Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta – Katedra fyziky

Graphical user interface pro MATLAB v7.1.

Bakalářská práce

Vedoucí práce: RNDr. Petr Bartoš, Ph.D.

Autor: František Brabec

Anotace

Hlavní cíl mé práce je zaměřen na vytvoření textu, který má posloužit jako určitý nástroj pro tvorbu a vývoj grafických rozhraní v programovacím jazyku MATLAB. Obsah této práce se zaměřuje na základní seznámení s programovacím balíkem MATLAB, jeho historií, tvorbou m-souborů a popis práce v GUI prostředí. Velký důraz je kladen hlavně na správný návrh GUI aplikace, na její funkčnost a výpočetní část tak, aby byla pro uživatele zajištěna maximální efektivita. Skloubením všech těchto částí v GUI prostředí by měla vzniknout aplikace s velkým potenciálem.

Abstract

The main aim of this bachelor thesis is focused on the creation of the text, which serves as a tool for creation and development of a graphic interface in the programming language MATLAB. The content of this thesis is focused on the basic introduction to the MATLAB programming package, its history, creation of m-files and description of the work during the creation of the Graphical User Interface. The main accent is put firstly on the correct proposal of the GUI application, on its utility and on its computational part in order to ensure the maximal effectiveness for users. By the combination of all these parts in GUI setting the application with a huge potential can be created.

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím literatury uvedené v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Č. Budějovicích 5. 5. 2008

Podpis studenta

Touto formou děkuji svému konzultantovi p. RNDr. Petru Bartošovi, Ph.D., za cenné rady a připomínky při zpracování mé práce…

Obsah

Úvod

V této práci se zaměřuji na počítačový program MATLAB a dále na možnost využití grafického uživatelského rozhraní (GUI) při tvorbě uživatelských aplikací. Při řešení či simulaci fyzikálních jevů můžeme použít celou řadu nadstaveb MATLABu, neboli tzv. toolboxů. Tyto nadstavby nám pomohou ušetřit mnoho času a celý výsledek zvládneme v relativně krátké době. Využití MATLABu a jeho nadstaveb ve vědecko-technickém odvětví je skutečně obrovské. Tento program v sobě skrývá neuvěřitelný potenciál. Tento fakt mě velice oslovil, a proto jsem si vybral bakalářskou práci na toto téma.

V první části mé práce se věnuji především historii MATLABu od jeho vzniku až po současnost. Dále popisuji pracovní prostředí, kde jsem kladl velký důraz na využití MATLABu ve vědecko-technické sféře díky rozšiřujícím balíčkům, tzv. toolboxům.

Další kapitola je věnována způsobům tvorby m-souborů. Ty obsahují posloupnost příkazů zapsaných uživatelem. Pomocí nich můžeme vytvářet funkce nebo skripty, které nám ulehčí práci u složitějších aplikací. Tyto soubory jsou nejen základním stavebním kamenem pro tvorbu grafického uživatelského rozhraní, ale jsou také součástí všech rozšiřujících balíčků tzv. toolboxů.

Po seznámení s programem a tvorbou m-souborů se zaměřuji na nástroj pro tvorbu interaktivního grafického rozhraní. Zde popisuji GUIDE prostředí, které je v MATLABu integrováno a pomocí kterého můžeme vytvářet a editovat MATLABovské grafické objekty. Velký důraz kladu především na správný postup GUI aplikací, kde by se programátor měl řídit podle bodů popsaných v kapitole 3.1. a 3.3..

Tyto poznatky využiji ve čtvrté kapitole, kde uvedu postup vytvořené aplikace. Tato aplikace je zaměřena na téma z oblasti fyziky kmitavého pohybu.

Tuto bakalářskou práci jsem rozvrhl tak, aby čtenář prošel všemi stádii při tvorbě grafického uživatelského rozhraní. Jednotlivé kapitoly a podkapitoly na sebe navazují a spojuje je jedno hlavní téma.

1. MATLAB

Program MATLAB je vyvíjen firmou The MathWorks, Inc. od roku 1984 v USA. S tímto softwarem můžeme provádět řadu operací, spojených s matematikou, grafikou, různými signály atd. Název MATLAB vznikl zkrácením z MATrix LABoratory. Tento software je interaktivní systém, jehož základním datovým prvkem je matice, u které se nezadává rozměr. To nám umožňuje řešit mnoho numerických problémů podstatně rychleji než při použití klasických programovacích jazyků (Fortran, Basic nebo C). Snad nejoceňovanější vlastností MATLABu je jeho snadná rozšiřitelnost, která nám umožňuje doplňovat systém o námi napsané funkce (M-soubory) i celé aplikace. K MATLABu si můžeme navíc pořídit celou řadu speciálně zaměřených nadstaveb, tzv. toolboxů, což je kolekce m-souborů, která je určena pro řešení jistých tříd problémů. V současnosti jsou k dispozici nadstavby pro následující okruhy problémů:

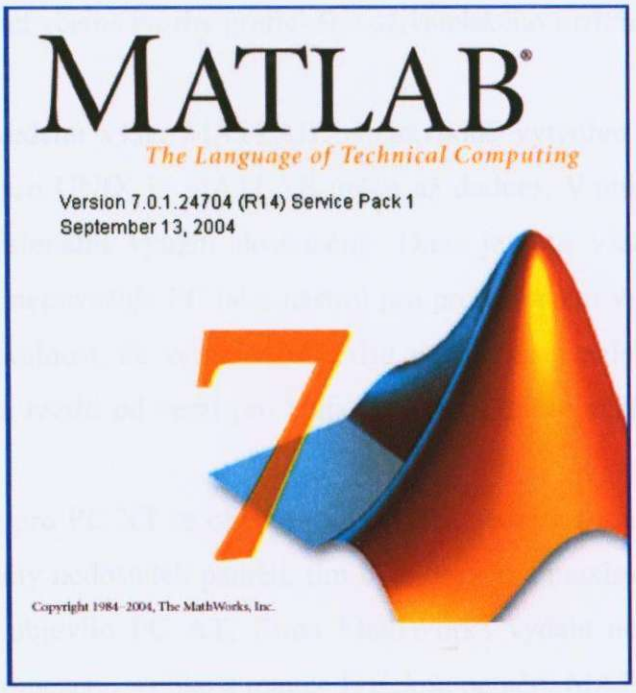
- zpracování signálů,
- zpracování obrazů,
- teorie řízení,
- identifikace systémů,
- optimalizace,
- neuronové sítě,
- spliny,
- statistika,
- symbolická matematika.

Jedna z prvních verzí MATLABu měla pro Windows následující požadavky na hardware:

- procesor 386 nebo vyšší,
- matematický koprocesor 387 nebo 487 (pokud již není součástí procesoru),
- disketovou mechaniku 31=2" (1.44 MB),
- 8 MB volného místa na disku,
- 4 MB paměti (8 MB nebo více při používání 3-D grafiky a funkcí pro zpracování obrazu).

V dnešní době už nepotřebujeme disketovou mechaniku a ani 8 MB volného místa na disku. MATLAB verze R2007b je v dnešní době dodáván na DVD a minimální požadavky na hardware jsou:

- procesor Intel Pentium IV,
- 600 MB volného místa na disku,
- 512 MB operační paměti,
- grafický akcelerátor.



Obr.1.1: Logo aplikace MATLAB.

1.1.Historie a vývoj MATLABu

Název MATLAB vznikl z anglického MATrix LABoratory. MATLAB byl vytvořen, aby poskytoval jednoduchý přístup k matematickým knihovnám, vyvinutým v projektech LINPACK a EISPACK. Původně byl určen pro operační systém UNIX a tato okolnost se dodnes projevuje v jednoduchém komunikačním rozhraním – příkazové řádce.

MATLAB je vysoce výkonný jazyk pro technické výpočty. Jde o interaktivní prostředí, jehož základním datovým typem jsou dvojrozměrná pole. Tato vlastnost spolu s množstvím zabudovaných funkcí umožňuje řešení 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,
- analýza dat,
- inženýrská grafika,
- vývoj aplikací včetně tvorby grafického uživatelského rozhraní.

Jak již bylo uvedeno výše, MATLAB byl původně vytvořen pro počítače s OS UNIX. Především pro UNIX je MATLAB určen až dodnes. V minulosti nebyl výkon počítačů pro profesionální využití dostatečný. Dnes je toto však minulostí. Firma MathWorks ovšem nepovažuje PC jako nástroj pro profesionální využití. Svědčí o tom krom jiného i ta okolnost, že verze MATLABu nejsou proti nelegálnímu kopírování zvlášť chráněny (na rozdíl od verzí pro Unix, kde je instalace vázána na sériové číslo procesoru).

První verze pro PC XT se objevila roku 1985. Jedním z hlavních problémů byl fakt, že počítače měly nedostatek paměti, tím byla omezena maximální možná velikost matic. Jakmile se objevilo PC AT, firma MathWorks vydala novou verzi pro tyto počítače. Zde byla omezena velikost matice fyzickou pamětí. MATLAB verze 386, jak už název vypovídá byla určena pro PC s procesorem 80386. Využívala jednu z možností tohoto procesoru – virtuální paměť. Znamená to, že program pracuje

s virtuální pamětí, která může být větší než fyzická paměť. Dochází k ukládání dat na pevný disk. To může způsobovat zpomalování výpočtů, ale můžeme provádět výpočty za využití opravdu velkých matic. Další verzí MATLABu byla verze 386 g z roku 1995. Všechny tyto verze pracovaly pod operačním systémem DOS, avšak verze 386 g byla nesrovnatelná v rychlosti výpočtů. V roce 1994 byla uvedena nová verze MATLABu, která se jmenovala MATLAB for Windows. Tato verze obsahovala podstatně bohatší možnosti grafického rozhraní než verze předchozí. Po výpočetní stránce toho mnoho nového nenabízela (výpočty byly na tom samém počítači pomalejší díky většímu zatížení procesoru). MATLAB for Windows skončila verzí 4.2c, která se přestalo prodávat v roce 1996. Nástupcem verze 4 byla verze 5 určená v rámci platformy Intel do 32 bitového prostředí. Tato verze je již plně 32 bitová a integruje v sobě novou možnost objektově orientovaného programování – tvorbu nových datových struktur, nové vizuální funkce, silnou podporu toolboxů v základní verzi a rychlejší grafiku. Po verzi 5 následovaly další, a to v podobě MATLAB 6 a MATLAB 6.5 a jejich release verzi až do verze MATLAB 7. V dnešní době máme MATLAB verze R2007b.

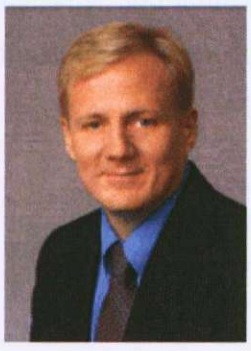
Obr.1.2: Cleve Moler.

V 70. letech se stal Cleve Moler vedoucím katedry počítačového výzkumu na univerzitě v Novém Mexiku. Tam dostal myšlenku využít knihoven LINPACK a EINSPACK k výuce kurzu lineární algebry. Jako největší problém se však ukázal samotný programovací jazyk FORTRAN. Tento jazyk se svojí složitostí nehodil pro výuku, avšak využití jeho knihoven bylo nepostradatelné.

Cleve Moler začal ve svém volném čase vyvíjet program, který by studentům umožnil snadný přístup k funkcím obsaženým v knihovnách LINPACK a EINPACK a pojmenoval jej MATLAB. V dalších letech Cleve Moler začal rozšiřovat a vylepšovat

tento programovací jazyk na univerzitních počítačích v USA. MATLAB se začal šířit mezi studenty a profesory na všech univerzitách v USA, ale i v Evropě. Díky jeho jednoduchosti je dnes využíván při řešení rozmanitých úloh reálného života.

Na začátku roku 1983 byl inženýr Jack Little seznámen s aplikací MATLAB během návštěvy profesora Cleva Molera na Standfordské Univerzitě. Ihned pochopil, že tento programovací jazyk v sobě skrývá neskutečné možnosti v inženýrských aplikací. Jack Little během téhož roku sestavuje nový tým. Za pomoci Cleva Molera a Steva Bangerta vyvinuli druhou generaci. Nyní má MATLAB už profesionální podobu. Celá aplikace byla napsána v jazyce C a již měla integrovanou grafickou formu výstupu. V roce 1984 tito tři kolegové (Cleve Moler, Jack Little, Steve Bangert) založily firmu Mathworks a vstoupili tak na softwarový trh se svou aplikaci MATLAB.



Obr. 1.3: Jack Little.

1.2. Základní struktura MATLABu

MATLAB je interaktivní prostředí, se kterým můžeme provádět:

- Matematické výpočty,
- Modelování,
- Analýzu a vizualizaci dat,
- Měření a zpracování dat,
- Vývoj algoritmů,
- Návrhy řídících a komunikačních systémů atd.

MATLAB se skládá z těchto základních částí:

- Výpočetní jádro,
- Grafický subsystém,
- Pracovní nástroje,
- Toolboxy,
- Otevřená architektura.

Výpočetní jádro provádí numerické operace s maticemi reálných či komplexních čísel. Kromě matic je podporováno i tzv. pole buněk. Mají strukturu podobnou maticím. Na rozdíl od matic může být každý prvek jiného typu (prvek může mít jinou datovou strukturu). Prvky už nejsou rozlišeny souřadnicemi, nýbrž jménem. Výpočty se implicitně provádějí v tzv. dvojnásobné přesnosti. Je však možné měnit formát zobrazeného čísla. V MATLABu můžeme pracovat i s vektory, které mohou být signály různých typů, časových řad apod.

Grafický subsystém zajišťuje snadné zobrazení výsledků. Práce s grafy je snadná a velmi rychlá. Můžeme tvořit dvojrozměrné (2D), ale i trojrozměrné (3D) grafy s mnoha možnosti nastavení.

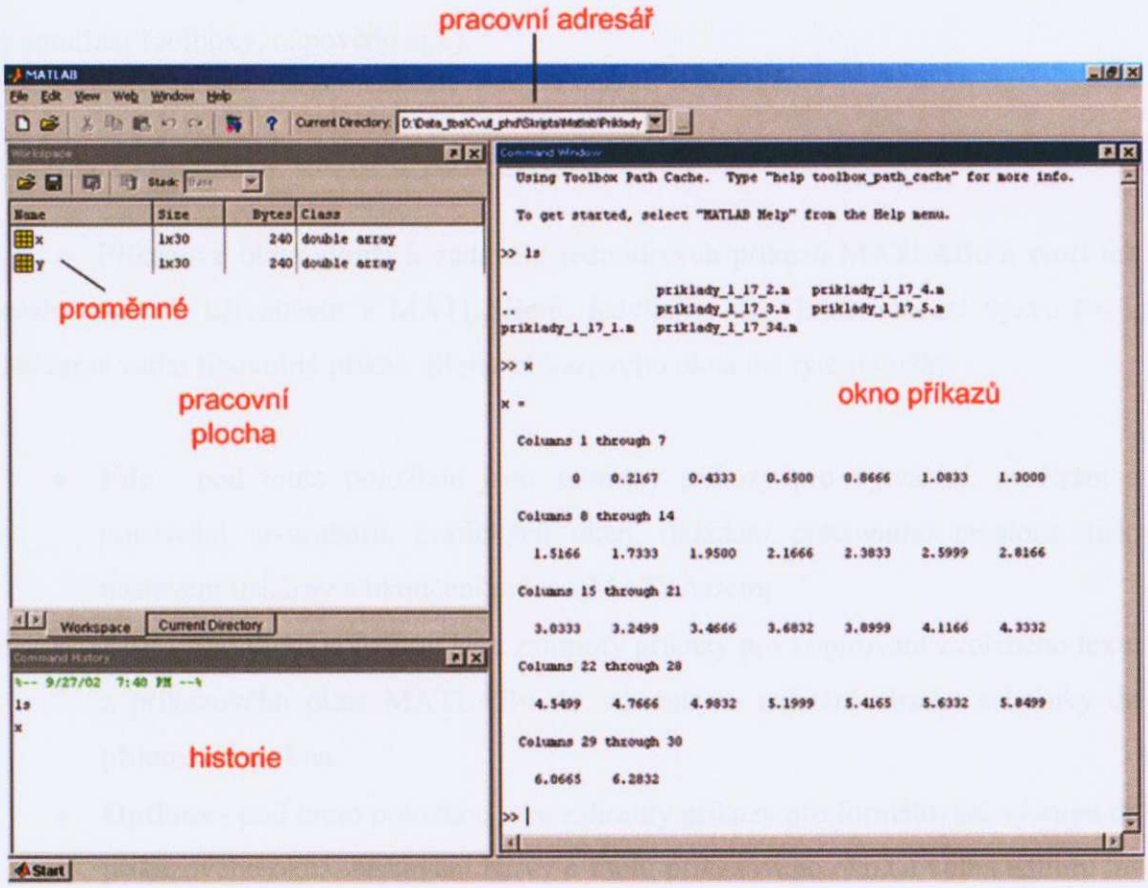
Pracovní nástroje umožňují úplné programování aplikací. MATLAB obsahuje plnohodnotný programovací jazyk čtvrté generace. Uživatel nalezne vše potřebné k programování a ladění zdrojových kódů. Systém navíc nabízí integrovanou tvorbu grafických prvků (tlačítek, menu atd.) a podporu, která usnadňuje načítání dat z jiných zdrojů. Dále si můžeme nastavit prvky pracovního prostředí podle svého.

Otevřená architektura MATLABu vedla ke vzniku knihoven funkcí nazývaných toolboxy. Toolboxy jsou specializované knihovny, které obsahující předdefinované funkce napsané v jazyce MATLAB, určené k řešení problémů v dané oblasti. Zpravidla byly vyvinuty (nebo alespoň jejich jádro) na univerzitě týmem okolo významného odborníka. Ke každému toolboxu existuje poměrně rozsáhlá dokumentace, součástí které jsou i odkazy na vědecké zdroje podrobně popisující použité algoritmy. Uživatel si je může dokoupit jako přídavné moduly. Máme toolboxy např. pro zpracování signálů a obrazů, pro práci s neuronovými sítěmi, pro návrh filmů, finance, ekonomiku aj.

Jako samostatnou kapitolu tvoří systém SIMULINK. Je to program, který využívá MATLAB a jeho funkce k simulaci dynamických systémů. Je mladší než MATLAB, jeho první verze je k dispozici s MATLABem verze 4. SIMULINK má trochu jiné uživatelské rozhraní než MATLAB. Zatímco u MATLABu je stále nejdůležitější příkazový řádek, ovládání SIMULINKu je jednodušší a intuitivnější, ale pokročilejší funkce nelze provádět bez znalosti jazyka MATLAB.

1.3. Popis prostředí při spuštění MATLABu

Pracovní prostředí, ale i samotný program MATLAB se během let měnili, což mělo za následek větší efektivnost a rychlost během psaní algoritmů. V této kapitole se zaměřím na MATLAB verze 7. Jinak toto pracovní prostředí známe již od verze 6.0.1.



Obr.1.4: Pracovní plocha MATLABu.

Historie (command window) je velmi užitečná pomůcka při psaní různých příkazů. Zobrazují se zde všechny příkazy zapsané a potvrzené v okně příkazů.

Potřebujeme-li již jednou zapsaný a potvrzený příkaz znovu použít, stačí je v tomto okně najít a poklepáním či přetažením myší je opět vložíme do okna příkazů. To samé můžeme udělat v okně příkazů pomocí kurzorových šipek nahoru a dolů.

Při prvním spuštění bude pracovní plocha (workspace) prázdná. Při používání proměnných v okně příkazů bude v tomto okně přehled všech použitých proměnných. Poklepeme-li na symbol některé proměnné, zobrazí se detailní informace o ní (rozměr, struktura atd.). To je užitečné při používání většího počtu proměnných, pro udržení přehledu.

Okno pracovní adresář (current directory) nám ukazuje seznam souborů v aktuálním adresáři či složce. Poklepáním myší na některý ze zobrazených souborů jej otevřeme ve vestavěném editoru. Složka current directory je umístěna nad oknem příkazů. Je možné ji podle potřeby změnit.

Od MATLABu verze 7 je v levé spodní části tlačítko start. Je to jistá obdoba tlačítka start v systému Windows. S jeho pomocí můžeme spouštět řadu aplikací (simulink, toolboxy, nápovědu atd.).

1.3.1. Příkazové okno a jeho menu

Příkazové okno slouží k zadávání jednotlivých příkazů MATLABu a tvoří tak rozhraní mezi uživatelem a MATLABem. Kdykoliv MATLAB zobrazí výzvu (>>), můžeme zadat libovolný příkaz. Menu příkazového okna má tyto položky:

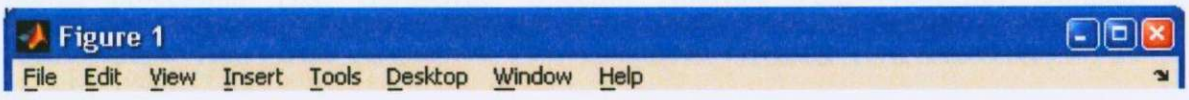
- **File** - pod touto položkou jsou zahrnuty příkazy pro vytváření, otevírání a používání m-souborů, grafických oken, ukládání pracovního prostoru, tisk, nastavení tiskárny a ukončení práce s MATLABem.
- **Edit** - pod touto položkou jsou zahrnuty příkazy pro kopírování zvoleného textu z příkazového okna MATLABu do schránky a načítání obsahu schránky do příkazového okna.
- **Options** - pod touto položkou jsou zahrnuty příkazy pro formátování výstupu do příkazového okna, nastavení barvy a fontů příkazového okna a volba editoru m-souborů.
- **Windows** - pod touto položkou je seznam všech oken MATLABu. Volbou položky se přepneme do nového aktivního okna.
- **Help** - pod touto položkou jsou zahrnuty příkazy pro nápovědu MATLABu.

Obr.1.5: Zobrazení menu příkazového okna.

1.3.2. Grafické okno a jeho menu

MATLAB zobrazuje veškeré grafické výstupy v jednotlivých grafických oknech. Grafická okna mají své vlastnosti, které můžeme modifikovat příkazy a funkcemi MATLABu. Počet grafických oken není nijak omezen. Menu grafického okna má tyto položky:

- **File** - pod touto položkou jsou zahrnuty příkazy pro vytváření nových grafických oken, tisk, nastavení tiskárny a ukončení práce s MATLABem.
- **Edit** - pod touto položkou jsou zahrnuty příkazy pro kopírování obsahu grafického okna do schránky ve formátu Windows Metafile nebo Windows Bitmap a vyčištění grafického okna.
- **Windows** - pod touto položkou je seznam všech oken MATLABu. Volbou položky se přepnete do nového aktivního okna.
- **Help** - pod touto položkou jsou zahrnuty příkazy pro nápovědu MATLABu.



