UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Používání knihoven Matlabu v programu

Radek Novotný

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Radek NOVOTNÝ**
Osobní číslo:	**I08127**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Používání knihoven Matlabu v programu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Teoretická část:
a) převod lineární diferenciální rovnice vyššího řádu na soustavu dif. rovnic řádu prvního
b) nastudovat a popsat možnosti používání výpočetních funkcí MATLABu v programu (jazyk JAVA)

Praktická část:
- vytvořit ukázkovou aplikaci pro výpočet lineární diferenciální rovnice zadané ve formě přenosu, která bude umožňovat:
a) volbu řádu a parametrů přenosu
b) výběr vstupního signálu a volbu jeho parametrů
c) vykreslení průběhu řešení pro zvolený časový interval

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***Matlab a Simulink Úvod do používání - Ing.František Dušek.Csc**
***Matlab Builder JA 2 Users Guide - Mathworks**
***Matlab 7 External Interfaces - Mathworks**
***MATLAB Reference Guide - MathWorks**
***Učebnice jazyka java - P.Herout**
***Matlab online documentations - Mathworks**
***Automatické řízení - Balátě J.**
***Teorie dynamických systémů - Štecha J., Havlena V.**

Vedoucí bakalářské práce: **doc. Ing. František Dušek, CSc.**
Katedra řízení procesů

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 2. 08. 2011 Radek Novotný

Poděkování

Chtěl bych touto cestou poděkovat doc. Ing. Františku Duškovi .CSc za vedení bakalářské práce a za věcné připomínky pří její tvorbě. Dále pak všem, kteří mi poskytli podmínky k vytvoření této práce.

Anotace

Tato práce se zabývá implementací knihoven programu Matlab do prostředí programového jazyka Java za účelem výpočtu dynamických systémů, které jsou zadané ve formě přenosu. Tento přenos je realizován jako diferenciální rovnice nejvýše 5. řádu, se speciální pravou stranou.

Klíčová slova

dynamický systém, diferenciální rovnice, compiler, matlab, java

Title

Using Matlab libraries in the application

Annotation

This work deals with the implementation of the library environment in Matlab to programming language Java in order to construct dynamic systems that are specified in the form of transfer. This transfer is implemented as differential equation up 5th regulations, the special right away.

Keywords

dynamic system, differential equations, compiler, matlab, java

Obsah

Seznam zkratek

API	Application Programming Interface
JDK	Java Development Kit
JRE	Java Runtime Enviroment
JVM	Java Virtual Machine
LTI	Linear Time Invariant systém
MCR	Matlab Compiler Runtime
MIMO	Multi Input Multi Output systém
RK4	Metoda Runge-Kutta 4. řádu
SISO	Single Input Single Output system

Seznam obrázků

Seznam tabulek

Úvod

Náplní bakalářské práce je vytvoření programu v jazyce Java využívajícího výpočetní jádro tvořené funkcemi vytvořených v programovém prostředí Matlab pro výpočet lineárních dynamických systémů s jednou vstupní veličinou a jednou výstupní veličinou. Lineární dynamický systém je popsán formou obecné lineární diferenciální rovnice maximálně 5. řádu s derivací pravé strany spolu s počátečními podmínkami a volbou vstupního signálu.

Cílem práce je poskytnout sledování časových průběhů lineárního dynamického systému. Uživatel může měnit chování dynamického systému a včetně parametrů levé a pravé strany diferenciální rovnice také i počáteční podmínky rovnice. Může volit parametry u tří typů vstupních signálů. Mezi vstupní signály, které jsou uživateli k dispozici, patří sinusový signál s volbou amplitudy, periody a fázového posunu, obdélníkový signál s volbou velikosti a periody a nakonec skoková změna se zadáním velikosti skoku a posunu. Aplikace umožňuje i volbu řádu diferenciální rovnice počínaje prvním řádem a končící maximálně řádem pátým.

Práce je tvořena dvěma částmi. V první části jsou uvedeny teoretické informace, které blíže souvisí s dynamickým systémem. Jsou popsány postupy pro numerický výpočet dynamického systému zadaného ve formě diferenciální rovnice vyššího řádu. Je zde uveden i postup převodu diferenciální rovnice vyššího řádu na soustavu diferenciálních rovnic řádu prvního a s tím spjatý přepočet počátečních podmínek. Do této části je také zahrnuta problematika možnosti používání výpočetních funkcí Matlabu v programu naprogramovaném v jazyku Java.

