

    disp(['Pro vektor t=' num2str(length(t)) ' prvku']);
    disp(['ukon trval ' num2str(cas) ' s.']);

end
```

Příloha 10

Testovací program závislosti doby vykreslení na vykreslované funkci – „Prog_DP_3b.m“

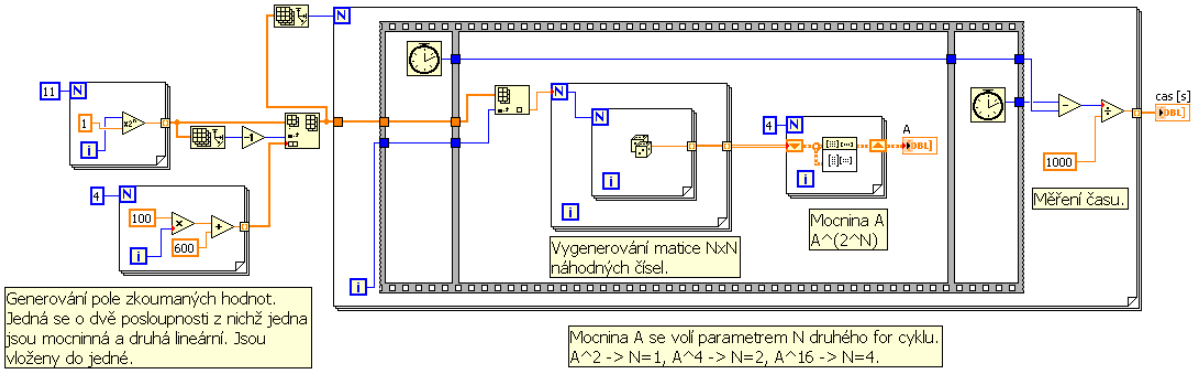
```
function Prog_DP_3b()
clear all;
close all;
clc;

vzorky = [10 20 30 40 50 60 61 62 63 64 65 66 67 68 69 70 80 90 100];
for k = 1:length(vzorky)

    t = 1:100000;
    tic;
    figure;
    plot(t, t.^(vzorky(k)), 'r--', 'LineWidth', 0.5);
    cas = toc;

    disp(['Pro funkci y=t.^' num2str(vzorky(k))]);
    disp(['ukon trval ' num2str(cas) ' s.']);

end
```

Příloha 11

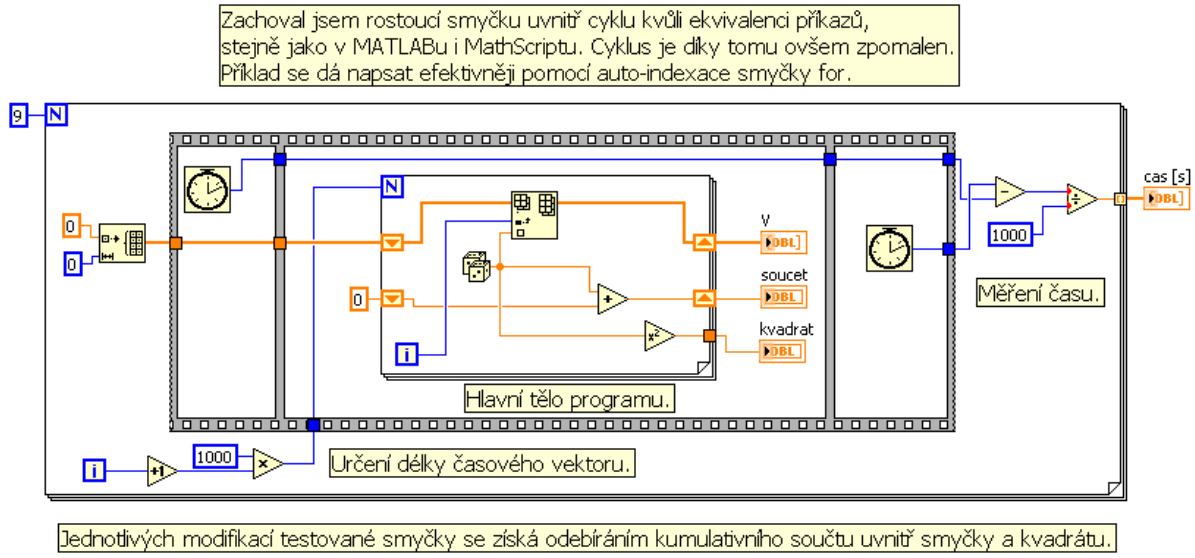
Testovací program Back-Propagation s pevným krokem učení – „Prog_DP_4.m“