Obr.1.6: Zobrazení menu grafického okna.

1.4. Toolboxy

Jak již bylo zmíněno, MATLAB je tvořen jádrem obsahujícím základní funkce a prostředky. Pro jednotlivé oblasti lze tento základ rozšířit pro problémově orientované funkce soustředěné do toolboxů (rozšiřující balíčky). Jde o funkce, které řeší specializované problémy pro dané odvětví. S využitím možností systému Handle Graphics dovolují tyto toolboxy využívat značně sofistikované matematické postupy i uživatelům nepříliš znalým v matematice. Umožní nám jednoduše zadat naše 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 takového toolboxu je Financial Toolbox (vyžaduje Statistic a Optimization toolbox) určený pro oblastbankovnictví.

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 Systém Tbx, Robust Control Tbx, Model Predictice Control Tbx , Nonlinear Control Design Tbx, LMI Control Tbx, μ-Analysis and Synthesi Tbx, Quantitative Feedback Tudory Tbx

Kromě výše uvedených toolboxů firmy MathWorks existují i toolboxy od jiných výrobců, které jsou součástí standardní nabídky a desítky dalších, které jsou vytvořeny obvykle na univerzitní půdě pro řešení problematiky v dané 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 procesy (DSP) – Fixed-Point Blockset, možnost použití DSP přímo v SIMULINKovém schématu atd…

Některé toolboxy, které jsou součástí standardní nabídky MATLABu:

- **Aerospace Blockset for Use with Simulink** - sada bloků pro SIMULINK sloužících k simulaci systémů používaných v letectví, armád a vesmírném programu.
- **MATLAB Link for Code Composer Studio (tm)** - toolbox pro komunikaci mezi programy MATLAB a Code Composer Studio přes technologickou kartu C 6701.
- **CDMA Reference Blockset for SIMULINK** - sada bloků SIMULINKu pro simulaci CDMA bezdrátových komunikačních systémů založených na americkém standardu IS – 95A CDMA (code division multiple access), nepracuje ale s Real–Time Workshopem.
- **MATLAB COM Builder** - je to dodatek compileru MATLABu, který umožňuje převádět aplikace napsané v MATLABu na COM objekty (Component Object Model).
- **Comunications Toolbox** - sada funkcí MATLABu pro návrh a analýzu pokročilých komunikačních systémů pokrývající okruhy: chybová analýza, kódování s diferenční pulsní kódovou modulací, konvoluční a lineární blokové kódování, analogová a digitální modulace, filtrace dat a výpočet Galoisových polí.
- **Communications Blockset for SIMULINK** - sada bloků SIMULINKu doplňujících systémů Communications Toolbox, pro simulaci komunikačních systémů v SIMULINKu.
- **MATLAB Compiler** - kompilátor MATLABu, který z m–souborů (výchozí soubory pro programování v MATLABu) vytváří zdrojový kód C nebo C++ nebo P kód. Dále umožňuje vytvořit pluginy kompatibilní s programem MS Excel a COM objekty.
- **Control System Toolbox** - sada funkcí pro MATLAB používáných techniky v oblasti řízení a regulace systémů.

optimalizaci rychlosti a využití paměti při vytváření aplikací v reálné čase v SIMULINKu pomocí Real–Time Workshopu.

- **Excel Link for MATLAB** - tento dodatkový software integrující MATLAB a MS Excel, který umožňuje vzájemnou komunikaci mezi těmito programy, umožňuje používat a volat funkce MATLABu z Excelu.
- **Filter Design Toolbox** - toolbox pro obsahující sadu pokročilých filtrů. Tento toolbox podporuje návrh, simulaci a analýzu filtrů s fixním bodem a též filtrů s měnícím se bodem.
- **Fix–Point Blockset for SIMULINK** - sada bloků pro SIMULINK pro vytváření časově diskrétních dynamických systémů, které využívají aritmetiku fixního bodu. Mohou se tak simulovat efekty, které jsou běžné u systémů s fixním bodem např. při filtraci v časové oblasti nebo při regulaci. Obsahuje speciální datové typy jako např. 11 bitový exponent nebo 52 bitový zlomek.
- **Fuzzy Logic Toolbox for MATLAB** - pro návrh a simulaci fuzzy systémů.
- **System Identification Toolbox** - toolbox pro tvorbu přesných a jednoduchých matematických modelů systémů ze zašuměných časových řad dat.
- **Image Processing Toolbox** - toolbox pro práci s obrázky s možností editace grafických procedur. Umožňuje různé transformace obrázků, filtrace, změny kvality vybraných částí obrázků atd.
- **Instrument Control Toolbox** - toolbox poskytující rozhraní pro komunikaci s přístroji podporujícími rozhraní IEEE-48, VISA, protokoly TCP/IP nebo UDP a sériový port (RS–232, RS–422 a RS–485). Data přenášená mezi MATLABem a přístrojem mohou být binární nebo textová (tj. příkazy SCPI jazyce). Přenos může být synchronní nebo asynchronní a je řízen pomocí událostí.
- **Mapping toolbox for MATLAB** - toolbox umožňující čtení, analýzu a zobrazení geografických dat v MATLABu s možností pravidelného importu geografických dat z Internetu.
- **Model–Based Calibration Toolbox for MATLAB** - toolbox pro návrh experimentu, statické modelování a kalibraci systémů.
- **Model Predictive Control Toolbox** - sada příkazů pro analýzu a návrh systémů s modelové prediktivním řízením.
- **mi-Analysis and Synthesis Toolbox** - sada funkcí pro analýzu a syntézu regulovaných systémů s modelovou čili parametrickou neurčitostí.

- **Nonlinear Control Design Blockset for SIMULINK** - poskytuje grafické rozhraní pro návrh robustního nelineárního regulátoru v časové oblasti v SIMULINKu.
- **Neural Network Toolbox** - toolbox pro práci neuronovými sítěmi.
- **Optimization Toolbox** - toolbox rozšiřující výpočetní schopnosti MATLABu, obsahuje rutiny pro mnoho typů optimalizace, jako např. neohraničenou nelineární minimalizaci, kvadratické a lineární programování, nelineární nejmenší čtverce, řešitel systému nelineárních rovnic.
- **Partial Differential Equation Toolbox** - nástroj pro studium a řešení parciálních diferenciálních rovnic ve dvou rozměrech a čase. Rovnice jsou diskretizovány pomocí metody konečných prvků (FEM).
- **SimMechanics for SIMULINK** - nástroj pro modelování mechanických systémů pomocí SIMULINKu.
- **SimPowerSystems for SIMULINK** - nástroj pro tvorbu modelů simulujících elektrárenské systémy.
- **Requirements Management Interface for SIMULINK** - rozhraní umožňující spojení požadavků simulinkových modelů, Stateflow a matlabovských m–souborů. Požadavky jsou v HTML, MS Wordu nebo MS Excelu.
- **Robust Control Toolbox for MATLAB** - toolbox pro návrh robustních regulátorů
- **Report Generátor** - software, který umožňuje vzít libovolnou informaci z MATLAB Workspace a exportovat do dokumentu ve formě zprávy formátu html, sgml, xml (podporuje Word 2003), rtf.
- **Real–Time Workshop for SIMULINK** - nástroj rozšiřující SIMULINK o schopnost rychlejšího chodu vytvořeného programem v reálném čase díky převodu modelu SIMULINKu na C kód a jeho převod do spustitelné aplikace běžící v reálném čase.
- **Real–Time Windows Target for Real-Time Workshop** - je to součást pro tvorbu regulačních systémů pracujících v reálném čase. Jedná se o systém, kde hostující a cílový počítač je jeden a ten sám. Vyžaduje PC kompatibilní počítač, I/O technologickou kartu a OS Windows 98, NT 4, Millenium, XP, 2003. Umožňuje regulaci systémů pracujících v reálném čase přes technologickou kartu s využitím grafického rozhraní SIMULINKu.
- **Matlab Runtime Server** - je to varianta MATLABu, díky které mohou softwarový vývojáři spojit svou aplikaci, která využívá MATLAB. Tato varianta MATLABu

obsahuje všechny grafické a výpočetní schopnosti komerčního MATLABu a je navržena tak, aby mohla spouštět samostatné aplikace MATLABu. Běžný uživatel Runtime Server aplikace nepotřebuje vlastnit MATLAB ani ho nemusí ovládat. Všechny m-soubory a modely je třeba převést na p–soubory.

1.5. SIMULINK

SIMULINK je program pro simulaci a modelování dynamických systémů, který využívá algoritmy MATLABu pro numerické řešení nelineárních diferenciálních rovnic. Jeho přítomnost v MATLABu je až od verze 4, kde využívá grafických možností hostitelské platformy (MS Windows®). Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic. Nový přístup k řešení diferenciálních rovnic dovoluje simulovat i rozsáhlé systémy rychle, přesně a s efektivním využitím paměti počítače.

SIMULINK je určen na časové řešení – simulaci – chování dynamického systému pokud známe jeho matematický popis. Můžeme tak určovat časové průběhy výstupních veličin v závislosti na časovém průběhu veličin vstupních s jejich počátečním stavem.

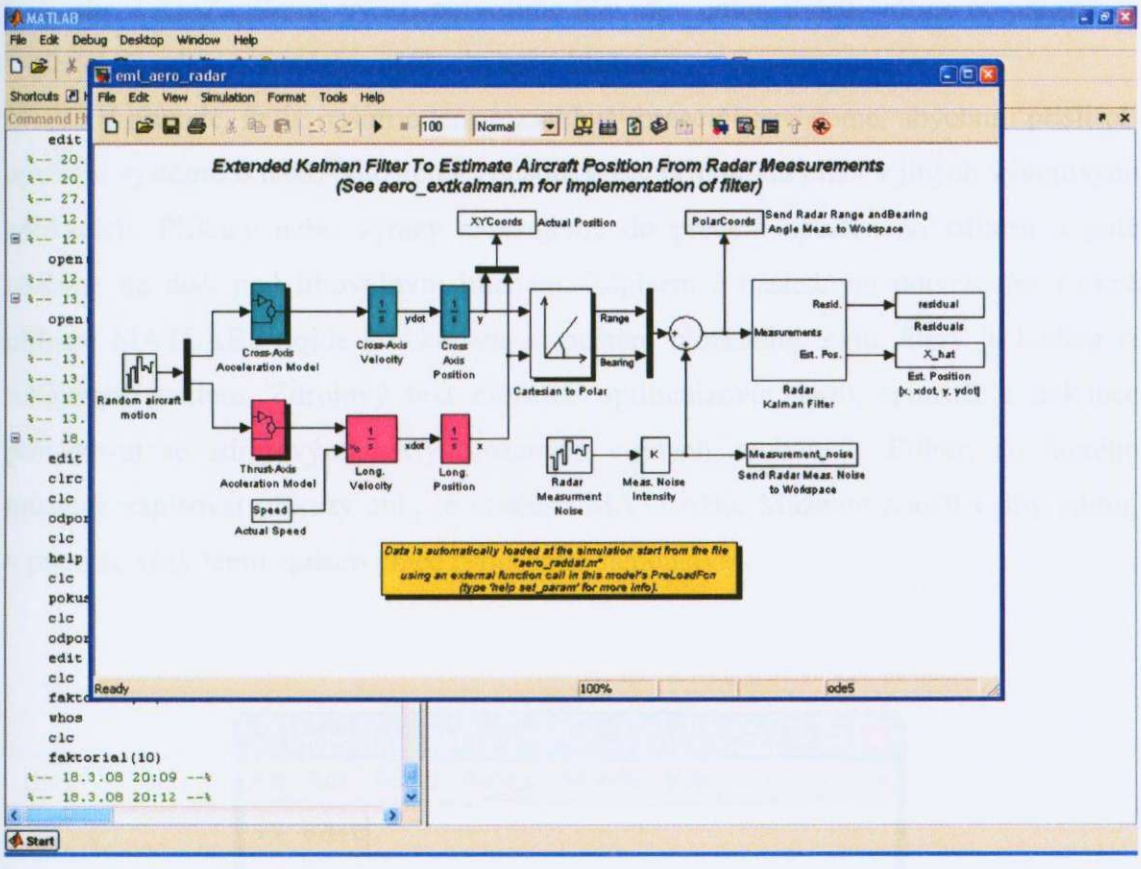
Přístup k návrhu zápisu problému silně připomíná návrh zapojení pro analogový počítač. Na první pohled zaujme grafický způsob zápisu, který se nazývá model. Z nabídky knihoven se myší přetahují výkonné bloky a pak se myší pospojují odpovídající vstupy a výstupy.

Složitější modely můžeme vytvářet tak, že sdružíme část modelu do dalšího bloku a následně je můžeme kopírovat či spojovat. Libovolné skupiny bloků lze uzavřít do subsystému a určit externí vstupy a výstupy této skupiny. Se skupinou se dá již pracovat jako se základním blokem.

Simulace chování dynamických systémů se neskládá jen z numerického řešení soustavy nelineárních diferenciálních rovnic (i když patří mezi nejdůležitější části). Pokud řešíme nějaký případ, musíme dbát na zvolení vhodné metody, zahrnutí počátečních stavů, volbu kroku řešení atd.

Systémy, které mohou být řešeny v diskrétním čase (lze je diskretizovat), můžeme přes RTW převést přímo do zdrojového kódu v jazyku C, který lze po doplnění zdrojovými texty uživatelem přímo kompilovat do strojových jazyků počítačů určených k řízení. Pak vytvoření i vcelku složitého systému může vypadat tak, že návrhář sestaví model navrhovaného požadovaného systému z předdefinovaných bloků SIMULINKu a po stisknutí jediného tlačítka (je-li již správně zvolena konfigurace) dojde ke kompletnímu překladu a stažení výsledného kódu do řídící jednotky.

Kromě vlastního grafického prostředí je SIMULINK vybaven standardními knihovnami, popřípadě rozšířen toolboxy. Dále podporuje příkazy a funkce dostupné přímo z prostředí MATLABu.

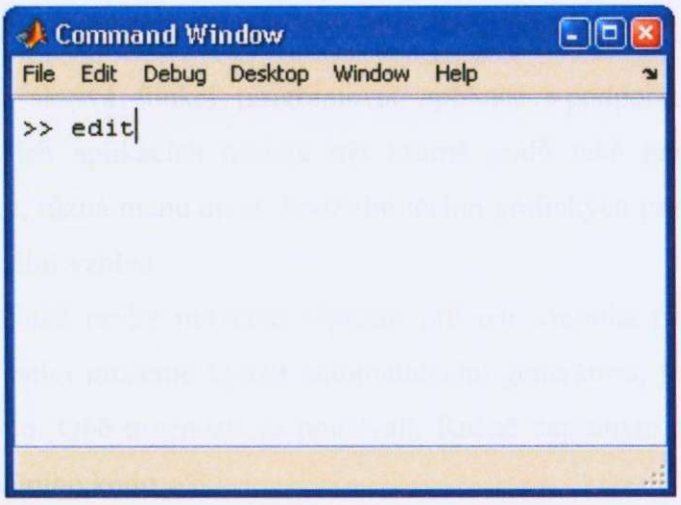


Obr.1.7: Ukázka aplikace v SIMULINKu.

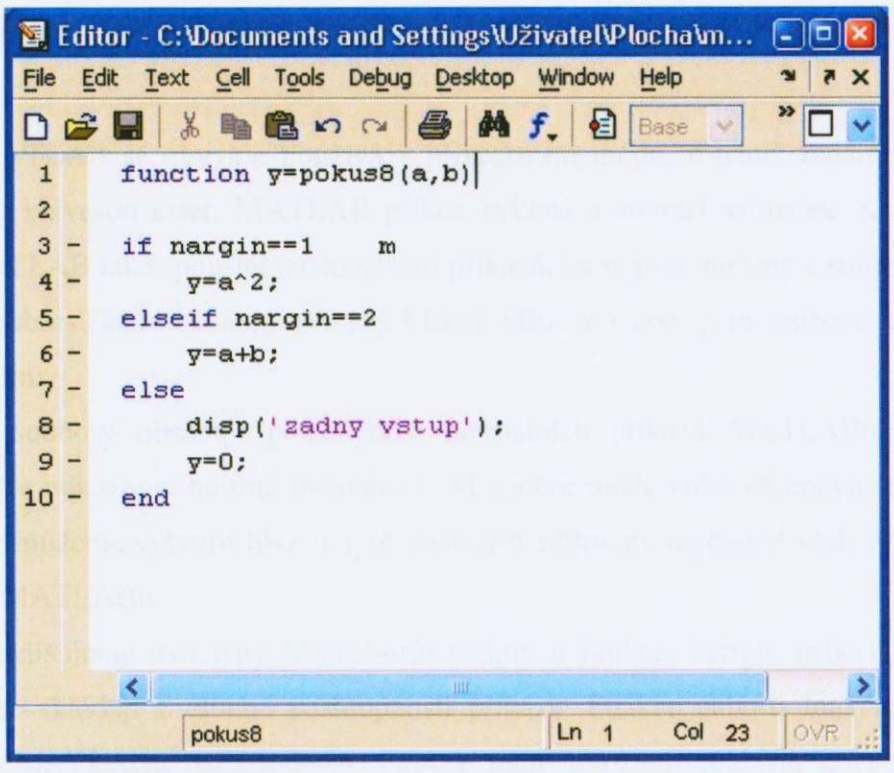
2. Úvod do tvorby uživatelských aplikací

Se systémem MATLAB můžeme pracovat několika způsoby. Pokud potřebujeme spočítat jen něco narychlo, použijeme klasický řádkový dialogový způsob (viz. obr. 1.4). Zapíšeme výraz, potvrdíme klávesou enter, ihned vidíme na obrazovce výsledek či odezvu systému, např. chybové hlášení.

V případě, že je naším cílem rozsáhlejší práce či nechceme, abychom přišli při opuštění systému o data, použijeme způsob, který připomíná práci v jiných vývojových aplikacích. Příkazy nebo výrazy zapisujeme do předem spuštěného editoru a poté uložíme na disk pod libovolným jménem. Zápisem a následným potvrzením v okně příkazů MATLABu dojde k vykonání a spuštění uloženého textu, který je kódem či zdrojovým textem. Zdrojový text můžeme optimalizovat, ladit, spouštět a dokonce provazovat se zdrojovými texty uložených v jiných souborech. Editor, do kterého můžeme zapisovat příkazy atd., je součástí MATLABu. Můžeme použít i jiný editor, v praxi se však tento způsob práce příliš často nepoužívá.



Obr.2.1: Vyvolání vestavěného editoru z příkazového okna.



Obr.2.2: Příklad zápisu v editoru MATLABu.

Výsledkem naší práce mohou být soubory, které mohou obsahovat velmi rozsáhlé zdrojové texty a mohou být provázány i s jinými již vytvořenými soubory. MATLAB v sobě skrývá funkci programovat aplikace s podporou grafiky. Z toho vyplývá, že v našich aplikacích mohou být kromě grafů také grafické prvky jako tlačítka, posuvníky, různá menu apod. Použitím těchto grafických prvků může mít naše aplikace profesionální vzhled.

Tvořit grafické prvky můžeme zápisem příkazů s mnoha parametry v editoru zdrojových textů nebo můžeme využít automatického generátoru, jenž je do systému MATLAB vestavěn. Obě možnosti se používají. Ručně napsaným zdrojovým textem dosáhneme optimálního kódu.

2.1. Práce s m-soubory

MATLAB se obvykle používá v příkazovém módu, v němž zadáme příkaz a potvrdíme klávesou ester. MATLAB příkaz vykoná a zobrazí výsledek. Kromě toho může MATLAB také spouštět posloupnosti příkazů, které jsou uloženy v souborech.

Soubory, které obsahují příkazy MATLABu, se nazývají m-soubory nebo mají příponu *.m.

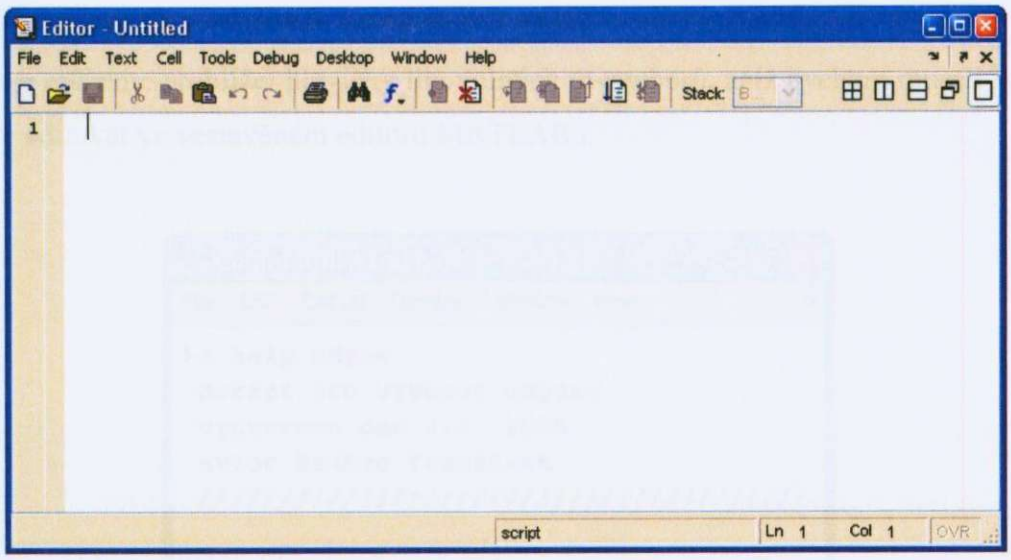
M-soubory obsahují posloupnost normálních příkazů MATLABu, které se mohou dále odkazovat na jiné m-soubory. M-soubor může volat rekursivně sám sebe. M-soubor můžeme vytvořit libovolným textovým editorem, nejčastěji však vestavěným editorem MATLABu.

Rozlišujeme dva typy M-souborů: scripty a funkce. Scripty nebo-li scriptové soubory se skládají z dlouhé posloupnosti příkazů. Funkce nebo-li funkční soubory poskytují MATLABu rozšiřitelnost. Funkce nám umožňují přidávat nové funkce k funkcím existujícím. Jednoduše napsáno, scripty neumí to, co umí funkce a naopak. Za svoji oblíbenost vděčí MATLAB v mnohém právě své schopnosti vytvářet nové funkce, které řeší uživatelem specifikované problémy. Scripty i funkce jsou obyčejné textové (ASCII) soubory.