V druhé části jsou popsány postupy postupné kompilace projektu až do finální podoby. Jsou zde k vidění funkce napsané v programu Matlab, jejich kompilace a volání z jazyka Java. Druhá část obsahuje zejména tvorbu vlastní uživatelské aplikace, pro výpočet lineárních diferenciálních rovnic zadaných ve formě přenosu, v programovacím jazyku Java. Nalezneme zde uživatelské rozhraní pro zvolení parametrů, počátečních podmínek a dalších vlastností popisujících zadávaný dynamický systém. Jeden z nejdůležitějších obsahů, je zde podrobný popis všech potřebných nástrojů pro spuštění aplikace na jakémkoliv počítači u jakéhokoliv uživatele.

Teoretická část

1. Dynamický systém

1.1 Úvod

Dynamický systém je systém, jehož chování je popsáno veličinami, které se v čase mění. Lze tedy říci, že dynamický systém je systém, který se v čase vyvíjí.

Okolí působí na systém pomocí vstupních veličin (signálů) označovaných *u(t)* a naopak systém působí na okolí prostřednictvím výstupních veličin (signálů) označovaných *y(t).* Symbol *t* zdůrazňuje, že jde o časově proměnné veličiny.

Působí-li na systém více (*m*) vstupních veličin, pak vstup uvažujeme jako vektorovou veličinu o *m* složkách:

$$u(t) = [u_1(t), u_2(t), ..., u_m(t)]^T \tag{1}$$

Stejně tak i výstupní veličina *y(t)* může být vektorová veličina o *r* složkách, pak:

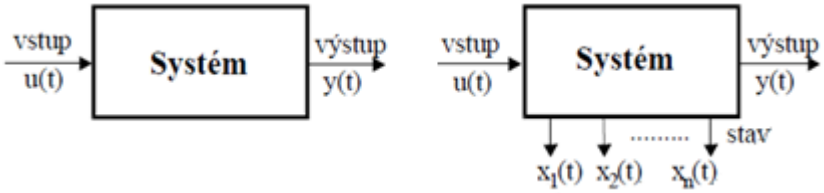
$$y(t) = [y_1(t), y_2(t), ..., y_r(t)]^T \tag{2}$$

Úplný popis dynamického systému vyžaduje, kromě informace o průběhu vstupů, také informaci o aktuálním stavu. Tato informace může být vyjádřena buď pomocí počátečních podmínek tj. derivací (ve spojitém případě) nebo minulých hodnot (v diskrétním případě) vstupů a výstupů nebo pomocí hodnot stavových veličin tzv. stavu. Stav systému je soubor vnitřních veličin systému označován jako *x(t).* Jedná se znovu o vektor, tentokrát o rozměru *n,* platí tedy:

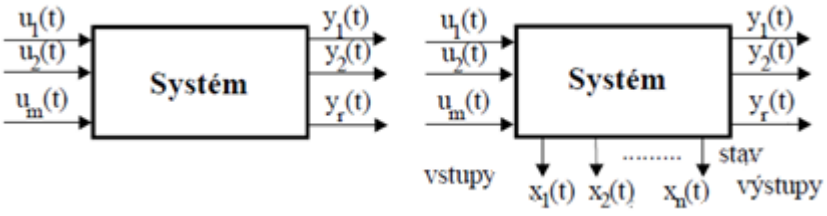
$$x(t) = [x_1(t), x_2(t), ..., x_n(t)]^T \tag{3}$$

1.2 Druhy dynamických systémů

Podle počtu vstupů a výstupů rozlišujeme dva základní typy a to systém *SISO* (Single Input Single Output), který se vyznačuje tím, že je tvořen pouze jednou vstupní a jednou výstupní veličinou (Obrázek 1). Naproti tomu existuje dynamický systém s názvem *MIMO* (Multi Input Multi Output), nebo-li systém tvořený vícero vstupními a několika výstupními veličinami (Obrázek 2).



Obrázek 1 - SISO systém



Obrázek 2 - MIMO systém

1.3 Popis dynamického systému

U dynamických systémů rozlišujeme dva základní druhy popisů. Vnitřní popis systému vyjadřuje dynamické závislosti mezi vstupem, vnitřním stavem a výstupem systému tzv. stavový popis. O stavovém popisu si něco řekneme v některé z dalších kapitol. Vnější popis systému vyjadřuje dynamické závislosti pouze mezi vstupem a výstupem systému.

V následující kapitole se budeme zabývat jen lineárními systémy, které jsou popsány lineární diferenciální rovnicí s konstantními časově nezávislými koeficienty. Takový systém se nazývá lineárně časově invariantní (LTI Linear Time Invariant). Dále budeme předpokládat, že systém bude typu SISO (Obrázek 1), tedy s jedním vstupem a jedním výstupem.

