MATLAB a SIMULINK

úvod do používání

Ing. František Dušek, CSc.

Univerzita Pardubice

2000

Děkuji firmě HUMUSOFT Praha, díky jejíž pomoci bylo možné tato skripta realizovat. Jmenovitě panu Ing. Petru Byronovi za vstřícný a konstruktivní přístup.

ISBN 80-7194-273-1

Obsah

Předmluva

Tato skripta vychází z učebního textu "*Úvod do používání MATLAB*", který vznikl v roce 1997 a byl napsán jako pracovní materiál pro předmět základního kurzu MATLAB, a ze zkušeností se třemi roky výuky tohoto předmětu. Jsou určena především pro studenty technické vysoké školy a při jejich psaní jsem byl veden snahou studenty neodradit už při čtení skript a nezahltit je množstvím podrobností, při studiu kterých se ztrácí smysl proč se to vlastně dělá. Proto nechť laskavý čtenář promine poněkud lehčí tón některých pasáží[1]. Také je potřeba hned na začátku upozornit na to, že se předpokládá, že čtenář má možnost[2] MATLAB používat, příklady si zkusit[3] a využívat vestavěnou nápovědu.

Skripta by měla pomoci vyřešit dva problémy - jednak relativní nedostupnost (hlavně finanční[4]) literatury o MATLABu pro studenty a jednak nedostatek stručné dokumentace pro základní praktické používání.

První část skript je určena pro získání znalostí jak používat MATLAB při řešení úloh všeobecného zaměření. Pro plné porozumění příkladům se předpokládá znalost matematiky (a pro komplexní příklady i fyziky) na úrovni I. ročníku VŠ technického směru. Nezabývá se používáním Toolboxů, které jsou zaměřeny na určitou oblast a jejich používání vyžaduje speciální znalosti z daného oboru.

Druhá část skript je věnována používání nadstavby MATLABu - SIMULINK. Tuto část lze studovat samostatně. Avšak vzhledem k tomu, že jde o nadstavbu MATLABu, je pro důkladnější pochopení a využití možností SIMULINKu vhodné mít aspoň základní znalosti MATLABu.

Jednou z předností MATLABu by mělo být, že není potřeba o něm příliš vědět a již je možné ho využívat i pro poměrně náročné výpočty. Tento text by měl představovat **stručný souhrn vybraných informací**[5] pro efektivní používání MATLABu. V žádném případě nelze text považovat za referenční příručku spíš za učebnicí. Naleznete zde proto jen některé vybrané základní příkazy a funkce, jejich použití v příkladech a upozornění na některé problémy či chyby často se vyskytující při běžném používání.

Cílem autora nebylo pouze popsat některé funkce a možnosti jednoho z nástrojů pro numerické řešení úloh, ale pokud možno i ukázat způsob řešení, respektive přístup k řešení problému. Chtěl bych zdůraznit, že mocný nástroj a jeho znalost sama o sobě žádný problém nevyřeší. Teprve když jsem daný problém zvládl (rozumím principu, vím co a proč to dělám) a jsem schopen navrhnout postup řešení, má význam sáhnout po nějakém nástroji umožňujícím konkrétní řešení. Samozřejmě znalost používaného nástroje a jeho možností může významně ovlivňovat návrh řešení. Výsledný postup jak návrhu řešení, tak i vlastního řešení je pak efektivní a rychlý. Na řešených příkladech kapitoly 9 by mělo být toto zpětné ovlivnění postupu řešení znalostí možností řešícího nástroje vidět (aspoň je to autorovo zbožné přání).

Jak již bylo řečeno skripta jsou rozdělena do dvou částí. První část - kapitola první až desátá jsou věnovány vlastnímu MATLABu. Část druhá - kapitoly deset až čtrnáct - se zabývají SIMULINKem. Uspořádání by mělo sledovat přirozený postup učení se něčemu novému. Nejprve je potřeba vyvolat zájem o to, co se budu učit.

K čemu je to dobré, proč zrovna toto a ne něco jiného, mám vůbec na to (myšleno hardware a software) - *kapitola první* **Úvod**.

Potom následuje první kontakt (co uvidím, na co mám máčknout, jak se to chová, jak se s tím zachází) - *kapitola druhá* **Začínáme.**

---

[1] , který má za účel zmírnit míru znechucení nešťastníka nucenčho tento text studovat

[2] zcela dostačující je The Student Edition of MATLAB - součást publikací [152,153]

[3] Neustále potvrzovaná zkušenost ukazuje, že pokud je studentům něco vysvětlováno, tvrdí, že je to jasné. Teprve v okamžiku, kdy mají sami něco udělat, zjistí, že to je jasné jako tunel.

[4] tuto větu píši v okamžiku, kdy se ještě neví co budou skripta stát. Doufám, že bude platit i v době, kdy bude výsledná cena známa.

[5] znalci MATLABu nechť prominou, že jsem vynechal (ať už úmyslně či neúmyslně) velké množství funkcí a možností, které MATLAB poskytuje a těmi, na které se dostalo, se zabývám jen stručně a obvykle jen v té nejjednodušší podobě

Když už vím co se mačká začne věda. Mělo by mě zajímat s čím se vlastně hlavně pracuje, s jakými základními prvky, na jakých principech je to postavené - *kapitola třetí* **Základní operace s maticemi**.

Umí to ale také něco složitějšího než prostou posloupnost příkazů ? Jaké jsou možnosti řešení složitějších případů ? - *kapitola čtvrtá* **Řízení průběhu výpočtu**.

Výpočty jsou sice hezké ale obrázek je obrázek (jestli je to dobře nevím, ale obrázek vypadá hezky) - *kapitola pátá* **Vizualizace**.

Jsem líný[6]. Musím pokaždé znovu vypisovat všechny příkazy? A co když chci provést ten samý výpočet pouze s jinými daty, lze si ulehčit práci? - *kapitola šestá* **Tvorba skriptů a funkcí**.

Co užitečného to ještě standardně umí? - *kapitola sedmá* **Funkce funkcí**.

Proboha, je z toho zkouška. Co asi po nás bude chtít? - *kapitola osmá* **Jednoduché řešené příklady (MATLAB)**.

Zaplať Pán Bůh, že je už konec, dál už to stejně číst nebudu (něco konkrétního řešit nebo dokonce pracovat to je až ta poslední možnost jak se živit) - *kapitola devátá* **Komplexní řešené příklady (MATLAB)**.

Poslední kapitolou věnovanou vlastnímu MATLABu je *kapitola desátá* **Další možnosti MATLABu**, která obsahuje stručný přehled některých dalších vlastností a možností, které jsou v posledních verzích k dispozici a autor je považoval za natolik zajímavé či užitečné, že stojí za to se o nich zmínit.

Následující kapitoly - počínaje *kapitolou jedenáctou* **Co je to SIMULINK** - jsou věnovány nadstavbě MATLABu pro simulaci[7] (časové chování) dynamických soustav.

Dvanáctá kapitola **Jak se v SIMULINKu pracuje** se zabývá způsobem grafického zápisu simulovaného problému - modelu, základními bloky standardních knihoven a parametry vlastní simulace.

V *kapitole třinácté* **Subsystémy a knihovny** jsou ukázány možnosti usnadnění tvorby složitých či opakovaných modelů.

Poslední *kapitola čtrnáctá* **Řešené příklady (SIMULINK)** je věnována konkrétním příkladů na použití SIMULINKu. Tyto komentované příklady ukazují použití nejdůležitějších bloků standardních knihoven a postup konstrukce modelu ve složitějších případech.

Pro uspokojení autora a pro ty, kteří by chtěli využívat tento text, aby se něco naučili, je na konci ještě uveden **Rejstřík** názvů, funkcí a pojmů, které se v textu objevují a autor považoval za užitečné je samostatně uvést. A úplně nakonec poměrně rozsáhlý seznam **Literatury**, která se na MATLAB odkazuje, řazený podle technických oborů.

I když jsem byl veden snahou popisovat používání MATLABu v podobě pokud možno nezávislé na verzi MATLABu a použité platformě, byly všechny použité funkce a příkazy ověřovány na verzi 5.2.1 pod Windows 95, kterou jsem měl k dispozici. Též část věnovaná SIMULINKu[8] vychází z možností verze 2.2.1 pro Windows a o případných omezeních či rozšířeních na platformě UNIX není autor informován. Z tohoto pohledu je kapitola 10 výjimkou, neboť se zabývá možnostmi poskytovanými posledními[9] verzemi.

---

[6] dle názoru autora jedna z dominantních lidských vlastností

[7] dovoluji si zdůraznit, že jde o plnohodnotný simulační prostředek - ne pouze numerické řešení soustavy nelineárních diferenciálních rovnic. To znamená že má vlastnosti očekávané u prostředků pro simulaci (např. automatické přizpůsobení okamžiku výpočtu při nespojitých nelinearitách, volbu různých metod výpočtu, řešení počátečních vnitřních stavů atd.)

[8] SIMULINK jako nadstavba MATLABu se objevil až s verzí 4 pod Windows. Jeho použití je silně závislé na grafických možnostech prostředí, ve kterém je MATLAB spouštěn

[9] možnosti verze 5.3, která je momentálně poslední, jsou uváděny podle informací získaných z podkladů konference MATLAB'99 pořádané firmou Humusoft v listopadu 1999

Pokud mi některý z čtenářů bude chtít sdělit jaké chyby v těchto skriptech objevil, co mu chybělo či co si myslí, že je naopak zcela zbytečné, může tak učinit nejlépe elektronickou poštou na adrese **frantisek.dusek@upce.cz**.

v Hradci Králové 29.února 2000 *František Dušek*

Rád bych poděkoval Ing. Danielu Honcovi za pečlivé přečtení textu, cenné připomínky a v neposlední řadě i za pomoc při závěrečném zalomení skripta.

1. Úvod

V překotně se vyvíjejícím světě výpočetní techniky a programového vybavení mají od prvopočátku nezastupitelné místo programy pro numerické výpočty. Zpočátku byla podpora uživatele realizována matematickými knihovnami pro obecné programovací jazyky. Později se vyvinuly samostatné programy, které je možné rozdělit asi do tří oblastí - tabulkové procesory s rozšířenými matematickými funkcemi, balíky specializované na řešení určitého okruhu problémů (statistika, kvantová chemie, chemické inženýrství) a profesionální univerzální matematické balíky. Do posledně jmenované oblasti patří mj. MathCad, Mathematica a MATLAB[10].

1.1 Co to je MATLAB ?

Název MATLAB vznikl z anglického *matrix laboratory*. MATLAB byl napsán, aby poskytoval jednoduchý přístup k matematickým knihovnám vyvinutým v projektech LINPACK a EISPACK. Byl původně určen pro operační systém UNIX a tato okolnost se dodnes (i v prostředí Windows) projevuje ve velmi jednoduchém základním komunikačním rozhraní - příkazové řádce.

Co to tedy je MATLAB ? Slovy firemní literatury *"MATLAB je vysoce výkonný jazyk pro technické výpočty. Integruje výpočty, vizualizaci a programování do jednoduše použitelného prostředí, kde problémy i řešení jsou vyjádřeny v přirozeném tvaru"*. Jde o interaktivní systém jehož základním datovým typem je dvourozměrné[11] pole (bez nutnosti deklarovat rozměry). Tato vlastnost spolu s množstvím zabudovaných funkcí umožňuje relativně snadné řešení mnoha technických problémů, speciálně takových, které vedou na vektorovou či maticovou formulaci, v mnohem kratším čase než řešení v klasických jazycích jako je C nebo FORTRAN. Typické oblasti použití jsou:

- inženýrské výpočty
- vývoj algoritmů
- modelování, simulace a vývoj prototypů
- analýza dat a jejich vizualizace
- inženýrská grafika
- vývoj aplikací včetně tvorby grafického uživatelského rozhraní

V univerzitním prostředí jde o standardní nástroj využívaný ve výuce matematiky a inženýrských oborech. V průmyslu je využíván jako vysoce efektivní nástroj pro výzkum, vývoj i analýzu dat. O tom, že je skutečně používán svědčí přes 300 publikací z různých oborů, které MATLAB při řešení problémů používají a řešení ve formě m-funkcí v knize nejen uvádějí, ale ve formě souborů buď na doprovodné disketě (CD) nebo na některém FTP serveru nabízejí. Seznam literatury využívající MATLAB evidovaný firmou MathWorks k 29.2.2000 a stažený z jejího FTP serveru (FTP.MATHWORKS.COM) je v poslední kapitole.

Aktuální informace o MATLABu lze získat na WWW serveru (http://www.mathworks.com) fy MathWorks nebo českého výhradního distributora (http://www.humusoft.cz) fy HUMUSOFT Praha.

1.2 Kdy použít MATLAB ?

Ve srovnání s jinými obdobnými produkty je MATLAB blíže programovacímu jazyku. Nemá tolik předpřipravených komplexních matematických funkcí jako např. Mathematica, nemá integrované vlastnosti

---

[10] MathCad® je chráněná značka fy MathSoft, Inc.

Mathematica® je chráněná značka fy Wolfram Software Inc.

MATLAB® je chráněná značka fy The MathWorks, Inc.

[11] od verze 5 existuje možnost obecnějšího datového typu - vícerozměrného pole

vého editoru jako např. MathCad. Zrovna tak není jeho standardní součástí podpora symbolických výpočtů jako u obou výše jmenovaných produktů.

Ale to neznamená, že je o tyto možnosti ochuzen. Obsahuje množství (více než 500) jednoduchých[12] i složitějších[13] matematických funkcí implementovaných ve formě vysoce efektivních a robustních algoritmů, které jsou součástí jádra (built-in function). Z těchto funkcí lze složit podle konkrétní aplikace libovolně složité další funkce. Skupiny takovýchto funkcí respektive funkcí hodících se k řešení určitého okruhu problémů se v MATLABu nazývají Toolboxy[14]. Obvykle byly vytvořeny (nebo aspoň jejich základ) na univerzitách týmem seskupeným okolo významného odborníka v dané oblasti. Velkou výhodou, aspoň pro teoretické využití, je to, že dokumentace k toolboxům obsahuje stručný princip a odkazy na literaturu, kde jsou použité algoritmy podrobně rozebrány.

Jako samostatná nadstavba MATLABu existuje SIMULINK - řešení soustavy nelineárních diferenciálních rovnic s grafickým zadáváním řešené soustavy připomínajícím zapojení na analogovém počítači. Umožňuje graficky sledovat průběhy veličin v libovolném místě zapojení. Používá se např. pro simulaci dynamického chování sledovaného systému.

Symbolickou matematiku a výpočty s definovanou přesností[15] lze zahrnout pomocí Symbolic Math Tbx. Tento toolbox zahrnuje výpočetní jádro MAPLE V[16].

Pro případ, že potřebujeme výpočty v MATLABu jako součást např. zprávy, je součástí instalace pro operační systém (OS) Windows na PC a počítače MacIntosh tzv. Notebook neboli šablona pro textový procesor MS Word, která umožňuje volat příkazy MATLABu a vkládat výsledky (včetně obrázků) přímo v prostředí editoru. Jako ukázka může sloužit přímo tento text, který je tímto způsobem tvořen.

Dalším rozšířením (se kterým nemá autor konkrétní zkušenost) je propojení MS Excelu s MATLABem pro snazší zadávání dat a prezentaci výsledků.

Mezi další možnosti MATLABu, se kterými autor nemá zkušenosti patří například

- vývoj aplikací (filtrace signálů, řízení) pro signálové procesory
- možnost překladu odladěného algoritmu do funkce v jazyce "C" použitelné ve vlastním programu
- převod hotové aplikace v MATLABu (včetně grafického uživatelského rozhraní) do samostatného spustitelného tvaru (bez nutnosti přítomnosti jádra MATLABu)
- použití MATLABu jako WEB serveru pro zajištění výpočtů v rámci WWW stránky

Kdy tedy MATLAB použít? Odpověď na tuto otázku je velice důležitá a měl by si ji položit každý. Odpověď není jednoznačná. Záleží na konkrétních požadavcích a podmínkách konkrétního uživatele. Obecné zdůvodnění snad lze shrnout v následujících větách.

V případě, že potřebuji robustní výpočty, zpracovávat rozsáhlé datové soubory, pracovat s velkými maticemi a vůbec v případech, kdy řešení problému se dá převést na vektorové či maticové operace. Vzhledem k možnostem programování je MATLAB s výhodou použitelný i v případech rozvětvených případně iterač-

---

[12] všechny běžné matematické funkce (pracující i s komplexními hodnotami) a základní operace s maticemi

[13] např. Besselovy funkce, gama funkce, generování prvočísel, transformace souřadnic, Fourierova transformace, maticové funkce, funkce pro práci s řídkými maticemi, funkce pro práci s polynomy, numerická integrace a derivace funkce, nulová hodnota fce, minimum funkce více proměnných, řešení soustavy diferenciálních rovnic I. řádu atd.

[14] problémově orientované (specializované) soubory funkcí pro řešení problematiky určitých oborů - Control System, Communications, Financial, Frequency Domain System Identification, Fuzzy Logic, Higher-Order Spectral Analysis, Image Processing, LMI Control, Model Predictive Control, μ-Analysis and Synthesis, NAG Foundation, Neural Network, Optimizatiom, Partial Differentuial Equation, QFT Control Design, Robust Control, Signal Processing, Spline, Statistics, Symbolic Math, System Identification

[15] např. hodnotu s π přesností 500 cifer

[16] MAPLE® je chráněná značka Waterloo Maple Software, Inc.

ních algoritmů řešení. Nezanedbatelnou výhodou je i to, že díky jednoduchému ovládání lze po cca 30 minutovém[17] vysvětlení s MATLABem samostatně pracovat a řešit jednodušší příklady [132,133]. Dále je běžným prostředkem používaným na většině univerzit technického zaměření na světě. Také to, že MATLAB je k dispozici na všech běžných platformách může při výběru hrát roli.

Kdy použití MATLABu není vhodné? Použití MATLABu není vhodné v situaci, kdy potřebujeme provádět opakovaně stále stejné (byť složité) výpočty nebo opakovaně zpracovávat velké objemy dat. Zkrátka v situacích, kdy je rozhodující vlastní rychlost výpočtu a ne vývoj algoritmu výpočtu. V tomto případě je vhodné hotový algoritmus zapsat v nějakém programovacím jazyce, který umožní vytvořit spustitelný kód a nepoužívat MATLAB, který je stále jen interpretr.

1.3 Jak se vyvíjel MATLAB

Jak již bylo uvedeno výše, původně byl MATLAB vyvinut pro počítače pracující pod OS Unix. Především pro Unix (na různých platformách[18]) je určen MATLAB dodnes. Donedávna nebyl výkon počítačů PC dostatečný pro profesionální práci. Dnes už to sice není tak úplně pravda, ale firma MathWorks nepovažuje PC za nástroj pro profesionální práci. Svědčí o tom krom jiného i ta okolnost, že verze MATLABu pro PC nejsou nikterak chráněny proti nedovolenému kopírování na rozdíl od verzí pro Unix, kde je instalace vázána na sériové číslo procesoru. Předcházející dvě věty byly napsány v roce 1997 - dnes (rok 2000) je již situace jiná. Platforma na bázi procesorů INTEL je již výkonově srovnatelná s RISCovými procesory. Na platformě Windows se využívá pro ochranu nedovoleného kopírování MATLABu toho, že je nutné (do Windows 95/98/NT) programy instalovat. Instalační program z instalačního CD, které obsahuje vše k příslušné verzi MATLABu (tzv. RELEASE), po zadání příslušného 25ti místného čísla dovolí instalovat pouze ty části, které si uživatel zakoupil.

První verze pro PC XT se objevila kolem roku 1985. Základním problém byl nedostatek paměti - a z toho plynoucí omezení na maximální velikost matic. Potom, co se objevilo PC AT rychle následovala verze MATLABu pro tento počítač. Zde byla maximální velikost matice omezena fyzickou velikostí paměti (pro PC AT max. 16 MB). Vzhledem k tehdejším cenám pamětí nebylo (zvláště v našich podmínkách) osazení větší pamětí běžné. Velké obliby doznala verze MATLAB 386, která byla, jak už název naznačuje, určena pro počítače PC s procesorem 80386. Využívala jednu z možností tohoto procesoru - virtuální paměť. Což znamená, že program pracuje s virtuální pamětí, která může být větší než je skutečná fyzická paměť. Dochází k odkládání dat na pevný disk počítače. Tato skutečnost sice vede (pakliže k odkládání dojde) k zpomalení výpočtů , ale je možné výpočty na velkých maticích[19] vůbec provést. Poslední verzi MATLAB386 g z října 1995 mnozí používají dodnes, protože je, vzhledem k tomu, že pracuje pod OS DOS, nepřekonatelná v rychlosti výpočtů. Všechny tyto verze MATLABu pracovaly pod OS DOS.

Přibližně v roce 1994 byla uvedena na trh verze MATLAB for Windows, která samozřejmě nabízela podstatně bohatší možnosti grafického rozhraní než předchozí verze. Po výpočetní stránce mnoho nového nenabízela (naopak výpočty na tom samém počítači byly díky většímu vytížení procesoru OS pomalejší). MATLAB for Windows skončil verzí 4.2c, která se přestala dodávat koncem roku 1996.

Nástupcem verze 4 je verze 5 určená v rámci platformy Intel do 32bitového prostředí (pod OS Windows 95 nebo Windows NT). Momentálně poslední verzí je MATLAB 5.3 z konce roku 1999. Tato verze je plně

---

[17] nutno brát s rezervou - záleží na zkušenostech a inteligenci potenciálního uživatele

[18] v BENCHi dodávaném s verzí 5.1 jsou uvedeny pro porovnání doby provádění testovacího programu na těchto počítačích: SUN Ultra 2/200, SGI Impact 1000/195, PentiumPro 200/LINUX, PentiumPro 200/Win95, PentiumPro 200/WinNT, IBM RS6000/3BT/67, DEC Alpha 400/233, MAC 8500/180, HP 735/125, SGI RS4400/150, SUN Sparc 10/41, Mac 5300c Powerbook, SUN Sparc 2.

Ve verzi 5.2: DEC Alpha 500, Sparc Ultra 2 (Dual 300), SGI Octane 195, Power Mac G3 /500, HP 780 /180, Mac Powerbook G3 /250, SGI O2 /180, Pentium II NT /300

[19] např. na počítači PC386/40 s 4 MB fyzické paměti bylo možné provést výpočet inverze matice 1000x1000. Samotná matice potřebuje pro uložení 8 milionů bytů tj. skoro 8 MB. Výpočet sice trval několik minut, ale byl proveditelný.

32bitová a obsahuje nové možnosti plynoucí z využití objektového programování - nové datové struktury[20] rychlejší grafiku, nové vizualizační funkce a hlavně další rozšíření nabídky základních funkcí.

1.4 Požadavky na instalaci

Všechny verze od prvopočátku bezpodmínečně vyžadují numerický koprocesor. Dnes v době procesorů Pentium je to sice samozřejmost, ale v případě, že chceme využít starší počítač či notebook je potřeba na to myslet. Další požadavky se liší podle toho, jakou verzi MATLABu máme k dispozici. V případě verze 5 jsou požadavky víceméně dané tím, že lze instalovat pouze na počítač s Windows 95/98 či Windows NT 3.51 či vyšší. Doporučená paměť je 16 MB a vzhledem k grafickým možnostem je vhodná rychlá grafická karta. Požadavek na velikost diskového prostoru je okolo 100-180 MB (podle velikosti alokační jednotky). Instalace se provádí z CD, které obsahuje kompletní nabídku a po zadání tzv. Personal Licence Code (25 číslic) se uvolní instalace částí, na které je zakoupena licence.

Komplexní on-line nápověda vyžaduje instalovaný WWW prohlížeč např. standardně s Windows 95 dodávaný Microsoft Internet Explorer. Úplná verze elektronické dokumentace je ve formátu Adobe PDF. K jejímu prohlížení a tisku je nutný program fy Adobe Systems Inc. - Acrobat Reader. Tento program je volně šiřitelný a je možné jej získat např. z Internetu (http:\\www.adobe.com\acrobat - verze 3.0 je součástí instalačního CD). V případě připojení k internetu lze využívat systém nápovědy na firemním serveru.

Pro využití ve školství jsou jednak poskytovány slevy a jednak pro vlastní výuku je možnost zakoupení multilicence tzv. Classroom Kit. Od 1.1.2000 došlo ke změně licenční politiky fy MathWorks v této oblasti. Jsou vytvořeny nové Studentské licence. Předpokládá se, že tyto licence zakoupí škola a bude je studentům na dobu studia půjčovat či prodávat.

Začínající uživatelé (předpokládá se univerzitní studenti) mohou využít tzv. **The Student Edition of MATLAB 5**[21] [152] a **The Student Edition of SIMULINK 2** [153]. V obou případech jde o knihu se stručným popisem příslušného produktu a součástí je instalační medium s "demo" verzí, která se od "ostré" verze liší pouze omezením na maximální velikost použitých matic. Cena není sice nejmenší, ale ve srovnání s cenou "ostré" verze pro komerční využití (94 000 Kč) je zanedbatelná.

1.5 Co za nás MATLAB nevyřeší

Jako poslední bod tohoto obecného povídání bych si dovolil upozornit na podstatnou věc, která neplatí jenom pro MATLAB. MATLAB je pouze nástroj, sice výkonný a efektivní ale je to pouze prostředek k dosažení cíle. Musím tedy nezávisle na MATLABu či jiném prostředku vědět:

co chci - mít jasnou a jednoznačnou formulaci problému

jak toho dosáhnout - vědět (aspoň principiálně) jak se daný problém řeší

jak ověřit výsledek - být schopen ověřit či odhadnout zda je získaný výsledek správný

Tyto tři body za nás žádný sebedokonalejší výpočetní systém nevyřeší. Může ale výrazným způsobem ovlivnit efektivitu a rychlost řešení problému. Výkonnost a efektivitu nástrojů typu MATLABu člověk plně ocení teprve až při řešení složitějších a rozsáhlejších problémů.

---

[20] hlavní změnou je možnost používání vícerozměrných matic

[21] cena Student Edition of MATLAB 5.3 je v současnosti (duben 2000) 3780 Kč + 5 % DPH.

2. Začínáme

Příkazy a funkce: `cd, clear, delete, dir, format, help, lookfor, type, who, whos`

Problém: zápis příkazů, editace příkazového řádku, předdefinovaná jména, proměnné a informace o nich, výpis proměnných, pojem funkce, práce se soubory, nápověda

Tato kapitola by měla dát základní informace o používání MATLABu, o principech práce, o možnostech prostředí a další informace nesouvisející s vlastními výpočty. Budeme se zabývat pouze zjednodušeným přístupem, vynecháme mnoho možností, které MATLAB poskytuje. Proto informace zde uvedené (i když jsou předváděny na verzi 5.2) by měly být použitelné pro všechny verze.

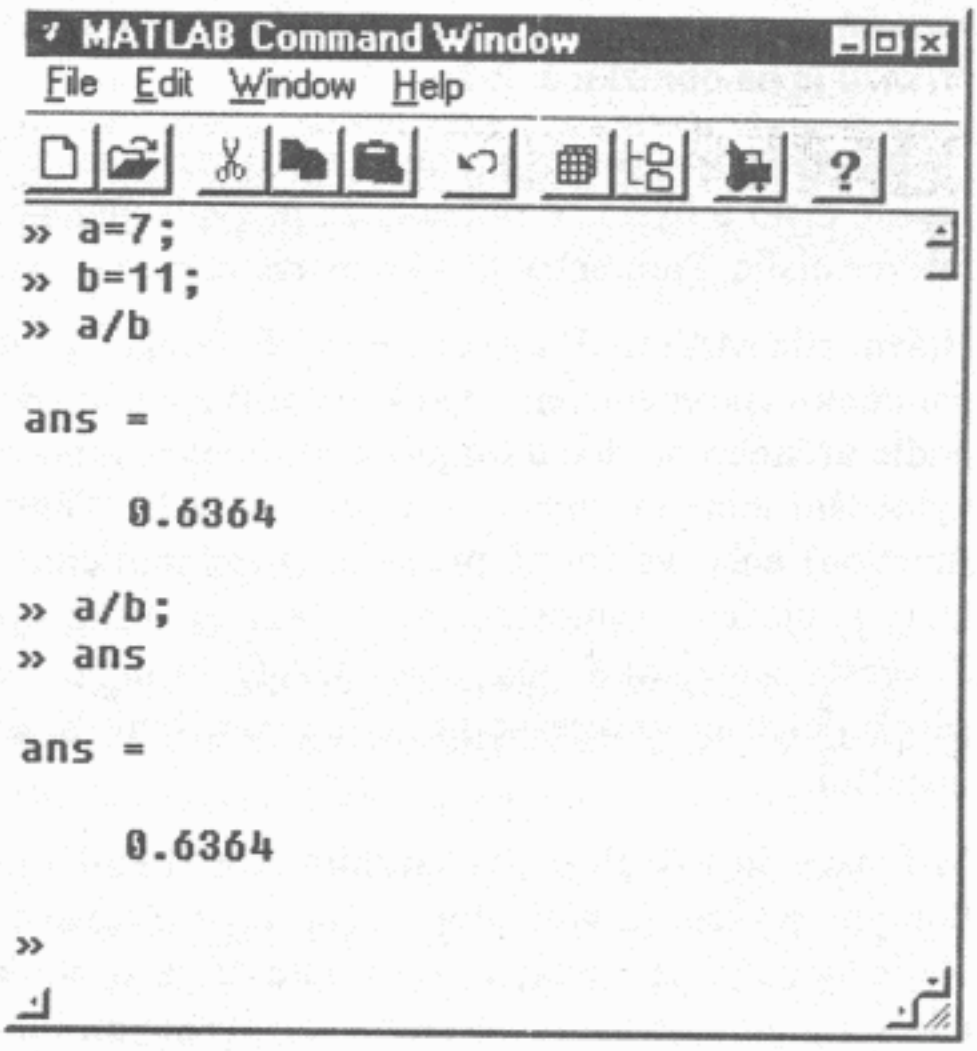
Na začátku bych rád zdůraznil aspoň dvě skutečnosti, které je vhodné si uvědomit. MATLAB používá poměrně hutný způsob zápisu příkazů. Toto je na jednu stranu příjemné, neboť toho nemusíme moc napsat, ale na druhou stranu zápis není příliš přehledný. A hlavně každý znak má své místo a význam - nemůžu libovolně zaměňovat tečku, čárku či středník nebo jiné znaky. Je potřeba dodržovat tzv. syntaxi jazyka - tedy věci, které jsou jasné uživatelům zvyklým používat nějaký programovací jazyk, které ale nejsou samozřejmé pro ostatní.

Druhým zdánlivě samozřejmým faktem je přečíst si pořádně informaci, která se vypíše při výskytu chyby. Souhlasím plně s názorem váženého čtenáře, který si nyní určitě řekl "*on si snad o mě myslí, že jsem úplně blbej*". Vzpomeňte si na toto upozornění až po dlouhém hledání chyby vlastními silami si konečně ze zoufalství přečtete, co Vám to vlastně oznamuje. Na druhou stranu, když si jednoduše přečtu kde je chyba, nezažiji ten slastný pocit vítězství jako v případě, že na chybu přijdu sám.

2.1 Jak to funguje

V nejjednodušším pohledu se můžeme na MATLAB dívat jako na velkou kalkulačku nebo na něco podobného interpretu jazyka BASIC. Příkazy se píší do příkazové řádky. Každý zapsaný příkaz se po stisku klávesy Enter okamžitě provede. V případě, že v příkazu neuvedeme název proměnné, do které se má uložit výsledek, uloží se do proměnné s názvem ans. Pokud příkaz ukončíme znakem středník (;) pak se příkaz pouze provede, ale výsledek se nevypíše[22]. Neuvedeme-li na konci příkazu znak středník, pouze příkaz ukončíme klávesou Enter, výsledek se i vypíše. Obrázek č. 2-1 ukazuje okno MATLABu verze 5 s ilustračním příkladem naplnění proměnných **a** a **b** hodnotami a výpočet podílu **a/b**. Napíšeme-li pouze název proměnné (bez středníku) vypíše se obsah proměnné.

Očekává-li MATLAB příkaz (dokončil-li předchozí příkaz), indikuje tento stav na začátku řádky úvodními znaky >>.



Obrázek 2-1 Okno MATLABu

[22]Tato zdánlivá maličkost může značně znepříjemnit práci. Vzhledem k tomu, že výsledek příkazu může být třeba matice 100x100, pak příkaz bez středníku představuje výpis 10 000 čísel na obrazovku. To znamená, kromě toho že tato informace nemá pro normálního člověka naprosto žádný smysl, že to také dost dlouho trvá.

Další užitečné všeobecně platné informace:

a) chceme-li přerušit provádění příkazu stiskneme kombinaci kláves Ctrl+C.
b) klávesy šipka nahoru/dolu umožňují pohyb v historii dříve napsaných příkazů. Vyhledání příkazu v historii zapsaných příkazů usnadní zapsání několika prvních znaků hledaného příkazu a poté vyhledání pomocí šipka nahoru. Vypisují se pouze příkazy splňující zapsanou masku
c) příkaz na řádku lze editovat tj. pohybovat se klávesami šipka vpravo/vlevo po napsaném textu a klávesami Delete či BackSpace mazat napsaný text a standardním způsobem zapisovat opravy
d) klávesa Esc vymaže celý řádek

2.2 Proměnné a funkce

Proměnné v MATLABu musí mít název začínající písmenem a může být až 31 znaků dlouhý. Pozor **rozlišují se velká a malá písmena**. Můžeme použít libovolný název, ale určitá jména mají dopředu přiřazenou hodnotu. Použijeme-li toto jméno, změníme původní hodnotu. Některá důležitá předdefinovaná jména jsou v následujícím seznamu.

ans	proměnná používaná systémem pro uložení výsledku, když neuvedeme vlastní proměnnou
pi	Ludolfovo číslo (3.14159265358897...)
i,j	imaginární jednotka
Inf	nekonečno (např. výsledek výrazu 1/0)
NaN	neplatná numerická hodnota (např. výsledek výrazu 0/0)
flops	počet operací v pohyblivé řádové čárce od spuštění MATLABu

Proměnné jsou zjednodušeně pouze dvojího typu - komplexní matice[23] nebo řetězec znaků. Proměnné není třeba deklarovat - vytváří se automaticky v potřebné velikosti při prvním použití. Po vytvoření proměnná existuje do ukončení programu nebo do zrušení příkazem **clear** jméno_proměnné_1 jméno_proměnné_2 ... Příkaz **clear** použitý bez názvů proměnných vymaže všechny proměnné.

Během práce můžeme vytvořit velké množství proměnných navíc s různou velikostí . Příkaz **who** vypíše seznam existujících jmen a příkaz **whos** opět seznam jmen, ale navíc s velikostí proměnných. Použití obou příkazů je na obrázku č. 2-2.

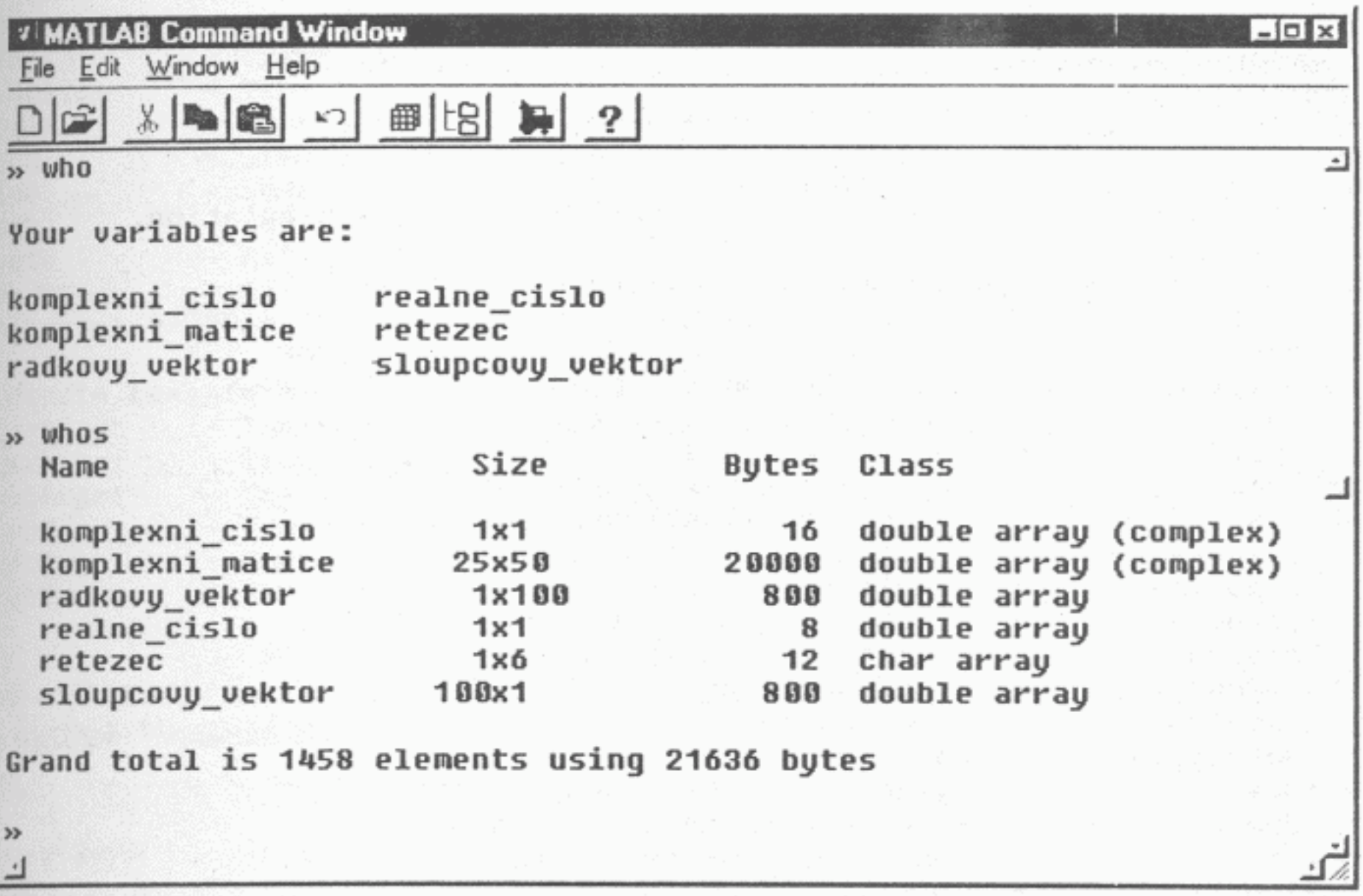
Hodnoty, které proměnné zastupují, jsou v paměti počítače uloženy ve formátu pohyblivé řádové čárky (každé číslo 8 byte). V případě, že je vypisujeme na obrazovku musí být konvertovány z tohoto formátu do řetězce číslic. Parametry této konverze můžeme nastavit příkazem **format**.

Hlavní síla MATLABu je ve funkcích. Funkce je obvykle složitější činnost, která jeden nebo více vstupních parametrů (proměnných nebo konstant) zpracuje do jednoho nebo více výstupních parametrů (proměnných) podle určitého předpisu (algoritmu). Funkce mohou být jednoduché (např. funkce sinus) nebo složité (např. vyhledání minima funkce více proměnných). Funkce mohou být přímo součástí jádra MATLABu (built-in function) nebo ve formě předpisu (kombinujícího matematické operace, built-in funkce či jiné m-funkce), který je uložen v samostatném souboru na disku s příponou .m. Takovéto funkce budeme nazývat m-funkce. Součástí standardní instalace MATLABu je velké množství m-funkcí organizovaných v adresářích sdružujících m-funkce podle jejich zaměření. Seznam adresářů, kde jsou m-funkce uloženy se vytváří při instalaci.

M-funkce se vytvářejí standardním ASCII editorem a proto si uživatel může vytvořit vlastní m-funkce. Je dobrým zvykem (a vřele doporučuji ho dodržovat) uživatelské m-funkce ukládat do samostatného adresáře. Nejjednodušší je nastavit tento adresář jako aktuální (nebo též pracovní) adresář. V případě že aktuální adresář je jiný, je nutné adresář s uživatelskými funkcemi zařadit do systému MATLABovských adresářů.

---

[23] ostatní běžné datové typy jako reálné číslo, vektor jsou vlastně speciálním případem komplexní matic



Obrázek 2-2 Výpis použitých proměnných

Pro manipulaci se soubory a adresáři jsou do MATLABu zahrnuty příkazy podobné jako jsou v OS DOS. Příkaz **cd** použitý bez parametrů vypíše cestu do aktuálního adresáře, s parametry umožňuje změnit aktuální adresář (včetně disku), příkaz **dir** vypíše obsah adresáře, příkaz **type** vypíše obsah určeného souboru, příkaz **delete** vymaže určený soubor.

Z hlediska používání není mezi built-in funkcemi a m-funkcemi rozdíl (uživatelské funkce). Vždy je potřeba vědět jak se jmenuje funkce, která dělá to co potřebuji, kolik má vstupních parametrů a jaký mají jednotlivé parametry význam a potom samozřejmě v jaké formě funkce vrací výsledek (kolik výstupních parametrů a jaký mají jednotlivé parametry význam). Obecně vypadá volání funkce (syntaxe příkazu) následovně:

[prom1,prom2,...] = nazev_funkce(par1,par2,...);

Nyní zbývá jen maličkost - jak tyto informace zjistit.

2.3 Nápověda - HELP

Začínáme-li s MATLABem (a nejenom tehdy) je patrně nejpoužívanějším příkazem[24] příkaz **help**. Tento příkaz nám slouží ke zjištění, jaké funkce máme k dispozici (včetně instalovaných toolboxů) a pokud je známo jméno funkce, pak i k určení syntaxe[25] dané funkce.

Tedy postupně - příkaz **help** sám o sobě vypíše seznam skupin funkcí, které jsou v daném okamžiku k dispozici např.

---

[24] v tomto textu je pojem příkaz používán v obdobném významu jako funkce - v obou případech je to pokyn pro MATLAB, aby něco udělal. Pojem funkce budeme používat pro pokyn jehož základním smyslem je vypočítat nějakou hodnotu (např. sin), pojem příkaz bude používán pro pokyn, která primárně nevrací číslo, ale vykonává nějakou činnost (např. příkaz cd, help, plot)

[25] způsob zápisu, který musí volání dané funkce splňovat

```
help
```

```
HELP topics:
matlab\general        -  General purpose commands.
matlab\ops            -  Operators and special characters.
matlab\lang           -  Programming language constructs.
matlab\elmat          -  Elementary matrices and matrix manipulation.
matlab\elfun          -  Elementary math functions.
matlab\specfun        -  Specialized math functions.
matlab\matfun         -  Matrix functions - numerical linear algebra.
matlab\datafun        -  Data analysis and Fourier transforms.
matlab\polyfun        -  Interpolation and polynomials.
matlab\funfun         -  Function functions and ODE solvers.
matlab\sparfun        -  Sparse matrices.
matlab\graph2d        -  Two dimensional graphs.
matlab\graph3d        -  Three dimensional graphs.
matlab\specgraph      -  Specialized graphs.
matlab\graphics       -  Handle Graphics.
matlab\uitools        -  Graphical user interface tools.
matlab\strfun         -  Character strings.
matlab\iofun          -  File input/output.
matlab\timefun        -  Time and dates.
matlab\datatypes      -  Data types and structures.
matlab\dde            -  Dynamic data exchange (DDE).
matlab\demos          -  Examples and demonstrations.
toolbox\signal        -  Signal Processing Toolbox.
toolbox\control       -  Control System Toolbox.
control\obsolete      -  (No table of contents file)
simulink\simulink     -  SIMULINK
simulink\blocks       -  SIMULINK block library.
simulink\simdemos     -  SIMULINK demonstrations and samples.
simulink\dee          -  Differential Equation Editor
toolbox\local         -  Preferences.
For more help on directory/topic, type "help topic".
```

Zajímají-li nás například speciální matematické funkce, použijeme příkaz

```
help specfun
```

```
 Specialized math functions.
   airy          - Airy functions.
   besselj       - Bessel function of the first kind.
   bessely       - Bessel function of the second kind.
   besselh       - Bessel functions of the third kind (Hankel function).
   besseli       - Modified Bessel function of the first kind.
   besselk       - Modified Bessel function of the second kind.
   beta          - Beta function.
   betainc       - Incomplete beta function.
   betaln        - Logarithm of beta function.
   ellipj        - Jacobi elliptic functions.
   ellipke       - Complete elliptic integral.
   erf           - Error function.
   erfc          - Complementary error function.
   erfcx         - Scaled complementary error function.
   erfinv        - Inverse error function.
   expint        - Exponential integral function.
   gamma         - Gamma function.
   gammainc      - Incomplete gamma function.
   gammaln       - Logarithm of gamma function.
   legendre      - Associated Legendre function.
   cross         - Vector cross product.
```

```
Number theoretic functions.
   factor      - Prime factors.
   isprime     - True for prime numbers.
   primes      - Generate list of prime numbers.
   gcd         - Greatest common divisor.
   lcm         - Least common multiple.
   rat         - Rational approximation.
   rats        - Rational output.
   perms       - All possible permutations.
   nchoosek    - All combinations of N elements taken K at a time.

Coordinate transforms.
   cart2sph    - Transform Cartesian to spherical coordinates.
   cart2pol    - Transform Cartesian to polar coordinates.
   pol2cart    - Transform polar to Cartesian coordinates.
   sph2cart    - Transform spherical to Cartesian coordinates.
   hsv2rgb     - Convert hue-saturation-value colors to red-green-blue.
   rgb2hsv     - Convert red-green-blue colors to hue-saturation-value.
```

Máme-li např. provést transformaci mezi kartézskými a sférickými souřadnicemi, potřebujeme znát syntaxi funkce cart2sph. Napíšeme tedy

```
help cart2sph

   CART2SPH Transform Cartesian to spherical coordinates.
    [TH,PHI,R] = CART2SPH(X,Y,Z) transforms corresponding elements of
    data stored in Cartesian coordinates X,Y,Z to spherical
    coordinates (azumith TH, elevation PHI, and radius R).  The arrays
    X,Y, and Z must be the same size (or any of them can be scalar).
    TH and PHI are returned in radians.
    TH is the counterclockwise angle in the xy plane measured from the
    positive x axis.  PHI is the elevation angle from the xy plane.
    See also CART2POL, SPH2CART, POL2CART.
```

Užitečný je také příkaz **lookfor**, za který když uvedeme slovo, pak jsou vyhledány všechny výskyty tohoto slova v nápovědách všech funkcí. Např. když chceme vědět, kde se vyskytuje slovo gamma, napíšeme

```
lookfor gamma
   GAMMA  Gamma function.
   GAMMAINC Incomplete gamma function.
   GAMMALN Logarithm of gamma function.
```

Od verze 5 pod Windows je součástí[26] Help tzv. Helpdesk tj. struktura souborů s nápovědou v HTML formátu využívající hypertextové odkazy. Vyžaduje instalovaný prohlížeč (Internet Explorer, Netscape atd.). K dispozici je i elektronická dokumentace v PDF formátu.

[26] zda má či nemá být nápověda v HTML a PDF formátu k dispozici se zadává při instalaci MATLABu. Kompletní nápověda je značně náročná na diskový prostor. Na instalačním CD je v nezakodované podobě.

3. Základní operace s maticemi

Příkazy a funkce: `conv, det, diag, exp, eye, interp1, inv, length, load, log, magic, ones, polyfit, polyval, rand, randn, roots, save, sin, size, sqrt, wavread, wavwrite, zeros`

Problém: plnění vektorů a matic, aritmetická řada čísel, komplexní čísla, transpozice, inverze, maticové násobení a dělení, operace prvek po prvku, speciální znaky, načtení a uložení dat do souboru, operace s polynomy

Kapitola třetí se zabývá základními operacemi s vektory a maticemi - jejich vytvořením a naplněním hodnotami. Ukážeme si jak vytvořit některé speciální tvary matic, jak pracovat se sloupcem či řádkem matice. Dále si ukážeme rozdíl mezi maticovými operacemi a operacemi mezi jednotlivými prvky matic. Také si ukážeme využití vektorového zápisu pro práci s polynomy.

Na základě zkušeností s výukou doporučuji této kapitole (práci s maticemi) věnovat dostatečnou pozornost, neboť nedovolené operace (násobení matic s nesouhlasnými rozměry) s maticemi jsou nejčastější chybou vyskytující se při práci s MATLABem.

Kompletní seznam příkazů a funkcí nabízených MATLABem pro práci a manipulaci s maticemi je uveden v kapitole 3.4.

3.1 Vytváření vektorů a matic

Skalár je matice s oběma rozměry rovnými jedné. Vektor je matice, kdy jeden rozměr má hodnotu jedna. Rozlišujeme vektory sloupcové a řádkové. Je zvykem, že první rozměr označuje počet řádků a druhý počet sloupců. Řekneme-li, že matice je rozměru 7x12, znamená to, že má 7 řádků a 12 sloupců. Pokud se hovoří o vektoru bez upřesnění (sloupcový či řádkový) předpokládá se sloupcový vektor. Toto platí jak v matematických zápisech tak i v MATLABu. Pro vyjádření opačného vektoru se používá symbol transpozice[27]. V matematice používané označení transpozice (velké T jako pravý horní index tj. ^{T}) je v zápisu MATLABu nahrazeno znakem ' (apostrof). Běžný matematický zápis

$$a^T = [1,2,3,4,5] \qquad b = [1,2,3,4,5]^T \qquad c = [1,2,3,4,5]$$

pak vyjadřuje, že a ,b jsou sloupcové vektory a c je řádkový vektor. Všechny o pěti prvcích s hodnotami 1,2,3,4,5.

Na matici můžeme pohlížet i jako na vektor vektorů. Potom nepřekvapí zápis matice též pomocí již definovaných vektorů či matic

$$A = \begin{bmatrix} 1\,2\,3\,4\,5 \\ 1\,2\,3\,4\,5 \end{bmatrix} = \begin{bmatrix} c \\ 1\,2\,3\,4\,5 \end{bmatrix} = \begin{bmatrix} a^T \\ b^T \end{bmatrix} = \begin{bmatrix} c \\ c \end{bmatrix}$$

[27] transpozici si nejsnáze představíme tak, že matici (vektor) doplníme na čtvercovou matici, tato se otočí kolem hlavní diagonály a odebere se přidaná část. To znamená, že z matice 2x5 se stane matice 5x2 a z řádkového vektoru se stane sloupcový vektor

3.1.1 Plnění vektorů a matic

Zápis vektorů a,b,c a matice A v MATLABu pak vypadá následovně

```
a=[1;2;3;4;5]
```
sloupcový vektor (oddělovač středník) s výpisem (na konci příkazu není středník)

```
   a =  1
        2
        3
        4
        5
```

```
c=[1,2,3,4,5]
```
řádkový vektor (oddělovač čárka nebo mezera) s výpisem

```
   c =  1     2     3     4     5
```

```
b=[1,2,3,4,5]';
```
sloupcový vektor (oddělovač čárka, transpozice tj. znak apostrof) bez výpisu (středník za příkazem)

Zápis matice s uvedením všech prvků je možný např.

```
A=[1,2,3,4,5;1,2,3,4,5]
```
dva řádkové vektory mezi sebou oddělené středníkem s výpisem

```
   A =  1     2     3     4     5
        1     2     3     4     5
```

Využijeme-li již existujícího vektoru c můžeme vytvořit matici A jednodušeji

```
A=[c;c]
```

```
   A =  1     2     3     4     5
        1     2     3     4     5
```

Následující příklad ukazuje nepřípustnou kombinaci (kombinuje do matice sloupcový a řádkový vektor - MATLAB ohlásí chybu)

```
C=[a,c]
??? All matrices on a row in the bracketed expression must have the
 same number of rows.
```

Potřebujeme-li pracovat s komplexními čísly, použijeme zápis ekvivalentní algebraickému zápisu komplexních čísel. Využijeme předdefinovanou konstantu i nebo j - imaginární jednotku. Např. vektor s kombinací reálných a komplexních čísel získáme

```
x=[1.23,7-3.6*i,1/7+3/11*i,pi*i]
   x = 1.2300    7.0000- 3.6000i    0.1429+ 0.2727i    0+ 3.1416i
```

Často potřebujeme vytvořit rozsáhlý vektor tvořený aritmetickou řadou např. od 0.75 s krokem 0.05 do 10. Takovýto vektor vytvoříme následujícím příkazem

```
v=0.75:0.05:10;
```
počáteční hodnota : krok : konečná hodnota

Nyní nás zajímá jaký se vlastně vektor vytvořil tj. kolik má prvků a jakého rozměru. Na zjištění těchto vlastností použijeme příkaz **size.** Tento příkaz vrací dvě hodnoty - počet řádků a počet sloupců. V následujícím příkladu použijeme tento příkaz, necháme uložit hodnoty do proměnných a zároveň vypsat na obrazovku

```
[poc_radku,poc_sloupcu]=size(v)
  poc_radku =      1
  poc_sloupcu =    186
```

neuvedeme-li výstupní parametry funkce **size** a samozřejmě ani ukončení středníkem, vypíšou se pouze hodnoty na obrazovku respektive uloží se do speciální proměnné **ans** a vypíšou

```
size(v)
  ans =  1    186
```

Vektor v má 186 prvků a je řádkový. Pro zjištění počtu prvků (délky) vektoru slouží příkaz **length**.

Vektory vzniklé plněním tímto způsobem (aritmetická řada) jsou řádkové. Chceme-li vytvořit sloupcový vektor, použijeme transpozici. Chceme-li vytvořit tímto způsobem matici použijeme např. tento příkaz. Všimněte si, že v prvním řádku je vynechán krok. V takovém případě MATLAB použije hodnotu 1

```
B=[1:5;5:-1:1]
  B =  1     2     3     4     5
       5     4     3     2     1
```

Shrňme nyní používané speciální znaky a jejich význam. Zdůrazňuji, že není jedno zda napíšeme čárku či středník, kulatou či hranatou závorku - každý znak má svůj přesný význam.

;	*středník*	ukončení příkazu bez výpisu, oddělovač sloupců
,	*čárka*	obecný oddělovač, oddělovač v řádku
	mezera	obecný oddělovač, oddělovač v řádku
:	*dvojtečka*	oddělovač ve výčtu (určení rozsahu)
'	*apostrof*	označení transpozice (umístěn za proměnnou)
()	*kulaté závorky*	přednost v mat. výrazu, seznam parametrů funkce, index matice
[]	*hranaté závorky*	určují, že jde o vektor či matici

3.1.2 Vektory a matice se speciálními hodnotami

Často jsou potřeba matice naplněné speciálními hodnotami. Jedním z nejčastěji se vyskytujících požadavků je vytvoření matice (vektoru) zadaných rozměrů naplněných konstantou. MATLAB poskytuje dvě funkce pro tyto účely. Funkce **zeros** vytvoří a naplní matici hodnotou nula a funkce **ones** hodnotou jedna. Následující seznam uvádí příklady použití funkcí vytvářejících některé často se vyskytující speciální matice

x=zeros(11,50)	matice o rozměrech 11x50 naplněná nulami
x=zeros(100)	nulová čtvercová matice 100x100
x=ones(1,1000)	řádkový vektor o 1000 prvcích plný hodnot 1
x=eye(100)	jednotková matice o rozměru 100x100
x=eye(25,50)	matice 25x50 s jednotkami na hlavní diagonále
x=rand(5,7)	matice 5x7 naplněná náh. čísly v intervalu (0,1) - rovnoměrné rozložení

x=randn(1,9) matice 1x9 naplněná náhodnými čísly s normálním rozložením N(0,1)

Informace o dalších speciálních typech matic viz **help elmat**. Jako příklad uveďme tzv. magický čtverec[28]

x=magic(6) čtvercová matice - magický čtverec

```
x = 35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
```

3.1.3 Práce s částí matice či vektoru

V příkladech využijeme vektory a matice vytvořené v kap.31.1. Práce s pouze jednou hodnotou vektoru či matice je standardní.

a(2) zjištění hodnoty 2 prvku vektoru a

```
ans =     2
```

c(5)=x(2,3) hodnotu z druhého řádku a třetího sloupce matice x dej do pátého prvku vektoru c

```
c =     1     2     3     4     7
```

V MATLABu lze také pracovat s částí vektoru či submaticí matice. V tomto případě se používá pro určení rozsahu indexů znak dvojtečka. Např. část matice x (submatici) lze získat příkazem

x(2:5,1:2)

```
ans =    3    32
        31     9
         8    28
        30     5
```

Často je potřebné vyjmout celý sloupec či řádek. To je přirozeně možné i předchozím způsobem - pomocí příkazu **size** zjistíme rozměry a tyto použijeme pro určení rozsahu. Existuje ale elegantnější způsob. Použití samotné dvojtečky na místě příslušného indexu získáme celý rozsah.

y=x(3,:) vyjmi třetí řádek, ulož do proměnné y a vypiš

```
y =    31     9     2    22    27    20
```

Další užitečným trikem je použití klíčového slova **end**[29] - v případě, že potřebujeme rozsah indexu od určité hodnoty do konce - např.

---

[28] čtvercová matice, kde součet každého řádku, sloupce i obou diagonál dává stejné číslo

[29] lze používat až od verze 5

```
r=3; s=4; y=x(r:end,s:end)

  y = 22    27    20
      17    10    15
      12    14    16
      13    18    11
```

V tomto příkladu jsme navíc použili tři příkazy na jednom příkazovém řádku.

V případě, že potřebujeme získat hodnoty na hlavní diagonále matice je užitečná funkce **diag**

```
diag(y)

  ans =    22
           10
           16
```

3.1.4 Zápis dat do souboru a čtení ze souboru

V MATLABu existují v podstatě dvě varianty uložení dat do souboru a odpovídajícího čtení ze souboru. První způsob je nejčastěji používaný v případě, že máme něco v MATLABu rozpracováno tj. máme naplněná data v proměnných a musíme ukončit práci. Při prostém ukončení programu nás příště čeká pracné plnění dat znova. Pro tento případ je určený příkaz **save.** Použitý bez dalších parametrů uloží všechny použité proměnné (tzv. pracovní prostor) do souboru matlab.mat v aktuálním adresáři. Obnovit proměnné z tohoto souboru lze příkazem **load**. Použití těchto příkazů s jedním parametrem (jméno souboru bez přípony) uloží/načte proměnné do/z souboru jmeno.mat.

```
save stav170100
```
uložení všech proměnných (včetně názvů) do souboru stav170100.mat

Pozor na to, že formát souboru je binární, speciální pro MATLAB a ukládají se i názvy proměnných. Uvedeme-li za jméno souboru ještě názvy proměnných, uloží se pouze vyjmenované proměnné.

Druhá varianta umožňuje výměnu dat s jinými programy. Ukládá data v textovém (čistém ASCII) formátu. Požadavek na uložení v tomto formátu je rozpoznán na základě uvedení klíčového slova -ASCII v příkazu save. Následující příklad uloží matici x v textovém formátu do souboru pokus.txt v aktuálním adresáři:

```
save pokus.txt x -ascii
```
uložení hodnot z proměnné x do souboru pokus.txt

Příkaz **load** načte tento soubor a uloží do proměnné stejného jména[30] jako je jméno souboru (bez přípony)

```
load pokus.txt
```
načtení dat ze souboru pokus.txt do proměnné pokus

```
pokus
```
výpis obsahu proměnné pokus

```
  pokus =    35     1     6    26    19    24
              3    32     7    21    23    25
             31     9     2    22    27    20
              8    28    33    17    10    15
             30     5    34    12    14    16
              4    36    29    13    18    11
```

[30] nelze pomocí příkazu load načíst ASCII data do proměnné jiného jména než je jméno souboru, ze kterého se data čtou

Data v textovém souboru musí být ve stejně dlouhých sloupcích . Čísla v řádcích by měla být oddělen mezerami, oddělovač desetinné části je tečka. Viz např. obsah souboru pokus.txt

3.5000000e+001	1.0000000e+000	6.0000000e+000	2.6000000e+001	1.9000000e+001	2.4000000e+001
3.0000000e+000	3.2000000e+001	7.0000000e+000	2.1000000e+001	2.3000000e+001	2.5000000e+001
3.1000000e+001	9.0000000e+000	2.0000000e+000	2.2000000e+001	2.7000000e+001	2.0000000e+001
8.0000000e+000	2.8000000e+001	3.3000000e+001	1.7000000e+001	1.0000000e+001	1.5000000e+001
3.0000000e+001	5.0000000e+000	3.4000000e+001	1.2000000e+001	1.4000000e+001	1.6000000e+001
4.0000000e+000	3.6000000e+001	2.9000000e+001	1.3000000e+001	1.8000000e+001	1.1000000e+001

Kromě těchto dvou formátů je možná práce ještě se zvukovými soubory formátu .WAV (MS Windows) a .AU (NexT/SUN) pomocí příkazů **wavread** a **wavwrite** (**auread, auwrite**). Je podporováno více kanálů a osmi i šestnácti bitové vzorkování (od verze 5.2)

3.2 Základní operace a funkce

V dalším výkladu se předpokládá, že čtenář je seznámen se základními pravidly práce s maticemi a maticovými operacemi.

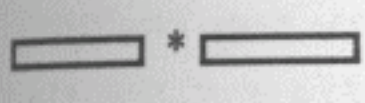
3.2.1 Sčítání, maticové násobení, inverze, operace prvek po prvku

Nyní ukážeme používání matematických operací pro práci s maticemi v MATLABu. Vytvoříme si znovu několik proměnných, na kterých předvedeme základní operace. Pokud dále v této kapitole nejsou v příkladech definovány použité proměnné, jsou myšleny tyto vektory a matice. Vektory budeme označovat[31] malými písmeny a matice velkými.

```
a=1:3;              řádkový vektor 1x3
b=[3:-1:1]';        sloupcový vektor 3x1
A=[a,b';b',a];      matice 2x6
B=A';               matice 6x2
   chybné zadaní matice
C=[1 2 3 4;         první řádek matice (4 řádky)
   5 6 7 8;         druhý řádek matice (4 sloupce)
   9 10 11];        třetí řádek matice (3 sloupce) chyba

Number of elements in each row must be the same.
```

[31] toto je konvence zavedená pouze pro zlepšení orientace v textu této kapitoly

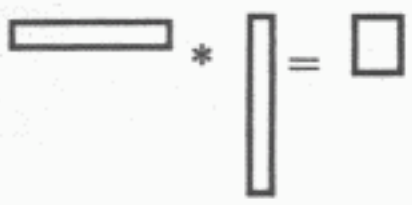
Následující příklad ukazuje, že násobení matic není asociativní a že sčítat se v MATLABu dají pouze odpovídající matice. Prostudujte si pozorně všechny varianty násobení a sčítání dvou vektorů.



chyba

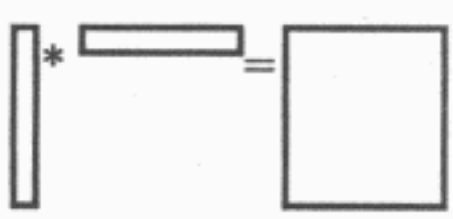
```
a*a
??? Error using
==> *
Inner matrix
dimensions must
agree.
```

```
c=a+a
   c =  2   4   6
```



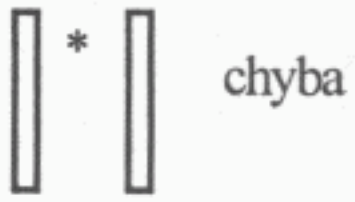
```
c=a*a'
   c = 14
```

```
a+a'
??? Error using
==> +
Matrix
dimensions must
agree.
```



```
C=a'*a
   C =  1   2   3
        2   4   6
        3   6   9
```

```
a'+a
??? Error using ==> +
Matrix dimensions
must agree.
```



chyba

```
a'*a'
??? Error using
==> *
Inner matrix
dimensions must
agree.
```

```
c=a'+a'
   c =  2
        4
        6
```

Obdobně jako při práci se skaláry odpovídá dělení v maticovém počtu operaci násobení převrácenou hodnotou. Tato operace - získání převrácené hodnoty - se nazývá inverzí a lze ji provést pouze se čtvercovou maticí. Stejně jako není definováno dělení nulou, neexistuje inverze k tzv. singulárním maticím. Singulární matice mají determinant rovný nule. Příkladem takové matice je matice C (třetí varianta násobení dvou vektorů z předchozího příkladu). Výpočet determinantu se v MATLABu provede funkcí **det** a výpočet inverzní matice funkcí **inv**. Inverze matice a determinant jsou v MATLABu definovány pro čtvercové matice.

```
1/0                          dělení skalární nulou
   Warning: Divide by zero.
   ans =   Inf
det(C)                       výpočet determinantu matice C
   ans =    0
inv(C)                       výpočet inverzní matice k matici C
   Warning: Matrix is singular to working precision.
   ans =   Inf   Inf   Inf
           Inf   Inf   Inf
           Inf   Inf   Inf
```

Vytvořme nyní matici $D = A * A^T$

```
D=A*A'                       násobení matice a transponované matice dá symetrickou matici
   D =   28   20
         20   28
```

a pro zajímavost spočítejme inverzní matici k matici D

```
inv(D)                       výpočet inverzní matice
   ans =   0.0729  -0.0521
          -0.0521   0.0729 .
```

Pro skaláry platí pro x≠0 rovnost $x*x^{-1}=1$. Ukažme si obdobný vztah pro matici D

```
D*inv(D)                    násobení matice a matice k ní inverzní dá jednotkovou matici
   ans =     1     0
             0     1
```

Operace násobení, jak byla do tohoto místa používána, je klasické maticové násobení. Kromě maticového násobení je v MATLABu definováno násobení člen po členu. Rozdíl mezi oběma případy je ukázán na definici obou násobení

$$A[r,m] \quad B[m,s] \quad C[r,s] \quad C = A*B \quad C(i,j) = \sum_{k=1}^{m} A(i,k)*B(k,j)$$

$$D[r,s] \quad E[r,s] \quad F[r,s] \quad F = D.*E \quad F(i,j) = D(i,j)*E(i,j)$$

```
C=D*D                       maticové násobení
   C =    1184     1120
          1120     1184
F=D.*D                      násobení prvek po prvku
   F =     784      400
           400      784
```

Některé operace na maticích mají smysl pouze prvek po prvku např.

```
A.^0.75
   ans =    1.0000    1.6818    2.2795    2.2795    1.6818    1.0000
            2.2795    1.6818    1.0000    1.0000    1.6818    2.2795
```

Zastavme se u operací prvek po prvku. Výhodou maticového zápisu MATLABu není jen možnost přirozené práce s maticemi, ale i možnost jednou operací zpracovat velké množství čísel. Následující příkaz spočítá všechny hodnoty výrazu odpovídající příslušným hodnotám x a uloží do proměnné y

```
x=1:5;
y=1./(1+x)
   y =    0.5000    0.3333    0.2500    0.2000    0.1667
```

Nyní podrobněji k operátorům / (maticové dělení zprava) a \ (maticové dělení zleva). Maticové dělení zprava je v podstatě ekvivalentní násobení inverzní maticí - A/B=A*inv(B) - s tím rozdílem, že je prováděno numericky jiným způsobem (efektivněji[32] než při použití inverze). Maticové dělení zleva je poněkud složitější. V případě dělení čtvercových matic stejné velikosti jde o obdobný případ jako u maticového dělené zprava tj. A\B=B*inv(A). V případě, že A je čtvercová matice rozměru N×N a b je sloupcový vektor rozměru N, pak výsledkem je sloupcový vektor x, který je řešením rovnice A*x=b získaným Gaussovou eliminací. Nejsložitější případ nastane když matice není čtvercová ale rozměru N×M a vektor b je sloupcový o rozměru N (případ N rovnic o M neznámých). Pokud je N>M (soustava je přeurčená) je výsledek počítán

[32] např. operace A/A trvá cca 78% času operace A*inv(A)

metodou nejmenších čtverců. V případě, že je N<M (soustava nedourčená) je výsledný vektor počítán tak, aby bylo co nejvíce elementů výsledku nenulových.

systém tří rovnic pro tři neznámé

```
A=[1 2 3;2 3 1;3 2 1];  b=[1; 2; 4];
x=A\b; x'
  ans =    1.5833   -0.4167    0.0833
```

přeurčený systém (čtyři rovnice pro tři neznámé)

```
A=[1 2 3;2 3 1;3 2 1;1 1 1];  b=[1; 2; 4; 2];
x=A\b; x'
  ans =    1.6667   -0.5000    0.1667
```

nedourčený systém (dvě rovnice pro tři neznámé)

```
A=[1 2 3;2 3 1];  b=[1; 2];
x=A\b; x'
  ans =         0    0.7143   -0.1429
```

3.2.2 Některé základní funkce

MATLAB disponuje většinou základních matematických funkcí. Specialitou je, že tyto funkce jsou aplikovány na všechny prvky proměnné (vektoru či matice). Např. zajímají-li mne hodnoty funkce sinus na intervalu $\langle 0,2\pi \rangle$ v 1000 bodech, lze toho dosáhnout dvěma příkazy

```
x=0:2*pi/999:2*pi; y=sin(x);
```
v proměnné y je tisíc hodnot funkce sin(x)

Jako argumenty funkce lze použít všechny hodnoty, pro které je daná funkce definovaná

```
x=-2:2; y=sqrt(x)
```
odmocnina (i ze záporných čísel)

```
  y =  0+ 1.4142i   0+ 1.0000i    0    1.0000    1.4142
```

```
log(x)
```
přirozený logaritmus

```
  Warning: Log of zero.
  ans = 0.6931+ 3.1416i   0+ 3.1416i  -Inf      0      0.6931
```

```
A=[1 i;-i 1]; B=exp(A)
```
exponenciální funkce

```
  B =   2.7183              0.5403+ 0.8415i
        0.5403- 0.8415i     2.7183
```

```
acsch(x)
```
inverzní hyperbolický kosekant

```
  Warning: Divide by zero.
  > In d:\matlab50\toolbox\matlab\elfun\acsch.m at line 8
  ans =   -0.4812   -0.8814       NaN    0.8814    0.4812
```

Seznam všech elementárních matematických funkcí použitelných v MATLABu získáme příkazem **help elfun** (kap. 3.4.2), seznam speciálních matematických funkcí příkazem **help specfun** (kap. 2.3) a seznam maticově orientovaných funkcí příkazem **help matfun** (kap. 3.4.3).

3.3 Práce s polynomy a interpolace

Práce s polynomy je běžná v inženýrských výpočtech. MATLAB tuto oblast též podporuje. Využívá vektorové reprezentace polynomů. V MATLABu je zavedena určitá konvence uložení polynomu. Koeficient u nejvyšší mocniny polynomu je uložen v prvním prvku vektoru a ostatní koeficienty u klesajících mocnin polynomu jsou umístěny postupně do dalších prvků. Nesmíme zapomenout zapsat i nulové koeficienty. Přepišme dva následující polynomy p(x) a q(x) do vektorů p a q

$$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11 \qquad q(x) = -x^4 + x^3 - x$$

```
p=[4 0 3.1 -7 0 11];  q=[-1 1 0 -1 0];
```

Pro práci s polynomy je k dispozici několik[33] funkcí, z nichž uvedeme pouze některé:

```
x=[-0.1,0,0.1,0.2];
```

```
y=polyval(p,x)                          vyčíslení polynomu p pro všechny hodnoty x
   y =   10.9269   11.0000   10.9331   10.7461
```

```
r=conv(p,q)                             násobení polynomů p a q
   r = -4.0000 4.0000 -3.1000 6.1000 -7.0000 -14.1000 18.0000 0 -11.0000 0
```

```
r=roots(p)                              kořcny polynomu p
   r =  -0.5124 + 1.4413i
        -0.5124 - 1.4413i
         0.9727 + 0.5747i
         0.9727 - 0.5747i
        -0.9207
```

Nalezení koeficientů polynomu zvoleného stupně prokládajícího data podle kritéria nejmenších čtverců

```
x=[1 2 3 4 5]; y=[5.5 43.1 100.2 190.7 218.4];
```

```
r=polyfit(x,y,3)                        nalezení koeficientů
   r =   -6.8583   62.6964 -110.3452   61.5800
```

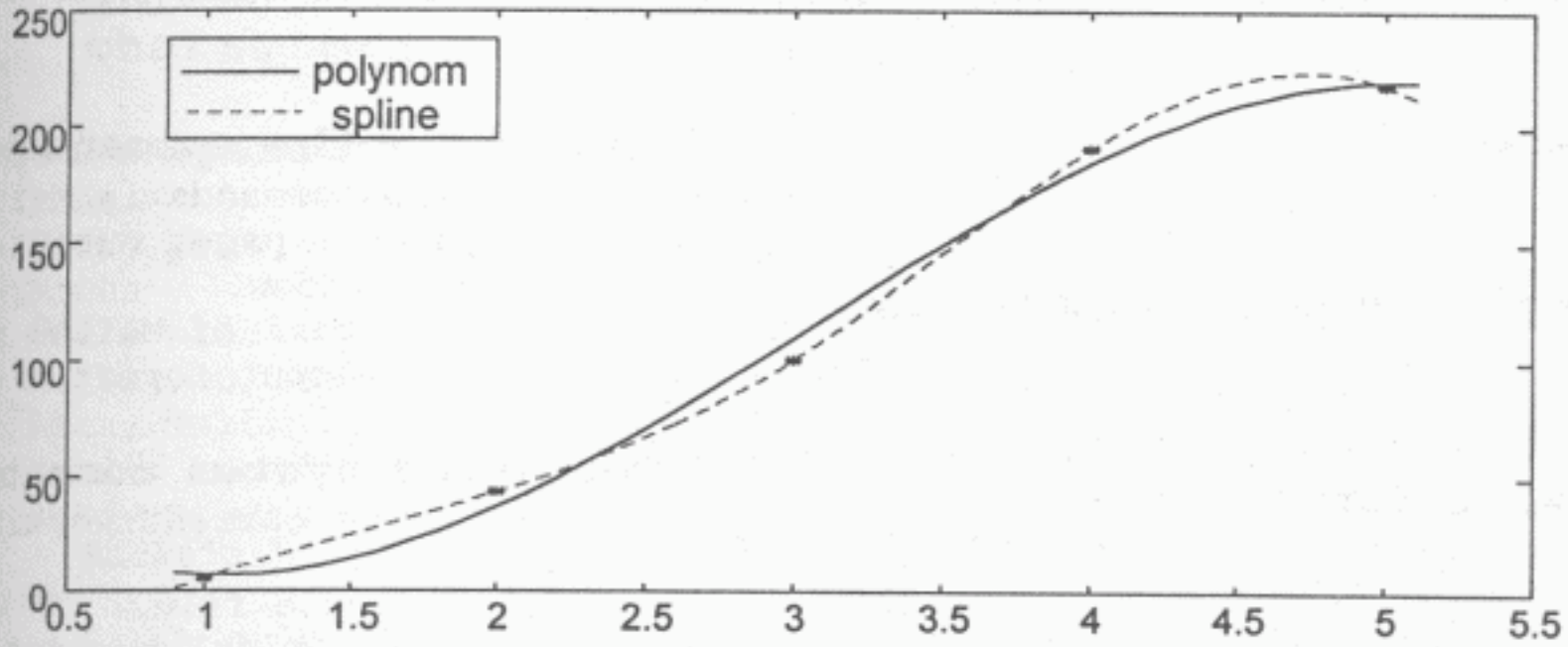
S proložením bodů pomocí polynomu souvisí interpolace. Respektive proložení polynomem je jeden ty interpolace. MATLAB poskytuje několik standardních funkcí pro různé typy interpolačních funkcí. Pop těchto funkcí a jejich použití je přesahuje rozsah tohoto textu.