2.2. Vytvoření nového m-souboru

Jak už jsem výše popsal, m-soubor je obyčejný textový soubor s příponou *.m. M-soubor obsahuje příkazy a povely a je většinou uložen na pevném disku. Jak již bylo také řečeno, nejlepším a nejjednodušším způsobem jak vytvořit m-soubor je použití vestavěného editoru MATLABu. Vestavěný editor vyvoláme v okně příkazů povelem edit a následně potvrdíme klávesou enter, viz obr. 2.1.

Odezvou na povel edit je otevření vestavěného editoru MATLABu m-souborů, viz. obr.2.3.

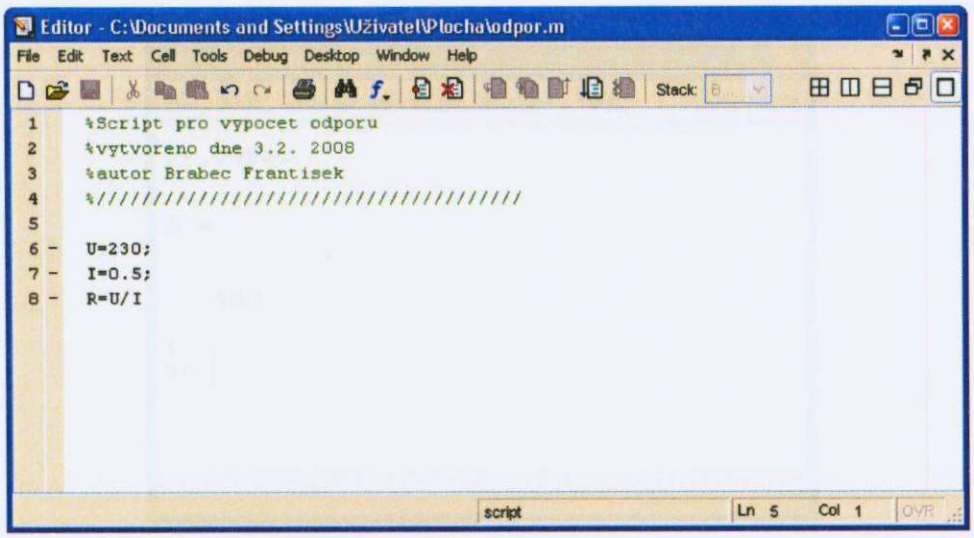


Obr. 2.3: Prázdné okno vestavěného editoru m-souborů.

2.3. Tvorba scriptů

Script označujeme jako nejjednodušší m-soubor. To znamená, že obsahuje pouze posloupnost příkazů, který se používají v příkazovém okně MATLABu.

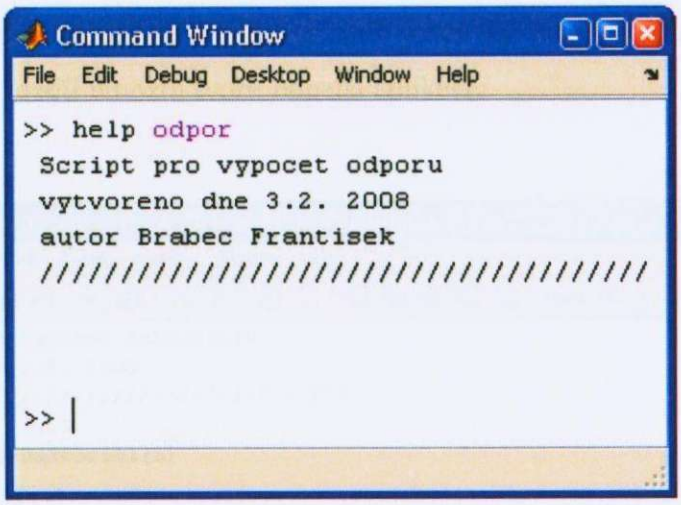
Skript obsahuje posloupnost příkazů pod názvem *odpor*, viz. obr. 2.4.



Obr. 2.4: Obsah scriptu odpor.m.

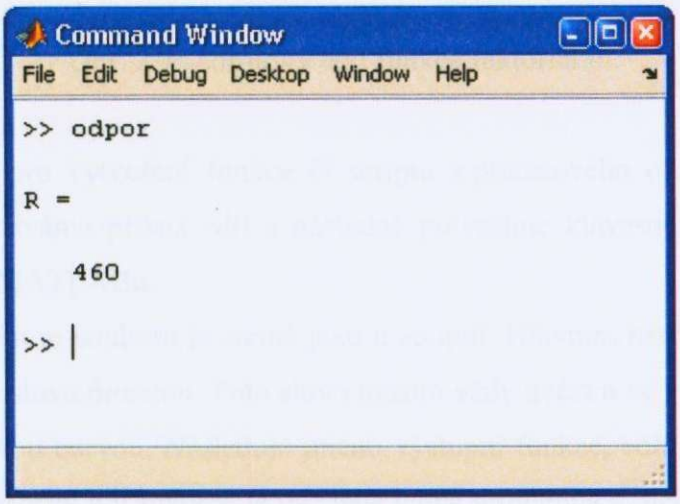
První 4 řádky, které začínají znakem %, jsou poznámky k programu (komentáře). Pomáhají nám k udržení přehlednosti v m-souborech. Mají však ještě dosti zajímavou funkci. V okně příkazů zapíšu: help odpor a potvrdím klávesou enter. Ihned

dojde k výpisu již zmíněných prvních čtyř řádků vytvořeného scriptu, viz. obr. 2.5. Takto si můžeme prohlížet hlavičky libovolných m-souborů, aniž bychom museli každý soubor editovat ve vestavěném editoru MATLABu.



Obr.2.5: Výpis hlavičky m-souboru v okně příkazů.

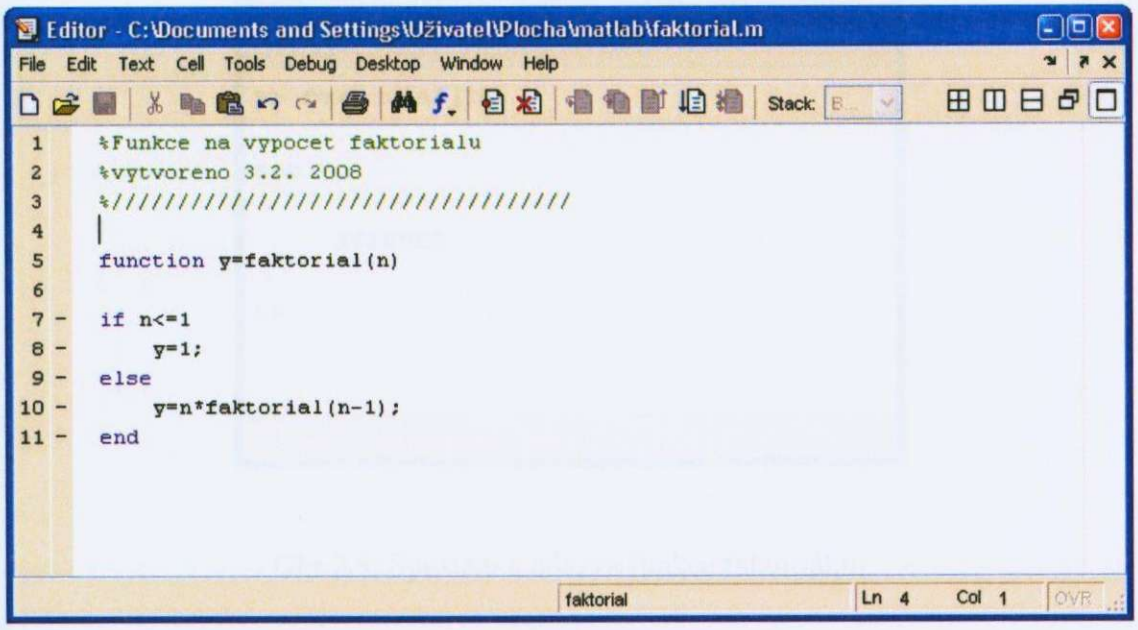
Nyní již zbývá vytvořený soubor z obr. 2.4 spustit. V okně příkazů MATLABu napíši jméno nového scriptu bez přípony, viz. obr. 2.6.



Obr. 2.6: Spuštění scriptu z prostředí MATLABu.

2.4. Tvorba funkcí

Jak již zde bylo zmíněno, existuje kromě scriptů ještě jeden typ m-souborů a to jsou funkce. Funkce nabízejí mnohem větší možnosti než scripty. Hlavní rozdíl je v tom, že funkce mohou, ale i nemusí, mít jeden či více vstupních a výstupních parametrů. Tento rozdíl umožní tvořit bohatší aplikace.



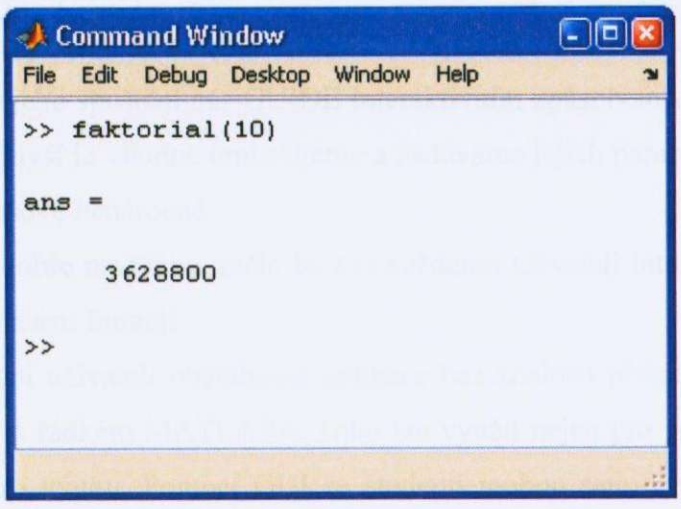
Obr. 2.7: Zdrojový text funkce faktorial.m.

Příkaz pro vytvoření funkce či scriptu z příkazového okna MATLABu je stejný. Opět používáme příkaz edit a následně potvrdíme klávesou enter. Vyvolá se vestavěný editor MATLABu.

Hlavička m-souboru je stejná jako u scriptů. Hlavním rozdílem od scriptu je použití klíčového slova *function*. Toto slovo musím vždy uvézt a ve vestavěném editoru je označeno modrou barvou. Následuje jméno výstupní funkce, což je y. Potom název funkce, což je *faktorial* a v závorce je uvedená jedna proměnná n, neboť chceme použít jen jeden vstupní parametr. Pokud máme více vstupních a výstupních proměnných, musíme je oddělit čárkou. Název proměnné či název funkce záleží jen na nás. Nesmíme jen zapomenout na uložení zdrojového kódu. Doporučuji ukládat pod stejným názvem,

jako je název funkce ve zdrojovém textu. Název funkce nesmí začínat číslem a nesmí to být rezervované slovo.

V kulatých závorkách je uveden jeden či více vstupních parametrů. V našem případě je uveden pouze jeden vstupní parametr a to proměnná n. Tato jednoduchá funkce vypočte faktoriál libovolného čísla. Spuštění funkce v okně příkazů MATLABu je na obrázku 2.8.



Obr.2.8: Spuštění a odezva funkce faktoriál.m.

Samozřejmě, že máme různé druhy funkcí:

- Funkce s jedním vstupním parametrem,
- Volání funkce uvnitř skriptu,
- Volání funkce uvnitř jiné funkce,
- Funkce bez vstupního parametru,
- Funkce volající sama sebe,
- Funkce se dvěma či více vstupními parametry,
- Funkce s proměnným počtem vstupních parametrů,
- Funkce s jedním výstupním parametrem,
- Funkce se dvěma či více vstupními parametry ,
- Funkce s proměnným počtem výstupních parametrů,
- Funkce se vstupními a výstupními parametry.

3. GUI- Graphical User Interface

Grafické uživatelské rozhraní GUI (Graphical User Interface) je uživatelské rozhraní sestavené z grafických objektů (komponent) jako jsou tlačítka, textová pole, posuvné seznamy, nabídky a pod. Uživatel si může vytvářet grafické objekty sám přímo v editoru zdrojových textů. V takovém případě můžeme napsat, že vytvořené GUI je optimální. Dalším způsobem jak si uživatel může vytvářet grafické objekty je integrovaný nástroj, který se nazývá GUIDE (Graphical User Interface Development Environment). Po jeho spuštění nás GUIDE interaktivním způsobem vede. Vybíráme si grafické objekty, myší je vhodně umisťujeme a zadáváme jejich parametry. Toto řešení je univerzální a časově nenáročné.

Je-li GUI dobře navrženo, mělo by být každému uživateli intuitivně zřejmé, jak jeho jednotlivé součásti fungují.

GUI umožní uživateli obsluhovat aplikace bez znalostí příkazů potřebných při práci s příkazovým řádkem MATLABu. Toho lze využít nejen pro praktická měření a simulace, ale i pro výuku. Pomocí GUI se studenti mohou sami detailně seznámit s vlastnostmi vybraných algoritmů zpracování signálů. V průběhu běhu aplikace můžeme měnit hodnoty různých parametrů algoritmu a sledovat tak projevy těchto změn na výstupu (resp. výstupech) algoritmu.

3.1. Základní parametry tvorby grafického uživatelského rozhraní

Při tvorbě grafického uživatelského rozhraní je třeba mít na mysli základní myšlenku GUI- snadné ovládání aplikace uživatelem. To znamená, že musíme splnit během celého běhu programu tři základní požadavky:

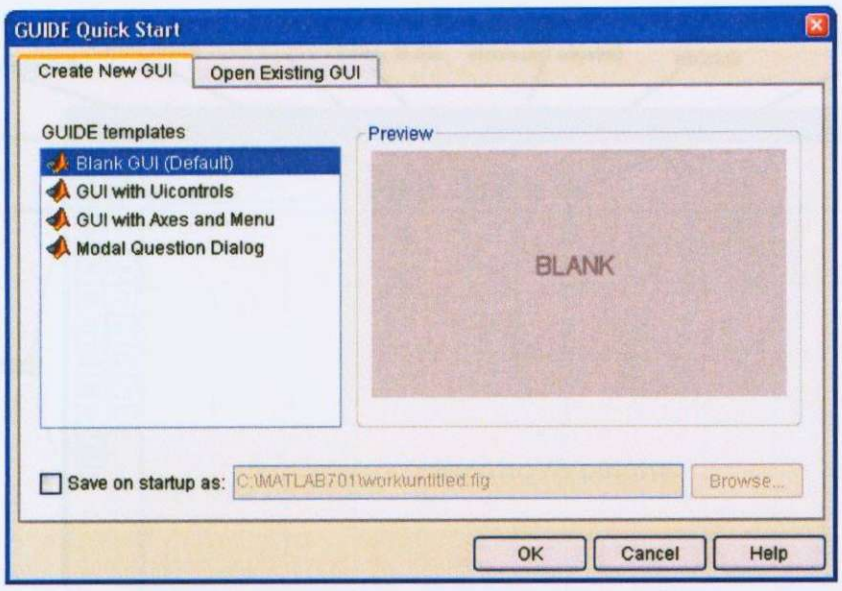
- **JEDNODUCHOST**- pohyb v naší aplikaci by měl vždy být jednoduchý a rychlý. Kliknutí na tlačítko či přepínač je vždy snadné a rychlé.
- **PROVÁZANOST**- uživatel se nesmí během běhu našeho programu ocitnout ve slepé uličce bez možnosti návratu
- **KOMPLEXNOST** (ošetření všech eventualit)

3.2. Prostředí GUIDE

Aplikace v grafickém uživatelském rozhraní se dají psát dvěma způsoby. První způsob je takový, že celá aplikace je vytvořena jen za použití m-file editoru. Tento způsob je velmi obtížný a bez větších znalostí nepoužitelný. Druhým způsobem je využití GUIDE v kombinaci m-file editoru.

GUIDE je integrované vývojové prostředí pro GUI aplikace. Pomocí něho můžeme vytvářet a upravovat uživatelská rozhraní a to prostřednictvím push butonů, list boxů, pull-down menu a dalších. Návrhové nástroje GUIDE jsou následující:

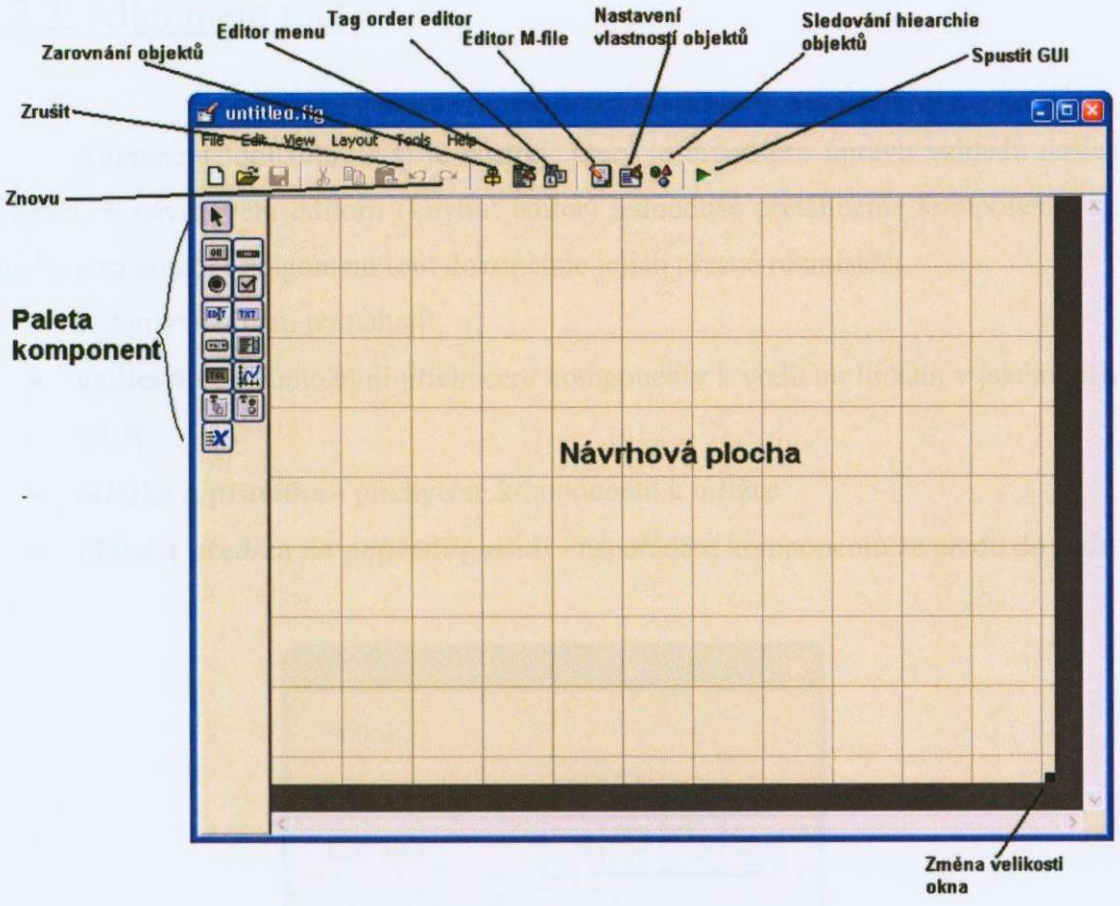
- **Layout Editor** – přidává a uspořádává objekty GUI do návrhu.
- **Alignment Tool** – seřadí objekty navzájem mezi sebou.
- **Property Inspector** – nastavuje a dohlíží na vlastnosti objektu.
- **Object Browser** – sleduje seznam ukazatelů objektu v aktuální sekci MATLABu.
- **Menu editor** – vytváří řádkové menu okna a kontextové menu.
- **Tag order editor** – mění pořadí, v kterém jsou prvky vybrané tabulátorem.



Obr. 3.1: Průvodce GUIDE.

3.2.1. Layout editor

Všechny tyto nástroje jsou přístupné z Návrhového editoru (Layout Editor- obr. 3.2). Pro spuštění Návrhového editoru použijeme příkaz guide z příkazové řádky MATLABu nebo z menu **File** -> **New** -> **GUI**. Otevře se dialog GUIDE Quick Start (obr. 3.1). Ponecháme přednastavenou volbu a klikneme na tlačítko OK. Otevře se Návrhový editor z kterého ovládáme ostatní nástroje s prázdnou návrhovou plochou. Návrhový editor umožňuje snadno a rychle vytvořit GUI přetažením součástí jako jsou tlačítka (Buttons), vysouvací nabídky (Popup Menu) nebo osy (Axes), z palety komponent do návrhové plochy. Vybereme komponentu kterou chceme umístit v GUI a klikneme na ni v paletě komponent. Kurzor se v návrhové ploše změní na kříž, který můžeme použít pro výběr pozice levého horního rohu komponenty nebo můžeme nastavit libovolnou velikost komponenty kliknutím v návrhové ploše a pak tažením kurzoru k spodnímu pravému rohu předtím, než uvolníme tlačítko myši. Paleta komponent obsahuje všechny komponenty, které lze využít při návrhu rozhraní. Návrhová plocha se při aktivaci stává vlastním GUI. Návrhový editor ukazuje obr.3.2.



Obr. 3.2: Návrhový editor (Layout editor).

Jakmile uložíme nebo spustíme GUI, GUIDE automaticky vytvoří dva soubory se stejným jménem, lišící se pouze příponou:

- **FIG-file** – soubor s příponou *.fig, který obsahuje kompletní popis grafické části GUI a všechny jeho součásti, právě tak, jako hodnoty všech vlastností objektu. Soubor editujeme v návrhovém editoru GUIDE.
- **M-file** – soubor s příponou *.m, který obsahuje kód ovládající GUI včetně callback funkcí jeho komponent. Tento soubor označujeme jako GUI m-file. Když poprvé spustíme nebo uložíme GUI z návrhového editoru, GUIDE vygeneruje GUI m-file s prázdnými útržky kódu pro každou callback funkci. Ty musíme naprogramovat v m-file Editoru sami.