```
function Prog_DP_4()
clear all;
close all;
clc;

tic

alfa = 0.015;           % Obvykle kolem jedne tisiciny
beta = 0.9;             % Obvykle 0.9
pocet_vah = 8;
vstup(1:pocet_vah) = 0;
data_in = sin(0:0.1*pi:10*pi);
vahy1(1:pocet_vah) = rand;
vahy2 = vstup;
vahy3 = vstup;

for i = 1:length(data_in)

    vystup = vstup * vahy1';
    chyba = -data_in(i) + vystup;
    data_out(i) = vystup;
    data_vahy(:, i) = vahy1';
    data_chyba(i) = chyba;

    vahy1 = vahy2 - alfa * chyba * vstup + beta * (vahy2 - vahy3);

    vahy3 = vahy2;
    vahy2 = vahy1;
    vstup(2:length(vstup)) = vstup(1:length(vstup)-1);
    vstup(1) = data_in(i);

end

figure;
subplot(3, 1, 1);
plot(data_in);
hold on;
plot(data_out, 'r');

subplot(3, 1, 2);
plot(data_chyba, 'r');

subplot(3, 1, 3);
plot(data_vahy');

toc
```

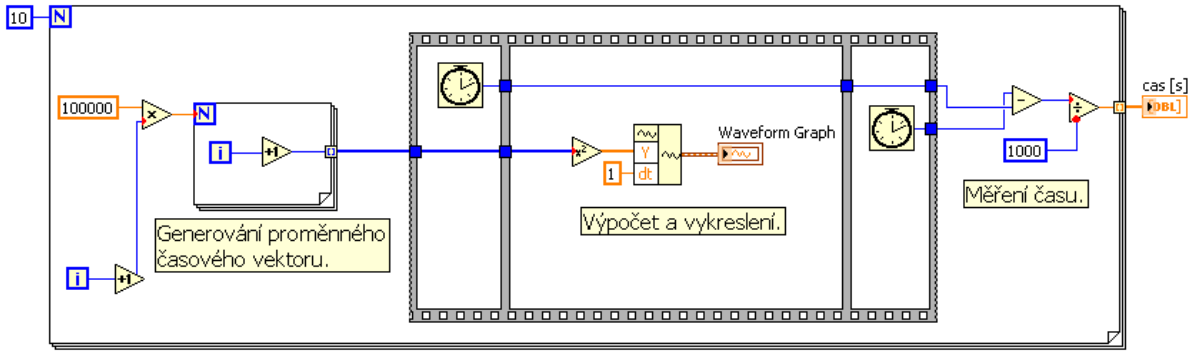
Příloha 12



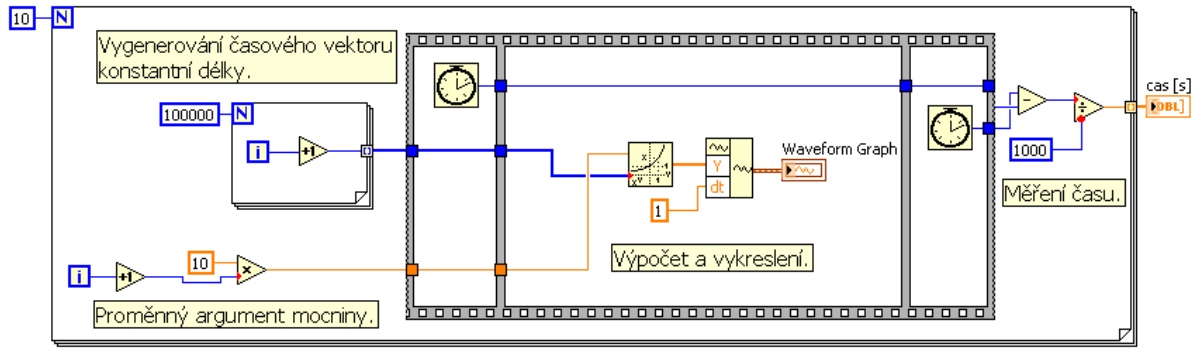
Příloha 13



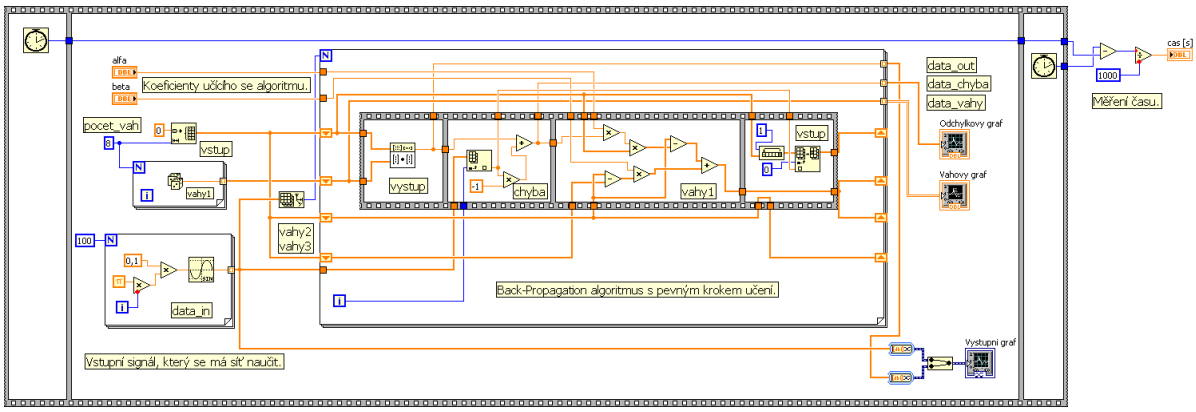
Příloha 14



Příloha 15



Příloha 16



Příloha 17

Obsah přiloženého CD-ROM		
cesta	soubor	popis