Pro ukázku použití funkce **polyval** předběhneme další výklad a použijeme příkaz plot pro grafické zobrazen původních bodů a bodů získaných proložením polynomem 3.stupně a interpolační funkcí - kubickými splin (viz obr. 3-1).

```
xx=0.9:0.1:5.1;                         body x, ve kterých budou počítány aproximace
yp=polyval(r,xx);                       body y získané polynomiální interpolací (MNČ)
yi=interp1(x,y,xx,'spline');            body y získané splinovou interpolací
plot(x,y,'*',xx,yp,xx,yi,'--'), legend('polynom','spline')
```

---

[33] seznam funkcí získáme příkazem **help polyfun**



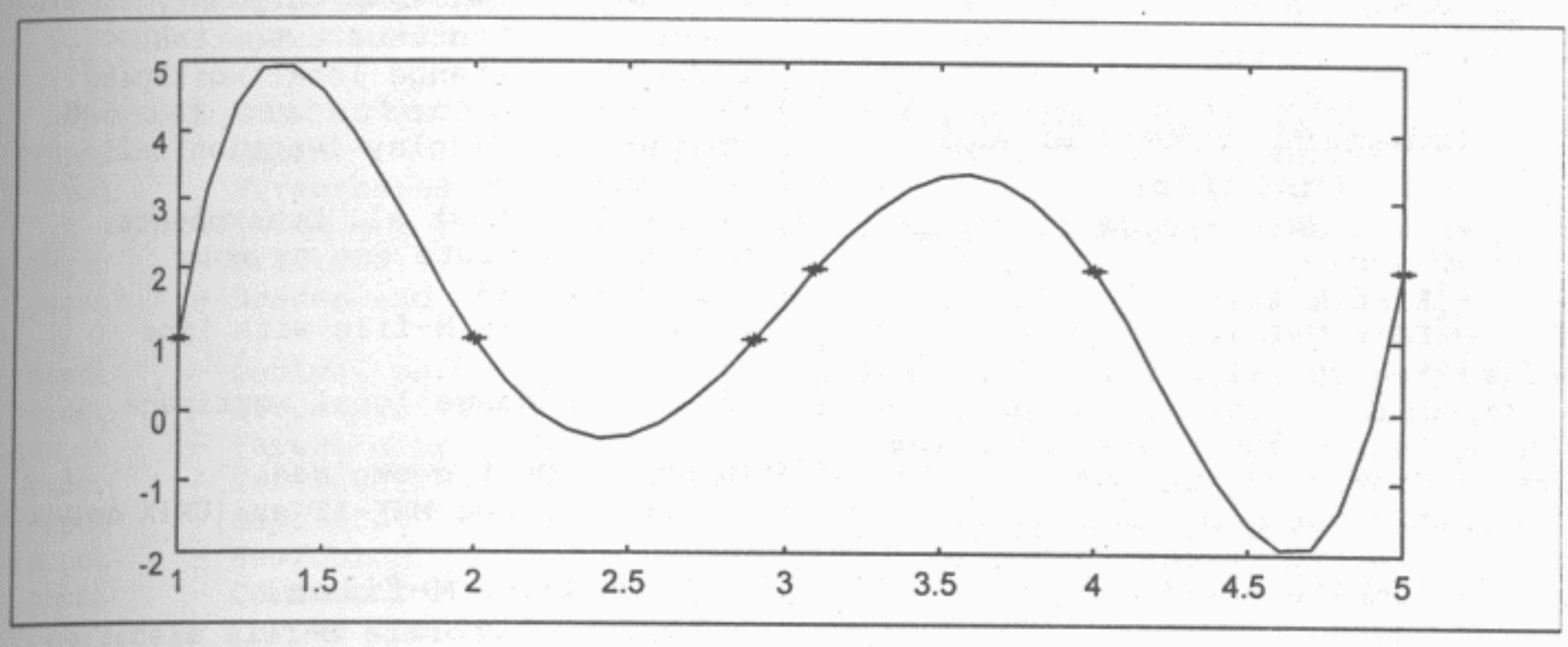
Obrázek 3-1 Aproximace kubickými spline

Ukažme si příklad, který ukazuje zrádnost bezmyšlenkového použití polynomiální aproximace. Mějme následující data (označené křížky na obr. 3-2.)

```
x=[1 2 2.9 3.1 4 5]; y=[1 1 1 2 2 2];
```

Pokud je aproximujeme polynomem 5.stupně, dopočítáme hodnoty pro dané x dostaneme ideální shodu. Pokud se však podíváme i na hodnoty mimo sledované body už spokojeni nebudeme. Aproximace se sice přesně shoduje v zadaných bodech ovšem průběh mezi zadanými body se značně odchyluje od očekávaného průběhu. Polynomiální aproximace není vhodná pro aproximaci průběhů s ostrými zlomy.

```
p=polyfit(x,y,5);  xp=1:0.1:5; yp=polyval(p,xp); plot(x,y,'*',xp,yp)
```



Obrázek 3-2 Polynomiální aproximace

3.4 Seznam funkcí a příkazů pro práci s maticemi

Na závěr třetí kapitoly uvedeme seznam všeobecných příkazů, přípustných operací, příkazů pro práci s daty, funkcí pro práci s maticemi, vytváření speciálních matic, standardních i speciálních matematických funkcí, funkcí pro práci s polynomy

3.4.1 Všeobecné příkazy, operace a práce s daty

Seznam příkazů a funkcí umístěných v adresářích matlab\general, matlab\ops a matlab\iofun

```
help general

                General purpose commands.
  MATLAB Toolbox   Version 5.2 18-Dec-1997

              General information
help      - On-line help, display text
        at command line.
helpwin   - On-line help, separate
        window for navigation.
helpdesk  - Comprehensive hypertext
        documentation and
        troubleshooting.
demo      - Run demonstrations.
ver       - MATLAB, SIMULINK, and
        toolbox version information.
whatsnew  - Display Readme files.
Readme    - What's new in MATLAB 5.1

            Managing the workspace.
who       - List current variables.
whos      - List current variables, long
        form.
clear     - Clear variables and
        functions from memory.
pack      - Consolidate workspace
        memory.
load      - Load workspace variables
        from disk.
save      - Save workspace variables to
        disk.
quit      - Quit MATLAB session.

          Managing commands and
                functions.
what      - List MATLAB-specific files
        in directory.
type      - List M-file.
edit      - Edit M-file.
lookfor   - Search all M-files for
        keyword.
which     - Locate functions and files.
pcode     - Create pre-parsed
        pseudo-code file (P-file).
inmem     - List functions in memory.
mex       - Compile MEX-function.

            Managing the search path
path      - Get/set search path.
addpath   - Add directory to search
        path.
rmpath    - Remove directory from search
        path.
editpath  - Modify search path.

          Controlling the command window.
echo      - Echo commands in M-files.
more      - Control paged output in
        command window.
diary     - Save text of MATLAB session.
format    - Set output format.

            Operating system commands
cd        - Change current working
        directory.
copyfile  - Copy a file.
pwd       - Show (print) current working
        directory.
dir       - List directory.
delete    - Delete file.
getenv    - Get environment variable.
mkdir     - Make directory.
 !        - Execute operating systém
        command (see PUNCT).
dos       - Execute DOS command and
        return result.
unix      - Execute UNIX command and
        return result.
vms       - Execute VMS DCL command and
        return result.
web       - Open Web browser on site or
        files.
computer  - Computer type.

              Debugging M-files.
debug     - List debugging commands.
dbstop    - Set breakpoint.
dbclear   - Remove breakpoint.
dbcont    - Continue execution.
dbdown    - Change local workspace
        context.
dbstack   - Display function call
        stack.
dbstatus  - List all breakpoints.
dbstep    - Execute one or more
        lines.
dbtype    - List M-file with line
        numbers.
dbup      - Change local workspace
        context.
dbquit    - Quit debug mode.
dbmex     - Debug MEX-files (UNIX only).

              Profiling M-files.
profile   - Profile M-file execution
        time.
```

```
help ops
```

Operators and special characters.

Arithmetic operators.

```
plus    - Plus                          +
uplus   - Unary plus                    +
minus   - Minus                         -
uminus  - Unary minus                   -
mtimes  - Matrix multiply               *
times   - Array multiply                .*
mpower  - Matrix power                  ^
power   - Array power                   .^
mldivide - Backslash or left matrix
    divide                              \
mrdivide - Slash or right matrix
    divide                              /
ldivide  - Left array divide            .\
rdivide  - Right array divide           ./
kron - Kronecker tensor product          ron
```

Relational operators.

```
eq      - Equal                         ==
ne      - Not equal                     ~=
lt      - Less than                     <
gt      - Greater than                  >
le      - Less than or equal            <=
ge      - Greater than or equal.        >=
```

Logical operators.

```
and     - Logical AND                   &
or      - Logical OR                    |
not     - Logical NOT                   ~
xor     - Logical EXCLUSIVE OR
any     - True if any element of
    vector is nonzero
all     - True if all elements of
    vector are nonzero
```

Special characters.

```
colon     - Colon                       :
paren     - Parentheses and
    subscripting                        ( )
paren     - Brackets                    [ ]
paren     - Braces and subscripting
                                        { }
punct     - Decimal point               .
punct     - Structure field access      .
punct     - Parent directory            ..
punct     - Continuation                ...
punct     - Separator                   ,
punct     - Semicolon                   ;
punct     - Comment                     %
punct     - Invoke operating systém
    command                             !
punct     - Assignment                  =
punct     - Quote                       '
transpose - Transpose                   .'
ctranspose - Complex conjugate
    transpose                           '
horzcat   - Horizontal concatenation
                                        [,]
vertcat - Vertical concatenation   [;]
subsasgn  - Subscripted assignment
                        ( ),{ },.
subsref   - Subscripted reference
                        ( ),{ },.
subsindex - Subscript index
```

Bitwise operators.

```
bitand - Bit-wise AND.
bitcmp - Complement bits.
bitor  - Bit-wise OR.
bitmax - Maximum floating point
    integer.
bitxor - Bit-wise XOR.
bitset - Set bit.
bitget - Get bit.
bitshift  - Bit-wise shift.
```

Set operators.

```
union     - Set union.
unique    - Set unique.
intersect - Set intersection.
setdiff   - Set difference.
setxor    - Set exclusive-or.
ismember  - True for set member.
```

help iofun

File input/output.

File opening and closing.

```
fopen     - Open file.
fclose    - Close file.
```

Binary file I/O.

```
fread     - Read binary data from
    file.
fwrite    - Write binary data to
    file.
```

Formatted file I/O.

```
fscanf    - Read formatted data from
    file.
fprintf   - Write formatted data to
    file.
fgetl     - Read line from file,
    discard newline character.
fgets     - Read line from file, keep
    newline character.
input     - Prompt for user input.
```

String conversion.

```
sprintf   - Write formatted data to
    string.
sscanf    - Read string under format
    control.
```

File positioning.

```
ferror - Inquire file error status.
feof   - Test for end-of-file.
fseek  - Set file position indicator.
ftell  - Get file position indicator.
frewind- Rewind file.
```

```
  File name handling
matlabroot  - Root directory of
     MATLAB installation.
filesep   - Directory separator for
     this platform.
pathsep   - Path separator for this
     platform.
mexext    - MEX filename extension
     for this platform.
fullfile  - Build full filename from
     parts.
fileparts - Filename parts.
partialpath - Partial pathnames.
tempdir   - Get temporary directory.
tempname  - Get temporary file.

  File import/export functions.
load    - Load workspace from
     MAT-file.
save    - Save workspace to MAT-file.
dlmread   - Read ASCII delimited
     file.
dlmwrite  - Write ASCII delimited
     file.
wk1read   - Read spreadsheet (WK1)
     file.
wk1write  - Write spreadsheet (WK1)
     file.
hdf     - MEX-file interface to the
     HDF library.

  Image file import/export.
imread - Read image from graphics
     file.
imwrite- Write image to graphics
     file.
imfinfo   - Return information about
     graphics file.

  Audio file import/export.
auwrite- Write NeXT/SUN (".au") sound
     file.
auread - Read NeXT/SUN (".au") sound
     file.
wavwrite  - Write Microsoft WAVE
     (".wav") sound file.
wavread   - Read Microsoft WAVE
     (".wav") sound file.

  Command window I/O
clc     - Clear command window.
home    - Send cursor home.
disp    - Display array.
input   - Prompt for user input.
pause   -  Wait  for  user  response.
```

3.4.2 Základní příkazy a funkce pro práci s maticemi

Seznam příkazů a funkcí umístěných v adresářích matlab\elmat a matlab\elfun

```
help elmat

    Elementary matrices and
    matrix manipulation.

  Elementary matrices.
zeros   - Zeros array.
ones    - Ones array.
eye     - Identity matrix.
repmat  - Replicate and tile array.
rand    - Uniformly distributed random
     numbers.
randn   - Normally distributed random
     numbers.
linspace  - Linearly spaced vector.
logspace  - Logarithmically spaced
     vector.
meshgrid  - X and Y arrays for 3-D
     plots.
:       - Regularly spaced vector and
     index into matrix.

  Basic array information.
size    - Size of matrix.
length  - Length of vector.
ndims   - Number of dimensions.
disp    - Display matrix or text.
isempty - True for empty matrix.
isequal - True if arrays are
     identical.
isnumeric - True for numeric arrays.
islogical - True for logical array.
logical   - Convert numeric values to
     logical.

  Matrix manipulation.
reshape - Change size.
diag    - Diagonal matrices and
     diagonals of matrix.
tril    - Extract lower triangular
     part.
triu    - Extract upper triangular
     part.
fliplr  - Flip matrix in left/right
     direction.
flipud  - Flip matrix in up/down
     direction.
flipdim - Flip matrix along specified
     dimension.
rot90   - Rotate matrix 90 degrees.
:       - Regularly spaced vector and
     index into matrix.
find    - Find indices of nonzero
     elements.
end     - Last index.
sub2ind - Linear index from multiple
     subscripts.
ind2sub - Multiple subscripts from l
     near index.
```

Special variables and constants.

```
ans     - Most recent answer.
eps     - Floating point relative
    accuracy.
realmax- Largest positive floating
    point number.
realmin- Smallest positive floating
    point number.
pi      - 3.1415926535897
i, j    - Imaginary unit.
inf     - Infinity.
NaN     - Not-a-Number.
isnan   - True for Not-a-Number.
isinf   - True for infinite elements.
isfinite  - True for finite elements.
flops   - Floating point operation
    count.
why     - Succinct answer.
```

Specialized matrices.

```
compan - Companion matrix.
gallery- Higham test matrices.
hadamard  - Hadamard matrix.
hankel - Hankel matrix.
hilb    - Hilbert matrix.
invhilb- Inverse Hilbert matrix.
magic   - Magic square.
pascal - Pascal matrix.
rosser - Classic symmetric eigenvalue
    test problem.
toeplitz  - Toeplitz matrix.
vander - Vandermonde matrix.
wilkinson - Wilkinson's eigenvalue
    test matrix.
```

help elfun

Elementary math functions.

Trigonometric.

```
sin     - Sine.
sinh    - Hyperbolic sine.
asin    - Inverse sine.
asinh   - Inverse hyperbolic sine.
cos     - Cosine.
cosh    - Hyperbolic cosine.
acos    - Inverse cosine.
acosh   - Inverse hyperbolic cosine.
tan     - Tangent.
tanh    - Hyperbolic tangent.
atan    - Inverse tangent.
atan2   - Four quadrant inverse
    tangent.
atanh   - Inverse hyperbolic tangent.
sec     - Secant.
sech    - Hyperbolic secant.
asec    - Inverse secant.
asech   - Inverse hyperbolic secant.
csc     - Cosecant.
csch    - Hyperbolic cosecant.
acsc    - Inverse cosecant.
acsch   - Inverse hyperbolic cosecant.
cot     - Cotangent.
coth    - Hyperbolic cotangent.
acot    - Inverse cotangent.
acoth   - Inverse hyperbolic
    cotangent.
```

Exponential.

```
exp     - Exponential.
log     - Natural logarithm.
log10   - Common (base 10) logarithm.
log2    - Base 2 logarithm and dissect
    floating point number.
pow2    - Base 2 power and scale
    floating point number.
sqrt    - Square root.
nextpow2  - Next higher power of 2.
```

Complex.

```
abs     - Absolute value.
angle   - Phase angle.
conj    - Complex conjugate.
imag    - Complex imaginary part.
real    - Complex real part.
unwrap  - Unwrap phase angle.
isreal  - True for real array.
cplxpair  - Sort numbers into complex
    conjugate pairs.
```

Rounding and remainder.

```
fix     - Round towards zero.
floor   - Round towards minus
    infinity.
ceil    - Round towards plus infinity.
round   - Round towards nearest
    integer.
mod     - Modulus (signed remainder
    after division).
rem     - Remainder after division.
sign    - Signum.
```

3.4.3 Další příkazy a funkce pro práci s maticemi a polynomy

Seznam příkazů a funkcí umístěných v adresářích matlab\matfun, matlab\datafun, matlab\sparfun a matlab\polyfun

```
help matfun
```

Matrix functions - numerical linear algebra.

Matrix analysis.

```
norm    - Matrix or vector norm.
normest - Estimate the matrix 2-norm.
rank    - Matrix rank.
det     - Determinant.
trace   - Sum of diagonal elements.
null    - Null space.
orth    - Orthogonalization.
rref    - Reduced row echelon form.
subspace - Angle between two
    subspaces.
```

Linear equations.

```
\ and /- Linear equation solution;
    use "help slash".
inv     - Matrix inverse.
cond    - Condition number with
    respect to inversion.
condest- 1-norm condition number
    estimate.
chol    - Cholesky factorization.
cholinc- Incomplete Cholesky factor
    zation.
lu      - LU factorization.
luinc   - Incomplete LU factorization.
qr      - Orthogonal-triangular
    decomposition.
nnls    - Non-negative least-squares.
pinv    - Pseudoinverse.
lscov   - Least squares with known
    covariance.
```

Eigenvalues and singular values.

```
eig  - Eigenvalues and eigenvectors.
svd  - Singular value decomposition.
gsvd - Generalized ingular value d
    composition.
eigs - A few eigenvalues.
svds - A few singular values.
poly - Characteristic polynomial.
polyeig- Polynomial eigenvalue
    problem.
condeig- Condition number with
    respect to eigenvalues.
hess - Hessenberg form.
qz   - QZ factorization for
    generalized eigenvalues.
schur  - Schur decomposition.
```

Matrix functions.

```
expm    - Matrix exponential.
logm    - Matrix logarithm.
sqrtm   - Matrix square root.
funm    - Evaluate general matrix
    function.
```

Factorization utilities

```
qrdelete  - Delete column from QR
    factorization.
qrinsert  - Insert column in QR
    factorization.
rsf2csf   - Real block diagonal form
    to complex diagonal form.
cdf2rdf   - Complex diagonal form to
    real block diagonal form.
balance   - Diagonal scaling to
    improve eigenvalue accuracy.
planerot  - Given's plane rotation.
cholupdate  - rank 1 update to
    Cholesky factorization.
qrupdate  - rank 1 update to QR
    factorization.
```

```
help datafun
```

Data analysis and Fourier transforms.

Basic operations.

```
max     - Largest component.
min     - Smallest component.
mean    - Average or mean value.
median  - Median value.
std     - Standard deviation.
sort    - Sort in ascending order.
sortrows  - Sort rows in ascending
    order.
sum     - Sum of elements.
prod    - Product of elements.
hist    - Histogram.
trapz   - Trapezoidal numerical
    integration.
cumsum - Cumulative sum of elements.
cumprod- Cumulative product of
    elements.
cumtrapz  - Cumulative trapezoidal
    numerical integration.
```

Finite differences.

```
diff    - Difference and approximate
    derivative.
gradient  - Approximate gradient.
del2    - Discrete Laplacian.
```

Correlation.

corrcoef - Correlation coefficients.
cov - Covariance matrix.
subspace - Angle between subspaces.

Filtering and convolution.

filter - One-dimensional digital filter.
filter2 - Two-dimensional digital filter.
conv - Convolution and polynomial multiplication.
conv2 - Two-dimensional convolution.
convn - N-dimensional convolution.
deconv - Deconvolution and polynomial division.

Fourier transforms.

fft - Discrete Fourier transform.
fft2 - Two-dimensional discrete Fourier transform.
fftn - N-dimensional discrete Fourier Transform.
ifft - Inverse discrete Fourier transform.
ifft2 - Two-dimensional inverse discrete Fourier transform.
ifftn - N-dimensional inverse discrete Fourier Transform.
fftshift - Shift DC component to center of spectrum.
ifftshift - Inverse FFTSHIFT.

Sound and audio.

sound - Play vector as sound.
soundsc - Autoscale and play vector as sound.
speak - Convert input string to speech (Macintosh only).
recordsound - Record sound (Macintosh only).
soundcap - Sound capabilities (Macintosh only).
mu2lin - Convert mu-law encoding to linear signal.
lin2mu - Convert linear signal to mu-law encoding.

Audio file inport/export.

auwrite- Write NeXT/SUN (".au") sound file.
auread - Read NeXT/SUN (".au") sound file.
wavwrite - Write Microsoft WAVE (".wav") sound file.
wavread - Read Microsoft WAVE (".wav") sound file.
readsnd - Read SND resources and files (Macintosh only).
writesnd - Write SND resources and files (Macintosh only).

help sparfun

Sparse matrices.

Elementary sparse matrices.

speye - Sparse identity matrix.
sprand - Sparse uniformly distributed random matrix.
sprandn- Sparse normally distributed random matrix.
sprandsym - Sparse random symmetric matrix.
spdiags- Sparse matrix formed from diagonals.

Full to sparse conversion.

sparse - Create sparse matrix.
full - Convert sparse matrix to full matrix.
find - Find indices of nonzero elements.
spconvert - Import from sparse matrix external format.

Working with sparse matrices.

nnz - Number of nonzero matrix elements.
nonzeros - Nonzero matrix elements.
nzmax - Amount of storage allocated for nonzero matrix elements.
spones - Replace nonzero sparse matrix elements with ones.
spalloc - Allocate space for sparse matrix.
issparse - True for sparse matrix.
spfun - Apply function to nonzero matrix elements.
spy - Visualize sparsity pattern.

Reordering algorithms.

colmmd - Column minimum degree permutation.
symmmd - Symmetric minimum degree permutation.
symrcm - Symmetric reverse Cuthill-McKee permutation.
colperm- Column permutation.
randperm - Random permutation.
dmperm - Dulmage-Mendelsohn permutation.

Linear algebra.

eigs - A few eigenvalues.
svds - A few singular values.
luinc - Incomplete LU factorization.
cholinc- Incomplete Cholesky factorization.
normest- Estimate the matrix 2-norm.
condest- 1-norm condition number e timate.
sprank - Structural rank.

Linear Equations (iterative methods).

pcg - Preconditioned Conjugate Gradients Method.
bicg - BiConjugate Gradients Method.
bicgstab - BiConjugate Gradients Stabilized Method.
cgs - Conjugate Gradients Squared Method.
gmres - Generalized Minimum Residual Method.
qmr - Quasi-Minimal Residual Method.

Operations on graphs (trees).

treelayout - Lay out tree or forest.
treeplot - Plot picture of tree.
etree - Elimination tree.
etreeplot - Plot elimination tree.
gplot - Plot graph, as in "graph theory".

Miscellaneous.

symbfact - Symbolic factorization analysis.
spparms - Set parameters for sparse matrix routines.
spaugment - Form least squares augmented system.

help polyfun

Interpolation and polynomials.

Data interpolation.

interp1 - 1-D interpolation (table lookup).
interp1q - Quick 1-D linear interpolation.
interpft - 1-D interpolation using FFT method.
interp2 - 2-D interpolation (table lookup).
interp3 - 3-D interpolation (table lookup).
interpn - N-D interpolation (table lookup).
griddata - Data gridding and surface fitting.

Spline interpolation.

spline - Cubic spline interpolation.
ppval - Evaluate piecewise polynomial.

Geometric analysis.

delaunay - Delaunay triangulation.
dsearch - Search Delaunay triagulation for nearest point.
tsearch - Closest triangle search.
convhull - Convex hull.
voronoi - Voronoi diagram.
inpolygon - True for points inside polygonal region.
rectint - Rectangle intersection area.
polyarea - Area of polygon.

Polynomials.

roots - Find polynomial roots.
poly - Convert roots to polynomial.
polyval - Evaluate polynomial.
polyvalm - Evaluate polynomial with matrix argument.
residue - Partial-fraction expansion (residues).
polyfit - Fit polynomial to data.
polyder - Differentiate polynomial.
conv - Multiply polynomials.
deconv - Divide polynomials.

4. Řízení průběhu výpočtu

Příkazy a funkce: `case, else, end, for, if, otherwise, switch, tic, toc, while`

Problém: programové konstrukce - podmínky a cykly, měření časového intervalu

V první kapitole jsme přirovnávali MATLAB k programovacímu interaktivnímu jazyku BASIC. Tento příměr byl použit nejen pro podobnost interaktivního režimu, kdy každý příkaz je hned proveden, ale i pro možnost použití programových konstrukcí. Je možné používat 4 základní programové konstrukce:

---

if *logický_výraz*
příkaz
end

nebo

if *logický_výraz*
příkaz1
else
příkaz2
end

podmíněný příkaz

---

switch *výraz* — výsledkem výrazu musí být hodnota nebo řetězec
case *hodnota1*
příkaz1
case *hodnota2*
příkaz2
...
otherwise
příkaz
end

přepínač

---

while *výraz*
příkaz
end

cykl s podmínkou na začátku

---

for *index=start:krok:konec*
příkaz
end

cykl s pevným počtem opakování

Úplný seznam příkazů týkajících se této oblasti je uveden v kapitole 6.3.

Hlavní využití těchto programových konstrukcí je asi až při psaní uživatelských m-funkcí, ale i v interaktivním režimu jsou občas užitečné. Zvláště příkaz **for** je často využíván. Především lidé, kteří mají zkušenosti s programováním, tento příkaz často využívají i když se dá mnoho případů vyřešit daleko efektivněji důsledným používáním maticových operací.

Nejjednodušší příklad na použití příkazu cyklu for může být např. naplnění vektoru o 100 prvcích i-tou odmocninou čísla 100, kde i je pořadové číslo prvku vektoru. Vyjádřeno v matematickém zápisu,

$$x(i) = \sqrt[i]{100} \quad i = 1...100$$

a zápis v MATLABu

```
for i=1:100
x(i)=100^(1/i);
end
```

Všimněte si, že jednotlivé příkazy mohou být zapsány na samostatné řádky a vykonávání se započne až po uvedení posledního klíčového slova end. Též je možný zápis do jednoho řádku, který je praktičtější[34].

```
for i=1:100, x(i)=100^(1/i); end
```

Řešení pomocí maticových a vektorových operací je výrazně rychlejší než použití příkazu for. Ukažme si vytvoření čtvercové matice 100x100 takové, že první řádek tvoří čísla od 1 do 100 , druhý řádek je dvojnásobkem prvního, třetí trojnásobkem prvního atd. Vytvořme tuto matici pomocí dvou vnořených cyklů a pomocí maticových operací. Pro zajímavost změříme i čas potřebný pro provedení pomocí funkcí **tic** - start měření a **toc** - konec měření časového intervalu

```
tic; for b=1:100, for a=1:100, X(b,a)=a*b; end; end; toc
   elapsed_time =    12.3600
tic; a=[1:100]; X=a'*a; toc
   elapsed_time =     0.1100
```

Vidíme, že maticové řešení je v tomto případě cca 100× rychlejší.

[34] praktičtější v tomto případě znamená, že je možné klávesou šipka nahoru se k tomuto (jednomu) řádku vrátit a příkaz zopak nebo případně opravit

5. Vizualizace

Příkazy a funkce: `fplot, grid, legend, mesh, meshgrid, plot, plot3, subplot, surf, title, xlabel, ylabel`

Problém: tvorba dvou a třírozměrných grafů, rozdělení oblasti kreslení, popis grafu

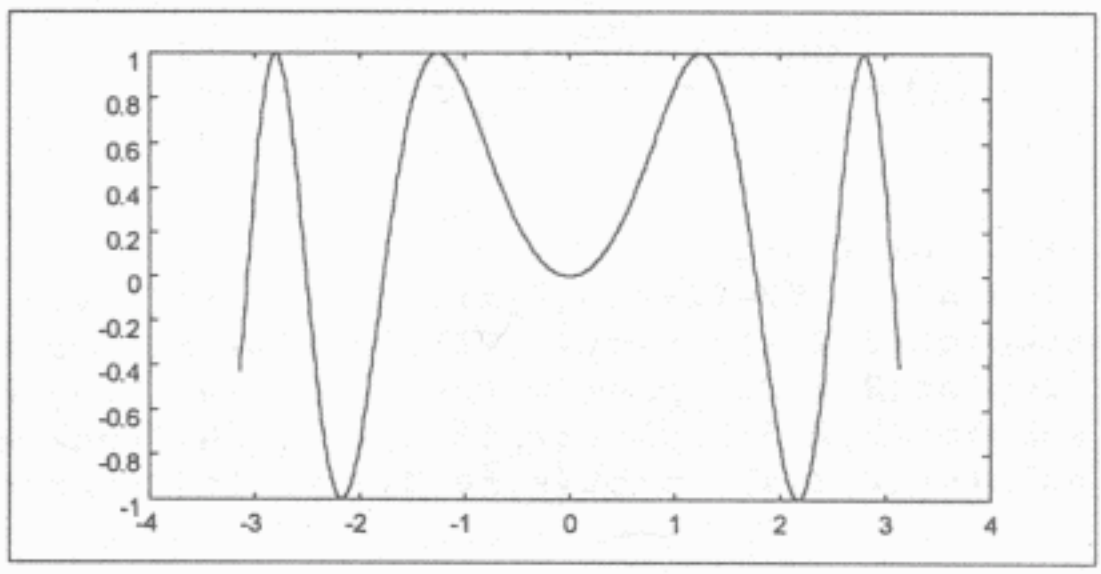
Důležitou vlastností programů typu MATLABu jsou jejich možnosti vizualizace. MATLAB v tomto směru poskytuje značné možnosti jak pro dvourozměrnou tak i třírozměrnou vizualizaci a tvorbu speciálních druhů grafů. Umožňuje též vytváření uživatelských rozhraní. Ukázku možností, co lze vytvořit, nejlépe získáme spuštěním dema (příkazem **demo**). V této kapitole se budeme zabývat pouze základními příkazy pro vizualizaci. Bližší informace o grafických příkazech je možné získat pomocí **help graph2d**, **help graph3d** a **help specgraph**. Další možnosti a seznam dalších příkazů je též v kapitole 10.3.1.

5.1 Dvourozměrné grafy

Základním příkazem pro vykreslení dvourozměrného grafu je příkaz **plot**. Pro úplný popis možností příkazu plot použijte **help plot**. Zde si ukážeme několik příkladů použití.

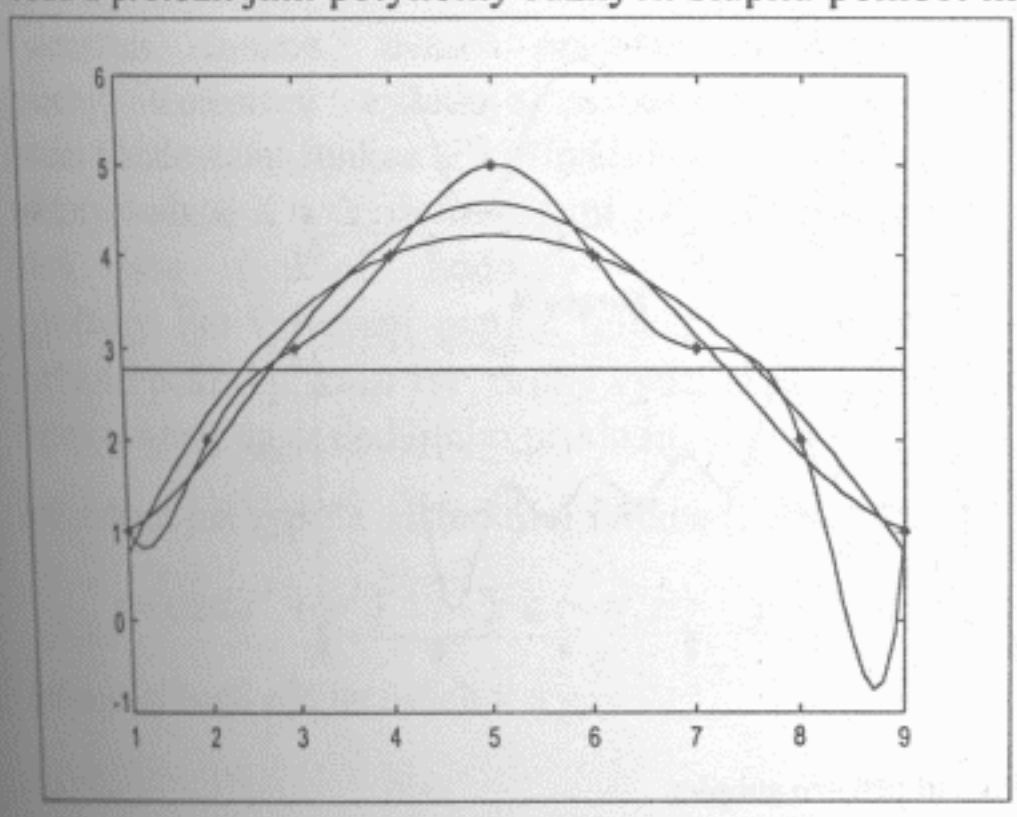
Máme nakreslit průběh funkce y=sin(x^2) na intervalu $x \in \langle -\pi, \pi \rangle$

```
x=-pi:0.01:pi;
y=sin(x.^2);
plot(x,y)
```



Obrázek 5-1 Použití příkazu plot

Chceme-li nakreslit několik průběhů do jednoho grafu, musíme si připravit odpovídající vektory jednotlivých grafů. Zkusme nadefinovat několik bodů a proložit jimi polynomy různých stupňů pomocí nám již známé funkce **polyfit**. Prokládané body jsou dány vektory x a y. V grafu budou vyznačeny hvězdičkami.

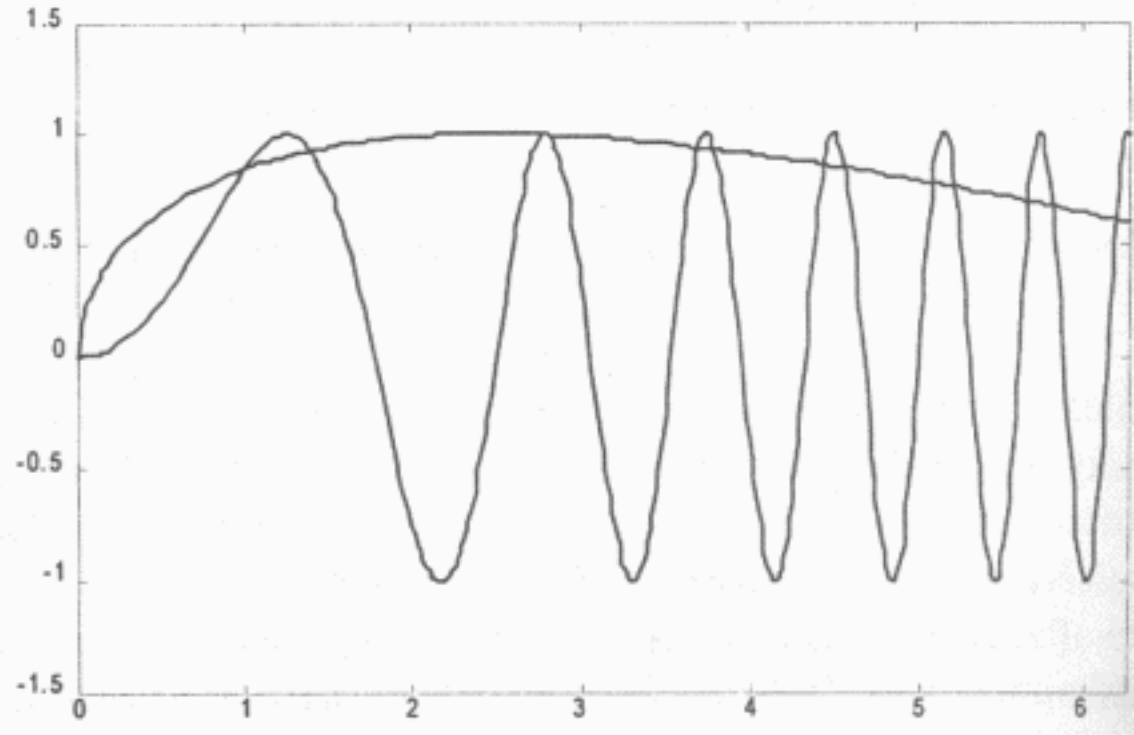


Obrázek 5-2 Více průběhů v jednom grafu

```
x=[1 2 3 4 5 6 7 8 9];
y=[1 2 3 4 5 4 3 2 1];
x1=1:0.1:9;
y1=polyval(polyfit(x,y,1),x1);
y2=polyval(polyfit(x,y,2),x1);
y5=polyval(polyfit(x,y,5),x1);
y9=polyval(polyfit(x,y,9),x1);
plot(x,y,'*',x1,y1,x1,y2,x1,y5,x1,y9)
```

Užitečná může být, pokud chceme vykreslit průběhy funkcí, také funkce **fplot**. Tato funkce je určená přímo na vykreslení průběhu funkce v zadaném rozsahu osy x a případně i y. Nakresleme průběh funkce sin(x^2) a sin(x^0.5) v intervalu $x \in \langle 0, 2\pi \rangle$ (obr. 5.3). Měřítko osy y budiž od -1.5 do 1.5.

```
fplot('[sin(x.^2),sin(sqrt(x))]'
,[0 2*pi -1.5 1.5])
```



Obrázek 5-3 Použití příkazu fplot

V případě, že potřebujeme několik celých grafů umístit do jednoho obrázku, je užitečný příkaz **subplot**. Tento příkaz rozdělí obrázek na několik části a umístí výstup následujícího příkazu plot do vybrané části. Ukažme tento případ na příkladě, kdy chceme umístit do obrázku rozděleného na čtyři části čtyři samostatné grafy opatřené titulkem (obr. 5-4)

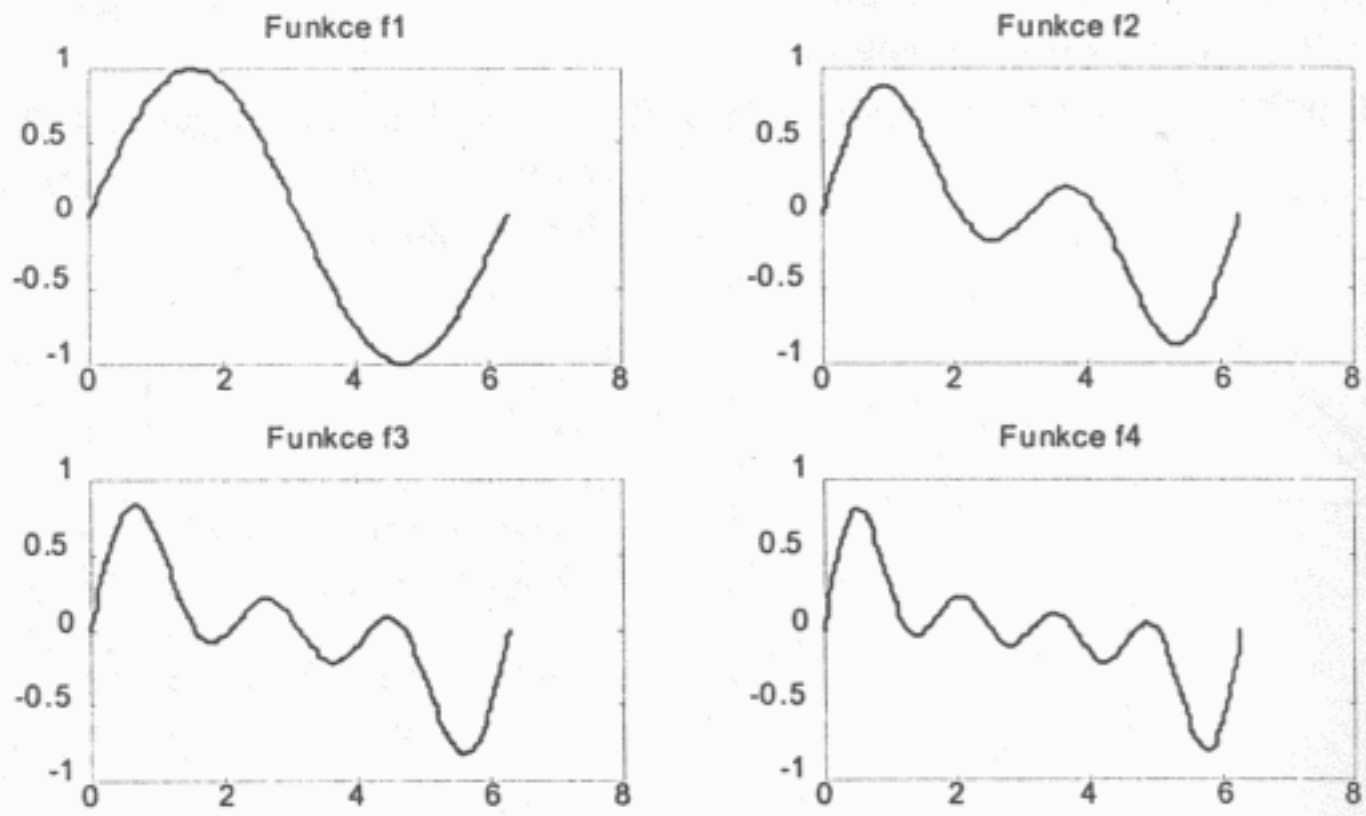
$$f_1(x) = \sin(x) \qquad x \in \langle 0, 2\pi \rangle$$

$$f_2(x) = \frac{\sin(x) + \sin(2x)}{2}$$

$$f_3(x) = \frac{\sin(x) + \sin(2x) + \sin(3x)}{3}$$

$$f_4(x) = \frac{\sin(x) + \sin(2x) + \sin(3x) + \sin(4x)}{4}$$

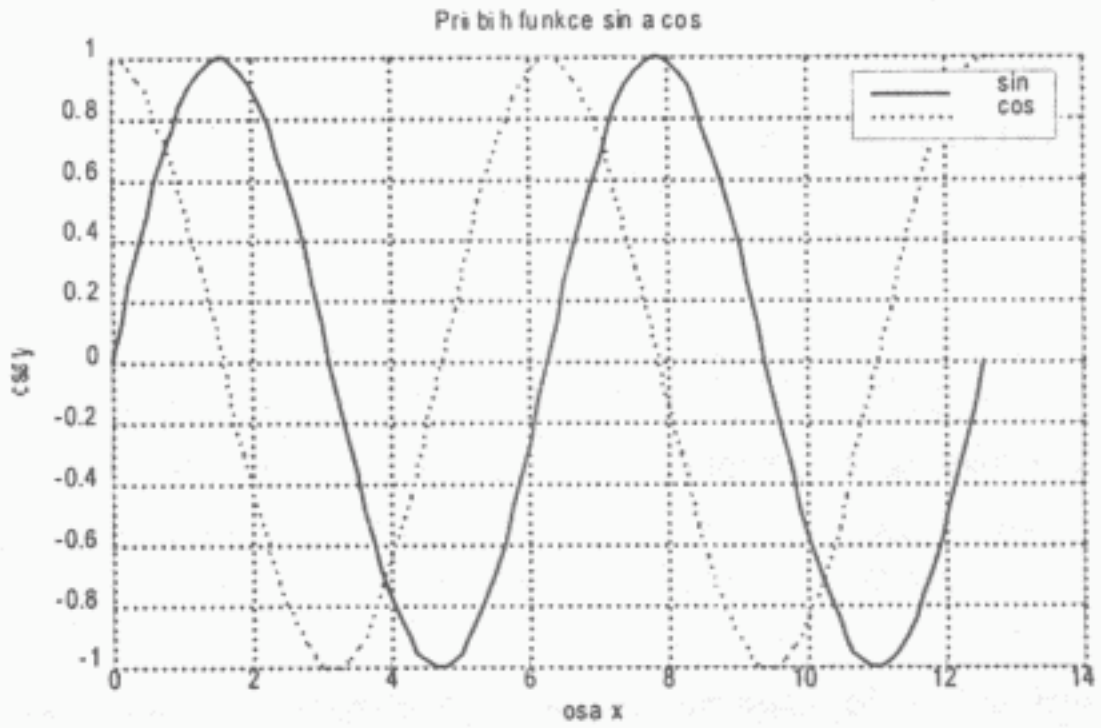
```
x=0:0.01:2*pi;
f1=sin(x); f2=f1+sin(2*x); f3=f2+sin(3*x); f4=f3+sin(4*x);
subplot(2,2,1); plot(x,f1); title('Funkce f1');
subplot(2,2,2); plot(x,f2/2); title('Funkce f2');
subplot(2,2,3); plot(x,f3/3); title('Funkce f3');
subplot(2,2,4); plot(x,f4/4); title('Funkce f4');
```



Obrázek 5-4 Použití příkazu subplot

Popis grafu lze doplnit o popis osy x (**xlabel**), osy y (**ylabel**), případně o popis křivek (**legend**) a mřížka (**grid**) např. jak je uvedeno v následujícím příkladě

```
t=0:4*pi/100:4*pi;
y1=sin(t); y2=cos(t);
plot(t,y1,'-',t,y2,':')
grid
title('Průběh funkce sin a cos')
xlabel('osa x')
ylabel('osa y')
legend('sin','cos')
```



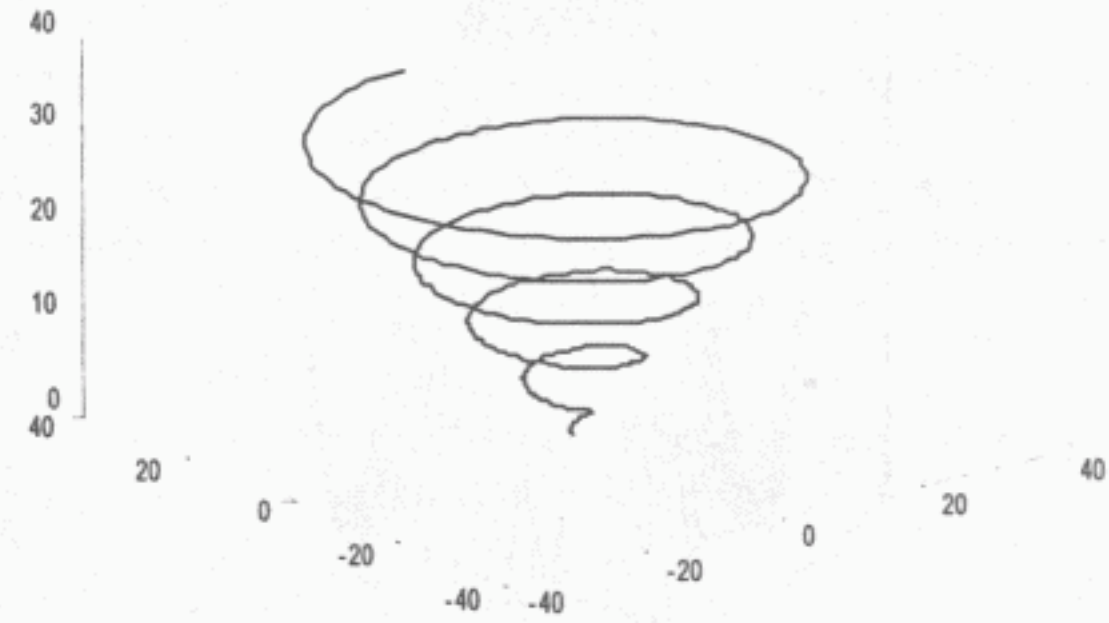
Obrázek 5-5 Popis grafu

5.2 Třírozměrné grafy

Příkazy pro kreslení třírozměrných grafů jsou v základní podobě sice také jednoduché, ale pozornost je potřeba věnovat přípravě dat pro vykreslení grafu. Nejen co se týká vlastních hodnot, ale mít také rozumné požadavky na jejich počet[35].

Grafy v prostoru mohou být buď čárové nebo plošné. Nakreslit čárový graf v prostoru je velmi jednoduché (funkce **plot3**). Stačí mít k dispozici tři vektory stejné délky, které obsahují souřadnice x,y a z (průběh funkce). Např. spirálu na obr. 5-6 nakreslíme příkazy

```
t=0:pi/50:10*pi;
plot3(t.*sin(t),t.*cos(t),t)
```



Obrázek 5-6 Použití příkazu plot3

Vykreslení funkce dvou proměnných (plochy) v prostoru vyžaduje minimálně matici s hodnotami funkce (Z) případně ještě vektory hodnot X a Y definujícími síť, ve které jsou funkční hodnoty matice vypočteny. Pro vytvoření matice funkčních hodnot dvourozměrné funkce lze použít dva postupy. První vychází z použití příkazu **for**, druhý využívá maticových operací a pomocné funkce **meshgrid**. Oba postupy budou ukázány na následujícím příkladu.

Nakresleme jak vypadá následující funkce (s dělením 21 bodů v každé ose)

$$z = \cos(x^2 + y^2) \qquad x \in \langle -\pi, \pi \rangle \quad y \in \langle -\pi, \pi \rangle$$

definování bodů sítě na osách x a y

---

[35] při tvorbě matice definující plochu v prostoru je třeba mít na paměti, že počet bodů narůstá s kvadrátem rozměru matice a že vykreslení plochy složené z např. 10000 bodů (matice 100x100) už může trvat (v závislosti na rychlosti procesoru a grafické karty) historicky významnou dobu.

```
x=[-pi/2:pi/20:pi/2]; y=x;
```

Varianta 1: naplnění matice funkčních hodnot s využitím příkazu for

```
for a=1:length(x),
    for b=1:length(y),
        Z(a,b)=cos(x(a)^2+y(b)^2);
    end
end
```

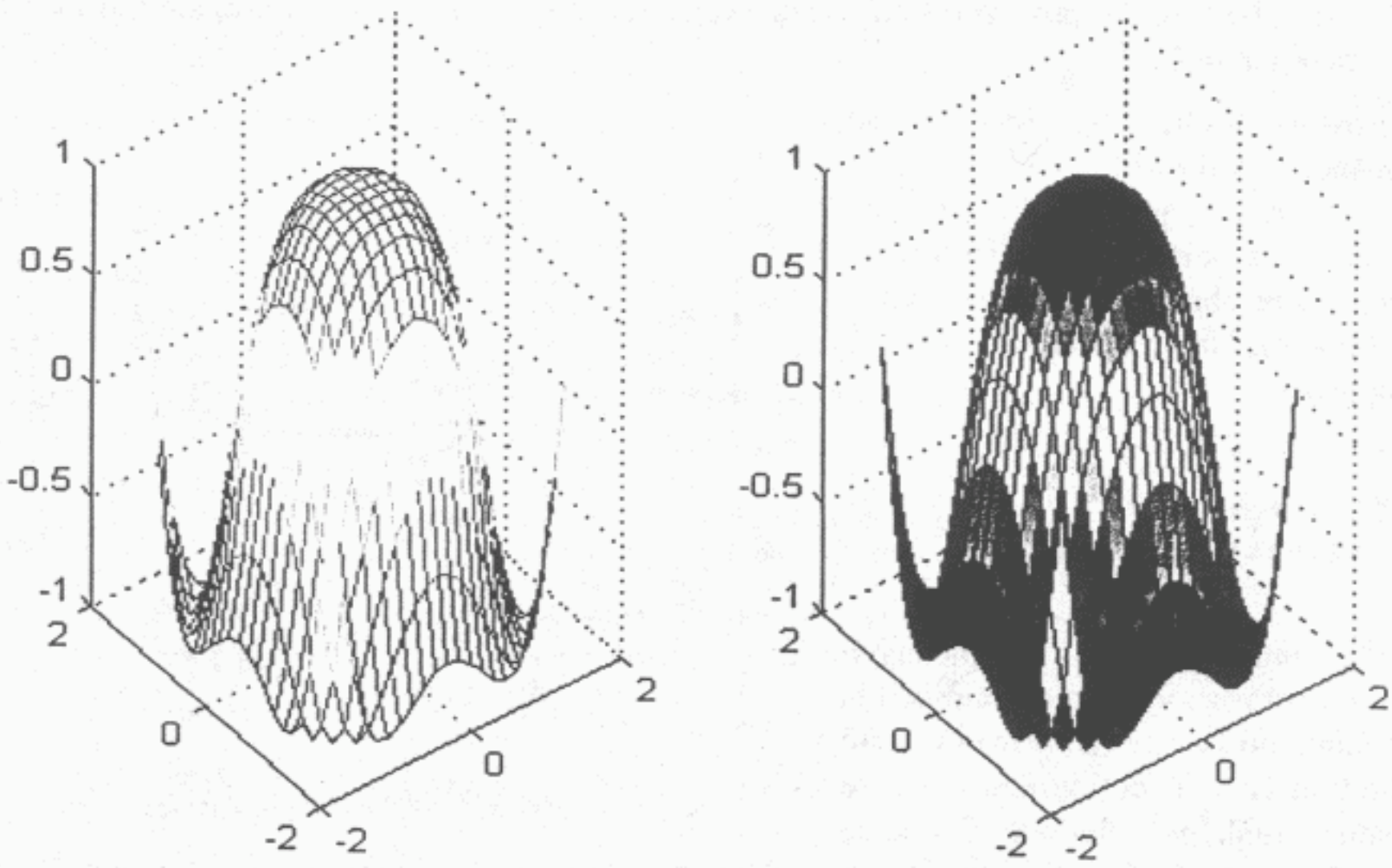
Varianta 2: naplnění matice funkčních hodnot s využitím maticových operací

```
[X,Y]=meshgrid(x,y);
```
pomocné matice

```
Z=cos(X.^2+Y.^2);
```
výpočet funkčních hodnot maticovými operacemi

Vlastní vykreslení prostorového pohledu se provede buď příkazem **mesh** (drátový model) nebo **surf** (stínovaná plocha). Použijeme oba příkazy a obě varianty vykreslíme do jednoho obrázku (obr. 5-7)

```
subplot(1,2,1); mesh(x,y,Z); subplot(1,2,2); surf(x,y,Z)
```



Obrázek 5-7 Použití příkazů mesh a surf

6. Tvorba skriptů a funkcí

Příkazy a funkce: `edit, function, nargin, nargout`

Problém: zápis skriptu a funkce, vstupní a výstupní parametry, komentář

Do tohoto okamžiku jsme s MATLABem pracovali v interaktivním režimu tj. každý příkaz po jeho dokončení byl ihned proveden. Často však potřebujeme určitou posloupnost příkazů opakovat - ať už v rámci jednoho spuštění MATLABu či po příštím spuštění. Pokud tato opakovaná činnost pracuje stále se stejnými daty nebo nad proměnnými určitých jmen je to úkol pro tzv. skript. Pokud potřebuji určitou činnost vykonávat nad různými daty a výsledek uložit do proměnné či proměnných předem neznámých jmen je vhodné napsat vlastní funkci.

Několik velice užitečných funkcí MATLABu vyžaduje definovat zadání ve formě funkce - např. řešení soustavy diferenciálních rovnic, integrál funkce, hledání minima funkce jedné nebo více proměnných atd.

Skripty i funkce se vytvářejí v textové podobě (ASCII) libovolným editorem, který tento výstupní formát umožňuje. Ve Windows může být používán standardní Notepad (Poznámkový blok). Součástí MATLABu verze 5 je vlastní editor/debbuger.

6.1 Skript

Skript je posloupnost příkazů zapsaná do souboru s příponou .m. Napíšeme-li jméno tohoto souboru (bez přípony) do příkazové řádky MATLABu, jsou jednotlivé příkazy postupně vykonány jako bychom je napsali z klávesnice. Proměnné, které byly před spuštěním skriptu definovány, lze ve skriptu použít. Nově vytvořené proměnné v rámci provádění skriptu po jeho skončení zůstanou zachovány.

Využití může být např. pro zpracování souboru naměřených dat. Uveďme si příklad skriptu, který proloží naměřená data uložená v souboru na disketě polynomem 3.stupně a do dalšího souboru uloží hodnoty získané z polynomu v 101 bodech x rovnoměrně rozdělených mezi minimální a maximální hodnotou x. Soubor s původními daty se jmenuje data.dat a obsahuje tři sloupce dat - pořadové číslo měření, hodnoty x a hodnoty y. Soubor s výsledky se bude jmenovat datarx.dat a bude obsahovat dva sloupce dat - hodnoty x a hodnoty y.

Pomocí editoru MATLABu (spuštění příkazem **edit**) nebo textovým editorem se vytvoří soubor, který bude obsahovat následující řádky a uloží pod jménem např. zprdata.m do aktuálního adresáře.

```
load a:\data.dat          %nacteni dat do matice jmenem data
x=data(:,2);              %vyjmuti druheho sloupce dat do promenne x
y=data(:,3);              %vyjmuti tretiho sloupce dat do promenne y
r=polyfit(x,y,3);         %urceni koeficientu prokladajiciho polynomu 3.stupně
xmin=min(x);              %minimalni hodnota x
xmax=max(x);              %maximalni hodnota x
xn=xmin:(xmax-xmin)/100:xmax;   %101 rovnomerne rozlozenych hodnot mezi xmin a xmax
yn=polyval(r,xn);         %vypocet hodnot y podle polynomu
xn=xn(:);yn=yn(:);        %tyto prikazy zajisti, že vysledky budou sloupc. vektory
A=[xn;yn];                %vytvoreni matice 100x2
save a:\datarx.dat A -ascii     %ulozeni matice A do vystupniho souboru
```

Nyní napíšeme-li

```
zprdata
```

příkazy se provedou a soubor s daty a:\data.dat se zpracuje. Pokud by nás nyní zajímalo jak vypadají naměřená data a proložení můžeme rovnou napsat příkazy

```
ya=polyval(r,x);        %vypocet prolozenych hodnot v puvodnich bodech x
plot(x,y,'*',x,ya)      %nakresleni jednotlivych bodu a krivky
```

Vykreslí se správná data, protože použité proměnné zůstaly definovány.

6.2 Funkce

Funkce jsou podstatně efektivnější nástroj k řešení problémů než skripty. Např. toolboxy - rozšíření MATLABu pro řešení určitého okruhu problémů - jsou většinou napsané ve formě uživatelských m-funkcí.

Čím se tedy liší funkce od skriptu, když obojí je tvořeno posloupností třeba stejných příkazů? Zdánlivě nepodstatnou, ale ve svých důsledcích nesmírně důležitou, maličkostí. První řádek souboru tvořícího funkci obsahuje tzv. hlavičku funkce. Tato hlavička má formální tvar

function [out1,out2,...]=*jmeno_funkce*(inp1,inp2,...)

a zajišťuje přenos dat do a z funkce. Vše, co se děje ve funkci, je samostatné. Proměnné vytvořené uvnitř funkce jsou lokální tj. po skončení posledního příkazu funkce zase zaniknou a nemají nic společného s proměnnými v pracovním prostoru MATLABu. Má to tu výhodu, že nemusíme přemýšlet nad názvy proměnných ve funkci, abychom náhodou nesmazali obsah nějaké důležité proměnné se stejným jménem. Jediné spojení směrem dovnitř funkce je zajištěno vstupními parametry a ven z funkce výstupními parametry.

První řádek - hlavička - musí začínat klíčovým slovem function, za kterým následuje seznam výstupních parametrů. Je-li jich více než jeden, je nutné je dát do hranatých závorek a oddělit čárkami. Poté následuje rovnítko a jméno funkce, které by mělo být shodné se jménem souboru[36], ve kterém je funkce uložena. Za jménem funkce je seznam vstupních parametrů uzavřený do kulatých závorek.

Na dalších řádcích se obvykle píše stručný popis (komentář) - co funkce dělá a jaké má parametry. Tyto informační řádky musí začínat znakem % (procento). Text na těchto řádcích se také vypíše, napíšeme-li **help** *jméno_funkce*. Znak % je možné použít kdekoli a označuje komentář. Od znaku % do konce řádku se text při provádění ignoruje.

Poté na následujících řádcích se píší příkazy funkce - tzv. tělo funkce. V příkazech mohu používat jména vstupních parametrů - při vlastním výpočtu se použijí hodnoty, které byly na místě příslušného vstupního parametru při volání funkce zapsány. Mohu používat pomocné proměnné libovolného jména. Uložím-li něco do výstupního parametru, přenese se tato hodnota do proměnné, která byla při volání funkce na místě příslušného výstupního parametru použita.

---

[36] rozhodující je jméno souboru. Používat různá jména v názvu funkce a souboru je sice možné, ale matoucí. Windows 95 a vyšší umožňují používání dlouhých názvů, ale pro použití v MATLABu není vhodné je používat. Název funkce v MATLABu nesmí obsahovat např. mezeru.

Jako příklad uveďme funkci, která určí výkonovou spektrální hustotu signálu vzorkovaného s frekvencí frek a změřená data jsou uložena ve vektoru data.

Funkce v MATLABu jsou značně flexibilní. Je možné vyvolat funkci nejen s různými typy vstupních parametrů (skalár, vektor, matice) ale i s různým počtem vstupních a výstupních parametrů. Uvnitř funkce je možné tuto skutečnost zjistit (funkce **nargin, nargout**) a např. podle přítomnosti výstupních parametrů měnit činnost funkce. Příkladem může být funkce **dstep** z Control Tbx (výpočet přechodové charakteristiky soustavy), která podle počtu vstupních parametrů určuje zda soustava je popsána přenosovou funkcí nebo stavovým modelem a podle přítomnosti výstupních parametrů rozhoduje zda vypočtené hodnoty má uložit do těchto parametrů nebo je pouze nakreslit do grafu.

```
function [f,vyk] = vsh(data,frek)
%
%       [f,vyk] = vsh(data,frek)
%
% Vykonova spektralni hustota signalu vzorkovaneho
% s frekvenci frek. Zmerene vzorky jsou ve
% vektoru data
% f   ...frekvence, na kterych lze zjistit vykon
% vyk ...vykon na odpovidajich frekvencich f
%
% Prevod na nulovou stredni hodnotu =
% odstraneni stejnosmerne slozky

n = length(data);     % pocet bodu mereneho signalu
y = data - sum(data)/n; % odecteni stredni hodnoty
Y = fft(y);           % Fourierova transformace
Pyy = Y.*conj(Y)/n;   % Vykonova spektralni hustota
f = frek/n*(0:n/2);   % Hodnoty realnych frekvenci
vyk = Pyy(1:n/2+1);   % Hodnoty vykonu
```

6.3 Seznam příkazů a funkcí pro tvorbu skriptů a funkcí

Příkazy a funkce využívané při tvorbě skriptů a funkcí jsou v adresáři matlab\lang a jejich seznam lze získat příkazem **help lang**

```
help lang

          Programming language
              constructs.

  Control flow.
if        - Conditionally execute
        statements.
else      - IF statement condition.
elseif    - IF statement condition.
end       - Terminate scope of FOR,
        WHILE, SWITCH, TRY and IF
        statements.
for       - Repeat statements a specific
        number of times.
while     - Repeat statements an
        indefinite number of times.
break     - Terminate execution of WHILE
        or FOR loop.
switch    - Switch among several cases
        based on expression
case      - SWITCH statement case.
otherwise - Default SWITCH statement
        case.

try       - Begin TRY block.
catch     - Begin CATCH block.
return    - Return to invoking function.

  Evaluation and execution.
eval      - Execute string with MATLAB
        expression.
feval     - Execute function specified
        by string.
evalin    - Evaluate expression in
        workspace.
builtin   - Execute built-in function
        from overloaded method.
assignin  - Assign variable in
        workspace.
run       - Run script.
```

Scripts,functions, and variables.

```
script - About MATLAB scripts and
     M-files.
function  - Add new function.
global - Define global variable.
persistent  - Define persistent
     variable.
mfilename - Name of currently
     executing M-file.
lists  - Comma separated lists.
exist  - Check if variables or
     functions are defined.
isglobal  - True for global
       variables.
mlock  - Prevent M-file from being
       cleared.
munlock- Allow M-file to be cleared.
mislocked - True if M-file cannot be
       cleared.
```

Argument handling.

```
nargchk- Validate number of input
     arguments.
nargin - Number of function input
     arguments.
nargout- Number of function output
     arguments.
varargin  - Variable length input
     argument list.
varargout - Variable length output
     argument list.
inputname - Input argument name.
```

Message display.

```
error  - Display error message and
     abort function.
warning- Display warning message.
lasterr- Last error message.
lastwarn  - Last warning message.
errortrap- Skip error during testing.
disp   - Display an array.
fprintf- Display formatted message.
sprintf- Write formatted data to a
     string.
```

Interactive input.

```
input  - Prompt for user input.
keyboard  - Invoke keyboard from
     M-file.
pause  - Wait for user response.
uimenu - Create user interface menu.
uicontrol - Create user interface
     control.
```

7. Funkce funkcí

Příkazy a funkce: `fmins, fzero, ode45, quad, quad8`

Problém: kořen funkce, numerická integrace, minimum funkce více proměnných, řešení soustavy diferenciálních rovnic

Při řešení inženýrských problémů velmi často narazíme na úlohy typu vyhledat minimum funkce či zjistit určitý integrál funkce nebo nalézt numerické řešení diferenciální rovnice.

MATLAB podporuje řešení těchto problémů:

a) nalezení nulové hodnoty funkce jedné proměnné
b) nalezení minima funkce jedné či více proměnných
c) nalezení hodnoty určitého integrálu jedné proměnné[37]
d) řešení soustavy diferenciálních rovnic

Všechny tyto úlohy mají společné to, že příslušné operace je potřeba vykonávat s analyticky zadanými vztahy[38].

Obecný postup přístup MATLABu je v těchto případech ten, že příslušné analytické vztahy se vyjádří m-funkcí MATLABu a použije se příslušná standardní funkce (ať už built-in či m-funkce) MATLABu na řešení daného problému. Tyto standardní funkce mají jako první parametr jméno nově vytvořené m-funkce. Další parametry jsou dány syntaxí příslušné funkce. Podmínkou pro používání těchto standardních funkcí je tedy znalost tvorby m-funkcí.

7.1 Vyhledání nulové hodnoty funkce

Potřebujeme-li nalézt takovou hodnotu nezávisle proměnné nějaké funkce[39], pro kterou je funkční hodnota nula, pak je to úkol pro standardní funkci **fzero**. Upozorňuji, že tato funkce nevyhledá všechny kořeny funkce. Jde o numerické vyhledání nějakého kořene v okolí výchozí hodnoty nezávisle proměnné[40]. Geometrický význam je průsečík funkce s osu x. Nejjednodušší syntaxe funkce fzero je následující

hodnota_korene = **fzero**(´nazev_funkce´,pocatecni_hodnota_x),

kde *nazev_funkce* je jméno m-funkce, která definuje funkční závislost jejíž kořen hledáme. Zadává se buď jméno m-funkce (text) v apostrofech nebo jméno proměnné typu řetězec obsahující název m-funkce. Tato m-funkce musí být napsána tak, aby pro zadanou hodnotu nezávisle proměnné vracela funkční hodnotu. Konstanta či proměnná *pocatecni_hodnota_x* představuje hodnotu nezávisle proměnné, v okolí které se vyhledává kořen. Návratová hodnota funkce **fzero** představuje hodnotu kořene určenou s určitou přesností. Bližší informace viz příkaz **help fzero**.

Ukažme si použití této funkce na řešení tří příkladů. Zároveň ukážeme, že pouze výkonný nástroj nestačí. I u takto jednoduchých příkladů je potřeba vědět co děláme a přemýšlet nad výsledkem. Mějme tyto tři rovnice

$$a)\ x e^{x} = 1 \qquad b)\ x^{2} + x + 1 = 0 \qquad c)\ \sin(x) = {}^{x}\!/_{10}$$

---

[37] v MATLABu verze 5 je i funkce výpočtu dvojného integrálu

[38] řešení je samozřejmě numerické. Pro analytické řešení (existuje-li) je nutný Symbolic Toolbox

[39] speciálně jde-li o polynom, pak se všechny kořeny získají pomocí funkce **roots**

[40] pouze v oboru reálných čísel

```
function y=fce_a(x)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% y=x*exp(x/10)-1
%                    y=fce_a(x)
%

y=x.*exp(x/10)-1;       %nasobeni prvek po prvku

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

function y=fce_b(x)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% y=x^2+x+1
%                    y=fce_b(x)
%

y=x.^2+x+1;             %nasobeni prvek po prvku

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

function y=fce_c(x)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% y=sin(x)-x/10
%                    y=fce_c(x)
%

y=sin(x)-x/10;
```

Všechny rovnice nejprve přepíšeme do tvaru f(x)=0 a tento tvar zapíšeme jako m-funkce MATLABu pod názvy fce_a, fce_b a fce_c. Tyto m-funkce mohou být zapsány třeba tak, jak je uvedeno v rámečku. Nyní můžeme již úlohy řešit. Příkaz

```
koren_a=fzero('fce_a',0)
   koren_a =    0.9128
```

nalezne kořen funkce ad a). Zkusme nyní stejným způsobem nalézt kořen funkce ad b)

```
koren_b=fzero('fce_b',0)
  NaN or Inf function value encountered during search for an interval containing a sign change. Function value at -1.716199e+154 is Inf. Aborting since no such interval was found. Check function or try again with a different starting value.
koren_b =   NaN
```

V tomto případě dostaneme poněkud neočekávaný výsledek. Co se stalo ? Chtěli jsme po MATLABu, aby řešil příklad, který v rámci předpokladů nemá řešení. V oboru reálných čísel řešení funkce ad b) neexistuje - jde o kvadratickou rovnici s dvojicí komplexně sdružených kořenů. O tomto se lehce přesvědčíme např. pomocí funkce **roots**

```
roots([1 1 1])
   ans =  -0.5000+ 0.8660i
          -0.5000- 0.8660i
```

vyhledání kořenů polynomu x^2+x+1

Třetí příklad, funkce ad c), je, pokud nad ním nepřemýšlíme, nejzrádnější. Použijme opět známý příkaz

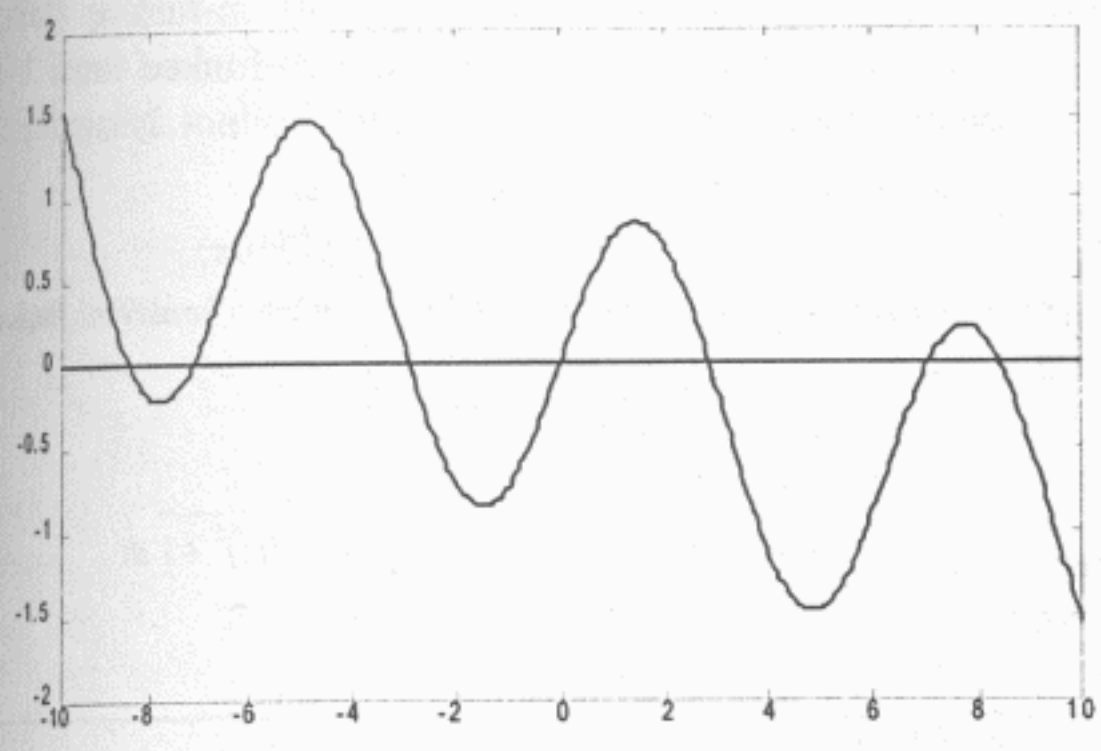
```
koren_c=fzero('fce_c',0)
   koren_c =     0
```

a dostaneme správný výsledek. Avšak pokud použijeme např. poněkud modifikovaný příkaz

```
koren_c=fzero('fce_c',10)
   koren_c =    8.4232
```

dostaneme výsledek sice také správný, ale zcela jiný. Zde je potřeba mít na mysli, že nelineární funkce může mít větší počet kořenů a ne vždy je jednoduché určit kolik jich je. Pro určení odhadu počtu a polohy kořenů nám může pomoci MATLAB. Stačí určit interval, kde se kořeny mohou vyskytovat a na tomto intervalu si nechat průběh funkce vykreslit. Z grafu určíme počet a odhadneme polohu kořenů. Změnou měřítka můžeme touto metodou určit polohu vybraného kořene s požadovanou přesností. Pro případ funkce c je oblast, kde se mohou vyskytovat kořeny dána intervalem (-10,10) (rozmyslete si proč). Vykresleme průběh funkce c na tomto intervalu (obr.7-1).

```
x=-10:0.05:10; plot(x,fce_c(x),x,zeros(size(x)))
```



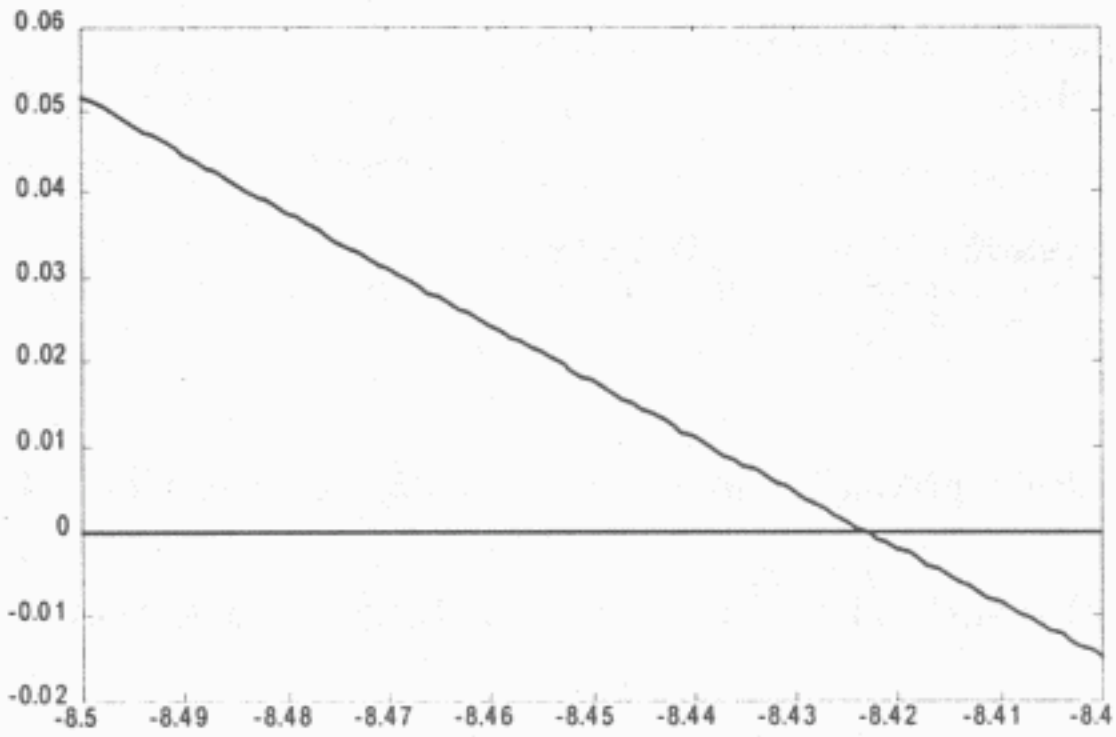
Obrázek 7-1 Grafický odhad kořene funkce

Vidíme, že funkce má 7 kořenů. Chceme-li určit graficky odhad polohy kořene s nejmenší hodnotou tj. kořene okolo hodnoty -8.5 můžeme toho docílit např. příkazem (viz obr.7-2)

```
x=-8.5:0.001:-8.4;
plot(x,fce_c(x),x,zeros(size(x))
)
```

Přesné hodnoty všech 7 kořenů získáme např. sadou příkazů

```
x1=fzero('fce_c',-10)
   x1 =    -8.4232
x2=fzero('fce_c',-6)
   x2 =    -7.0682
x3=fzero('fce_c',-4)
   x3 =    -2.8523
x4=fzero('fce_c',0)
   x4 =     0
x5=fzero('fce_c',3)
   x5 =     2.8523
x6=fzero('fce_c',7)
   x6 =     7.0682
x7=fzero('fce_c',10)
   x7 =     8.4232
```



Obrázek 7-2 Grafický odhad kořene funkce

7.2 Numerická integrace

MATLAB podporuje numerickou integraci funkcí dvěmi standardními funkcemi **quad** a **quad8**. Funkce se liší použitým algoritmem numerické integrace. První z nich používá adaptivní Simpsonovo pravidlo a druhá adaptivní Newton-Cotesovo pravidlo. Základní syntaxe funkce quad (quad8) je následující

hodnota_integralu = **quad**(´nazev_funkce´,dolni_mez,horni_mez)

kde *nazev_funkce* je jméno m-funkce popisující integrand. Zadává se buď jméno m-funkce (text) v apostrofech nebo jméno proměnné typu řetězec obsahující název m-funkce. Tato m-funkce musí být napsána tak, aby pro zadaný vektor hodnot nezávisle proměnné vracela vektor funkčních hodnot. Následující dva parametry představují integrační meze.