Vytvoření GUI zahrnuje tedy dvě základní úlohy:

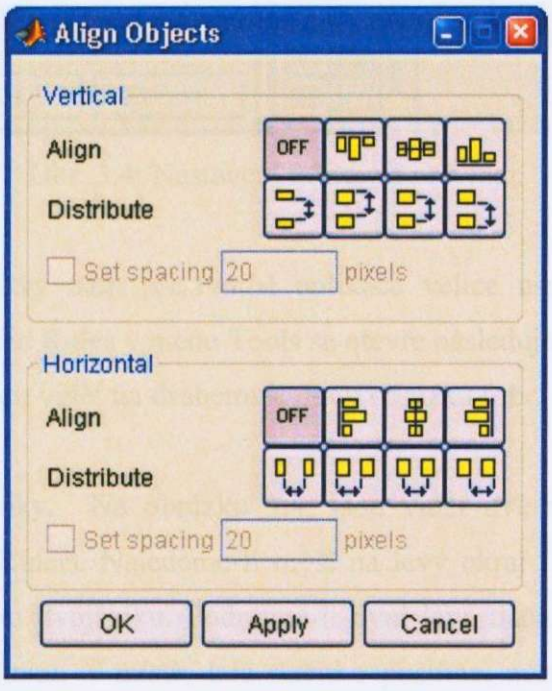
- Vybrat komponenty a uspořádat je v návrhovém editoru (Layout Editor) GUIDE.
- Naprogramovat callback funkce použitých komponent.

3.2.2. Alignment tool

Alignment tool (obr. 3.3) je nástroj, který je určen pro úpravu vzhledu našich aplikací. V návrhovém editoru (Layout editor) jednoduše přetáhneme komponenty na plochu a za pomoci Alignment tool dokončíme jejich přesné rozmístění.

K zarovnání nám pomáhají:

- **vodící linky** - umožňují přichycení komponenty k vodícím linkám v jakémkoliv místě
- **mřížka a pravítko** - přichycení komponentů k mřížce
- **přenést před/za na popředí/pozadí** - uspořádání komponentů ze předu do zadu



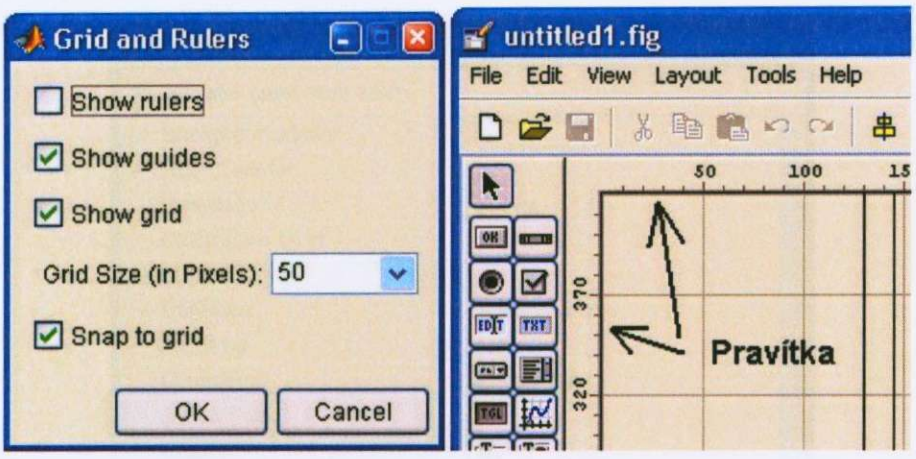
Obr. 3.3: Zarovnávací nástroj (Alignment tool).

Zarovnávací nástroj (Alignment tool) pro zarovnávání grafických objektů na:

- horní nebo levý okraj,
- na středy,
- na spodní nebo pravý okraj.

Nebo jejich rozložení na všechny možné způsoby:

- pevná vzdálenost mezi objekty,
- pevná vzdálenost mezi horními nebo levými okraji,
- pevná vzdálenost mezi středy objektů,
- pevná vzdálenost mezi spodními nebo levými okraji,



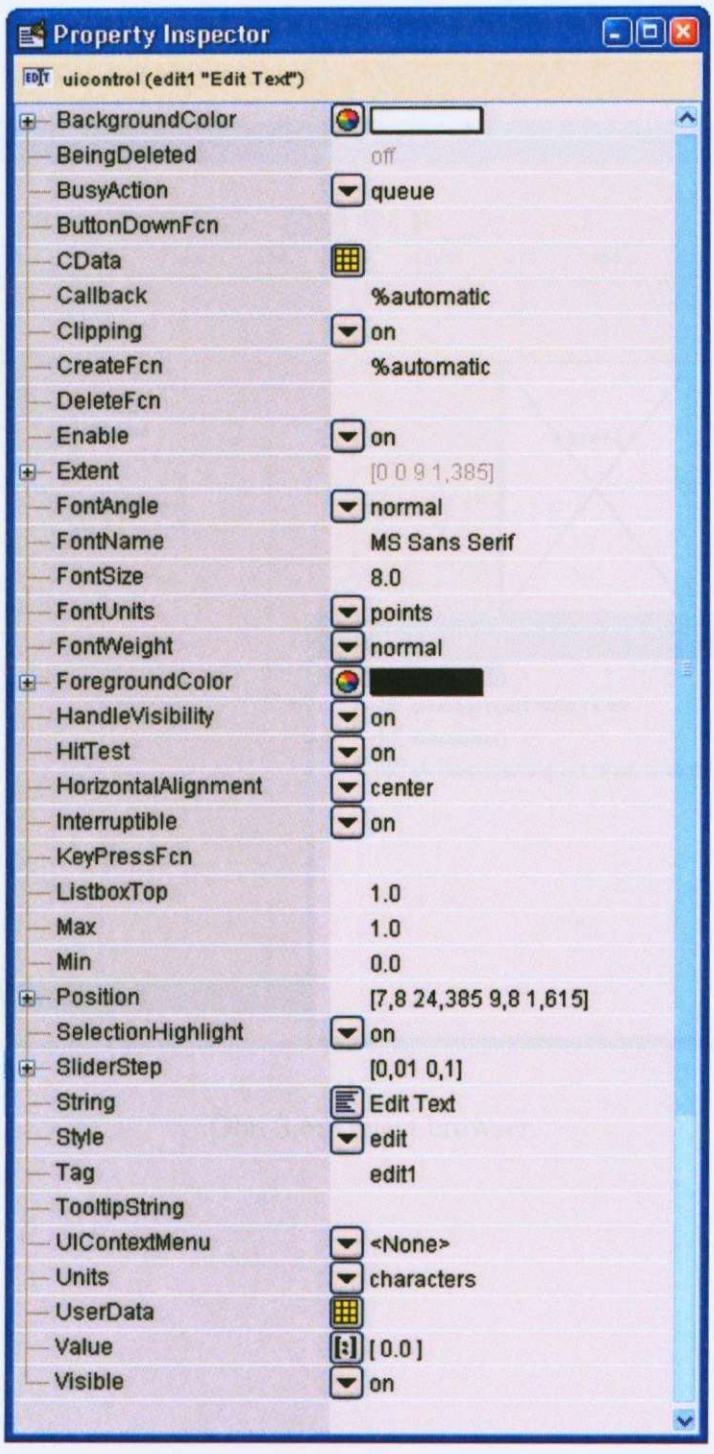
Obr. 3.4: Nastavení mřížky a pravítka.

Pravítka a mřížky nám při tvorbě aplikace velice usnadňují zarovnání. Po zvolení položky Grid and Rules v menu Tools se otevře následující okénko viz. obr. 3.4. Rules jsou pravítka. Jsou vidět na druhém ze dvou obrázků (obr. 3.4). Stupnice je po 50 pixelech.

Guides jsou vodící linky. Na obrázku 3.4. jsou vidět dvě vertikální. Získáme je vytažením z měřítka (Ruler). Najedeme-li myší na levý okraj Layout Area, změní se nám kurzor na opačnou dvojšipku. Podržíme-li nyní levé tlačítko a táhneme vpravo, vytáhneme si vodící linku. V místě, kde tlačítko pustíme, zůstane linka zafixována. Tento postup lze opakovat několikrát a vytahat si libovolné množství linek.

3.2.3. Property inspektor

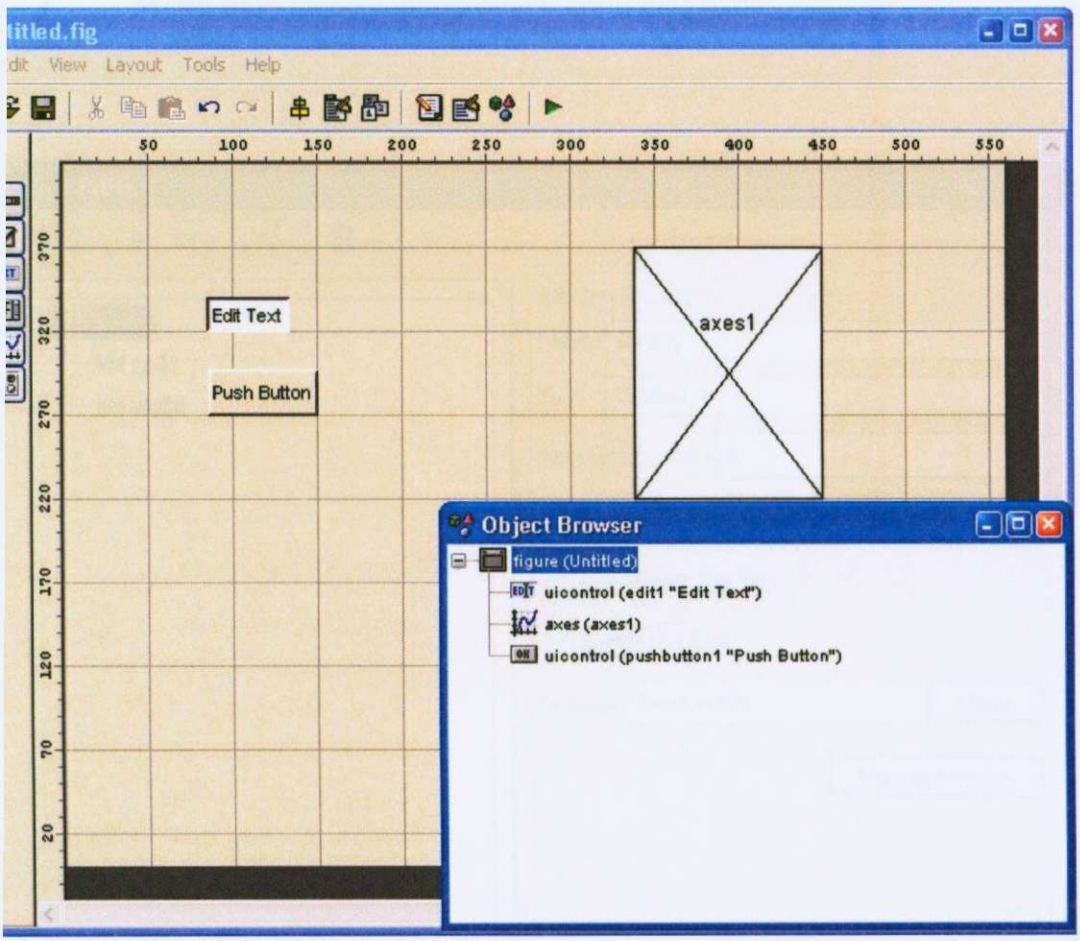
Property inspektor slouží k editaci a zobrazení vlastností námi vytvořených objektů. Každý objekt se liší počtem položek v Property inspektoru, avšak určitý počet položek je shodný pro všechny typy objektů (např. barva, barva pozadí, identifikátor, viditelnost…).



Obr. 3.5: Property inspektor.

ł. Object browser

Object browser zobrazuje celý seznam objektů v pořadí, jak byly vytvářeny. ɔj poslouží také k jednoduché orientaci mezi objekty a k přístupu vlastností ɔerty inspecturu. Stačí jen poklepat myší na název objektu a ihned vyvoláme rty inspektor.

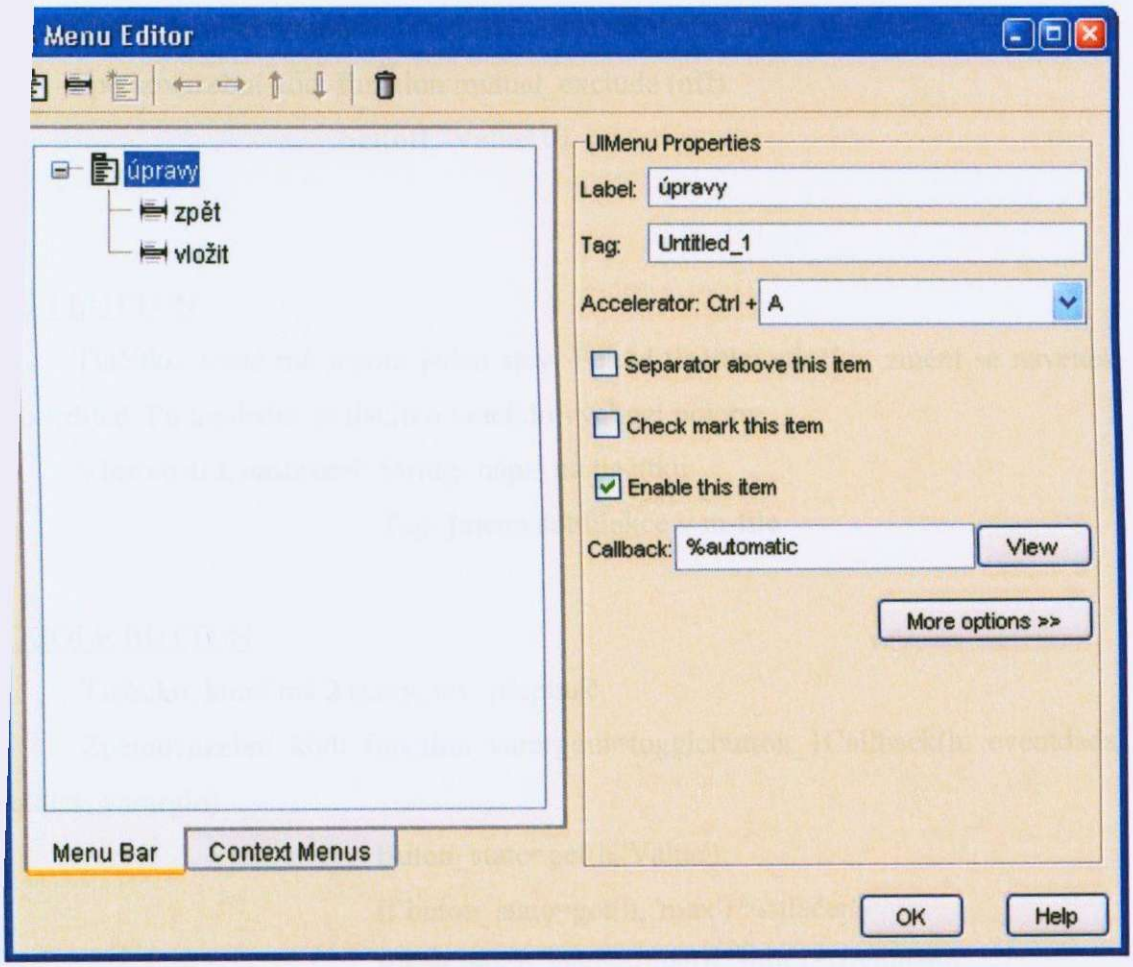


Obr. 3.6: Object browser.

5. Menu editor

Při tvorbě obsáhlejších aplikací se neobejdeme bez menu, které zajišťuje přístup atním funkcím dané aplikace. V GUIDE již existuje menu editor, který byl navržen návrh takovýchto menu. Z obr. 3.7 je zřejmé, že můžeme navrhovat dva druhy u:

- **Kontextové menu-** menu, které se zobrazí po kliknutí pravého tlačítka myši na objektu
- **Menu bar-** menu zobrazované v horní liště okna.



Obr. 3.7: Menu editor.

3.2.6. Přehled jednotlivých Uicontrol objektů a jejich zpětnovazební kódy

V této kapitole bych se rád zaměřil na položky grafických objektů, jejich zpětnovazební kódy, které lze využít při „oživování" grafického uživatelského rozhraní.

RADIO BUTTON

Je velmi podobné funkcí zaškrtávacímu políčku (checkbox). Radionbutton má 2 stavy „vybrán-nevybrán", což je uchováno ve vlastnosti value „max-min" (0 nebo 1). Každý radiobutton musí obsahovat subfunkci. Jejím argumentem je matice obsahující handle ostatních radobuttonů.

Zpětnovazební kód: function mutual_exclude (off)

Set(off, 'Value',0)

PUSH BUTTON

Tlačítko, které má jenom jeden stav. Pokud tlačítko stlačím, změní se navenek jeho vzhled. Po uvolnění se tlačítko vrací do výchozí polohy.

Vlastnosti k nastavení: String- nápis na tlačítku

Tag- jméno subfunkce v m-file

TOGGLE BUTTON

Tlačítko, které má 2 stavy, tzv. přepínač.

Zpětnovazební kód: function varargout=togglebutton_1Callback(h, eventdada, handles, varargin)

```
Buton_state=get(h,'Value');
If buton_state=get(h, 'max') %stlačeno
Elseif buton_state= =get(h, 'min') %uvolněno
End
```

CHECK BOX

Zaškrtávací políčko vytváří akci je-li vybráno. Tento svůj stav uchovává ve vlastnosti value „Max-Min" (1-0). Určení aktuálního stavu se děje dotazem na value.

Zpětnovazební kód: function varargout=checkbox1_Callback(h, eventdata, handles, varargin)

```
If (get(h,'Value')= =get(h,'max'))
  %"check box je vybrán" proveď příslušnou akci
Else
  %"checkbox není vybrán" proveď příslušnou akci
End
```

EDIT TEXT

Umožňuje nám vládat či upravovat text. Vlastností string obsahuje text vložený uživatelem.

Zpětnovazební kód: function varargout=edittext1_Callback(h, evendata, handles, varargin)

```
User_string=get(h,'String')
%pokračuje callback
```

STATIC TEXT

Užívá se nejčastěji jako název k uicontrol objektům.

SLIDER

Je to numerický vstup v určitém rozsahu, čímž umožňuje uživateli pohybovat jezdcem.

Vlastnosti k nastavení: Value- obsahuje numerickou hodnotu pozice posuvníku. Posuvník lze ovládat i z jiného místa programu. Potom zpětná vazba musí obsahovat příkaz: slider_value=get(handles.slider1,'Value')

Max- definuje maximální hodnotu vlastnosti Value

Min- definuje minimální hodnotu vlastnosti Value

SliderStep- řídí jemnost změny hodnoty

FRAME

Rámy jsou obdélníkové oblasti zahrnující části okna Figure. Používají se k vizuálnímu ohraničení objektů s výjimkou Axes. Nemají zpětnovazebné volání a jsou neprůhledné.

LIST BOX

Zobrazuje seznam, ze kterého si můžeme vybírat jednu nebo více položek. Vlastnost string obsahuje seznam řetězců, které se v nabídce zobrazují. První položka má index roven 1.

Hodnoty vlastností Min a Max nám určují, zda je možné si vybrat více položek nebo jen jednu.

Je-li Max – Min > 1, pak je možný výběr více položek,

Je-li Max – Min <= 1, není možný výběr více položek.

POPUP MENU

Kliknutím na šipku se zobrazí seznam, ze kterého si můžeme volit jen jednu položku. Vlastnost String obsahuje seznam řetězců, které se v nabídce zobrazují. Vlastnost Value obsahuje index odpovídající řetězci vybrané položky. První položka v seznamu nabídek má index 1.

Zpětnovazební kód: function varargout=popupmenu1_Callback(h, evendata, handles,varargin)

```
Val=get(h,'Value');
String_list=get(h,'String');
Selected_string=string_list(val); %konverze na řetězec
```

AXES

Umožňují zobrazit a editovat grafy. Není to objekt uicontrol, ale v mnoha případech je výstupem GUI.

3.3. Vytváření grafických objektů v GUIDE

Po vysvětlení pracovního prostředí GUIDE, nyní můžeme přistoupit k samotnému návrhu grafického uživatelského rozhraní. Postup, jak vytvářet samotná GUI bych rozdělil do 5 hlavních skupin.

1. Skupina - dobré nastudování dané problematiky, možnosti řešení, zvláštní případy, které mohou nastat. Při tvorbě aplikace, musí programátor rozumět danému problému.
2. Skupina - nakreslit si na kus papíru, jak by naše GUI mělo vypadat. Dále si rozmyslet, jaké veličiny budu ovládat, počet ovládacích prvků a následné rozmístění ovládacích prvků na návrhovou plochu, aby GUI bylo přehledné.
3. Skupina - umístění jednotlivých ovládacích prvků na návrhovou plochu (Layout area), upravit jejich velikost, rozmístění a samozřejmě zarovnání. Jestliže máme více stejných ovládacích prvků, je dobré, když mají stejné rozměry.
4. Skupina - spuštění GUI, kde se nám automaticky vygenerují 2 soubory s příponou *.fig a *.m. Dále je potřebné zkontrolovat vzhled. Pokud nebude dostačující, v GUIDE si můžeme upravit rozmístění. V property inspectoru můžeme libovolně nastavit různé vlastnosti daných objektů.
5. Skupina - Do vygenerovaného m-file souboru musíme doplnit zpětné vazby (callback), abychom dané objekty „oživily“.

4. Tvorba GUI aplikace

V následující kapitole a jejích podkapitolách se pokusím nastínit, jak se postupuje při tvorbě GUI aplikace. Návrh GUI aplikace jsem rozdělil na dvě části: teoretickou a praktickou. V teoretické části shrnuji nezbytné znalosti z teorie kmitavého pohybu, v praktické pak popisuji postup při vytváření GUI aplikace.

V praktické části si musíme dobře rozmyslet rozmístění jednotlivých objektů, aby byla zajištěna přehlednost aplikace a její jednoduchost na obsluhu. Nesmíme však zapomenout na základní požadavky pro běh celého programu: jednoduchost, provázanost a komplexnost.

4.1. Teorie kmitání a optiky

Pro prezentaci GUI aplikace, jsem zvolil úlohu z teorie kmitání a optiky. Toto téma jsem zvolil především proto, že není až tak matematicky náročné a nabízí dobré grafické znázornění.