\PROGRAMY\LabVIEW	Prog_DP_1_LV.vi	Operace s maticemi v prostředí LabVIEW
	Prog_DP_2_LV.vi	Operace s cykly v prostředí LabVIEW
	Prog_DP_3a_LV.vi	Vykreslení při proměnné délce vektoru v prostředí LabVIEW
	Prog_DP_3b_LV.vi	Vykreslení při proměnné funkci v prostředí LabVIEW
	Prog_DP_4_LV.vi	Back-Propagation s pevným krokem učení v prostředí LabVIEW
\PROGRAMY\MathScript	Prog_DP_1.m	Operace s maticemi v prostředí MathScript
	Prog_DP_2.m	Operace s cykly v prostředí MathScript
	Prog_DP_3a.m	Vykreslení při proměnné délce vektoru v prostředí MathScript
	Prog_DP_3b.m	Vykreslení při proměnné funkci v prostředí MathScript
	Prog_DP_4.m	Back-Propagation s pevným krokem učení v prostředí MathScript
\TEXT	0_DP_vypocty.xls	Data a výpočty k Diplomové práci
	1_DP_uvod.pdf	Úvodní list Diplomové práce.
	2_DP_zadani.tif	Zadání Diplomové práce
	3_DP_zacatek.doc	Abstrakt, citace, prohlášení k Diplomové práci
	4_DP_text.doc	Vlastní text Diplomové práce
	5_DP_prilohy.doc	Přílohy k Diplomové práci
	Diplomova_prace.pdf	Kompletně seřazená Diplomová práce
\	ctete.txt	Informační soubor s obsahem CD