Numerickou integraci funkce si ukážeme na dvou příkladech. Mějme křivku v prostoru (spirálu použitou v kapitole 5.2 pro dokumentaci třírozměrných grafů) a máme určit její délku. Přesné zadání je uvedeno vpravo.

parametricky zadaná prostorová funkce

$x = t\sin(t) \quad y = t\cos(t) \quad z = t \qquad t \in \langle 0,10\pi \rangle$

délka obecné parametricky zadané funkce a konkrétní funkce

$$l = \int_{t_1}^{t_2} \sqrt{\left(dx/dt\right)^2 + \left(dy/dt\right)^2 + \left(dz/dt\right)^2}\, dt$$

$$l = \int_{0}^{10\pi} \sqrt{\left(\sin(t) + t\cos(t)\right)^2 + \left(\cos(t) - t\sin(t)\right)^2 + 1}\, dt$$

Pro řešení je potřeba opět tuto funkci, jejíž integrál hledáme, zapsat ve formě m-funkce např. s názvem fce_i1 (viz rámeček). Vlastní výpočet integrálu realizujeme příkazem

```
function y=fce_i1(x)
% uzivatelska funkce pro 'Uvod do ...'
%y= sqrt((sin(x)+x*cos(x))^2+(cos(x)-x*sin(x))^2+1)
%
%                    y=fce_i1(x)
%

y=sqrt((sin(x)+x.*cos(x)).^2+(cos(x)-x.*sin(x)).^2+1);
```

```
l=quad('fce_i1',0,10*pi)
   l =   497.7746
```

případně s využitím jiného algoritmu.

```
l1=quad8('fce_i1',0,10*pi)
   l1 =   497.7724
```

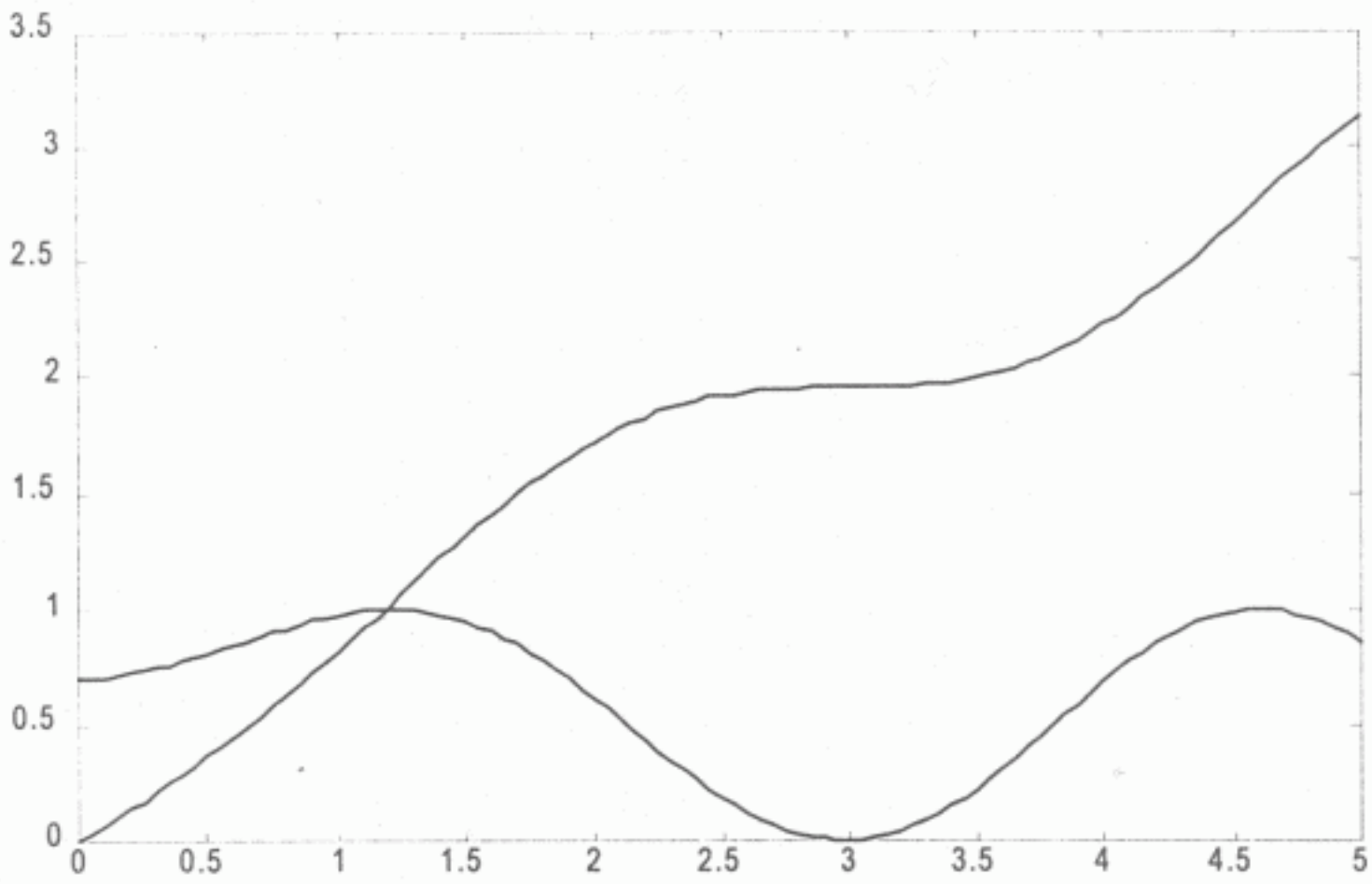
Pro druhý příklad zvolíme trochu složitější případ. Mějme funkci y(x) a bude nás zajímat na daném intervalu průběh funkce y(x) a integrálu funkce.

$$y(x) = \sin^2(\sqrt{1+x^2}) \quad iy(x) = \int_0^x y(t)dt \qquad x \in \langle 0,5 \rangle$$

M-funkce fce_i2 může vypadat např. jak je uvedeno v dalším rámečku. Vlastní příkazy pro získání grafu obou funkcí jsou nyní trochu složitější.

```
function y=fce_i2(x)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% y=(sin(sqrt(1+x^2))^2
%
%                  y=fce_i2(x)
%

y=(sin(sqrt(1+x.^2))).^2;
```

```
x=0:0.05:5;
y=fce_i2(x);
for a=1:length(x),
 iy(a)=quad8('fce_i2',0,x(a));
end
plot(x,y,x,iy)
```



Obrázek 7-3 Integrál jako funkce horní meze

7.3 Minimum funkce více proměnných

Jako velice mocná funkce se ukazuje být funkce vyhledání minima[41] funkce více proměnných. Samozřejmě zase není schopná řešit problém globálního minima ani formulovat řešení ze slovního zadání. Avšak má nezastupitelný význam pro nelineární regresi - např. prokládání naměřených dat funkcí nelineární v parametrech.

Základní syntaxe funkce **fmins** je následující

vekt_x = **fmins**(´nazev_funkce´,vekt_x0)

kde *nazev_funkce* je jméno m-funkce popisující funkční závislost jejíž minimum hledáme. Zadává se buď jméno m-funkce (text) v apostrofech nebo jméno proměnné typu řetězec obsahující název m-funkce. Tato m-funkce musí být napsána tak, aby pro zadaný vektor hodnot nezávisle proměnných vracela funkční hodnotu. Parametr *vekt_x0* je startovací bod v okolí kterého se vyhledává lokální minimum. Vracený vektor *vekt_x* obsahuje souřadnice vyhledaného lokálního minima. Bližší informace viz příkaz **help fmins**.

[41] maximum funkce se jednoduše hledá jako minimum záporně vzaté původní funkce

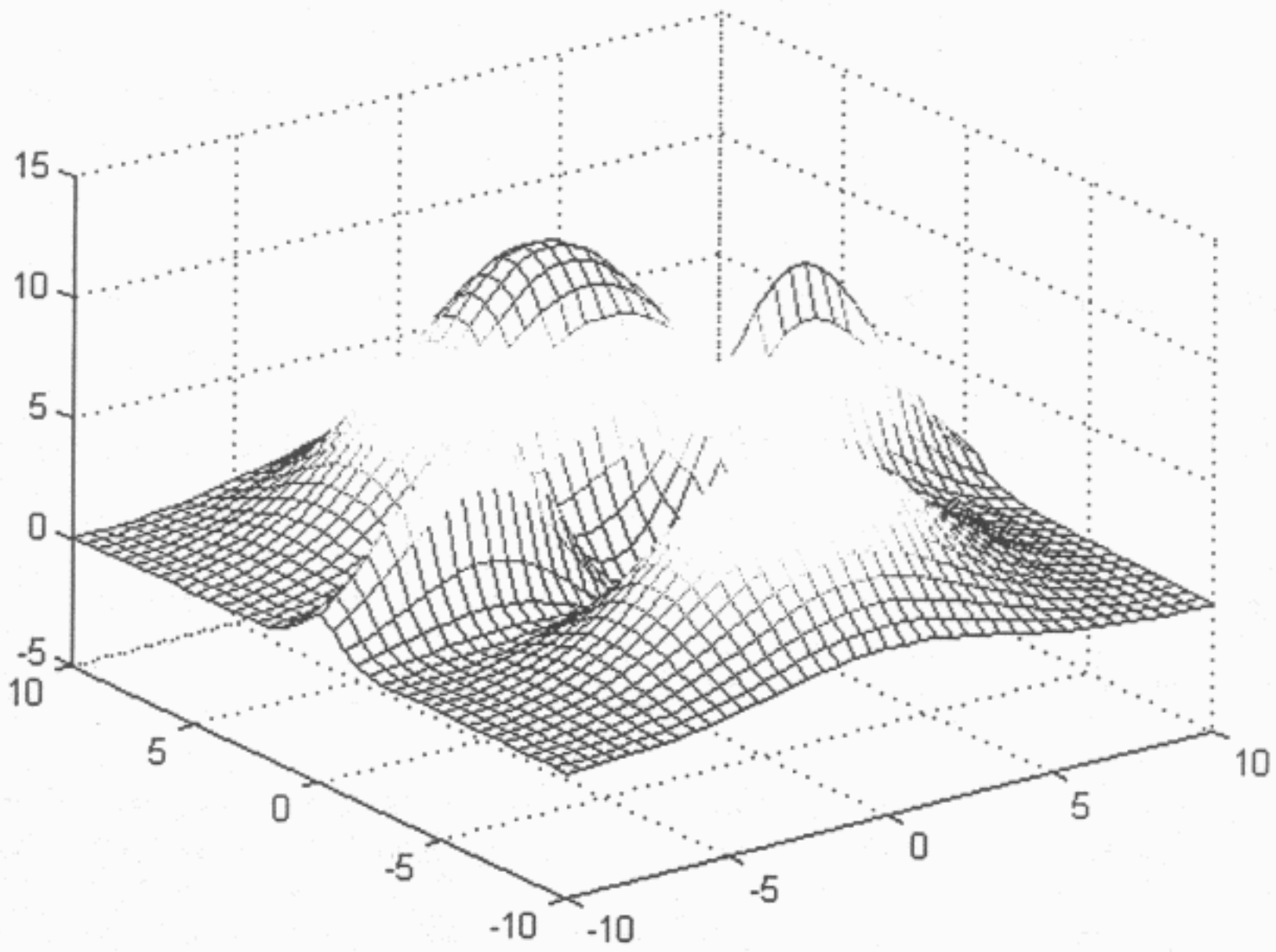
Ukažme si opět dva příklady na použití této funkce. Nejprve řešme standardní úlohu - nalezení minima funkce dvou proměnných. Dvě proměnné jsou zvoleny proto, aby bylo možné příslušnou plochu vykreslit a získat představu o poloze minim. Mějme tedy funkci definovanou jako

$$z(x,y)=\left(x^2+x+y+\frac{y^2}{1+x^2}\right)e^{-\frac{x^2+y^2}{25}} \qquad x\in\langle -10,10\rangle \quad y\in\langle -10,10\rangle$$

Tuto funkci přepíšeme do tvaru m-funkce s názvem fce_m1 např. jak je uvedeno v rámečku. Jak tato funkce vypadá (obr. 7-4) zjistíme po vykreslení např. následujícími příkazy

```
x=-10:0.5:10;y=x;n=length(x);
for a=1:n,
  for b=1:n,
    Z(a,b)=fce_m1([x(a),y(b)]);
  end
end
mesh(x,y,Z)
```

```
function z=fce_m1(par)
%uzivatel.fce pro 'Uvod do pouzivani MATLAB'
%z=(x^2+x+y+y^2/(1+x^2))*exp(-(x^2+y^2)/25)
%
%           z=fce_m1(par)
%

x=par(1); y=par(2);
z=(x.^2+x+y+y.^2./(1+x.^2)).*exp(-
(x.^2+y.^2)/25);
```



Obrázek 7-4 Maximum funkce více proměnných

Zdá se, že minimum plochy je někde v okolí bodu [0,0]. Zkusme vyhledat přesnou polohu a hodnotu minima příkazy

```
souradnice_minima=fmins('fce_m1',[0,0])
hodnota_minima=fce_m1(souradnice_minima)
  souradnice_minima =    -0.6338    -0.6787
  hodnota_minima =       -0.5624
```

Pokud by nás zajímala maxima, pak bychom museli otočit znaménko u výpočtu proměnné z v m-funkci fce_m1 (vznikne nová m-funkce - např. fce_m2) a zkusit různé startovací hodnoty, abychom vyhledali všechny extrémy funkce

```
souradnice_maxima_1=fmins('fce_m2',[0,1])
hodnota_maxima_1=fce_m2(souradnice_maxima_1)
  souradnice_maxima_1 =     0.0218     4.7767
  hodnota_maxima_1 =      -11.0818                    lokální extrém
souradnice_maxima_2=fmins('fce_m2',[0,-1])
hodnota_maxima_2=fce_m2(souradnice_maxima_2)
  souradnice_maxima_2 =     0.0180    -5.2819
  hodnota_maxima_2 =       -7.4123                    lokální extrém
souradnice_maxima_3=fmins('fce_m2',[1,0])
hodnota_maxima_3=fce_m2(souradnice_maxima_3)
  souradnice_maxima_3 =     4.7345     0.4706
  hodnota_maxima_3 =      -11.1722                    absolutní extrém

souradnice_maxima_4=fmins('fce_m2',[-1,0])
hodnota_maxima_4=fce_m2(souradnice_maxima_4)
  souradnice_maxima_4 =    -5.2177     0.5758
  hodnota_maxima_4 =       -7.5042                    lokální extrém
```

Jako druhý příklad na hledání minima funkce si ukážeme prokládání daných bodů libovolnou funkcí. V případě, že je funkce lineární v parametrech jde o relativně jednoduchý problém lineární regrese vedoucí na soustavu lineárních rovnic. V případě, že funkce lineární v parametrech není, jde o problém nelineární regrese vedoucí na optimalizační úlohy.

Pro jednoduchost vygenerujeme "naměřené" body analytickou funkcí a jako kritérium shody původních bodů a bodů získaných aproximační funkcí použijeme kritérium součtu absolutních hodnot odchylek. Zadání "generující funkce" a tvaru aproximační funkce následuje

původní funkce: $y = \sin(x) \quad x \in \langle 0, \pi \rangle$

aproximační funkce: $y = \dfrac{a + bx + cx^2}{1 + dx^e}$

Nejprve je nutné vytvořit m-funkci, která bude mít minimum pro takovou sadu parametrů a-e, která splňuje naše kriterium. Navíc tato m-funkce musí splňovat formální požadavky na tvar funkce použitelný pro standardní funkci **fmins**. Tj. první návratový parametr musí být skalár a jeho hodnota je minimalizována měněním hodnot vektoru parametrů, který musí být uveden jako první vstupní parametr m-funkce. Naším požadavkem je zjištění součtu odchylek od "naměřených" hodnot. Proto musí v m-funkci být k dispozici vektor naměřených hodnot. Uvedeme ho jako druhý vstupní parametr. Pro vykreslení průběhu aproximující funkce budeme potřebovat vektor těchto aproximujících hodnot. Tento vektor je také potřeba pro určení maximální odchylky - tudíž uvnitř funkce se musí počítat. Stačí ho pouze "zveřejnit" mimo funkci prostřednictvím druhého výstupního parametru. Potom m-funkce (s jménem fce_m3) splňující tyto požadavky může vypadat např. jak je uvedeno v rámečku

```
function [krit,ya]=fce_m3(par,x,y)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% aproximace bodu [x,y] funkci ya(x)
% ya=(a+b*x+c*x^2)/(1+d*x^e)
%
%        [krit,ya]=fce_m3(par,x,y)
%

a=par(1);    b=par(2);    c=par(3);    d=par(4);
e=par(5);

% prubeh aprox.fce pro aktualni sadu parametru
ya=(a+b*x+c*x.^2)./(1+d*x.^e);
% vypocet kriteria shody (minimalizuje se)
krit=sum(abs(ya-y));
```

Nyní můžeme vygenerovat zkušební data

```
x=0:0.01:pi; y=sin(x);
```

dále je vhodné připravit si počáteční vektor parametrů s "rozumnými" hodnotami a odzkoušet zda uživatelská funkce fce_m3 dává smysluplný výsledek

```
p0=[1,0,0,0,0]; [kr,ya]=fce_m3(p0,x,y); kr
   kr =  115.0010
```

Protože naše uživatelská funkce obsahuje více než jeden vstupní parametr, musíme informovat funkci **fmins**, že je potřeba dodat další parametry pro výpočet fce_m3. Po prostudování helpu k funkci **fmins** zjistíme, že to lze zajistit uvedením dalších parametrů ve funkci **fmins.** Třetí parametr představuje vektor parametrů algoritmu - implicitní hodnoty jsou použity při zadání prázdného vektoru. Přítomnost čtvrtého parametru (opět prázdný vektor) indikuje, že následují parametry, které je potřeba přenést do uživatelské funkce.

```
p=fmins('fce_m3',p0,[],[],x,y),
[kr,ya]=fce_m3(p,x,y); kr
   p  =    0.1584    1.4214   -0.4802    0.3127   -0.8079
   kr =    3.7473
```

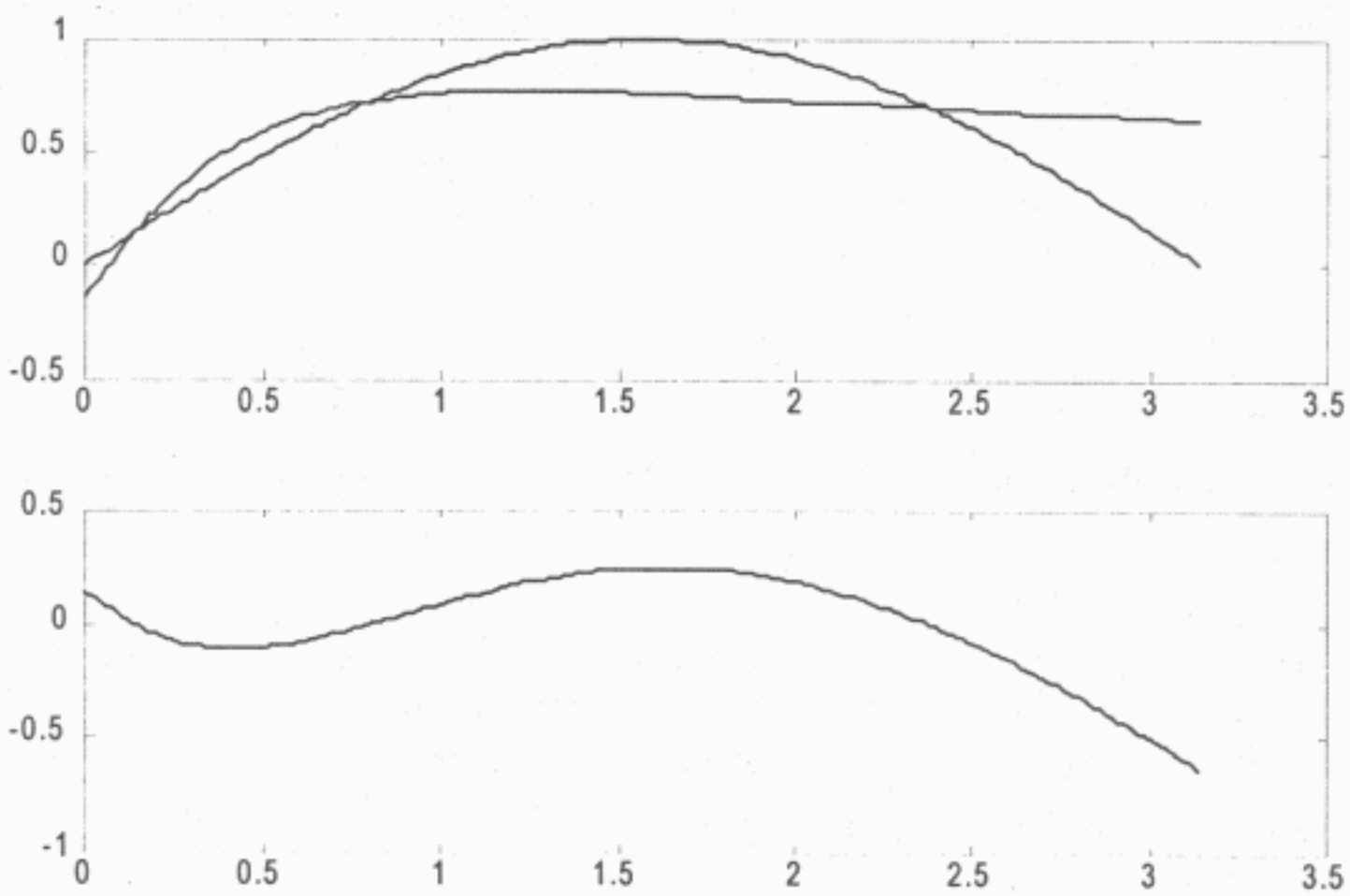
Výsledek aproximace jedné půlperiody funkce sinus racionální lomenou funkcí je možné ukázat také graficky. Je zřejmé, že v tomto případě tvar aproximační funkce není vhodný - není schopný sledovat základní průběh aproximované funkce. Přesto vidíme, že není problém nalézt optimální (tj. minimalizující kritérium) hodnoty parametrů aproximující funkce.

```
subplot(211)
plot(x,y,x,ya)
subplot(212)
plot(x,y-ya)
```

tučná čára - aproximovaná funkce

tečkovaná čára - aproximace racionální lomenou funkcí

dolní obrázek - chyba aproximace



Obrázek 7-5 Určení parametrů funkce - nelineární regrese

7.4 Řešení soustavy diferenciálních rovnic

Poslední standardní funkcí, kterou se budeme v této kapitole zabývat, je funkce pro řešení soustavy obyčejných diferenciálních rovnic prvního řádu. (Ordinary Differential Equations). Pokud si kladete otázku, jak se budou řešit diferenciální rovnice vyšších řádů, pak je potřeba si uvědomit, že diferenciální rovnice vyššího řádu než prvního lze převést na ekvivalentní soustavu diferenciálních rovnic řádu prvního.

Vrátíme-li se k MATLABu, pak tento poskytuje několik standardních funkcí pro řešení soustavy obyčejných diferenciálních rovnic. Základní standardní funkcí je funkce ode45 pro nonstiff[42] rovnice nebo ode15s pro stiff rovnice. Základní syntaxe funkce **ode45** je následující

[T,Y] = **ode45**(´nazev_funkce´,casovy_interval,pocatecni_podminky)

kde *nazev_funkce* je jméno m-funkce popisující soustavu diferenciálních rovnic. Zadává se buď jméno m-funkce v apostrofech nebo jméno proměnné typu řetězec obsahující název m-funkce. Tato m-funkce musí mít syntaxi

dy=nazev_funkce(t,y)

Výstupní sloupcový vektor *dy* představuje hodnoty derivací pro vektor hodnot y v čase t. Parametr *casovy_interval* představuje vektor o dvou prvcích - počáteční čas řešení t_0 a konečný čas řešení . Parametr

[42] V souvislosti s numerickým řešením diferenciálních rovnic je potřeba upozornit na problém stiff systémů. Jde velmi jednoduše řečeno o diferenciální rovnice, u kterých řešení obsahuje zároveň rychlé a pomalé složky řešení. V případě lineárních diferenciálních rovnic to je např. v případě, že charakteristická rovnice má kořeny, které se velmi liší ve velikosti. Praktický důsledek tohoto jevu se při řešení pomocí standardního algoritmu, který s tímto nepočítá, projeví tak, že rychlé jevy nejsou v řešení zachyceny. Proto na stiff problémy je potřeba používat speciální algoritmy.

pocatecni_podminky představuje vektor počátečních podmínek y_0 takový, že platí $y(t_0)=y_0$. Výstupní parametr T - sloupcový vektor- obsahuje časové okamžiky, ve kterých jsou určeny hodnoty řešení y. Vlastní řešení je uloženo v matici Y - počet řádků odpovídá počtu řádků T, počet sloupců odpovídá počtu veličin y řešené soustavy.

Ukažme si příklad - řešení Van der Polovy rovnice. Tato nelineární diferenciální rovnice druhého řádu má tvar

$$\frac{d^2 y_1}{dt^2} - \mu(1 - y_1^2)\frac{dy_1}{dt} + y_1 = 0$$

tuto rovnici druhého řádu lze přepsat na tvar soustavy dvou rovnic řádu prvního s vyjádřenými derivacemi

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = \mu(1 - y_1^2)y_2 - y_1$$

Takovouto soustavu můžeme vyjádřit formálně vektorovým zápisem

$$\frac{d\mathbf{y}}{dt} = \mathbf{F}(t,\mathbf{y}) \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad \frac{d\mathbf{y}}{dt} = \begin{bmatrix} dy_1/dt \\ dy_2/dt \end{bmatrix}$$

a v této podobě přepsat do uživatelské funkce fce_vdp např. tak jak je uvedeno v rámečku.

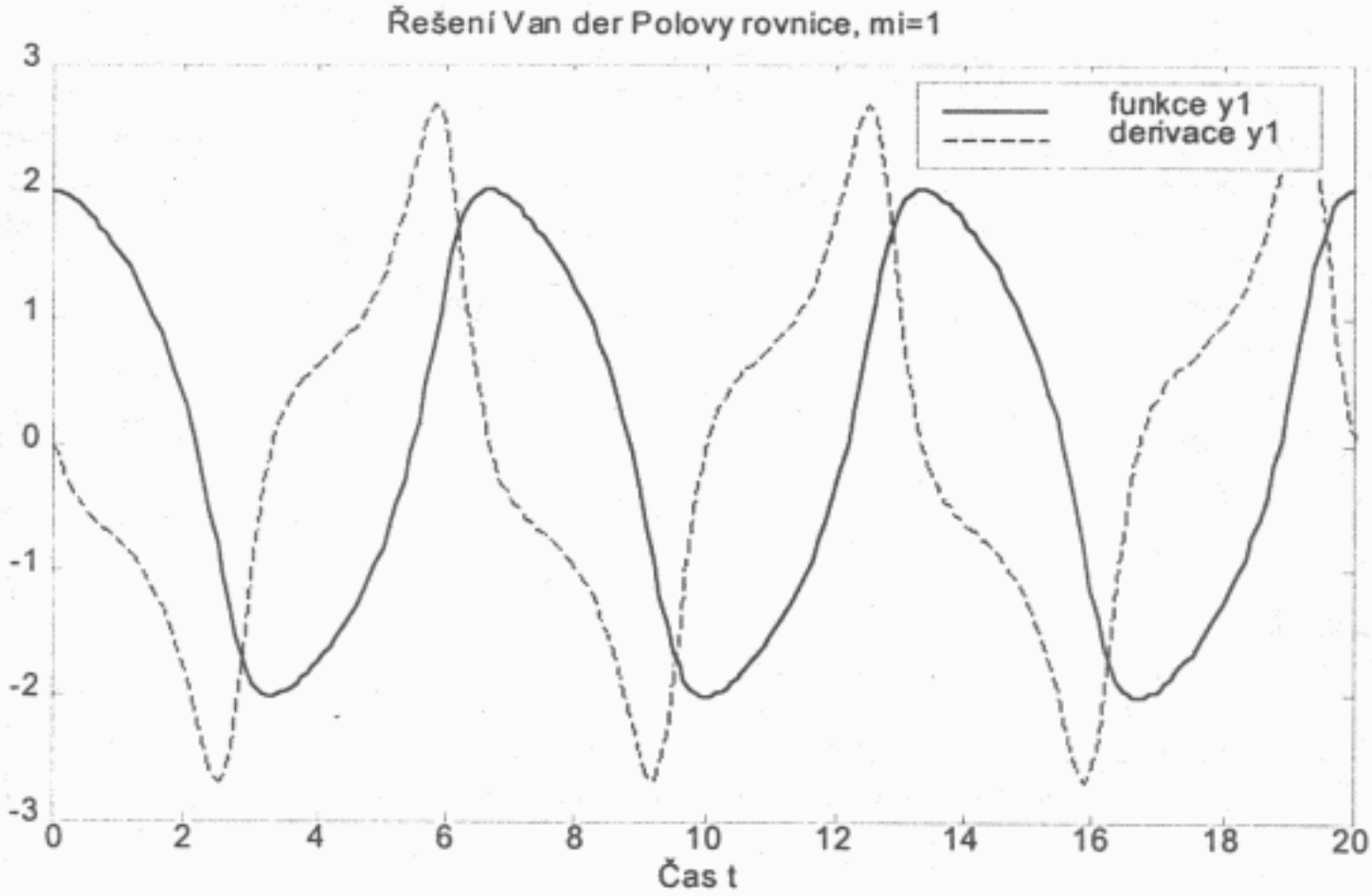
Určíme-li nyní počáteční podmínky např. hodnota y1 v čase 0 bude mít hodnotu 2 a derivace v čase 0 bude mít hodnotu 0 pak můžeme pro časový interval 0-20 rovnici řešit příkazem

```
function dy=fce_vdp(t,y)
%uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% Van der Polova diferencialni rovnice
%
%                    dy=fce_vdp(t,y)
%


mi=1;
dy=[y(2);mi*(1-y(1)^2)*y(2)-y(1)];
```

```
[t,y]=ode45('fce_vdp',[0,20],[2;0]);
```

a řešení nechat vykreslit příkazy

```
plot(t,y(:,1),'-',t,y(:,2),'--')
title('Řešení Van der Polovy rovnice, mi=1');
xlabel('čas t');
legend('funkce y1','derivace y2')
```



Obrázek 7-6 Průběh řešení Van der Poolovy rovnice

7.5 Seznam "funkcí funkcí" a příkazů pro řešení difer. rovnic

```
help funfun
```

```
  Function functions and ODE solvers.

  Optimization and root finding.
fmin      - Minimize function of one variable.
fmins     - Minimize function of several variables.
fzero     - Find zero of function of one variable.

  Numerical integration (quadrature).
quad      - Numerically evaluate integral, low order method.
quad8     - Numerically evaluate integral, higher order method.
dblquad   - Numerically evaluate double integral.

  Plotting.
ezplot    - Easy to use function plotter.
fplot     - Plot function.

  Inline function object.
inline    - Construct INLINE function object.
argnames  - Argument names.
formula   - Function formula.
char      - Convert INLINE object to character array.

  Utilities.
vectorize - Vectorize string expression or INLINE function object.
```

```
  Ordinary differential equation solvers.
  (If unsure about stiffness, try ODE45 first, then ODE15S.)
ode45     - Solve non-stiff differential equations, medium order method.
ode23     - Solve non-stiff differential equations, low order method.
ode113    - Solve non-stiff differential equations, variable order method.
ode23t    - Solve moderately stiff differential equations, trapezoidal rule.
ode15s    - Solve stiff differential equations, variable order method.
ode23s    - Solve stiff differential equations, low order method.
ode23tb   - Solve stiff differential equations, low order method.
odefile   - ODE file syntax.

  ODE Option handling.
odeset    - Create/alter ODE OPTIONS structure.
odeget    - Get ODE OPTIONS parameters.

  ODE output functions.
odeplot   - Time series ODE output function.
odephas2  - 2-D phase plane ODE output function.
odephas3  - 3-D phase plane ODE output function.
odeprint  - Command window printing ODE output function.
```

8. Jednoduché řešené příklady (MATLAB)

V této kapitole jsou uvedeny řešené příklady od nejjednodušších po složitější na procvičení práce s MATLABem. U většiny příkladů existuje více řešení, některá více některá méně elegantní či výpočetně efektivní. Rozhodující je to, aby výsledek byl správný. Obecně je preferováno vektorové řešení (je výrazně rychlejší) než programové řešení např. s využitím cyklu for ... end. Tento kurz je však úvodem do používání MATLABu proto není rozhodující elegance, ale správný výsledek.

8.1 Plnění vektorů

Vytvořte vektory či matice

a) sloupcový vektor a obsahující celá čísla 1..10

```
a=1:10; a=a';
```

b) řádkový vektor b obsahující celá čísla 10..1

```
b=10:-1:1;
```

c) matici A obsahující dva řádky celých čísel 1..10 a 10..1

```
A=[1:10;10:-1:1];
```
nebo
```
A=[a';b];
```

d) matici B obsahující dva sloupce celých čísel 1..10 a 10..1

```
B=[[1:10]',[10:-1:1]'];
```
nebo
```
B=[a,b'];
```

e) jednotkovou matici E o rozměru 10x10

```
E=eye(10,10);
```
nebo
```
E=eye(10);
```

f) nulovou matici N rozměru 5x10

```
N=zeros(5,10);
```

g) matici jednotek J o rozměru 10x5

```
J=ones(10,5);
```

h) matici náhodných čísel NC o rozměru 5x7

```
NC=rand(5,7);
```

8.2 Základní operace s maticemi a vektory

Vypočtěte

a) součin vektorů a,b a b,a

```
a*b

ans =    1     2     3     4     5     6     7     8     9    10
         2     4     6     8    10    12    14    16    18    20
         3     6     9    12    15    18    21    24    27    30
         4     8    12    16    20    24    28    32    36    40
         5    10    15    20    25    30    35    40    45    50
         6    12    18    24    30    36    42    48    54    60
         7    14    21    28    35    42    49    56    63    70
         8    16    24    32    40    48    56    64    72    80
         9    18    27    36    45    54    63    72    81    90
        10    20    30    40    50    60    70    80    90   100
```

```
b*a

ans =   385
```

b) druhou mocninu prvků vektoru b

```
b.^2

ans =    1     4     9    16    25    36    49    64    81   100
```

c) součin matic A,B a B,A

```
A*B

ans =   385   385
        220   220
```

```
B*A

ans =    11    11    11    11    11    11    11    11    11    11
         22    22    22    22    22    22    22    22    22    22
         33    33    33    33    33    33    33    33    33    33
         44    44    44    44    44    44    44    44    44    44
         55    55    55    55    55    55    55    55    55    55
         66    66    66    66    66    66    66    66    66    66
         77    77    77    77    77    77    77    77    77    77
         88    88    88    88    88    88    88    88    88    88
         99    99    99    99    99    99    99    99    99    99
        110   110   110   110   110   110   110   110   110   110
```

d) tabulku funkce (vektor) ys =sin(x) pro $x \in \langle 0, 2\pi \rangle$ v 1000 bodech

```
x=0:2*pi/999:2*pi; ys=sin(x);
```

e) tabulku funkce (vektor) ys1=sin(x^2)+sin(x)^2 pro x∈<0,2π> v 1000 bodech

```
x=0:2*pi/999:2*pi;
ys1=sin(x.^2)+sin(x).^2;
```

f) tabulku funkce (matici) YS=cos(x*y) pro x,y∈<-π,π> v 100 bodech

```
x=-pi:2*pi/99:pi; y=x;
```

řešení první (přímočaré)

```
for a=1:length(x),
    for b=1:length(y),
        XY(a,b)=x(a)*y(b);
    end
end
YS=cos(XY);
```

řešení druhé (s využitím funkce MATLABu **meshgrid**[43])

```
[X,Y]=meshgrid(x,y);
YS=cos(X.*Y);
```

g) matici NCN=NC*NC' její determinant a matici inverzní

```
NCN=NC*NC'
```

součin matice a matice transponované

```
    NCN =   2.0716    1.5065    1.7063    1.7785    1.7123
            1.5065    2.4858    2.4758    2.2091    1.8211
            1.7063    2.4758    3.2830    1.9539    1.9714
            1.7785    2.2091    1.9539    2.5666    1.7849
            1.7123    1.8211    1.9714    1.7849    2.0978
```

```
D=det(NCN)
```

determinant matice

```
    D =       0.2946
```

```
iNCN=inv(NCN)
```

inverzní matice

```
    iNCN =   2.3591    1.9015   -0.8484   -1.6971   -1.3350
             1.9015    5.8267   -2.5514   -3.5780   -1.1682
            -0.8484   -2.5514    1.8540    1.3771   -0.0067
            -1.6971   -3.5780    1.3771    3.3639    0.3350
            -1.3350   -1.1682   -0.0067    0.3350    2.3017
```

[43] tato funkce vytvoří dvě matice - jednu se stejnými hodnotami v každém řádku a druhou se stejnými hodnotami v každém sloupci. Tyto matice je pak možné použít (ve spojení s operacemi člen po členu) jako přímou náhradu proměnných x a y ve skalárních matematických výrazech

h) ukažte, že součin matice a její inverze je jednotková matice

```
NCN*iNCN

    ans =    1.0000    0.0000    0.0000    0.0000    0.0000
             0.0000    1.0000    0.0000         0         0
             0.0000    0.0000    1.0000    0.0000         0
             0.0000    0.0000    0.0000    1.0000         0
             0.0000         0    0.0000    0.0000    1.0000
```

i) součet čísel od 1 do 1000

řešení první (přímočaré)

```
S=0; for a=1:1000, S=S+a; end; S

    S =       500500
```

řešení druhé (prácc s vektory - součin řádkového a sloupcového vektoru stejné délky)

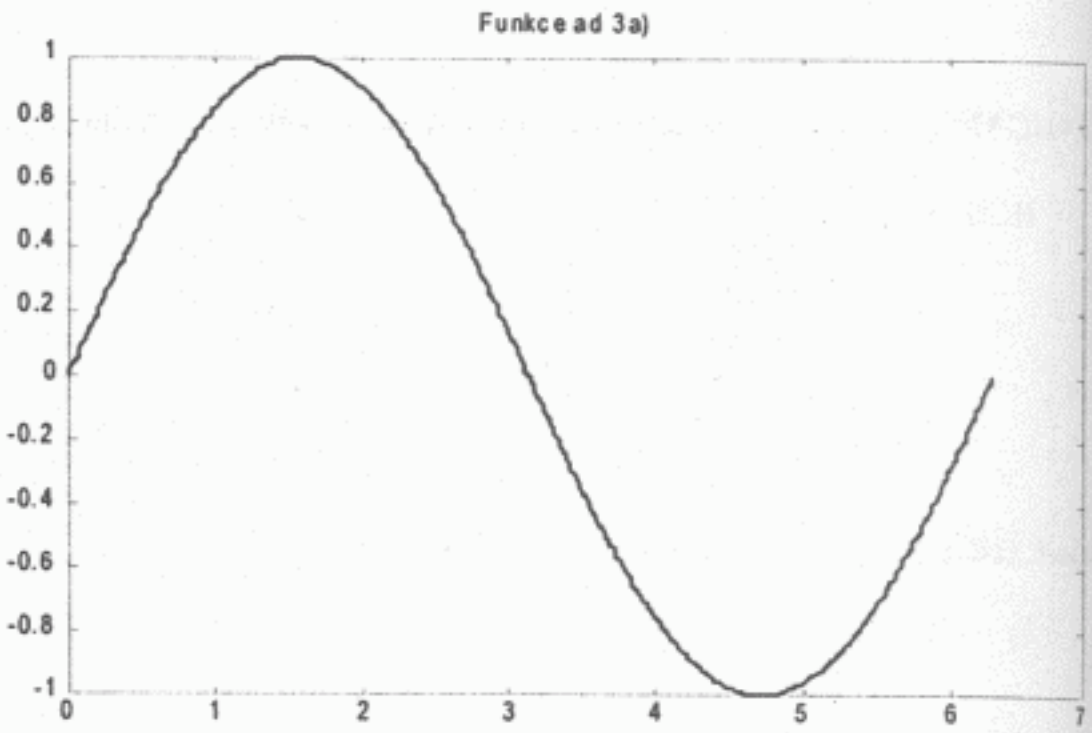
```
S=[1:1000]*ones(1000,1)

    S =       500500
```

8.3 Kreslení funkcí

Nakreslete průběh funkce

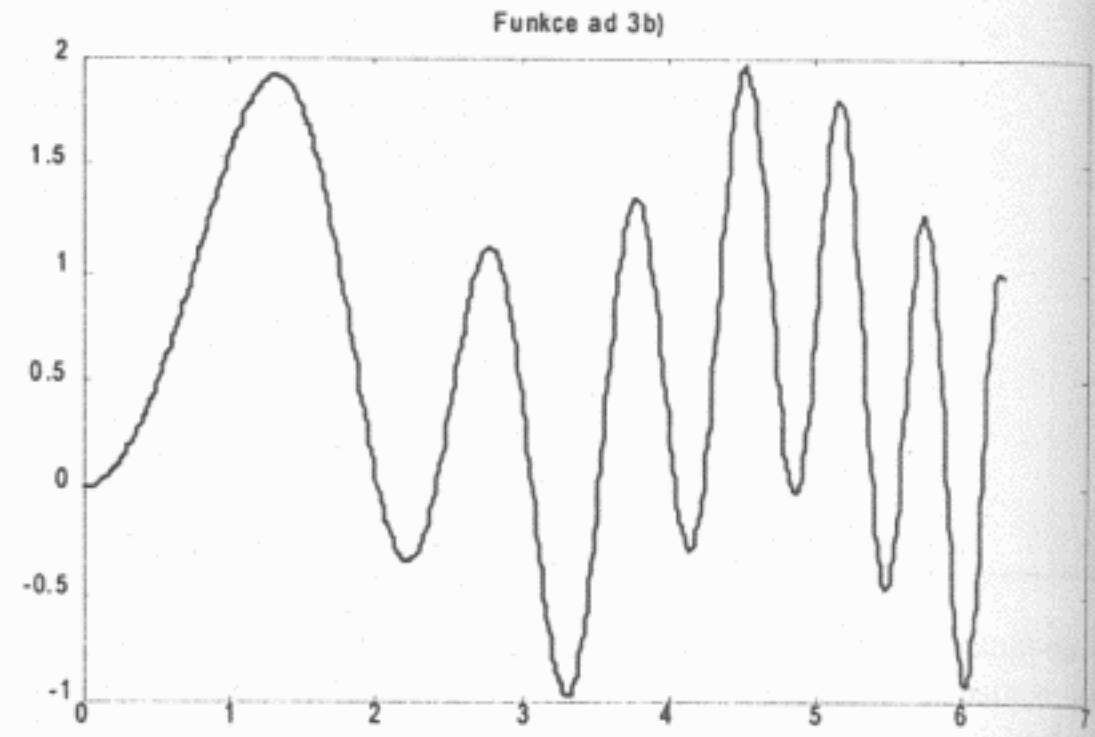
a) y=sin(x) pro x∈<0,2π> v 1000 bodech

```
x=0:2*pi/999:2*pi;
y=sin(x);
plot(x,y);
title('Funkce ad 3a)')
```



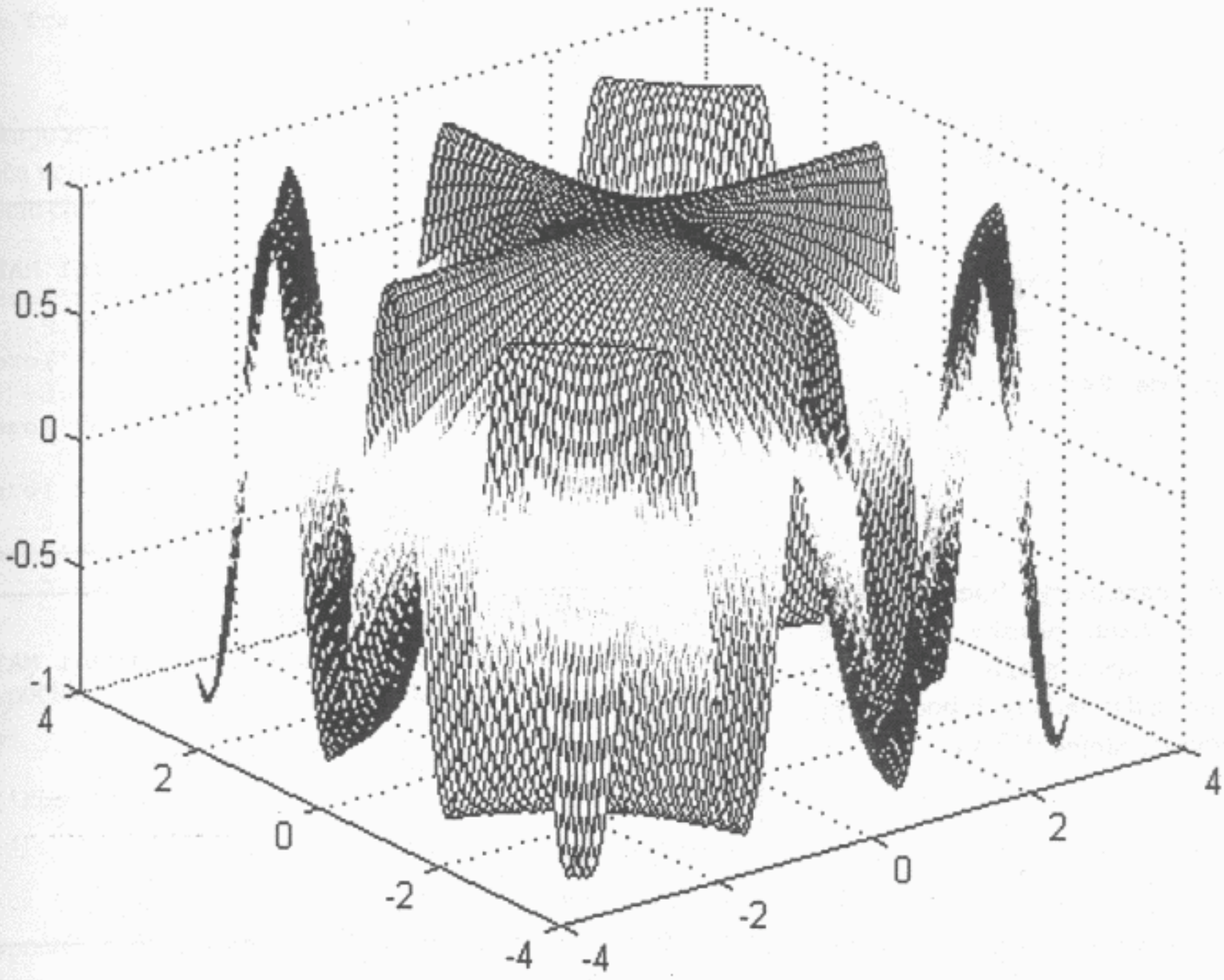
b) y=sin(x^2)+sin(x)^2 pro x∈<0,2π> v 1000 bodech}

```
x=0:2*pi/999:2*pi;
y=sin(x.^2)+sin(x).^2;
plot(x,y);
title('Funkce ad 3b)')
```



c) Z=cos(x*y) pro x,y∈<-π,π> v 100 bodech}

```
x=-pi:2*pi/99:pi; y=x;[X,Y]=meshgrid(x,y); Z=cos(X.*Y);
mesh(x,y,Z)
```



8.4 Tvorba uživatelských funkcí

Vytvořte m-funkci, která:

a) vypočte hodnotu funkce y=(exp(x)+exp(-x))/2 (parametr x může být i vektor)

Příklad použití

```
t=-2*pi:4*pi/999:2*pi;
plot(t,fce_84a(t));
```

```
function y=fce_84a(x)
%uzivatel.funkce pro 'Uvod do pouzivani MATLAB'
%
%                    y=fce_84a(x)

y=(exp(x)+exp(-x))/2;
```

b) vypočtc hodnotu funkce y=x*cos(x)
{musí se použít operace násobení člen po členu, aby nedošlo k vektorovému násobení}

Příklad použití
```
x=-2*pi:4*pi/999:2*pi;
plot(x,fce_84b(x))
```

```
function y=fce_84b(x)
%uzivatel.funkce pro 'Uvod do pouzivani MATLAB'
%
%                    y=fce_84b(x)

y=x.cos(x);
```

c) naplní matici hodnot funkce z=sin(x^2+y^2) v bodech daných hodnotami vektoru x a y

Příklad použití
```
x=-2*pi:4*pi/99:2*pi;
y=x;
mesh(x,y,fce_84c(x,y))
```

```
function z=fce_84c(x,y)
%uzivatel.funkce pro 'Uvod do pouzivani MATLAB'
%
%                    z=fce_84c(x,y)

[X,Y]=meshgrid(y,x);
z=sin(X.^2+Y.^2);
```

d) určí parametry lineární regrese (směrnici a posun přímky optimálně - z hlediska minima kvadrátů - prokládající měřená data) z dat zadaných body [x,y] (x a y jsou vektory stejné délky)

```
function [k,q]=fce_84d(x,y)
%uzivatel.funkce pro 'Uvod do pouzivani MATLAB'
%
%                    [k,q]=fce_84d(x,y)

r=polyfit(x,y,1);
k=r(1); q=r(2);
```

8.5 Použití uživatelských funkcí

S využitím vlastních m-funkcí a standardních funkcí vypočtěte a nakreslete

a) nalezněte řešení rovnice $x e^x = 10$ (funkce fzero)

```
koren=fzero('fce_85a',0)
   koren =    1.7455
```

```
function y=fce_85a(x)
% uzivatel. fce pro 'Uvod do pouzivani MATLAB'
%
%                    y=fce_85a(x)

y=x.*exp(x)-10;
```

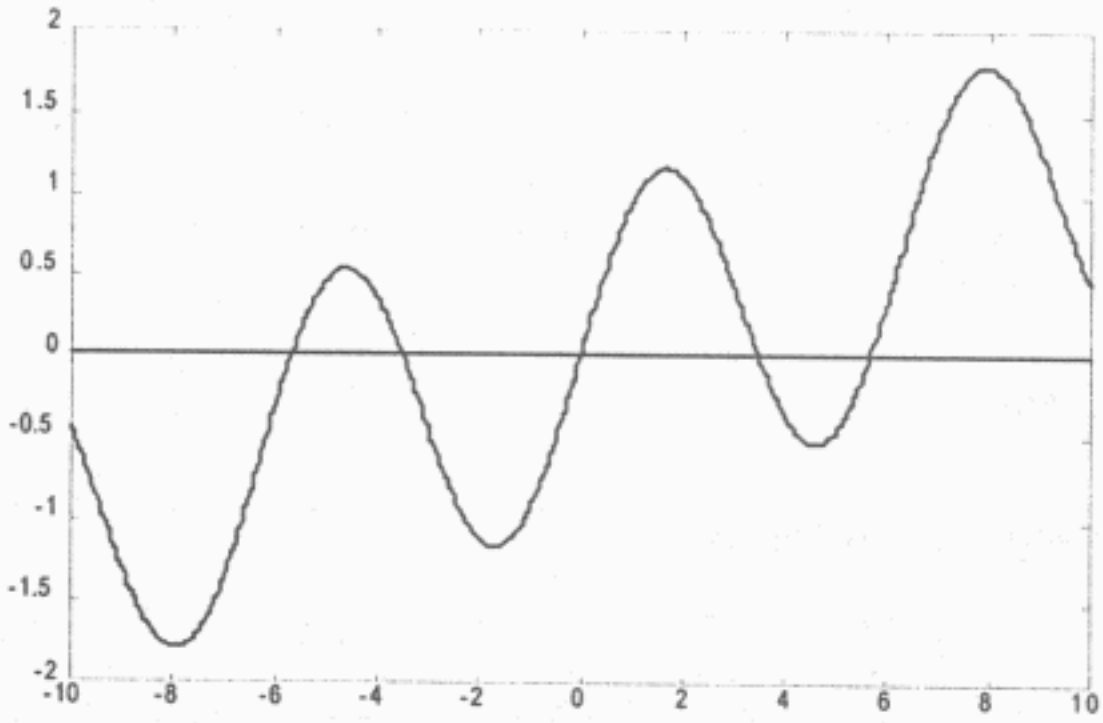
b) nalezněte řešení rovnice

$$\sin(x)+\frac{x}{10}=0 \quad \text{(funkce fzero)}$$

průběh funkce (více řešení)

```
x=-10:0.01:10;
z=zeros(size(x));
plot(x,fce_85b(x),x,z)
```

```
function y=fce_85b(x)
% uzivatel. fce pro 'Uvod do pouzivani MATLAB'
%
%                    y=fce_85b(x)

y=sin(x)+x/10;
```



z obrázku je zřejmé, že příklad má 5 řešení - je potřeba volit výchozí body poblíž řešení, ke kterému chceme dospět

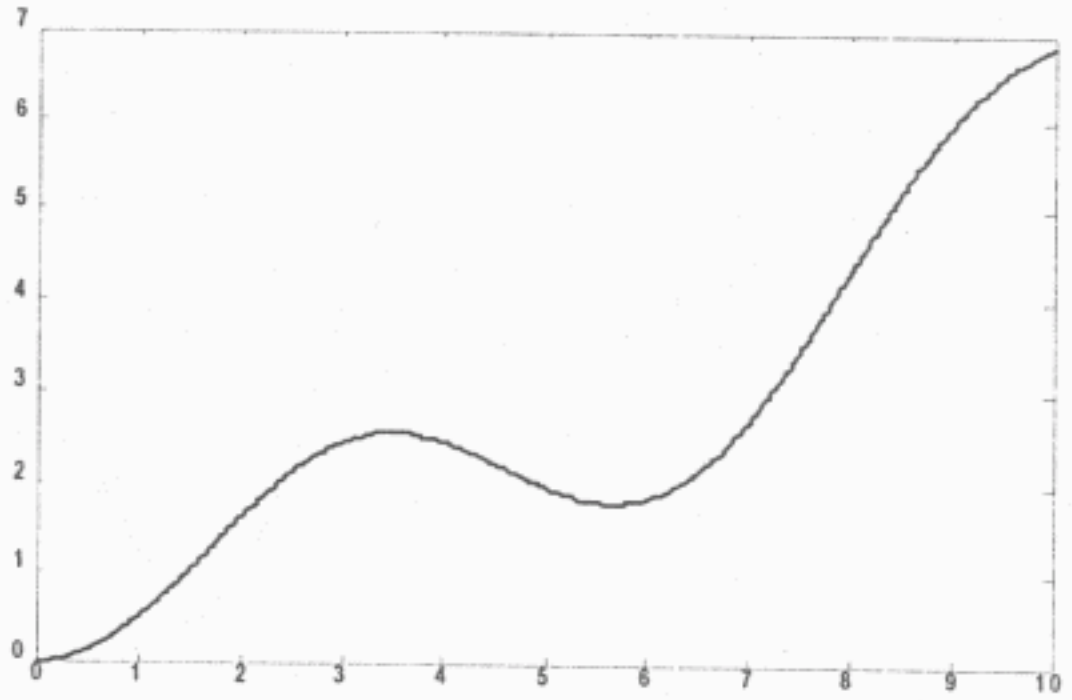
```
k1=fzero('fce_85b',-6)
k1 =   -5.6792
k2=fzero('fce_85b',-3)
k2 =   -3.4991
k3=fzero('fce_85b',0)
k3 =      0
k4=fzero('fce_85b',3)
k4 =    3.4991
k5=fzero('fce_85b',6)
k5 =    5.6792
```

c) vypočtěte integrál od 0 do 10 funkce dle 8.5b) (funkce quad8)

```
quad8('fce_85b',0,10)
ans =    6.8391
```

d) vypočtěte a nakreslete průběh integrálu od 0 do t funkce dle 8.5b) pro t∈<0,10> v 1000 bodech (funkce quad8)

```
t=0:10/999:10;
for a=1:1000,
  y(a)=quad('fce_85b',0,t(a));
end
plot(t,y)
```



e) řešení diferenciální rovnice $2\frac{dh}{dt}+0.5\sqrt{h}=5 \quad h(t=0)=0 \quad t\in\langle 0,100\rangle$ (fce ode45)

```
[t,y]=ode45('fce_85h',[0;100],0);
plot(t,y)
```

```
function dy=fce_85e(t,x)
% uzivatel. fce pro 'Uvod do pouz. MATLAB'
%
%        dy=fce_85e(t,x)

dy=(5-0.5*sqrt(x(1)))/2;
```

f) řešení diferenciální rovnice $\frac{d^2y}{dx^2}+0.75\frac{dy}{dx}+6y=\sin(\sqrt{x}), \quad \left.\frac{dy}{dx}\right|_{x=0}=0, \; y(0)=1, \; x\in<0,10>$

(diferenciální rovnici řádu druhého musíte převést na soustavu dvou diferenciálních rovnic řádu prvého, funkce ode45)

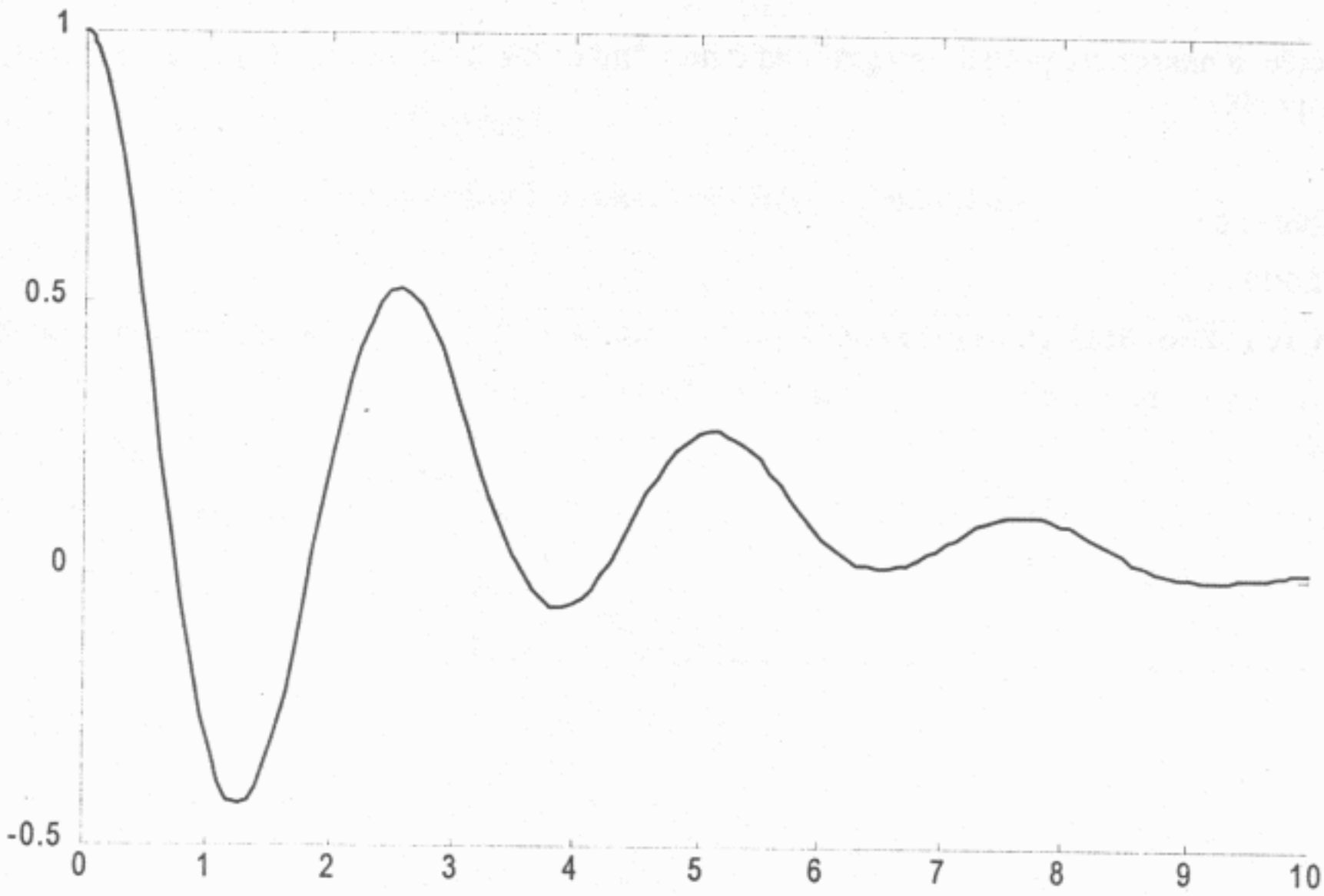
substituce $\frac{dy}{dx}=z$

$$\frac{dz}{dx}+0.75z+6y=\sin(\sqrt{x})$$

$$Y=\begin{bmatrix} z \\ y \end{bmatrix} \quad \frac{dY}{dx}=\begin{bmatrix} \frac{dz}{dx} \\ \frac{dy}{dx} \end{bmatrix}=\begin{bmatrix} \sin(\sqrt{x})-0.75z-6y \\ z \end{bmatrix}$$

```
function dy=fce_85f(x,y)
% uzivatel.fce pro 'Uvod do MATLAB'
%
%        dy=fce_85f(x,y)
dy=zeros(2,1);
dy(1)=sin(sqrt(x))-0.75*y(1)-6*y(2);
dy(2)=y(1);
```

```
[x,y]=ode45('fce_85f',[0;10],[0;1]);  plot(x,y(:,2))
```



8.6 Kombinace více funkcí

Vyřešte následující příklady

a) nalezněte hodnotu parametru a, pro který platí rovnice

$$\int_{-0.5}^{+0.5} e^{-x^2/a}\,dx = 0.5$$

(funkce quad8 a fzero).

Řešení: nejprve je potřeba vytvořit m-funkci definující zadaný integrand (fce_86a1) a potom další m-funkci definující funkci jejichž nulovou hodnotu hledáme

```
function y=fce_86a1(x,a)
% uzivatel. fce pro 'Uvod do pouzivani MATLAB'
%
%        y=fce_86a1(x,a)
y=exp(-x.^2/a);
```

```
function y=fce_86a(a)
% uzivatel. fce pro 'Uvod do pouzivani MATLAB'
%
%        y=fce_86a(a)
y=quad8('fce_86a1',-0.5,0.5,[],[],a)-0.5;
```

```
a=fzero('fce_86a',0.1)
a =    0.0818
```

b) určete v kterém čase t dosáhne řešení y(t) příkladu 8.5c) hodnoty y(t)=50

(funkce ode45 a fzero)

Řešení: využijeme již hotové m-funkce fce_85e, která byla použita pro řešení diferenciální rovnice. Dále vytvoříme další m-funkci, která definuje funkci jejichž nulovou hodnotu hledáme (fce_86b)

```
function y=fce_86b(t)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
%
%        y=fce_86b(t)

[x,yp]=ode45('fce_85e',[0;t],0);
y=yp(end)-50;
```

```
t=fzero('fce_86b',50)
t =   41.6672
```

c) řešte aproximaci horní polokružnice (se středem v počátku souřadnic a poloměrem 10) polynomem 4. řádu s kritériem minima sumy kvadrátů odchylek, minima sumy absolutních hodnot a minimaxu[44]. Nakreslete průběh chyby (odchylku) pro všechny aproximace (funkce fmins)

Řešení: je nutné vytvořit m-funkci, která určí průběh aproximační křivky pro danou sadu parametrů a vyhodnotí hodnotu kritéria. Je vhodné aby tato m-funkce měla jako vstupní parametr vektor původních hodnot a jako výstupní vektor vracela přímo vektor odchylek. Protože tato m-funkce bude argumentem funkce fmins musí mít jako první výstupní parametr minimalizovanou hodnotu a jako první

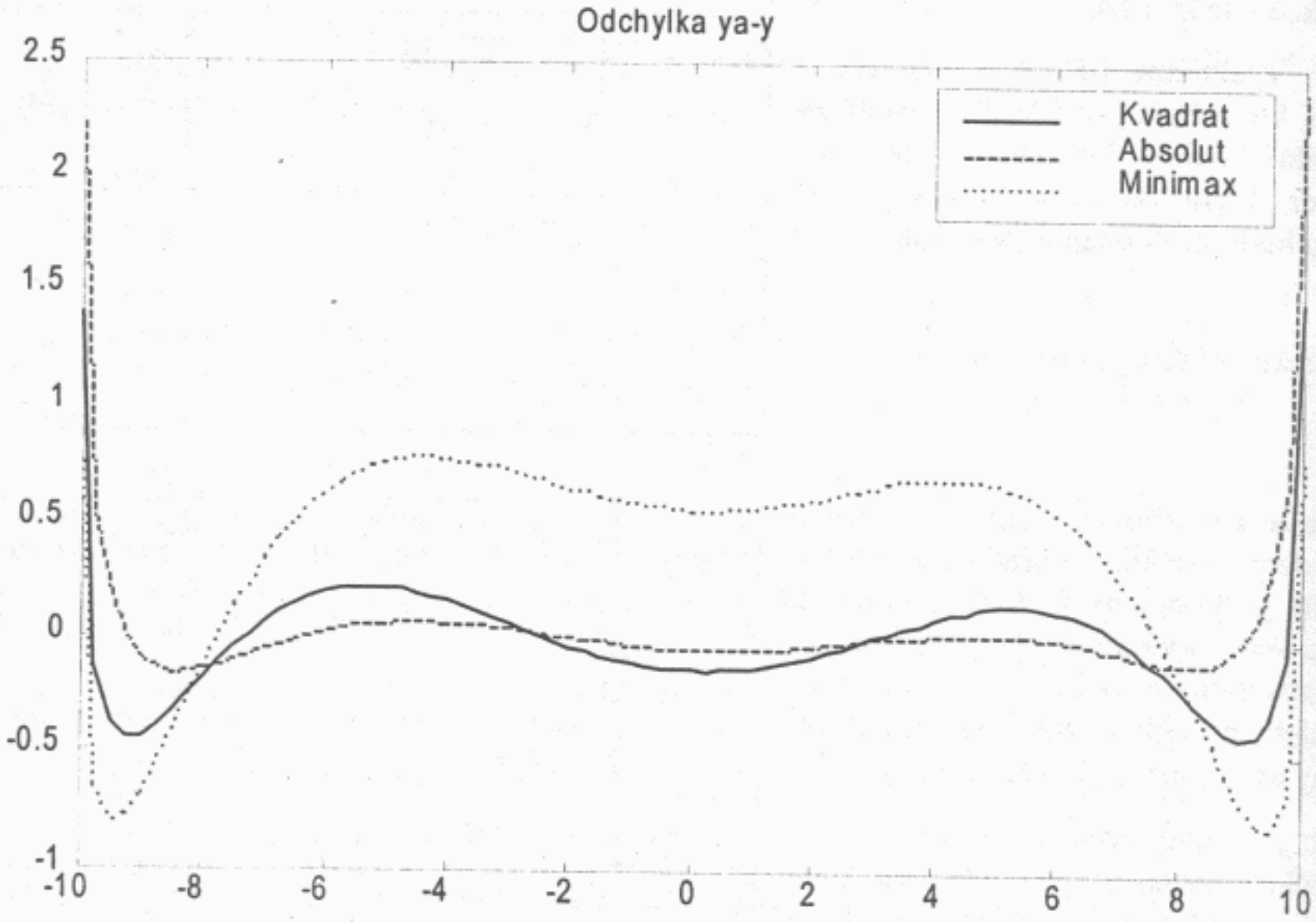
```
function [Kr,e]=fce_86c(par,x,y,Kriterium)
% uzivatel. fce pro 'Uvod do pouzivani MATLAB'
%
%        [Kr,e]=fce_86c(par,x,y,Kriterium)

N=length(x);          %pocet prvku vektoru
x=x(:);y=y(:);        %preved na sloupcove vektory
ya=polyval(par,x); %vypocet aprox. v bodech x
e=ya-y;               %odchylka v bodech x
%suma kvadratu odchylek
Kr1=e'*e;          %skalarni soucin dvou vektoru
%suma absolutnich hodnot
Kr2=sum(abs(e));
%maximalni odchylka (v absolutni hodnote)
Kr3=max(abs(e));
switch lower(Kriterium)
  case 'kvadrat', Kr=Kr1;
  case 'absolut', Kr=Kr2;
  case 'minimax', Kr=Kr3;
  otherwise, disp('Nezname kriterium');
end
```

[44] maximální odchylka aproximace na uvažovaném intervalu je ze všech přípustných aproximací nejmenší

vstupní parametr vektor parametrů (fce_86c)

data polokružnice a počáteční hodnoty parametrů

```
x=-10:0.25:10;
y=sqrt(10^2-x.^2);
p0=[0 0 0 0 1];
p1=fmins('fce_86c',p0,[],[],x,y,'kvadrat');          vyhledání minima
p2=fmins('fce_86c',p1,[],[],x,y,'absolut');          vyhledání minima
p3=fmins('fce_86c',p1,[],[],x,y,'minimax');          vyhledání minim
[kr,e1]=fce_86c(p1,x,y,'kvadrat');                   výpočet odchylky aproximace
[kr,e2]=fce_86c(p2,x,y,'absolut');                   výpočet odchylky aproximace
[kr,e3]=fce_86c(p3,x,y,'minimax');                   výpočet odchylky aproximace
plot(x,e1,x,e2,'--',x,e3,':'); title('Odchylka ya-y');
ylabel('Odchylka'); legend('Kvadrát','Absolut','Minimax');
```



d) nalezněte parametry a,b,c,d,e funkce $y=\left(a+bx+cx^2\right)e^{dx+ex^2}$ takové, aby aproximace horní polokružnice (o středu v počátku souřadnic a poloměru 10) touto funkcí byla z hlediska minima maximální odchylky optimální (funkce fmins). Nakreslete průběh chyby.

```
function [Kr,e]=fce_86d(p,x,y)
%uzivatel.fce pro 'Uvod do pouzivani MATLAB'
%
%        [Kr,e]=fce_86d(p,x,y)


x=x(:);y=y(:);
ya=(p(1)+p(2)*x+p(3)*x.^2).*exp(p(4)*x+p(5)*
x.^2);
e=ya-y;
Kr=max(abs(e));
```

data polokružnice

```
x=-10:0.5:10;y=sqrt(10^2-x.^2);
```

nástřel parametrů

```
p0=[1 1 0 0 0];
```

výpočet parametrů minim. kritérium

```
p=fmins('fce_86d',p0,[],[],x,y)
```

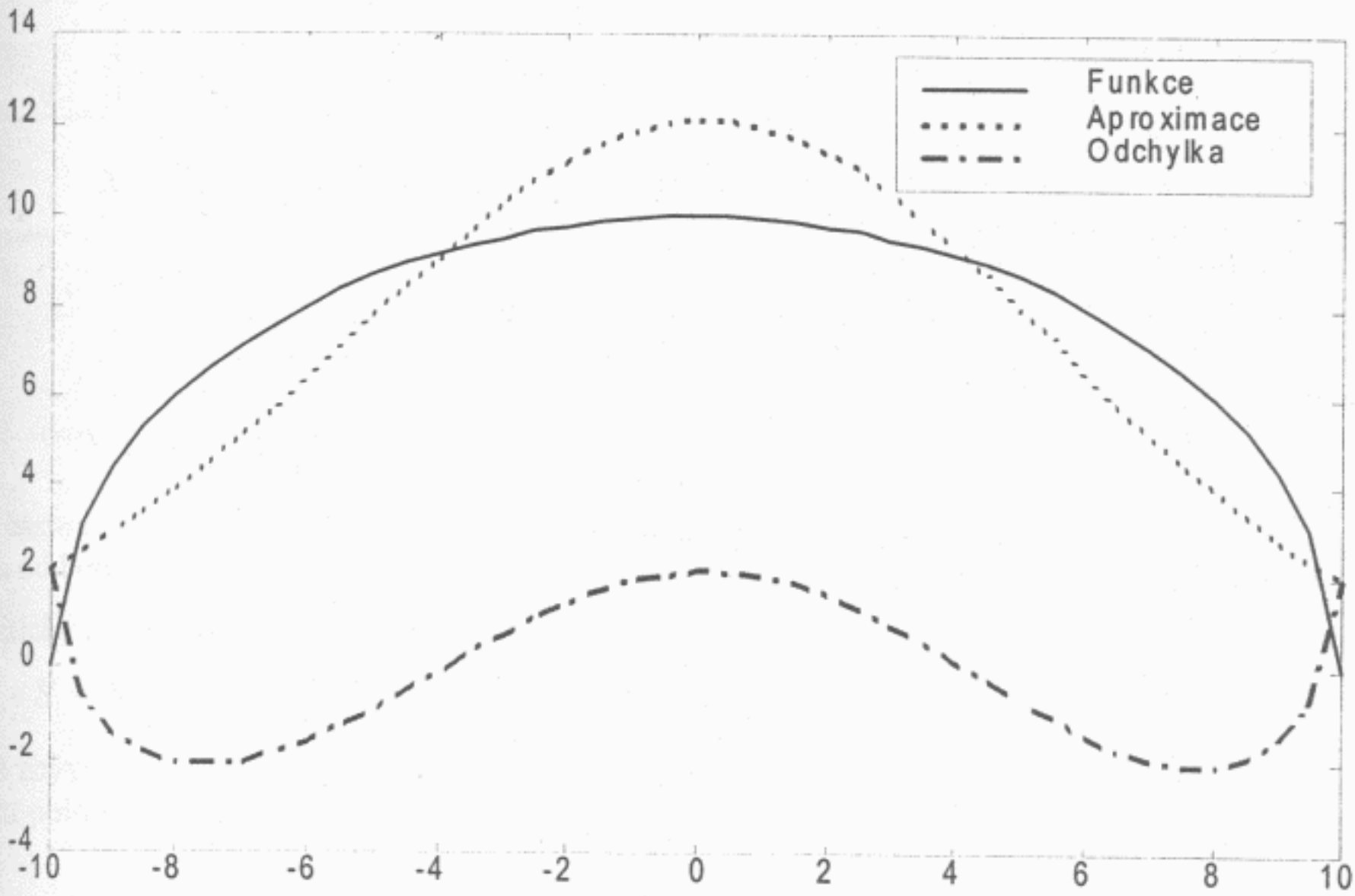
```
p =12.3264 0.4814 -0.0066 -0.0332 -0.0163
```

výpočet aproximace

```
ya=(p(1)+p(2)*x+p(3)*x.^2).*exp(p(4)*x+p(5)*x.^2);
```

výpočet odchylky a vykreslení původní funkce, aproximace a odchylky

```
plot(x,y,x,ya,':',x,ya-y,'-.'), legend('Funkce','Aproximace','Odchylka')
```



9. Komplexní řešené příklady (MATLAB)

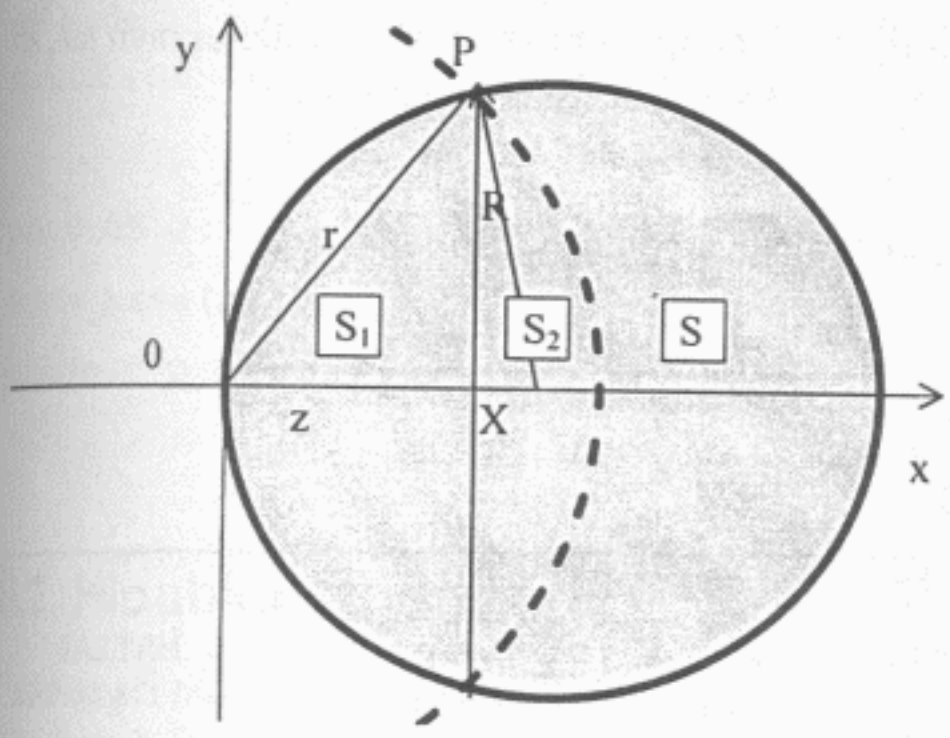
Následující příklady by měly dokumentovat využití MATLABu pro řešení složitějších úloh. Princip řešení jednotlivých úloh nemá samozřejmě s MATLABem nic společného, ale znalost možností tohoto prostředku umožňuje jednodušší návrh řešení. Lze navrhnout řešení s poměrně složitými částmi, které už jsou kompletně řešitelné prostředky MATLABu.