4.1.1. Rovnice kmitavého pohybu hmotného bodu

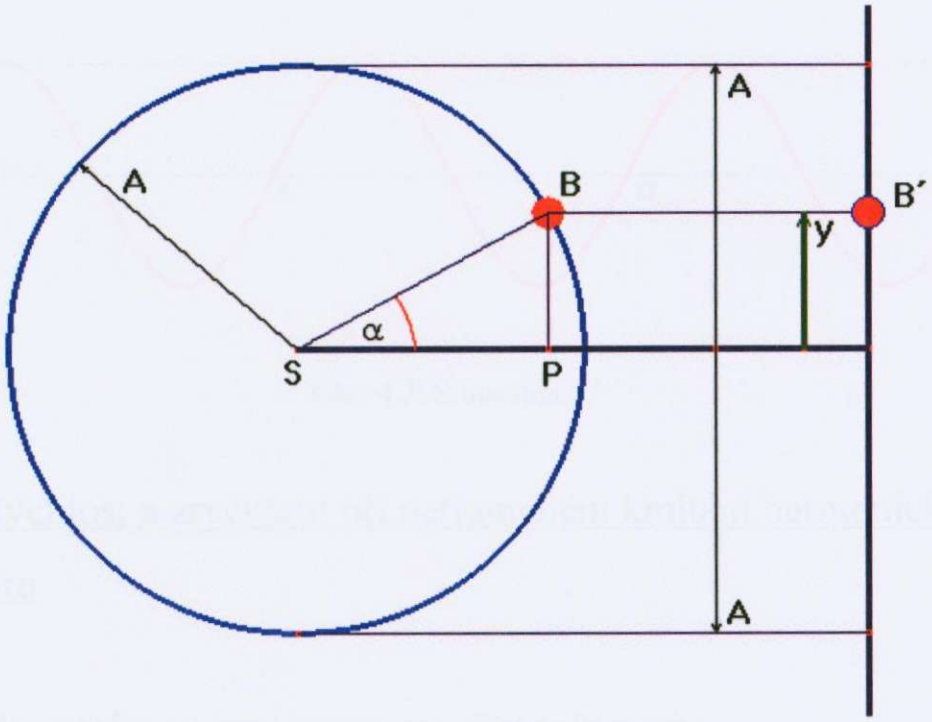
Rovnici kmitavého pohybu hmotného bodu můžeme odvodit několika způsoby. První způsob využívá srovnání kmitavého pohybu a rovnoměrného pohybu hmotného bodu po kružnici, druhý je teoretičtější a využívá matematického aparátu. Zde popíši odvození kmitavého pohybu prvním způsobem.

Odvození na základě porovnání kmitavého pohybu a rovnoměrného pohybu hmotného bodu po kružnici

Představme si situaci znázorněnou na obrázku 4.1. Hmotný bod B, vykonává rovnoměrný harmonický pohyb po kružnici o poloměru A s úhlovou rychlostí w .

Předpokládejme, že v počátečním okamžiku (v čase t_0) svírala spojnice středu S a bodu B s vodorovnou osou úhel φ_0 . Za čas t urazí hmotný bod při úhlové rychlosti ω

úhel ωt. S vodorovným směrem tedy spojnice SB svírá úhel $\alpha = \omega \cdot t + \varphi_0$ (při uvážení hodnoty úhlu do 360°, jinak je potřeba převést do intervalu od 0 do 360 ° odečtením vhodného poctu násobku 360°).



Obr. 4.1: Znázornění harmonického pohybu.

Promítneme bod *B* do bodu *B'* (bod *B'* leží na přímce rovnoběžné s osou *y* a zároveň na rovnoběžce s osou *x* vedenou bodem *B*). Pokud se bod *B* pohybuje po kružnici, tak nám bod *B'* vykonává harmonický netlumený kmitavý pohyb. Okamžitá výchylka tohoto bodu z rovnovážné polohy je rovna velikosti úsečky PB, jejíž velikost můžeme určit z pravoúhlého trojúhelníka SPB takto:

$$\frac{|BP|}{|SB|} = \sin\alpha$$

$$\frac{\mathbf{y}}{A} = \sin\alpha \Rightarrow \mathbf{y} = A \cdot \sin\alpha \tag{1}$$

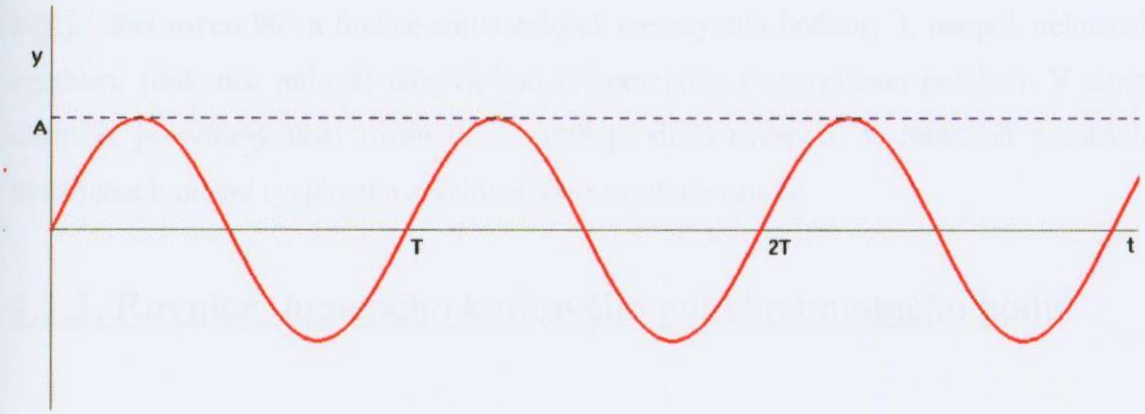
Jelikož jsme odvodili, že hodnota α je $\omega t + \varphi_0$, dostáváme výsledný vztah popisující harmonický netlumený kmitavý pohyb

$$\mathbf{y} = A \cdot \sin(\boldsymbol{\omega} t + \varphi_0) \tag{2}$$

Veličinu ω nazýváme úhlová frekvence, můžeme ji vypočítat ze vztahu:

$$\boldsymbol{\omega} = 2\pi f$$

veličinu φ_0 j nazýváme počáteční fáze a výraz $\omega t + \varphi_0$ v závorce fáze kmitavého pohybu. Z výše uvedeného plyne i grafické znázornění závislosti výchylky hmotného bodu na čase. Je jím známá sinusoida (Obr 4.2).



Obr. 4.2: Sinusoida.

4.1.2. Rychlost a zrychlení při netlumeném kmitání harmonického oscilátoru

Už víme, že výchylku y v čase t můžeme vypočítat podle rovnice

$$\mathbf{y} = A \cdot \sin(\boldsymbol{\omega} t + \varphi_0) \tag{3}$$

Výchylka nám popisuje polohu tělesa v daném čase. Z mechaniky víme, že rychlost daného bodu můžeme počítat jako derivaci polohy podle času, tedy

$$v = \frac{dy}{dt} = A \cdot \omega \cdot \cos(\omega \cdot t + \varphi_0) \tag{4}$$

Obdobně víme, že zrychlení je možno počítat jako derivace rychlosti podle času, tedy

$$a = \frac{dv}{dt} = -A \cdot \omega^2 \cdot \sin(\omega \cdot t + \varphi_0) \tag{5}$$

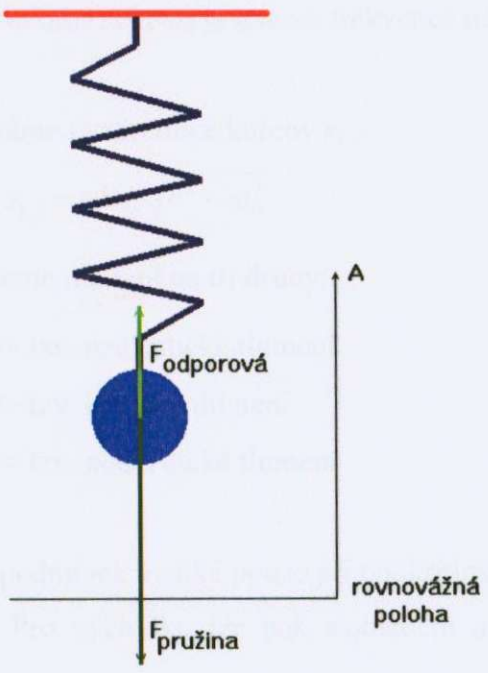
Je dobré si uvědomit, že rychlost harmonicky kmitajícího bodu je největší v rovnovážné poloze, kdy je úhel 0° a funkce cos nabývá tedy své maximální hodnoty 1.

Nejmenší rychlosti dosahuje bod v okamžiku, kdy se nalézá v krajním bode. V tento okamžik je úhel roven 90° a funkce cos nabývá hodnoty 0. Rychlost je tedy také nulová.

Obdobně je to se zrychlením. Největší hodnotu má zrychlení v krajních bodech, kdy je úhel roven 90° a funkce sinus nabývá své největší hodnoty 1, naopak nejmenší zrychlení (dokonce nulové) nabývá bod při průchodu rovnovážnou polohou. V tento okamžik je svíraný úhel roven 0° a tudíž je sinus roven 0. V ostatních polohách využijeme k určení rychlosti a zrychlení výše uvedené vzorce.

4.1.3. Rovnice tlumeného kmitového pohybu hmotného bodu

Tlumený mechanický oscilátor odevzdává část své energie svému okolí a snižuje tak svou amplitudu při kmitaní. Situaci nám znázorňuje následující obrázek 4.3.



Obr. 4.3: Příklad tlumeného kmitání.

Při pohybu je odporová síla velmi často úměrná rychlosti pohybu a působí v opačném směru než rychlost. Pro tento případ lze pro velikost odporové síly psát.

$$\mathbf{F}_{odporová} = -r \cdot \mathbf{v} \qquad (6)$$

Dále platí jako v předešlé kapitole,

$$\mathbf{F}_{pružiny} = -k \cdot \mathbf{y} \tag{7}$$

Platí, že

$$\mathbf{F}_{celková} = \mathbf{F}_{pružiny} + \mathbf{F}_{odporová} \tag{8}$$

a z toho vyplývá rovnice

$$\mathbf{F}_{celková} = -k \cdot \mathbf{y} - r \cdot \mathbf{v} \tag{9}$$

Provedeme obdobné úpravy jako v předchozí kapitole a dostaneme tak diferenciální rovnici, kterou budeme opět řešit metodou charakteristické rovnice.

$$\begin{aligned} m \cdot \frac{\mathbf{d}^2\mathbf{y}}{dt^2} + r \cdot \frac{\mathbf{dy}}{dt} + k \cdot \mathbf{y} = 0 \\ \frac{\mathbf{d}^2\mathbf{y}}{dt^2} + \frac{r}{m} \cdot \frac{\mathbf{dy}}{dt} + \frac{k}{m} \cdot \mathbf{y} = 0 \end{aligned} \tag{10}$$

Konečný diferenciální tvar rovnice vypadá takto:

$$\frac{\mathbf{d}^2\mathbf{y}}{dt^2} + 2b\frac{\mathbf{dy}}{dt} + \omega_0^2 \cdot \mathbf{y} = 0 \tag{11}$$

Konstanta b je tzv. koeficient útlumu a ω_0 je úhlová frekvence netlumených kmitů.

Celou rovnici řešíme a získáme kvadratické kořeny $s_{1,2}$,

$$s_{1,2} = -b \pm \sqrt{b^2 - \omega_0^2} \tag{12}$$

podle hodnoty $\sqrt{b^2 - \omega_0^2}$ rozdělujeme tlumení na tři druhy:

- $\mathbf{b^2 - \omega^2 > 0}$ – $b > \omega^2$ => tzv. nadkritické tlumení
- $\mathbf{b^2 - \omega^2 = 0}$ – $b = \omega^2$ => tzv. kritické tlumení
- $\mathbf{b^2 - \omega^2 < 0}$ – $b < \omega^2$ => tzv. podkritické tlumení

Tlumený pohyb za těchto podmínek vzniká pouze při podkritickém tlumení, kdy je tlumící síla dostatečně slabá. Pro výchylku lze pak s ohledem na obecné řešení diferenciálních rovnic psát:

$$\mathbf{y} = e^{-bt} \cdot (C_1 \cdot \sin \boldsymbol{\omega} t + C_2 \cdot \cos \boldsymbol{\omega} t) \tag{13}$$

Stejně jako v případě netlumených kmitů i u této rovnice můžeme dojít k obecnému řešení využitím počátečných podmínek a získat tak konečný tvar rovnice:

$$\mathbf{y} = A \cdot e^{-bt} \cdot \sin(\boldsymbol{\omega} t + \varphi) \tag{14}$$

4.1.4. Rovnice složených kmitů

Metod pro skládání dvou a více kmitavých pohybů jsou. Patří mezi ně metoda matematická (početní), vektorové skládání. Stejně tak můžeme skládat pohyby (kmity) stejného směru nebo např. směru kolmého (u takového skládání můžou být výsledným obrazcem Lissajousovy obrazce).

Pro využití matematického aparátu v MATLABu jsem zvolil vyjádření kmitů za pomoci matematické metody. V úvahu však může přijít i vektorová metoda, oba postupy vedou k totožným výsledkům.

Nejjednodušším případem je skládání pohybů po přímce, kdy směr kmitů je vždy stejný. Výsledný pohyb, však již není sinusovitý, ale je vždy harmonický.

Speciálně užíváme početní metodu v případech, kdy mají kmity podobně velké frekvence (například 400 Hz a 410 Hz) a kmity mají stejný směr, fázi a amplitudu. Nechť mají jednotlivé kmity rovnice:

$$\begin{aligned} \psi_1 &= \psi_0 \sin(\omega_1 t + \varphi) \\ \psi_2 &= \psi_0 \sin(\omega_2 t + \varphi) \end{aligned} \tag{15}$$

Výsledný kmit je dán dle zákona superpozice součtem okamžitých výchylek ψ_1 a ψ_2:

$$\psi = \psi_1 + \psi_2 = \psi_0 \sin(\omega_1 t + \varphi) + \psi_0 \sin(\omega_2 t + \varphi) \tag{16}$$

Nyní vytkneme hodnotu amplitudy:

$$\psi = \psi_0 \cdot [\ \sin(\omega_1 t + \varphi) + \sin(\omega_2 t + \varphi)\ ] \tag{17}$$

Užitím obecného matematického vzorce:

$$\sin\alpha + \sin\beta = 2 \cdot \sin\frac{a+b}{2} \cdot \cos\frac{a+b}{2} \tag{18}$$

Tím dostane výsledný vzorec pro skládání kmitů blízkých frekvencí:

$$\psi = 2 \cdot \psi_0 \cdot \cos\left(\frac{\omega_1 - \omega_2}{2} \cdot t\right) \cdot \sin\left(\frac{\omega_1 + \omega_2}{2} \cdot t + \varphi\right) \tag{19}$$

4.1.5. Lissajousovy obrazce

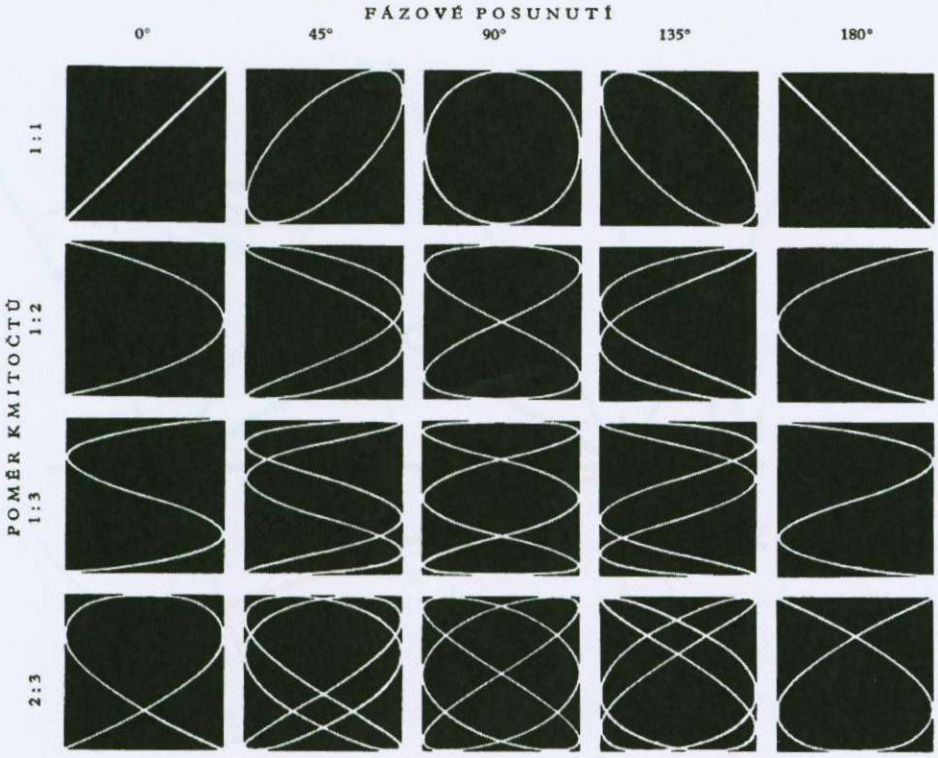
Početní metodu lze využít i k zvláštním případům složeného kmitání a to ke kmitům kolmých. Poměry dob kmitů musí být v poměru celých čísel a pak platí:

$$n_1 \cdot T_1 = n_2 \cdot T_2 \tag{20}$$

Počáteční fáze a amplitudy mohou být různé.

Výsledkem tohoto skládání je rovinný pohyb. Křivka je uzavřená a na vzniklé obrazce nazýváme Lissajousovy obrazce (Obr 4.4). Pro několik počátečních fází můžeme vidět na obrázku jednotlivé obrazce.

Pokud nebude poměr frekvencí z oboru celých čísel, výsledný obrazec nebude uzavřená křivka a výsledných kmit nebude harmonickým pohybem.

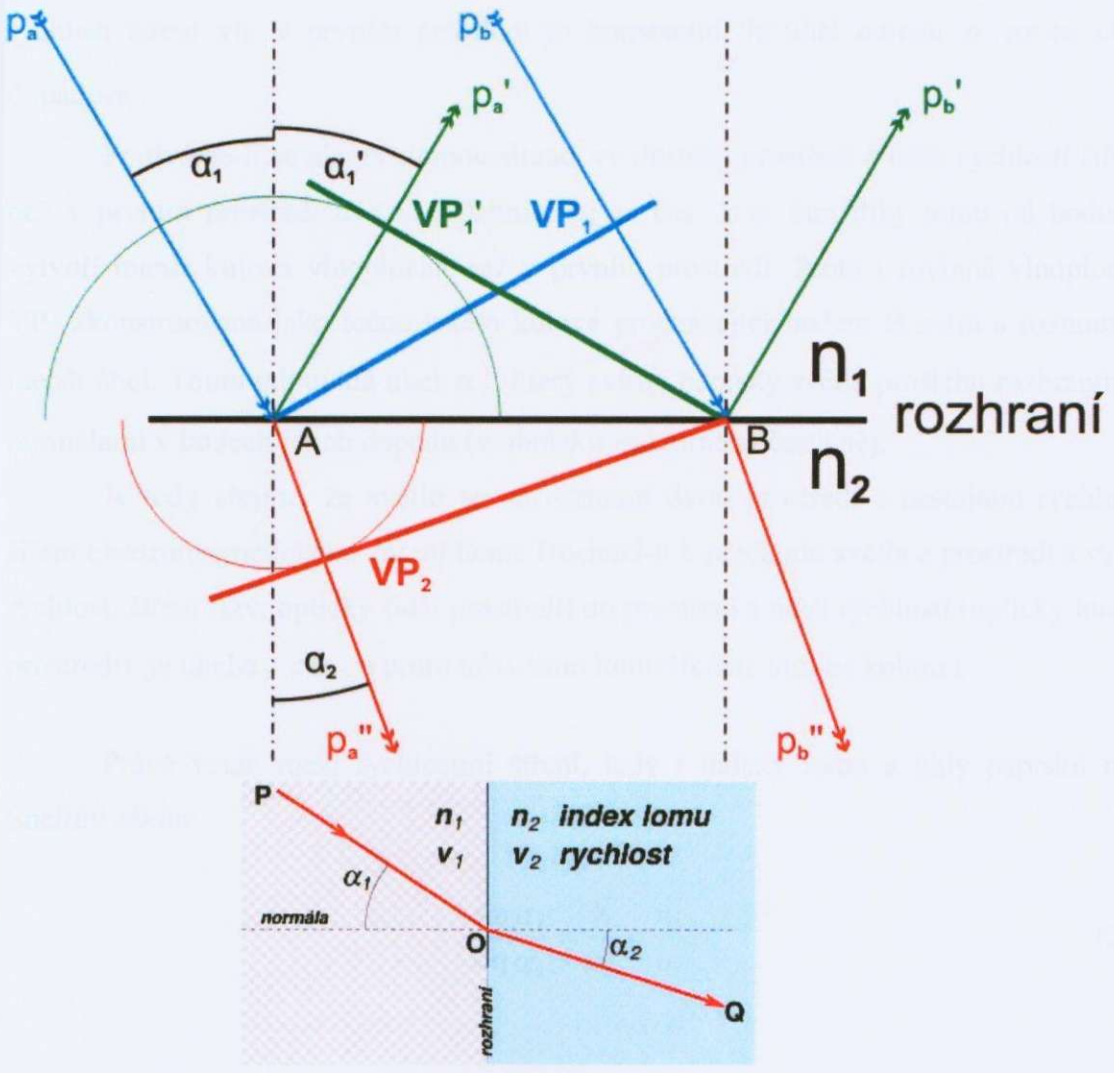


Obr. 4.4: Příklad Lissajousových obrazců.

4.1.6. Snellův zákon

Snellův zákon popisuje chování elektromagnetického záření na rovinném rozhraní dvou prostředí s odlišnými vlastnostmi šíření. Je to jeden ze základních zákonů geometrické optiky a úzce souvisí s rychlostí šíření světla (obecněji elektromagnetického záření). Rychlost šíření elektromagnetického záření ve vakuu označovaná c je 299 792 458 metrů za sekundu a je to nejvyšší možná rychlost šíření informací. V jakékoliv látce je rychlost šíření elektromagnetického záření vždy menší než ve vakuu. Poměr rychlosti ve vakuu a rychlosti v daném prostředí v nazýváme index lomu a značíme n.

$$n = \frac{c}{v} \tag{21}$$



Obr. 4.5: Snellův zákon.

Situaci, kdy světlo dopadá na rovinné rozhraní dvou prostředí, si můžeme jednoduchým způsobem geometricky zkonstruovat. Nechť prostředí, u nichž se světlo šíří mají indexy lomu n_1 a n_2 a na jejich rozhraní paprsky dopadají šikmo pod úhlem sevřeným mezi paprskem a normálou rozhraní v bodě dopadu α_1. Nechť paprsek dopadající paprsek (na obrázku znázorněný modře) p_a dopadá na rozhraní v bodě A a paprsek p_b v bodě B. Z obrázku je zřejmé, že rovinná vlnoplocha VP_1 tvořená dopadajícími paprsky do bodu B dorazí o čas Δt později než dorazila do bodu A.

Vzhledem k tomu, že každý bod takového rozhraní můžeme považovat za elementární zdroj záření, v době kdy dorazila vlnoplocha dopadajícího záření do bodu B, vytvořila se v bodě A v prvním prostředí již kulová vlnoplocha, jejíž poloměr vyplývá z rychlosti šíření v tomto prostředí a času Δt. Zkonstruujeme-li tečnu k takové kulové vlnoploše procházející bodem B, získáme rovinnou vlnoplochu VP_1' odpovídající odraženým paprskům (v obrázku znázorněny zeleně) p_a' a p_b'. Protože rychlost šíření vln v prvním prostředí je konstantní, je úhel odrazu α'_1 roven úhlu dopadu α_1.

Podíváme-li se ale na stejnou situaci ve druhém prostředí s nižší rychlostí šíření než v prvním prostředí $n1<n2$, zjistíme, že za čas Δt se tam díky tomu od bodu A vytvoří menší kulová vlnoplocha než v prvním prostředí. Proto i rovinná vlnoplocha VP_2 zkonstruovaná jako tečna k této kulové procházející bodem B svírá s rozhraním menší úhel. Tomu odpovídá úhel α_2, který svírají paprsky světla prošlého rozhraním s normálami v bodech jejich dopadu (v obrázku znázorněny červeně).

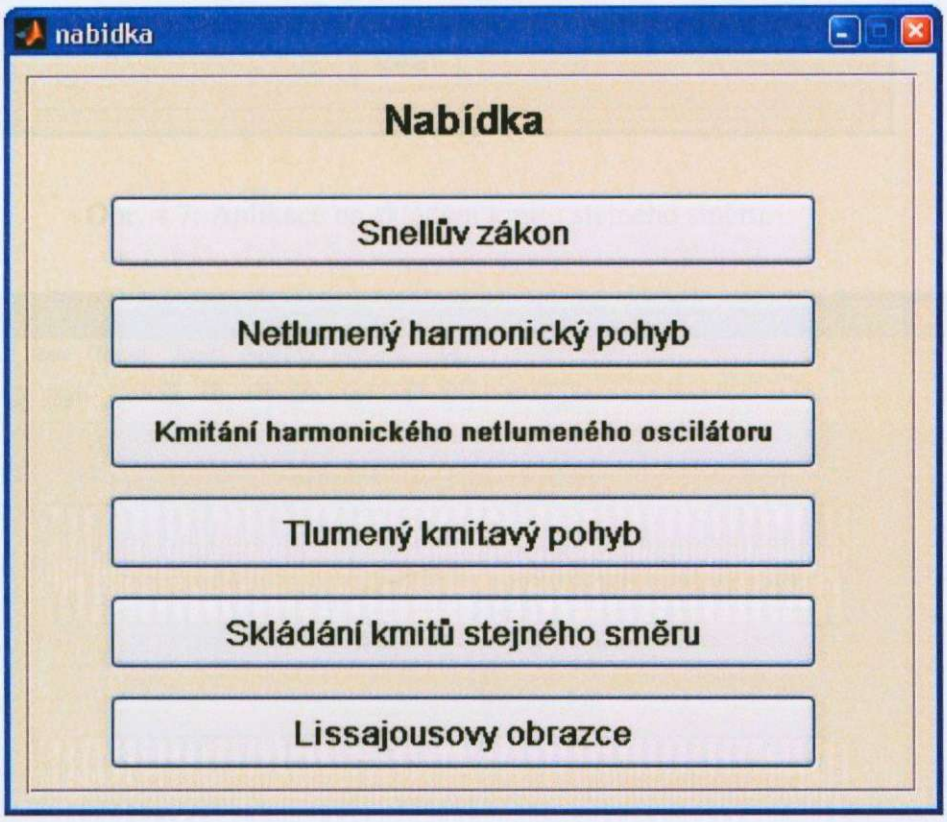
Je tedy zřejmé, že světlo se na rozhraní dvou prostředí s nestejnou rychlostí šíření elektromagnetického záření láme. Dochází-li k přechodu světla z prostředí s vyšší rychlostí šíření (tzv. opticky řidší prostředí) do prostředí s nižší rychlostí (opticky hustší prostředí), je úhel $\alpha_2 < \alpha_1$, a proto takovému lomu říkáme lom ke kolmici.

Právě vztah mezi rychlostmi šíření, tedy i indexy lomu a úhly paprsků řeší Snellův zákon:

$$\frac{\sin\alpha_1}{\sin\alpha_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1} \tag{22}$$

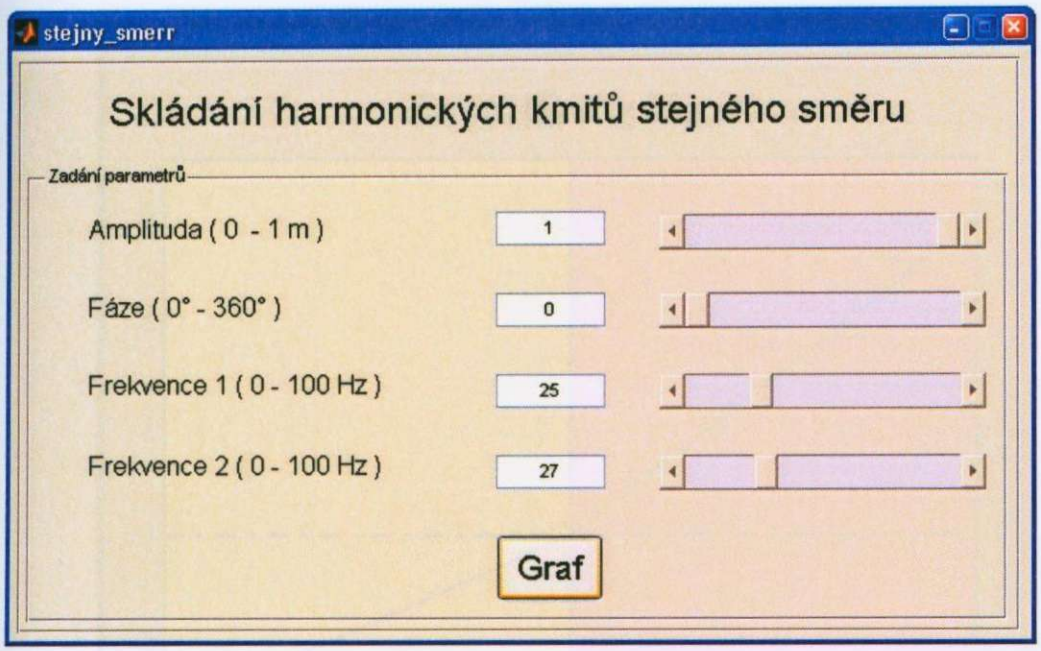
4.2. Úprava a vzhled GUI aplikace

Vytvořený program jsem navrhoval tak, aby namodelovaná problematika byla přístupná z jednoho prostředí. Jednotlivé položky jsou zobrazeny na ploše aplikace. Kliknutím myši na libovolnou položku se otevře okno s naprogramovaným fyzikálním jevem. Pokud si budeme chtít vybrat jinou položku, nemusíme pozavírat jednotlivá okna, protože se po přepnutí na jinou položku všechna okna pozavírají a otevře se námi vybraná aplikace. Každý program s naprogramovaným fyzikálním jevem pracuje na bázi jednoho tlačítka. To znamená zaručení maximální přehlednosti aplikace.

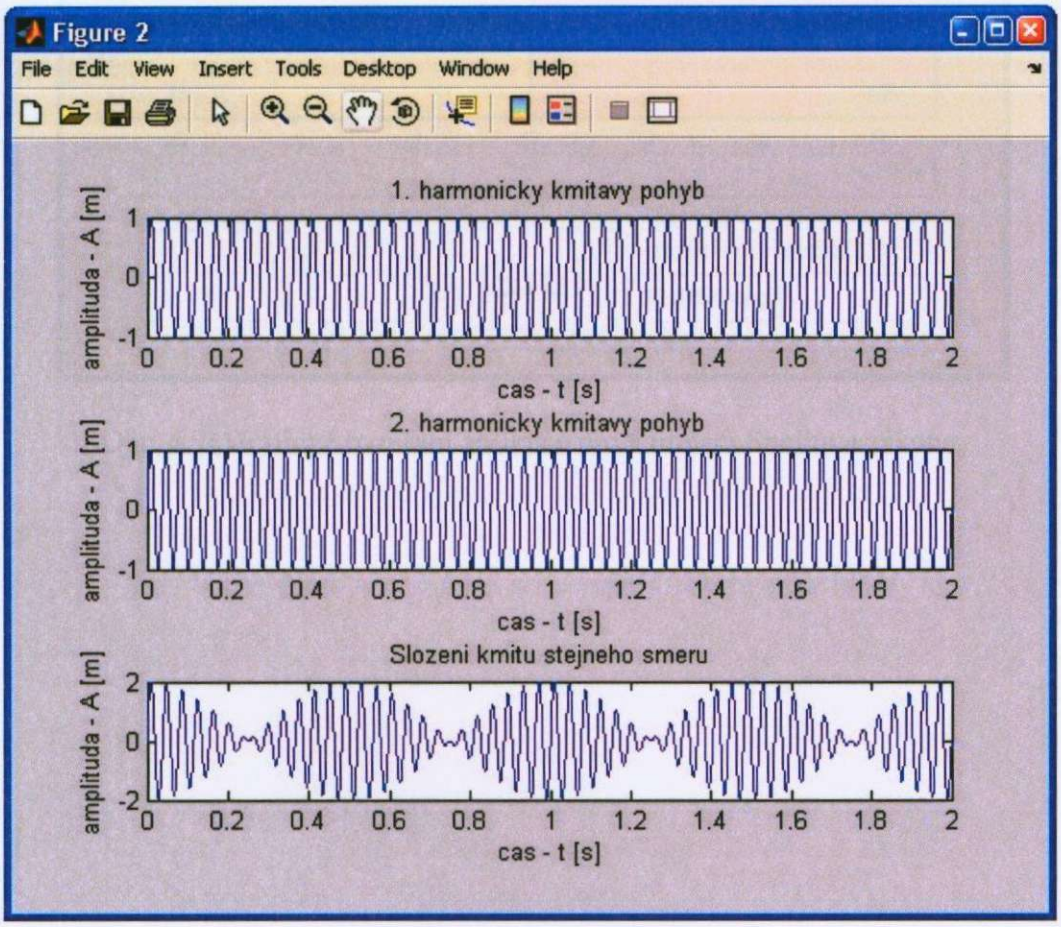


Obr. 4.6: Návrh aplikace.

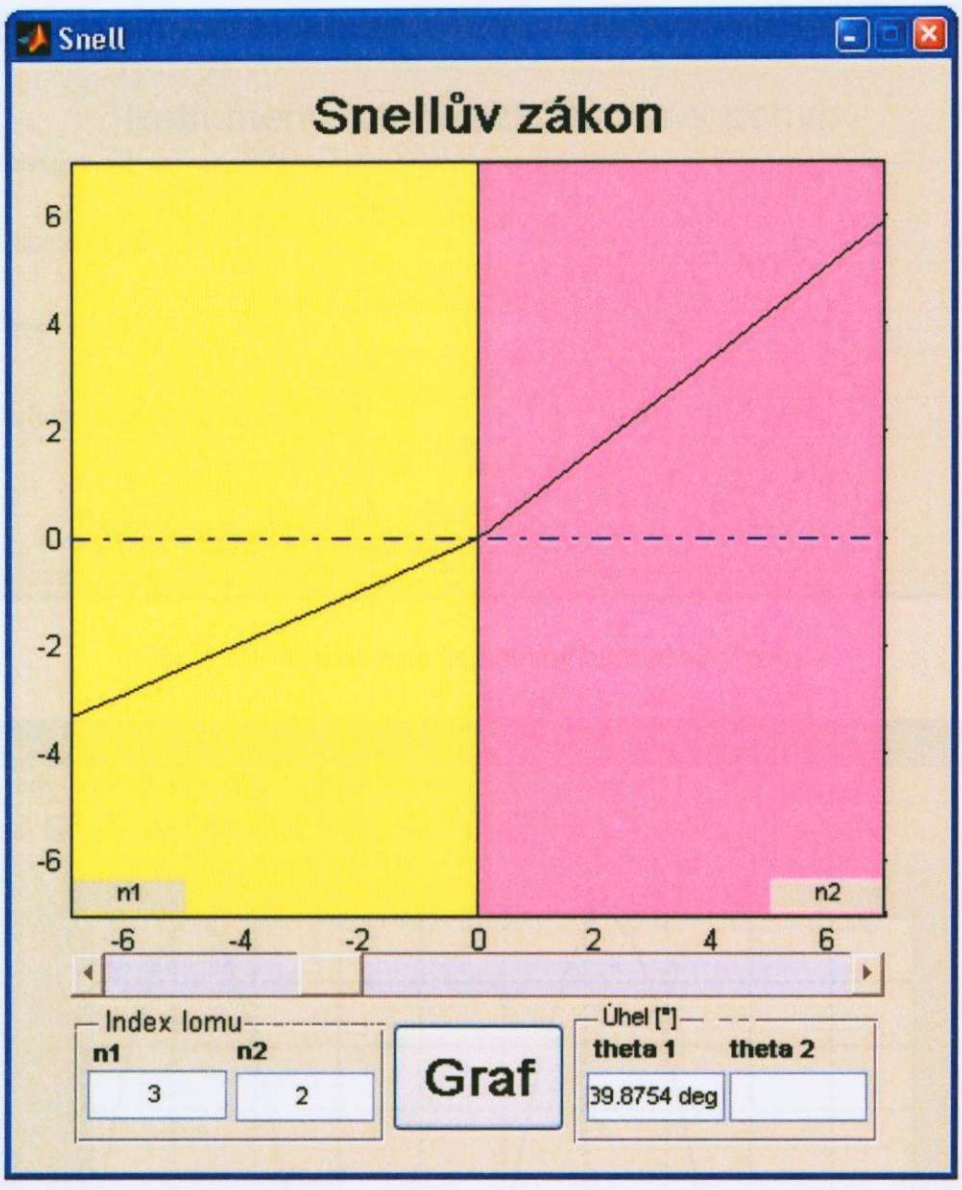
Celá aplikace by potřebovala více času z hlediska vytvoření nápovědy a ošetření jednotlivých chyb. V nápovědě by určitě neměla chybět vysvětlená teorie k řešené problematice, dále návod pro práci v aplikaci. Ošetření chyb by mělo být na vyšší úrovni. Pokud by uživatel zadal nesprávnou hodnotu, aplikace by ho měla upozornit, že v průběhu chodu programu vznikla chyba. Uživateli by mělo být naprosto zřejmé, kde chyba vznikla, aby se následně této chybě vyvaroval. Toto bych chtěl řešit v další době studia.



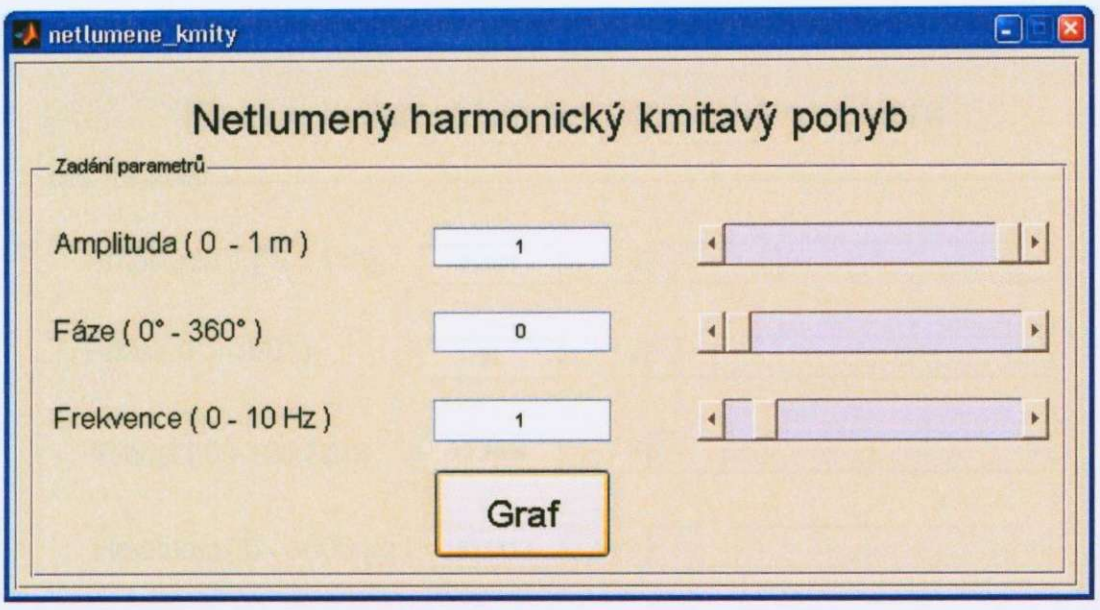
Obr. 4.7: Aplikace na skládání kmitů stejného směru.



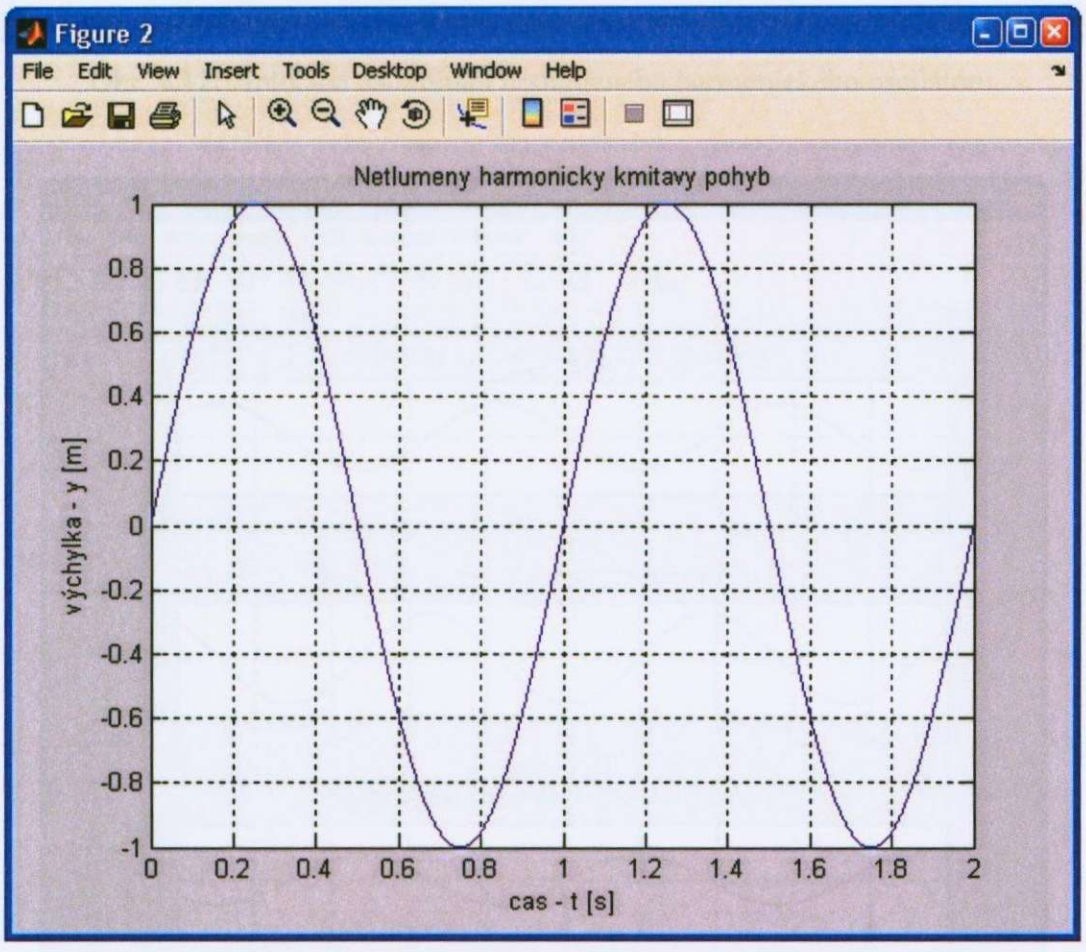
Obr. 4.8: Graf (skládání kmitů stejného směru).



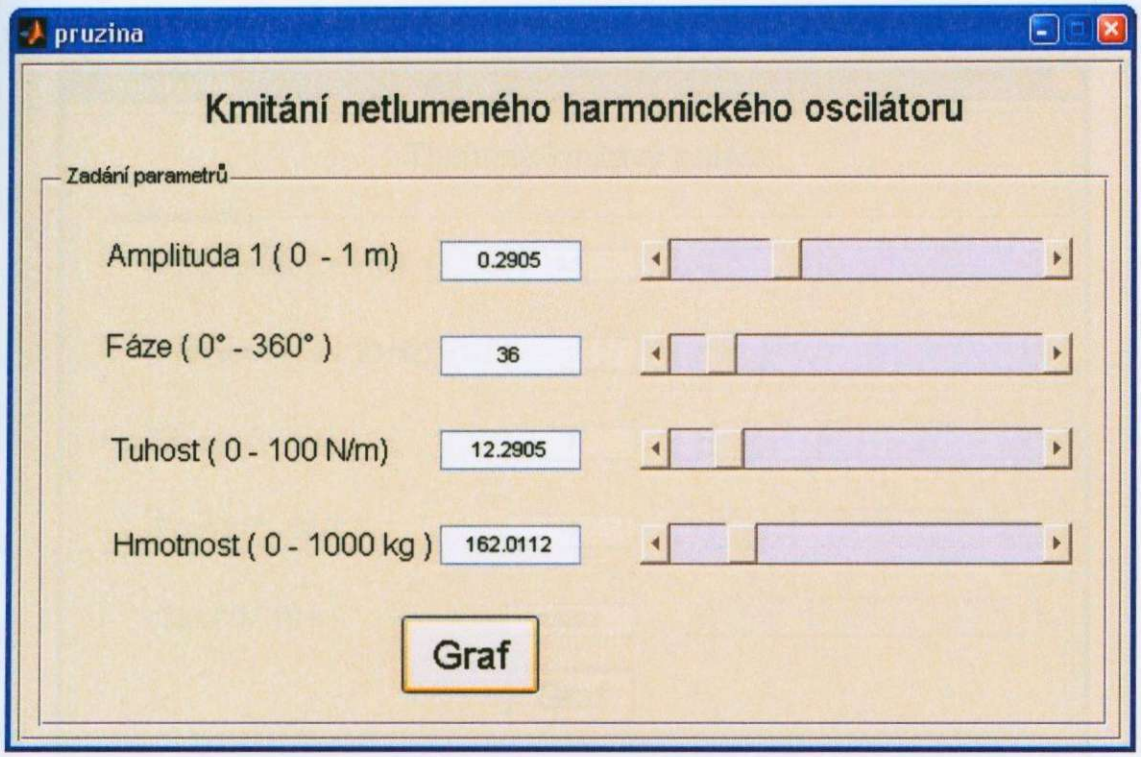
Obr. 4.9: Grafické rozhraní aplikace pro simulaci Snellůva zákona.