Možnosti MATLABu a relativní jednoduchost jeho použití umožňují provádění třeba složitých pomocných výpočtů. Např. nemám-li k dispozici tabulky s tabelovanými hodnotami některých fyzikálních konstant (třeba koeficienty přestupu tepla, koeficienty odporu prostředí atd.) můžu provést pomocné výpočty pro získání smysluplných odhadů jejich hodnot. Také možnost grafického zobrazení průběhu veličin může přispět k verifikaci (ověření) toho, zda spočítaný výsledek je správný či zda neexistují ještě další řešení.

Většina následujících příkladů vychází z jednoduchých a snadno pochopitelných zadání a předpokládá pouze všeobecné znalosti matematiky a fyziky a schopnost logického uvažování.

9.1 Koza na pastvě

Máme před sebou zdánlivě jednoduchý úkol - na jak dlouhý provaz uvázat kozu, aby vypásla přesně polovinu kruhového palouku. Koza bude uvázána ke kolíku zatlučenému na obvodu palouku.



Obrázek 9-1 Schéma pro příklad Koza na pastvě

Pokud máme začít řešit úlohy tohoto typu pak základní pravidlo (zavedené autorem) zní *"Dobře nakreslený obrázek s vhodně zvolenými souřadnicemi je polovina úspěchu"*.

Zvolme počátek souřadnic v místě zatlučení kolíku, označme délku provazu jako r a poloměr palouku jako R. Viz také schéma na obr. 9-1. Bod X je průsečík kolmice spuštěné z bodu P - místa, kde se protíná obvod oblasti vypasené kozou s obvodem palouku, s osou x. Jeho vzdálenost od počátku souřadnic označíme třeba z. Vzhledem k tomu, že plocha je symetrická podle osy x, můžeme se zabývat pouze částí plochy nad osou x.

Plocha vypasená kozou je pak tvořena dvěma částmi - plochou S_1 (vymezenou úsečkami 0X , XP a obloukem P0) a plochou S_2 (vymezenou úsečkou od bodu X k průsečíku provazu s osou x, obloukem z tohoto místa po bod P a úsečkou PX).

Matematická formulace úlohy - základní rovnice je pak

$$S_1 + S_2 = S$$

Nyní už zbývá pouze jednotlivé plochy vyjádřit a rovnici vyřešit. Nejjednodušší je vyjádření plochy S. Ta má tvořit polovinu plochy palouku - v našem vyjádření ploch pouze nad osou x jde o čtvrtinu celé plochy - tedy

$$S = {\pi R^2}/{4}$$

Plocha S_1 je plocha nad osou x (od 0 do z) omezená shora kružnicí se středem v bodě [R,0] a poloměrem R a lze ji tedy vyjádřit pomocí integrálu

$$\textit{obvod palouku} \quad (x-R)^2 + y^2 = R^2 \qquad \textit{plocha} \quad S_1 = \int_0^z \sqrt{R^2 - (x-R)^2}\,dx$$

Plocha S_2 je plocha nad osou x (od z do r) omezená shora kružnicí se středem v bodě [0,0] a poloměrem r a lze ji tedy vyjádřit pomocí integrálu

$$\textit{obvod vypasené plochy} \quad x^2 + y^2 = r^2 \qquad \textit{plocha} \quad S_2 = \int_z^r \sqrt{r^2 - x^2}\, dx$$

Nyní zbývá pouze vyjádřit hodnotu z v závislosti na hodnotách r a R. Vyjdeme-li opět z obrázku snadno zjistíme, že úsečka XP tvoří výšku v trojúhelníku jehož všechny strany jsou známy. Aplikací Pythagorovy věty lze získat vztah pro závislost délky z na hodnotách r a R.

$$r^2 = v^2 + z^2 \quad \wedge \quad R^2 = v^2 + (R - z)^2 \quad \Rightarrow \quad z = \frac{r^2}{2R}$$

Základní vztah vyjádřený pomocí parametrů r a R lze nyní vyjádřit jako

$$\int_0^{\frac{r^2}{2R}} \sqrt{R^2 - (x - R)^2}\, dx + \int_{\frac{r^2}{2R}}^{r} \sqrt{r^2 - x^2}\, dx = \pi \frac{R^2}{4}$$

Nyní už by bylo možné pro danou hodnotu R určit odpovídající hodnotu r, která by zajišťovala platnost předchozí rovnice. My ale budeme rafinovanější. Následujícími substitucemi lze rovnici dále upravit tak, že se v ní bude vyskytovat pouze jediná neznámá **p** - podíl délky provazu k poloměru palouku.

$$t = \frac{x}{R} \quad u = \frac{x}{r} \quad \cdots \quad R^2 \int_0^{r^2/2R^2} \sqrt{1 - (1 - t)^2}\, dt + r^2 \int_{r/2R}^{1} \sqrt{1 - u^2}\, du = \pi \frac{R^2}{4}$$

$$\mathbf{p} = \frac{r}{R} \quad \cdots \quad \int_0^{\frac{\mathbf{p}^2}{2}} \sqrt{1 - (1 - t)^2}\, dt + \mathbf{p}^2 \int_{\frac{\mathbf{p}}{2}}^{1} \sqrt{1 - u^2}\, du - \pi/4 = 0$$

Tato rovnice je již výchozí pro konkrétní řešení v MATLABu. Co pro její řešení potřebujeme? Dvě standardní funkce MATLABu - hledání kořenu funkce a výpočet určitého integrálu. Musíme vytvořit tři m-funkce - dvě m-funkce definující funkce, které chceme integrovat a jednu m-funkci, jejichž kořen chceme nalézt. Jak mohou např. vypadat m-funkce popisující integrandy je uvedeno v rámečku.

```
function y=fce_int1(x)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% y=sqrt(1-(1-x)^2)
%                        y=fce_int1(x)
y=sqrt(1-(1-x).^2);

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


function y=fce_int2(x)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% y=sqrt(1-x^2)
%                        y=fce_int2(x)
y=sqrt(1-x.^2);
```

M-funkce popisující vztah, jehož kořen se bude hledat, využije dříve definované m-funkce popisující integrandy. Tato m-funkce (umožňující jako vstupní parametr použít vektor) může vypadat např. takto.

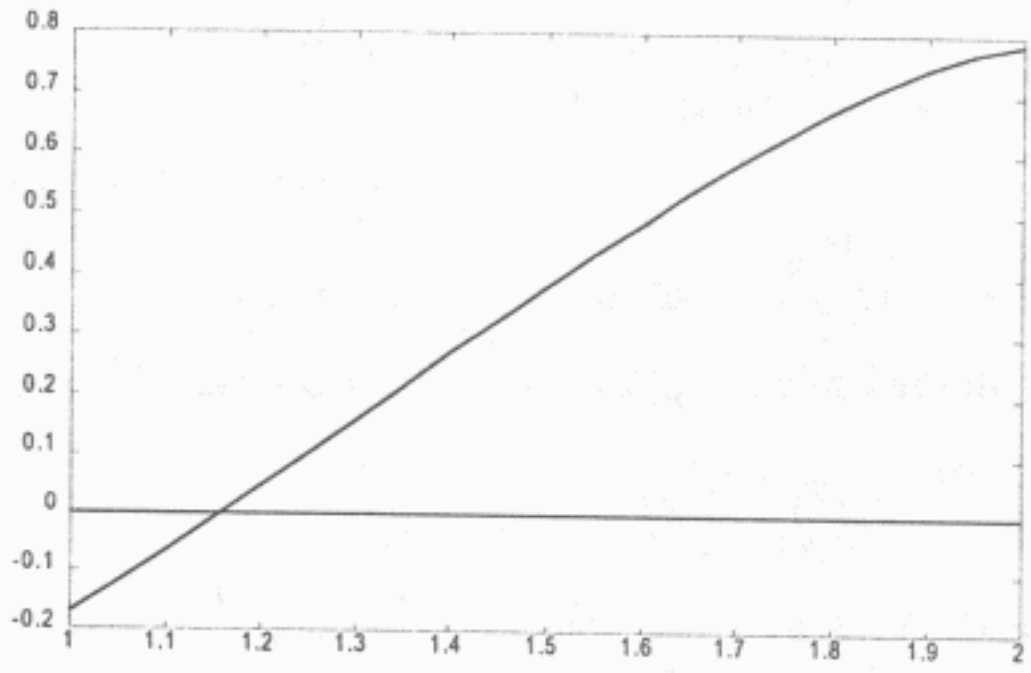
```
function y=fce_koza(p)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% priklad "Koza na pastve"
%
%                          y=fce_koza(p)

for i=1:length(p),
    S1(i)=quad8('fce_int1',0,p(i)^2/2);
    S2(i)=quad8('fce_int2',p(i)/2,1);
end
y=S1+p.^2.*S2-pi/4;
```

Vlastní řešení, vyhledání takového poměru délky provazu ku poloměru palouku p, který splňuje uvedenou rovnici, je záležitostí jediného příkazu

```
p=fzero('fce_koza',1)
  p =    1.1587
```

Nyní je vhodné ověřit, zda neexistuje více řešení. Z obrázku plyne, že nemá smysl řešení mimo interval $p \in \langle 1,2 \rangle$. Ukažme si, že na tomto intervalu je pouze jedno řešení tj. naše rovnice pro p má na uvedeném intervalu pouze jeden průsečík s osou x.

```
p=1:0.05:2;
f=fce_koza(p);
plot(p,f,p,zeros(size(p)))
```



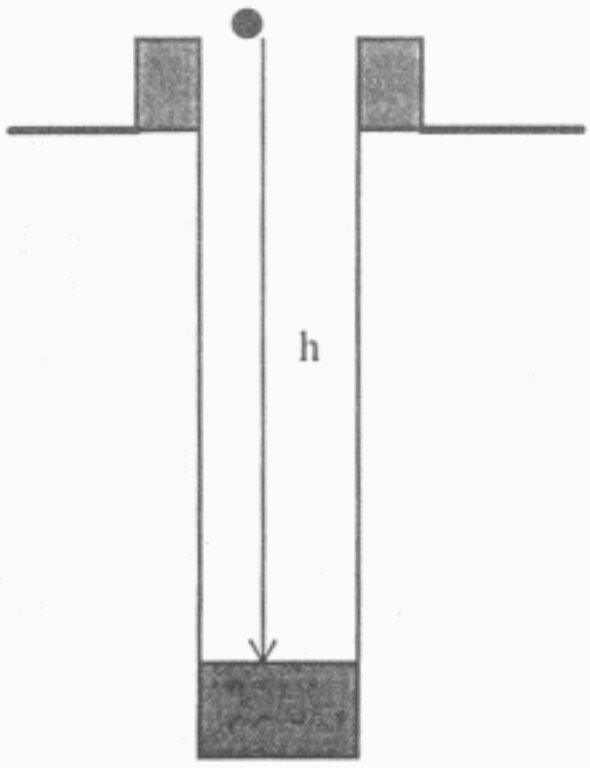
9.2 Hloubka studně

Často se při návštěvě nějakého hradu dozvíme různé zcela zbytečné informace mimo jiné třeba i o hloubce hradní studně. Jak ale tuto informaci zjistit když není po ruce průvodce (který by si ji pohotově vymyslel) a my jsme se vsadili třeba s přítelkyní? Jedna možnost je vytáhnout olovnici, dostatečně dlouhý provaz a hloubku k hladině změřit. Avšak takto vybaveni na výlet s přítelkyní obvykle nevyrážíme. Zato hodinky se stopkami jsou poměrně běžné a tak můžeme provést jiné měření. Spustíme stopky a současně vrhneme do studně ocelovou kuličku[45] a posloucháme až to šplouchne. V tomto okamžiku zastavíme stopky. Pak už jen stačí provést referenční experiment na studni známé hloubky, sehnat počítač s instalovaným MATLABem a když nezapomeneme změřené časy, tak po chvíli počítání můžeme oznámit hloubku studně. Vzhledem k tomu, že to přítelkyně po nás pravděpodobně nedokáže přepočítat, určitě sázku vyhrajeme.

Jak tedy ze známého času t, který uplynul od okamžiku upuštění předmětu (s nulovou počáteční rychlostí, v gravitačním poli Země) do okamžiku, kdy jsme zaslechli šplouchnutí, spočítat hloubku studně h? Pokud zanedbáme odpor prostředí a rychlost šíření zvuku pak jde o triviální výpočet. Sice dospějeme k značně odlišnému výsledku od skutečnosti, ale výpočet je jednoduchý. Dokonce i v případě uvažování rychlosti šíření zvuku jde o jednoduchý výpočet. Komplikace nastávají v případě uvažování odporu prostředí a to ze dvou důvodů - jednak musíme řešit nelineární diferenciální rovnici a jednak potřebujeme znát fyzikální konstantu - součinitel odporu vzduchu c_x pro použitý předmět vržený do studny. S prvním problémem nám

[45] vyjímky, které s sebou ocelové kuličky nenosí, mohou do studny plivnout. Tuto variantu obzvláště nedoporučuji v případě, že se někde v okolí vyskytuje kombinace cedule s nápisem Ochranné pásmo vodního zdroje a hlídač. Nemuseli byste mít dostatek času na dokončení experimentu.

pomůže MATLAB a pro určení součinitele odporu provedeme referenční experiment[46], který vyhodnotíme opět v MATLABu.



Obrázek 9-2 Schéma pro příklad Hloubka studně

Řešení zahájíme opět aplikací základního pravidla - nakreslíme obrázek s označením viz obr. 9-2. V tomto případě je obrázek jednoduchý. Zavedeme pouze konvenci, že dráha padající kuličky s bude uvažována s kladnou hodnotou.

Jednoduchou rovnici popisující náš problém získáme na základě následující úvahy. Celkový čas t, který změříme je dán součtem dvou složek. Času t_k, který potřebuje kulička k překonání vzdálenosti od místa puštění k hladině, a času t_z, který potřebuje zvuk k překonání té samé vzdálenosti. Je-li dráha kuličky v závislosti na čase popsána funkcí $s_k(t)$ a dráha zvuku $s_z(t)$, pak můžeme naší úlohu matematicky formulovat jako dvě rovnice

$$s_k(t_k) = s_z(t_z) \qquad t = t_k + t_z$$

Rychlost šíření zvuku ve vzduchu budeme uvažovat jako konstantní s hodnotou $v_z = 330\ ms^{-1}$ tj. dráha s_z, kterou zvuk urazí za čas t je popsána rovnicí $s_z(t) = v_z t$.

Složitější je nalezení druhé funkce $s_k(t)$. Tato funkce je řešením diferenciální rovnice $m\frac{d^2 s_k}{dt^2} = mg$ popisující pohyb tělesa v gravitačním poli. Neuvažujeme-li odpor prostředí jde o triviální řešení a pro případ nulových počátečních podmínek (v(t=0)=0 a s(t=0)=0) je řešení této rovnice

$$s_k(t) = \frac{1}{2} g t^2$$

a celkové řešení úlohy spočívá ve vyřešení kvadratické rovnice (pouze jeden kořen má fyzikální význam) pro určení času t_k (nebo t_z) a pomocí něj i hloubky h

$$t = t_k + t_z \quad \wedge \quad v_z t_z = \frac{1}{2} g t_k^2 \quad \Leftrightarrow \quad v_z(t - t_k) = \frac{1}{2} g t_k^2 \quad \Leftrightarrow \quad \frac{1}{2} g t_k^2 + v_z t_k - v_z t = 0$$

$$t_k = \frac{-v_z \pm \sqrt{v_z^2 + 2 g v_z t}}{g} \qquad h = v_z(t - t_k)$$

Pro případ uvažování odporu vzduchu jde o nelineární diferenciální rovnici, pro kterou analytické řešení neexistuje. Musíme vyjít ze základní rovnice popisující chování reálné kuličky v gravitačním poli. V těchto úlohách klasické mechaniky se vychází z rovnováhy sil - gravitační síly F_g a proti působící síly setrvačné F_s a síly odporu prostředí F_o. Odpor prostředí je úměrný kvadrátu rychlosti. Konstanta úměrnosti tohoto vztahu je tvořena součinem čelní plochy S a součinitele aerodynamického odporu[47] c_x. Pro kuličku lze čelní plochu S i hmotnost m vyjádřit pomocí poloměru r a hustoty ρ.

$$F_s + F_o = F_g \qquad m = \frac{4}{3}\pi r^3 \rho \qquad S = \pi r^2$$

$$m\frac{d^2 s_k}{dt^2} + c_x S\left(\frac{ds_k}{dt}\right)^2 = mg \quad \Leftrightarrow \quad \frac{d^2 s_k}{dt^2} + c_x \frac{S}{m}\left(\frac{ds_k}{dt}\right)^2 = g \quad \Leftrightarrow \quad \frac{d^2 s_k}{dt^2} + \underbrace{c_x \frac{4}{3r\rho}}_{K}\left(\frac{ds_k}{dt}\right)^2 = g$$

---

[46] aby byly experimenty srovnatelné, je vhodné vrhat tentýž předmět (proto doporučená ocelová kulička). Z tohoto důvodu nedoporučuji vrhat přítelkyni. Jednak byste měli problém s provedením referenčního experimentu (nutnost sehnat novou přítelkyni stejných parametrů) a jednak by neměl kdo ocenit vaší schopnost používání MATLABu.

[47] veličina důvěrně známá především fandům automobilů

Tato diferenciální rovnice určuje časový průběh dráhy padající kuličky tj. $s_k(t)$. Řešení je zcela stejné jako v předchozí variantě. Opět hledáme takovou hodnotu t_k, pro kterou je splněna základní rovnice

$$s_k(t_k) = s_z(t_z) \qquad t = t_k + t_z \quad \Leftrightarrow \quad s_k(t_k) - v_z(t - t_k) = 0$$

Tuto rovnice nelze řešit analyticky a je nutné použít numerické řešení. Zde nám pomůžou funkce MATLABu **fzero** a **ode45**. Než ji však začneme řešit je potřeba znát hodnotu součinitele aerodynamického odporu c_x. V případě kuličky by to neměl být až takový problém, ve slušných fyzikálních tabulkách by se měly dát hodnoty součinitele c_x pro základní geometrické tvary nalézt. Ovšem ty jedinci, kteří s sebou kuličky nenosí a vrhli něco jiného budou mít asi smůlu. Tito nešťastníci budou muset provést referenční experiment a koeficient z výsledku dopočítat. Tato úloha je ekvivalentní původní úloze.

Pro řešení v MATLABu potřebujeme pro obě úlohy přepsat původní diferenciální rovnici druhého řádu do soustavy dvou rovnic řádu prvního a zapsat ve formě m-funkce

$$\frac{ds_k}{dt} = v_k \qquad K = c_x \frac{4}{3r\rho}$$

$$\frac{dv_k}{dt} = g - Kv_k^2$$

Protože pro řešení budeme využívat standardní funkci **ode45** s předáváním parametrů do volané funkce, jsou první tři parametry m-funkce pevně dány[48]. Od čtvrtého parametru už význam parametrů závisí na nás. Čtvrtým parametrem bude pro nás koeficient odporu prostředí.

```
function ds=fce_stu(t,s,flag,K)
% uzivatel. fce pro 'Uvod do pouzivani MATLAB'
% diferencialni rov. volneho padu real. telesa
%
%            ds=fce_stu(t,s,flag,K)
%
% s=[sk      ds=[dsk/dt
%    vk]         dvk/dt]

g=9.81;          % gravitacni zrychleni
ds=zeros(2,1); % priprava sloup.vystup.vektoru
ds(1)=s(2);      % prvni derivace je rychlost
ds(2)=g-K*s(2)^2;
```

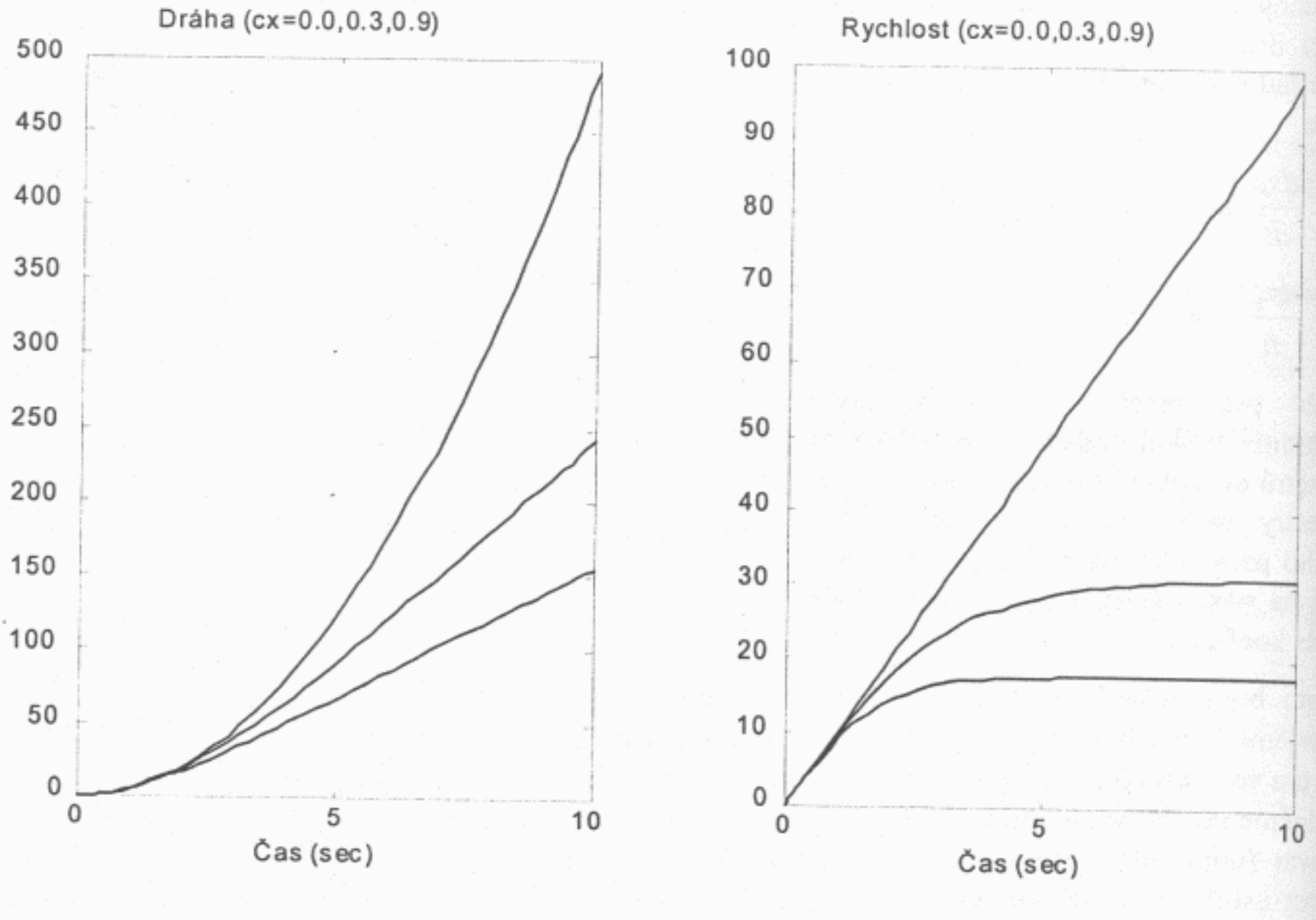
V tomto bodě řešení je vhodné ověřit, zda rovnice i přepis do MATLABu je správný a smysluplný. Nakresleme si jak bude vypadat prvních 10 sec dráhy a rychlosti volného pádu železné kuličky o poloměru r=0.5 cm ve vakuu (c_x=0) a pro dvě hodnoty součinitele aerodynamického odporu (c_x=0.3 a c_x=0.9) a logicky odhadněme jakého výsledku (v hrubých rysech) bychom měli dosáhnout. Ve vakuu se bude rychlost neustále zvyšovat (odpovídá výše uvedenému analytickému řešení), při existenci nějakého odporu prostředí se rychlost ustálí na určité maximální hodnotě, která bude záviset na hodnotě c_x a to nepřímo úměrně. Čím vyšší hodnota c_x tím nižší ustálená rychlost.

```
ro=7800;                %hustota zeleza
r=0.005;                %polomer kulicky
s0=[0;0];               %pocatecni pdminky (poloha a rychlost)
K1=0.0;                 %koeficient odporu pro cx=0
K2=0.3*4/3/r/ro;        %pro cx=0.3
K3=0.9*4/3/r/ro;        %pro cx=0.9
[t1,s1]=ode45('fce_stu',[0;10],s0,[],K1);          %cx=0.0
[t2,s2]=ode45('fce_stu',[0;10],s0,[],K2);          %cx=0.3
[t3,s3]=ode45('fce_stu',[0;10],s0,[],K3);          %cx=0.9
subplot(1,2,1); plot(t1,s1(:,1),t2,s2(:,1),t3,s3(:,1));
```

[48] čas t, který se v našem případě nevyužívá, vektor hodnot s v čase t a příznak typu volání flag

```
title('Dráha (cx=0.0,0.3,0.9)');
xlabel('Čas (sec)'); ylabel('Dráha (m)');
subplot(1,2,2); plot(t1,s1(:,2),t2,s2(:,2),t3,s3(:,2));
title('Rychlost (cx=0.0,0.3,0.9)');
xlabel('Čas (sec)'); ylabel('Rychlost (m/sec)');
```

Na následujícím obrázku 9-3 je vidět, že očekávané chování se potvrdilo. Dále je zřetelné, že maximální rychlosti se pro zvolené hodnoty dosáhne poměrně rychle.



Obrázek 9-3 Volný pád s uvažováním odporu prostředí

Po této kontrole můžeme pokračovat v řešení. Předpokládejme, že pomocí referenčního experimentu jsme naměřili čas t=7.5 sec při hloubce studny 20 m. Tuto vzdálenost urazí zvuk za čas t_z=20/330 sec. Je tedy potřeba nalézt takovou hodnotu koeficientu odporu K, pro kterou za čas 7.5-t_z sec bude mít řešení diferenciální rovnice hodnotu 20 - tedy řešit nulovou hodnotu funkce

$$s_k(K, t - t_z) - h = 0 \quad t = 7.5 \quad h = 20 \quad t_z = \frac{h}{v_z} = \frac{20}{330}$$

Použijeme funkci **fzero**. Musíme vytvořit novou m-funkci (přepsat předchozí rovnici do m-funkce fce_stK), která bude pro funkcí fzero navržené hodnoty K dopočítávat rozdíl řešení diferenciální rovnice a hodnoty 20. Taková m-funkce je uvedena v rámečku (fce_stK). Vlastní řešení - vyhledání hodnoty koeficientu K z dat referenčního experimentu - získáme příkazem

```
function f=fce_stK(K)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% rovnice pro urceni koef. odporu prostredi
%
%                    f=fce_stK(K)
%

tk=7.5-20/330;                  % cas padu
[t,h]=ode45('fce_stu',[0;tk],[0;0],[],K);
f=h(end,1)-20;
```

```
K=fzero('fce_stK',1)

  K =    1.2871
```

Nyní můžeme konečně přikročit k výpočtu vlastní úlohy. Zopakujme tvar základní rovnice, kterou je třeba řešit. Předpokládejme, že naměřený čas byl t=31 sec.

$$s_k(K,t_k) - v_z(t - t_k) = 0$$

$$K = 1.2871 \quad v_z = 330 \quad t = 31$$

Musíme opět vytvořit m-funkci fce_stR, kterou použijeme ve standardní funkci fzero. Vlastní řešení - určení času pádu předmětu t_k z naměřeného času t a koeficientu odporu prostředí K určeného z referenčního experimentu a určení hloubky h - získáme příkazy

```
function f=fce_stR(tk)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% rovnice pro urceni casu tk
%
%                     f=fce_stR(tk)
%

tc=31;              %celkovy cas
vz=330;             %rychlost zvuku
K=1.2871;           %koeficient odporu prostredi
tint=[0;tk];        %casovy interval reseni
[t,h]=ode45('fce_stu',tint,[0;0],[],K);
f=h(end,1)-330*(tc-tk);
```

```
tk=fzero('fce_stR',31)

  tk =    30.7444

h=330*(31-tk)

  h =    84.3395
```

Pro zvolený čas 31 sec a uvedené parametry vyšla hloubka studně 84.3 m. V případě kdyby se neuvažoval odpor vzduchu by vyšla hodnota 2613 metrů.

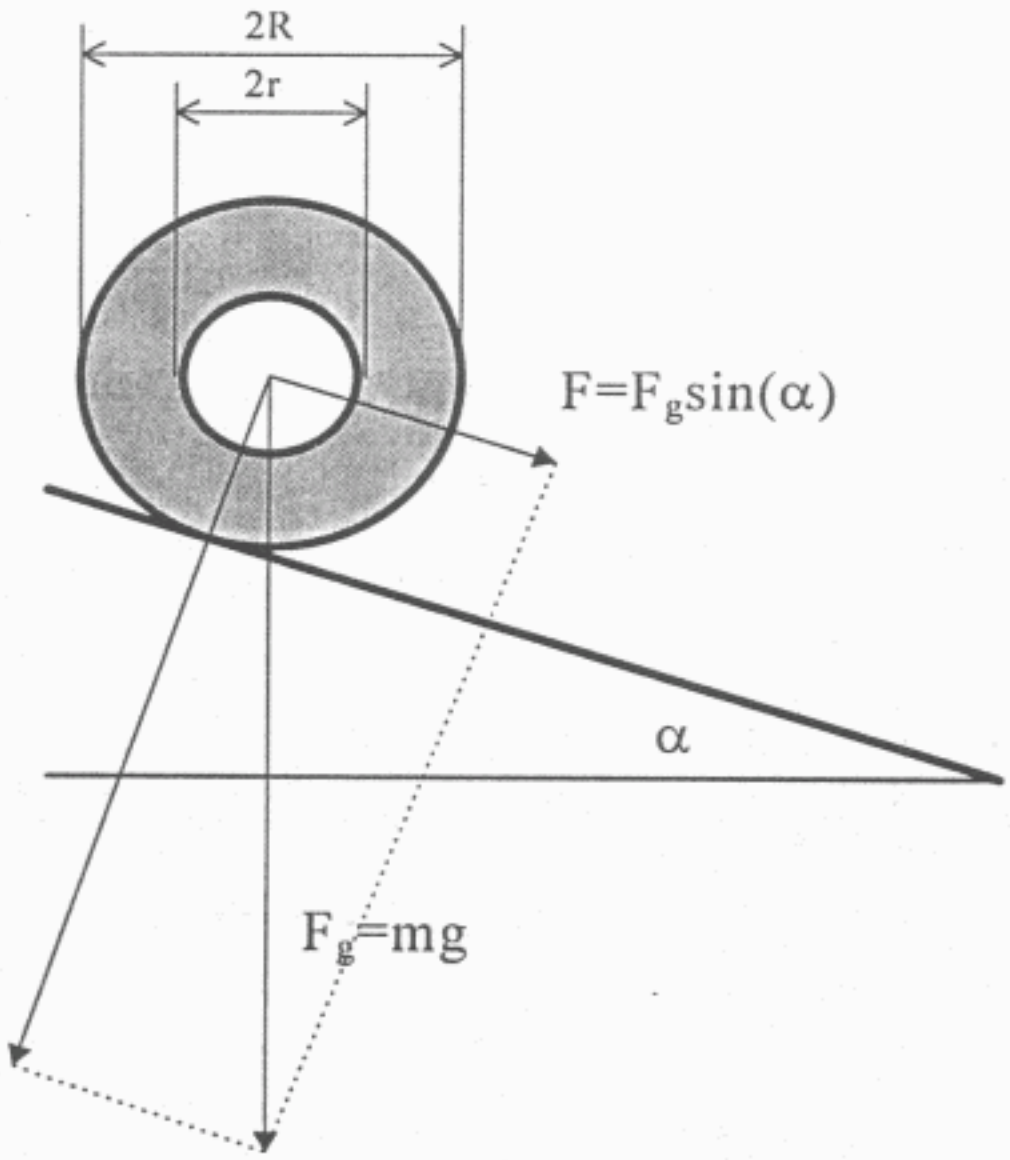
9.3 Dutá koule

Představte si, že máte dvě stejně velké, těžké a stejně vypadající koule. Jedna z nich je dutá a druhá je plná. Vaším úkolem je určit, která z nich je dutá. Nemáte k dispozici žádné speciální vybavení (rentgen atd.) a nesmíte koule poškodit (provrtat atd.). Vymyslet jednoduchou použitelnou metodu pro jednoznačné odlišení není snadné.

Rozlišení je možné podle jedné fyzikální vlastnosti, která musí být u obou koulí různá. Touto vlastností je moment setrvačnosti. Tato veličina závisí na rozložení měrné hmotnosti (hustoty) materiálu vzhledem k ose otáčení. Protože koule jsou jinak stejné (rozměr a váha) musí dutá koule mít větší hustotu materiálu při okraji a tudíž větší moment setrvačnosti.

Jak se změna momentu setrvačnosti projeví ? Postavíme-li obě koule na nakloněnou rovinu a pustíme je, začnou se obě pohybovat. Koule s větším momentem setrvačnosti by se měla opožďovat za plnou koulí.

Mějme tedy dvě koule o poloměru R a hmotnosti m. Jedna z těchto koulí má uprostřed dutinu tvaru soustředné koule o poloměru r. Obě tyto koule umístíme na šikmou plochu o sklonu s úhlem α a pustíme (viz obr. 9-4). Nyní nás bude zajímat časový průběh dráhy obou koulí. Budeme uvažovat pouze odpor prostředí a nikoliv odpor valivého tření[49]. Následující obrázek ukazuje dutou kouli umístěnou na šikmé ploše a rozložení sil vyvolané gravitačním působením. Obě koule se pokouší uvést do pohybu stejná gravitační síla F. Proti působí odpor prostředí F_{op} a síla setrvačná sestávající se ze dvou složek - F_p posuvný pohyb a F_o otáčivý pohyb. Rovnice popisující tyto dvě síly jsou



Obrázek 9-4 Schéma pro příklad Dutá koule

$$F_p = m\frac{d^2 s}{dt^2} \qquad M = RF_o = J\frac{d^2\varphi}{dt^2}$$

kde M je moment síly (síla působící na rameni R), J je moment setrvačnosti vzhledem k ose a φ je úhel otáčení.

Rozbor koeficientu odporu K pro kouli byl uveden v příkladě o měření hloubky studně. Základní rovnici popisující sledovaný systém - rovnováhu sil - lze popsat jako

$$F_{op} + F_p + F_o = F$$

$$K\left(\frac{ds}{dt}\right)^2 + m\frac{d^2 s}{dt^2} + \frac{1}{R}J\frac{d^2\varphi}{dt^2} = mg\sin(\alpha)$$

Souvislost mezi drahou s, kterou koule urazí po nakloněné rovině a úhlem otočení φ je dána vztahem $s=R\varphi$. Pro homogenní tělesa (konstantní hustota ρ) lze moment setrvačnosti vzhledem k ose vyjádřit pomocí objemového momentu setrvačnosti J_v vztahem $J=\rho J_v$. Pro kouli o poloměru R lze v tabulkách nalézt hodnotu J_{vR} a pro naší dutou kouli lze získat vztah J_{vRr}. Hustota materiálu plné koule musí být m/V_R a duté koule m/V_{Rr}.

$$J_{vp} = \frac{8}{15}\pi R^5 \qquad J_{vd} = \frac{8}{15}\pi(R^5 - r^5)$$

$$\rho_p = \frac{3m}{4\pi R^3} \qquad \rho_d = \frac{3m}{4\pi(R^3 - r^3)}$$

$$J_p = \rho_p J_{vp} = \frac{2}{5}mR^2 \qquad J_d = \rho_d J_{vd} = \frac{2m(R^5 - r^5)}{5(R^3 - r^3)}$$

Po dosazení za koeficient odporu K a za moment setrvačnosti J pro plnou a dutou kouli a nahrazení úhlu otáčení φ uraženou drahou s, můžeme základní diferenciální rovnici zapsat jako

[49] Odpor valivého tření je závislý na poloměru a je tedy u obou koulí stejný. Budeme-li sledovat rozdíl dráhy obou koulí, pak tento rozdíl na valivém odporu nezávisí.

plná koule

$$K\left(\frac{ds_p}{dt}\right)^2 + m\frac{d^2s_p}{dt^2} + \frac{1}{R^2}J_p\frac{d^2s_p}{dt^2} = mg\sin(\alpha)$$

$$\Leftrightarrow \quad \frac{7}{5}\frac{d^2s_p}{dt^2} + \frac{K}{m}\left(\frac{ds_p}{dt}\right)^2 = g\sin(\alpha)$$

dutá koule

$$K\left(\frac{ds_d}{dt}\right)^2 + m\frac{d^2s_d}{dt^2} + \frac{1}{R^2}J_d\frac{d^2s_d}{dt^2} = mg\sin(\alpha)$$

$$\Leftrightarrow \quad \left(1+\frac{2(R^5-r^5)}{5R^2(R^3-r^3)}\right)\frac{d^2s_d}{dt^2} + \frac{K}{m}\left(\frac{ds_d}{dt}\right)^2 = g\sin(\alpha)$$

Nyní můžeme zkusit spočítat časový průběh rozdílu drah dvou koulí ve fiktivním experimentu. Mějme dvě koule o průměru 2R=0.1 m a váze m=3 kg. Tyto koule pustíme po šikmé ploše se sklonem φ=15°. Koeficient odporu je u obou koulí stejný a jeho hodnota[50] je K=0.0024. Bude nás zajímat prvních 5 sec pohybu.

Pro řešení v MATLABu musíme přepsat původní diferenciální rovnici druhého řádu na soustavu dvou rovnic řádu prvého a vyjádřit ve formě m-funkce.

$$k_1 = \frac{K}{m} \quad k_2 = g\sin(\alpha)$$

$$k_3 = \frac{7}{5} \quad nebo \quad 1+\frac{2(R^5-r^5)}{5R^2(R^3-r^3)}$$

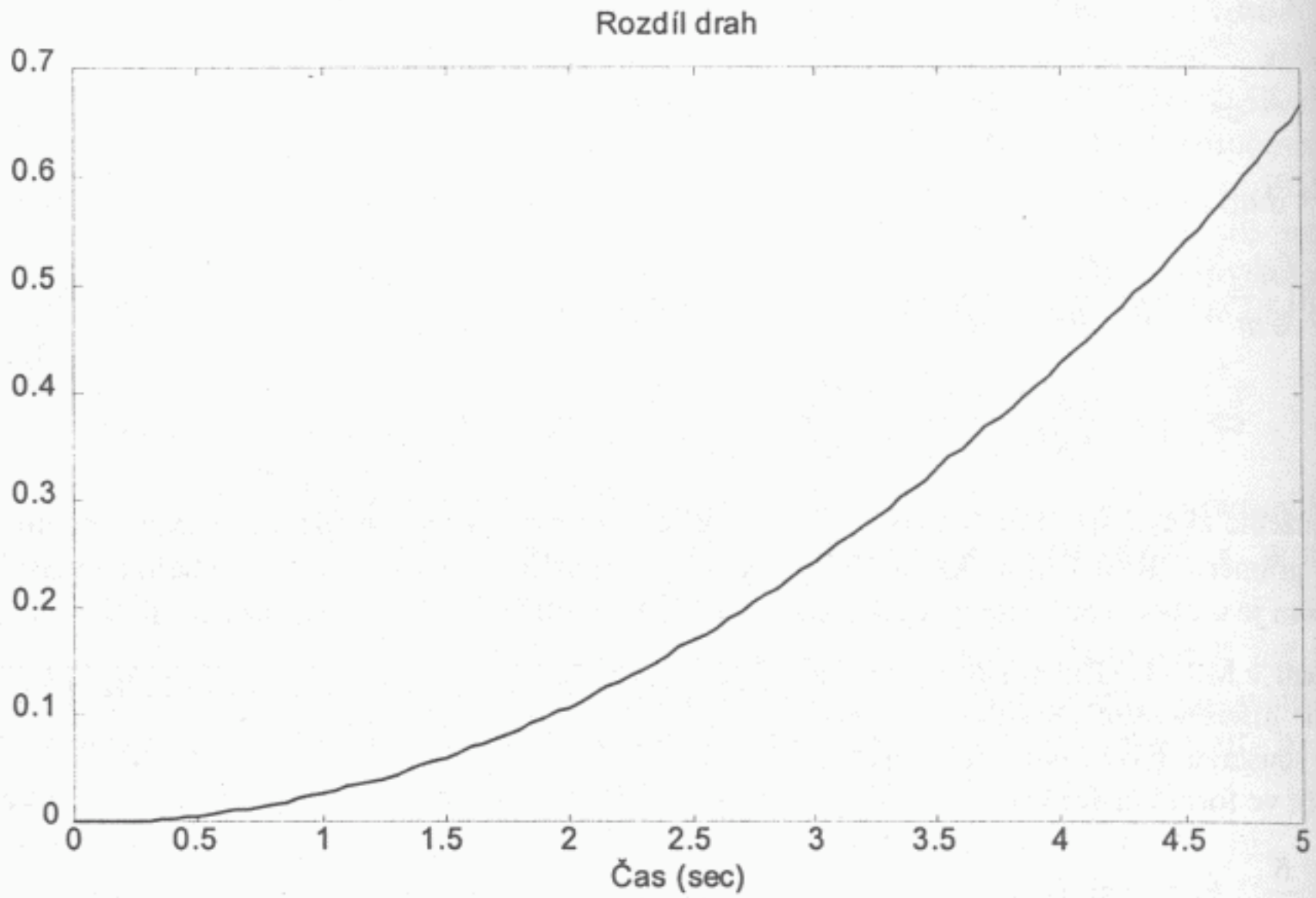
$$\frac{ds}{dt} = v \qquad \frac{dv}{dt} = \frac{k_2 - k_1 v^2}{k_3}$$

```
function ds=fce_kou(t,s,flag,k1,k2,k3)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% diferencialni rov.pohybu koule na sikme plose
%
% ds=fce_kou(t,s,flag,k1,k2,k3)
%
% s=[sk      ds=[dsk/dt
%    vk]         dvk/dt]

ds=zeros(2,1); % priprava sloupc.vystup.vektoru
ds(1)=s(2);    % prvni derivace je rychlost
ds(2)=(k2-k1*s(2)^2)/k3;
```

Vlastní řešení se provede následujícími příkazy včetně vykreslení časového průběhu rozdílu drah obou koulí (viz obr.9-5)

```
R=0.05; r=0.025; cx=0.25; m=3; alfa=2*pi/360*15;      %zadané parametry
K=cx*pi*R^2; k1=K/m; k2=9.81*sin(alfa);               %pomocné proměnné
k3p=7/5; k3d=1+2/5*(R^5-r^5)/R^2/(R^3-r^3);
t=0:0.05:5;                                           %čas, ve kterém nás zajímá výsledek
[tp,sp]=ode45('fce_kou',t,[0;0],[],k1,k2,k3p);        %výpočet pro plnou kouli
[td,sd]=ode45('fce_kou',t,[0;0],[],k1,k2,k3d);        %výpočet pro dutou kouli
plot(t,(sp(:,1)-sd(:,1))); title('Rozdíl drah')
xlabel('Čas (sec)'); ylabel('Vzdálenost (m)')
```

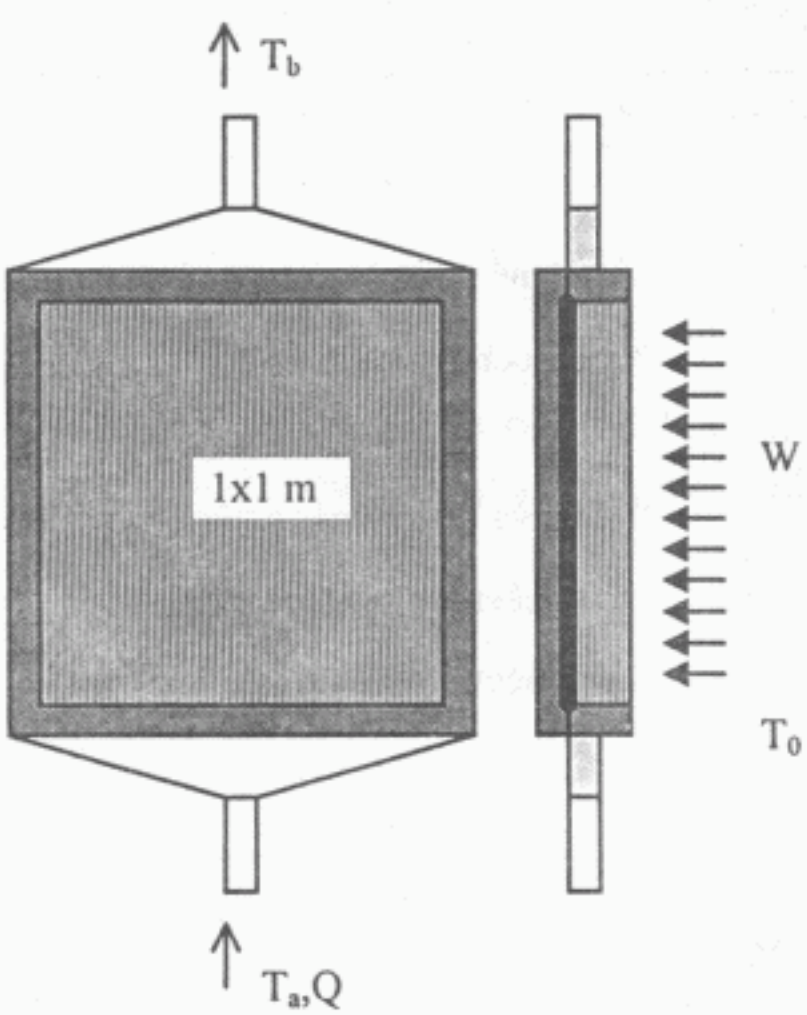
[50] tuto hodnotu získáme z hodnoty součinitele aerodynamického odporu c_x=0.25 a čelní plochy koule



Obrázek 9-5 Rozdíl v poloze duté a plné koule

9.4 Sluneční kolektor

Námět na tuto úlohu vzešel z rozhovoru s jedním známým, který začal vyrábět, kromě jiného, sluneční kolektory. Pravil, že jsou perfektní neboť voda se v nich ohřeje až na 80 °C. Tato informace je sice zajímavá, ale když se nad problémem zamyslíte, naprosto nic nevypovídá o kvalitě vlastního kolektoru. Základní informací by mělo být o kolik stupňů dokáže kolektor o jednotkové ploše ohřát určitý průtok. Ještě lepší by byla informace o účinnosti kolektoru. Jenže pro určení účinnosti je potřeba znát teoretický výkon. A jaký je vlastně teoretický výkon slunečního kolektoru ?

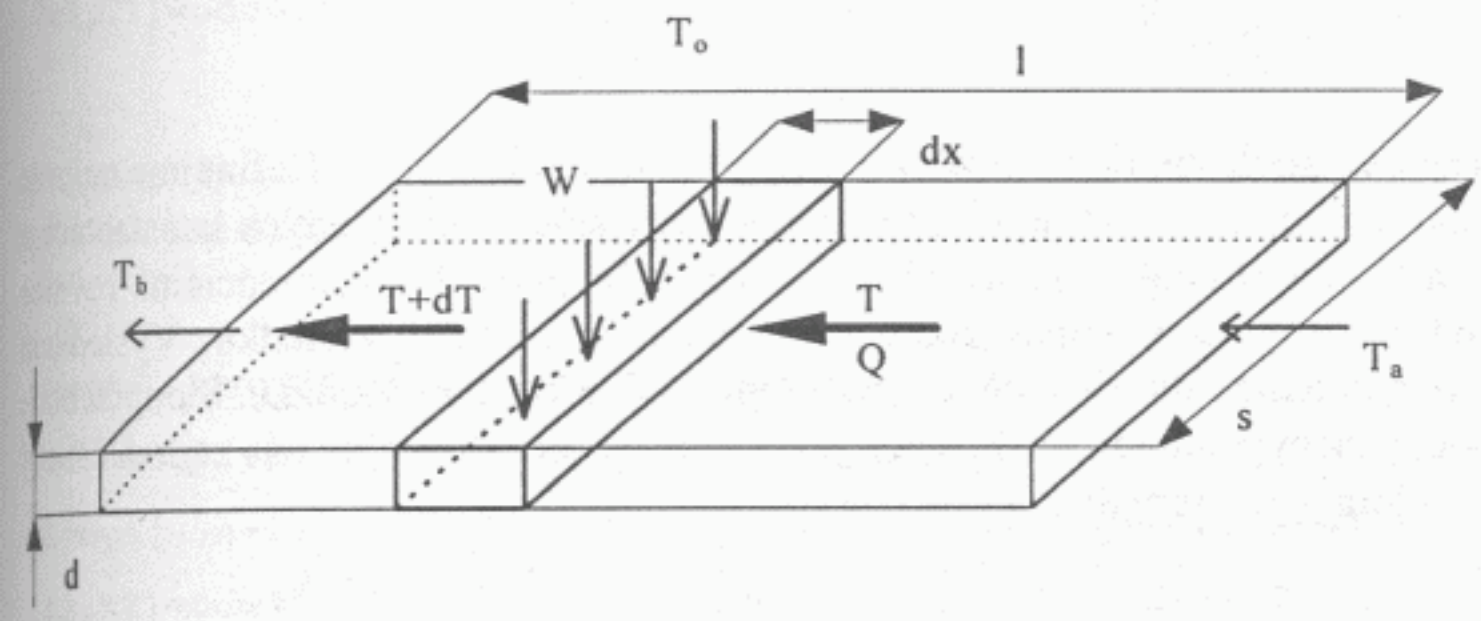


Obrázek 9-6 Schéma 1 pro příklad Sluneční kolektor

Samozřejmě základní veličinou, která tento výkon ovlivňuje, je dopadající výkon slunečního záření. Za ideálních podmínek je asi $W=750\ Wm^{-2}s^{-1}$. Dalším faktorem jsou tepelné ztráty. Část obrácená ke slunci nemůže být izolována na 100% a část tepla (úměrná rozdílu teploty okolí a teploty ohřívané kapaliny) bude unikat. Zkusme namodelovat chování takového ideálního slunečního kolektoru o rozměrech l×s (délka l=1 m a šířka s=1 m) s vrstvou vody o tloušťce d=1 cm (viz obr.9-6). Sluneční záření prochází beze ztráty skleněnou tabulí se zanedbatelnou tepelnou kapacitou a dopadá (skrz vrstvu vody) na černé dno, kde se beze zbytku adsorbuje a přeměňuje v teplo. Toto teplo ohřívá vodu a ta i sklo. Při spodní hraně vstupuje (rovnoměrně rozložená po šířce) studená voda o teplotě T_a, průtoku Q a na

horním konci je teplá voda o teplotě T_b odváděna. Voda se postupně ohřívá od jednoho konce k druhému. Vzhledem k okolní teplotě T_0, dochází na skle k odvodu tepla (s koeficientem prostupu tepla α). Ostatní strany kolektoru jsou ideálně izolovány a k úniku tepla nedochází.

Uvnitř kolektoru v podélném směru (směru proudění) nedochází k promíchávání a ve směru mezi dnem a sklem budeme uvažovat charakteristickou teplotu[51] $\underline{T}$. Protože uvažujeme, že voda do kolektoru vstupuje i vystupuje rovnoměrně podél příčného směru, budeme předpokládat že i teplota v příčném směru je konstantní. Za takto zavedených předpokladů je charakteristická teplota $\underline{T}$ funkcí pouze času a délky $\underline{x}$.



Obrázek 9-7 Schéma II pro příklad Sluneční kolektor

Rovnici popisující chování teploty $\underline{T}$ získáme z tepelné bilance objemu kapaliny o délce dx nakreslené na obrázku 9-7. Musí platit, že součet tepla vstupující je roven součtu tepla vystupujícího a akumulace. Je-li $\underline{Q}$ objemový průtok kapaliny, ρ její hustota a c_m měrná tepelná kapacita kapaliny, pak lze uvedenou bilanci vyjádřit rovnicí

$$Q\rho c_m T + W dS = Q\rho\, c_m (T + dT) + \alpha\, dS(T - T_o) + dV\rho\, c_m \frac{dT}{dt}$$

kde $dS = s\,dx \qquad dV = dS\,d$

$$\frac{W}{d\rho\, c_m} = \frac{Q}{sd}\frac{\partial T}{\partial x} + \frac{\alpha}{d\rho\, c_m}(T - T_o) + \frac{\partial T}{\partial t} \quad \Leftrightarrow \quad \frac{W + \alpha\, T_o}{d\rho\, c_m} = \frac{Q}{sd}\frac{\partial T}{\partial x} + \frac{\alpha}{d\rho\, c_m} T + \frac{\partial T}{\partial t}$$

počáteční a okrajové podmínky: $T(x = 0, t) = T_a \quad T(x, t = 0) = T_a$

Získali jsme jednoduchou parciální diferenciální rovnici, kterou je nyní nutné řešit. Numerických řešení je možné použít několik. My zvolíme nejjednodušší metodu, při které derivaci podle délkové proměnné $\underline{x}$ nahradíme podílem diferencí (kolektor rozdělíme na $\underline{n}$ příčných dílů). Předpokládejme, že teplota uvnitř i-tého dílu je rovná výstupní teplotě T_i a teplota $T_{i=0}=T_a$. Pro zjednodušení dalších zápisů zavedeme substituce a můžeme pro i-tý díl zapsat rovnici

$$a = \frac{W + \alpha\, T_o}{d\rho\, c_m} \quad b = \frac{Q}{sd} \quad c = \frac{\alpha}{d\rho\, c_m} \quad \Delta x = x_i - x_{i-1} = \frac{l}{n} \quad \bar{b} = \frac{b}{\Delta x} = \frac{nQ}{lsd}$$

$$\frac{dT_i}{dt} = a - b\frac{T_i - T_{i-1}}{\Delta x} - cT_i \quad \Leftrightarrow \quad \frac{dT_i}{dt} = a - (\frac{b}{\Delta x} + c)T_i + \frac{b}{\Delta x} T_{i-1}$$

$$\frac{dT_i}{dt} = a - (\bar{b} + c)T_i + \bar{b}\, T_{i-1}$$

Soustavu diferenciálních rovnic, na kterou je nyní řešení parciální diferenciální rovnice převedeno, lze vyjádřit v maticovém tvaru **M*T**+**m**=d**T**/dt, kde **M**(n x n+1), **T**(n+1 x 1) a **m**(n x 1)

---

[51] místo teploty s rozložením v příčném směru budeme uvažovat střední (nebo charakteristickou) teplotu, která zastupuje ve výpočtech teplotu rozloženou

$$\begin{bmatrix} \bar{b} & -(\bar{b}+c) & 0 & 0 & \cdots & 0 & 0 \\ 0 & \bar{b} & -(\bar{b}+c) & 0 & \cdots & 0 & 0 \\ 0 & 0 & \bar{b} & -(\bar{b}+c) & \cdots & 0 & 0 \\ & \vdots & & & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & -(\bar{b}+c) & 0 \\ 0 & 0 & 0 & 0 & \cdots & \bar{b} & -(\bar{b}+c) \end{bmatrix} * \begin{bmatrix} T_a \\ T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{n-1} \\ T_n \end{bmatrix} + \begin{bmatrix} a \\ a \\ a \\ \vdots \\ a \\ a \end{bmatrix} = \begin{bmatrix} \frac{dT_1}{dt} \\ \frac{dT_2}{dt} \\ \frac{dT_3}{dt} \\ \vdots \\ \frac{dT_{n-1}}{dt} \\ \frac{dT_n}{dt} \end{bmatrix}$$

Takto připravenou soustavu diferenciálních rovnic již můžeme v MATLABu přímo řešit. Definujme nejprve hodnoty konstant, potom provedeme výpočty pomocných proměnných závislých na zvolených konstantách a dalších volbách. Volba počtu dílů n ovlivňuje jednak přesnost náhrady parciální diferenciální rovnice soustavou diferenčně-diferenciálních rovnic a jednak rozměry pracovních matic i výsledku. Výsledkem řešení bude matice kde každý sloupec odpovídá časovému vývoji teploty T v každém proužku. Sloupců bude tedy n. Počet řádků odpovídá počtu časových bodů, ve kterých je získáno řešení. Primárně nás zajímá časový vývoj teploty vytékající vody - teplota T v n-tém dílu.

rozměry kolektoru a počet dělení

```
l=1; s=1; d=0.01; n=100;
```

fyzikální konstanty - hustota a měrná tepelná kapacita vody

```
ro=1000; cm=4180;
```

Hodnoty následujících fyzikálních konstant (výkon slunečního záření a koeficient prostupu tepla) jsou diskutabilní. Výkon slunečního záření W je sice znám, ale jeho hodnota v konkrétních podmínkách závisí na zeměpisné šířce, denní době a zataženosti oblohy. Stejně tak koeficient prostupu tepla α je závislý na konstrukčním řešení, rychlosti proudění[52] obou medií a také na teplotě[53]. Proto uvedené hodnoty jsou pouze ilustrační.

```
function dT=kolektor(t,T,flag,M,m,Ta)
% uzivatel.fce pro 'Uvod do pouzivani MATLAB'
% priklad "Slunecni kolektor"
%
%           dT=kolektor(t,T,flag,M,m,Ta)
%


dT=M*[Ta;T]+m;
```

```
alfa=20; W=750;
```
koef.prostupu tepla a výkon slunečního záření

```
To=25; Ta=25;
```
teplota okolí a teplota vstupující vody

```
Q=0.001/60;
```
průtok 1 litr za minutu

pomocné proměnné

```
a=(W+alfa*To)/d/ro/cm; b=n*Q/l/s/d; c=alfa/d/ro/cm;
m=ones(n,1)*a;
M1=eye(n,n+1)*b; M2=[zeros(n,1),eye(n,n)]*(b+c); M=M1-M2;
```

počáteční podmínky (kolektor je naplněn vodou o vstupní teplotě)

---

[52] je zřejmé že bude-li kolcktor ofukován větrem bude ochlazování výraznější

[53] při vyšších teplotách se mění mechanismus předávání tepla - mimo vedení tepla začíná hrát významnější úlohu i odvod tepla sáláním

```
T0=ones(n,1)*Ta;
```

Výpočet provedeme pro dva průtoky Q_1=1 l/min a Q_2=0.05 l/min. Pro průtok Q_1 spočítáme matici teplotního rozložení T_1 pomocí příkazu

výpočet (Q_1) pro časový úsek 0-20 minut po 10 sec. tj. 121 bodů

```
[t1,T1]=ode45('kolektor',[0:10:20*60],T0,[],M,m,Ta);
```

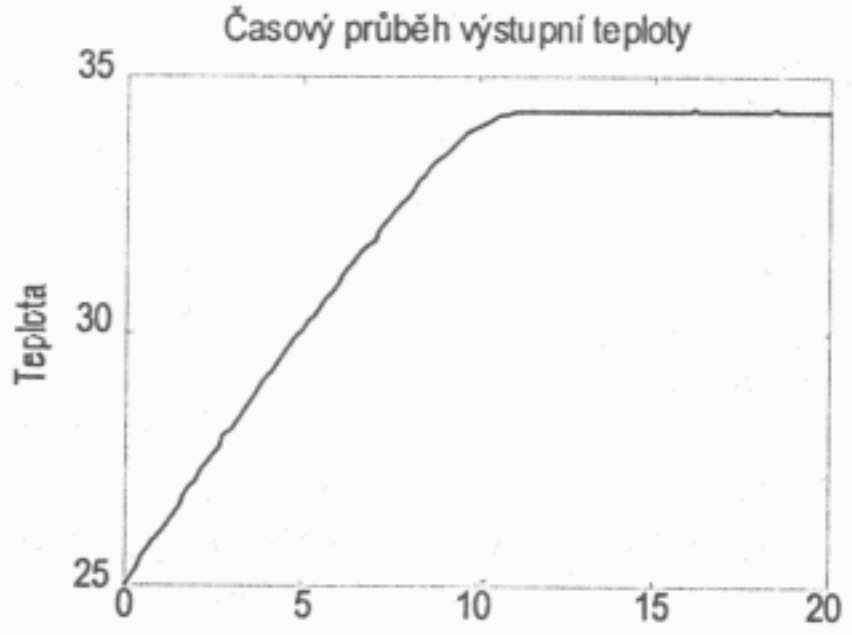
Pro průtok Q_2 je nutné přepočítat parametr M a provést výpočet matice teplotního rozložení T_2 znova. Tentokráte v jiném časovém měřítku.

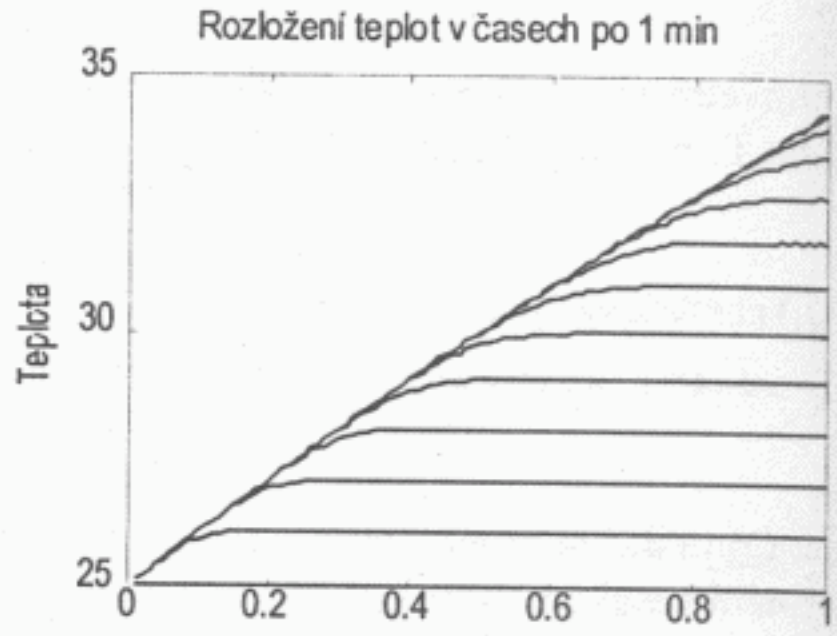
výpočet (Q_1) pro časový úsek 0-180 minut po 60 sec. tj. 181 bodů

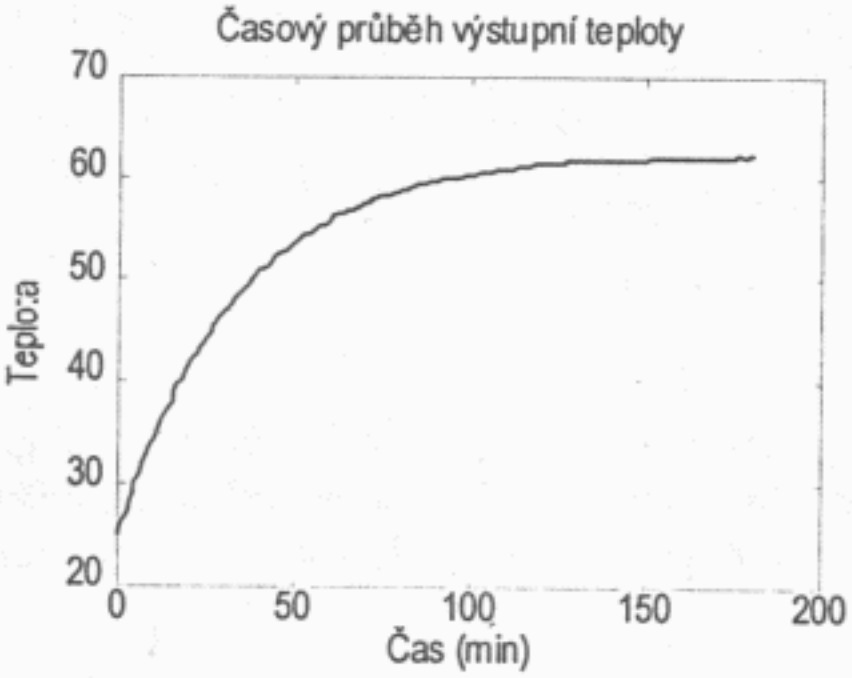
```
Q=0.00005/60; b=n*Q/l/s/d;
M1=eye(n,n+1)*b; M2=[zeros(n,1),eye(n,n)]*(b+c); M=M1-M2;
[t2,T2]=ode45('kolektor',[0:60:3*3600],T0,[],M,m,Ta);
```

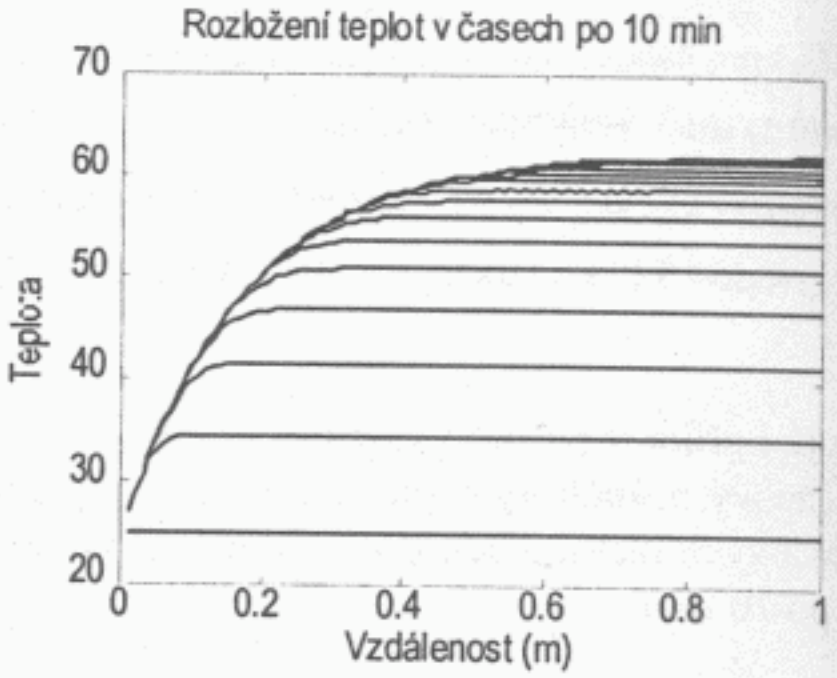
Následující příkazy potom vykreslí výsledky ve formě čtyř grafů do jednoho obrázku (9-8). První graf představuje časový průběh výstupní teploty a druhý graf představuje vývoj teplotního rozložení po délce výměníku v časových okamžicích po 1 minutě. Oba pro průtok Q. Zbývající dva grafy zobrazují totéž pro průtok Q a čas po 10 min.

```
mer1=[1:6:121];          pomocný vektor - každý šestý bod tj. minuta
mer2=[1:10:181];         pomocný vektor - každý desátý bod tj. 10 minut
x=[0.01:0.01:1];         pomocný vektor - poloha míst, kde se sleduje teplota (dílů)
subplot(221);
plot(t1/60,T1(:,100)); title('Časový průběh výstupní teploty');
ylabel('Teplota'); xlabel('Čas (min)');
subplot(222);
plot(x,T1(mer1,:)'); title('Rozložení teplot v časech po 1 min');
ylabel('Teplota'); xlabel('Vzdálenost (m)');
subplot(223);
plot(t2/60,T2(:,100)); title('Časový průběh výstupní teploty');
ylabel('Teplota'); xlabel('Čas (min)');
subplot(224);
plot(x,T2(mer2,:)'); title('Rozložení teplot v časech po 10 min');
ylabel('Teplota'); xlabel('Vzdálenost (m)');
```







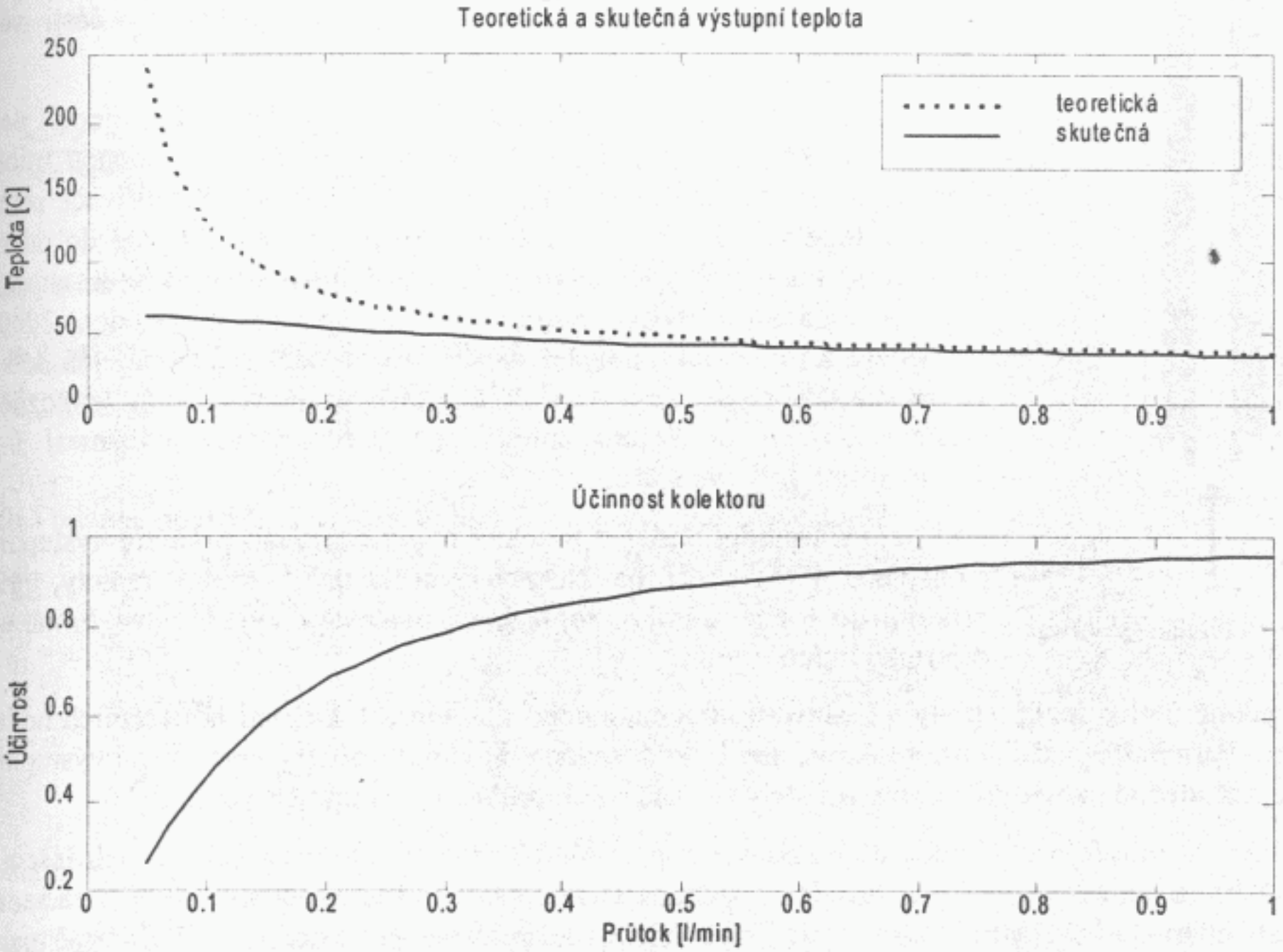


Obrázek 9-8 Průběh výstupní teploty a podélné rozložení teplot v kolektoru

Vraťme se nyní k otázce položené v úvodu tj. jakou účinnost má námi simulovaný kolektor. Odpověď na ní není jednoduchá neboť účinnost (poměr skutečné a teoretické teploty) závisí na výstupní teplotě a výstupní teplota na průtoku. Účinnost je daná především "mírou izolovanosti" vyjádřené v našem případě koeficientem prostupu tepla. Spočítejme tedy nyní závislost účinnosti na průtoku pro náš případ. Pro výpočet ustáleného stavu využijeme výše uvedenou soustavu diferenciálních rovnic - v ustáleném stavu jsou derivace nulové. Řešení potom přejde na řešení soustavy n rovnic o n neznámých. Vlastní řešení v MATLABu je uvedeno ve formě skriptu v rámečku. Výsledkem jsou dva grafy na obr. 9-9. První předsta-

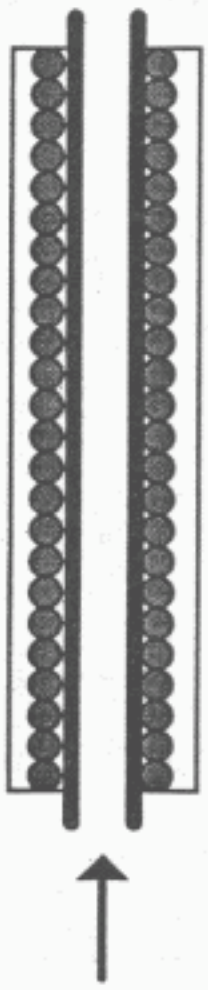
```
N=50;           % počet bodů mezi Q1 a Q2
Q2=0.001/60;    % 1 l/min
Q1=0.00005/60; % 0.05 l/min
Q=Q1:(Q2-Q1)/(N-1):Q2;  % vektor hodnot průtoků
mq=m; T=zeros(N,1); Tt=T;
for i=1:N
   Tt(i)=Ta+W/ro/cm/Q(i);  % teoretická teplota
   b=n*Q(i)/l/s/d;         % přepočet parametru
   mq(1)=m(1)+b*Ta;
   M1=[zeros(1,n);eye(n-1,n)*b];
   M2=eye(n,n)*(b+c); M=M1-M2;
   Tp=M\(-mq);             % řešení ust.tepl.
   T(i)=Tp(n);             % uložení kon. teploty
   uc(i)=T(i)/Tt(i);       % výpočet účinnosti
end
Q=Q*60000;                 % hodnoty v l/min
subplot(211), plot(Q,Tt,':',Q,T)
legend('teoretická','skutečná')
title('Teoretická a skutečná výstupní teplota')
ylabel('Teplota [C]')
subplot(212), plot(Q,uc)
title('Účinnost kolektoru')
xlabel('Průtok [l/min]')
ylabel('Účinnost')
```

vuje závislost teoretické a vypočítané výstupní teploty kolektoru na průtoku v rozmezí 0.05 až 1 l/min. Druhý představuje účinnost počítanou jako podíl spočítané a teoretické ustálené výstupní teploty. Pro průtok 1 l/min je účinnost cca 96% a pro průtok 0.05 l/min je účinnost cca 28%.