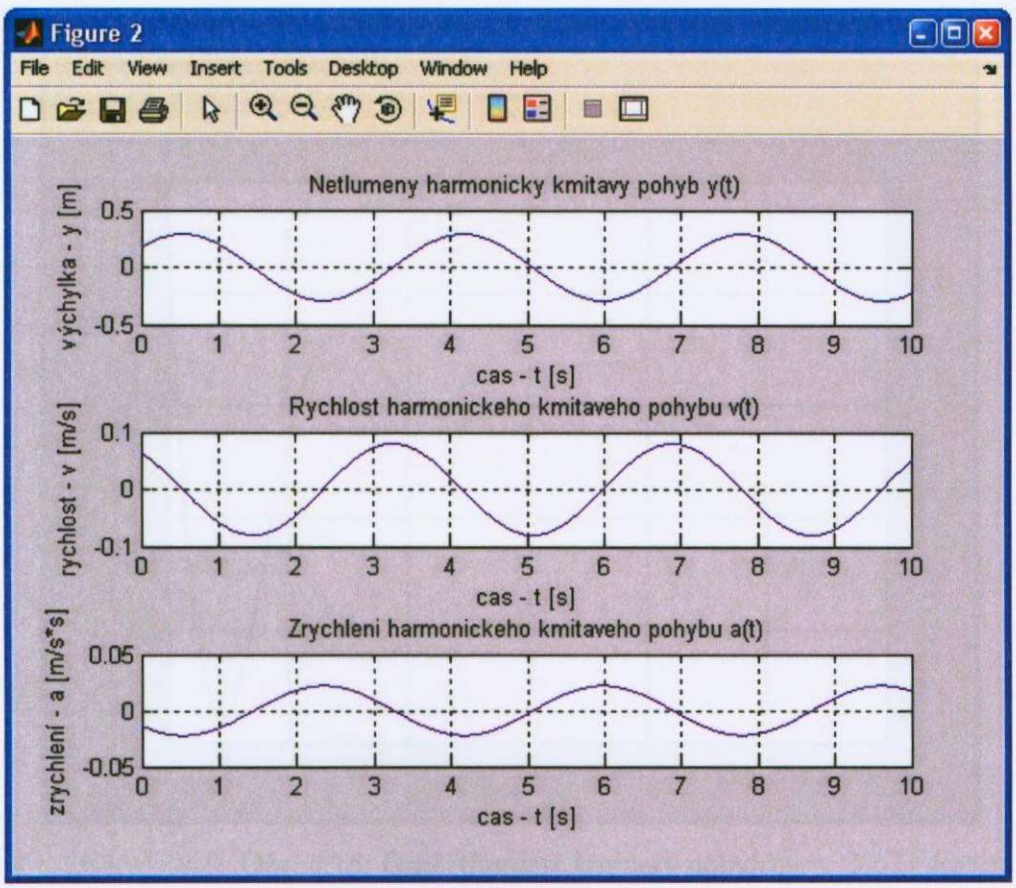
Obr. 4.10: Aplikace na netlumený harmonický pohyb.



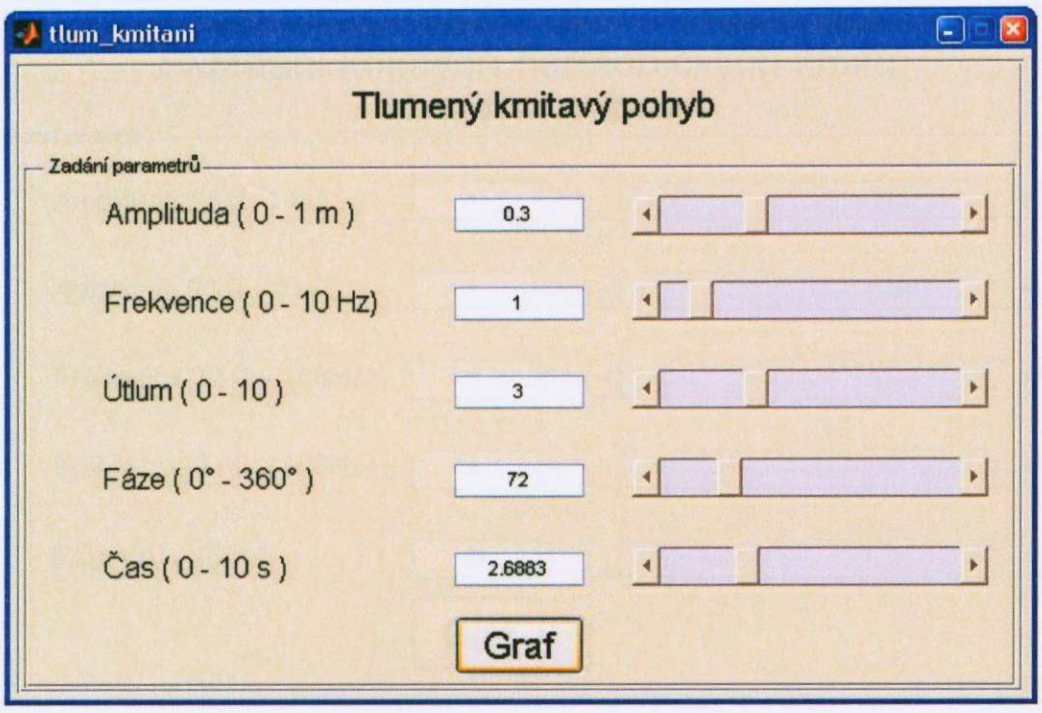
Obr. 4.11: Netlumený harmonický pohyb – závislost výchylky na čase.



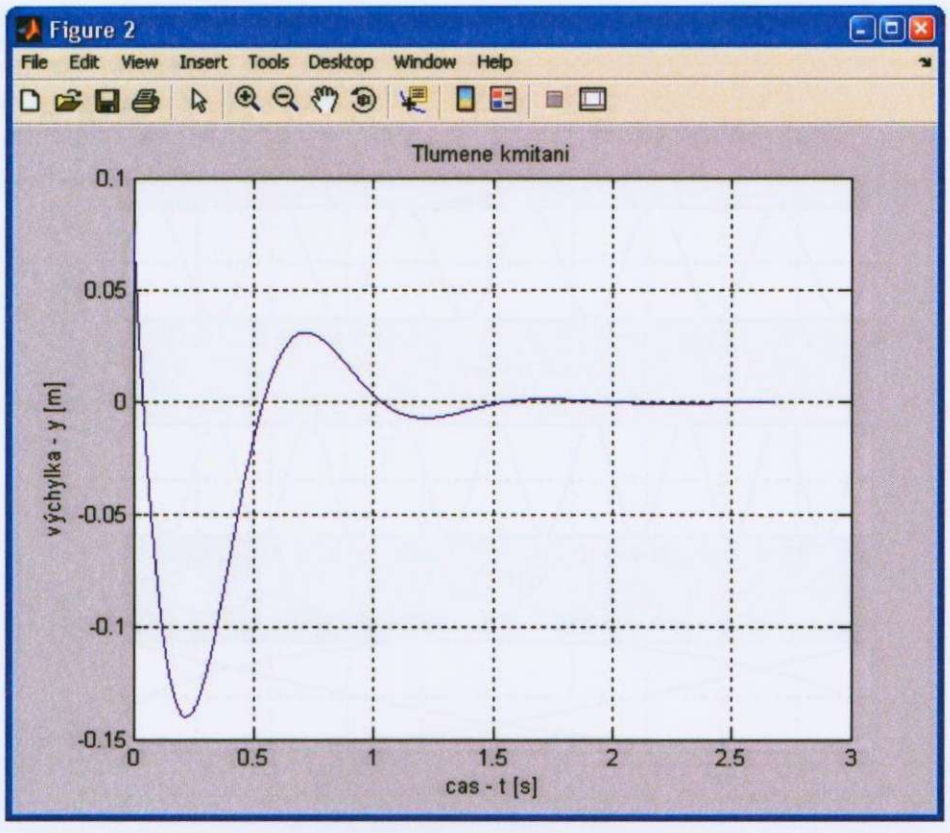
Obr. 4.12: Aplikace na kmitání netlumeného harmonického oscilátoru.



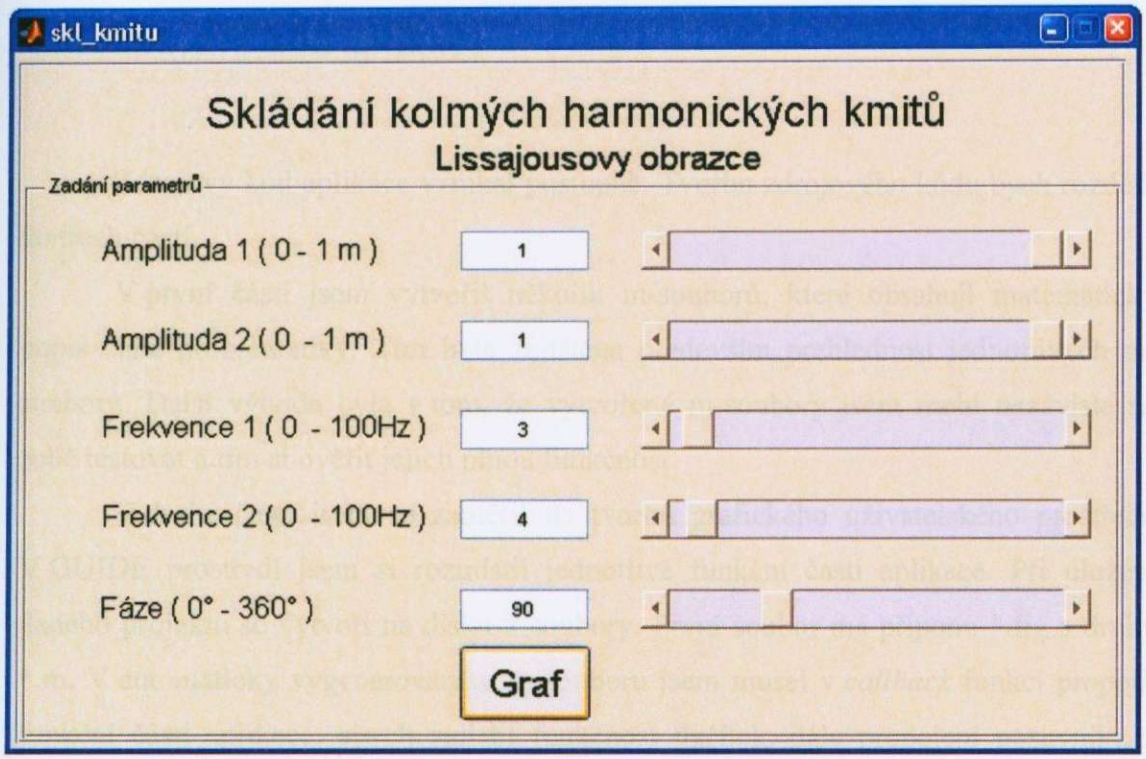
Obr. 4.13: Graf (kmitání netlumeného harmonického oscilátoru).



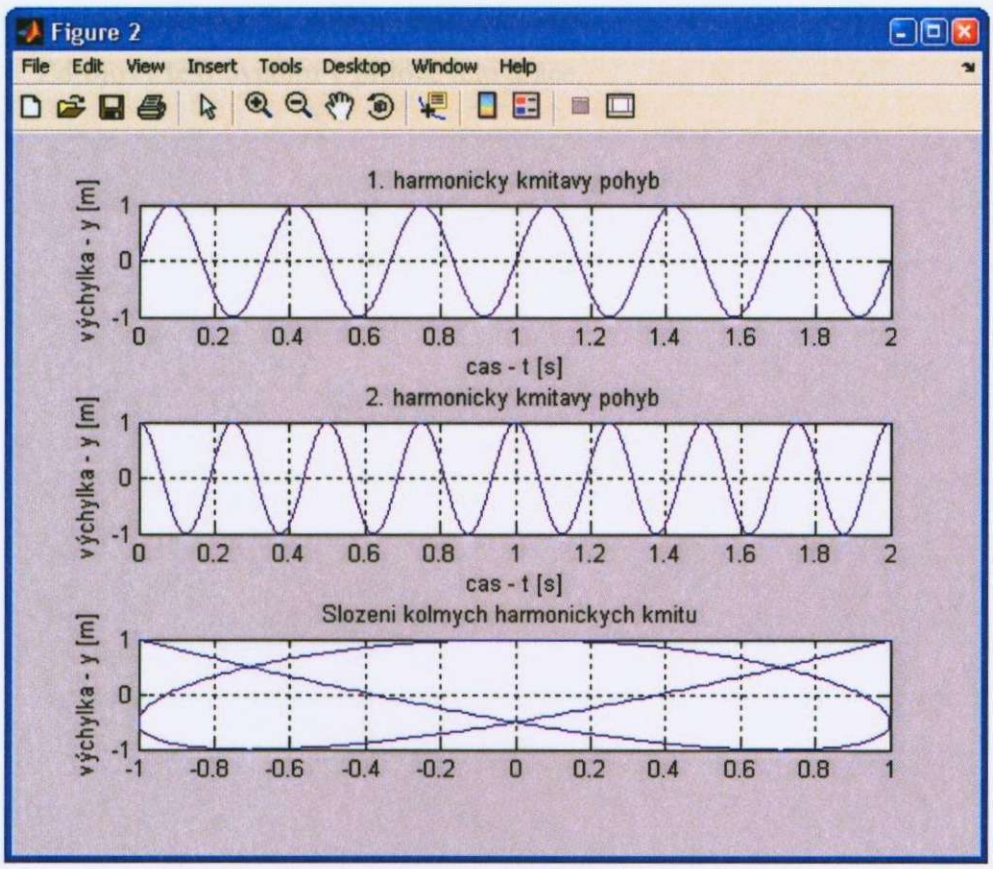
Obr. 4.14: Aplikace na tlumený kmitavý pohyb.



Obr. 4.15: Graf (tlumený kmitavý pohyb).



Obr. 4.16: Aplikace na Lissajousovy obrazce.



Obr. 4.17: Graf (Lissajousovy obrazce).

4.3. Zdrojový kód aplikace

Zdrojový kód aplikace vznikal postupně. Tvorbu zdrojového kódu bych rozdělil do třech částí.

V první části jsem vytvořil několik m-souborů, které obsahují matematický popis dané problematiky. Tím byla zajištěna především přehlednost jednotlivých m-souborů. Další výhoda byla v tom, že vytvořené m-soubory jsem mohl nezávisle na sobě testovat a tím si ověřit jejich plnou funkčnost.

V druhé časti jsem se zaměřil na tvorbu grafického uživatelského prostředí. V GUIDE prostředí jsem si rozmístil jednotlivé funkční časti aplikace. Při uložení daného projektu se vytvoří na disku 2 soubory. První soubor má příponu *.fig a druhý *.m. V automaticky vygenerovaném m-souboru jsem musel v *callback* funkcí propojit funkční části aplikace, abych zajistil funkčnost tlačítek, dále propojení posuvníku a textového pole, pro zadávání číselných hodnot.

V třetí části jsem se věnoval propojení již vytvořených m-souborů tak, abych vytvořil co nejpřehlednější aplikaci (viz. Obr. 4.6).

Zdrojový text uvádím v příloze této práce.

ZÁVĚR

Hlavním cílem mé bakalářské práce byla tvorba manuálu pro vytvoření aplikace v MATLABu. V práci jsem se snažil klást důraz na správné zásady návrhu aplikace a výpočetní části tak, aby byla zajištěna maximální efektivita pro uživatele.

Při psaní a návrhu aplikace jsem se potýkal s jedním velkým problém. Pro tvorbu a návrh aplikací v GUIDE prostředí neexistuje žádný odborný text. Problém jsem řešil tím, že jsem hledal odkazy na internetu a využíval dobře zpracovanou nápovědu MATLABU.

Po počátečních problémech při práci v GUI, které se však zcela určitě vyskytují také při první práci v jiných grafických aplikacích, se mi, dle mého názoru, podařilo připravit text a aplikace, které názorně demonstrují postup práce v GUI.

Při psaní bakalářské práce jsem se přesvědčil, že fyzikální jevy mohou být za pomoci MATLABu a jeho grafického uživatelského rozhraní studovány pohodlně a rychle.

Seznam použité literatury

[1] Dušek, F.: *MATLAB A SIMULINK – úvod do používání*, Univerzita Pardubice, Pardubice, 2002, ISBN 80-7194-273-1

[2] Zaplatílek, K., Doňar, B.: *MATLAB - pro začátečníky*, Praha: nakladatelství BEN, 2007, ISBN 80-7300-175-6

[3] Zaplatílek, K., Doňar, B.: *MATLAB - začínáme se signály*, Praha: nakladatelství BEN, 2006, 80-7300-200-0

[4] Zaplatílek, K., Doňar, B.: *MATLAB - tvorba uživatelských aplikací*, Praha: nakladatelství BEN, 2005, 80-7300-133-0

[5] Bartoš, P.: - *Učební texty k přednáškám z předmětu Kmitání, vlnění a optika*

[6] MATLAB- manuály k jednotlivým toolboxům, MATLAB – help

[7] Karban, P.: *MATLAB a SIMULINK - Výpočty a simulace v programech*, Praha: nakladatelství Computer Press, 2006, ISBN 80-251-1448-3

Seznam literatury dostupné na internetu

[8] Mathworks: – adresa: http://www.humusoft.com/

[9] Humusoft – adresa: http://www.humusoft.cz/matlab/matlab.htm

[10] MATLAB – adresa: http://en.wikipedia.org/wiki/MATLAB

[11] MATLAB Summary and Tutorial

adresa: http://www.math.ufl.edu/help/matlab-tutorial/

[12] Matlab, laboratoř nejen pro matematiky

Adresa: http://cmp.felk.cvut.cz/~pisa/Public/ST_matlab.html

PŘÍLOHY

Zdrojový text aplikace - nabídka:

```
function varargout = Nabidka(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end

elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end

function pushbutton1_Callback(hObject, eventdata, handles)
skl_kmitu
close(pruzina)
close(snell)
close (figure (2))
close (tlum_kmitani)
close (netlumene_kmity)
close (stejny_smerr)

function pushbutton2_Callback(hObject, eventdata, handles)
snell
close(pruzina)
close(netlumene_kmity)
close (figure (2))
close (tlum_kmitani)
close (netlumene_kmity)
close (stejny_smerr)

function pushbutton3_Callback(hObject, eventdata, handles)
pruzina
close(skl_kmitu)
```

```
close(snell)
close (figure (2))
close (tlum_kmitani)
close (netlumene_kmity)
close (stejny_smerr)

function pushbutton4_Callback(hObject, eventdata, handles)
stejny_smerr
close(skl_kmitu)
close(snell)
close (figure (2))
close (tlum_kmitani)
close (netlumene_kmity)
close (pruzina)

function pushbutton5_Callback(hObject, eventdata, handles)
tlum_kmitani
close(skl_kmitu)
close(snell)
close (figure (2))
close (stejny_smerr)
close (pruzina)
close (netlumene_kmity)

function pushbutton6_Callback(hObject, eventdata, handles)
netlumene_kmity
close(skl_kmitu)
close(snell)
close (figure (2))
close (stejny_smerr)
close (pruzina)
close (tlum_kmitani)
```

Zdrojový text aplikace - netlumený harmonický kmitavý pohyb:

```
function varargout = netlumene_kmity(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end

elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end

function slider1_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',num2str(get(handles.slider1,'value')));

amplituda= get(handles.slider1,'value')
kmitocet= get (handles.slider3,'value')
faze=get(handles.slider2,'value')


function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider2_Callback(hObject, eventdata, handles)
set(handles.edit2,'string',num2str(get(handles.slider2,'value')));

amplituda= get(handles.slider1,'value')
kmitocet= get (handles.slider3,'value')
faze=get(handles.slider2,'value')

function slider2_CreateFcn(hObject, eventdata, handles)
```

```
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider3_Callback(hObject, eventdata, handles)
set(handles.edit3,'string',num2str(get(handles.slider3,'value')));

amplituda= get(handles.slider1,'value')
kmitocet= get (handles.slider3,'value')
faze=get(handles.slider2,'value')

function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function edit1_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit1,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider1,'min') & ...
      val<= get(handles.slider1,'max')
      set(handles.slider1,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit1,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
kmitani_harmonickee(amplituda,faze,kmitocet)

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit2,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider2,'min') & ...
      val<= get(handles.slider2,'max')
      set(handles.slider2,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit2,'string',...
```

```
        [handles.errorstring,num2str(handles.number0ferrors)]);
    guidata(gcbo,handles);
end

function edit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
kmitani_harmonickee(amplituda,faze,kmitocet)


function edit3_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit3,'string'))'
if isnumeric(val) & length(val)==1 & ...
        val>= get(handles.slider3,'min') & ...
        val<= get(handles.slider3,'max')
        set(handles.slider3,'value',val);
else
    handles.number0ferrors=handles.number0ferrors+1;
    set(handles.edit3,'string',...
        [handles.errorstring,num2str(handles.number0ferrors)]);
    guidata(gcbo,handles);
end
kmitani_harmonickee(amplituda,faze,kmitocet)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pushbutton3_Callback(hObject, eventdata, handles)
figure (2)
amplituda= get(handles.slider1,'value')
kmitocet= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
kmitani_harmonickee(amplituda,faze,kmitocet)
```