Obrázek 9-9 Teoretická teplota a účinnost kolektoru

9.5 Průtokový ohřívač



Obrázek 9-10 Průtokový ohřívač

Tento příklad ukazuje využití MATLABu pro vyhodnocení experimentálních dat. Vzhledem k tomu, že představuje příklad praktického použití MATLABu při řešení konkrétního problému je jeho zadání poněkud složitější a vlastní řešení pracnější. Na tento příklad navazují i příklady v části věnované SIMULINKu.

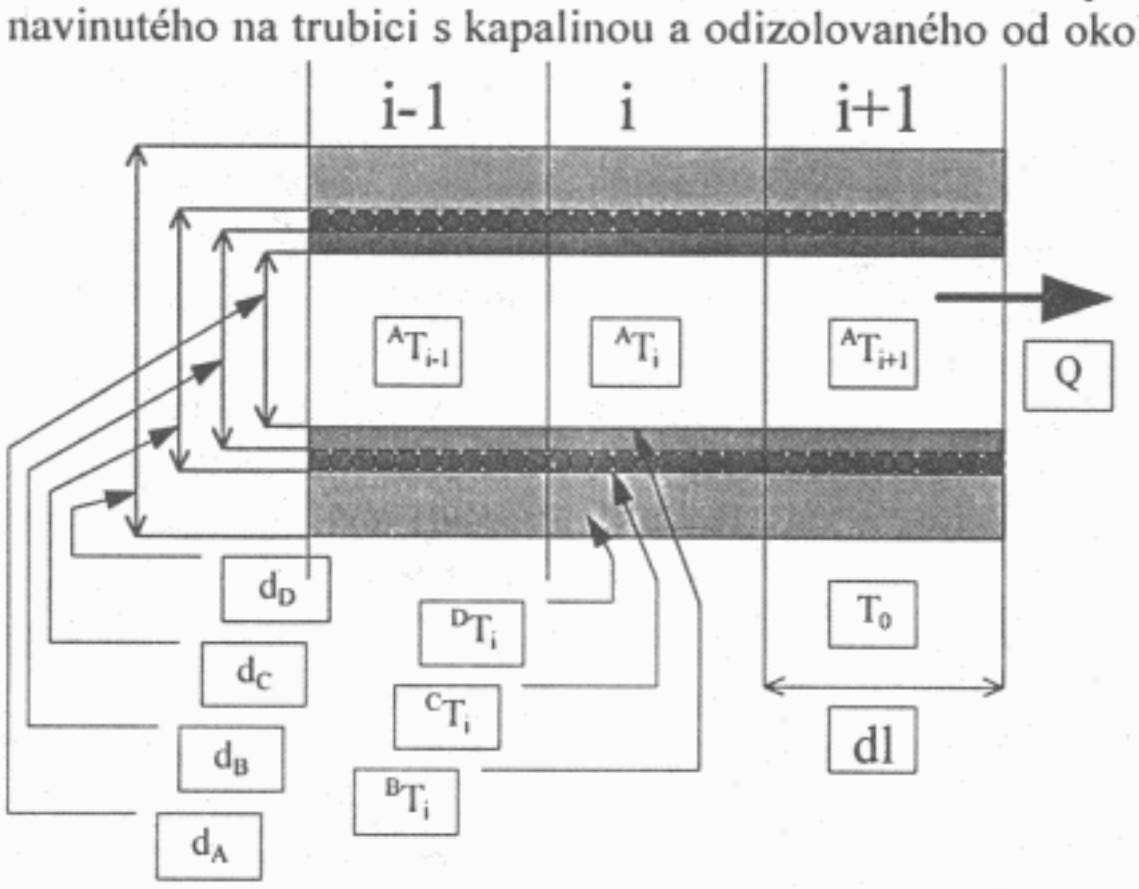
Mějme za úkol nalézt spojitý dynamický matematický model popisující chování konkrétního průtokového ohřívače tvořeného skleněnou trubkou, na které je navinutý odporový drát (viz obr. 9-10). Povrch ohřívače směrem do okolí je tepelně odizolovaný. Síťové napětí 230 V přiváděné do odporového drátu je spínáno polovodičovým relé, které spíná při průchodu napětí nulou. Relé je ovládané šířkově modulovaným signálem s periodou 2000 msec. Vzhledem k frekvenci síťového napětí je nejkratší realizovatelná délka pulsu 10 msec a také další rozšiřování délky pulsu do délky periody je možné s tímto krokem. Příkon do topné spirály lze tedy ovládat v rozmezí 0-100 % maximálního příkonu s krokem 0.5 %.

Pod pojmem model budeme rozumět popis časového průběhu výstupní teploty z ohřívače v závislosti na časových průbězích vstupní teploty, průtoku a příkonu do topné spirály. Popis bývá nejčastěji vyjádřen ve formě soustavy diferenciálních rovnic.

Řešení zadané úlohy spočívá tedy v sestavení matematického modelu a v zjištění konkrétních hodnot jeho parametrů. Parametry je třeba určit takové, aby bylo dosaženo maximální shody vypočítané výstupní teploty z modelu a skutečné naměřené teploty při stejných časových průbězích vstupních veličin.

Korektní řešení tohoto problému vede na soustavu parciálních diferenciálních rovnic. Jejich řešení je však značně složité a nepraktické. Proto zavedeme zjednodušení spočívající v rozdělení ohřívače na sadu N dílů popsaných charakteristickými[54] teplotami. Tímto zjednodušením se převede popis ve formě parciálních diferenciálních rovnic na soustavu diferenčně diferenciálních rovnic. Takou soustavu lze elegantně zapsat v maticovém tvaru. Maticový zápis takovéto soustavy přímo vybízí k použití MATLABu.

Pro vytvoření dynamického matematického modelu průtokového ohřívače s vyhříváním pomocí vinutí navinutého na trubici s kapalinou a odizolovaného od okolí vyjdeme ze schématu na obr. 9-11. Předehřívač délky L je rozdělen na N dílů o délce dl. Radiální i axiální rozložení teplot je nahrazeno charakteristickou teplotou všude stejnou v daném díle. Veškeré odpory v přestupu tepla jsou pak soustředěny do přestupů mezi díly a jsou vyjádřeny koeficienty přestupu tepla α. Potřebné vlastnosti jednotlivých částí jsou hustota ρ a hmotnostní tepelná kapacita c. Pro kapalinu je uvažováno dokonalé míchání - charakteristická teplota je rovná teplotě výstupní. Dále se uvažuje přestup tepla mezi díly kapaliny v axiálním směru.



Obrázek 9-11 Schéma jednoho dílu průtokového ohřívače

Vlastní model i-tého dílu je vytvořen aplikací bilance tepelného toku - součet tepel do dílu vstupujících je rovný součtu tepel z dílu odebíraných + akumulace (změna celkového

[54] charakteristickou teplotu sledovaného dílu zavedeme jako jakousi "střední teplotu" tj. teplotu, která je v celém dílu stejná a charakterizuje díl z hlediska celkového popisu

tepla dílu v čase)

- tepelná bilance kapaliny A

$$\underbrace{Q_i \rho_A c_A {}^{A}T_{i-1} + \alpha_{AB} \pi d_A dl \left({}^{B}T_i - {}^{A}T_i \right)}_{\text{teplo přicházející}} + \underbrace{\alpha_A \, \pi/4 \, d_A^2 \left({}^{A}T_{i+1} - {}^{A}T_i \right)}_{\text{vedení tepla}} =$$

$$= \underbrace{Q_i \rho_A c_A {}^{A}T_i}_{\text{teplo odcházející}} + \underbrace{\alpha_A \, \pi/4 \, d_A^2 \left({}^{A}T_i - {}^{A}T_{i-1} \right)}_{\text{vedení tepla}} + \underbrace{\pi/4 \, d_A^2 dl \, \rho_A c_A \frac{d\,{}^{A}T_i}{dt}}_{\text{akumulace}} \qquad \left[kJ\, s^{-1} \right]$$

- tepelná bilance trubky předehříváku B

$$\underbrace{\alpha_{BC} \pi d_B dl \left({}^{C}T_i - {}^{B}T_i \right)}_{\text{teplo přicházející}} = \underbrace{\alpha_{AB} \pi d_A dl \left({}^{B}T_i - {}^{A}T_i \right)}_{\text{teplo odcházející}} + \underbrace{\pi/4 \left(d_B^2 - d_A^2 \right) dl \, \rho_B c_B \frac{d\,{}^{B}T_i}{dt}}_{\text{akumulace}} \quad \left[kJ\, s^{-1} \right]$$

- tepelná bilance vinutí C - celkový výkon W, výkon v jednom dílu E_i

$$\underbrace{E_i}_{\text{teplo přicházející}} = \underbrace{\alpha_{CD} \pi d_C dl \left({}^{C}T_i - {}^{D}T_i \right)_i + \alpha_{BC} \pi d_B dl \left({}^{C}T_i - {}^{B}T_i \right)}_{\text{teplo odcházející}} +$$

$$+ \underbrace{\pi/4 \left(d_C^2 - d_B^2 \right) dl \, \rho_C c_C \frac{d\,{}^{C}T_i}{dt}}_{\text{akumulace}} \qquad \left[kJ\, s^{-1} \right]$$

- tepelná bilance izolace D

$$\underbrace{\alpha_{CD} \pi d_C dl \left({}^{C}T_i - {}^{D}T_i \right)}_{\text{teplo přicházející}} = \underbrace{\alpha_D \pi d_D dl \left({}^{D}T_i - T_0 \right)}_{\text{teplo odcházející}} + \underbrace{\pi/4 \left(d_D^2 - d_C^2 \right) dl \, \rho_D c_D \frac{d\,{}^{D}T_i}{dt}}_{\text{akumulace}} \quad \left[kJ\, s^{-1} \right]$$

Zavedeme-li následující substituce pomocí vztahů

$$A_1 = \frac{4}{\pi d_A^2 dl} \qquad A_2 = \frac{1}{\rho_A c_A dl} \qquad A_3 = \frac{4}{d_A \rho_A c_A}$$

$$B_1 = \frac{4 d_B}{\left(d_B^2 - d_A^2 \right)} \qquad B_2 = \frac{4 d_A}{\left(d_B^2 - d_A^2 \right)}$$

$$C_1 = \frac{4}{\pi \left(d_C^2 - d_B^2 \right) dl} \qquad C_2 = \frac{4 d_C}{\left(d_C^2 - d_B^2 \right)} \qquad C_3 = \frac{4 d_B}{\left(d_C^2 - d_B^2 \right)}$$

$$D_1 = \frac{4 d_C}{\left(d_D^2 - d_C^2 \right)} \qquad D_2 = \frac{4 d_D}{\left(d_D^2 - d_C^2 \right)}$$

dostaneme soustavu rovnic, kde jsou odděleny známé a neznámé hodnoty parametrů

$$\frac{d^{A}\mathbf{T}_i}{dt} = A_1\mathbf{Q}\left({}^{A}T_{i-1} - {}^{A}\mathbf{T}_i\right) + \alpha_A A_2\left({}^{A}T_{i+1} - 2{}^{A}\mathbf{T}_i + {}^{A}T_{i-1}\right) + \alpha_{AB} A_3\left({}^{B}T_i - {}^{A}\mathbf{T}_i\right)$$

$$\frac{d^{B}T_i}{dt} = \frac{\alpha_{BC} B_1\left({}^{C}T_i - {}^{B}T_i\right) - \alpha_{AB} B_2\left({}^{B}T_i - {}^{A}\mathbf{T}_i\right)}{\underbrace{\rho_B c_B}_{P_B}}$$

$$\frac{d^{C}T_i}{dt} = \frac{C_1\mathbf{E}_i - \alpha_{CD} C_2\left({}^{C}T_i - {}^{D}T_i\right) - \alpha_{BC} C_3\left({}^{C}T_i - {}^{B}T_i\right)}{\underbrace{\rho_C c_C}_{P_C}}$$

$$\frac{d^{D}T_i}{dt} = \frac{\alpha_{CD} D_1\left({}^{C}T_i - {}^{D}T_i\right) - \alpha_D D_2\left({}^{D}T_i - T_0\right)}{\underbrace{\rho_D c_D}_{P_D}}$$

Parametry $A_1, A_2, A_3, B_1, B_2, C_1, C_2, C_3, D_1$ a D_2 jsou dány známými fyzikálními vlastnostmi a geometrickým uspořádáním. Zbývajících osm parametrů α_A, α_{AB}, α_{BC}, α_{CD}, α_D, P_B, P_C, a P_D je potřeba určit z experimentálních dat.

Pro vlastní výpočet je vhodné převést soustavu 4 rovnic popisujících i-tý díl do maticového vyjádření. Prvním krokem je sloučení proměnných a parametrů

$$\frac{d^{A}T_i}{dt} = \underbrace{\left(A_1 Q + \alpha_A A_2\right)}_{a_1}{}^{A}T_{i-1} + \underbrace{-\left(A_1 Q + 2\alpha_A A_2 + \alpha_{AB} A_3\right)}_{a}{}^{A}T_i + \underbrace{\alpha_A A_2}_{a_2}{}^{A}T_{i+1} + \underbrace{\alpha_{AB} A_3}_{a_B}{}^{B}T_i$$

$$\frac{d^{B}T_i}{dt} = \underbrace{\frac{\alpha_{AB} B_2}{P_B}}_{b_A}{}^{A}T_i + \underbrace{-\frac{\alpha_{BC} B_1 + \alpha_{AB} B_2}{P_B}}_{b}{}^{B}T_i + \underbrace{\frac{\alpha_{BC} B_1}{P_B}}_{b_C}{}^{C}T_i$$

$$\frac{d^{C}T_i}{dt} = \underbrace{\frac{\alpha_{BC} C_3}{P_C}}_{c_B}{}^{B}T_i + \underbrace{-\frac{\alpha_{CD} C_2 + \alpha_{BC} C_3}{P_C}}_{c}{}^{C}T_i + \underbrace{\frac{\alpha_{CD} C_2}{P_C}}_{c_D}{}^{D}T_i + \underbrace{\frac{C_1}{P_C}}_{c_E}E_i$$

$$\frac{d^{D}T_i}{dt} = \underbrace{\frac{\alpha_{CD} D_1}{P_D}}_{d_C}{}^{C}T_i + \underbrace{-\frac{\alpha_{CD} D_1 + \alpha_D D_2}{P_D}}_{d}{}^{D}T_i + \underbrace{\frac{\alpha_D D_2}{P_D}}_{d_T}T_0$$

Model popisující celý průtokový ohřívač je pak popsán následující maticovou rovnicí

$$\mathbf{M}\times\mathbf{T}+\mathbf{p}=\frac{d\mathbf{T}}{dt}$$

$$\begin{bmatrix} a & a_B & 0 & 0 \\ b_A & b & b_C & 0 \\ 0 & c_B & c & c_D \\ 0 & 0 & d_C & d \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} a & a_B & 0 & 0 & a_2 & 0 & 0 & 0 \\ b_A & b & b_C & 0 & 0 & 0 & 0 & 0 \\ 0 & c_B & c & c_D & 0 & 0 & 0 & 0 \\ 0 & 0 & d_C & d & 0 & 0 & 0 & 0 \\ a_1 & 0 & 0 & 0 & a & a_B & 0 & 0 \\ 0 & 0 & 0 & 0 & b_A & b & b_C & 0 \\ 0 & 0 & 0 & 0 & 0 & c_B & c & c_D \\ 0 & 0 & 0 & 0 & 0 & 0 & d_C & d \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a & a_B & 0 & 0 \\ b_A & b & b_C & 0 \\ 0 & c_B & c & c_D \\ 0 & 0 & d_C & d \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a & a_B & 0 & 0 \\ b_A & b & b_C & 0 \\ 0 & c_B & c & c_D \\ 0 & 0 & d_C & d \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} a+a_2 & a_B & 0 & 0 \\ b_A & b & b_C & 0 \\ 0 & c_B & c & c_D \\ 0 & 0 & d_C & d \end{bmatrix}$$

$$* \begin{bmatrix} {}^{A}T_1 \\ {}^{B}T_1 \\ {}^{C}T_1 \\ {}^{D}T_1 \\ \hline {}^{A}T_{i-1} \\ {}^{B}T_{i-1} \\ {}^{C}T_{i-1} \\ {}^{D}T_{i-1} \\ {}^{A}T_i \\ {}^{B}T_i \\ {}^{C}T_i \\ {}^{D}T_i \\ {}^{A}T_{i+1} \\ {}^{B}T_{i+1} \\ {}^{C}T_{i+1} \\ {}^{D}T_{i+1} \\ \hline {}^{A}T_N \\ {}^{B}T_N \\ {}^{C}T_N \\ {}^{D}T_N \end{bmatrix} + \begin{bmatrix} a_1\,{}^{A}T_0 \\ 0 \\ c_E E_i \\ \frac{d_T T_0}{0} \\ 0 \\ c_E E_i \\ d_T T_0 \\ 0 \\ 0 \\ c_E E_i \\ d_T T_0 \\ 0 \\ 0 \\ c_E E_i \\ \frac{d_T T_0}{0} \\ 0 \\ c_E E_i \\ d_T T_0 \end{bmatrix} = \begin{bmatrix} d\,{}^{A}T_1/dt \\ d\,{}^{B}T_1/dt \\ d\,{}^{C}T_1/dt \\ d\,{}^{D}T_1/dt \\ \hline d\,{}^{A}T_{i-1}/dt \\ d\,{}^{B}T_{i-1}/dt \\ d\,{}^{C}T_{i-1}/dt \\ d\,{}^{D}T_{i-1}/dt \\ d\,{}^{A}T_i/dt \\ d\,{}^{B}T_i/dt \\ d\,{}^{C}T_i/dt \\ d\,{}^{D}T_i/dt \\ d\,{}^{A}T_{i+1}/dt \\ d\,{}^{B}T_{i+1}/dt \\ d\,{}^{C}T_{i+1}/dt \\ d\,{}^{D}T_{i+1}/dt \\ \hline d\,{}^{A}T_N/dt \\ d\,{}^{B}T_N/dt \\ d\,{}^{C}T_N/dt \\ d\,{}^{D}T_N/dt \end{bmatrix}$$

kde **M** diagonální matice s opakujícími se submaticemi stejných koeficientů (hodnoty některých koeficientů jsou závislé na aktuálním průtoku

p sloupcový vektor parametrů zahrnující vliv příkonu topení a okolní teploty (ovlivňují každý díl) a vstupní teploty (ovlivňuje pouze první díl)

T sloupcový vektor neznámých charakteristických teplot sdružených po čtveřicích ke každému dílu

Řešení v MATLABu vychází z příslušné soustavy rovnic a volby datových struktur. Nejprve definujme datové struktury. Naměřená data budou v souboru o struktuře (matice Data), kde každý řádek představuje data naměřená v jednom časovém kroku

t [sec] E [0-200] Q [$m^3\ s^{-1}$] Tin [°C] Tout [°C]

V rámci provedených experimentů byla data měřena po 500 msec a doba trvání experimentu byla obvykle 10 - 20 min. Datová matice o pěti sloupcích tedy měla od 1200 do 2400 řádků.

Neznámé parametry (odhady) jsou shrnuty ve vektoru **Px** a známé parametry či konstanty v matici **Pk** jak je ukázáno dále. Kromě struktury je i naznačeno pojmenování proměnných v rovnicích (řecká písmena) a ve vztazích MATLABu. Symbol $\underline{W}$ označuje celkový maximální příkon, T_o teplotu okolí.

$$P_x = \begin{bmatrix} \alpha_A = aA \\ \alpha_{AB} = aAB \\ \alpha_{BC} = aBC \\ \alpha_{CD} = aCD \\ \alpha_D = aD \\ \rho_B c_B = Pb \\ \rho_C c_C = Pc \\ \rho_D c_D = Pd \end{bmatrix} \quad P_k = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & 0 \\ C_1 & C_2 & C_3 \\ D_1 & D_2 & 0 \\ N & T_0 & W \end{bmatrix}$$

Hodnoty konstant a počáteční odhady parametrů přepočteme ze zvolených geometrických rozměrů a známých fyzikálních konstant pomocí skriptu, který je nazván **ohrivacD.m**.

ohrivacD.m

```
N=10;                          % pocet dilu
T0=21;                         % teplota okoli
W=230^2/83;                    % celkovy vykon v W
% rozmery ohrivace
l=0.3;                         % delka vinuti
dA=0.0115;  dB=0.0153;         % vnitrni a vnejsi prumer trubky
dC=0.0163;  dD=0.040;          % vnejsi prumer vinuti a izolace
dl=l/N;                        % delka i-teho dilu
% fyzikalni konstanty
roA=1000;   roB=3600;          % hustota kapaliny/trubky (voda 1000/sklo 3600)
roC=8800;                      % hustota dratu (konstantan 8800)
roD=800;                       % hustota izolace (sypana sadra 800)
aA=50;                         % koeficient prestupu voda-voda
aAB=800;                       % koeficient prestupu voda-sklo
aBC=650;                       % koeficient prestupu vinuti-sklo
aBD=0.02;                      % koeficient prestupu sklo-okoli
aCD=25;                        % koeficient prestupu vinuti-izolace
aD=35;                         % koeficient prestupu izolace-okoli
cA=4180;                       % merna tepelna kapacita kapalina (H20 4.18)
cB=740;                        % merna tepelna kapacita trubka (SiO2 0.74)
cC=452;                        % merna tepelna kapacita vinuti (Fe 0.452)
cD=70;                         % merna tepelna kapacita izolace
% matice konstant Pk
Pk=[4/pi/dA^2/dl,1/dl/roA/cA,4/dA/roA/cA;
   4*dB/(dB^2-dA^2),4*dA/(dB^2-dA^2),0;
   4/pi/(dC^2-dB^2)/dl,4*dC/(dC^2-dB^2),4*dB/(dC^2-dB^2);
   4*dC/(dD^2-dC^2),4*dD/(dD^2-dC^2),0;
    N,T0,W];
% vektor pocatecniho odhadu parametru Px0
Px0=[aA;aAB;aBC;aCD;aD;roB*cB;roC*cC;roD*cD];
% meritko pro optimalizaci
Px0(6)=Px0(6)*1e-6; Px0(7)=Px0(7)*1e-6; Px0(8)=Px0(8)*1e-6;
```

Odhad neznámých parametrů provedeme minimalizací kriteriální funkce **p_ohr_v** ve tvaru

```
Px=fmins('p_ohr_v',Px0,[],[],Pk,Data);
```

Kriteriální funkce **p_ohr_v** počítá hodnotu kritéria jako sumu kvadrátů odchylek výstupních teplot naměřených a vypočtených z modelu na základě aktuálních hodnot parametrů. Tj. funkce **p_ohr_v** musí být ve tvaru

```
Kr=p_ohr_v(Px,Pk,Data);
```

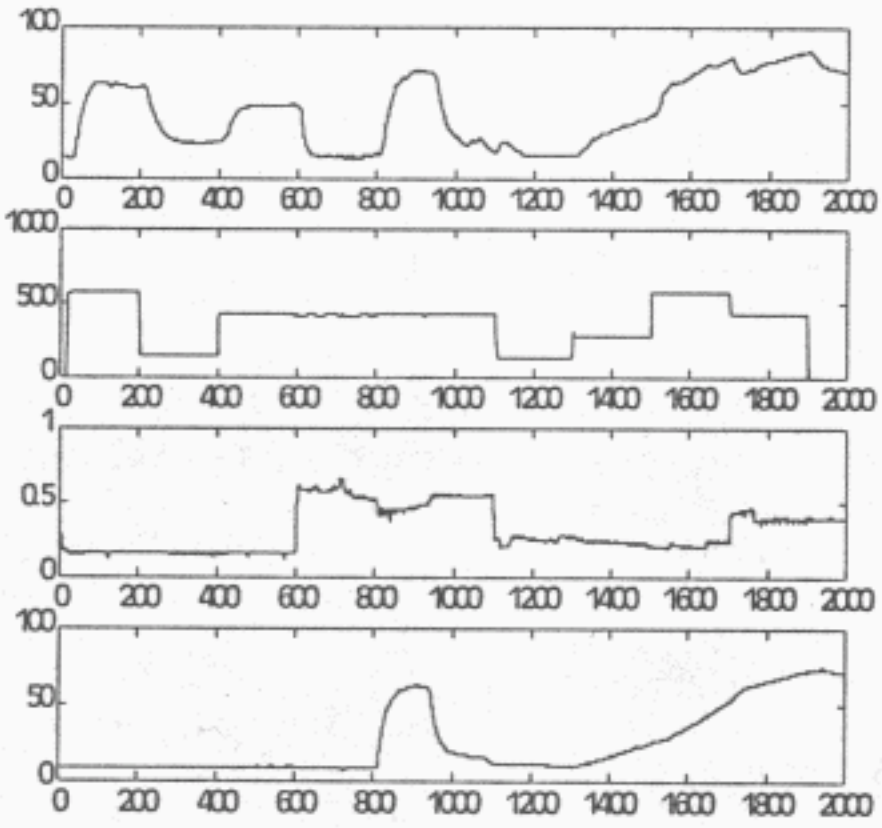
K výpočtu kritéria je tedy potřeba znát průběh výstupní teploty vypočtený z modelu pro příslušné průběhy vstupů a průběh skutečné změřené teploty vyvolané stejnými průběhy vstupů. Tento průběh se získá numerickým řešením soustavy diferenciálních rovnic (např. metodou Runge-Kutta 4.řádu) s konstantním časovým krokem. Situaci komplikuje možná změna příkonu v průběhu jednoho časového kroku (příkon je ovládán šířkově modulovanými pulsy, kdy příkon je buď maximální nebo nulový a mění se doba sepnutí).

Výpočet kroku, ve kterém došlo ke změně je potom rozdělen do dvou částí různě dlouhých podle okamžiku změny příkonu. Pro možnost výpočtu rovnice (průběhy vstupních veličin) i kritéria (měřená výstupní teplota) musí být uvnitř funkce k dispozici měřená data. Funkci **p_ohr_v.m** je pak možné zapsat např. jako

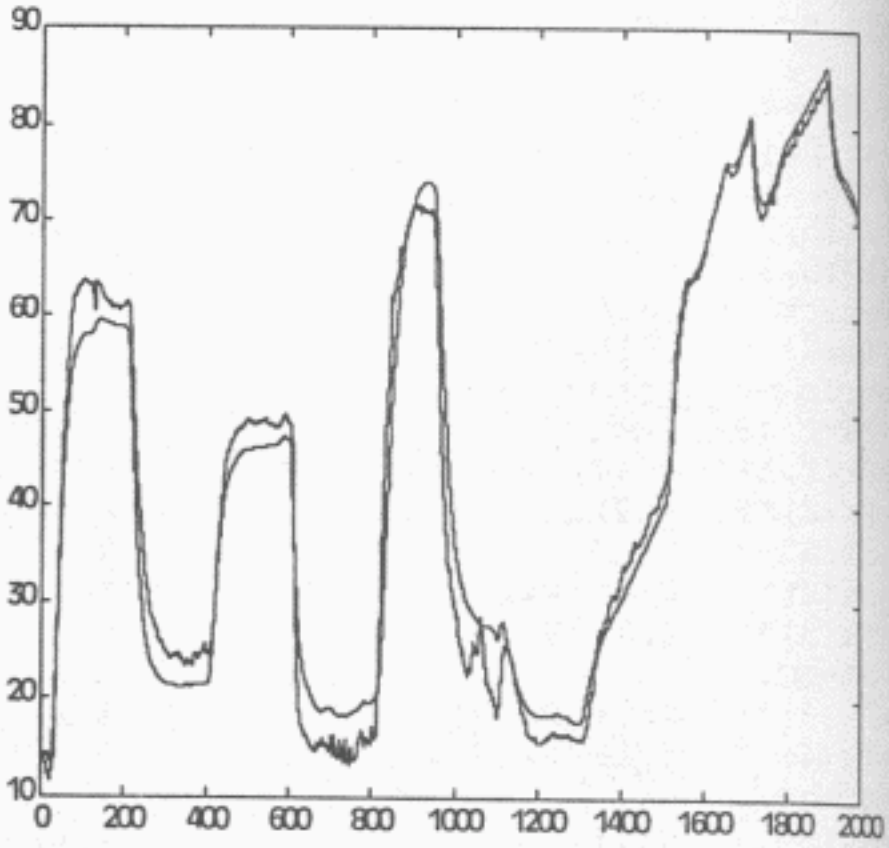
```
function [Kr,Tm]=p_ohr_v(Px,Pk,Data)

% kriterialni funkce - vypocet odchylky modelu od exp. dat
%        [Kr,Tm]=p_ohr_v(Px,Pk,Data)
% viz Prutokovy ohrivac.doc

[nr,ns]=size(Data);
Tm=zeros(nr,4);                          % vektor vystupnich teplot
N=Pk(5,1);                               % pocet dilu
T0=Pk(5,2);                              % okolni teplota
Ei=Pk(5,3)/N;                            % maximalni prikon do jednoho dilu
% meritko Px(6:8)
Px(6)=Px(6)*1e+6; Px(7)=Px(7)*1e+6; Px(8)=Px(8)*1e+6;
% priprava parametru
a2=Px(1)*Pk(1,2);                        % aA*A2
a1=Pk(1,1)*Data(1,3)+a2;                 % A1*Q+aA*A2
aB=Px(2)*Pk(1,3);                        % aAB*A3
a=-a1-a2-aB;                             % -A1*Q-2*aA*A2-aAB*A3
bA=Px(2)*Pk(2,2)/Px(6);                  % aAB*B2/PB
bC=Px(3)*Pk(2,1)/Px(6);                  % aBC*B1/PB
b=-bA-bC;                                % -aAB*B2/PB-aBC*B1/PB
cB=Px(3)*Pk(3,3)/Px(7);                  % aBC*C3/PC
cD=Px(4)*Pk(3,2)/Px(7);                  % aCD*C2/PC
c=-cD-cB;                                % -aCD*C2/PC-aBC*C3/PC
cE=Pk(3,1)/Px(7);                        % C1/PC
dC=Px(4)*Pk(4,1)/Px(8);                  % aCD*D1/PD
dT=Px(5)*Pk(4,2)/Px(8);                  % aD*D2/PD
d=-dC-dT;                                % -aCD*D1/PD-aD*D2/PC
m=[a,aB,0,0;bA,b,bC,0;0,cB,c,cD;0,0,dC,d];
M=zeros(4*N); p=zeros(4*N,1);
T=ones(4*N,1)*Data(1,5);                 % vektor teplot - pocatecni teploty
cykl=4*(N-1)+1;                          % pomocny rozmer
for i=1:4:cykl, M(i:i+3,i:i+3)=m; p(i+3)=dT*T0; end
M(cykl,cykl)=a+a2;
for i=1:4:4*(N-2)+1, M(i,i+4)=a2; M(i+4,i)=a1; end
% vypocet diferencialni rovnice
for k=1:nr,
  % zahrnuti prutoku
  a1=Pk(1,1)*Data(k,3)+a2;               % A1*Q+aA*A2
  a=-a1-a2-aB;                           % -A1*Q-2*aA*A2-aAB*A3
  for i=1:4:4*(N-2)+1, M(i,i)=a; M(i+4,i)=a1;   end    % aktualizace a,a1
  M(cykl,cykl)=a+a2;                     % posledni a na diagonale+konvekce
  p(1)=a1*Data(k,4);                     % zahrnuti vstupni teploty
  % zahrnuti aktualniho vykonu
  E=Data(k,2);   dt=0.5;  Wi=cE*Ei;
  if E<50,              % krok se zmenou prikonu nebo nulovy prikon
    if E==0, Wi=0;      % bez prikonu
    else                % mezivypocet pri zmene prikonu
      dt=0.5*E/50;
      for i=1:4:cykl, p(i+2)=Wi; end   % aktualizace prikonu
      % jeden krok R-K4
      k1=M*T+p; k2=M*(T+k1*dt/2)+p; k3=M*(T+k2*dt/2)+p; k4=M*(T+k3*dt)+p;
      T=T+(k1+2*k2+2*k3+k4)*dt/6;
      dt=0.5-dt; Wi=0;
    end
  end
  % standardni krok vypoctu R-K4
  for i=1:4:cykl, p(i+2)=Wi; end   % aktualizace prikonu
  % jeden krok R-K4
  k1=M*T+p; k2=M*(T+k1*dt/2)+p; k3=M*(T+k2*dt/2)+p; k4=M*(T+k3*dt)+p;
  T=T+(k1+2*k2+2*k3+k4)*dt/6;
  Tm(k,:)=T(cykl:cykl+3)';
end
sys=tf(1,[16.5 1]);                      % dynamika cidla
Tm(:,1)=lsim(sys,Tm(:,1)-Data(1,5),Data(:,1))+Data(1,5);
% vypocet kriteria
Kr=(Tm(:,1)-Data(:,5))'*(Tm(:,1)-Data(:,5))/nr;
```

Na obr. 9-12 jsou naměřené průběhy vstupních veličin v pořadí od shora - výstupní teplota, příkon, průtok a vstupní teplota.

Na obr. 9-13 jsou průběhy měřené výstupní teploty a teploty vypočítané z modelu s parametry nalezenými uvedenou optimalizací.



Obrázek 9-12 Měřené průběhy



Obrázek 9-13 Měřený a vypočtený průběh teploty

9.6 Ochlazování místnosti

Klasickou ukázkou nesmyslného zadání je následující příklad. Má sloužit jako ukázka toho, jak je důležité uvažovat nad zadáním globálně, abychom pracně neřešili nesmyslný, byť jinak zcela korektně zadaný příklad.

Představte si, že máme dokonale tepelně izolovanou místnost s objemem vzduchu $\underline{V}$ [m^3]. V místnosti je ustálená teplota $\underline{T}$ [°C]. Měrná tepelná kapacita vzduchu je $\underline{c}$ [$Jkg^{-1}K^{-1}$] a jeho hustota ρ [kgm^{-3}]. Aby tedy poklesla teplota v místnosti o 1 °C, je nutné odebrat energii (množství tepla) $E = V \times \rho \times c$ [J].

Místnost je úplně prázdná. Pouze uprostřed stojí normální kompresorová lednička o příkonu W_l [W] s otevřenými dveřmi. Účinnost chlazení je $\varepsilon = 60\%$ - chladící výkon (odebírané množství tepla za 1 sec) je tedy $W_l \varepsilon$ [W].

Před ledničkou je elektrický ventilátor s příkonem W_v [W] pro zajištění cirkulace vzduchu v místnosti. Pro zjednodušení nebudeme uvažovat tepelné kapacity ledničky a ventilátoru.

Vaším úkolem je nyní odpovědět na otázku

"Za jak dlouho poklesne teplota v místnosti o 5 °C ?"

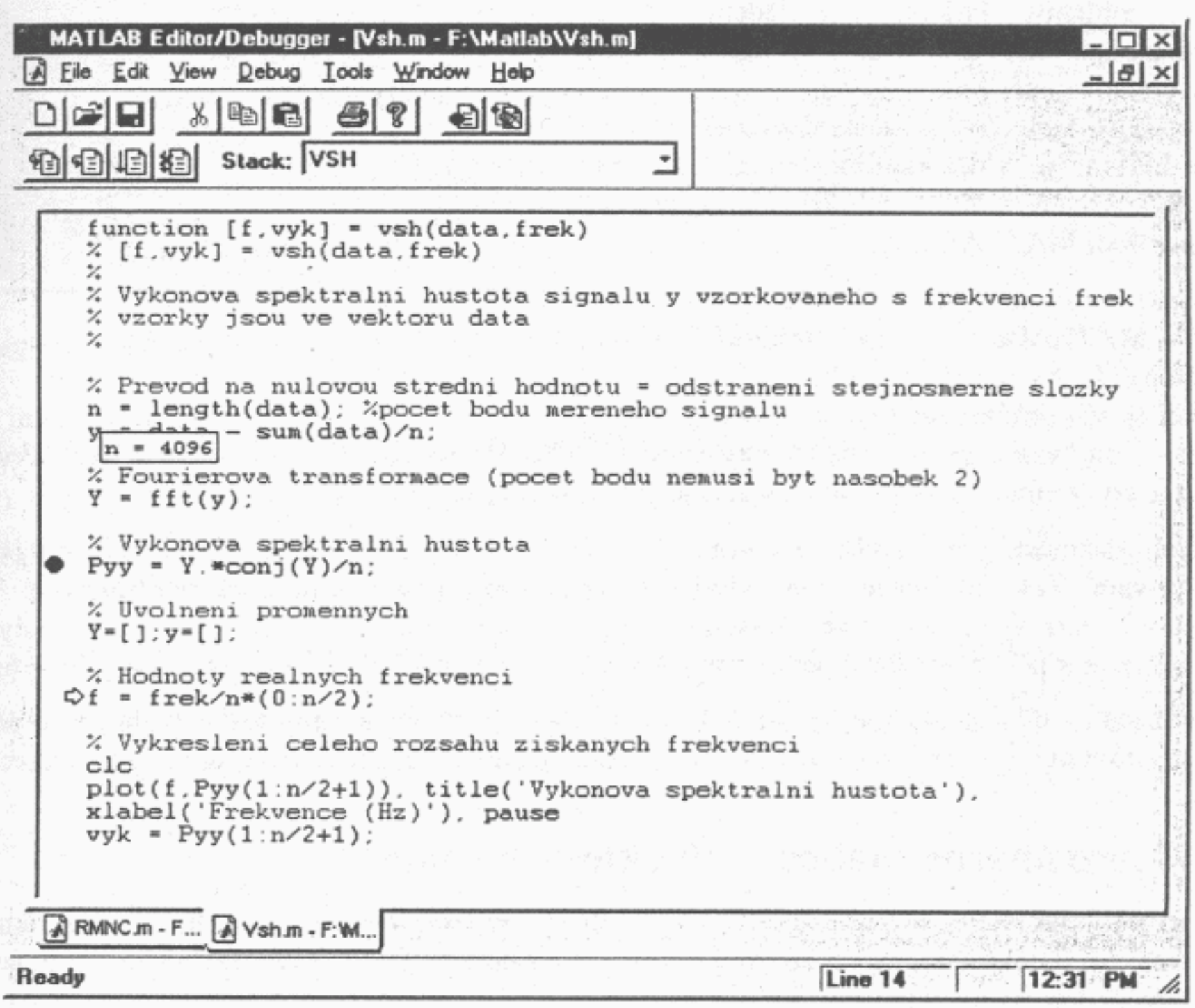
Odpověď bez dalších výpočtů je zřejmá - nikdy. Proč? Lednička sice ochlazuje vnitřní prostor, ovšem odvedené teplo ohřívá vnější tepelný výměník, takže z místnosti se žádné teplo neodvede. Navíc k tomuto procesu je potřeba energii dodávat a veškerá přivedená elektrická energie se mění nakonec v teplo (včetně mechanické energie ventilátoru). Výsledkem je, že teplota v místnosti bude stoupat.

10. Další možnosti MATLABu pod Windows

MATLAB představuje v současné době celý propracovaný systém různých prostředků a lze jej využívat nejen k samotným numerickým výpočtům. V této kapitole se velmi stručně zmíníme o některých z možností, které jsou nabízeny ve verzi 5.2 (Release 11) a něco z verze 5.3 (Release 12). Nejde o ucelený popis, ale o subjektivní výběr několika věcí, o kterých si autor myslí, že by mohly být pro zájemce o MATLAB zajímavé a považuje za vhodné se o nich zmínit. U některých možností je uveden i seznam funkcí, které tuto zmíněnou oblast podporují.

10.1 Editor/debbuger/profiler

Do verze 5.0 se k zápisu a opravě uživatelem vytvářených skriptů a funkcí používal jakýkoli[55] textový editor umožňující uložit čistý ASCII text. Ladění sice bylo systémem MATLAB podporováno, ale pouze na úrovni řádkových příkazů. Uživatel se musel starat[56] hlavně sám. Od verze 5.0 je součástí MATLABu i vlastní editor, který zároveň slouží jako debbuger[57]. Jde na první pohled o poměrně jednoduchý editor (viz obr. 10-1), který ale zahrnuje některé užitečné vlastnosti (zvýrazňuje klíčová slova MATLABu, při déle trvajícím umístění kurzoru myši nad názvem proměnné se otevře okno s hodnotou - hodnota proměnné n v obrázku).



Obrázek 10-1 Okno Editoru/Debbugeru

Užitečné je jeho současné využití pro účely ladění - **debbuger**. Při ladění poskytuje základní funkce od debbugeru požadované:

---

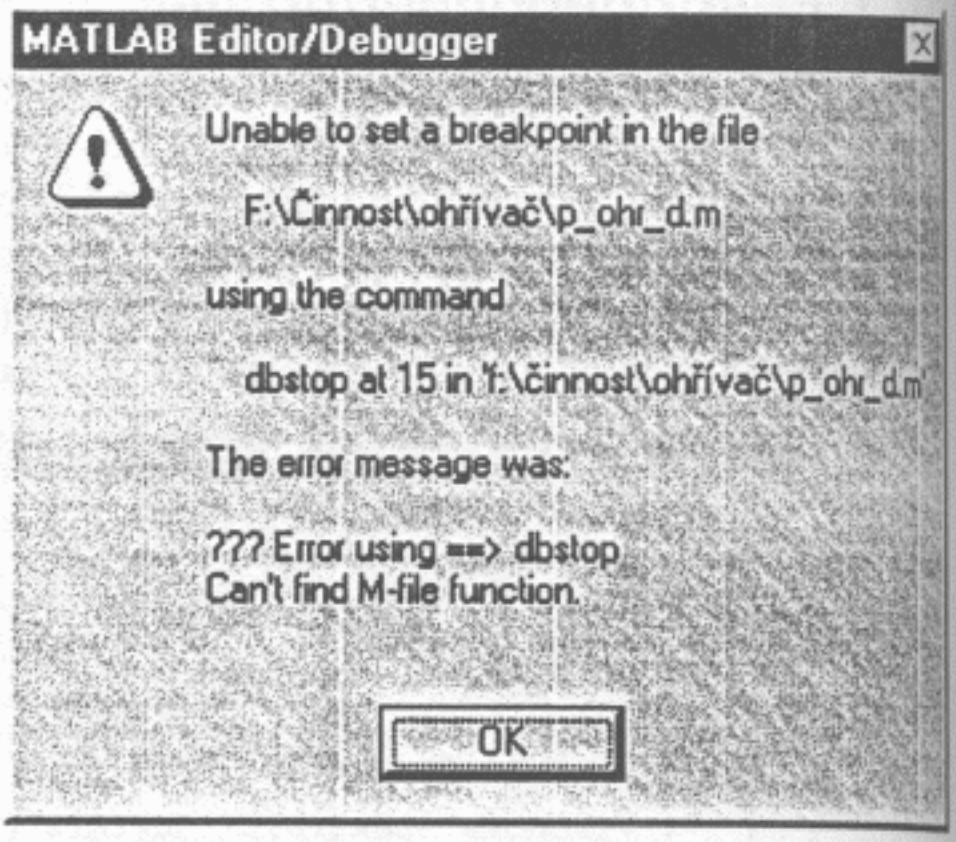
[55] obvykle editor, který je součásti sytému (EDIT pod MS DOS, NOTEPAD pod MS Windows)

[56] pomocný výpis proměnných, přepis ladčné funkce do několika kratších skriptů, využití globálních proměnných atd.

[57] debbuger je speciální prostředek usnadňující ladění vytvořených scriptů a funkcí

- indikaci příkazu, který má být vykonán (žlutá šipka)
- umístění a odebrání bodu zastavení - breakpoint (velká červená tečka, klávesa F12)
- vykonání jednoho příkazu (klávesa F10)
- vnoření se do příkazu obsahujícího volání skriptu či funkce (klávesa F11)
- pokračování ve vykonávání příkazů
- indikaci úrovně vnoření (řádek v záhlaví Stack)
- možnost zastavení provádění příkazů při výskytu chyby, varování či výsledku NaN a Inf

Ladění lze provádět též přímo z prostředí MATLABu pomocí příslušných příkazů (DBCONT, DBSTEP, DBCLEAR, DBTYPE, DBSTACK, DBUP, DBDOWN, DBSTATUS, DBQUIT). Při ladění souborů (m-funkcí) umístěných ve složkách, které obsahují dlouhé názvy či znaky s diakritikou (čeština) nastávají v MATLABu verze 5.2 pod českými Windows problémy. Příkazy pro ladění (ať už používané z okna MATLABu či z debuggeru), které potřebují umístit např. bod zastavení (breakpoint) do souboru hlásí, že tento soubor nemohou nalézt. Příklad takového hlášení je vidět na obr. 10-2. Na tomto obrázku je zároveň vidět i syntaxe ekvivalentního příkazu prostředí MATLABu.



Obrázek 10-2 Informační hlášení debbugeru

V souvislosti s editorem zmiňme ještě jednu novou vlastnost MATLABu a to **lokální funkce** (subfunction). Je možné do jednoho souboru umístit více funkcí tj. více řádků začínajících klíčovým slovem function. Tyto funkce jsou pak lokální vzhledem k souboru - jsou "vidět" pouze z ostatních funkcí v souboru a lze je pouze v nich používat. První funkce v souboru je tzv. primární a pouze tato funkce je přístupná zvenčí.

Další novou vlastností je používání **privátních funkcí**. Jde o m-funkce (soubory) umístěné v podadresáři jménem private. Tyto m-funkce jsou "viditelné" (volatelné) pouze z adresáře nadřazeného. Tento rys dovoluje použít názvy funkcí, které již existují. Je to užitečné při tvorbě vlastních toolboxů, kdy nemusím dávat takový pozor při volbě názvů pomocných funkcí, aby nekolidovaly s již existujícími názvy funkcí.

Kromě možnosti ladění poskytuje MATLAB i prostředky pro vyhledání příkazů a funkcí, kde při průběhu výpočtu se stráví nejvíce času. Jde o tzv. **profiler**. Tato možnost je užitečná při časové optimalizaci řešení.

10.2 Vícerozměrné matice a objektová orientace

Verze 5 přináší principiálně nové možnosti MATLABu. Kromě základních typů - char (string), uint8, double a sparse[58] zavádí datové typy **cell** (cell array) a **struct** (structure array). Zavedení těchto datových struktur souvisí s přechodem na objekty.

Cell array (pole buněk) je datová struktura odpovídající vícerozměrné matici. Vytvořit proměnnou tohoto typu lze příkazem **cell** nebo pomocí složených závorek {}. Např. třírozměrné pole prázdných prvků vytvoříme příkazem

---

[58] tzv. řídká matice tj. matice, která obsahuje významný počet nulových prvků. Tento datový typ používá paměťově úsporný způsob uložení příslušné matice. Všechny funkce jádra lze použít pro operace s řídkými maticemi nebo kombinace řídká a plná matice.

```
C=cell(2,3,4)

  C(:,:,1) =
       []       []       []
       []       []       []
  C(:,:,2) =
       []       []       []
       []       []       []
  C(:,:,3) =
       []       []       []
       []       []       []
  C(:,:,4) =
       []       []       []
       []       []       []
```

nebo s kombinací různých datových typů

```
CC={pi 'Ahoj' magic(3); 5 C 7}

  CC =
      [3.1416]    'Ahoj'           [3x3 double]
      [     5]    {2x3x4 cell}     [          7]
```

Počet operací použitelných pro proměnné tohoto typu je omezený. Navíc záleží na kombinaci datových typů, zda operace je přípustná se všemi prvky.

```
C=ones([3 3 3]);        % vytvoření třírozměrné matice 3x3x3
C=C+4;                  % přičtení konstanty
C=C+C;                  % součet dvou prvků stejného typu
C=C*0.1;                % násobení konstantou
C(2,:,2)=[2 2 2]        % vložení vetoru

  C(:,:,1) =
       1       1       1
       1       1       1
       1       1       1
  C(:,:,2) =
       1       1       1
       2       2       2
       1       1       1
  C(:,:,3) =
       1       1       1
       1       1       1
       1       1       1

sin(CC)

  ??? Function 'sin' not defined for variables of class 'cell'.
```

```
sin(C)
```

```
   ans(:,:,1) =
        0.8415    0.8415    0.8415
        0.8415    0.8415    0.8415
        0.8415    0.8415    0.8415
   ans(:,:,2) =
        0.8415    0.8415    0.8415
        0.9093    0.9093    0.9093
        0.8415    0.8415    0.8415
   ans(:,:,3) =
        0.8415    0.8415    0.8415
        0.8415    0.8415    0.8415
        0.8415    0.8415    0.8415
```

Structure array (pole struktur) odpovídá přímo významu struktury používané v programovacích jazycích. Jednotlivé prvky pole jsou pojmenované. Strukturu lze vytvořit příkazem **struct** nebo naplněním jednotlivých položek. **Pozor!** Proměnné vytvořené příkazem struct (osoba1) a naplněním položek (osoba2) nejsou shodné jak by člověk předpokládal

```
osoba1=struct('jmeno',{'Jan','Novak'},'narozeni',[26,7,1970])
```

```
   osoba1 =
   1x2 struct array with fields:
       jmeno
       narozeni
```

```
osoba2.jmeno={'Jan','Novak'}
```

```
   osoba2 =
       jmeno: {1x2 cell}
```

```
osoba2.narozeni=[26,7,1970]
```

```
   osoba2 =
        jmeno: {1x2 cell  }
     narozeni: [26 7 1970]
```

```
whos
```

```
  Name          Size              Bytes  Class

  C             3x3x3               216  double array
  CC            2x3                 752  cell array
  osoba1        1x2                 496  struct array
  osoba2        1x1                 472  struct array

Grand total is 106 elements using 1936 bytes
```

Přístup k hodnotě měsíce narození je pak realizován různými příkazy podle typu proměnné

```
osoba1(1,2).narozeni(2)
```

```
   ans =     7
```

```
osoba2.narozeni(2)
```

```
   ans =     7
```

Datový typ structure array je využíván v objektech. Objekty a objektový přístup je nová vlastnost MATLABu od verze 5.0. Objekty jsou plně využívány systémem Handle Graphics. Výpočetní jádro MATLABu tuto možnost nabízí, ale zatím standardně nevyužívá. Jedno z prvních standardních využití bylo v Control System Toolboxu verze 4. Zde lze vytvořit tzv. LTI (Linear Time-Invariant) model - objekt kompletně popisující dynamický systém. Vlastní matematický popis chování může být ve formě stavového modelu, přenosové funkce či daný póly a nulami. Objekt dále obsahuje informace o dopravním zpoždění, o tom zda model je spojitý či diskrétní (interval vzorkování), dokonce může obsahovat i názvy vstupů a výstupů. Následující příkazy vytvoří LTI model spojitého systému daného póly, nulami a zesílením s jedním vstupem (označení x), jedním výstupem (označení y) a dopravním zpožděním. Příkaz **set** umožňuje nastavit jednotlivé vlastnosti, příkaz **get** umožňuje získat hodnoty vlastností - uveden bez označení vlastnosti vypíše vše.

```
model=zpk([],[-0.1 -0.2],0.02);
set(model,'Td',0.5);
set(model,'InputName','x','OutputName','y','Notes','spojitý model');
get(model)
```

```
z = {[0x1 double]}
p = {1x1 cell}
k = 0.02
Variable = 's'
Ts = 0
Td = 0.5
InputName = {'x'}
OutputName = {'y'}
Notes = {'spojitý model'}
UserData = []
```

10.3 Grafické možnosti - systém Handle Graphics

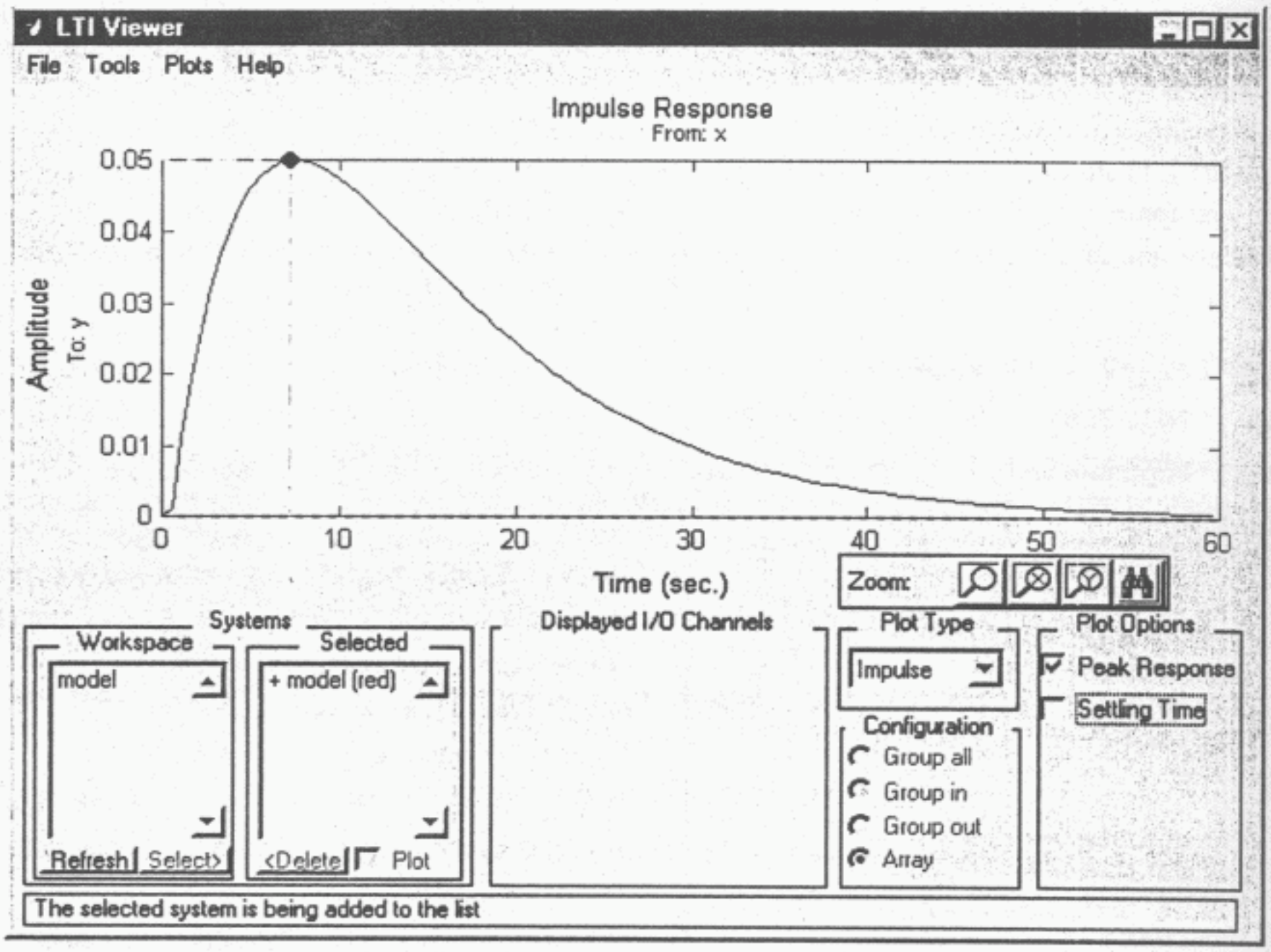
Objektový přístup je využit při práci s grafikou. Dříve zmiňované příkazy pro kreslení (plot, mesh atd.) jsou jen zlomkem možností, které MATLAB pro práci s grafikou a pro tvorbu uživatelského rozhraní poskytuje. Jde o nové přepsání (do objektové podoby) systému Handle Graphics známého již z verze 4 pro Windows. Tento systém umožňuje nastavování všech možných (i nemožných) vlastností grafických objektů i tvorbu uživatelského rozhraní ve stylu programů Windows. Pro aspoň orientační přiblížení možností uveďme dva jednoduché příklady.

První ukazuje skript (v rámečku vpravo), který po spuštění ukazuje pohledy kamery kroužící kolem třírozměrného objektu (koule) a pak pohledy, když se kamera začne pohybovat směrem do středu koule (včetně pohledu, kdy je vidět současně část vnějšího i část vnitřního povrchu kulové plochy).

Druhý příklad (obr. 10-3) ukazuje interaktivní

```
sphere(48)                          % nakresli kulovou plochu
h=findobj('Type','surface');%najdi objekt typu povrch
set(h,'FaceLighting','phong',... % vlastnosti objektu
   'FaceColor','interp',...
   'EdgeColor',[.4 .4 .4],...
   'BackFaceLighting','reverselit')
light('Position',[1 3 2]);          % zdroj svetla c.1
light('Position',[-3 -1 3]);        % zdroj svetla c.2
axis off
material shiny                % nastav odrazove vlastnosti
% vlastnosti objektu camera
set(gca,'CameraPositionMode','manual',...
   'CameraTargetMode','manual',...
   'CameraViewAngleMode','manual');
set(gca,'CameraTarget',[0 0 0]);
set(gca,'CameraPosition',[10 10 10]);
set(gca,'CameraViewAngle',7);
pause(1)
% krouzeni kamery okolo koule
N=15; R=sqrt(2*N^2);
for i=1:N,
   z=10*cos(2*pi/N*i);  % vypocet x,y,z pozice kamery
   x=R*sin(2*pi/N*i+pi/4); y=R*cos(2*pi/N*i+pi/4);
   set(gca,'CameraPosition',[x y z]);
   pause(0.5)
end
% pohyb kamery smerem do stredu koule
for i=1:N,
   p=N/2/(i+1);
   set(gca,'CameraPosition',[p p p]); % nastav pozici
   set(gca,'CameraViewAngle',7+3*i);   % zvetsuj uhel
   pause(0.5)
end
```

uživatelské rozhraní prohlížeče objektů LTI z Control Toolboxu (odpovídající m-file je si možné prohlédnout např. příkazem type **ltiview.m**). Je vidět, že lze používat standardní prvky Windows jako jsou menu, roletové menu, radiobuttom atd.



Obrázek 10-3 Ukázka uživatelského rozhraní - LTI Viewer

10.3.1 Seznam příkazů a funkcí pro práci s grafy a tvorbu uživ. rozhraní

Příkazy a funkce využívané při tvorbě grafů jsou v adresářích matlab\graph2d, matlab\graph3d a matlab\specgraph, pro tvorbu uživatelského rozhraní v adresáři matlab\graphic a matlab\uitools. Jejich seznam lze získat pomocí příslušných příkazů help.

```
help graph2d

  Two dimensional graphs.

  Elementary X-Y graphs.
plot    - Linear plot.
loglog  - Log-log scale plot.
semilogx  - Semi-log scale plot.
semilogy  - Semi-log scale plot.
polar   - Polar coordinate plot.
plotyy  - Graphs with y tick labels on
     the left and right.

  Axis control.
axis    - Control axis scaling and
     appearance.
zoom    - Zoom in and out on a
     2-D plot.
grid    - Grid lines.
box     - Axis box.
hold    - Hold current graph.
axes    - Create axes in arbitrary
     positions.
subplot   - Create axes in tiled
     positions.
daspect   - Data aspect ratio.
pbaspect  - Plot box aspect ratio.
xlim    - X limits.
ylim    - Y limits.

  Graph annotation.
legend  - Graph legend.
title   - Graph title.
xlabel  - X-axis label.
ylabel  - Y-axis label.
text    - Text annotation.
gtext   - Place text with mouse.
plotedit  - Experimental graph
     editing and annotation tools.
```

Hardcopy and printing.

print - Print graph or SIMULINK system; or save graph to M-file.
printopt - Printer defaults.
orient - Set paper orientation.

help graph3d

Three dimensional graphs.

Elementary 3-D plots.

plot3 - Plot lines and points in 3-D space.
mesh - 3-D mesh surface.
surf - 3-D colored surface.
fill3 - Filled 3-D polygons.

Color control.

colormap - Color look-up table.
caxis - Pseudocolor axis scaling.
shading - Color shading mode.
hidden - Mesh hidden line removal mode.
brighten - Brighten or darken color map.

Lighting.

surfl - 3-D shaded surface with lighting.
lighting - Lighting mode.
material - Material reflectance mode.
specular - Specular reflectance.
diffuse - Diffuse reflectance.
surfnorm - Surface normals.

Color maps.

hsv - Hue-saturation-value color map.
hot - Black-red-yellow-white color map.
gray - Linear gray-scale color map.
bone - Gray-scale with tinge of blue color map.
copper - Linear copper-tone color map.
pink - Pastel shades of pink color map.
white - All white color map.
flag - Alternating red, white, blue, and black color map.
lines - Color map with the line colors.
colorcube - Enhanced color-cube color map.
jet - Variant of HSV.
prism - Prism color map.
cool - Shades of cyan and magenta color map.
autumn - Shades of red and yellow color map.
spring - Shades of magenta and yellow color map.
winter - Shades of blue and green color map.
summer - Shades of green and yellow color map.

Axis control.

axis - Control axis scaling and appearance.
zoom - Zoom in and out on a 2-D plot.
grid - Grid lines.
box - Axis box.
hold - Hold current graph.
axes - Create axes in arbitrary positions.
subplot - Create axes in tiled positions.
daspect - Data aspect ratio.
pbaspect - Plot box aspect ratio.
xlim - X limits.
ylim - Y limits.
zlim - Z limits.

Viewpoint control.

view - 3-D graph viewpoint specification.
viewmtx - View transformation matrix.
rotate3d - Interactively rotate view of 3-D plot.

Camera control.

campos - Camera position.
camtarget - Camera target.
camva - Camera view angle.
camup - Camera up vector.
camproj - Camera projection.

High level camera control.

camorbit - Orbit camera.
campan - Pan camera.
camdolly - Dolly camera.
camzoom - Zoom camera.
camroll - Roll camera.
camlookat - Move camera and target to view specified objects.

High level light control.

camlight - Creates or sets position of a light.
lightangle - Spherical position of a light.

Graph annotation.

title - Graph title.
xlabel - X-axis label.
ylabel - Y-axis label.
zlabel - Z-axis label.
colorbar - Display color bar (colorscale).
text - Text annotation.
gtext - Mouse placement of text.
plotedit - Experimental graph editing and annotation tools.

Hardcopy and printing.

```
print   - Print graph or SIMULINK
    system; or save graph to M-file.
printopt  - Printer defaults.
orient    - Set paper orientation.
vrml      - Save graphics to VRML 2.0
file.
```

help specgraph

Specialized graphs.

Specialized 2-D graphs.

```
area    - Filled area plot.
bar     - Bar graph.
barh    - Horizontal bar graph.
bar3    - 3-D bar graph.
bar3h   - Horizontal 3-D bar graph.
comet   - Comet-like trajectory.
errorbar  - Error bar plot.
ezplot  - Easy to use function
    plotter.
feather - Feather plot.
fill    - Filled 2-D polygons.
fplot   - Plot function.
hist    - Histogram.
pareto  - Pareto chart.
pie     - Pie chart.
pie3    - 3-D pie chart.
plotmatrix - Scatter plot matrix.
ribbon  - Draw 2-D lines as ribbons in
    3-D.
scatter - Scatter plot.
stem    - Discrete sequence or "stem"
    plot.
stairs  - Stairstep plot.
```

Contour and 2-1/2 D graphs.

```
contour - Contour plot.
contourf  - Filled contour plot.
contour3  - 3-D Contour plot.
clabel  - Contour plot elevation
    labels.
pcolor  - Pseudocolor (checkerboard)
    plot.
quiver  - Quiver plot.
voronoi - Voronoi diagram.
```

Specialized 3-D graphs.

```
comet3  - 3-D comet-like trajectories.
meshc   - Combination mesh/contour
    plot.
meshz   - 3-D mesh with curtain.
stem3   - 3-D stem plot.
quiver3   - 3-D quiver plot.
scatter3  - 3-D scatter plot.
slice   - Volumetric slice plot.
surfc   - Combination surf/contour
    plot.
trisurf - Triangular surface plot.
trimesh - Triangular mesh plot.
waterfall - Waterfall plot.
```

Images display and file I/O.

```
image   - Display image.
imagesc - Scale data and display as
    image.
colormap  - Color look-up table.
gray    - Linear gray-scale color map.
contrast  - Gray scale color map to
    enhance image contrast.

brighten  - Brighten or darken color
    map.
colorbar  - Display color bar (color
    scale).
imread  - Read image from graphics
    file.
imwrite - Write image to graphics
    file.
imfinfo - Information about graphics
    file.
```

Movies and animation.

```
capture - Screen capture of current
    figure.
moviein - Initialize movie frame
    memory.
getframe  - Get movie frame.
movie   - Play recorded movie frames.
qtwrite - Translate movie into
    QuickTime format (Macintosh
    only).
rotate  - Rotate object about
    specified orgin and direction.
frame2im  - Convert movie frame to
indexed image.
im2frame  - Convert index image into
movie format.
```

Color related functions.

```
spinmap - Spin color map.
rgbplot - Plot color map.
colstyle  - Parse color and style
    from string.
```

Solid modeling.

```
cylinder  - Generate cylinder.
sphere  - Generate sphere.
patch   - Create patch.
```

help graphics

Handle Graphics.

Figure window creation and control.

```
figure  - Create figure window.
gcf     - Get handle to current
    figure.
clf     - Clear current figure.
shg     - Show graph window.
close   - Close figure.
refresh - Refresh figure.
```

Axis creation and control.

```
subplot- Create axes in tiled
    positions.
axes   - Create axes in arbitrary
    positions.
gca    - Get handle to current axes.
cla    - Clear current axes.
axis   - Control axis scaling and
    appearance.
box    - Axis box.
caxis  - Control pseudocolor axis
    scaling.
hold   - Hold current graph.
ishold - Return hold state.
```

Handle Graphics objects.

```
figure - Create figure window.
axes   - Create axes.
line   - Create line.
text   - Create text.
patch  - Create patch.
surface - Create surface.
image  - Create image.
light          - Create light.
uicontrol      - Create user interface
control.
uimenu         - Create user interface
menu.
uicontextmenu - Create user interface
context menu.
```

Handle Graphics operations.

```
set    - Set object properties.
get    - Get object properties.
reset  - Reset object properties.
delete - Delete object.
gco    - Get handle to current
    object.
gcbo   - Get handle to current
    callback object.
gcbf   - Get handle to current
    callback figure.
drawnow- Flush pending graphics
    events.
findobj- Find objects with specified
    property values.
copyobj- Make copy of graphics object
    and its children.
```

Hardcopy and printing.

```
print  - Print graph or SIMULINK
    system; or save graph to M-file.
printopt - Printer defaults.
orient - Set paper orientation.
```

Utilities.

```
closereq  - Figure close request
    function.
newplot   - M-file preamble for
    NextPlot property.
ishandle - True for graphics handles.
```

ActiveX Client Functions (PC Only).

```
actxcontrol - Create an ActiveX
    control.
actxserver  - Create an ActiveX
    server.
```

help uitools

Graphical user interface tools.

GUI functions.

```
uicontrol - Create user interface
    control.
uimenu - Create user interface menu.
ginput - Graphical input from mouse.
dragrect  - Drag XOR rectangles with
    mouse.
rbbox  - Rubberband box.
selectmoveresize - Interactively
    select, move, resize, or copy
    objects.
waitforbuttonpress - Wait for
    key/buttonpress over figure.
waitfor  - Block execution and wait
    for event.
uiwait   - Block execution and wait
    for resume.
uiresume  - Resume execution of
    blocked M-file.
uisuspend - Suspend the interactive
    state of a figure.
uirestore - Restore the interactive
    state of a figure.
```

GUI design tools.

```
guide    - Design GUI.
align    - Align uicontrols and
    axes.
cbedit   - Edit callback.
menuedit  - Edit menu.
propedit  - Edit property.
```

Dialog boxes.

```
dialog   - Create dialog figure.
axlimdlg  - Axes limits dialog box.
errordlg  - Error dialog box.
helpdlg   - Help dialog box.
inputdlg  - Input dialog box.
listdlg   - List selection dialog
    box.
menu      - Generate menu of choices
    for user input.
msgbox    - Message box.
questdlg  - Question dialog box.
warndlg   - Warning dialog box.
uigetfile - Standard open file dialog
    box.
uiputfile - Standard save file dialog
    box.
uisetcolor - Color selection dialog
    box.
```

uisetfont - Font selection dialog box.
pagedlg - Page position dialog box.
printdlg - Print dialog box.
waitbar - Display wait bar.

Menu utilities.

makemenu - Create menu structure.
menubar - Computer dependent default setting for MenuBar property.
umtoggle - Toggle "checked" status of uimenu object.
winmenu - Create submenu for "Window" menu item.

Toolbar button group utilities.

btngroup - Create toolbar button group.
btnstate - Query state of toolbar button group.
btnpress - Button press manager for toolbar button group.
btndown - Depress button in toolbar button group.
btnup - Raise button in toolbar button group.

User-defined figure/axes property utilities.

clruprop - Clear user-defined property.
getuprop - Get value of user-defined property.
setuprop - Set user-defined property.

Miscellaneous utilities.

allchild - Get all object children.
findall - Find all objects.
hidegui - Hide/unhide GUI.
edtext - Interactive editing of axes text objects.
getstatus - Get status text string in figure.
setstatus - Set status text string in figure.
popupstr - Get popup menu selection string.
remapfig - Transform figure objects' positions.
setptr - Set figure pointer.
getptr - Get figure pointer.
overobj - Get handle of object the pointer is over.

10.4 Generování algoritmů v "C" a vytvoření spustitelného kódu

Kromě toho, že mohu chtít rozšířit funkce MATLABu o vlastní algoritmy[59] vytvořené ve standardním programovacím jazyku (podporován je jazyk "C" a FORTRAN) je velmi zajímavá i možnost opačná. Použití jak výpočetních algoritmů, které používá jádro MATLABu, tak i použití komplexních řešení vytvořených v MATLABu (v podobě m-souborů) ve vlastních programech. První variantu je možné řešit pomocí knihoven MATLAB C/C++ Math Library, která dovoluje používat (volat z vlastního programu psaného v "C" či "C++") stejné funkce jaké jsou používány jádrem MATLABu. Druhou variantu řeší MATLAB Compiler. Compiler dovoluje přeložit řešení problému realizované skriptem (m-souborem), který může obsahovat volání jak standardních funkcí jádra tak i uživatelských m-funkcí a výsledný program samostatně spustit. Do verze MATLABu 5.2 nebylo možné používat v překládaném řešení grafické možnosti MATLABu. Od verze 5.3 je nabízena i grafická knihovna MATLAB C/C++ Graphics Library umožňující vygenerovat (pomocí opět MATLAB Compiler) spustitelný (na MATLABu) nezávislý program obsahující jak numerické řešení tak i interaktivní grafické uživatelské rozhraní (vytvořené prostředky MATLABu).

10.5 Něco málo o toolboxech

Jak již bylo zmíněno na začátku MATLAB je tvořen jádrem obsahujícím základní funkce a prostředky. Pro jednotlivé oblasti použití je možné tento základ rozšířit o problémově (na danou oblast) orientované funkce soustředěné do toolboxu (Tbx) - něco jako knihovny. Jde o funkce řešící specializované problémy dané oblasti. S využitím možností systému Handle Graphics dovolují tyto tbx využívat značně sofistikované matematické postupy i uživatelům nepříliš zběhlým v matematice. Umožní uživateli jednoduše zadat jeho data, vybrat metodu a spustit výpočet. Přitom mohou tyto funkce komunikovat s uživatelem v pojmech, které jsou v dané oblasti standardní a uživatel nemusí o použitých matematických metodách na které je převedeno řešení jeho problému nic vědět. Příkladem takovéhoto toolboxu je Financial Toolbox (vyžaduje Statistic a Optimization Tbx) určený pro oblast bankovnictví (zpracování a predikce časových řad).

Další toolboxy jsou určeny pro určité oblasti použití

- obecné použití - NAG Foundation Tbx, Symbolic Math Tbx, Partial Differential Equation Tbx, Statistic Tbx, Spline Tbx, Optimization Tbx, Neural Network Tbx, Fuzzy Logic Tbx, Mapping Tbx
- zpracování signálu a obrazu - Wavelet Tbx, System Identification Tbx, Signal Processing Tbx, High-Order Spectral Analysis Tbx, Frequency Domain Identification Tbx, Data Acquistion Tbx, Quantized Filtering Tbx, Communications Tbx
- návrhu řízení - Control System Tbx, Robust Control Tbx, Model Predictice Control Tbx, Nonlinear Control Design Tbx, LMI Control Tbx, μ-Analysis and Synthesis Tbx, Quantitative Feedback Theory Tbx

Toolboxy tohoto typu většinou využívají pouze funkce jádra případně jiných tbx.

Kromě výše uvedených tbx fy MathWorks existují ještě tbx jiných výrobců, které jsou součástí standardní nabídky (oficiálně podporované) a desítky dalších, které jsou vytvořeny obvykle na univerzitách pro řešení problematiky v určité oblasti.

Některé toolboxy obsahují vlastní výkonné programy a příkazy v syntaxi MATLABu k nim zprostředkovávají přístup. Do této kategorie patří např. podpora pro návrh a tvorbu algoritmů určených pro digitální signálové procesory (DSP) - Fixed-Point Blockset, možnost použití DSP přímo v SIMULINKovém schématu (DSP Blockset) atd.. Samostatnou zmínku zasluhují dva produkty z této kategorie tbx - Symbolic Math Toolbox a Real Time Toolbox.