Zdrojový text aplikace – kmitání netlumeného harmonického oscilátoru

```
function varargout = netlumene_kmity(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end

elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end


function edit1_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit1,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider1,'min') & ...
      val<= get(handles.slider1,'max')
      set(handles.slider1,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit1,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
pohybnapruzine(amplituda,hmotnost,faze,tuhost)

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit2,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider2,'min') & ...
      val<= get(handles.slider2,'max')
      set(handles.slider2,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit2,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
pohybnapruzine(amplituda,hmotnost,faze,tuhost)

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit3_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit3,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider3,'min') & ...
      val<= get(handles.slider3,'max')
      set(handles.slider3,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit3,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
pohybnapruzine(amplituda,hmotnost,faze,tuhost)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit4,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider4,'min') & ...
      val<= get(handles.slider4,'max')
```

```
    set(handles.slider4,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit4,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
pohybnapruzine(amplituda,hmotnost,faze,tuhost)

function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function pushbutton1_Callback(hObject, eventdata, handles)
figure (2)
amplituda= get(handles.slider1,'value')
tuhost= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
hmotnost=get(handles.slider4,'value')
pohybnapruzine(amplituda,hmotnost,faze,tuhost)


function slider1_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',num2str(get(handles.slider1,'value')));

amplituda= get(handles.slider1,'value')
tuhost= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
hmotnost=get(handles.slider4,'value')

function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function slider2_Callback(hObject, eventdata, handles)
set(handles.edit2,'string',num2str(get(handles.slider2,'value')));

amplituda= get(handles.slider1,'value')
tuhost= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
hmotnost=get(handles.slider4,'value')

function slider2_CreateFcn(hObject, eventdata, handles)
```

```
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider3_Callback(hObject, eventdata, handles)
set(handles.edit3,'string',num2str(get(handles.slider3,'value')));

amplituda= get(handles.slider1,'value')
tuhost= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
hmotnost=get(handles.slider4,'value')
function slider3_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider4_Callback(hObject, eventdata, handles)
set(handles.edit4,'string',num2str(get(handles.slider4,'value')));

amplituda= get(handles.slider1,'value')
tuhost= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
hmotnost=get(handles.slider4,'value')

function slider4_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

Zdrojový text aplikace – Lissajousovy obrazce

```
function varargout = skl_kmitu(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end

elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end


function slider1_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',num2str(get(handles.slider1,'value')));

amplituda_2= get(handles.slider5,'value')
amplituda_1= get(handles.slider1,'value')
kmitocet_1= get (handles.slider2,'value')
kmitocet_2= get (handles.slider3,'value')
faze=get(handles.slider4,'value')

function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider2_Callback(hObject, eventdata, handles)
set(handles.edit2,'string',num2str(get(handles.slider2,'value')));

amplituda_1= get(handles.slider1,'value')
amplituda_2= get(handles.slider5,'value')
kmitocet_1= get (handles.slider2,'value')
```

```
kmitocet_2= get (handles.slider3,'value')
faze=get(handles.slider4,'value')

function slider2_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider3_Callback(hObject, eventdata, handles)
set(handles.edit3,'string',num2str(get(handles.slider3,'value')));

amplituda_1= get(handles.slider1,'value')
amplituda_2= get(handles.slider5,'value')
kmitocet_1= get (handles.slider2,'value')
kmitocet_2= get (handles.slider3,'value')
faze=get(handles.slider4,'value')
function slider3_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider4_Callback(hObject, eventdata, handles)
set(handles.edit4,'string',num2str(get(handles.slider4,'value')));

amplituda_1= get(handles.slider1,'value')
amplituda_2= get(handles.slider5,'value')
kmitocet_1= get (handles.slider2,'value')
kmitocet_2= get (handles.slider3,'value')
faze=get(handles.slider4,'value')

function slider4_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider5_Callback(hObject, eventdata, handles)
set(handles.edit5,'string',num2str(get(handles.slider5,'value')));

amplituda_1= get(handles.slider1,'value')
amplituda_2= get(handles.slider5,'value')
kmitocet_1= get (handles.slider2,'value')
kmitocet_2= get (handles.slider3,'value')
faze=get(handles.slider4,'value')
```

```
function slider5_CreateFcn(hObject, eventdata, handles)


if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function edit1_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit1,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider1,'min') & ...
      val<= get(handles.slider1,'max')
      set(handles.slider1,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit1,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles);
end
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit2,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider2,'min') & ...
      val<= get(handles.slider2,'max')
      set(handles.slider2,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit2,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles);
end
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function edit3_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit3,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider3,'min') & ...
      val<= get(handles.slider3,'max')
      set(handles.slider3,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit3,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles);
end
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit4,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider4,'min') & ...
      val<= get(handles.slider4,'max')
      set(handles.slider4,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit4,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles);
end
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)

function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit5_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit5,'string'))'
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider5,'min') & ...
      val<= get(handles.slider5,'max')
      set(handles.slider5,'value',val);
```

```
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit5,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles);
end
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)

function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function pushbutton1_Callback(hObject, eventdata, handles)
figure (2)
amplituda_1= get(handles.slider1,'value')
amplituda_2= get(handles.slider5,'value')
kmitocet_1= get (handles.slider2,'value')
kmitocet_2= get (handles.slider3,'value')
faze=get(handles.slider4,'value')
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)
```

Zdrojový text aplikace – Snellův zákon

```
function varargout = Snell(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end

% End initialization code - DO NOT EDIT
elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end


function varargout = Snell_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
set(handles.indexbutton,'Visible','off')
set(handles.criticalbutton,'Visible','off')


function n1_input_Callback(hObject, eventdata, handles)

function n1_input_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function n2_input_Callback(hObject, eventdata, handles)

function n2_input_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function alpha1_input_Callback(hObject, eventdata, handles)

alpha1_deg = str2double(get(handles.alpha1_input,'String'));
if alpha1_deg>90
   set(handles.slider1,'value',90)
elseif alpha1_deg<0
   set(handles.slider1,'value',0)
else
   set(handles.slider1,'value',eval(get(hObject,'string')))
end

function alpha1_input_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function plot_button_Callback(hObject, eventdata, handles)

n1 = str2double(get(handles.n1_input,'String'));
n2 = str2double(get(handles.n2_input,'String'));
alpha1_deg = str2double(get(handles.alpha1_input,'String'));

% hlavni
set(handles.indexbutton,'Visible','off')
text(-7,-9,'Chyba: n<1 nebo n>4','FontSize',12,'Color',[0.925 0.914
0.847]);
if (n1<1)
    set(handles.indexbutton,'Visible','off')
    set(handles.indexbutton,'string','Help')
    text(-7,-9,'Chyba: n<1 nebo n>4','FontSize',12,'Color',[0.925 0.914
0.847]);
    set(handles.n1_input,'string','1')
    n1=1;
end
if (n1>4)
    set(handles.indexbutton,'Visible','off')
    set(handles.indexbutton,'string','Help')
    text(-7,-9,'Chyba: n<1 nebo n>4','FontSize',12,'Color',[0.925 0.914
0.847]);
```

```
    set(handles.n1_input,'string','4')
    n1=4;
end
if (n2<1)
    set(handles.indexbutton,'Visible','off')
    set(handles.indexbutton,'string','Help')
    text(-7,-9,'Chyba: n<1 nebo n>4','FontSize',12,'Color',[0.925 0.914
0.847]);
    set(handles.n2_input,'string','1')
    n2=1;
end
if (n2>4)
    set(handles.indexbutton,'Visible','off')
    set(handles.indexbutton,'string','Help')
    text(-7,-9,'Chyba: n<1 nebo n>4','FontSize',12,'Color',[0.925 0.914
0.847]);
    set(handles.n2_input,'string','4')
    n2=4;
end
if (alpha1_deg>90)
    alpha1_deg=90;
    set(handles.alpha1_input,'string','90')
end
if (alpha1_deg<0)
    alpha1_deg=0;
    set(handles.alpha1_input,'string','0')
end


b1=-n1/3+4/3;
b2=-n2/3+4/3;
alpha1 = alpha1_deg/(180/pi);
alpha2=asin(n1*sin(alpha1)/n2);
if imag(alpha2)~=0;
     alpha2=0;
end
alpha2_deg = alpha2*180/pi;

% vytvoreni grafu
axes(handles.snell_axes)
f=7;
patch([-f -f 0 0 -f],[-f f f -f -f],[1 1 b1])
hold on;
patch([f f 0 0 f],[-f f f -f -f],[1 b2 1])
plot([-f f],[0 0],'-.')

set(handles.criticalbutton,'Visible','off')
if (n1>n2)&(alpha1>asin(n2/n1))
   text(10,-9,'\theta_1>\theta_C','FontSize',18, 'FontWeight', 'bold',
'Color',[0.925 0.914 0.847]);
```

```
    set(handles.criticalbutton,'Visible','off')
    set(handles.criticalbutton,'String','Help')

    y=@(x) tan(alpha1)*x;
    z=@(x) - tan(alpha1)*x;
    fplot(y, [-f 0], 'k-');
    fplot(z, [-f 0], 'k-');
    axis([-f f -f f])
    set(handles.alpha1_input,'string',['X'])
else
    text(3,-9,'\theta_1>\theta_C','FontSize',18, 'FontWeight', 'bold',
'Color',[0.925 0.914 0.847]);

    y=@(x) tan(alpha1)*x*(1-sign(x))/2 + tan(alpha2)*x*(1+sign(x))/2;
    fplot(y, [-f f], 'k-');
    axis([-f f -f f])
    set(handles.alpha1_input,'string',[num2str(alpha2_deg) ' deg'])
end

set(handles.snell_axes,'XMinorTick','on')
grid on

function slider1_Callback(hObject, eventdata, handles)

set(handles.alpha1_input,'string',get(hObject,'value'))


function slider1_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function edit4_Callback(hObject, eventdata, handles)


function edit4_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit7_Callback(hObject, eventdata, handles)
```

```
function edit7_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColo
r'));
end
```

Zdrojový text aplikace – Skládání kmitů stejného směru

```
function varargout = stejny_smerr(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end

elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end

function slider1_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',num2str(get(handles.slider1,'value')));

amplituda= get(handles.slider1,'value')
kmitocet_1= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
kmitocet_2=get(handles.slider4,'value')

function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
function slider2_Callback(hObject, eventdata, handles)
set(handles.edit2,'string',num2str(get(handles.slider2,'value')));

amplituda= get(handles.slider1,'value')
kmitocet_1= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
kmitocet_2=get(handles.slider4,'value')

function slider2_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider3_Callback(hObject, eventdata, handles)
set(handles.edit3,'string',num2str(get(handles.slider3,'value')));

amplituda= get(handles.slider1,'value')
kmitocet_1= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
kmitocet_2=get(handles.slider4,'value')

function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider4_Callback(hObject, eventdata, handles)
set(handles.edit4,'string',num2str(get(handles.slider4,'value')));

amplituda= get(handles.slider1,'value')
kmitocet_1= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
kmitocet_2=get(handles.slider4,'value')

function slider4_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function edit1_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit1,'string'));
if isnumeric(val) & length(val)==1 & ...
```

```
    val>= get(handles.slider1,'min') & ...
    val<= get(handles.slider1,'max')
    set(handles.slider1,'value',val);
else
  handles.number0ferrors=handles.number0ferrors+1;
  set(handles.edit1,'string',...
    [handles.errorstring,num2str(handles.number0ferrors)]);
  guidata(gcbo,handles)
end
skladani_kmitu_smer(amplituda,kmitocet_1,kmitocet_2,faze)

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
  set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit2,'string'));
if isnumeric(val) & length(val)==1 & ...
    val>= get(handles.slider2,'min') & ...
    val<= get(handles.slider2,'max')
    set(handles.slider2,'value',val);
else
  handles.number0ferrors=handles.number0ferrors+1;
  set(handles.edit2,'string',...
    [handles.errorstring,num2str(handles.number0ferrors)]);
  guidata(gcbo,handles)
end
skladani_kmitu_smer(amplituda,kmitocet_1,kmitocet_2,faze)

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
  set(hObject,'BackgroundColor','white');
end


function edit3_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit3,'string'));
if isnumeric(val) & length(val)==1 & ...
    val>= get(handles.slider3,'min') & ...
    val<= get(handles.slider3,'max')
    set(handles.slider3,'value',val);
else
  handles.number0ferrors=handles.number0ferrors+1;
  set(handles.edit3,'string',...
    [handles.errorstring,num2str(handles.number0ferrors)]);
  guidata(gcbo,handles)
```

```
end
skladani_kmitu_smer(amplituda,kmitocet_1,kmitocet_2,faze)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit4,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider4,'min') & ...
      val<= get(handles.slider4,'max')
      set(handles.slider4,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit4,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
skladani_kmitu_smer(amplituda,kmitocet_1,kmitocet_2,faze)

function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function pushbutton1_Callback(hObject, eventdata, handles)
figure (2)
amplituda= get(handles.slider1,'value')
kmitocet_1= get (handles.slider3,'value')
faze=get(handles.slider2,'value')
kmitocet_2=get(handles.slider4,'value')
skladani_kmitu_smer(amplituda,kmitocet_1,kmitocet_2,faze)
```

Zdrojový text aplikace – Tlumený kmitavý pohyb

```
function varargout = tlum_kmitani(varargin)

if nargin==0
   fig=openfig(mfilename,'reuse');
   set(fig,'color',get(0,'defaultuicontrolbackgroundcolor'));
   handles=guihandles(fig);
   guidata(fig,handles);

   if nargout>0
      varargout{1}=fig;
   end


elseif ischar(varargin{1})
   try
      if (nargout)
         [varargout{1:nargout}]=feval(varargin{:});
      else
         feval(varargin{:});
      end
   catch
      disp(lasterr);
   end
end

function edit1_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit1,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider1,'min') & ...
      val<= get(handles.slider1,'max')
      set(handles.slider1,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit1,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
tlumene_kmity(amplituda,kmitocet,faze,utlum)

function edit1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit2,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider2,'min') & ...
      val<= get(handles.slider2,'max')
      set(handles.slider2,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit2,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
tlumene_kmity(amplituda,kmitocet,faze,utlum)

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit3_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit3,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider3,'min') & ...
      val<= get(handles.slider3,'max')
      set(handles.slider3,'value',val);
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit3,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
tlumene_kmity(amplituda,kmitocet,faze,utlum)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit4_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit4,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider4,'min') & ...
      val<= get(handles.slider4,'max')
      set(handles.slider4,'value',val);
```

```
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit4,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
tlumene_kmity(amplituda,kmitocet,faze,utlum)

function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function slider1_Callback(hObject, eventdata, handles)
set(handles.edit1,'string',num2str(get(handles.slider1,'value')));

amplituda= get(handles.slider1,'value')
utlum= get (handles.slider3,'value')
kmitocet=get(handles.slider2,'value')
faze=get(handles.slider4,'value')
cas=get(handles.slider5,'value')

function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider2_Callback(hObject, eventdata, handles)
set(handles.edit2,'string',num2str(get(handles.slider2,'value')));

amplituda= get(handles.slider1,'value')
utlum= get (handles.slider3,'value')
kmitocet=get(handles.slider2,'value')
faze=get(handles.slider4,'value')
cas=get(handles.slider5,'value')

function slider2_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider3_Callback(hObject, eventdata, handles)
set(handles.edit3,'string',num2str(get(handles.slider3,'value')));
```

```
amplituda= get(handles.slider1,'value')
utlum= get (handles.slider3,'value')
kmitocet=get(handles.slider2,'value')
faze=get(handles.slider4,'value')
cas=get(handles.slider5,'value')

function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function slider4_Callback(hObject, eventdata, handles)
set(handles.edit4,'string',num2str(get(handles.slider4,'value')));

amplituda= get(handles.slider1,'value')
utlum= get (handles.slider3,'value')
kmitocet=get(handles.slider2,'value')
faze=get(handles.slider4,'value')
cas=get(handles.slider5,'value')

function slider4_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function pushbutton1_Callback(hObject, eventdata, handles)
figure (2)
amplituda= get(handles.slider1,'value')
utlum= get (handles.slider3,'value')
kmitocet=get(handles.slider2,'value')
faze=get(handles.slider4,'value')
cas=get(handles.slider5,'value')
tlumene_kmity(amplituda,kmitocet,faze,utlum,cas)



function edit5_Callback(hObject, eventdata, handles)
val=str2double(get(handles.edit5,'string'));
if isnumeric(val) & length(val)==1 & ...
      val>= get(handles.slider5,'min') & ...
      val<= get(handles.slider5,'max')
      set(handles.slider5,'value',val);
```

```
else
   handles.number0ferrors=handles.number0ferrors+1;
   set(handles.edit5,'string',...
      [handles.errorstring,num2str(handles.number0ferrors)]);
   guidata(gcbo,handles)
end
tlumene_kmity(amplituda,kmitocet,faze,utlum)

function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function slider5_Callback(hObject, eventdata, handles)
set(handles.edit5,'string',num2str(get(handles.slider5,'value')));

amplituda= get(handles.slider1,'value')
utlum= get (handles.slider3,'value')
kmitocet=get(handles.slider2,'value')
faze=get(handles.slider4,'value')
cas=get(handles.slider5,'value')

function slider5_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

Zdrojový text funkce - Netlumený harmonický kmitavý pohyb

```
%Funkce pro netlumene harmonicke kmity
%-----------------------%
function kmitani_harmonickee(amplituda, faze, kmitocet)% hlavicka
funkce

%Definice konstant a promennych
%---------------------------%
figure(2)
a=amplituda;
f=faze;
k=kmitocet;
t=0:0.001:2;

%Vypocet kmitu a vykresleni grafu
%-----------------------------%
y=a*sin(((2*pi*k)*t)+(pi/180)*f);
```

```
plot(t,y)                %vystup grafu
grid on
title('Netlumeny harmonicky kmitavy pohyb')
xlabel('cas - t [s]')
ylabel('výchylka - y [m]')
```

Zdrojový text funkce - Kmitání harmonického netlumeného oscilátoru

```
%Funkce pro netlumeny harmonicky oscilator
%-------------------------%
function pohybnapruzine(amplituda,hmotnost,faze,tuhost)  % hlavicka
funkce

%Definice konstant a promennych
%---------------------------%
figure(2)
a=amplituda;
f=faze;
kk=tuhost;
m=hmotnost;
k=(kk/m)^(0.5);
t=0:0.001:10;

%Vypocet kmitu a vykresleni grafu
%-----------------------------%
y = a*sin(((2*pi*k)*t) +((pi/180)*f));
v=a*k*cos(((2*pi*k)*t) +((pi/180)*f));
a=-a*(k)^2*sin(((2*pi*k)*t) +((pi/180)*f));

 plot(t,y)
 subplot(3,1,1);
 plot(t,y);
 grid on;
 title('Netlumeny harmonicky kmitavy pohyb y(t)')
 xlabel('cas - t [s]')
 ylabel('výchylka - y [m]')%vystup 1. grafu

 subplot(3,1,2);
 plot(t,v);              %vystup 2. grafu
 grid on;
 title('Rychlost harmonickeho kmitaveho pohybu v(t)')
 xlabel('cas - t [s]')
 ylabel('rychlost - v [m/s]')

 subplot(3,1,3);
 plot(t,a);              %vystup 3. grafu
 grid on;
 title('Zrychleni harmonickeho kmitaveho pohybu a(t)')
 xlabel('cas - t [s]')
 ylabel('zrychlení - a [m/s*s]')
```

Zdrojový text funkce - Tlumený kmitavý pohyb

```
%Funkce pro tlumene kmity
%-----------------------%
function tlumene_kmity(amplituda,kmitocet,faze,utlum,cas) %
hlavicka funkce

%Definice konstant a promennych
%-------------------------%
figure(2)
a=amplituda;
f=faze;
k = (2*pi*kmitocet);
t=0:0.001:cas;

%Vypocet kmitu a vykresleni grafu
%-------------------------%
y = amplituda.*exp(-(utlum.*t)).*sin(k.*t + faze); % rovnice kmitu %
vystup kmitu na graf


 plot(t,y)    %vystup kmitu na graf
 grid on
 title('Tlumene kmitani')
 xlabel('cas - t [s]')
 ylabel('výchylka - y [m]')
```

Zdrojový text funkce – Skládání kmitů stejného směru

```
%Funkce pro skladani kmitu stejneho smeru
%-------------------------------------%
function skladani_kmitu_smer(amplituda,kmitocet_1,kmitocet_2,faze)
% hlavicka funkce

%Definice konstant a promennych
%-------------------------%
figure(2)
a=amplituda;
f=faze;
k_1=kmitocet_1;
k_2=kmitocet_2;
t=0:0.001:2;

%Vypocet kmitu a vykresleni grafu
%-------------------------%
x = a*sin(((2*pi*k_1)*t));              % rovnice 1. kmitu
y = a*sin(((2*pi*k_2)*t) +((pi/180)*faze)); % rovnice 2. kmitu
z = x + y;                              % soucet obou kmitu ve smeru osy X
```

```
subplot(3,1,1);
grid on
plot(t,x);                    %vystup 1. kmitu na graf
title('1. harmonicky kmitavy pohyb')
xlabel('cas - t [s]')
ylabel('amplituda - A [m]')

subplot(3,1,2);
grid on
plot(t,y);                    %vystup 2. kmitu na graf
title('2. harmonicky kmitavy pohyb')
xlabel('cas - t [s]')
ylabel('amplituda - A [m]')

subplot(3,1,3);
grid on
plot(t,z);                    %vystup slozenych kmitu na graf
title('Slozeni kmitu stejneho smeru')
xlabel('cas - t [s]')
ylabel('amplituda - A [m]')
```

Zdrojový text funkce – Lissajousovy obrazce

```
%Funkce pro skladani kolmych kmitu
%-------------------------------------%
function
skladani_kmitu(amplituda_1,amplituda_2,kmitocet_1,kmitocet_2,faze)
% hlavicka funkce

%Definice konstant a promennych
%----------------------------%
figure(2)
a_1=amplituda_1;
a_2=amplituda_2;
f=faze;
k_1=kmitocet_1;
k_2=kmitocet_2;
t=0:0.001:2;

%Vypocet kmitu a vykresleni grafu
%-----------------------------%
x = a_1*sin(2*pi*k_1*t);              % rovnice 1. kmitu
y = a_2*sin(2*pi*k_2*t +(pi/180)*f);  % rovnice 2. kmitu
z = x + y;


  subplot(3,1,1);
  plot(t,x);
```

```
grid on;
title('1. harmonicky kmitavy pohyb')
xlabel('cas - t [s]')
ylabel('výchylka - y [m]')   %vystup 1. kmitu na graf

subplot(3,1,2);
plot(t,y);                   %vystup 2. kmitu na graf
grid on;
title('2. harmonicky kmitavy pohyb')
xlabel('cas - t [s]')
ylabel('výchylka - y [m]')

 subplot(3,1,3);
 plot(x,y);     %vystup slozenych kolmych harmonickych kmitu na
 graf
  grid on;
  title('Slozeni kolmych harmonickych kmitu ')
  ylabel('výchylka - y [m]')
```