---

[59] ať už z důvodů vyšší rychlosti řešení algoritmu, skrytí vlastního algoritmu před jinými uživateli či pro využití možností, které MATLAB neposkytuje (např. využití dat uložených ve speciální databázi)

10.5.1 Symbolic Math Toolbox

Toolbox pro symbolické výpočty je typickým příkladem tbx, součástí kterého je jiné výpočetní jádro - v tomto případě (MATLAB 5.2) jádro MAPLE V Release 4 od fy Waterloo Maple, Inc. Opět jsou využívány výhody objektového přístupu. Ukažme si několik komentovaných příkladů

```
x=sym('x');                                   % vytvoření objektu symbolické proměnné x
```

výpočty s definovaným počtem platných cifer

```
vpa(erf(0.5),40)                              % výpočet chybové funkce erf na 40 cifer
  ans =     .5204998778130465186819719747290946543217
M=[0.1 -1.2; 4.3 1.1];
vpa(inv(M),20)                                % inverze matice  na 20 cifer
  ans =  [ .20872865275142314991, .22770398481973434535]
         [-.81593927893738140417, .18975332068311195446e-1]
```

maticové operace

```
A=[sin(x),-sin(x);cos(x),sin(x)]              % vytvoření matice funkcí
  A =    [  sin(x), -sin(x)]
         [  cos(x),  sin(x)]
det(A)                                        % výpočet determinantu (symbolicky)
  ans =  sin(x)^2+cos(x)*sin(x)
inv(A)                                        % výpočet inverzní matice (symbolicky)
  ans =
  [                1/(sin(x)+cos(x)),                1/(sin(x)+cos(x))]
  [ -cos(x)/sin(x)/(sin(x)+cos(x)),                1/(sin(x)+cos(x))]
```

integrace a derivace

```
int(x^2*sin(x))                               % integrace analyticky řešitelná
  ans =  -x^2*cos(x)+2*cos(x)+2*x*sin(x)
int(exp(-x^2))                                % integrace analyticky neřešitelná (chybová fce erf(x))
  ans =  1/2*pi^(1/2)*erf(x)
diff(x^2*sin(x^2))                            % derivace funkce
  ans =  2*x*sin(x^2)+2*x^3*cos(x^2)
```

diferenciální rovnice $\frac{d^2}{dt^2}y + a\frac{d}{dt}y + b = \sin(t) \qquad \frac{d}{dt}y\Big|_{t=0} = a \quad y(t=0) = b$

```
r=dsolve('D2y+a*Dy+b=sin(t)','Dy(0)=a','y(0)=b')
  r =  (-b*t*a^3-a*b*t-cos(t)*a^3+b*a^2+b-a^2*sin(t)+(a*b+a+1)*a^3
       +(a*b+a+1)*a-(b*a^2+b+a+a^4+a^2)*a^2/(a^2+1)*exp(-a*t)
       -(b*a^2+b+a+a^4+a^2)/(a^2+1)*exp(-a*t)
       )/a^2/(a^2+1)
```

```
simplify(r)                                   % zjednodušení výrazu
  ans =(-a^4*exp(-a*t)+a^4+b*a^4-b*t*a^3-cos(t)*a^3+a^3
        -a^2*exp(-a*t)-exp(-a*t)*b*a^2+a^2+2*b*a^2-a^2*sin(t)
        +a-a*b*t-a*exp(-a*t)+b-exp(-a*t)*b
       )/a^2/(a^2+1)
```

integrální transformace

```
f=fourier(exp(-x^2))                          % Fourierova transformace
  f =     pi^(1/2)*exp(-1/4*w^2)
ifourier(f)                                   % zpětná Fourierova transformace
  ans =   1/2*4^(1/2)*exp(-x^2)
a=sym('a');                                   % symbolická proměnná a
l=laplace(x^2*exp(-a*x))                      % Laplaceova transformace
  l =     2/(s+a)^3
ilaplace(l)                                   % zpětná Laplaceova transformace
  ans =   t^2*exp(-a*t)
```

Příkazy a funkce používané v Symbolic Math Tbx jsou v adresáři toolbox\symbolic. Jejich seznam lze získat pomocí příkazu **help symbolic**.

```
help symbolic

  Symbolic Math Toolbox.
  Version 2.0.1 21-Nov-1997

  Calculus.
diff    - Differentiate.
int     - Integrate.
limit   - Limit.
taylor  - Taylor series.
jacobian  - Jacobian matrix.
symsum  - Summation of series.

  Linear Algebra.
diag    - Create or extract diagonals.
triu    - Upper triangle.
tril    - Lower triangle.
inv     - Matrix inverse.
det     - Determinant.
rank    - Rank.
rref    - Reduced row echelon form.
null    - Basis for null space.
colspace  - Basis for column space.
eig  - Eigenvalues and eigenvectors.
svd     - Singular values and singular
     vectors.
jordan - Jordan canonical (normal)
     form.
poly    - Characteristic polynomial.
expm    - Matrix exponential.

  Simplification.
simplify  - Simplify.
expand - Expand.
factor - Factor.
collect- Collect.
simple - Search for shortest form.
numden - Numerator and denominator.
horner - Nested polynomial
     representation.
subexpr- Rewrite in terms of
     subexpressions.
subs    - Symbolic substitution.

  Solution of Equations.
solve   - Symbolic solution of
     algebraic equations.
dsolve - Symbolic solution of
     differential equations.
finverse  - Functional inverse.
compose- Functional composition.

  Variable Precision Arithmetic.
vpa     - Variable precision
     arithmetic.
digits - Set variable precision
     accuracy.
```

Integral Transforms.

```
fourier- Fourier transform.
laplace- Laplace transform.
ztrans - Z transform.
ifourier  - Inverse Fourier
     transform.
ilaplace  - Inverse Laplace
     transform.
iztrans   - Inverse Z transform.
```

Conversions.

```
double - Convert symbolic matrix to
     double.
poly2sym  - Coefficient vector to
     symbolic polynomial.
sym2poly  - Symbolic polynomial to
     coefficient vector.
char    - Convert sym object to
     string.
```

Basic Operations.

```
sym     - Create symbolic object.
syms    - Short-cut for constructing
     symbolic objects.
findsym- Determine symbolic
     variables.
pretty - Pretty print a symbolic
     expression.
latex   - LaTeX represention of a
     symbolic expression.
ccode   - C code represention of a
     symbolic expression.
fortran- Fortran represention of a
     symbolic expression.
```

Special Functions.

```
sinint - Sine integral.
cosint - Cosine integral.
zeta    - Riemann zeta function.
    lambertw    - Lambert W function.
```

String handling utilities.

```
isvarname - Check for a valid
     variable name.
vectorize - Vectorize a symbolic
     expression.
```

Pedagogical and Graphical Applications.

```
rsums   - Riemann sums.
ezplot - Easy to use function
     plotter.
funtool- Function calculator.
```

Demonstrations.

```
symintro  - Introduction to the
     Symbolic Toolbox.
symcalcdemo - Calculus demonstration.
symlindemo  - Demonstrate symbolic
     linear algebra.
symvpademo  - Demonstrate variable
     precision arithmetic
symrotdemo  - Study plane rotations.
symeqndemo  - Demonstrate symbolic
     equation solving.
```

Access to Maple. (Not available with Student Edition.)

```
maple   - Access Maple kernel.
mfun    - Numeric evaluation of Maple
     functions.
mfunlist  - List of functions for
     MFUN.
mhelp   - Maple help.
procread - Install a Maple procedure.
(Requires Extended Toolbox.)
```

10.5.2 Real Time Toolbox

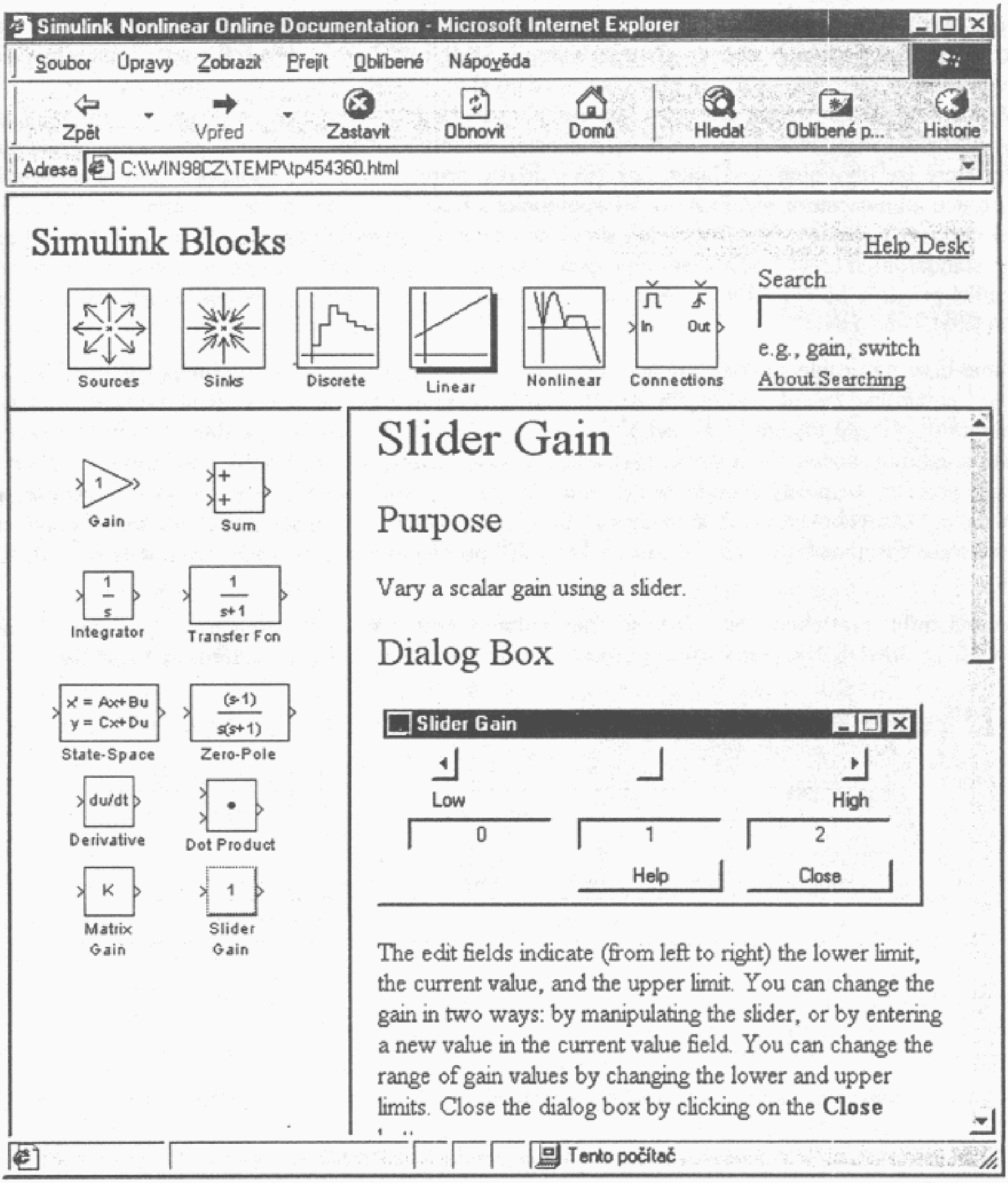
Příkladem úzce specializovaného tbx je Real Time Toolbox, který byl vytvořen českou firmou HUMUSOFT, distributorem MATLABu pro Českou republiku. Je určen pro propojení MATLABu a SIMULINKu s reálným světem a pouze pro platformu Windows/Intel. Umožňuje sběr dat v reálném čase, jejich bezprostřední zpracování příkazy MATLABu či modelem SIMULINKu a okamžitou realizaci vypočtených hodnot (ovládání výstupních signálů). Poskytuje také několik vlastních funkcí, které nahrazují standardní příkazy (z principu pomalé díky tomu, že MATLAB je interpretr). Ve spojení s toolboxem pro SIMULINK - Real Time Workshop - dovoluje realizovat SIMULINKové modely, které využívají data v reálném čase z okolí a používají hardwarově realizované bloky např. v signálových procesorech.

Ideální využití tohoto toolboxu je např. v oblasti výuky řízení procesů. Umožňuje využití flexibility při tvorbě algoritmu, komplexní nabídky matematických funkcí a robustnosti funkcí jádra MATLABu a výpočetní sílu současných počítačů při návrhu a jednoduché realizaci řízení laboratorního modelu v reálném čase. Že tento způsob řízení není omezen jen na zkoušení v laboratoři dokazuje i článek "*Roháľ-Ilkiv,B. ;Benčurik,B.: Viacrozmerne prediktivné riadenie tunelovej pece. AT&P Journal, 1999, s,41-43*", kde je použit MATLAB ve verzi 4.2 pro řízení ve skutečném provoze.

11. Co je SIMULINK ?

Rozšíření MATLABu (nadstavba) SIMULINK je k dispozici až od verze 4. Využívá grafických možností hostitelské platformy (Windows). Způsob zápisu modelu i celkový způsob práce je značně odlišný od práce s vlastním MATLABem, který je orientován na řádkové příkazy. To svádí k tomu, používat SIMULINK "samostatně", nezávisle na znalosti (neznalosti) MATLABu. I tento přístup je možný, ale vystačíme s ním pouze pro nejjednodušší využití. Pro efektivní využití a při řešení složitějšího problému je znalost MATLABu prakticky nezbytná.

Na pozdější původ SIMULINKu ukazuje i jiný systém nápovědy. K dispozici je dobře udělaná on-line nápověda - tlačítka help na jednotlivých blocích. Nápověda je ve formátu HTML a využívají se hypertextové odkazy. Tato forma nápovědy umožňuje používat obrázky (viz obr. 11-1). Určitou nevýhodou může být to,



Obrázek 11-1 Příklad nápovědy v SIMULINKu

že je nutné mít instalovaný prohlížeč (Netscape, MS Explorer či jiný). Kromě této on-line nápovědy je k dispozici též elektronická dokumentace ve formátu PDF.

SIMULINK je určen na časové řešení - simulaci - chování dynamického systému pokud známe jeho matematický popis. Lze s jeho pomocí určit časové průběhy výstupních veličin (a všech vnitřních) v závislosti na časovém průběhu veličin vstupních a počátečním stavu. Popis soustavy může být značně rozsáhlý a složitý.

Je potřeba si uvědomit, že SIMULINK nové metody či principiálně jiné možnosti řešení než MATLAB nenabízí. Co nabízí je jednoduchý přístup k propracovaným[60] metodám MATLABu pro časové řešení soustavy nelineárních diferenciálních rovnic, prostředky pro relativně snadný zápis problému (vytvoření modelu) a vizualizaci výsledků a automatické řešení mnoha problémů, které při řešení vznikají.

Vzhledem k třídě problémů, na řešení kterých je SIMULINK určený, se u uživatele předpokládají alespoň základní znalosti z oblasti matematického modelování (speciálně dynamických systémů) a simulace.

Přístup k návrhu zápisu problému silně připomíná návrh zapojení pro analogový počítač. Co zaujme na první pohled, je grafický způsob zápisu - v terminologii SIMULINKu nazývaný **model**. Z nabídky příslušné knihovny se myší přetahují výkonné **bloky** a pak se myší pospojují odpovídající vstupy a výstupy. Při tvorbě složitějších modelů se využívá možnosti část modelu sdružit do dalšího bloku a ty opět standardně kopírovat a spojovat. Jako vstupy modelu se používají nejčastěji signály z knihovních bloků generujících základní typy signálů, které lze libovolně poskládat. Lze též používat připravená data ze souboru nebo z matic připravených v pracovním prostoru MATLABu. Ve spolupráci s Real Time Tbx lze jako vstupní data použít i přímo měřené hodnoty v reálném čase. Výsledek simulace - časový průběh řešení - se zobrazuje nejčastěji graficky pomocí standardních bloků - zobrazení typu osciloskop či XY graf. Další možností je výstup do proměnných pracovního prostoru MATLABu či ukládání do diskového souboru. Opět je možná přímá realizace prostřednictvím Real Time Tbx.

Podíváme-li se na problematiku simulace chování dynamických systémů podrobněji zjistíme, že simulace není jen[61] numerické řešení soustavy nelineárních diferenciálních rovnic (i když je to asi nejdůležitější část). V konkrétním případě musíme řešit několik dalších problémů - od výběru vhodné metody řešení pro daný problém a zahrnutí počátečních stavů, přes volbu kroku řešení (souvisí s požadovanou přesností a rychlostí výpočtů), problém nespojitých nelinearit ("zahuštění" bodů řešení v okolí nespojitosti), synchronizaci výpočtu na zadaný čas, problém algebraických smyček až po problém závislostí daných tabulkou. Právě automatické (i s možností manuální volby) řešení některých[62] problémů a podporu pro řešení dalších[63] SIMULINK nabízí.

Kromě vlastního grafického prostředí se standardními knihovnami (a případných rozšíření o knihovny z toolboxů) je SIMULINK podporován i příkazy a funkcemi dostupnými z prostředí MATLABu.

---

[60] pod pojmem propracované je myšleno, že je nabízeno více metod, které jsou jak robustní (numericky stabilní) tak efektivní (rychlé řešení)

[61] proto také vzniklo několik nových programovacích jazyků specializovaných na oblast simulace

[62] automatická volba kroku, nespojité nelinearity, synchronizace výpočtu, tabulkou zadané závislosti

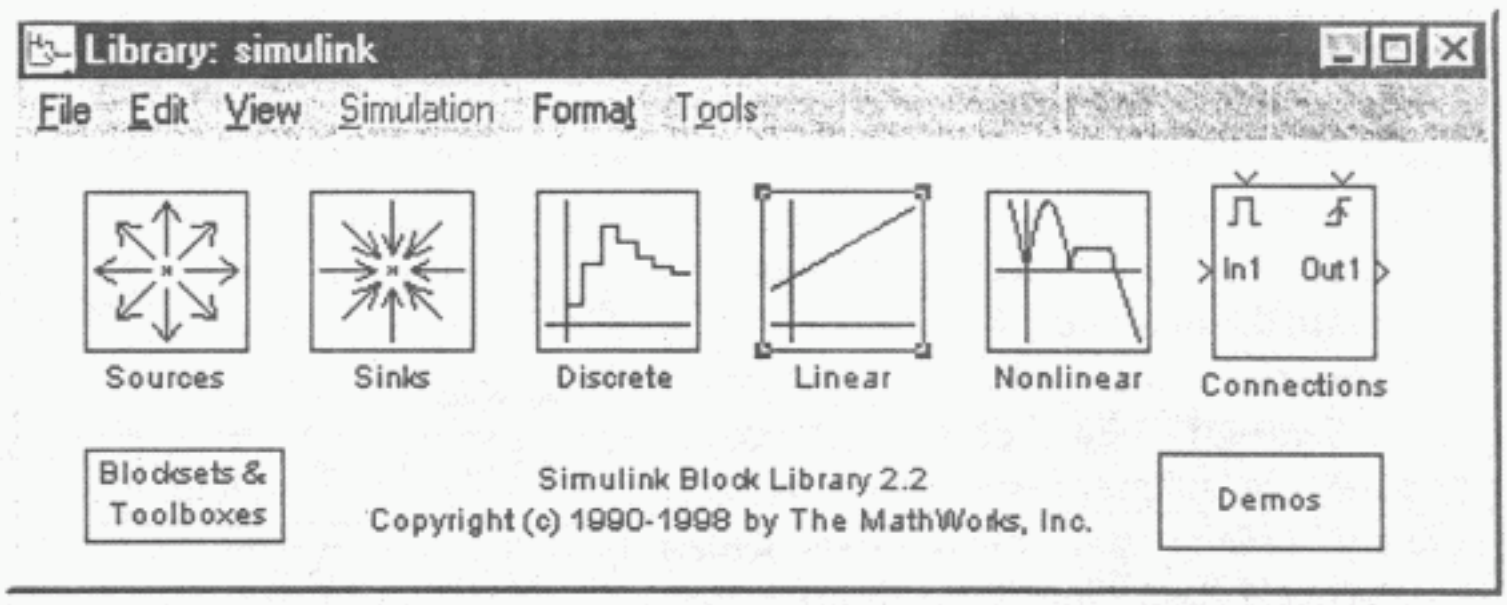
[63] linearizace ve vybraném bodě, řešení algebraických smyček

```
help simulink

  Simulink
  Version 2.2.1  24-Mar-98

      Model analysis and
    construction functions.

  Simulation.
sim     - Simulate a Simulink model.
sldebug- Debug a Simulink model.
simset  - Define options to SIM
    Options structure.
simget  - Get SIM Options structure

  Linearization and trimming.
linmod  - Extract linear model from
    continuous-time system.
linmod2- Extract linear model,
    advanced method.
dlinmod- Extract linear model from
    discrete-time system.
trim    - Find steady-state operating
    point.

  Model Construction.
close_systém   - Close open model or
    block.
new_systém   - Create new empty model
    window.
open_systém - Open existing model or
    block.
save_systém - Save an open model.
add_block    - Add new block.
add_line     - Add new line.
delete_block   - Remove block.
delete_line - Remove line.
find_systém - Search a model.
replace_block  - Replace existing
    blocks with a new block.
set_param - Set parameter values for
    model or block.
get_param - Get simulation parameter
    values from model.
bdclose   - Close a Simulink window.
bdroot    - Root level model name.
gcb     - Get the name of the current
    block.
gcs     - Get the name of the current
    system.
getfullname - get the full path name
    of a block
slupdate  - Update older 1.x models
    to 2.x.
addterms  - Add terminators to
    unconnected ports.

  Masking.
hasmask- Check for mask.
hasmaskdlg - Check for mask dialog.
hasmaskicon - Check for mask icon.
iconedit  - Design block icons using
    ginput function.
movemask  - Restructure masked built-
    in blocks as masked subsystems.

  Library.
libinfo- Get library information for
    a system.

  Hardcopy and printing.
frameedit - Edit print frames for
    annotated model printouts.
print     - Print graph or SIMULINK
    system; or save graph to M-file.
printopt  - Printer defaults.
orient    - Set paper orientation.
```

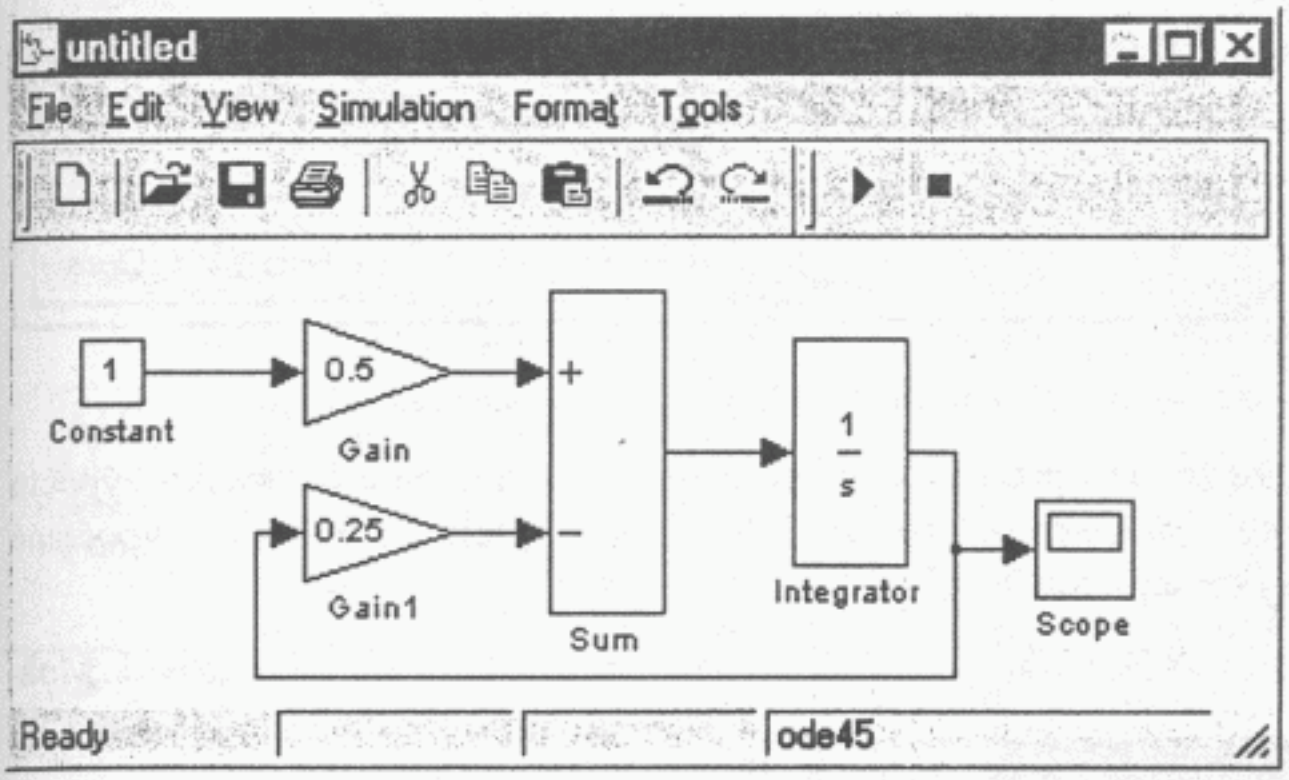
12. Jak se v SIMULINKu pracuje

Základní podmínkou pro spuštění SIMULINKu je spuštěný MATLAB.

Z příkazové řádky MATLABu ho spustíme příkazem **simulink**. Objeví se okno se standardními knihovnami (obr. 12-1) a okno pro zápis modelu (obr. 12-2). Na vedlejším obrázku je již v okně modelu zápis matematického modelu soustavy I. řádu popsaného rovnicí



Obrázek 12-1 Okno základních knihoven



Obrázek 12-2 Okno pro zápis modelu

$$\frac{dx}{dt} + 0.25x = 0.5$$ s nulovými počátečními podmínkami. Výsledek je zobrazován pomocí bloku osciloskopu (blok `Scope`). Poklepáním myší na bloku osciloskopu se otevře samostatné okno ve kterém je vykreslován v průběhu simulace časový průběh příslušného signálu.

Vytvoření modelu na základě standardního zadání bude ukázáno v kapitole 14 *Řešené příklady*.

12.1 Vytvoření modelu a jeho spuštění

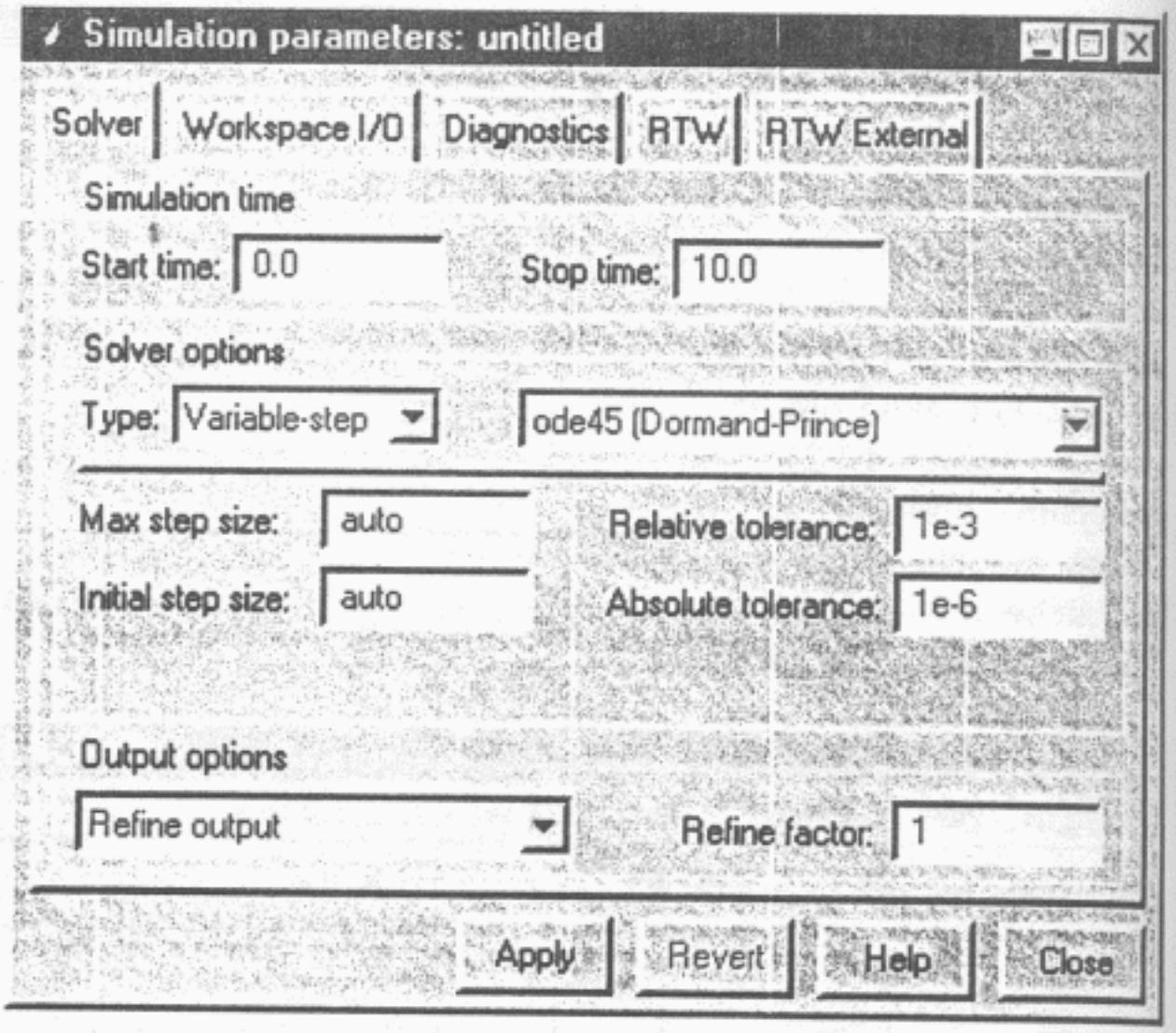
Vlastní model se vytváří výběrem bloků (bloků) z knihoven a jejich přesunem[64] do okna modelu myší. Krome přesunu z knihovny je možné i bloky v okně modelu standardním způsobem kopírovat. Po přesunu (či kopírování) bloků je možné zadat jejich parametry. Název nového bloku se nastaví automaticky tak, aby byl v rámci okna modelu **jednoznačný**. Např. první přesunutý blok `Sum` bude mít název `Sum`, druhý `Sum1` atd. Tyto názvy je možné změnit. Změna se provede dvojklikem na měněném názvu - přejde se do režimu editace názvu. Je potřeba dbát na to, že názvy musí být v rámci okna modelu jednoznačné. Když toto pravidlo nedodržím, SIMULINK mi nedovolí stejný název vytvořit.

Dále je potřeba pospojovat odpovídající vstupy a výstupy bloků. Výstup jednoho bloku může být připojen na libovolný počet vstupů jiných bloků či toho samého bloku. Spojení se provede myší tak, že kurzor myši se umístí na značku vstupu (či výstupu) bloku a při stisknutém levém tlačítku se kurzor přemístí na výstup (vstup) jiného bloku a tlačítko se pustí. V případě většího požadovaného počtu pravoúhlých zalomení je možné taženou čáru ukončit bez ukončujícího připojení a z tohoto místa táhnout další čáru. V libovolném

[64] pojem přesun není v tomto případě zcela přesný. Nejde o přesun ve významu používaném ve Windows - blok v knihovně nezmizí. Nejde ale ani o kopírování ve smyslu vytvoření nezávislé kopie. Ve skutečnosti se vytvoří tzv. Link tj. vazba na knihovní blok (viz kapitola 13.3)

místě okna modelu je možné umístit komentář - vysvětlující text. Dvojklikem na požadovaném místě se dostaneme do režimu editace a můžeme zapsat požadovaný text.

Po vytvoření modelu nastavíme základní parametry simulace. Po otevření nabídky *Simulation* z hlavního menu okna modelu a výběru položky *Parameters* (nebo *Ctrl_E*) se objeví okno parametrů simulace se záložkami *Solver*, *Workspace I/O* a *Diagnostic*. Další záložky (*RTW* a *RTW External*) se objeví pouze v případě, že máme instalovány další toolboxy určené pro SIMULINK (v tomto případě Real Time Workshop).

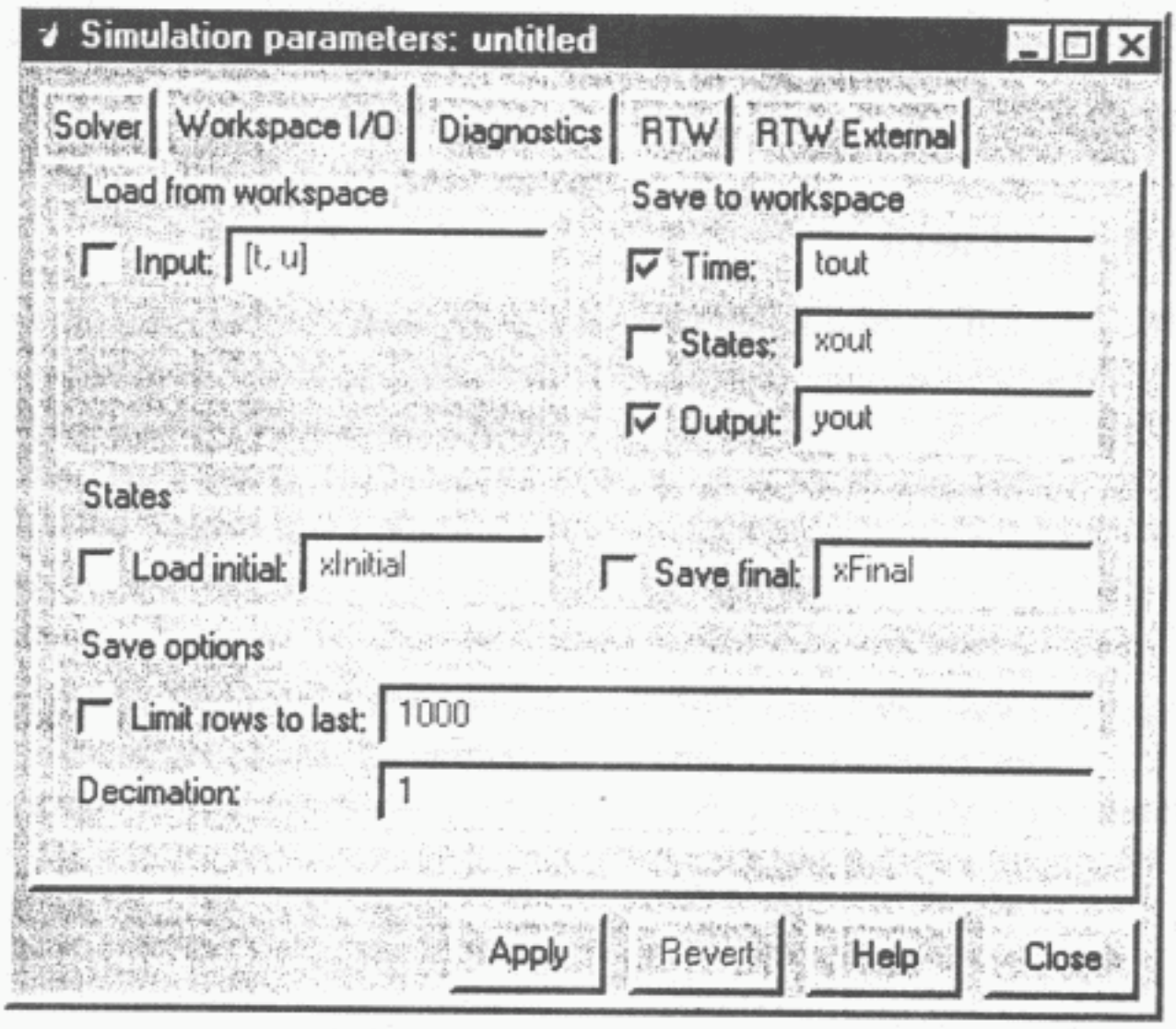


Obrázek 12-3 Okno parametrů simulace - Solver

Pod záložkou **Solver** je možné nastavit čas simulace (*Start time* a *Stop time*), volbu řešitele (použitou metodu řešení) a volby týkající se automatické velikosti kroku a volbu týkající se požadavků na okamžiky, kdy má být k dispozici hodnota výstupu. Implicitní volby jsou na obr. 12-3.

Volby pod záložkou **Workspace I/O** se týkají možností napojení na pracovní prostor MATLABu. Je možné předepsat, že se mají přebírat počáteční stavy, požadované okamžiky výpočtu a průběhy vstupních veličin z proměnných v pracovním prostoru MATLABu. Dále že vybrané vypočtené veličiny se mají ukládat do předepsaných proměnných MATLABu (obr. 12-4).

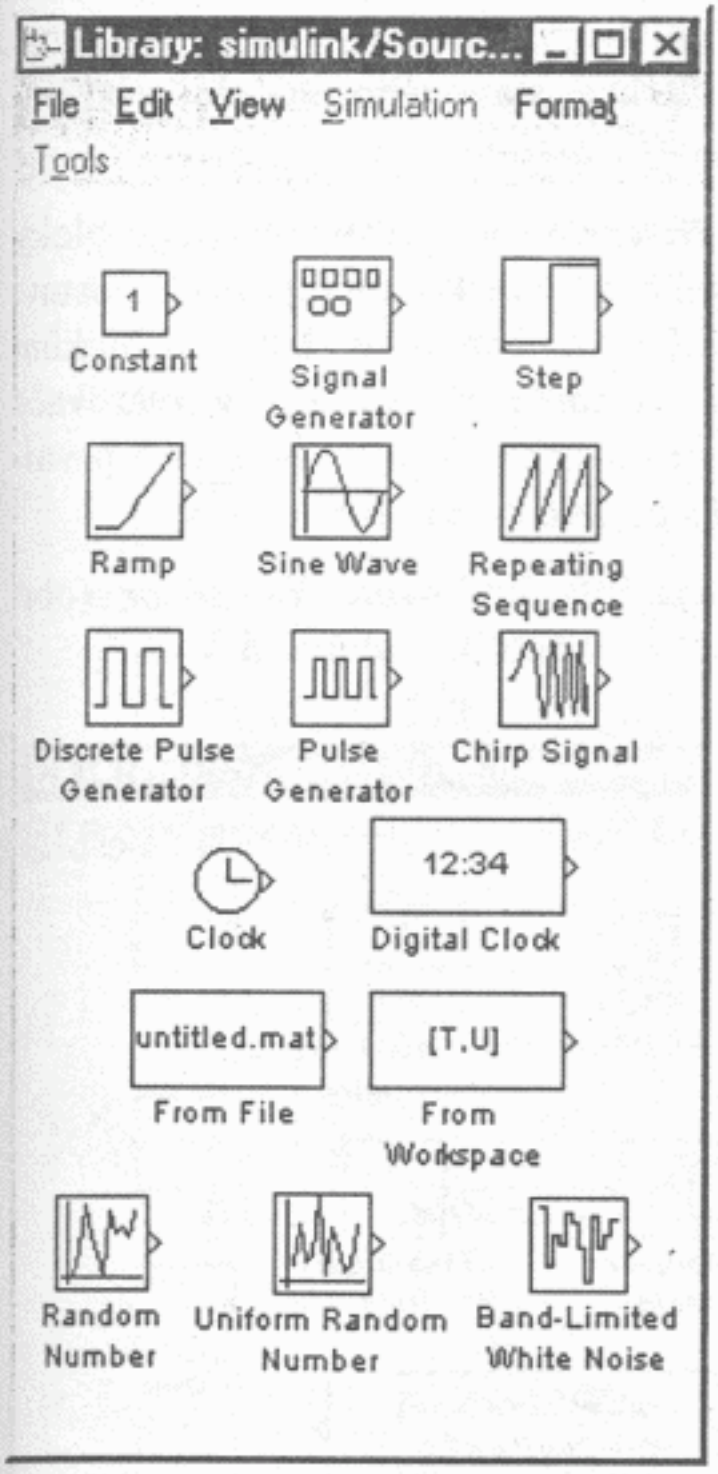
Volby pod záložkou **Diagnostics** umožňují nastavit zda události mají vyvolat hlášení a na jaké úrovni. Možné úrovně jsou hlášení jsou *None* (N), *Warning* (W) a *Error* (E).



Obrázek 12-4 Okno parametrů simulace - Workspace

Nyní je již možné příkazem *Simulation - Start* (nebo *Ctrl_T*) spustit simulaci. V případě, že nebyl zařazen žádný blok pro vizualizaci průběhu výsledku se pouze v stavovém řádku okna modelu ukazuje průběh času výpočtu. V případě, že použijeme některý zobrazovací blok je potřeba dvojklikem na bloku otevřít zobrazovací okno. Pokud to neuděláme (ať už před spuštěním simulace nebo po skončení) nedojde k zobrazení sledovaných veličin. Pozor na to, že zobrazovací okno má implicitně nastavené určité měřítko a když jsou naše výsledky výrazně jiné, nemusíme nic vidět. V tomto případě je třeba manuálně změnit měřítko v zobrazovacím okně (např. pomocí ikony s dalekohledem nastavíme měřítku tak, aby se zobrazilo vše).

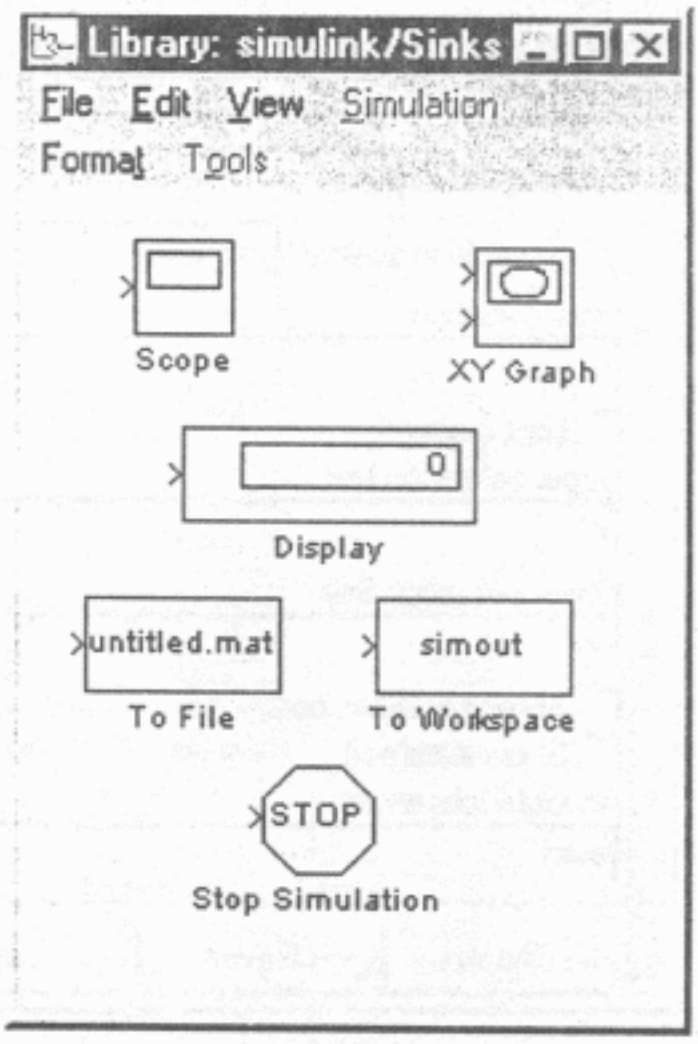
12.2 Standardní knihovny



Obrázek 12-5 Knihovna Sources

Každá knihovna obsahuje jednotlivé bloky, které lze myší přetáhnout do okna modelu. Většina bloků dovoluje zadat hodnoty parametrů. Hodnoty lze zadat buď jako konstanty nebo i pomocí názvů proměnných, které jsou umístěny v pracovním prostoru MATLABu. Tento způsob dovoluje inicializaci hodnot pomocí naplnění hodnot proměnných např. skriptem v MATLABu před spuštěním modelu v SIMULINKu. Bloky jsou v knihovnách seskupeny podle oblasti použití.

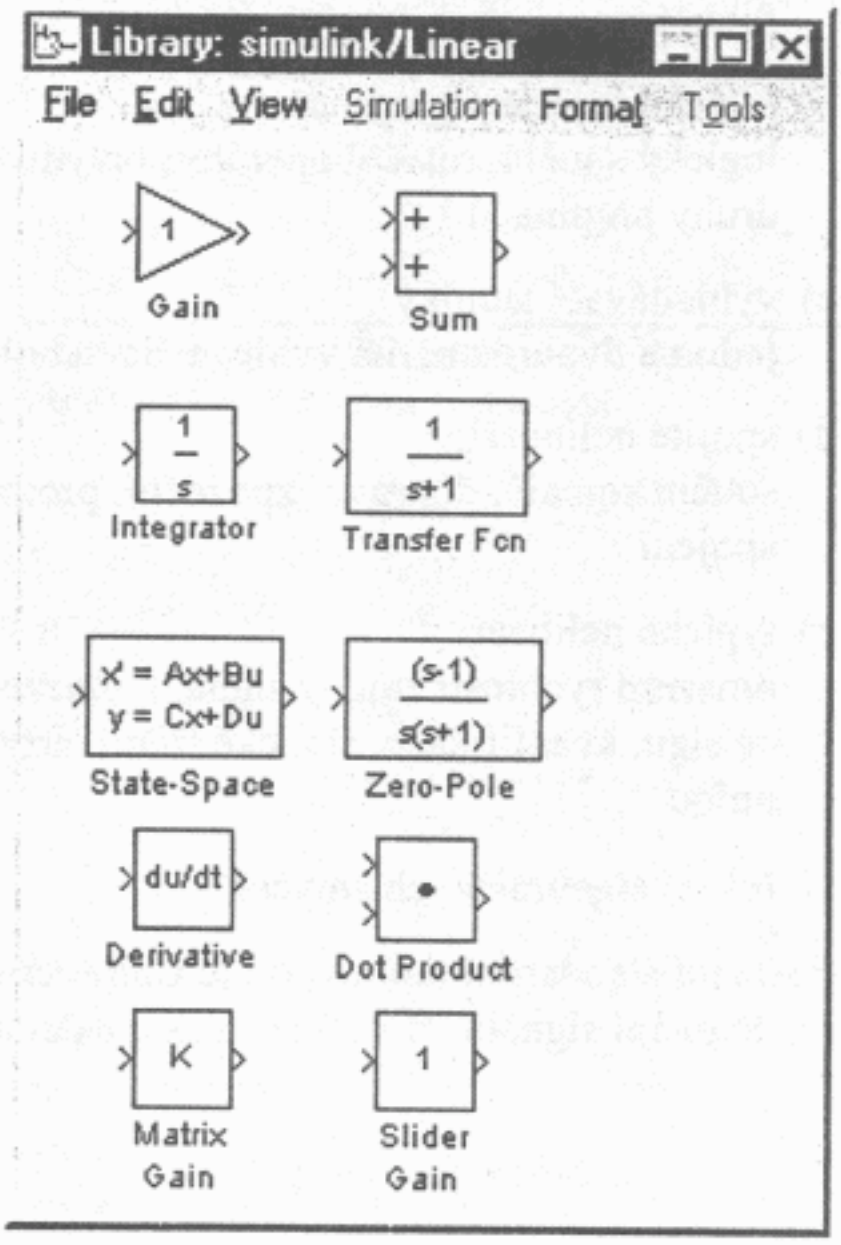
V knihovně ***Sources*** (obr. 12-5) jsou bloky použitelné jako zdroje běžných druhů signálů. Nejjednodušším je zdroj konstantního signálu. Je zde i blok umožňující načítat data z diskového souboru i blok, který generuje časový signál.



Obrázek 12-6 Knihovna Sinks

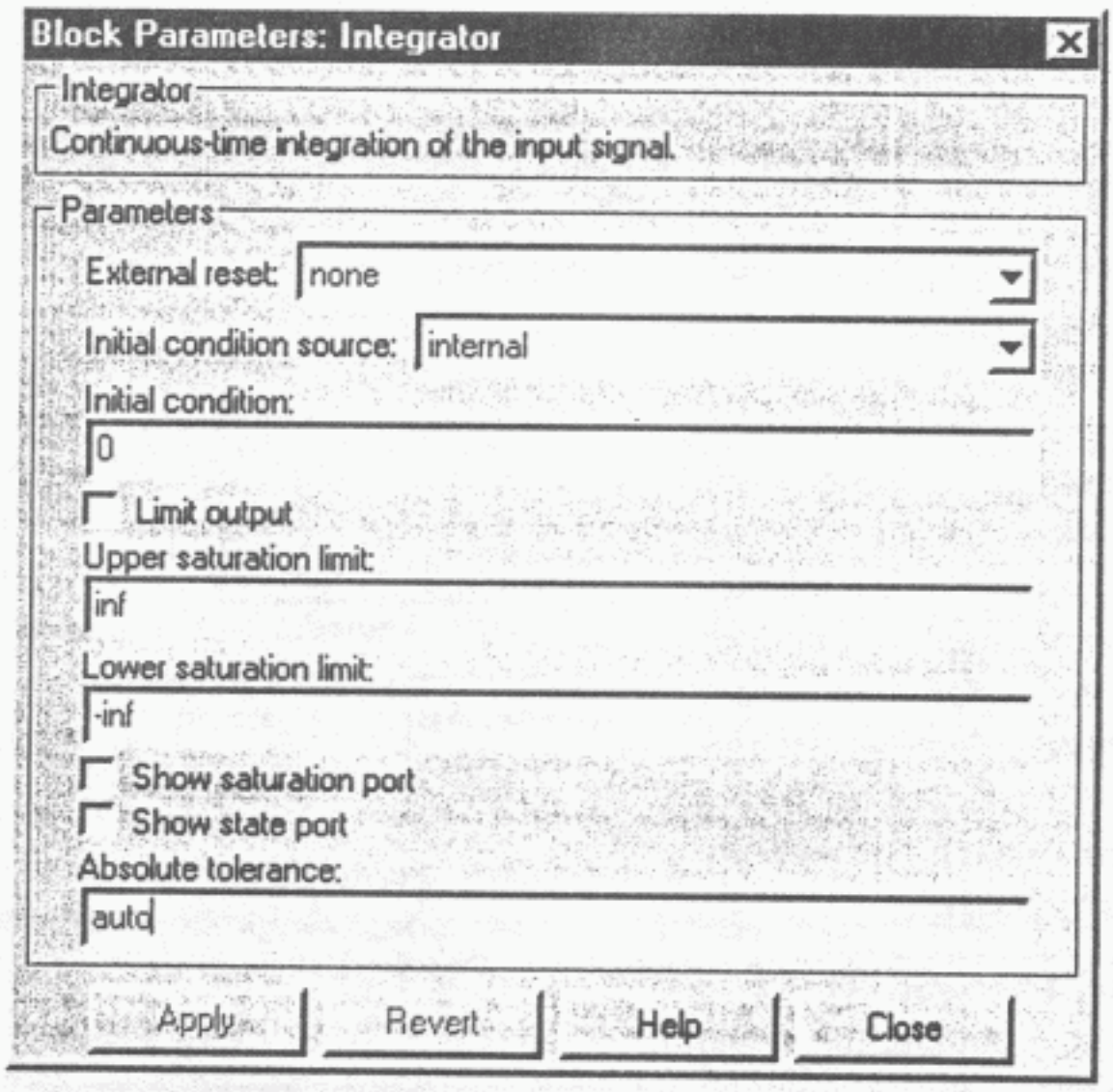
V knihovně ***Sinks*** (obr. 12-6) jsou bloky pro zobrazení a uložení průběhu signálů a blok umožňující ukončit simulaci z modelu.

Knihovna ***Linear*** (obr. 12-7) obsahuje bloky sloužící pro vytvoření lineárních modelů. Prvky z knihovny Linear tvoří obvykle základ dynamického modelu. Jaké prvky to jsou? Mezi základní patří blok `Gain` (zesílení) a `Sum`[65] (součet). Blok `Slider Gain` (měnitelné zesílení) dovoluje měnit zesílení v průběhu simulace pomocí okna s ovládacím prvkem. Bloky `Dot Product` (násobení prvek po prvku) a `Matrix Gain` (maticové násobení) jsou užitečné v případě, že signály jsou tvořeny vektory. Blok numerické derivace `Derivative` je nutné používat s rozvahou. Principiálně jasná operace derivace je obtížně numericky aproximovatelná. Dalším problémem je možná nespojitost signálu, kdy hodnota derivace není definovaná. Asi nejdůležitějším blokem je blok `Integrator` - numerická integrace (obr. 12-8). Blok dovoluje zadat počáteční stav a omezení (saturaci) výstupu. Obě veličiny je možné zadat buď hodnotou parametru nebo formou samostatného signálu. Lze použít reset (nastavení do počátečního stavu) na základě změny na řídicím signálu resetu (na vzestupnou, sestupnou ne-



Obrázek 12-7 Knihovna Linear

[65] Parametrem bloku Sum je počet a znaménko signálů braných do součtu. Podle počtu parametru se změní symbol bloku Sum- změní se počet vstupů.



Obrázek 12-8 Okno parametrů bloku Integrator

a) matematické funkce
absolutní hodnota, matematické funkce, trigonometrické funkce, minimum a maximum, obecná funkce a přístup k m-funkcím

b) logické funkce a přepínače
logický součin, relační operátor, pravdivostní tabulka, tři druhy přepínačů

c) vyhledávací tabulky
jedno a dvourozměrná vyhledávací tabulka

d) spojité nelinearity
součin signálů, dopravní zpoždění, proměnné dopravní spojení

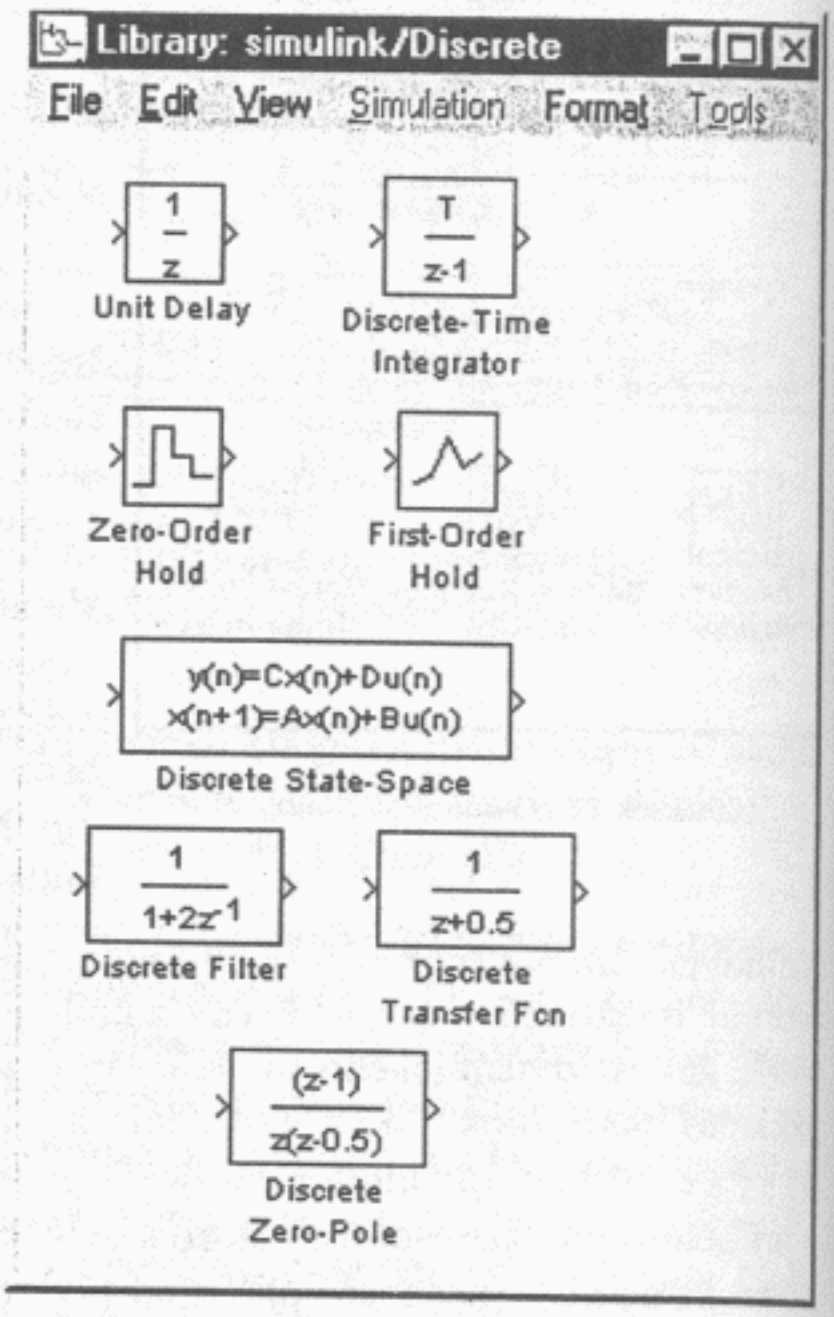
e) typické nelinearity
omezení rychlosti změny signálu, omezení (saturace), funkce sign, kvantifikace, statické tření, mrtvé pásmo, průchod nulou

f) řešení algebraických smyček

bo obě hrany). Zbývající bloky - `Transfer Fcn`, `State-Space` a `Zero-Pole` - jsou bloky dovolující zadat popis soustavy ve formě přenosové funkce, stavového modelu nebo pomocí nul, pólů a zesílení.

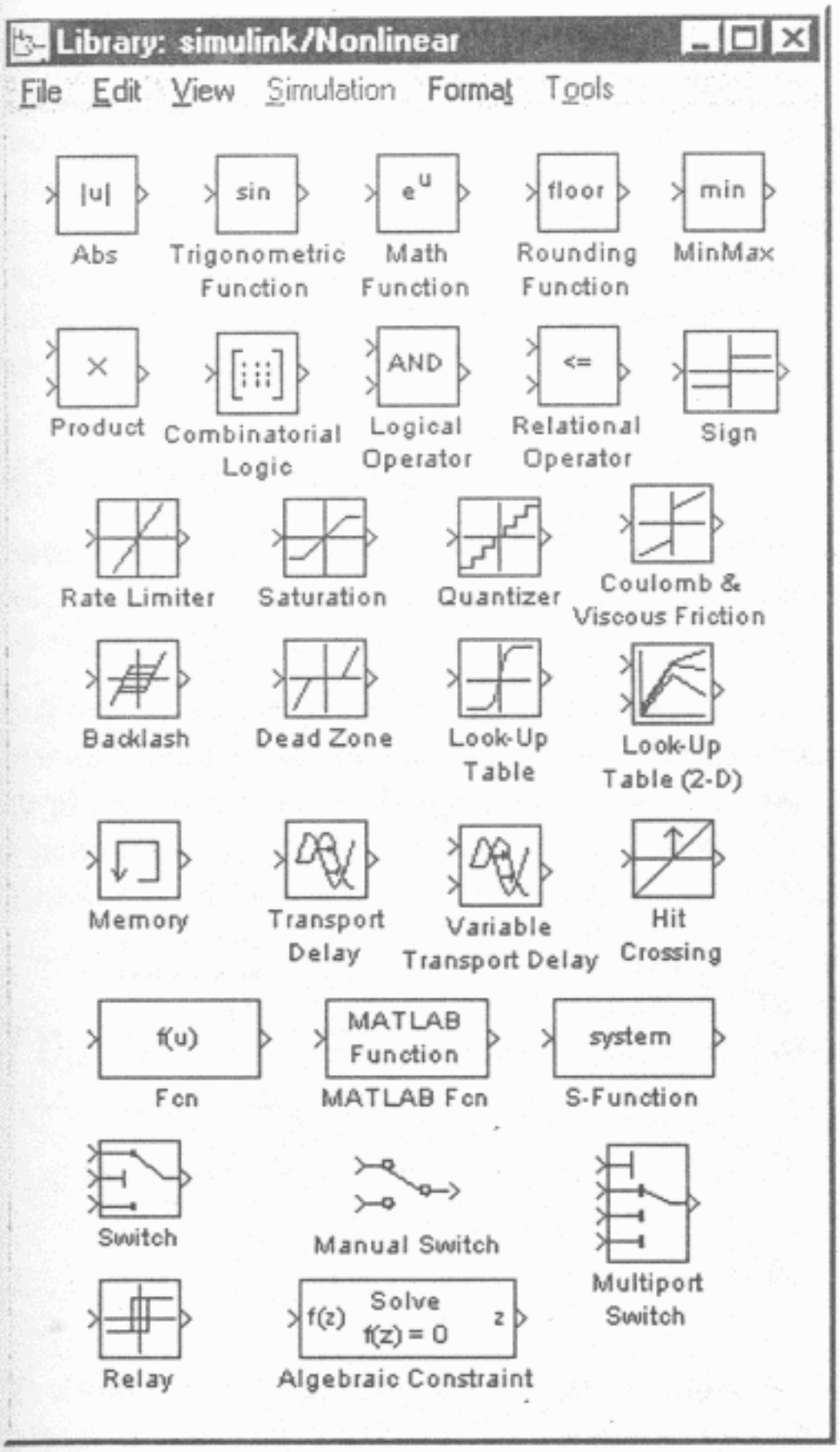
Knihovna ***Discrete*** (obr. 12-9) obsahuje bloky potřebné pro tvorbu diskrétních popisů soustav. Kromě bloků víceméně ekvivalentních blokům z knihovny Linear jsou zde bloky tvarovače nultého řádu - `Zero-order Hold` - a prvního řádu - `First-order Hold`.

Poměrně rozsáhlá knihovna ***Nonlinear*** (obr. 12-10) obsahuje několik druhů bloků

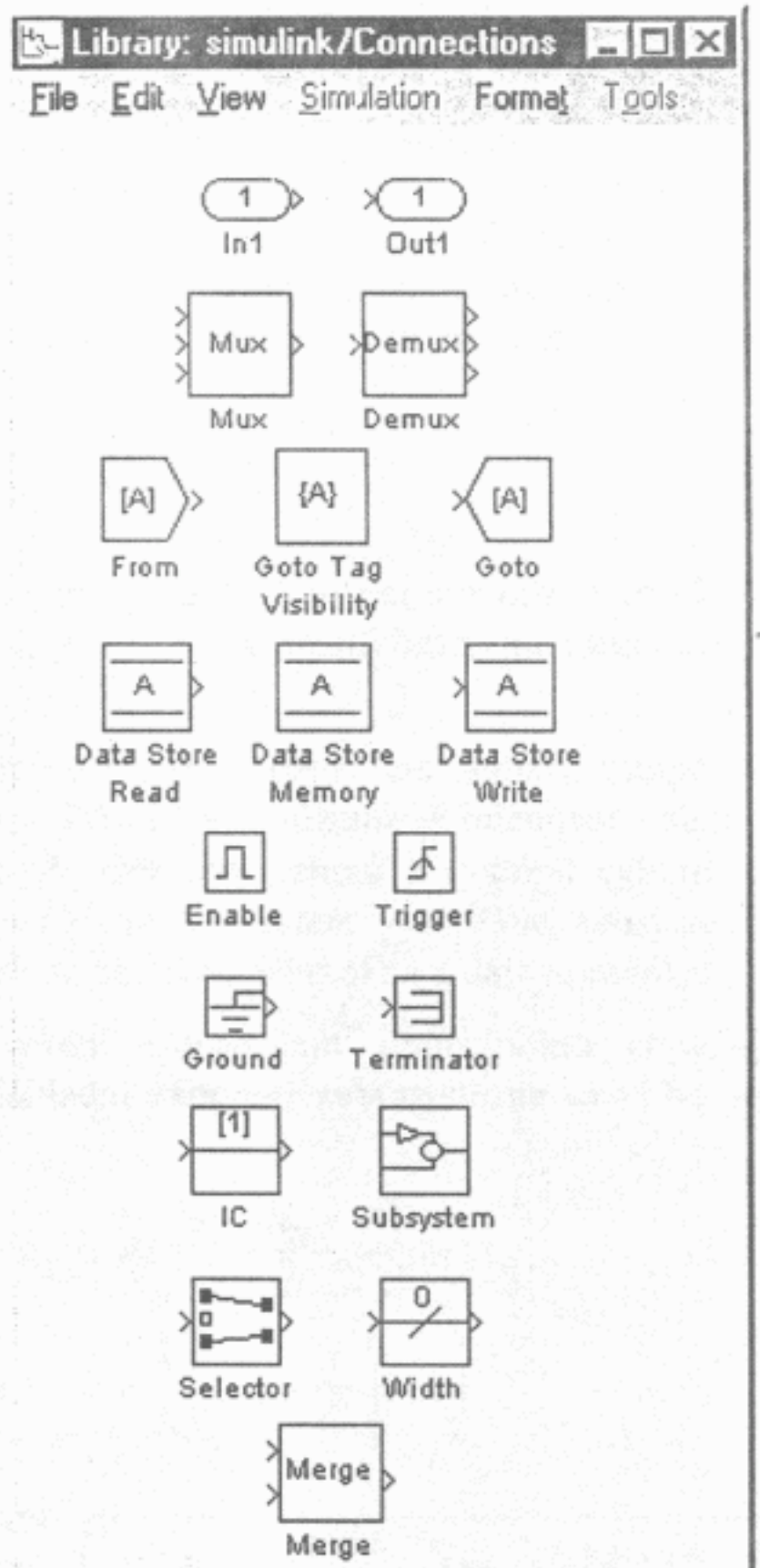


Obrázek 12-9 Knihovna Discrete

Poslední standardní knihovnou je ***Connections*** (obr. 12-11). Tato knihovna obsahuje bloky pro slučování a rozdělování signálů (`Mux` a `Demux`), ukládání a čtení do proměnných MATLABu a další pomocné bloky.



Obrázek 12-10 Knihovna Nonlinear



Obrázek 12-11 Knihovna Connection

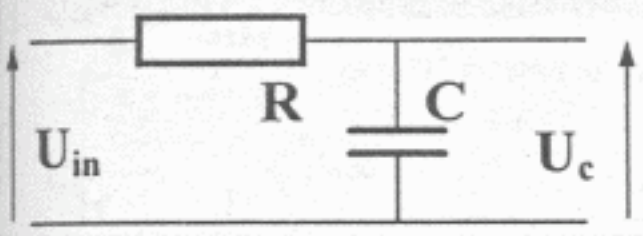
13. Subsystémy a knihovny

Důvody pro tvorbu subsystémů a knihoven jsou obecně stejné jako důvody pro psaní podprogramů a vytváření knihoven v klasickém programování. Subsystémy umožňují přehledný zápis složitého modelu a zjednodušují tvorbu modelu v případě, že se sestává z více stejných nebo podobných částí. Také usnadnění ladění složitých modelů může být nezanedbatelným důvodem pro rozdělení modelu na samostatně odladitelné subsystémy. Tvorba knihoven umožňuje oddělit obecně použitelnou část programu od části specializované na řešení konkrétního problému.

13.1 Subsystémy

Subsystém obvykle zahrnuje určitý logický celek (spolu související operace), který si s okolím vyměňuje relativně omezené množství informací. Tato výměna informací je realizována prostřednictvím vstupních a výstupních proměnných (signálů).

Subsystém v SIMULINKu vytvoříme tak, že sestavíme standardně model a označíme signály vstupní a výstupní pomocí bloků knihovny ***Connections*** s názvy `In` a `Out`. Tyto bloky můžeme pojmenovat - názvy se přenesou do bloku, který vznikne po vytvoření subsystému. Po označení vstupů a výstupů vybereme všechny bloky, které mají být v subsystému, a volbou *Edit - Create Subsystem* vytvoříme subsystém. Vznikne nový blok, který lze kopírovat či např. dvojklikem otevřít do samostatného okna a dále upravovat.



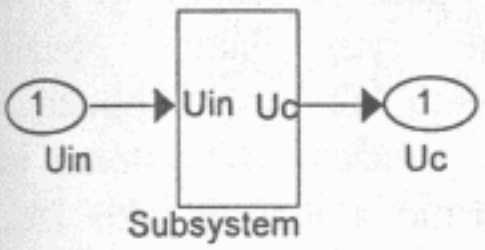
Obrázek 13-1 Schéma RC členu

Ukažme si příklad vytvoření subsystému popisujícího chování elektrického RC členu. Základní zapojení zobrazené na obr.13-1 je popsáno rovnicemi

$$U_{in} = u_R + u_C \quad kde \quad u_R = Ri \qquad \frac{du_C}{dt} = \frac{1}{C} i$$

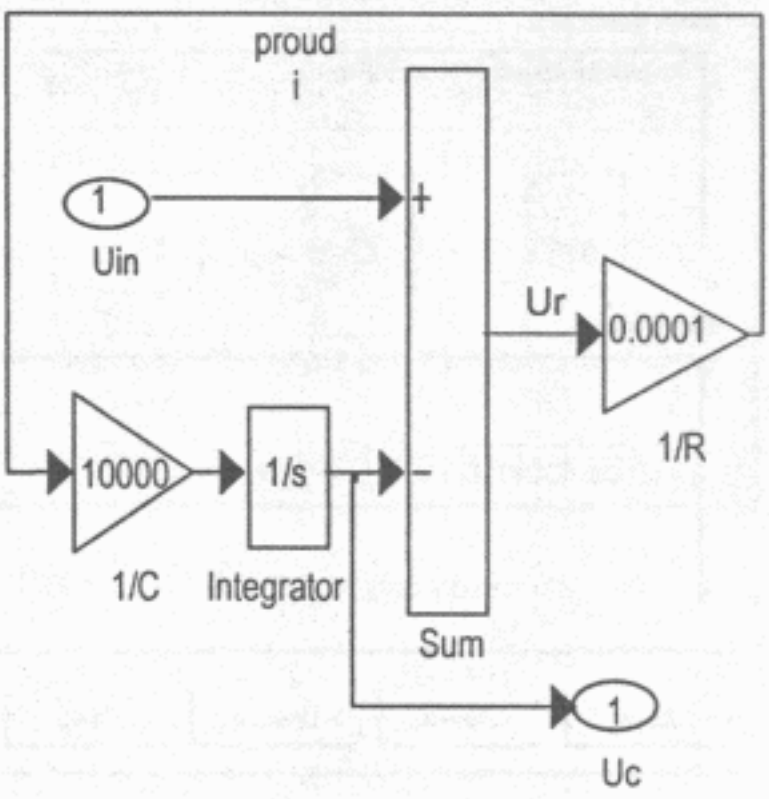
$$U_{in} = Ri + \frac{1}{C}\int_0^t i\,dt \quad \Leftrightarrow \quad i = \frac{1}{R}\left(U_{in} - \frac{1}{C}\int_0^t i\,dt\right)$$

Zvolíme-li např. kapacitu C=100 μF a odpor R=10 kΩ můžeme zapsat model (včetně označení vstupu a výstupu tj. připravený vytvoření subsystému) např. jak je uvedeno na obr. 13-2. Označíme-li nyní všechny bloky a vytvoříme subsystém dostaneme nový blok s jedním vstupem a jedním výstupem jak je ukázáno na obr. 13-3.



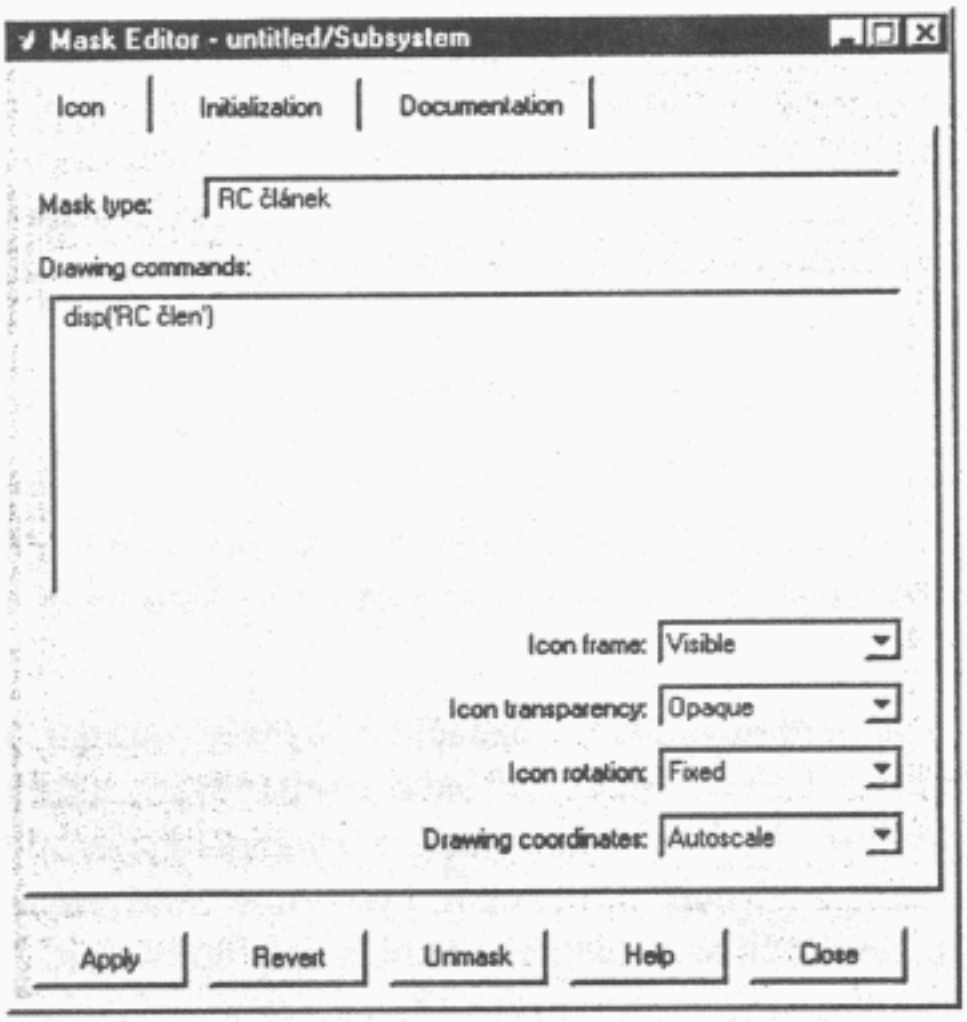
Obrázek 13-3 Subsystém RC člen

Vzniklý subsystém můžeme kopírovat a upravovat. Ovšem chceme-li nyní použít kopie s různými hodnotami parametrů musíme nové kopie editovat - změnit přímo číselné hodnoty. Také doplnění celkového popisu subsystému není možné. Pokud tyto možnosti potřebujeme, SIMULINK nabízí maskování subsystému.



Obrázek 13-2 Model RC členu

13.2 Maska subsystému

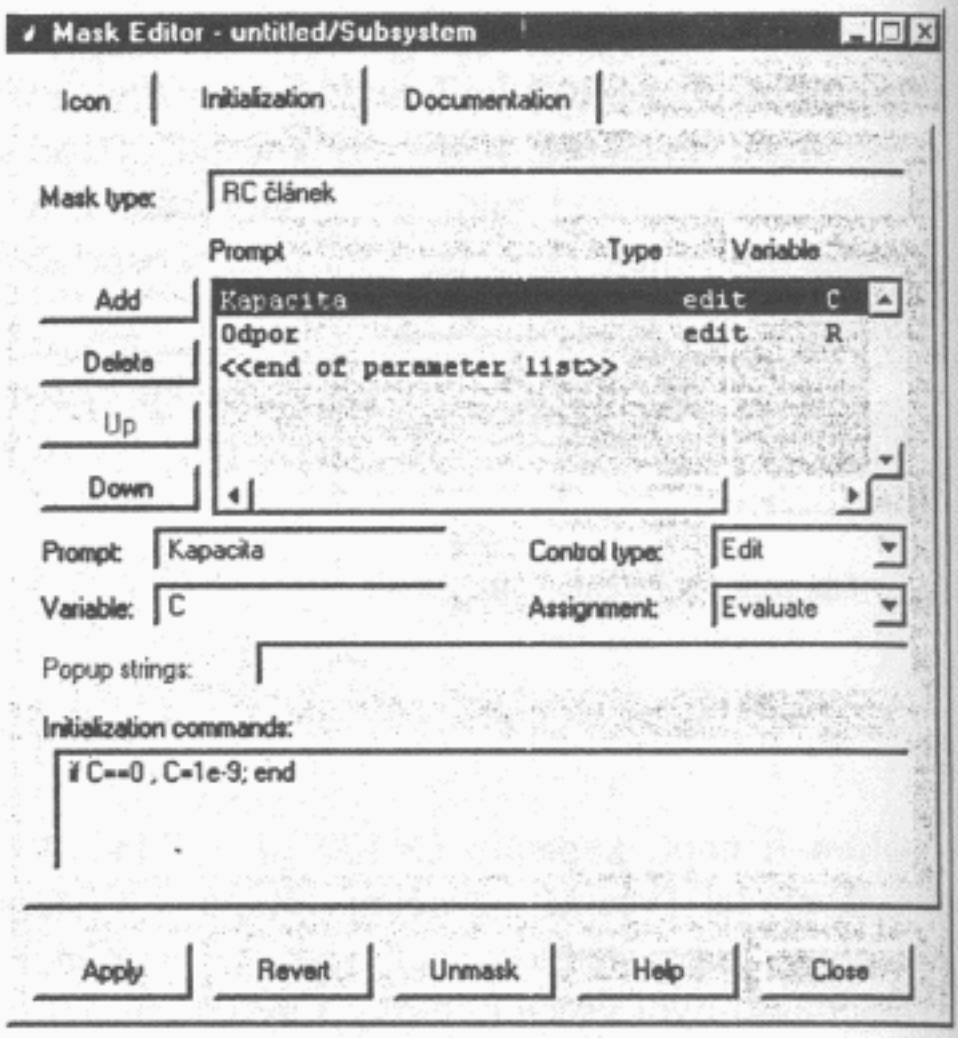


Obrázek 13-4 Mask Subsystem - Icon

Maskování subsystému - vyvolané po vybrání subsystému volbou *Edit - Mask Subsystem* - dovoluje v okně, jež se objeví zadat:

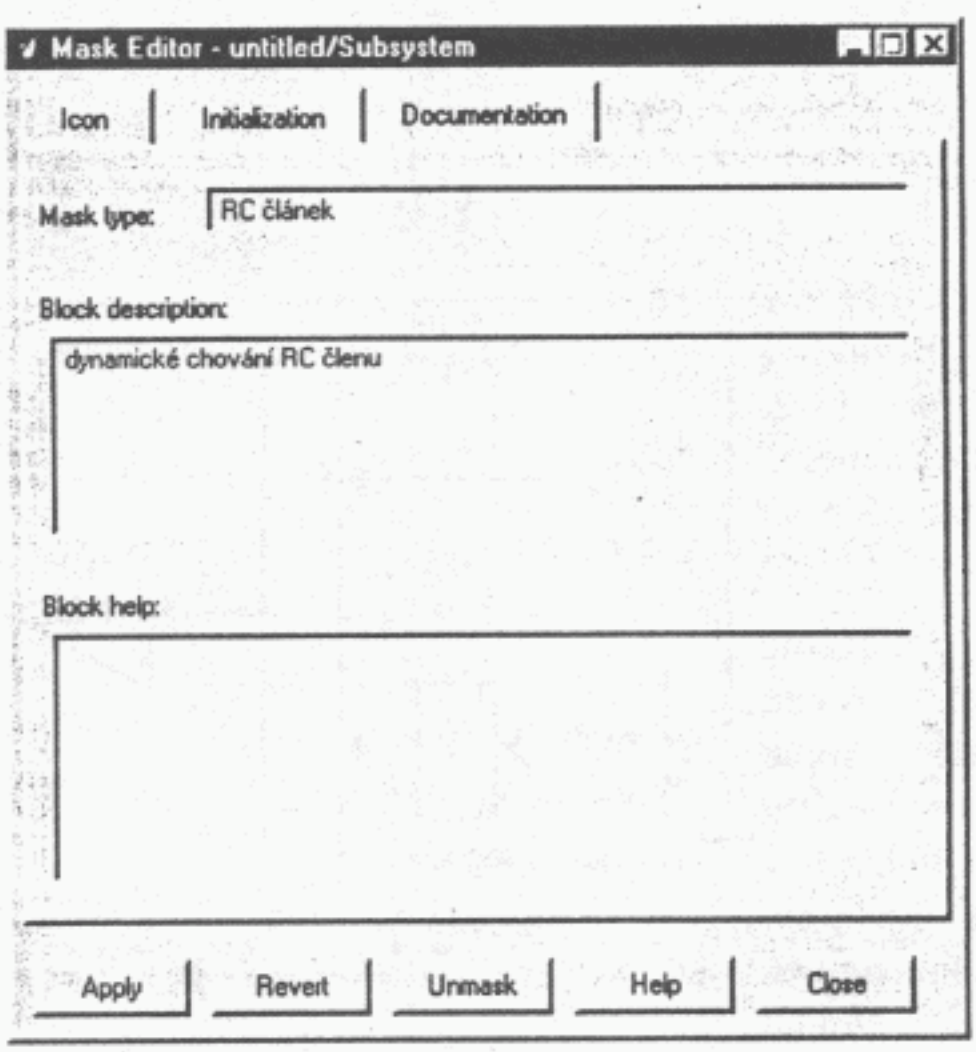
a) ikonu subsystému viz obr. 13-4. Zadává se pomocí standardních příkazů MATLABu. V daném případě je použit pouze jednoduchý příkaz pro výpis textu - **disp**. Je možné použít příkaz **plot** či **patch**

b) názvy a případné implicitní hodnoty proměnných použitých v parametrech bloků maskovaného subsystému, případně příkazy MATLABu, které se mají provést po zadání hodnot - viz obr. 13-5.



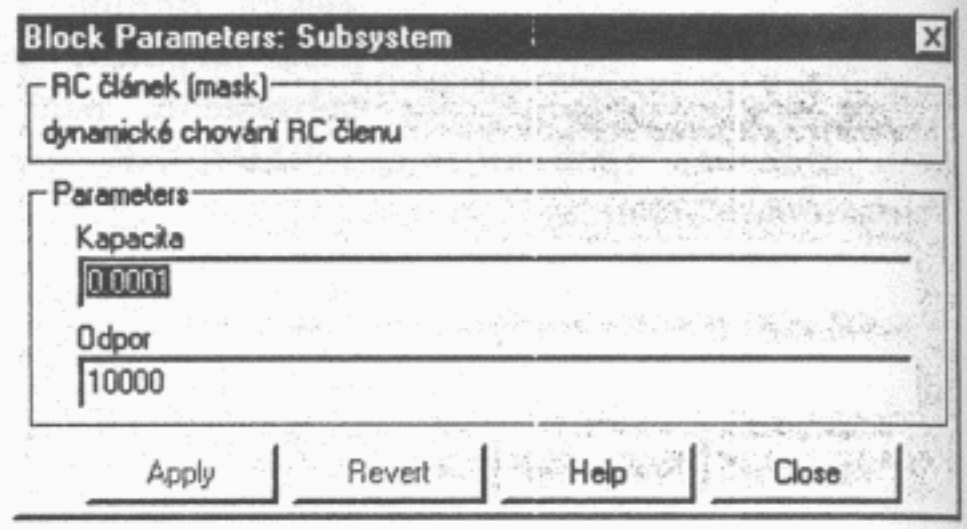
Obrázek 13-5 Mask Subsystem - Initialization

c) popis a případnou nápovědu jak je ukázáno na obr. 13-6.



Obrázek 13-6 Mask Subsystém - Documentation

Chceme-li zadávat parametry pomocí masky tj. přes názvy proměnných musíme tyto názvy použít v parametrech bloků subsystému. Je možné použít i výrazy s těmito názvy. Např. v našem subsystému je použita kapacita C jednou přímo a jednou jako 1/C. Úprava zamaskovaného subsystému je možná volbou *Edit - Look Under Mask*. Dvojklik na zamaskovaném subsystému totiž nyní vyvolá masku tj. objeví se v našem případě okno jako je ukázáno na obr. 13-7.



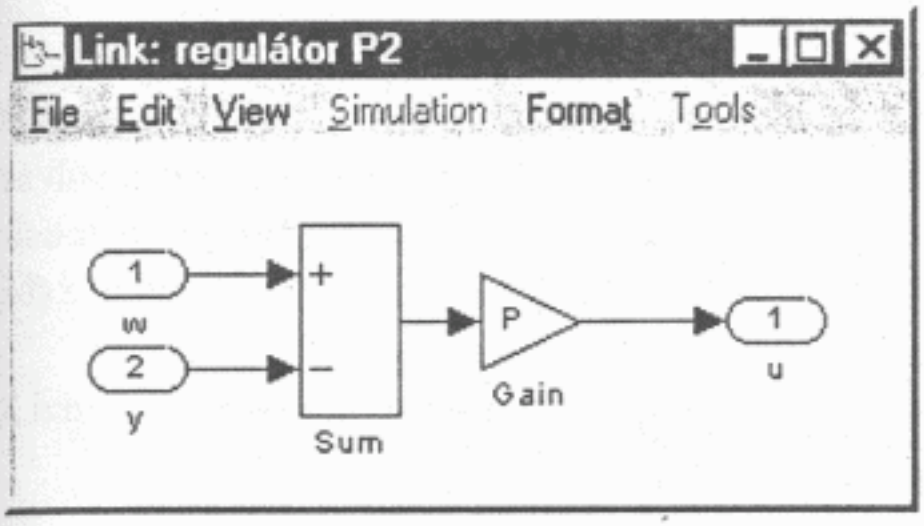
Obrázek 13-7 Vyplnění masky subsystému

13.3 Knihovny

Rozšíření subsystémů představuje knihovna. Vytvářet knihovnu (sadu bloků) má význam v případě, že předpokládáme, že tyto bloky budeme opakovaně používat v různých modelech. Uložení bloků ve formě knihovny umožňuje jejich jednoduché přetahování z této knihovny do tvořeného modelu a automatickou aktualizaci knihovních bloků použitých v modelu v případě, že knihovna byla upravena.

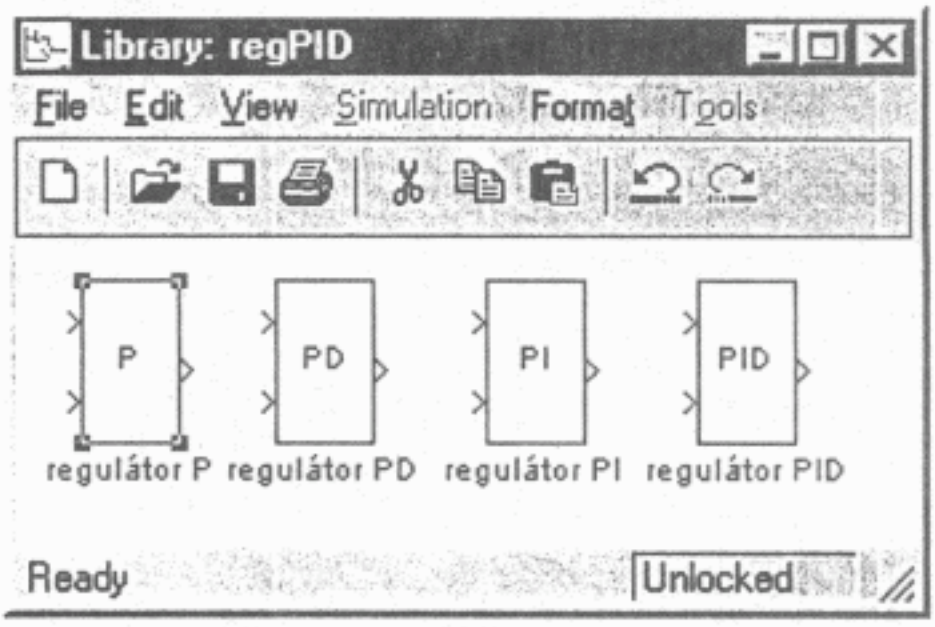
Ukažme si jednoduchý příklad vytvoření knihovny "PID regulátor", která bude obsahovat např. čtyři varianty ideálního spojitého PID regulátoru (P, PI, PD a PID). Knihovnu vytvoříme volbou *File-New-Library*, která založí nové okno pro tvorbu bloků knihovny. Než budeme moci používat bloky z nové knihovny musí být tato nejprve uložena. Pokud otevřeme již existující knihovnu, je chráněna proti nechtěným změnám. Před prováděním změn je nutné ochranu zrušit *Edit-Unlock Library*. Uzavřením okna knihovny se opět ochrana nastaví.

Když zkopírujeme blok z existující knihovny můžeme ho editovat, ale nelze ho zamaskovat (pokud není maskovaný) či upravit masku (pokud je maskovaný). Je vytvořena vazba na příslušný knihovní blok. Aktualizace použitého bloku (dle stavu v knihovně) se provádí při načtení modelu nebo knihovny nebo při použití příkazu *Edit-Update Diagram*.



Obrázek 13-8 Vázaný blok knihovny

Vrátíme-li se k našemu příkladu sestavíme model pro nejjednodušší regulátor P, označíme jako subsystém a zamaskujeme ho - vytvoříme masku pro zadání parametru P. To, že pracujeme s knihovnou se projeví v okamžiku, kdy budeme chtít použít tento blok jako základ dalšího bloku. Když ho budeme chtít zkopírovat, musíme nejdříve knihovnu uložit - např. pod názvem ***regPID***. Otevřeme-li zkopírovaný blok pro úpravy (*Edit-Look Under Mask*) zjistíme, že otevřeme vazbu (link) na původní blok (obr. 13-8). Pokud chceme vytvořit nový nezávislý blok, musíme zrušit vazbu příkazem *Edit-Break Library Link*. Po zrušení vazby můžeme po příkazu *Edit-Look Under Mask* upravit schéma bloku a po příkazu *Edit-Edit Mask* masku nového bloku. Takto můžeme připravit všechny bloky. Výsledné okno knihovny pak obsahuje čtyři bloky jak je ukázáno na obr. 13-9. Příznak Unlocked v pravém dolním rohu indikuje, že jsme stále v režimu úprav knihovny. Po uzavření okna a novém otevření bude příznak nastaven na Locked. Takto připravenou knihovnu můžeme používat v dalších aplikacích.



Obrázek 13-9 Uživatelská knihovna

14. Řešené příklady (SIMULINK)

V kapitole 12 jsou ukázány základní prvky, způsob jejich použití a spuštění simulace výsledného modelu. V kapitole 13 jsou uvedeny postupy jak vytvořit nové uživatelské bloky (subsystémy) a případně uživatelské knihovny. Používání subsystémů je užitečné v případě složitého modelu a tvorba knihoven se vyplatí při předpokládaném častém používání v budoucnu. V této kapitole si ukážeme jak vytvořit schéma modelu a několik příkladů na použití nejdůležitějších bloků. Studium řešených příkladů představuje efektivní cestu jak se naučit se SIMULINKem pracovat.

14.1 Sestavení modelu - výtok z nádrže

Použité bloky: `Integrátor, Sum, Gain, Pulse Generator, Math Function, Scope, Mux`

Problém: nelineární funkce (odmocnina), počáteční stav integrátoru, omezení (saturace)

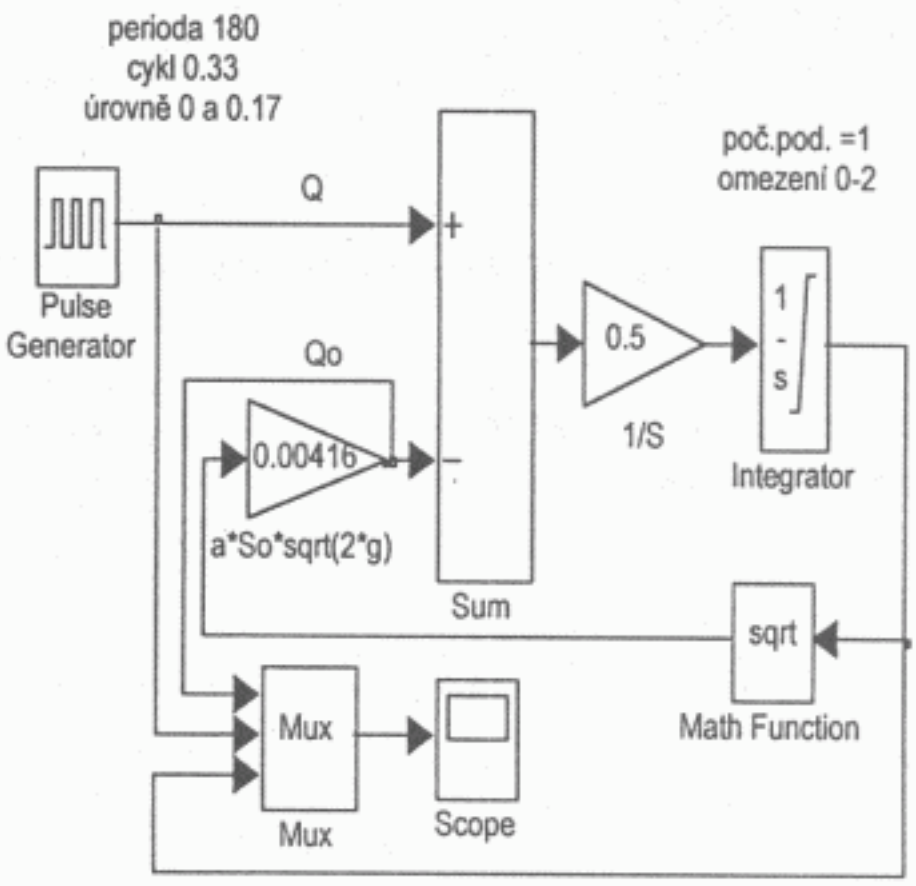
Mějme následující úlohu. Z nádoby o průřezu S=2 m^2 a výšce H=2 m vytéká voda (ρ=1000 kg m^{-3}) otvorem ve dně o průřezu S_o=0.001 m^2. Hodnota hydrodynamického součinitele je α=0.94. Výška hladiny na počátku sledovaného období je h=1 m. Zjistěte, jak se bude měnit v čase výška hladiny h a vytékající množství Q_o. když nebude vždy 1 minutu nic přitékat (Q=0) a pak 2 minuty bude přitékat množství 0.17 m^3 s^{-1}.

Z hmotové bilance systému $\underbrace{\rho Q}_{\text{vstupující hmota}} = \underbrace{\rho Q_o}_{\text{vystupující hmota}} + \underbrace{\frac{d(\rho Sh)}{dt}}_{\text{akumulace hmoty}}$, Torriceliho vztahu pro výtokovou rychlost (s korekcí na skutečnou rychlost pomocí hydrodynamického součinitele, g je tíhové zrychlení) $v_o = \alpha\sqrt{2gh}$ a souvislosti mezi hmotovým průtokem a rychlostí $\rho Q_o = \rho v_o S_o$ snadno sestavíme výslednou nelineární diferenciální rovnici prvního řadu pro závislost výšky hladiny h na přitékajícím množství Q

$$\mathbf{Q} = \alpha S_o\sqrt{2g\mathbf{h}} + S\frac{d\mathbf{h}}{dt} \quad h(t=0)=1 \quad h \in <0,2>$$

Pro potřeby snadného převodu do schématu modelu tuto rovnici upravíme na ekvivalentní tvar implicitní (neznámá h se nachází na obou stranách) integrální rovnice

$$\mathbf{h} = \int_0^t \frac{1}{S}\left(\mathbf{Q} - \underbrace{\alpha S_o\sqrt{2g}}_{konstanta}\sqrt{\mathbf{h}}\right)dt$$



Obrázek 14-1 Model Výtok z nádrže

Tato rovnice již můžeme přímo přepsat do schématu modelu SIMULINKu. Potřebujeme jeden integrátor, jeden součet, dvě zesílení, funkci odmocniny a generátor vstupního signálu. Výstupem integrátoru bude hledaný časový průběh funkce h. Jeho vstup je tvořen součtem dvou signálů (blok `Sum`) vyděleným konstantou S (blok `Gain`). Jeden signál je představován časovým průběhem průtoku Q (blok `Pulse Generator`). Druhý, se znaménkem mínus, je tvořen odmocninou (blok `Sqrt`) z hledané funkce h vynásobené opět konstantou (blok `Gain`). Počáteční stav i omezení hledané funkce h - výšky hladiny - je zajištěno přímo nastavením parametrů integrátoru. Stejně tak obdélníkový průběh vstupního signálu - průtoku Q - je přímo generován knihovním blokem s příslušně nastavenými parametry. Vizualizace výsledku je zajištěna blokem `Scope` (osciloskop). Aby se všechny tři průběhy zobrazovaly v jednom grafu jsou pomocí bloku `Mux` sloučeny do jednoho signálu (vektoru), který je veden do bloku `Scope`. Toto řešení nemusí být optimální v případě, že

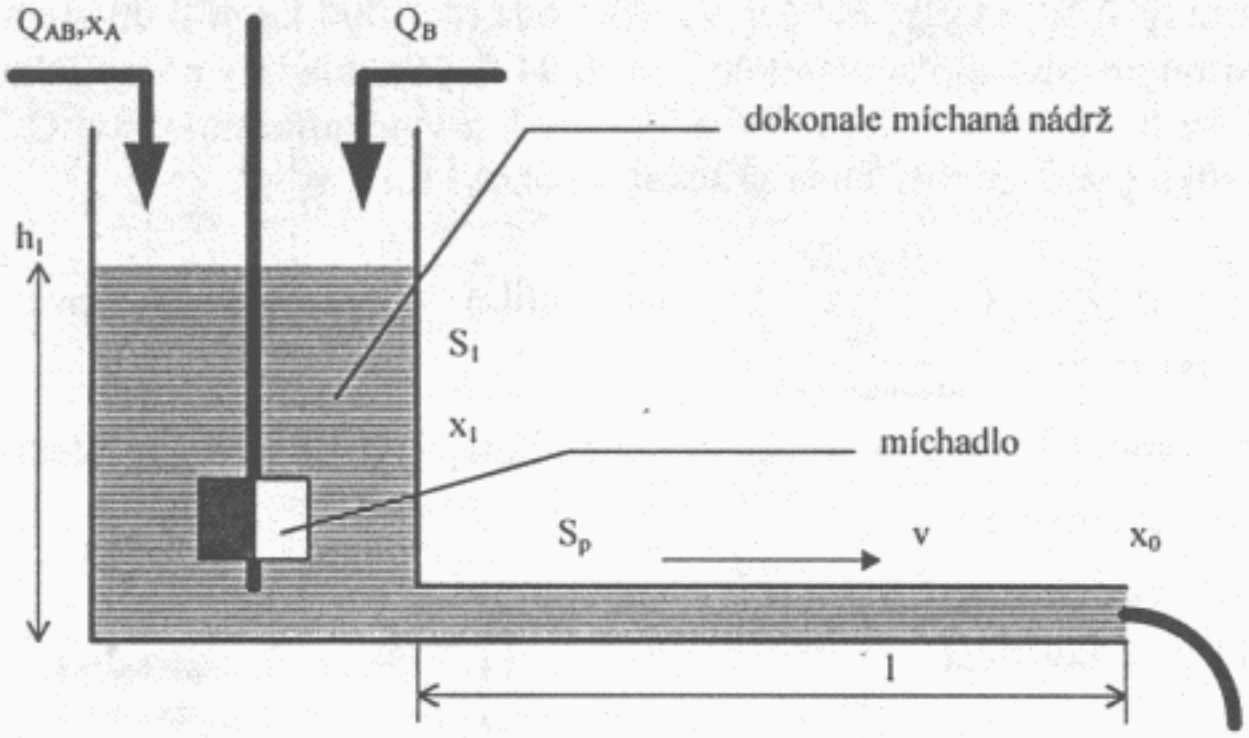
sledované signály jsou značně odlišné. Potom je potřeba měnit měřítko pro zobrazení jednotlivých průběhů. Nyní se musíme rozhodnout jak dlouho budeme systém sledovat. Pokud budeme sledovat po tři periody opakování vstupního signálu nastavíme dobu sledování v parametrech simulace na dobu 540 sec. Potom již je možné řešení spustit. Např. kromě již zmiňovaných způsobů i tlačítkem (s trojúhelníčkem) na nástrojové liště. Jestliže chceme vývoj řešení pozorovat, dvojklikem na bloku `Scope` otevřeme okno osciloskopu před spuštěním simulace. V opačném případě otevřeme okno až po dokončení simulace.

14.2 Koncentrace v nádrži s dlouhým potrubím

Použité bloky: `Integrátor, Sum, Gain, Product, Sine Wave, Fcn, Variable Transport Delay, Scope, Mux`

Problém: součin proměnných, obecná funkce (převrácená hodnota), proměnné dopravní zpoždění, počáteční stav integrátoru, omezení (saturace)

Mějme systém nádrž s odtokem dlouhým potrubím podle obrázku. Do nádrže (průřez S_1) jsou přiváděny dva proudy kapaliny o hmotových průtocích Q_{AB} a Q_B. Proud AB obsahuje dvě látky A a B. Množství látky A je vyjádřeno hmotovým zlomkem $x_A=A/(A+B)$. Proud B obsahuje pouze látku B. Z konce výtokového potrubí vytéká kapalina o výsledné koncentraci x_o. (délka l, průřez S_p, rychlost proudění v). Naším úkolem je zjistit, jak se bude měnit výstupní koncentrace x_o při změnách průtoku kapaliny Q_B.



Obrázek 14-2 Schéma příkladu Koncentrace v nádrži s dlouhým potrubím

Zaveďme předpoklady, že hustoty ρ látek A a B jsou stejné, že nádrž je dokonale promíchávaná[66], že proudění v potrubí je laminární a postup koncentrační změny odpovídá pístovému toku, že kapalina je dokonale nestlačitelná a v celém prostoru potrubí je průměrná průtočná rychlost v. Za výše uvedených předpokladů lze sestavit celkovou hmotovou bilanci nádrže

$$Q_{AB}+Q_B=\underbrace{\rho S_p v}_{odtok\ potrubím}+\frac{d(\rho S_1 h_1)}{dt}$$

a hmotovou bilanci látky A v nádrži

$$Q_{AB}x_A=\rho S_p x_1 v+\frac{d(\rho S_1 h_1 x_1)}{dt}$$

Průměrnou průtočnou rychlost v zjistíme z rovnováhy sil. Síla daná tlakovým rozdílem na obou koncích potrubí a jeho průřezem působí na hmotu kapaliny v potrubí a urychluje ji.

$$\underbrace{S_p\Delta p}_{F}=\underbrace{S_p l\rho\frac{dv}{dt}}_{ma}$$

---

[66] tj. koncentrace uvnitř nádrže je stejná jako koncentrace na výstupu

Rozdíl tlaků na obou koncích potrubí závisí na výšce sloupce kapaliny v nádrži a na ztrátách při proudění v potrubí. Teoretické ztráty při proudění v potrubí popisuje univerzální vzorec Darcy-Weisbachův ve tvaru

$$\Delta p_z = \lambda \frac{l}{d} \rho \frac{v^2}{2}$$

kde pro laminární proudění lze určit parametr λ jako funkci Reynoldsova čísla Re

$$\lambda = \frac{64}{\mathrm{Re}} \quad \mathrm{Re} = \frac{vd}{\gamma} \quad \gamma = \frac{\eta}{\rho}$$

Potom rozdíl tlaků na obou koncích potrubí je dán rozdílem hydrostatických tlaků zmenšeným o tlakovou ztrátu díky proudění a je popsán následujícím vztahem

$$\Delta p = \underbrace{\rho g h_1}_{\text{rozdíl výšek}} - \underbrace{vl \frac{8\eta\pi}{S_p}}_{\text{tlaková ztráta}}$$

Koncentrace látky A vystupující z potrubí odpovídá vstupní koncentraci časově posunuté (o dopravní zpoždění) o hodnotu τ. Tato hodnota je závislá na délce potrubí l a rychlosti v podle vztahu τ=l/v. Výsledná soustava rovnic je

$$\frac{d\mathbf{h}_1}{dt} = \frac{Q_{AB} + Q_B}{\rho S_l} - \frac{S_p}{S_l}\mathbf{v} = 0.0005\left(Q_{AB} + Q_B\right) - 0.000157\mathbf{v}$$

$$\frac{d\mathbf{v}}{dt} = \frac{g}{l}\mathbf{h}_1 - \frac{8\eta\pi}{\rho S_p}\mathbf{v} = 1.962\mathbf{h}_1 - 0.08\mathbf{v}$$

$$\frac{d\mathbf{x}_1}{dt} = \frac{Q_{AB}x_A}{\rho S_l \mathbf{h}_1} - \frac{Q_{AB} + Q_B}{\rho S_l}\frac{\mathbf{x}_1}{\mathbf{h}_1} = 0.0005\left[\frac{Q_{AB}x_A}{\mathbf{h}_1} - \left(Q_{AB} + Q_B\right)\frac{\mathbf{x}_1}{\mathbf{h}_1}\right]$$

$$x_o(t) = x_l(t - \tau) \qquad \tau = {}^{l}\!/_{\mathbf{v}} = {}^{5}\!/_{\mathbf{v}}$$

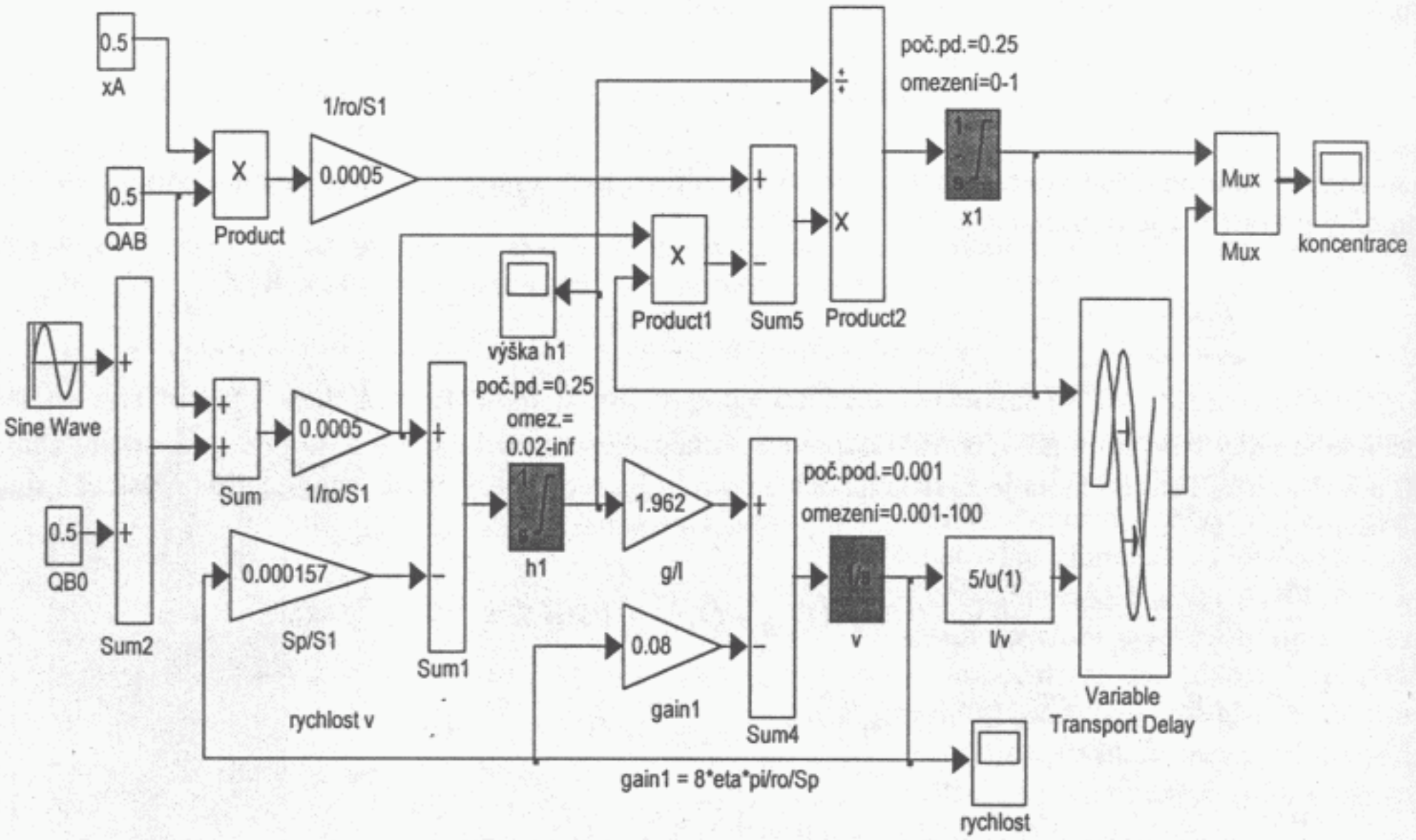
Zvolme konkrétní hodnoty parametrů jak je uvedeno v tabulce

Označení	Hodnota	Rozměr	Význam
Q_{AB}	0.5	kg s^{-1}	hmotový průtok proudu AB
Q_B	0.0-1.0	kg s^{-1}	hmotový průtok proudu B
x_A	0.5	kg kg^{-1}	hmotový zlomek látky A
S_l	2	m^2	plocha průřezu nádrže 1
S_p	0.0003141	m^2	plocha průřezu propojovacího potrubí (průměr 2 cm)
l	5	m	délka propojovacího potrubí
ρ	1000	kg m^{-3}	hustota látek A a B (voda)
η	0.001	Pa s	dynamická viskozita

V následujícím modelu je vstupní průtok Q_B představován časově proměnným signálem popsaným rovnicí

$$Q_B(t) = 0.5 + 0.5\sin\left(\frac{2\pi}{300}t\right)$$

tj. mění se s periodou 300 sec v rozmezí 0-1. Počáteční hodnoty proměnných (výchozí hodnoty integrátorů) jsou nastaveny přibližně do stavu blízkého hodnotám signálů v ustálení. Dobu experimentu je vhodné nastavit na dobu aspoň 600 sec (dvě periody vstupního signálu). Sledovány jsou obě koncentrace, rychlost v potrubí a výška v nádrži.

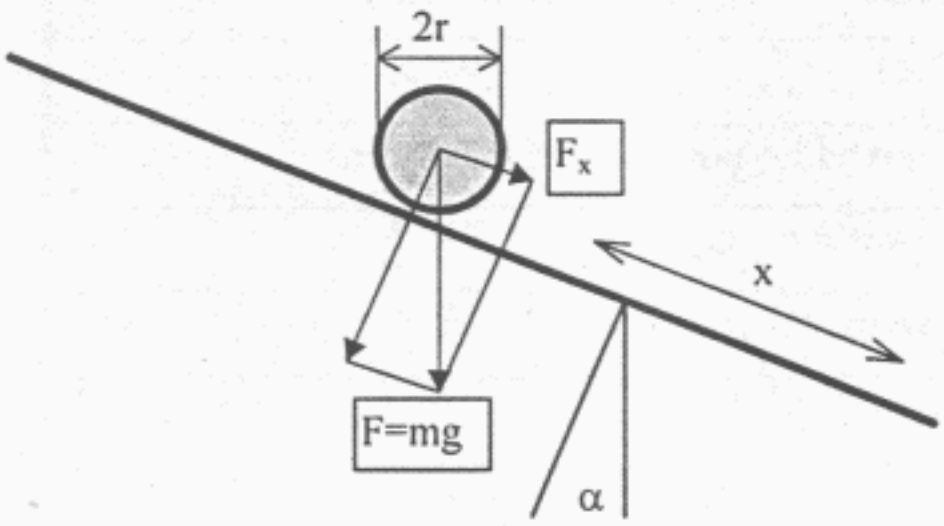


Obrázek 14-3 Model Koncentrace v nádrži s dlouhým potrubím

14.3 Kulička na tyči (Ball on Beam)

Použité bloky: `Integrátor, Sum, Gain, Product, Repeating Sequence, Math Function, Trigonometric Function, Dead Zone, Hit Crossing, Switch, Memory, Sign, Scope, To Workspace`

Problém: externí reset a počáteční podmínky integrátoru, součin proměnných, přepínač, indikace průchodu nulou, zápis do proměnné MATLABu



Obrázek 14-4 Schéma příkladu Kulička na tyči

Na univerzitách technického směru je možné setkat se s laboratorním modelem "Kulička na tyči (Ball on Beam)". Tento model je často používán v laboratořích z předmětu Teorie automatické regulace. Je tvořen kuličkou odvalující se po tyči. Tyč je uprostřed uchycena a je možné ji naklánět o určitý úhel na obě strany. Popišme časovou změnu polohy x kovové kuličky o poloměru r a hustotě ρ v závislosti na časové změně úhlu α naklonění tyče.

Sestavení pohybové rovnice vychází z rovnováhy sil. Toto sestavení bylo popsáno v kapitole 9.3 a zde uvedeme pouze základní a výsledné vztahy.

$$\underbrace{K\left(\frac{dx}{dt}\right)^2}_{odpor} + \underbrace{m\frac{d^2x}{dt^2}}_{posuvný\ pohyb} + \underbrace{\frac{1}{r}J\frac{d^2\varphi}{dt^2}}_{točivý\ pohyb} = \underbrace{mg\sin(\alpha)}_{působící\ síla}$$

$$J=\frac{8}{15}\pi r^5\rho \quad x=r\varphi \quad m=\rho\frac{4}{3}\pi r^3 \quad K=c_x\pi r^2$$

Tato rovnice popisuje ideální chování. Ve skutečnosti je pohyb ovlivňován statickým třením o hodnotě F_{st}. Jde o sílu, která působí proti síle pohyb vyvolávající jen pokud je rychlost nulová. Jinak tento odpor nepůsobí. Definujme její velikost pomocí úhlu α_{st}, o který je potřeba tyč naklonit, aby se kulička začala pohybovat tj. $F_{st}=mg\sin(\alpha_{st})$.

Další komplikací je to, že tyč je obvykle konečné délky a na koncích jsou dorazy. Na těchto dorazech dochází k speciální variantě tlumeného rázu (odrazu kuličky). Pro náš příklad budeme předpokládat, že tento ráz se projeví změnou orientace vektoru rychlosti kuličky (rychlost změní znaménko). Protože ráz není ideální dojde i ke ztrátám, které zahrneme poměrným snížením hodnoty rychlosti tj. $v=-\beta v$.

Zapišme výslednou soustavu rovnic (včetně převodu diferenciální rovnice druhého řádu na soustavu dvou rovnic prvního řádu) pro tyč délky 0.5 m a železnou kuličku (ρ=7800 kg m^{-3}) o poloměru r=0.01 m. Poloha 0 bude ve středu tyče tj. dorazy budou ve vzdálenosti a=±0.25 m. Ztráty při odrazu budou 25% tj. β=0.75. Úhel pomocí kterého definujeme sílu statického tření nechť je α_s=2°.

Je potřeba dát pozor, aby síly odporů působily vždy proti síle vyvolávající pohyb - proto je použita funkce sign.

$$\frac{dx}{dt}=v \qquad poloha$$

$$v=-\beta v \qquad x=-a\vee x=+a \qquad odraz\ v\ poloze\ \pm a$$

$$\frac{dv}{dt}=\begin{cases}\dfrac{mg\sin(\alpha)-sign(v)\times F_{st}-sign(v)\times Kv^2}{m+J/r^2} & v=0\wedge|F_x|>F_{st_s}\\ 0 & v=0\wedge|F_x|\le F_{st}\\ \dfrac{mg\sin(\alpha)-sign(v)\times Kv^2}{m+J/r^2} & v\neq 0\end{cases}$$

$$mg=0.320517 \quad \frac{1}{m+J/r^2}=21.861942 \qquad K=0.00015708$$

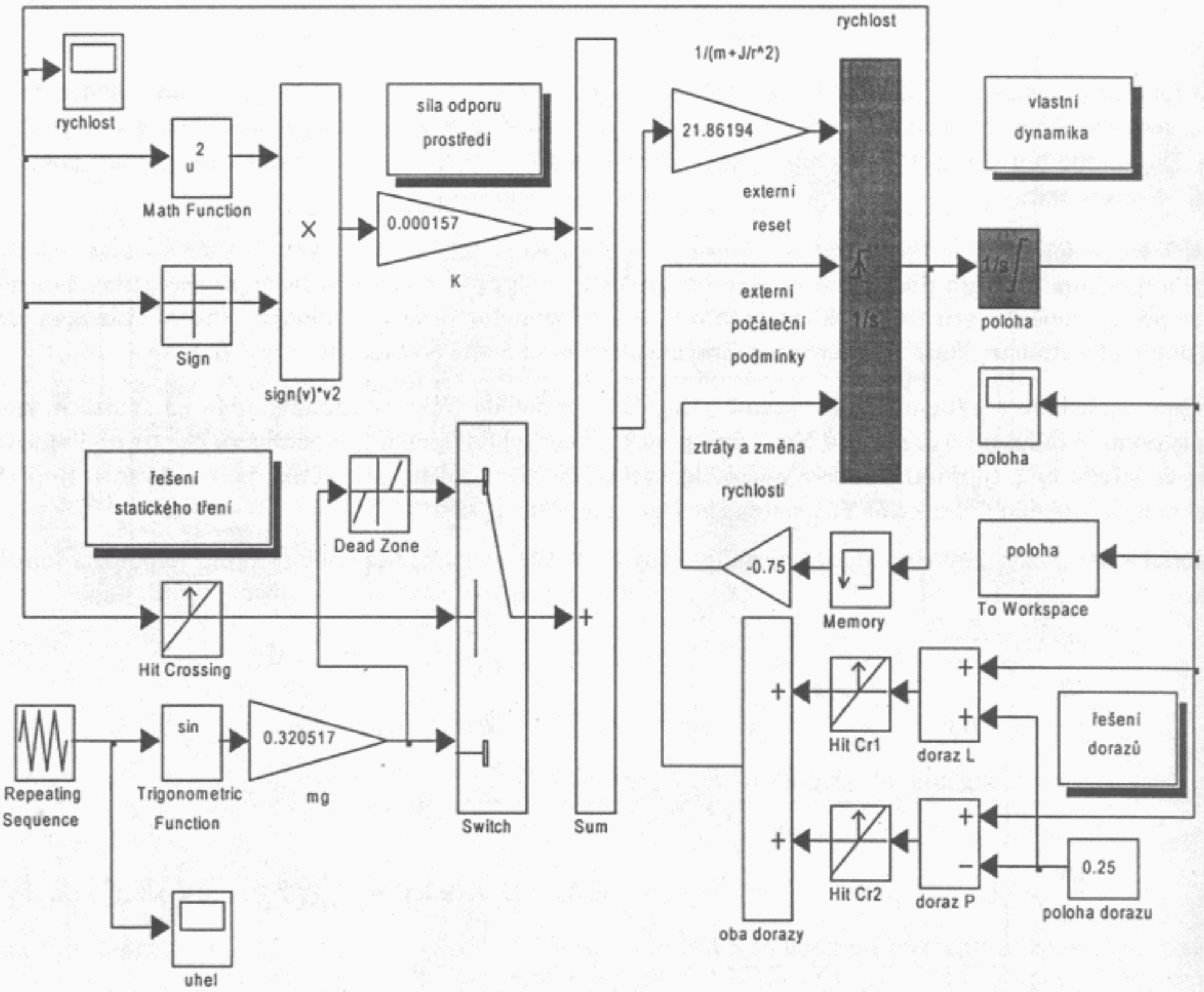
$$F_{st}=mg\sin(\alpha_{st})=0.0111859 \quad a=0.25 \quad \beta=0.75 \quad c_x=0.5$$

Sestavení modelu v SIMULINKu je vhodné provést ve třech krocích. Nejprve je vhodné sestavit model odpovídající ideální rovnici. Zde je pouze nelinearita `Sign` pro zajištění správného smyslu působící síly odporu prostředí. Druhý krok představuje řešení statického tření a ve třetím kroku přidat řešení odrazů.

Pro řešení statického tření použijeme tři nelineární bloky z knihovny ***Nonlinear*** - `Dead Zone`, `Hit Crossing` a `Switch`. První udržuje výstup na nule pro zadaný rozsah vstupního signálu a druhý indikuje průchod vstupního signálu nulou. Blok indikace průchodu nulou vytváří řídicí signál pro blok přepínače, který přepíná hodnotu plnou hodnotu působící síly a hodnotu zmenšenou o sílu statického tření. Jako vstupní signál je použit pilový (trojúhelníkový) signál ve významu úhlu natočení v radiánech s amplitudou ±π/180×15° a periodou 10 sec. začínající z nuly.

Při řešení situace na dorazech je využita možnost externího resetu integrátoru. Pomocí bloků indikace průchodu nulou (`Hit Cr1` a `Hit Cr2`) je indikován okamžik dosažení polohy levého a pravého dorazu. Součet signálů indikací (vzestupná hrana - nastaveno v bloku integrátoru) ovládá externí reset integrátoru. Při

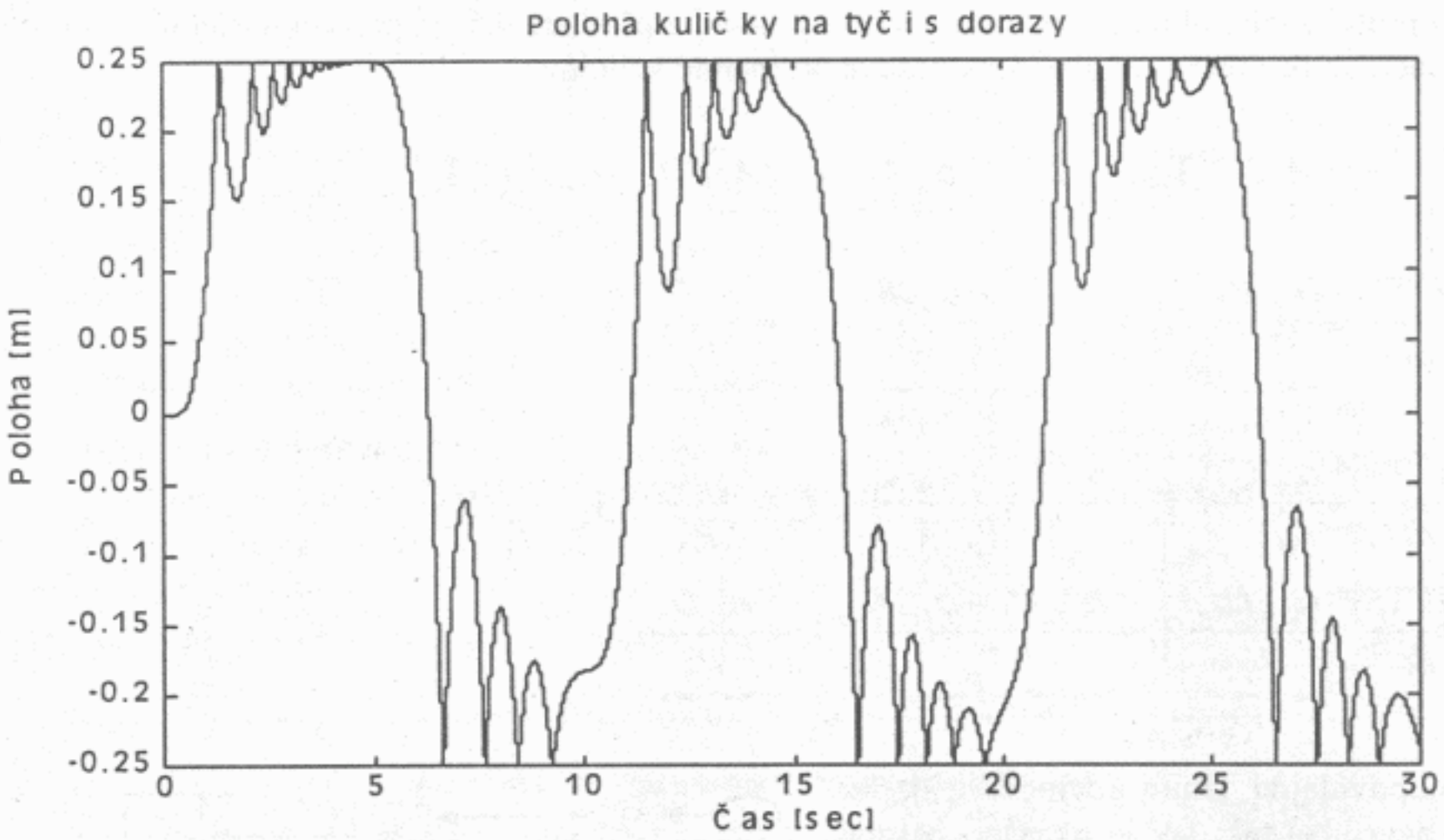
resetu se nastaví nová počáteční hodnota integrátoru rychlosti zadávaná opět externě. Protože se používá hodnota závislá na výstupu integrátoru je nutné použít blok `Memory`. Tento blok udržuje v paměti hodnotu signálu v minulém integračním kroku. Řešení doplňuje omezení na výstupu integrátoru polohy. Výsledný model v SIMULINKu je na obrázku 14-5.



Obrázek 14-5 Model Kulička na tyči

Model uvedený na obr. 14-5 není zcela správný. Pro uvedené hodnoty parametrů sice poskytuje správné výsledky, ale např. při výrazně tlumených odrazech na koncích nebo pomalejší změně natočení tyče by už výsledky neodpovídaly skutečnosti. Problém je v tom, že sice integrátor polohy díky saturaci omezí polohu kuličky, ale integrace rychlosti (její nárůst) pokračuje, ač by měla zůstat rychlost nulová. Výsledkem je, že kulička zůstane na dorazu déle než ve skutečnosti. Korektní řešení úlohy je samozřejmě možné. Jde o řešení logických stavů, na které není SIMULINK stavěný. Zápis řešení by značně znepřehlednil celý model a proto je řešení ponecháno ve tvaru uvedeném na obr. 14-5.

Existence dorazů ovlivňuje dominantním způsobem časový průběh polohy. Zobrazme pro uvedené hodnoty parametrů (pro které dává model korektní výsledky) a start z nulových podmínek časový průběh polohy kuličky během prvních třiceti sekund (tří period) trvání pilového vstupního signálu (úhel naklonění tyč se během 10 sec mění od -15° do +15° a zpět). Na obr. 14-6 je vidět časový průběh polohy kuličky (odskakování) na dorazech umístěných v rozmezí ±0.25 m od středu tyče.



Obrázek 14-6 Časový průběh polohy kuličky

Vykreslení časového průběhu polohy kuličky je realizováno v prostředí MATLABu pomocí příkazu **plot**. Přenos okna osciloskopu (blok `Scope`) do textového editoru např. MS Word je možný pouze kopírováním obrazovky. To je sice možné, ale problémem je černé pozadí obrázku, které při tisku může dělat problémy. Jednodušší je uložit výsledky pomocí bloku `To Workspace` do proměnné a výsledné zpracování udělat v MATLABu.

14.4 Průtokový ohřívač

Použité bloky: `Integrator, Sum, Gain, Slider Gain, In, Out, Product, Repeating Sequence, Zero-Order Hold, Sign, Switch, Scope, Mux`

Problém: subsystém, součin proměnných, generování šířkově modulovaných pulsů, změna hodnoty parametru během simulace

Tento příklad je pokračováním úlohy řešené v kapitole 9.5. Řekněme, že chceme ověřit zda má sestavený matematický model očekávané chování - např. proměnné dopravní zpoždění při změně průtoku, jiné zesílení v různých pracovních bodech apod. Dále můžeme chtít navrhnout regulaci výstupní teploty průtokového ohřívače a ověřit, jak se navržený regulátor vypořádá s uvedenými nelinearitami. Ve všech těchto případech nám pomůže simulace. Využijme tedy matematický model z příkladu 9.5 a sestavme odpovídající simulační model v SIMULINKu. Kromě matematického modelu využijeme i skript na přepočítání parametrů a možnost použít v parametrech bloků SIMULINKu přímo názvy proměnných MATLABu. Na řešení příkladu 9.5 navazujeme i tím, že 8 parametrů modelu je určených optimalizací z naměřených dat. Simulační model sestavme pro dělení na 50 dílů. Výsledný model (jeho simulační schéma) je již značně rozsáhlý a proto využijeme možnost tvorby subsystémů. Protože jednotlivé základní subsystémy (čtveřice diferenciálních rovnic tvořících popis jednoho dílu) jsou stejné, ušetří nám jejich použití i práci s jejich vícenásobnou tvorbou.

Jeden díl průtokového ohřívače (při zachování označení z příkladu 9.5) je popsán následující soustavou rovnic, kde tučným písmem jsou označeny časově proměnné veličiny

$$\frac{d\,{}^{A}\mathbf{T}_i}{dt} = A_1\mathbf{Q}\left({}^{A}\mathbf{T}_{i-1} - {}^{A}\mathbf{T}_i\right) - \underbrace{\left(2\alpha_A A_2 + \alpha_{AB} A_3\right)}_{a_1}{}^{A}\mathbf{T}_i + \underbrace{\alpha_A A_2}_{a_2}\left({}^{A}\mathbf{T}_{i-1} + {}^{A}\mathbf{T}_{i+1}\right) + \underbrace{\alpha_{AB} A_3}_{a_3}{}^{B}\mathbf{T}_i$$

$$\frac{d\,{}^{B}\mathbf{T}_i}{dt} = \underbrace{\frac{\alpha_{AB} B_2}{P_B}}_{b_1}{}^{A}\mathbf{T}_i - \underbrace{\frac{\alpha_{BC} B_1 + \alpha_{AB} B_2}{P_B}}_{b_2}{}^{B}\mathbf{T}_i + \underbrace{\frac{\alpha_{BC} B_1}{P_B}}_{b_3}{}^{C}\mathbf{T}_i$$

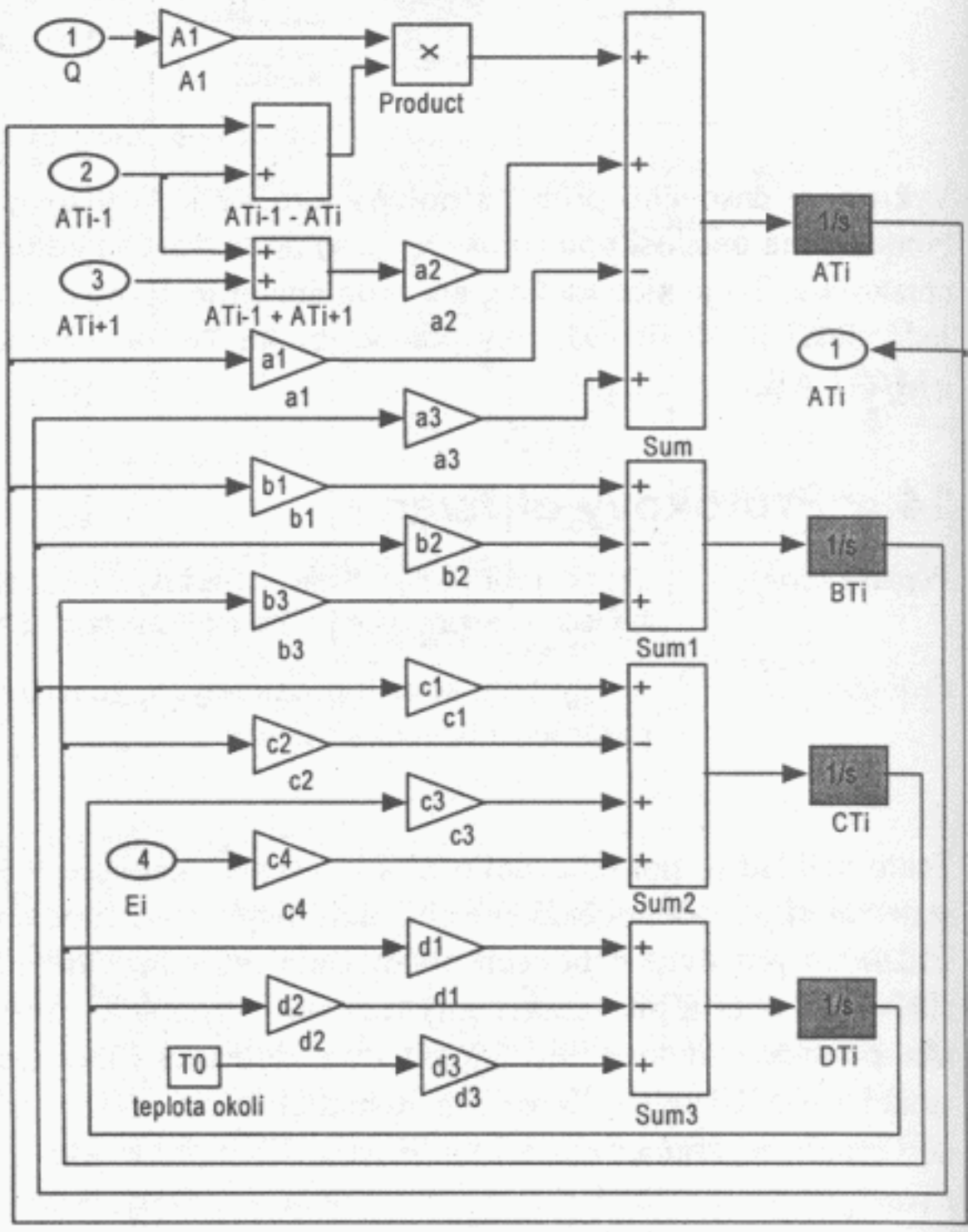
$$\frac{d\,{}^{C}\mathbf{T}_i}{dt} = \frac{\alpha_{BC} C_3}{P_C}{}^{B}\mathbf{T}_i - \frac{\alpha_{CD} C_2 + \alpha_{BC} C_3}{P_C}{}^{C}\mathbf{T}_i + \frac{\alpha_{CD} C_2}{P_C}{}^{D}\mathbf{T}_i + \frac{C_1}{P_C}\mathbf{E}_i$$

$$\frac{d\,{}^{D}\mathbf{T}_i}{dt} = \underbrace{\frac{\alpha_{CD} D_1}{P_D}}_{d_1}{}^{C}\mathbf{T}_i - \underbrace{\frac{\alpha_{CD} D_1 + \alpha_D D_2}{P_D}}_{d_2}{}^{D}\mathbf{T}_i + \underbrace{\frac{\alpha_D D_2}{P_D}}_{d_3} T_0$$

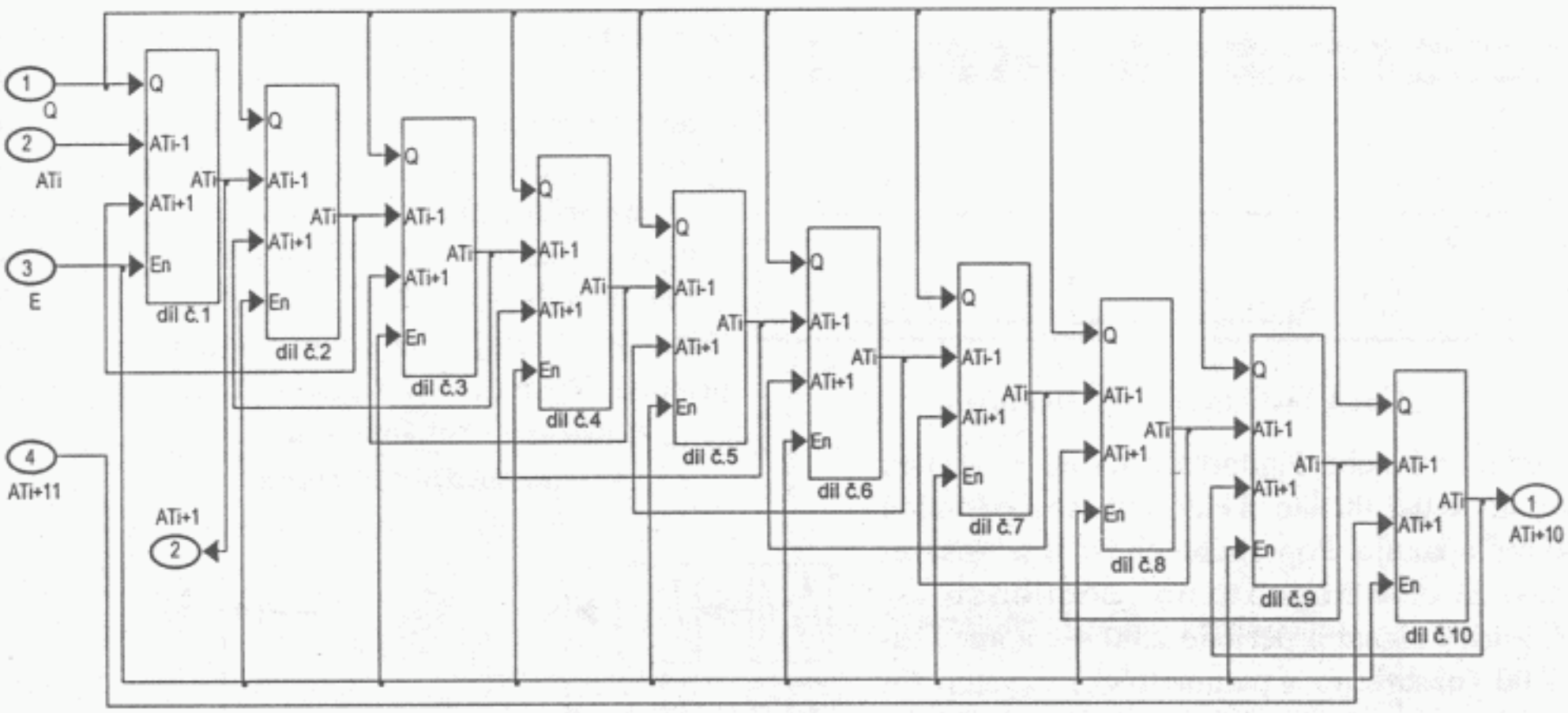
Model odpovídající těmto rovnicím pak lze vyjádřit například tak, jak je ukázáno na obr. 14-7.

V modelu jako parametry bloků figurují přímo názvy proměnných vytvořených pro tyto účely připraveným skriptem. Tento skript musí být samozřejmě spuštěn v okně MATLABu před zahájením simulace. Model je označen jako subsystém tj. vstupy modelu (teplota vody předcházejícího dílu ${}^{A}T_{i-1}$, následujícího dílu ${}^{A}T_{i+1}$, průtok Q a příkon připadající na tento díl E_i) a výstup modelu (teplota vody tohoto dílu ${}^{A}T_i$) jsou vedeny z/do bloků `In/Out` standardní knihovny ***Connection***.

Těchto subsystémů bude potřeba 50. Proto vytvoříme ještě jeden subsystém sdružující 10 těchto dílčích modelů (obr. 14-8). V tomto modelu je dobře vidět propojení vstupů a výstupů jednotlivých základních dílů. Signál průtok vstupuje do všech dílů současně stejně jako signál o aktuálním příkonu (musí mít význam příkonu do jednoho dílu). Hodnota výstupní teploty vstupuje do dílu předcházejícího i následujícího.

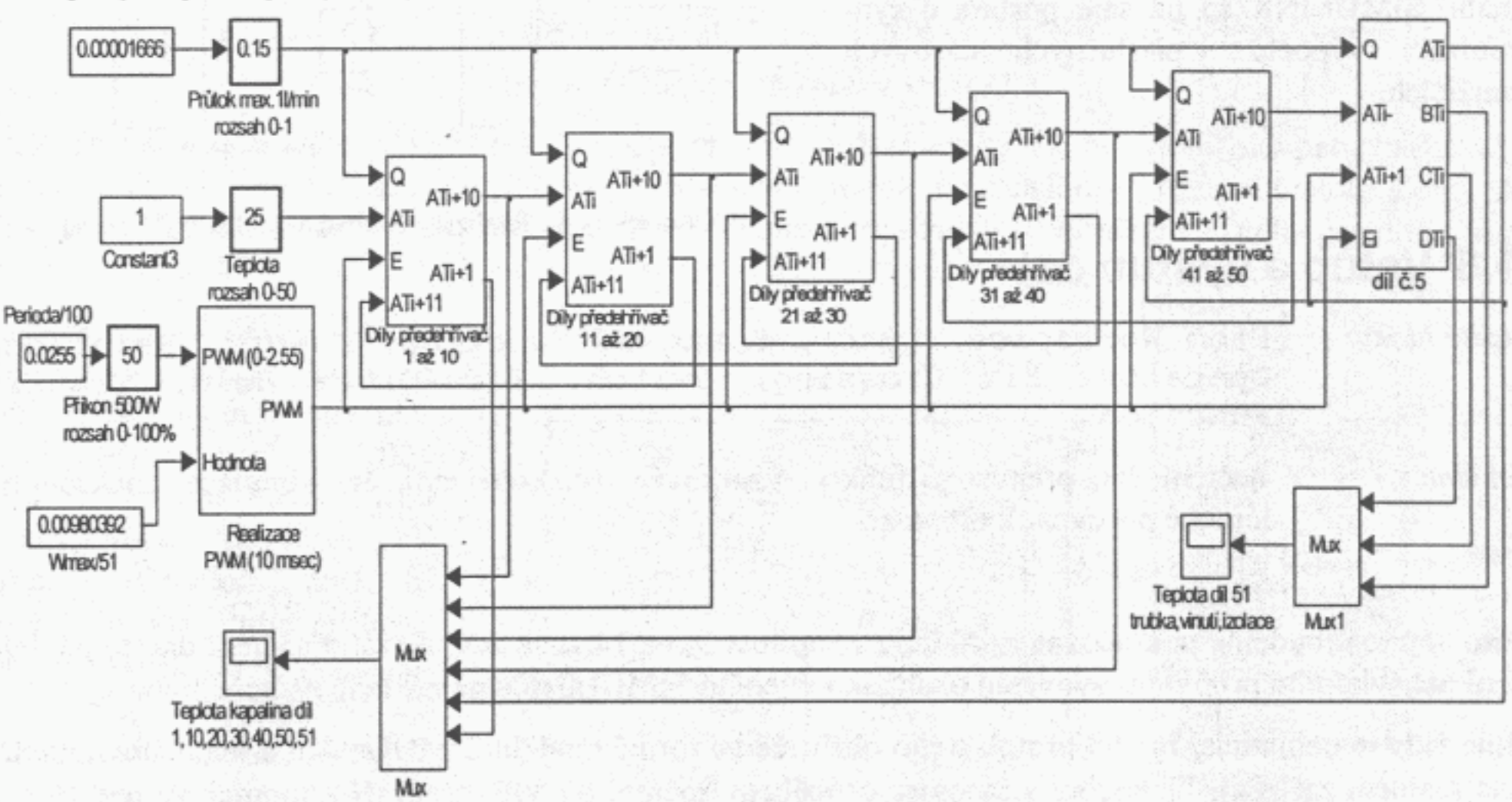


Obrázek 14-7 Jeden díl průtokového ohřívače - model SIMULINK



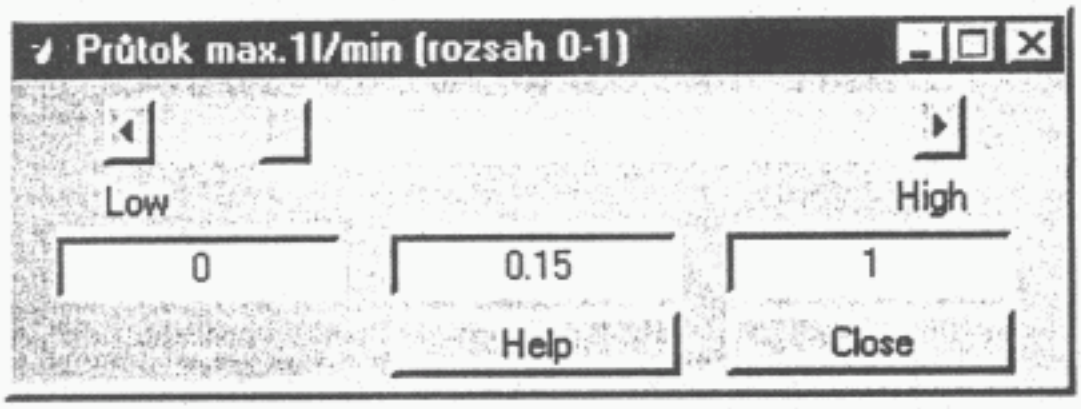
Obrázek 14-8 Deset dílů předehřívače - model SIMULINK

Celkový model pak můžeme znázornit pomocí schématu na obr. 14-9. Zde je přidán ještě jeden modifikovaný díl na konci, aby bylo možné jednoduše sledovat chování teplot všech kapacit jednoho dílu. Do grafů se zobrazují výstupní teploty vody z 10, 20, 30, 40 a 51 dílu a teploty všech kapacit posledního dílu.



Obrázek 14-9 Model průtokový ohřívač (51 dílů)

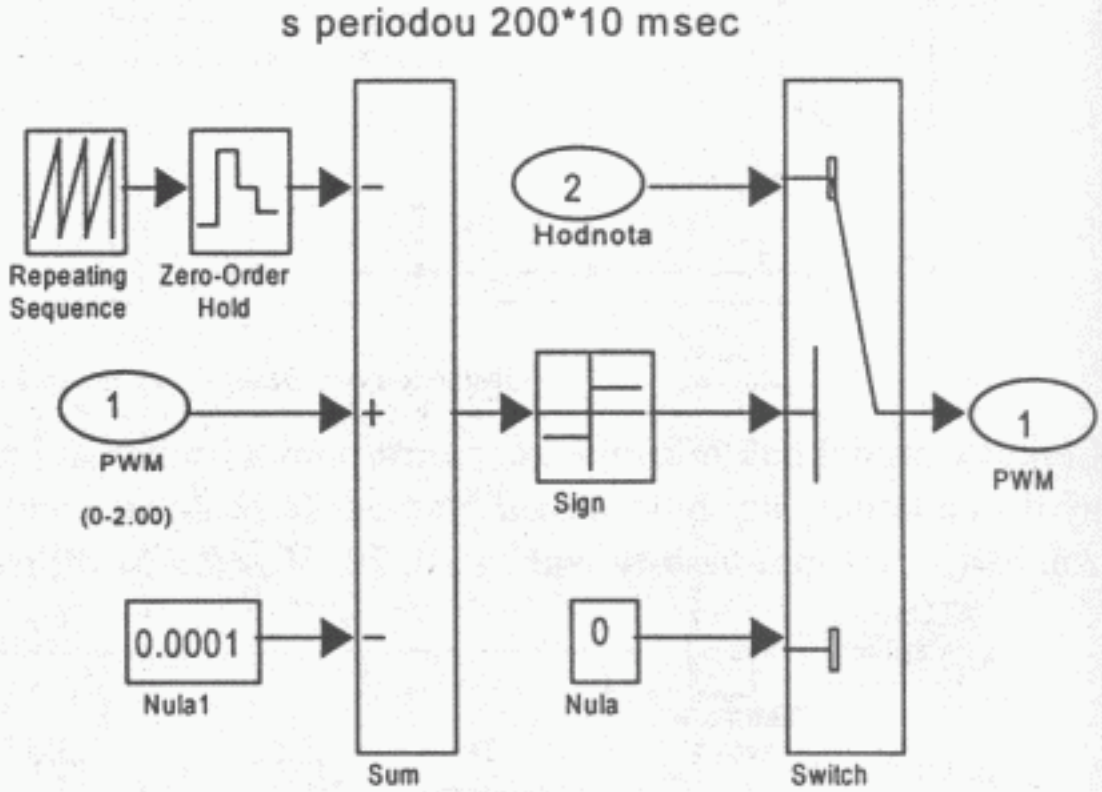
Pro možnost změny vstupních hodnot během průběhu simulace, která v tomto případě již trvá delší dobu, jsou do modelu zahrnuty bloky `Slider Gain` ze standardní knihovny ***Linear*** (označené silnějším obrysovou čarou) umožňující interaktivní změnu zesílení. Okno, pomocí kterého je možné změny zesílení v zadaném rozsahu provádět je na obr. 14-10.



Obrázek 14-10 Okno bloku Slider Gain

vá hodnotu 0 nebo Hodnota a trvání na úrovni Hodnota je 0-2.00 sec. Tento puls se s periodou 2.00 sec opakuje. Popsaného chování je dosaženo tak, že blok `Repeating Sequence` vytváří pilový signál o periodě 2.00 sec a amplitudě 2.00 (nastaveno v parametrech). Výstup tohoto bloku je vzorkován po 10 msec v bloku `Zero-Order Hold`. Odečtení vyvzorkovaného signálu od vstupního signálu a průchod přes blok `Sign` vytváří komparátor, jehož výstupní signál ovládá přepínač dvou úrovní výstupního signálu. SIMULINK se již sám postará o synchronizaci výpočtů v příslušných časových okamžicích.

V modelu je použit ještě blok subsystému vytvářející šířkově modulované pulsy podle hodnoty vstupního signálu. Princip činnosti tohoto subsystému na obr. 14-11 není tak přímočarý a proto ho nyní přiblížíme. Do subsystému vstupují dva vstupní signály. První (PWM) představuje šířku pulsu v hodnotě 0-2.00 v sekundách a druhý (Hodnota), který představuje spínanou hodnotu příkonu. Výstupem je signál (PWM), který nabý-



Obrázek 14-11 Realizace šířkově modulovaných pulsů

14.5 Vstup a výstup dat

Použité bloky: `From Workspace, Clock, Rounding Function, Memory, Relational Operator, Hit Crossing, Switch, Zero-Order Hold, Transfer Function`

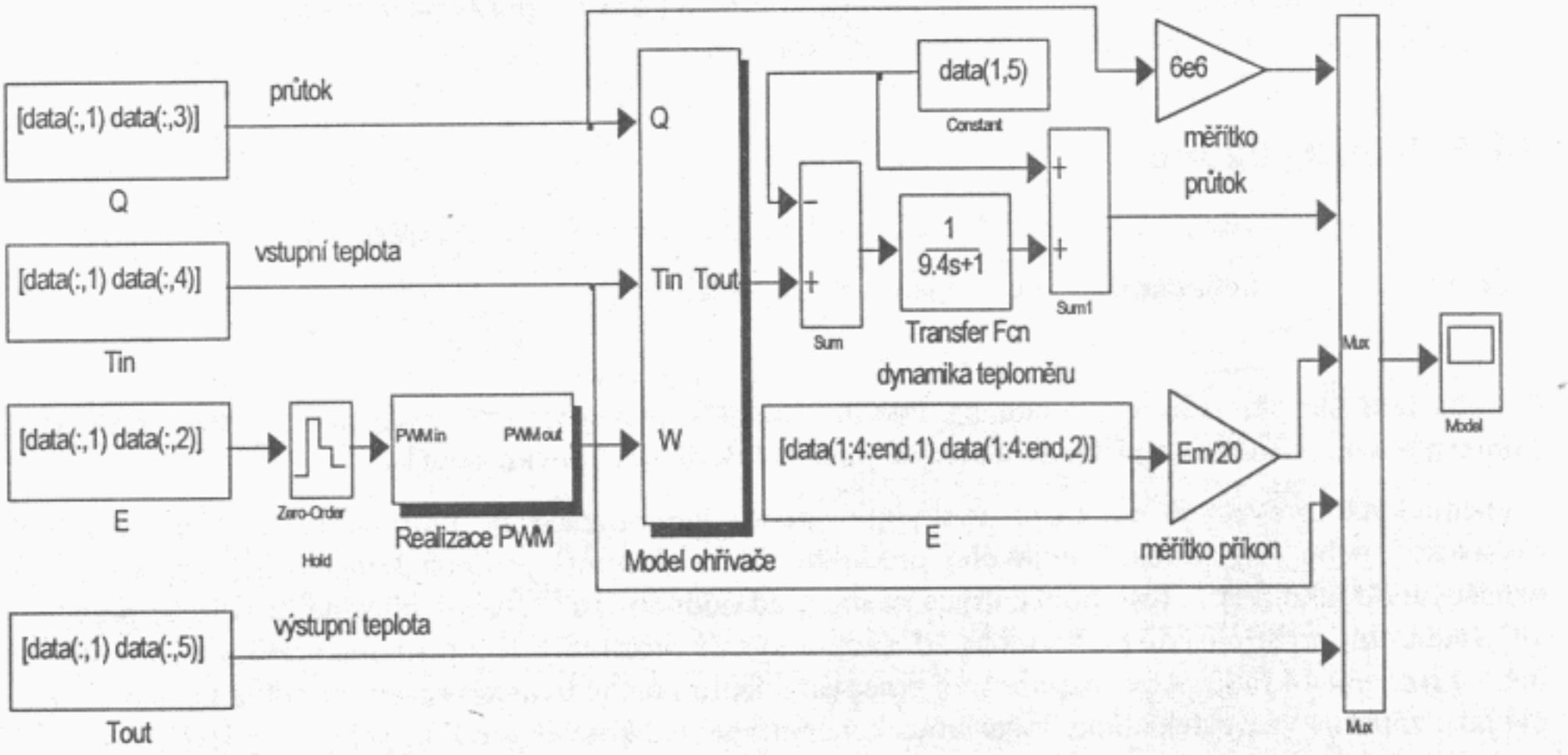
Problém: načtení dat, přenosová funkce, vzorkování (diskretizace), čas simulace, zaokrouhlení, logické porovnání, přepínač

V této kapitole budeme pokračovat v příkladu z kapitoly 9.5 a 14.4. Je zde ukázáno načtení dat do modelu a řešení netriviálního problému vynucení okamžiku výpočtu SIMULINKu na základě načtené hodnoty.

Máme tedy matematický model průtokového ohřívače ve formě modelu SIMULINKu a máme data naměřená na reálném zařízení. Tj. soubor s časovým průběhem (měření po 500 msec) tří vstupních veličin (vstupní teplota, průtok a příkon) a jedné výstupní veličiny (výstupní teploty). Zajímá nás porovnání skutečné výstupní teploty a výstupní teploty vypočtené z modelu. Jako vstupní signál do modelu je potřeba použít změřené hodnoty a jeden signál použít jako vstup zobrazení. Jedna z možných realizací modelu, který náš požadavek splňuje, je na obr. 14-12. V modelu je zahrnuta pomocí bloku `Transfer Function` z knihovny ***Linear*** známá dynamika teplotního čidla (kapacita I.řádu, jednotkové zesílení a časová konstanta 9.4 sec).

Data je možné načítat buď přímo ze souboru na disku (blok `From File`) nebo z matice v pracovním prostoru MATLABu (`From Workspace`) - bloky z knihoven ***Sources*** a ***Sinks***. V případě souboru na disku musí tento obsahovat matici minimálně o dvou řádcích. První řádek obsahuje vzrůstající hodnoty času, druhý - případně další řádky - obsahují hodnoty proměnné v uvedeném čase. Hodnoty proměnných v časových okamžicích výpočtu SIMULINKu jsou automaticky interpolovány. Určitou nevýhodou je to, že formát souboru musí být vnitřní binární formát MATLABu (.mat).

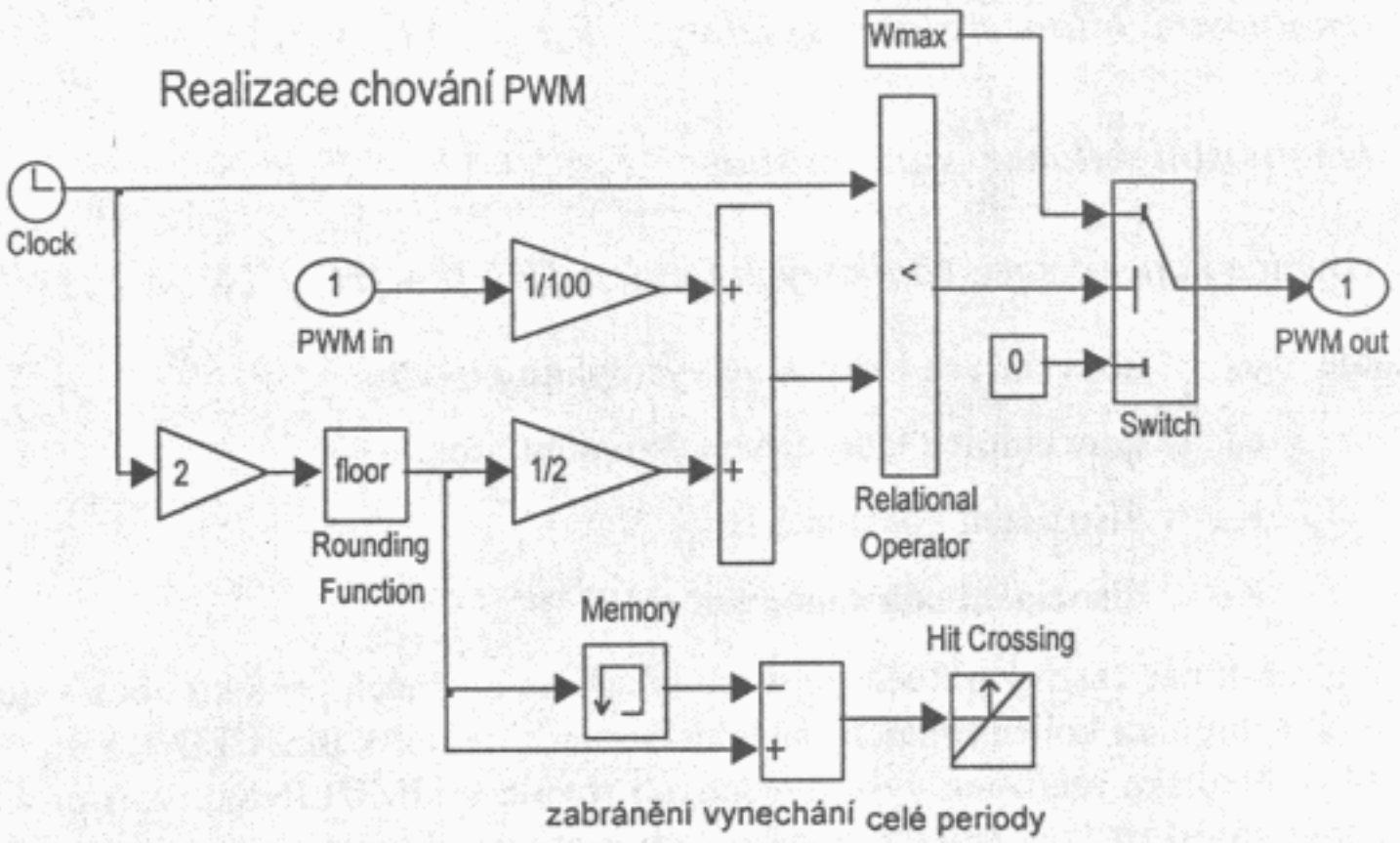
V modelu na obr.14-12 je jako zdroj dat určena matice data. Tato matice v pracovním prostoru MATLABu je tvořená pěti sloupci. Matice musí obsahovat v prvním sloupci (jiná organizace oproti předchozímu případu čtení ze souboru) opět hodnoty vzrůstajícího času. V druhém sloupci a dalších jsou příslušná data. Před



Obrázek 14-12 Čtení dat z matice MATLABu

spuštěním modelu je potřeba naplnit matici data změřenými hodnotami tak, aby první sloupec obsahoval čas měření, druhý sloupec informaci o příkonu, třetí sloupec průtok, čtvrtý sloupec vstupní teplotu a pátý sloupec výstupní teplotu. Všechny sloupce jsou stejně dlouhé a odpovídají času měření v řádku prvním. Čas měření začíná od nuly.

V našem případě činí problémy informace o příkonu. Příkon je řízen šířkově modulovaný pulsy (PWM) o periodě 2000 msec. Hodnoty na zařízení jsou měřeny každých 500 msec synchronně s periodou PWM. Informace o aktuálním příkonu pak vypadá tak, že je zaznamenána aktuální hodnota čítače trvání úrovně topení, který je po 10 msec dekrementován. Maximální hodnota čítače je 200 (topení trvale zapnuto). V tomto případě budou měřené hodnoty příkonu tvořeny opakující se čtveřicí čísel 200,150,100 a 50. Pokud bude požadavek na topení na úrovni 95% maximálního výkonu budou se opakovat čtveřice 190,140,90,40. V případě požadavku topení na 20% budou opakující se čísla 40,0,0,0. Pro jednoduché zjištění průměrné hodnoty příkonu v rámci periody stačí vyčíst hodnotu na začátku každé periody - toto stačí pro vykreslení časového průběhu pro zobrazení výsledků. Pro simulaci je potřeba rekonstruovat skutečný časový průběh příkonu tj. určit okamžik vypnutí topení.



Obrázek 14-13 Vynucení okamžiku výpočtu

K tomuto slouží blok subsystému `Realizace PWM` v obr. 14-12. Realizace subsystému je na obr. 14-13. Sestavení tohoto bloku již není triviální záležitostí. V bloku jsou použity bloky `Clock` (čas výpočtu) z knihovny ***Sources***, `Rounding Function` (zaokrouhlení na celé číslo), `Memory` (paměť hodnoty v minulém okamžiku výpočtu), `Relational Operator` (logické porovnání) a `Hit Crossing` (vynucení výpočtu v daném čase) z knihovny ***Nonlinear***.

14.6 Neutralizace

Použité bloky: `Sum, Dot Product, Gain, Integrator, Algebraic Constraint`

Problém: nelineární rovnice, algebraická smyčka

Protože tato skripta jsou vytvářena na fakultě chemickotechnologické, uveďme poslední příklad právě z oblasti chemie. Na tomto příkladě ukážeme použití řešení algebraické smyčky.

V chemických provozech se často vyskytuje proces neutralizace tj. potřeba dodržet určitou hodnotu "kyselosti" nebo "zásaditosti" nějakého produktu. Tato vlastnost je daná koncentrací vodíkových iontů označovanou jako $[H]^+$. Tato koncentrace se mění od hodnoty 10^{-14} (velmi zásadité prostředí) přes hodnotu 10^{-7} (neutrální prostředí) až po hodnotu 10^0 (velmi kyselé prostředí). Koncentrace vodíkových iontů se tedy mění v rozsahu 14 řádů. Aby se nemuselo pracovat s těmito čísly, byla zavedena jednotka kyselosti prostředí pH jako záporně vzatý dekadický logaritmus koncentrace vodíkových iontů tj. $pH = - \log [H]^+$.

Vrátíme-li se k problému neutralizace znamená to dodržovat určitou hodnotu pH. Vyrábíme-li např. produkt s $pH < 7$ (kyselé prostředí) je možné zvýšit pH přidáním vhodné zásady ($pH > 7$). Následující rovnice popisují časový průběh vývoje koncentrace vodíkových iontů při smíchání proudu látky A (slabé kyseliny) o průtoku F_A a koncentraci c_{A0} s proudem látky B (silné zásady) o průtoku F_B a koncentraci c_{B0} v dokonale míchaném reaktoru o objemu V

hmotnostní bilance slabé kyseliny: $$F_A c_{A0} = (F_A + F_B) c_A + V \frac{dc_A}{dt}$$

hmotnostní bilance silné zásady: $$F_B c_{B0} = (F_A + F_B) c_B + V \frac{dc_B}{dt}$$

rovnice koncentrace vodíkových iontů: $$[H^+]^3 + [H^+]^2 (K_A + c_B) + [H^+]\left[K_A (c_B - c_A)\right] - K_A K_V = 0$$

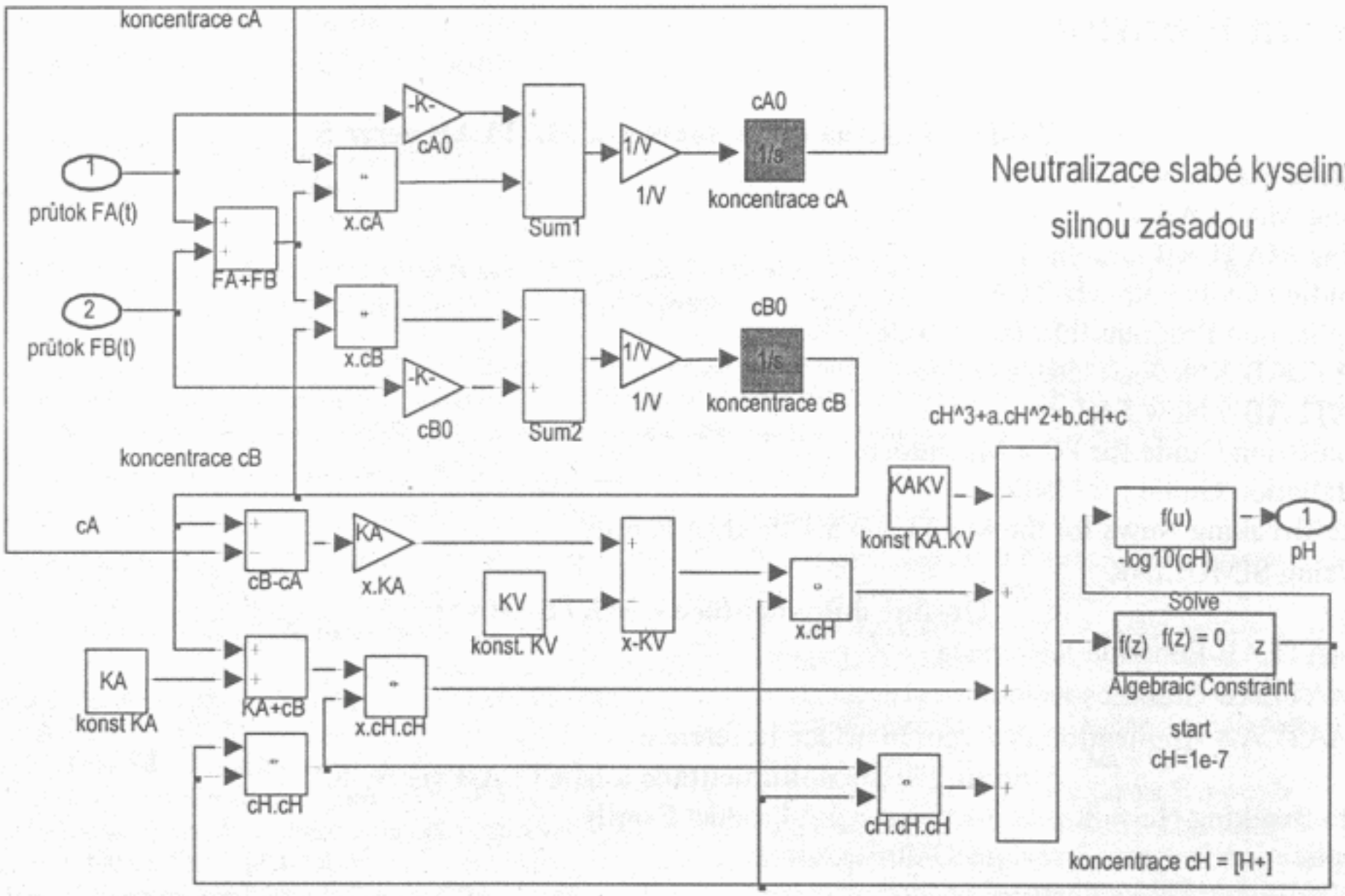
kde c_A koncentrace látky A ve výstupním proudu

c_B koncentrace látky B ve výstupním proudu

K_A disociační konstanta látky A

K_V disociační konstanta vody (10^{-14})

Zajímá-li nás časový průběh pH v závislosti na změnách průtoků obou vstupních proudů při jejich konstantních vstupních koncentracích můžeme sestavit model v SIMULINKu např. v formě subsystému na obr. 14-14. Z hlediska realizace výše uvedených rovnic v SIMULINKu tvoří problém pouze kubická rovnice popisující závislost koncentrace vodíkových iontů na aktuálních koncentracích obou látek a vlastnostech slabé kyseliny vyjádřené její disociační konstantou. Z hlediska simulace jde o algebraickou smyčku. SIMULINK umožňuje řešit tento problém blokem `Algebraic Constraint` z knihovny ***Nonlinear***. Vstupem tohoto bloku je funkční hodnota závislá na výstupu bloku. V průběhu simulace je v každém kroku simulačního výpočtu vyhledána taková hodnota výstupu bloku, aby vstupní funkční hodnota byla nulová. Velký význam v těchto případech hraje volba počáteční hodnoty výstupu bloku při začátku simulace. Konkrétně v našem případě, kdy kubická rovnice může mít tři reálná řešení, špatná volba počáteční hodnoty způsobí, že SIMULINK nedokáže model řešit. Na druhou stranu vhodné volby počátečních hodnot zajistí automatický výběr správného řešení v případě, že algebraická smyčka má více řešení.



Obrázek 14-14 Neutralizace slabé kyseliny silnou zásadou

Tento příklad dobře dokumentuje výpočetní sílu SIMULINKu respektive MATLABu - řešení se pohybuje i při relativně malých změnách průtoků vstupních proudů v rozmezí cca deseti řádů!

Seznam literatury

Základní tištěná dokumentace k MATLAB verze 5

Getting Started with MATLAB
Using MATLAB
Using MATLAB Graphics
Building GUIs with MATLAB
Application Program Interface Guide
MATLAB Notebook User's Guide
MATLAB 5 New Features
Installation Guide for PC a Macintosh
Installation Guide for UNIX
Late-Breaking News for the MATLAB 5.1 Product Family
♠Using SIMULINK

On-line dokumentace k MATLAB verze 5

♣MATLAB Function Reference
♣MATLAB Graphics Reference
♣MATLAB Application Program Interface Reference

Update tištěné dokumentace k MATLAB verze 5.2

Late.Breaking News for the MATLAB 5.2 Product Family
Application Program Interface Guide
♠Simulink 2.1 New Features
♠MATLAB Compiler User's Guide
♠MATLAB C Math Library User's Guide
♠MATLAB C ++ Math Library User's Guide
♠Communications Toolbox New Features Guide (version 1.3)
♠Financial Toolbox User's Guide
♠Fuzzy Logic Toolbox User's Guide
♠Neural Network Toolbox User's Guide
♠Spline Toolbox User's Guide

Update on-line dokumentace k MATLAB verze 5.2 (format PDF)

♣Using MATLAB
♣Using MATLAB Graphics
♣MATLAB Function Reference }includes language and graphics)
♣Application Program Interface Guide
♣MATLAB Notebook User's Guide
♣MATLAB C Math Library User's Guide
♣MATLAB C ++ Math Library User's Guide
♣Using SIMULINK
♣Real-Time Woorkshop User's Guide
♣Stateflow User's Guide
♣Control System Toolbox User's Guide
♣Financial Toolbox User's Guide
♣Frequency Domain System Identification Toolbox User's Guide
♣Image Processing Toolbox User's Guide
♣Mapping Toolbox
♣Partial Differential Equation Toolbox User's Guide
♣Robust Control Toolbox User's Guide
♣Signal Processing Toolbox User's Guide
♣System Identification Toolbox User's Guide
♣Wavelet Toolbox User's Guide

♣DSP Toolbox User's Guide
♣Power System Toolbox User's Guide

♠ *při zakoupení příslušného produktu*
♣ *na společném instalačním CD*

literatura z různých oblastí využívající MATLAB získaná z FTP serveru fy MathWorks (2.3.2000)

MATLAB Based Books for use with MATLAB, MATLAB Toolboxes, and SIMULINK
UPDATED LIST OF BOOKS
You can receive a hardcopy version of this directory listing (including pictures of the book covers) by requesting a copy via info@mathworks.com, or by calling your account representative at The MathWorks at (508) 647-7000. This list is also available on the World Wide Web at http://www.mathworks.com. In addition, new books are announced in The MathWorks quarterly newsletter, MATLAB News and Notes.

List of Books Tue Feb 15 08:44:52 EST 2000

CHEMISTRY/CHEM. ENG.

1. **Duong D. Do**: *Adsorption Analysis: Equilibria and Kinetics.* Imperial College Press, 1998, ISBN 1-86094-137-0
2. **E.J. Karjalainen & U.P. Karjalainen**: *Data Analysis for Hyphenated Techniques.* Elsevier Science, 1996, ISBN 0-444-82237-2
3. **H. Scott Fogler**: *Elements of Chemical Reaction Engineering, 3e.* Prentice Hall, 1999, ISBN 0-13-531708-8
4. **William J. Thomson**: *Introduction to Transport Phenomena.* Prentice Hall, 2000, ISBN 0-13-454828-0
5. **Alkis Constantinides & Navid Mostoufi**: *Numerical Methods for Chemical Engineers with MATLAB Applications.* Prentice Hall, 1999, ISBN 0-13-013851-7
6. **Michael B. Cutlip & Mordechai Shacham**: *Problem Solving in Chemical Engineering with Numerical Methods* Prentice Hall, 1999, ISBN 0-13-862566-2

COMMUNICATIONS

7. **Leo Diaz**: *Antenna Engineering Using Physical Optics: Practical CAD Techniques and Software* Artech House, 1995, ISBN 0-89006-732-5
8. **John G. Proakis & Masoud Salehi**: *Contemporary Communication Systems Using MATLAB* PWS Publishing Company, 1998, ISBN 0-534-93804-3
9. **Leon W. Couch II**: *Digital and Analog Communication Systems, 5e.* Prentice Hall, 1997, ISBN 0-13-522583-3
10. **Andy Bateman**: *Digital Communications: Design for the Real World.* Addison-Wesley, 1999, ISBN 0-201-34301-0
11. **Peter Gerdsen & Peter Kröger**: *Digitale Signalverarbeitung in der Nachrichtenübertragung: Elemente, Bausteine, Systeme und ihre Algorithmen.* Springer-Verlag, 1997, ISBN 3-540-61194-0
12. **John Carroll / James Whiteaway & Dick Plumb**: *Distributed Feedback Semiconductor Lasers.* IEE, 1998, ISBN 0-85296-917-1
13. **Leon W. Couch II**: *Modern Communication Systems: Principles and Applications.* Prentice Hall, 1995,ISBN 0-02-325286-3

CONTROLS & SYSTEMS

14. **Jerzy Moscinski & Zbigniew Ogonowski**: *Advanced Control with MATLAB and SIMULINK* Prentice Hall, 1995, ISBN 0-13-309667-X
15. **Stanley M. Shinners**: *Advanced Modern Control System Theory and Design* John Wiley & Sons, Inc., 1998, ISBN 0-471-31857-4
16. **Joseph Z. Ben-Asher & Isaac Yaesh**: *Advances in Missile Guidance Theory.* AIAA, 1998, ISBN 1-56347-275-9
17. **Luigi Mangiacasale**: *Airplane Control Systems: μ-Synthesis with MATLAB.* Edizioni Levrotto & Bella - Torino - Italy, 1996
18. **Chi-Tsong Chen**: *Analog and Digital Control System Design: Transfer-Function, State-Space, and Algebraic Methods.* Oxford University Press, 1993, ISBN 0-03-094070-2
19. **Benjamin C. Kuo**: *Automatic Control Systems, 7e* John Wiley & Sons, Inc., 1995, ISBN 0-471-36608-0
19. **Theodore E. Djaferis**: *Automatic Control: The Power of Feedback.* PWS Publishing Company, 1998, ISBN 0-534-95051-5
20. **Wodek Gawronski**: *Balanced Control of Flexible Structures.* Springer-Verlag, 1996, ISBN 3-540-76017-2
21. **Paul H. Lewis & Chang Yang**: *Basic Control Systems Engineering.* Prentice Hall, 1997, ISBN 0-13-597436-4
22. **Hadi Saadat**: *Computational Aids in Control Systems Using MATLAB.* McGraw-Hill, 1993, ISBN 0-07-911358-3
23. **Goerge A. Perdikaris**: *Computer Controlled Systems: Theory and Applications* Kluwer Academic Publishers, 1991, ISBN 0-7923-1422-0
24. **Karl J. Äström & Bjorn Wittenmark**: *Computer-Controlled Systems: Theory and Design, 3e* Prentice Hall, 1997, ISBN 0-13-314899-8
25. **Robert Strum & Donald Kirk**: *Contemporary Linear Systems Using MATLAB 4.0* PWS Publishing Company, 1994, ISBN 0-534-947107
26. **Arthur E. Bryson, Jr.**: *Control of Spacecraft and Aircraft.* Princeton University Press, 1994, ISBN 0-691-08782-2
27. **Bahram Shahian & Michael Hassul**: *Control System Design Using MATLAB.* Prentice Hall, 1993, ISBN 0-13-174061-X
28. **Norman S. Nise**: *Control Systems Engineering, 2e* John Wiley & Sons, Inc., 1995, ISBN 0-471-36736-2
29. **William C. Messner & Dawn M. Tilbury**: *Control Tutorials for MATLAB and Simulink: A Web-Based Approach* Addison-Wesley, 1998, ISBN 0-201-47700-9
30. **J. Brzózka**: *Cwiczenia Z Automatyki W Matlabie I Simulinku.* MIKOM, 1997, ISBN 83-87102-25-3
31. **Katsuhiko Ogata**: *Designing Linear Control Systems with MATLAB.* Prentice Hall, 1994, ISBN 0-13-293226-1
32. **Gene F. Franklin / J. David Powell & Michael L. Workman**: *Digital Control of Dynamic Systems, 3e* Addison-Wesley, 1998, ISBN 0-201-82054-4
33. **Charles L. Phillips & H. Troy Nagle**: *Digital Control System Analysis and Design, 3e.* Prentice Hall, 1995, ISBN 0-13-309832-X
34. **Farzad Nekoogar & Gene Moriarty**: *Digital Control Using Digital Signal Processing.* Prentice Hall, 1999, ISBN 0-13-089103-7
35. **Richard J. Vaccaro**: *Digital Control: A State-Space Approach.* McGraw-Hill, 1995, ISBN 0-07-066781-0

36. **Tom T. Hartley / Guy O. Beale & Stephen P. Chicatelli**: *Digital Simulation of Dynamic Systems: A Control Theory Approach.* Prentice Hall, 1994, ISBN 0-13-219957-2
37. **Howard Kaufman / Itzhak Barkana & Kenneth Sobel**: *Direct Adaptive Control Algorithms: Theory and Applications, 2e* Springer-Verlag, 1998, ISBN 0-387-94884-8
38. **Katsuhiko Ogata**: *Discrete-Time Control Systems, 2e* Prentice Hall, 1995, ISBN 0-13-034281-5
39. **J. Lowen Shearer / Bohdan T. Kulakowski & John F. Gardner**: *Dynamic Modeling and Control of Engineering Systems, 2e.* Prentice Hall, 1997, ISBN 0-13-356403-7
40. **Wodek Gawronski**: *Dynamics and Control of Structures: A Modal Approach.* Springer-Verlag, 1998, ISBN 0-387-98527-1
41. **Kemin Zhou & John C. Doyle**: *Essentials of Robust Control.* Prentice Hall, 1998, ISBN 0-13-525833-2
42. **Yaakov Bar-Shalom & Xiao Rong Li**: *Estimation and Tracking: Principles, Techniques, and Software* Artech House, 1993, ISBN 0-89006-643-4
43. **Gene F. Franklin / J. David Powell & Abbas Emami-Naeini**: *Feedback Control of Dynamic Systems, 3e* Addison-Wesley, 1994, ISBN 0-201-52747-2
44. **Dean K. Frederick & Joe H. Chow**: *Feedback Control Problems Using MATLAB and the Control System Toolbox* PWS Publishing Company, 1995, ISBN 0-534-93798-5
45. **Charles L. Phillips & Royce D. Harbor**: *Feedback Control Systems, 3e.* Prentice Hall, 1996,ISBN 0-13-371691-0
46. **Luigi Mangiacasale**: *Flight Mechanics of a µ-Airplane: With a MATLAB Simulink Helper* Edizioni Libreria CLUP - Milano - Italy, 1998
47. **R. Brockhaus**: *Flugregelung* Springer-Verlag, 1994, ISBN 3-540-55416-5
48. **John S. Bay**: *Fundamentals of Linear State Space Systems.* WCB/McGraw-Hill, 1999, ISBN 0-256-24639-4
49. **Alberto Cavallo / Roberto Setola & Francesco Vasca**: *Guida Operativa a MATLAB, SIMULINK e Control Toolbox* Liguori Editore, 1994, ISBN 88-207-2474-X
50. **Louis V. Schmidt**: *Introduction to Aircraft Flight Dynamics.* AIAA, 1998, ISBN 1-56347-226-0
51. **John L. Junkins & Youdan Kim**: *Introduction to Dynamics and Control of Flexible Structures.* AIAA, 1993, ISBN 1-56347-054-3
52. **Edward Scheinerman**: *Invitation to Dynamical Systems.* Prentice Hall, 1996, ISBN 0-13-185000-8
53. **Maciej Szymkat**: *Komputerowe wspomaganie w projektowaniu ukladów regulacji* Wydawnictwa Naukowo- Techniczne, 1993, ISBN 83-204-1655-8
54. **John J. D'Azzo & Constantine H. Houpis**: *Linear Control Stystem Analysis and Design: Conventional and Modern, 4e.* McGraw-Hill, 1995, ISBN 0-07-016321-9
55. **Donald G. Schultz / James L. Melsa & Charles E. Rohrs**: *Linear Control Systems.* McGraw-Hill, 1993, ISBN 0-07-041525-0
56. **Jeffrey B. Burl**: *Linear Optimal Control: H2 and H-infinity Methods.* Addison-Wesley, 1999, ISBN 0-201-80868-4
57. **Chi-Tsong Chen**: *Linear System Theory and Design.* Oxford University Press, 1999, ISBN 0-19-511777-8
58. **Peter Dorato / Chaouki Abdallah & Vito Cerone**: *Linear-Quadratic Control: An Introduction* Prentice Hall, 1995, ISBN 0-02-329962-2
59. **Adolph H. Glattfelder & Walter Schaufelberger**: *Lineare Regelsysteme: Eine Einführung mit MATLAB* vdf Hochschulverlag AG, 1997, ISBN 3-7281-2275-0
60. **Helmut Bode**: *MATLAB in der Regelungstechnik: Analyse linearer Systeme.* B.G. Teubner, 1998, ISBN 3-519-06252-6
61. **A.J. Chipperfield & P.J. Fleming**: *MATLAB Toolboxes and Applications for Control* Peter Peregrinus Ltd./IEE, 1993, ISBN 0-86341-290-4
62. **Duane C. Hanselman & Benjamin C. Kuo**: *MATLAB Tools for Control System Analysis and Design, 2e.* Prentice Hall, 1995, ISBN 0-13-202293-1 (PC), ISBN 0-13-202574-4 (Macintosh)
63. **Dobrivoje Popovic & Ljubo Vlacic**: *Mechatronics in Engineering Design and Product Development* Marcel Dekker, Inc., 1999, ISBN 0-8247-0226-3
64. **Charles Close & Dean K. Frederick**: *Modeling and Analysis of Dynamic Systems, 2e* John Wiley & Sons, Inc., 1993, ISBN 0-471-12517-2
65. **Lorenzo Sciavicco & Bruno Siciliano**: *Modeling and Control of Robot Manipulators.* McGraw-Hill, 1996, ISBN 0-07-057217-8
66. **John W. Woods & Kent L. Lawrence**: *Modeling and Simulation of Dynamic Systems* Prentice Hall, 1997, ISBN 0-13-337379-1
67. **William J. Palm III**: *Modeling, Analysis, and Control of Dynamic Systems, 2e.* John Wiley & Sons, Inc., 2000, ISBN 0-471-07370-9
68. **Katsuhiko Ogata**: *Modern Control Engineering, 3e* Prentice Hall, 1997, ISBN 0-13-227307-1
69. **Stanley M. Shinners**: *Modern Control System Theory and Design, 2e.* John Wiley & Sons, Inc., 1998, ISBN 0-471-24906-8
70. **Robert H. Bishop**: *Modern Control Systems Analysis & Design Using MATLAB & SIMULINK* Addison-Wesley, 1997, ISBN 0-201-49846-4
71. **Z. Gajic & M. Lelic**: *Modern Control Systems Engineering.* Prentice Hall, 1996, ISBN 0-13-134116-2
72. **Richard C. Dorf & Robert H. Bishop**: *Modern Control Systems, 8e.* Addison-Wesley, 1998, ISBN 0-201-30864-9
73. **Sigurd Skogestad & Ian Postlethwaite**: *Multivariable Feedback Control* John Wiley & Sons, Inc., 1996, ISBN 0-471-94277-4
74. **Marco Tibaldi**: *Note Introduttive a MATLAB e Control System Toolbox.* Progetto Leonardo, 1993
75. **W. Fred Ramirez**: *Process Control and Identification* Academic Press, 1994, ISBN 0-12-577240-8
76. **Thomas E. Marlin**: *Process Control: Designing Process and Control Systems for Dynamic Performance* McGraw-Hill, 1995, ISBN 0-07-040491-7
77. **B. Wayne Bequette**: *Process Dynamics: Modeling, Analysis, and Simulation.* Prentice Hall, 1998, ISBN 0-13-206889-3
78. **Paolo Bolzern**: *Programmi MATLAB per Exercitazioni di Elementi di Automatica.* Masson, 1994, ISBN 88-214-0678-4
79. **Oded Yaniv**: *Quantitative Feedback Design of Linear and Nonlinear Control Systems* Kluwer Academic Publishers, 1999, ISBN 0-7923-8529-2
80. **Jan Lunze**: *Regelungstechnik 2: MehrgrõBensysteme Digitale Regelung.* Springer-Verlag, 1997, ISBN 3-540-61898-8
81. **Petros A. Ioannou & Jing Sun**: *Robust Adaptive Control.* Prentice Hall, 1996, ISBN 0-13-439100-4
82. **Rick Lind & Marty Brenner**: *Robust Aeroservoelastic Stability Analysis.* Springer-Verlag, 1999, ISBN 1-85233-096-1
83. **S.P. Bhattacharyya / H. Chapellat & L.H. Keel**: *Robust Control: The Parametric Approach* Prentice Hall, 1995, ISBN 0-13-781576-X
84. **M.J. Grimble**: *Robust Industrial Control* Prentice Hall, 1994, ISBN 0-13-655283-8
85. **B. S. Bennett**: *Simulation Fundamentals* Prentice Hall, 1995, ISBN 0-13-813262-3
86. **C. Bonivento / C. Melchiorri & R. Zanasi**: *Sistemi di Controllo Digitale.* Progetto Leonardo, 1996, ISBN 88-85040-96-9
87. **Katsuhiko Ogata**: *Solving Control Engineering Problems with MATLAB.* Prentice Hall, 1994, ISBN 0-13-045907-0
88. **Eronini Umez-Eronini**: *System Dynamics and Control.* Brooks/Cole Publishing Company, 1999, ISBN 0-534-94451-5

89. **Katsuhiko Ogata**: *System Dynamics, 3e*
Prentice Hall, 1998, ISBN 0-13-675745-6
90. **Ernest O. Doebelin**: *System Dynamics: Modeling, Analysis, Simulation, Design.* Marcel Dekker, Inc., 1998, ISBN 0-8247-0126-7
91. **El-Kébir Boukas**: *Systčmes Asservis*. Editions De L'Ecole Polytechnique De Montreal, 1995, ISBN 2-553-00430-3
92. **Naim A. Kheir**: *Systems Modeling and Computer Simulation, 2e.* Marcel Dekker, Inc., 1996, ISBN 0-8247-9421-4
93. **Paul Zarchan**: *Tactical and Strategic Missile Guidance, 3e.* AIAA, 1997, ISBN 1-56347-254-6
94. **Lou Shuntian & Yu Wei**: *The Analysis and Design of Systems Using MATLAB: Control System*
Xidian University Press, 1998, ISBN 7-5606-0657-1
95. **Ken Dutton / Steve Thompson & Bill Barraclough**: *The Art of Control Engineering.* Addison-Wesley, 1997, ISBN 0-201-17545-2
96. **Naomi Ehrich Leonard & William S. Levine**: *Using MATLAB to Analyze and Design Control Systems, 2e*
Addison-Wesley, 1995, ISBN 0-8053-2193-4
97. **Alberto Cavallo / Roberto Setola & Francesco Vasca**: *Using MATLAB, Simulink and Control System Toolbox: A Practical Approach.* Prentice Hall, 1996, ISBN 0-13-261058-2

ECONOMICS/FINANCE

98. **Neil A. Chriss: Black-Scholes and Beyond**: *Options Pricing Models.* Irwin/McGraw Hill, 1997, ISBN 0-7863-1025-1

ELECTRONICS

99. **Teresa Serrano-Gotarredona / Bernabé Linares-Barranco & Andreas G. Andreou**: *Adaptive Resonance Theory Microchips: Circuit Design Techniques*
Kluwer Academic Publishers, 1998, ISBN 0-7923-8231-5
100. **Kendall Su**: *Analog Filters*
Chapman Hall, 1996, ISBN 0-412-63840-1
101. **Chee-Mun Ong**: *Dynamic Simulation of Electric Machinery: Using MATLAB/Simulink.* Prentice Hall, 1998, ISBN 0-13-723785-5
102. **James W. Nilsson & Susan A. Riedel**: *Electric Circuits, 5e.* Addison-Wesley, 1996, ISBN 0-201-55707-X
103. **Ion Boldea & S. A. Nasar**: *Electric Drives.*
CRC Press, Inc., 1999, ISBN 0-8493-2521-8
104. **Stephen J. Chapman**: *Electric Machinery Fundamentals, 3e.* WCB/McGraw-Hill, 1999, ISBN 0-07-011950-3
105. **Karl E. Lonngren**: *Electromagnetics with MATLAB.* International Science Publishing, 1997, ISBN 1 898326 517
106. **John O. Attia**: *Electronics and Circuit Analysis using MATLAB.* CRC Press, Inc., 1999, ISBN 0-8493-1176-4
107. **Richard C. Dorf & James A. Svoboda**: *Introduction to Electric Circuits, 4e.* John Wiley & Sons, Inc., 1998, ISBN 0-471-19246-5
108. **Raymond DeCarlo & Pen-Min Lin**: *Linear Circuit Analysis: Time Domain, Phasor, and Laplace Transform Approaches.* Prentice Hall, 1995, ISBN 0-13-473869-1
109. **James G. Gottling**: *Matrix Analysis of Circuits Using MATLAB.* Prentice Hall, 1995, ISBN 0-13-127044-3
110. **William D. Stanley**: *Network Analysis With Applications, 3e.* Prentice Hall, 2000, ISBN 0-13-010589-9
111. **William F. Egan**: *Phase-Lock Basics.* John Wiley & Sons, Inc., 1998, ISBN 0-471-24261-6
112. **Jean-Pierre Colinge & Fernand Van de Wiele**: *Physique des Dispositifs Semi-conducteurs*
De Boeck Universite, 1996, ISBN 2-8041-2107-0
113. **Hadi Saadat**: *Power System Analysis*
WCB/McGraw-Hill, 1999, ISBN 0-07-561634-3
114. **Robert F. Pierret**: *Semiconductor Device Fundamentals.* Addison-Wesley, 1996, ISBN 0-201-54393-1
115. **Robert R. Boyd**: *Tolerance Analysis of Electronic Circuits Using MATLAB.* CRC Press, Inc., 1999, ISBN 0-8493-2276-6
116. **William D. Stanley**: *Transform Circuit Analysis for Engineering and Technology, 4e.* Prentice Hall, 2000, ISBN 0-13-020035-2

GENERAL

117. **Eva Pärt-Enander & Anders Sjöberg**: *Anvandarhandledning for MATLAB 5*
Department of Scientific Computing, 1998, ISBN 91-506-1292-1
118. **M. Mokhtari & M. Marie**: *Applications de MATLAB 5 et Simulink 2.* Springer-Verlag, 1998, ISBN 2-287-59651-8
119. **Robert J. Schilling & Sandra L. Harris**: *Applied Numerical Methods for Engineers: Using MATLAB and C.* Brooks/Cole Publishing Company, 2000, ISBN 0-534-37014-4
120. **M. Mokhtari & A. Mesbah**: *Apprendre et Maitriser MATLAB.* Springer-Verlag, 1997, ISBN 3-540-62773-1
121. **Andrew Knight**: *Basics of MATLAB and Beyond*
CRC Press, Inc., 2000, ISBN 0-8493-2039-9
122. **Joe King**: *Engineer's Toolkit: MATLAB 5.0 for Engineers.* Addison-Wesley, 1998, ISBN 0-201-35094-7
123. **Delores M. Etter**: *Engineering Problem Solving with MATLAB, 2e.* Prentice Hall, 1997, ISBN 0-13-397688-2
124. **Brian D. Hahn**: *Essential MATLAB for Scientists and Engineers.* Arnold, 1997, ISBN 0-470-25013-5,,ISBN 0-340-69144-1 (outside North America)
125. **Rudra Pratap**: *Getting Started with MATLAB 5: A Quick Introduction for Scientists and Engineers*
Oxford University Press, 1999, ISBN 0-19-512947-4
126. **Patrick Marchand**: *Graphics and GUIs with MATLAB, 2e.* CRC Press, Inc., 1999, ISBN 0-8493-9001-X
127. **Mark Austin & David Chancogne**: *Introduction to Engineering Programming: in C, MATLAB, and Java*
John Wiley & Sons, Inc., 1999, ISBN 0-471-00116-3
128. **William J. Palm III**: *Introduction to MATLAB for Engineers.* McGraw-Hill, 1998, ISBN 0-07-047328-5
129. **Delores M. Etter**: *Introduction to MATLAB for Engineers and Scientists.* Prentice Hall, 1996, ISBN 0-13-519703-1
130. **Finn Haugen**: *Laer MATLAB: pa 5 timer!*
Haugen, 1996, ISBN 82-91748-00-4
131. **Finn Haugen**: *Laer Simulink: pa 3 timer!*
Haugen, 1996, ISBN 82-91748-01-2
132. **Finn Haugen**: *Learn MATLAB 5 in 6 hours!*
Tech Teach, 1997, ISBN 82-91748-02-0
133. **Finn Haugen**: *Learn Simulink 2 in 3 hours!*
Tech Teach, 1997, ISBN 82-91748-03-9
134. **Duane C. Hanselman & Bruce Littlefield**: *Mastering MATLAB 5: A Comprehensive Tutorial and Reference*
Prentice Hall, 1998, ISBN 0-13-858366-8
135. **James B. Dabney & Thomas L. Harman**: *Mastering SIMULINK 2.* Prentice Hall, 1998, ISBN 0-13-243767-8
136. **Adrian Biran & Moshe M.G. Breiner**: *MATLAB 5 for Engineers.* Addison-Wesley, 1999, ISBN 0-201-36043-8
137. **Adrian Biran & Moshe M.G. Breiner**: *MATLAB 5 für Ingenieure.* Addison-Wesley, 1999, ISBN 3-8273-1416-X
138. **Bogumila Mrozek & Zbigniew Mrozek**: *MATLAB 5.x, SIMULINK 2.x: poradnik uzytkownika*
Wydawnictwo PLJ, 1998, ISBN 83-7101-376-0
139. **Tsutomu Oguni**: *MATLAB And Its Usage: Modern Applied* Mathematics and Computer Graphics, ISBN 4-7819-0763-6
140. **William J. Palm III**: *MATLAB for Engineering Applications*
WCB/McGraw-Hill, 1999, ISBN 0-07-047330-7

141. **Tsutomu Oguni**: *MATLAB Graphical Library*
Asakura Shoten, 1997, ISBN 4-254-11073-1
142. **Ibrahim Yüksel**: *MATLAB ile Mühendislik Sistemlerinin Analizi ve Cözümü.* Uludag University, 1996, ISBN 975-564-049-5
143. **Kermit Sigmon**: *MATLAB Primer, 5e*
CRC Press, Inc., 1998, ISBN 0-8493-1305-8
144. **Lou Shuntian & Yu Wei**: *MATLAB Programming Language*
Xidian University Press, 1997, ISBN 7-5606-0537-0
145. **Josef Hoffmann**: *MATLAB und SIMULINK: Beispielorientierte Einfuhrung in die Simulation dynamischer Systeme*
Addison-Wesley, 1997, ISBN 3-8273-1077-6
146. **Bogumila Mrozek & Zbigniew Mrozek**: *MATLAB, Uniwersalne srodowisko do obliczen naukowo-technicznych, 3e*
Wydawnictwo PLJ, 1996, ISBN ISBN 83-7101-325-6
147. **Shoichiro Nakamura**: *Numerical Analysis and Graphic Visualization with MATLAB* Prentice Hall, 1996, ISBN 0-13-051518-3
148. **J. Brzózka & Lech Dorobczynski**: *Programowanie w MATLAB.* MIKOM, 1998, ISBN 83-7158-120-3
149. **Walter Gander & Jiri Hrebicek**: *Solving Problems in Scientific Computing Using Maple and MATLAB, 3e*
Springer-Verlag, 1997, ISBN 3-540-61793-0
150. **Darren Redfern & Colin Campbell**: *The MATLAB 5 Handbook.* Springer-Verlag, 1998, ISBN 0-387-94200-9
151. **Eva Pärt-Enander & Anders Sjöberg**: *The MATLAB 5 Handbook.* Addison-Wesley, 1999, ISBN 0-201-39845-1
152. ** The MathWorks, Inc.: *The Student Edition of MATLAB 5 User's Guide.* Prentice Hall, 1997,ISBN 0-13-272550-9
153. ** The MathWorks, Inc.: *The Student Edition of SIMULINK 2 User's Guide.* Prentice Hall, 1998, ISBN 0-13-659699-1
154. **Zhi-Yong Zhang / Rui-Zhen Liu & Zu-Ying Yang**: *Undertanding and Mastering MATLAB.*
BUAA Press, 1997,ISBN 7-81012-702-0

MATHEMATICS

155. **W. F. Carroll**: *A Primer for Finite Elements in Elastic Structures.* John Wiley & Sons, Inc., 1999, ISBN 0-471-28345-2
156. **Nicholas Higham**: *Accuracy and Stability of Numerical Algorithms.* SIAM, 1996, ISBN 0-89871-355-2
157. **Thomas L. Harman / James B. Dabney & Norman John Richert**: *Advanced Engineering Mathematics Using MATLAB*
PWS Publishing Company, 1997, ISBN 0-534-94350-0
158. **Reza Malek-Madani**: *Advanced Engineering Mathematics With Mathematica and MATLAB: Volume I and II.* Addison-Wesley, 1998, ISBN 0-201-59881-7(Vol I), ISBN 0-201-32549-7(Vol II)
159. **Silvano Matarasso & Tommaso Ruggeri**: *Analisi Matematica 2: Calcolo Differenziale.* Progetto Leonardo, 1995
160. **Valeriano Comincioli**: *Analisi Numerica Metodi Modelli Applicazioni.* McGraw-Hill, 1990, ISBN 88-386-0646-3
161. **Ben Noble & James W. Daniel**: *Applied Linear Algebra, 3e*
Prentice Hall, 1988, ISBN 0-13-041260-0
162. **Curtis F. Gerald & Patrick O. Wheatley**: *Applied Numerical Analysis, 6e.* Addison-Wesley, 1999, ISBN 0-201-87072-X
163. **James W. Demmel**: *Applied Numerical Linear Algebra.*
SIAM, 1997, ISBN 0-89871-389-7
164. **Steven J. Leon / Eugene Herman & Richard Faulkenberry**: *ATLAST Computer Exercises for Linear Algebra.* Prentice Hall, 1996, ISBN 0-13-270273-8
165. **Marco Frontini & Aldo Tagliani**: *Calcolo Numerico-Esercizi*
Clup CittíStudi, 1992,ISBN 88-251-7056-4
166. **Ignacio Martin Llorente & Victor M. P García**: *Cálculo numérico para computación en Ciencia e Ingenieria: Desarrollo práctico con MATLAB.* Editorial Sintesis, 1998, ISBN 84-7738-586-6
167. **Gerald L. Bradley & Karl J. Smith**: *Calculus*
Prentice Hall, 1995,ISBN 0-13-178617-2
168. **C. Henry Edwards & David E. Penney**: *Calculus with Analytic Geometry, 4e.* Prentice Hall, 1994, ISBN 0-13-463993-6
169. **Glen D. Anderson / Mavina K. Vamanamurthy & Matti K. Vuorinen**: *Conformal Invariants, Inequalities, and Quasiconformal Maps.* John Wiley & Sons, Inc., 1997, ISBN 0-471-59486-5
170. **C. Henry Edwards & David E. Penney**: *Differential Equations and Boundary Value Problems: Computing and Modeling*
Prentice Hall, 1996, ISBN 0-13-382094-7
171. **Kevin R. Coombes / Brian R. Hunt / Ronald L. Lipsman / John E. Osborn & Garrett J. Stuck**: *Differential Equations with MATLAB.* John Wiley & Sons, Inc., 2000, ISBN 0-471-32227-X
172. **C. Henry Edwards & David E. Penney**: *Differential Equations: Computing and Modeling.* Prentice Hall, 1996, ISBN 0-13-382102-1
173. **Paul Bugl**: *Differential Equations: Matrices and Models*
Prentice Hall, 1995, ISBN 0-02-316540-5
174. **Paul Davis**: *Differential Equations: Modeling with MATLAB*
Prentice Hall, 1999, ISBN 0-13-736539-X
175. **Arthur E. Bryson, Jr.**: *Dynamic Optimization*
Addison-Wesley, 1999, ISBN 0-201-59790-X
176. **Richard O. Hill, Jr.**: *Elementary Linear Algebra with Applications, 3e.* Saunders College Publishing, 1996, ISBN 0-03-010347-9
177. **Roland Larson & Bruce Edwards**: *Elementary Linear Algebra, 3e.* DC Heath, 1996, ISBN 0-669-39641-9
178. **Stanley I. Grossman**: *Elementary Linear Algebra, 5e*
Saunders College Publishing, 1994, ISBN 0-03-097354-6
179. **David R. Hill**: *Experiments in Computational Matrix Algebra*
McGraw-Hill, 1988, ISBN 0-394-35678-0
180. **Thomas F. Coleman & Charles F. Van Loan**: *Handbook for Matrix Computations.* SIAM, 1988, ISBN 0-89871-227-0
181. **Hans Benker**: *Ingenieurmathematik mit Computeralgebra-Systemen.* Vieweg, 1998, ISBN 3-528-05673-8
182. **Yurii Nesterov & Arkadi Nemirovskii**: *Interior-Point Polynomial Algorithms in Convex Programming*
SIAM, 1994, ISBN 0-89871-319-6
183. **Gilbert Strang**: *Introduction to Linear Algebra, 2e*
Wellesley-Cambridge Press, ISBN 0-9614088-5-5
184. **Lee W. Johnson / R. Dean Riess & Jimmy T. Arnold**: *Introduction to Linear Algebra, 4e*
Addison-Wesley, 1998, ISBN 0-201-82416-7
185. **Jeffery M. Cooper**: *Introduction to Partial Differential Equations with MATLAB.* Birkhauser, 1998, ISBN 0-8176-3967-5
186. **Charles F. Van Loan**: *Introduction to Scientific Computing, 2e: A Matrix-Vector Approach Using MATLAB*
Prentice Hall, 2000, ISBN 0-13-949157-0
187. **Bernard Kolman**: *Introductory Linear Algebra with Applications, 6e.* Prentice Hall, 1997, ISBN 0-13-266313-9
188. **C.T. Kelley**: *Iterative Methods for Linear and Nonlinear Equations.* SIAM, 1995, ISBN 0-89871-352-8
189. **C.T. Kelley**: *Iterative Methods for Optimization*
SIAM, 1999, ISBN 0-89871-433-8
190. **Francoise Chaitin-Chatelin & Valerie Fraysse**: *Lectures on Finite Precision Computations.* SIAM, 1996, ISBN 0-89871-358-7
191. **Terry Lawson**: *Linear Algebra*
John Wiley & Sons, Inc., 1996, ISBN 0-471-30897-8
192. **Martin Golubitsky & Michael Dellnitz**: *Linear Algebra and Differential Equations Using MATLAB*
Brooks/Cole Publishing Company, 1999, ISBN 0-534-35425-4
193. **David C. Lay**: *Linear Algebra and Its Applications, 2e*
Addison-Wesley, 1997, ISBN 0-201-82478-7
194. **David R. Hill & David E. Zitarelli**: *Linear Algebra LABS with MATLAB, 2e.* Prentice Hall, 1996, ISBN 0-13-505439-7

195. **George Nakos & David Joyner**: *Linear Algebra with Applications* Brooks/Cole Publishing Company, 1998, ISBN 0-534-95526-6
196. **Gareth Williams**: *Linear Algebra with Applications, 3e* Wm. C. Brown Publishers, 1996, ISBN 0-697-26849-7
197. **Steven J. Leon**: *Linear Algebra with Applications, 5e* Prentice Hall, 1998, ISBN 0-13-849308-1
198. **John B. Fraleigh & Raymond A. Beauregard**: *Linear Algebra, 3e.* Addison-Wesley, 1995, ISBN 0-201-52675-1
199. **Gilbert Strang & Kai Borre**: *Linear Algebra, Geodesy, and GPS.* Wellesley-Cambridge Press, 1997, ISBN 0-9614088-6-3
200. **Tsutomu Oguni & Jack Dongarra**: *Linear Computation Software with MATLAB.* Maruzen Corporation, 1998, ISBN 4-621-04442-7
201. **Ke Chen / Alan Irving & Peter Giblin**: *Mathematical Explorations with MATLAB* Cambridge University Press, 1999, ISBN 0-521-63920-4
202. **Franz Bachmann / Hans Rudolf Schärer & Louis-Sepp Willimann**: *Mathematik mit MATLAB: Aufgaben und Losungen* vdf Hochschulverlag AG, 1996, ISBN 3-7281-2308-0
203. **Franca Caliň & Marco Frontini**: *MATLAB Esercitazioni di Calcolo Numerico Assistite da Calcolatore* Clup CittíStudi, 1991, ISBN 88-251-0002-7
204. **Karen Donnelly**: *MATLAB Manual: Computer Laboratory Exercises.* Saunders College Publishing, 1995, ISBN 0-03-094896-7
205. **Marvin Marcus**: *Matrices and MATLAB: A Tutorial* Prentice Hall, 1993, ISBN 0-13-562901-2
206. **Gene H. Golub & Charles F. Van Loan**: *Matrix Computations, 2e.* Johns Hopkins University Press, 1989, ISBN 0-8018-3739-1
207. **Jack L. Goldberg**: *Matrix Theory with Applications* McGraw-Hill, 1991, ISBN 0-07-557200-1
208. **David Kincaid & Ward Cheney**: *Numerical Analysis, 2e* Brooks/Cole Publishing Company, 1996, ISBN 0-534-33892-5
209. **Lloyd N. Trefethen & David Bau, III**: *Numerical Linear Algebra.* SIAM, 1998, ISBN 0-89871-361-7
210. **Biswa Nath Datta**: *Numerical Linear Algebra and Applications* Brooks/Cole Publishing Company, 1995, ISBN 0-534-17466-3
211. **Steven C. Chapra & Raymond P. Canale**: *Numerical Methods for Engineers: With Programming and Software Applications, 3e.* WCB/McGraw-Hill, 1998, ISBN 0-07-010938-9
212. **George Lindfield & John Penny**: *Numerical Methods Using MATLAB, 2e.* Prentice Hall, 2000, ISBN 0-13-012641-1
213. **John H. Mathews & Kurtis D. Fink**: *Numerical Methods Using MATLAB, 3e.* Prentice Hall, 1999, ISBN 0-13-270042-5
214. **G. J. Borse**: *Numerical Methods With MATLAB: A Resource for Scientists and Engineers* PWS Publishing Company, 1997, ISBN 0-534-93822-1
215. **John C. Polking & David Arnold**: *Ordinary Differential Equations Using MATLAB, 2e* Prentice Hall, 1999, ISBN 0-13-011381-6
216. **Gunnar Backstrom**: *Practical Mathematics Using MATLAB 5* Studentlitteratur, 1997, ISBN 91-44-00544-X
217. **Gunnar Backstrom**: *Praktisk matematik med MATLAB 5* Studentlitteratur, 1997, ISBN 91-44-00224-6
218. **Michael T. Heath**: *Scientific Computing: An Introductory Survey.* McGraw-Hill, 1997, ISBN 0-07-027684-6
219. **Richard Barrett / Michael Berry / Tony F. Chan / James W. Demmel / June Donato / Jack Dongarra / Victor Eijkhout / Roldan Pozo / Charles Romine & Henk van der Vorst**: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, 1994, ISBN 0-89871-328-5
219. **Richard Barrett / Michael Berry / Tony F. Chan / James W. Demmel / June Donato / Jack Dongarra / Victor Eijkhout / Roldan Pozo / Charles Romine & Henk van der Vorst**: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Asakura Shoten, 1994, ISBN 4-254-11401-X
220. **Rick L. Smith**: *The MATLAB Project Book for Linear Algebra* Prentice Hall, 1997, ISBN 0-13-521337-1

MECHANICAL ENG

221. **Robert L. Clark / Willliam R. Saunders & Gary P. Gibbs**: *Adaptive Structures: Dynamics & Control* John Wiley & Sons, Inc., 1998, ISBN 0-471-12262-9
222. **Howard B. Wilson & Louis H. Turcotte**: *Advanced Mathematics and Mechanics Applications Using MATLAB, 2e* CRC Press, Inc., 1997, ISBN 0-8493-1686-3
223. **D.E. Newland**: *An Introduction to Random Vibrations, Spectral & Wavelet Analysis, 3e* Longman Scientific & Technical, 1994, ISBN 0-582-21584-6
224. **Louis H. Turcotte & Howard B. Wilson**: *Computer Applications in Mechanics of Materials Using MATLAB* Prentice Hall, 1998, ISBN 0-13-749060-7
225. **Daniel J. Inman**: *Engineering Vibration* Prentice Hall, 1994, ISBN 0-13-951773-1
226. **James O. Wilkes**: *Fluid Mechanics for Chemical Engineers* Prentice Hall, 1999, ISBN 0-13-739897-2
227. **James A. Liggett & David A. Caughey**: *Fluid Mechanics: An Interactive Text* American Society of Civil Engineers, 1998, ISBN 0-7844-0310-4
228. **Kenneth J. Waldron & Gary L. Kinzel**: *Kinematics, Dynamics, and Design of Machinery* John Wiley & Sons, Inc., 1999, ISBN 0-471-58399-5
229. **Douglas W. Hull: Mastering Mechanics 1**: *Using MATLAB 5* Prentice Hall, 1999,ISBN 0-13-864034-3
230. **Haym Benaroya**: *Mechanical Vibration: Analysis, Uncertainties, and Control* Prentice Hall, 1998, ISBN 0-13-948373-X
231. **M.F. Golnaraghi / D. Boulahbal & R.L. Leask**: *Solving Solid Mechanics Problems with MATLAB 5* Prentice Hall, 1999, ISBN 0-13-021537-6
232. **Roy Levy**: *Structural Engineering of Microwave Antennas* IEEE Press, 1996, ISBN 0-7803-1020-9
233. **Young Kwon & Hyochoong Bang**: *The Finite Element Method Using MATLAB* CRC Press, Inc., 1997, ISBN 0-8493-9653-0
234. **William T. Thomson & Marie Dillon Dahleh**: *Theory of Vibration with Applications, 5e* Prentice Hall, 1998, ISBN 0-13-651068-X

NATURAL SCIENCES

235. **Richard A. Reyment & K.G. Joreskog**: *Applied Factor Analysis in the Natural Sciences* Cambridge University Press, 1993, ISBN 0-521-41242-0
236. **Leslie F. Marcus / Elisa Bellos & Antonio Garcia-Valdecasas**: *Contributions to Morphometrics* Museo Nacional de Ciencias Naturales, 1993, ISBN 84-00-07353-3
237. **Gerard V. Middleton**: *Data Analysis in the Earth Sciences Using MATLAB.* Prentice Hall, 2000, ISBN 0-13-393505-1
238. **George M. Hornberger / Jeffrey P. Raffensperger & Keith N. Eshleman**: *Elements of Physical Hydrology* Johns Hopkins University Press, 1998, ISBN 0-8018-5857-7
239. **Jonathan Roughgarden**: *Primer of Ecological Theory* Prentice Hall, 1998, ISBN 0-13-442062-4

NEURAL/FUZZY

240. **Leften H. Tsoukalas & Robert E. Uhrig**: *Fuzzy and Neural Approaches in Engineering* John Wiley & Sons, Inc., 1997, ISBN 0-471-16003-2
241. **Kevin M. Passino & Stephen Yurkovich**: *Fuzzy Control Addison-Wesley, 1998, ISBN 0-201-18074-X*

242. **John Yen & Reza Langari**: *Fuzzy Logic: Inteligence, Control, and Information.* Prentice Hall, , ISBN 0-13-525817-0
243. **Robert Babuska**: *Fuzzy Modeling for Control*
Kluwer Academic Publishers, 1998, ISBN 0-7923-8154-8
244. **Herve Abdi**: *Les Réseaux de Neurones*
Presses Universitaires de Grenoble, 1994, ISBN 2-7061-0554-2
245. **Chin-Teng Lin & C.S. George Lee**: *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*
Prentice Hall, 1996, ISBN 0-13-235169-2
246. **Martin T. Hagan / Howard B. Demuth & Mark Beale**: *Neural Network Design*
PWS Publishing Company, 1996, ISBN 0-534-94332-2
247. **N. K. Bose & P. Liang**: *Neural Network Fundamentals with Graphs, Algorithms, and Applications*
McGraw-Hill, 1996, ISBN 0-07-006618-3
248. **Simon Haykin**: *Neural Networks: A Comprehensive Foundation, 2e.* Prentice Hall, 1999, ISBN 0-13-273350-1
249. **Jyh-Shing Roger Jang / Chuen-Tsai Sun & Eiji Mizutani**: *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*
Prentice Hall, 1997, ISBN 0-13-261066-3
250. **Lou Shuntian**: *The Analysis and Design of Systems Using MATLAB: Neural Network.*
Xidian University Press, 1998, ISBN 7-5606-0646-6

PHYSICS

251. **Alejandro L. Garcia & Cecile Penland**: *MATLAB Projects for Scientists and Engineers*
Prentice Hall, 1996, ISBN 0-13-460171-8
252. **Alejandro L. Garcia**: *Numerical Methods for Physics, 2e*
Prentice Hall, 2000, ISBN 0-13-906744-2

SIGNAL PROCESSING

253. **Boaz Porat**: *A Course in Digital Signal Processing*
John Wiley & Sons, Inc., 1997, ISBN 0-471-14961-6
254. **Henrik V. Sorensen & Jianping Chen**: *A Digital Signal Processing Laboratory Using the TMS320C30*
Prentice Hall, 1997, ISBN 0-13-741828-0
255. **Simon Haykin**: *Adaptive Filter Theory, 3e*
Prentice Hall, 1996, ISBN 0-13-322760-X
256. **B. Farhang-Boroujeny**: *Adaptive Filters: Theory and Applications.* John Wiley & Sons, Inc., 1998, ISBN 0-471-98337-3
257. **Ashok Ambardar**: *Analog and Digital Signal Processing*
PWS Publishing Company, 1995, ISBN 0-534-94086-2
258. **Mircea Grigoriu**: *Applied Non-Gaussian Processes*
Prentice Hall, 1995, ISBN 0-13-367095-3
259. **Charles F. Van Loan**: *Computational Frameworks for the Fast Fourier Transform.* SIAM, 1992, ISBN 0-89871-285-8
260. **Anthony Teolis**: *Computational Signal Processing with Wavelets.* Birkhauser, 1998, ISBN 0-8176-3909-8
261. **John R. Buck / Michael M. Daniel & Andrew C. Singer**: *Computer Explorations in Signals and Systems Using MATLAB*
Prentice Hall, 1997, ISBN 0-13-732868-0
262. **James H. McClellan / C. Sidney Burrus / Alan V. Oppenheim / Thomas W. Parks / Ronald W. Schafer & Hans W. Schuessler**: *Computer-Based Exercises for Signal Processing Using MATLAB 5.* Prentice Hall, 1998, ISBN 0-13-789009-5
263. **Leland B. Jackson**: *Digital Filters and Signal Processing, 3e - with MATLAB Excercises*
Kluwer Academic Publishers, 1996, ISBN 0-7923-9559-X
264. **Sanjit K. Mitra**: *Digital Signal Processing Laboratory Using MATLAB.* WCB/McGraw-Hill, 1999, ISBN 0-07-232246-2
265. **Vinay K. Ingle & John G. Proakis**: *Digital Signal Processing Problems Using MATLAB v4*
PWS Publishing Company, 1997, ISBN 0-534-93805-1
266. **Sanjit K. Mitra**: *Digital Signal Processing: A Computer-Based Approach.* McGraw-Hill, 1998, ISBN 0-07-042953-7
267. **H. W. Schuessler**: *Digitale Signalverarbeitung I: Analyse diskreter Signale und Systeme, 4e*
Springer-Verlag, 1994, ISBN 3-540-57428-X
268. **K.D. Kammeyer & K. Kroschel**: *Digitale Signalverarbeitung: Filterung und Spektralanalyse mit MATLAB-Übungen*
B.G. Teubner, 1998, ISBN 3-519-36122-1
269. **Daniel Ch. von Grünigen**: *Digitale Signalverarbeitung: Grundlagen und Anwendungen Beispiele und Ubungen mit MATLAB.* AT Verlag, 1993, ISBN 3-905214-16-4
270. **Charles W. Therrien**: *Discrete Random Signals and Statistical Signal Processing.* Prentice Hall, 1992, ISBN 0-13-852112-3
271. **James H. McClellan / Ronald W. Schafer & Mark A. Yoder**: *DSP First: A Multimedia Approach*
Prentice Hall, 1998, ISBN 0-13-243171-8
272. **Mehrdad Soumekh**: *Fourier Array Imaging*
Prentice Hall, 1994, ISBN 0-13-063769-6
273. **Raymond G. Wilson**: *Fourier Series and Optical Transform Techniques in Contemporary Optics: An Introduction*
John Wiley & Sons, Inc., 1995, ISBN 0-471-30357-7
274. **Edward W. Kamen & Bonnie S. Heck**: *Fundamentals of Signals and Systems Using MATLAB*
Prentice Hall, 1997, ISBN 0-02-361942-2
275. **Jaideva C. Goswami & Andrew K. Chan**: *Fundamentals of Wavelets: Theory, Algorithms, and Applications*
John Wiley & Sons, Inc., 1999, ISBN 0-471-19748-3
276. **Chrysostomos L. Nikias & Athina P. Petropulu**: *Higher-Order Spectra Analysis.* Prentice Hall, 1993, ISBN 0-13-678210-8
277. **Carlos E. D'Attellis / Marta T. Anaya / Maria I. Cavallaro & Francisco F. Villaverde: Introducción a las Onditas**: *Una presentacion para Cursos de Grado de Ingenieria con MATLAB*
Neuva Librería, 1995, ISBN 950-9088-77-3
278. **Richard Shiavi**: *Introduction to Applied Statistical Signal Analysis, 2e.* Academic Press, 1999, ISBN 0-12-640010-5
279. **Robert Grover Brown & Patrick Y. C. Hwang**: *Introduction to Random Signals and Applied Kalman Filtering: with MATLAB Excercises and Solutions, 3e*
John Wiley & Sons, Inc., 1997, ISBN 0-471-12839-2
280. **Sophocles J. Orfanidis**: *Introduction to Signal Processing*
Prentice Hall, 1996, ISBN 0-13-209172-0
281. **Douglas K. Lindner**: *Introduction to Signals and Systems*
WCB/McGraw-Hill, 1999, ISBN 0-256-25259-9
282. **Petre Stoica & Randolph L. Moses**: *Introduction to Spectral Analysis.* Prentice Hall, 1997, ISBN 0-13-258419-0
283. **C. Sidney Burrus / Ramesh A. Gopinath & Haitao Guo**: *Introduction to Wavelets and Wavelet Transforms: A Primer*
Prentice Hall, 1998, ISBN 0-13-489600-9
284. **Virginia Stonick & Kevin Bradley**: *Labs for Signals and Systems Using MATLAB*
PWS Publishing Company, 1995, ISBN 0-534-93808-6
285. **Jerry M. Mendel**: *Lessons in Estimation Theory for Signal Processing, Communications, and Control*
Prentice Hall, 1995, ISBN 0-13-120981-7
286. **B.P. Lathi**: *Linear Systems and Signals*
Berkeley-Cambridge Press, 1992, ISBN 0-941413-34-9
287. **Ashok Ambardar & Craig Borghesani**: *Mastering DSP Concepts Using MATLAB.* Prentice Hall, 1998, ISBN 0-13-534976-1
288. **Todd K. Moon & Wynn C. Stirling**: *Mathematical Methods and Algorithms for Signal Processing*
Prentice Hall, 2000, ISBN 0-201-36186-8

289. **Fred J. Taylor**: *Principles of Signals and Systems*
McGraw-Hill, 1994, ISBN 0-07-911171-8
290. **Gordon E. Carlson**: *Signal and Linear System Analysis: with MATLAB, 2e.* John Wiley & Sons, Inc., 1998, ISBN 0-471-12465-6
291. **B.P. Lathi**: *Signal Processing & Linear Systems*
Berkeley-Cambridge Press, 1998, ISBN 0-941413-35-7
292. **Samuel D. Stearns & Ruth A. David**: *Signal Processing Algorithms in MATLAB.* Prentice Hall, 1996, ISBN 0-13-045154-1
293. **Bradley G. Boone**: *Signal Processing Using Optics: Fundamentals, Devices, Architectures, and Applications*
Oxford University Press, 1998, ISBN 0-19-508424-1
294. **James V. Candy**: *Signal Processing: The Model-Based Approach.* McGraw-Hill, 1986, ISBN 0-07-009725-9
295. **Rodger E. Ziemer & William H. Tranter**: *Signals & Systems: Continuous and Discrete, 4e*
Prentice Hall, 1998, ISBN 0-13-496456-X
296. **Simon Haykin & Barry Van Veen**: *Signals and Systems*
John Wiley & Sons, Inc., 1999, ISBN 0-471-13820-7
297. **Alan V. Oppenheim / Alan S. Willsky & S. Hamid Nawab**: *Signals and Systems, 2e* Prentice Hall, 1997, ISBN 0-13-814757-4
298. **Leland B. Jackson**: *Signals, Systems, and Transforms*
Addison-Wesley, 1991, ISBN 0-201-09589-0
299. **Charles L. Phillips & John M. Parr**: *Signals, Systems, and Transforms, 2e.* Prentice Hall, 1999, ISBN 0-13-095322-9
300. **Monson Hayes**: *Statistical Digital Signal Processing and Modeling.* John Wiley & Sons, Inc., 1996, ISBN 0471-59431-8
301. **J.F. Böhme**: *Stochastische Signale: mit Übungen und einem MATLAB-Praktikum.* B.G. Teubner, 1998, ISBN 3-519-16160-5
302. **Mehrdad Soumekh**: *Synthetic Aperture Radar Signal Processing: with MATLAB Algorithms*
John Wiley & Sons, Inc., 1999, ISBN 0-471-29706-2
303. **Philip Denbigh**: *System Analysis & Signal Processing: with emphasis on the use of MATLAB*
Addison-Wesley, 1998, ISBN 0-201-17860-5
304. **Chi-Tsong Chen**: *System and Signal Analysis, 2e*
Oxford University Press, 1994, ISBN 0-03-097709-6
305. **Lou Shuntian & Li Bohan**: *The Analysis and Design of Systems Using MATLAB: Signal Processing*
Xidian University Press, 1998, ISBN 7-5606-0634-2
306. **Gérard Blanchet & Maurice Charbit**: *Traitement numérique du signal: simulation sous MATLAB*
Editions HERMES, 1998, ISBN 2-88601-667-X
307. **William A. Sethares**: *Tuning, Timbre, Spectrum, and Scale*
Springer-Verlag, 1998, ISBN 3-540-76173-X
308. **Raghuveer M. Rao & Ajit S. Bopardikar**: *Wavelet Transforms: Introduction to Theory and Applications* Addison-Wesley, 1998, ISBN 0-201-63463-5
309. **Gilbert Strang & Truong Nguyen**: *Wavelets and Filter Banks*
Wellesley-Cambridge Press, 1996, ISBN 0-9614088-7-1

STATS & PROBABILITY

310. **Bernard Flury**: *A First Course in Multivariate Statistics*
Springer-Verlag, 1997, ISBN 0-387-98206-X
311. **Edward P.C. Kao**: *An Introduction to Stochastic Processes*
Duxbury Press, 1997, ISBN 0-534-25518-3
312. **Paul E. Pfeiffer**: *Basic Probability Topics Using MATLAB*
PWS Publishing Company, 1995, ISBN 0-534-94536-8
313. **Rodger E. Ziemer**: *Elements of Engineering Probability & Statistics.* Prentice Hall, 1997, ISBN 0-02-431620-2
314. **Steven M. Kay**: *Fundamentals of Statistical Signal Processing: Detection Theory Volume II*
Prentice Hall, 1998, ISBN 0-13-504135-X
315. **Jorge I. Aunon & V. Chandrasekar**: *Introduction to Probability and Random Processes*
McGraw-Hill, 1997, ISBN 0-07-001563-5
316. **Valeriano Comincioli**: *Metodi Numerici e Statistici per le Scienze Applicate*
C.E.A. Casa Editrice Ambrosiana, 1992, ISBN 88-408-0757-8
317. **V.G. Kulkarni**: *Modeling, Analysis, Design, and Control of Stochastic Systems.* Springer-Verlag, 1999, ISBN 0-387-98725-8
318. **Valen E. Johnson & James H. Albert**: *Ordinal Data Modeling*
Springer-Verlag, 1999, ISBN 0-387-98718-5
319. **George R. Cooper & Clare D. McGillem**: *Probabilistic Methods of Signal and System Analysis, 3e*
Oxford University Press, 1999, ISBN 0-19-512354-9
320. **Yannis Viniotis**: *Probability and Random Processes for Electrical Engineers.* WCB/McGraw-Hill, 1998, ISBN 0-07-067491-4
321. **Donald G. Childers**: *Probability and Random Processes: Using MATLAB with Applications to Continuous and Discrete Time Systems.* Irwin/McGraw Hill, 1997, ISBN 0-256-13361-1
322. **Roy D. Yates & David J. Goodman**: *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers* John Wiley & Sons, Inc., 1999, ISBN 0-471-17837-3
323. **Henry Stark & John W. Woods**: *Probability, Random Processes, and Estimation Theory for Engineers, 2e*
Prentice Hall, 1994, ISBN 0-13-728791-7
324. **David Y. Hsu**: *Spatial Error Analysis: A Unified Application-Oriented Treatment.* IEEE Press, 1999, ISBN 0-7803-3453-1
325. **Robin T. Clarke**: *Statistical Modelling in Hydrology*
John Wiley & Sons, Inc., 1994, ISBN 0-471-95016-5

SYSTEM ID

326. **J. Schoukens & R. Pintelon**: *Identification of Linear Systems: A Practical Guideline to Accurate Modeling*
Pergamon Press, 1991, ISBN 0-08040734-X
327. **Lennart Ljung & Torkel Glad**: *Modeling of Dynamic Systems*
Prentice Hall, 1994, ISBN 0-13-597097-0
328. **Peter van Overschee & Bart De Moor**: *Subspace Identification for Linear Systems: Theory, Implementation Applications*
Kluwer Academic Publishers, 1996, ISBN 0-7923-9717-7
329. **Shuichi Adachi**: *System Identification for Control Using MATLAB.* Tokyo Denki Publishing, 1997, ISBN 4-254-11073-1
330. **Lennart Ljung**: *System Identification: Theory for the User, 2e*
Prentice Hall, 1999, ISBN 0-13-656695-2

Rejstřík

Seznam vyobrazení