2. Vnější popis dynamického systému

V praxi se nejčastěji využívá vnějšího popisu, i pro svoji větší omezenost, než je tomu v případě vnitřního popisu. Jeho základní výhodou je jednoduchost. Získání vnějšího popisu je možné z experimentálně získaných průběhů vstupních a výstupních veličin. Nyní se seznámíme s několika možnostmi vnějšího popisu dynamického systému. Vnější popis systému SISO popisuje závislost pouze mezi jedním vstupem a jedním výstupem. Existuje několik vyjádření. Všechny tyto druhy vnějšího popisu jsou stejné (ekvivalentní). Mezi způsoby vnějšího popisu patří:

- diferenciální rovnice systému
- operátorový přenos systému
- rozložení pólů a nul systému
- frekvenční přenos systému
- frekvenční, impulsní nebo přechodová charakteristika

2.1 Diferenciální rovnice systému

Pokud lineární dynamický systém má *n* nezávislých kapacit, pak je popsán diferenciální rovnicí *n*-tého řádu s konstantními koeficienty a s *n* počátečními podmínkami. A jedná-li se o diferenciální rovnici s pravou stranou tzv. nehomogenní diferenciální rovnici, pak je z matematiky známo, že výsledek je dán součtem řešení homogenní diferenciální rovnice a libovolného partikulárního řešení rovnice s pravou stranou. V této práci se budeme zabývat speciálním tvarem pravé strany tvořeným lineární kombinací derivací vstupního signálu. Zkráceně lze zapisovat ve tvaru:

$$\sum_{i=0}^{r} a_i \frac{d^i y(t)}{dt^i} = \sum_{i=0}^{m} b_i \frac{d^i u(t)}{dt^i} \tag{4}$$

U reálných dynamických systémů popsaných diferenciální rovnicí platí podmínka fyzikální realizovatelnosti, kdy $r \geq m$ nebo $r > m$. Podmínka fyzikální realizovatelnosti je podmínkou kauzality. Kdyby byl totiž stupeň jmenovatele nižší než stupeň čitatele, vznikala by odezva na budící signál ještě dříve než samotný budící signál. Pro konkrétní řešení rovnice (4) je potřeba kromě počátečních podmínek znát i konkrétní průběh vstupního signálu. Protože k numerickému řešení bude využit algoritmus Matlabu vyžadující popis ve formě soustavy diferenciálních rovnic prvního řádu (ekvivalent stavového popisu), bude pozornost věnována i převodu vstupně výstupního popisu na stavový a transformaci počátečních podmínek do ekvivalentního počátečního stavu.

2.2 Operátorová forma popisu systému

Jedná se o vyjádření diferenciální rovnice dynamického systému v jiné formě. Operátorovou formu získáme aplikací Laplaceovy integrální transformace na původní lineární diferenciální rovnici.

Máme-li dynamický systém popsaný diferenciální rovnicí zkráceně zapsanou (4) s nulovými počátečními podmínkami můžeme provést Laplaceovu transformaci na levou i pravou stranu diferenciální rovnice takto:

$$L\left\{\sum_{i=0}^{r} a_i \frac{d^i y(t)}{dt^i}\right\} = L\left\{\sum_{i=0}^{m} b_i \frac{d^i u(t)}{dt^i}\right\} \tag{5}$$

Využijeme-li vlastnosti linearity a vlastnosti o obrazu derivace poté obdržíme obraz původní diferenciální rovnice jako:

$$\sum_{i=0}^{r} a_i s^i Y(s) = \sum_{i=0}^{m} b_i s^i U(s) \tag{6}$$

Zavedeme-li přenos jako podíl obrazu výstupu *Y(s)* ku obrazu vstupu *U(s)* při nulových počátečních podmínkách, dostáváme poměr dvou polynomů, v čitateli je to polynom $B_m(s)$, který je řádu *m* a je roven nejvyšší derivaci diferenciální rovnice na pravé straně. Stejně tak ve jmenovateli obdržíme polynom $A_r(s)$, který je řádu *r* a je roven nejvyšší derivaci diferenciální rovnice na levé straně. Polynom $A_r(s)$ se nazývá *charakteristický polynom* a jeho řád odpovídá řádu dynamického systému.

$$F(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{i=0}^{m} b_i s^i}{\sum_{i=0}^{r} a_i s^i} = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_0}{a_r s^r + a_{r-1} s^{r-1} + ... + a_0} = \frac{B_m(s)}{A_r(s)} \tag{7}$$

2.3 Rozložení pólů a nul

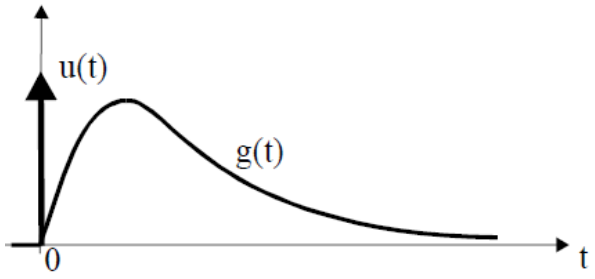
V předešlé kapitole jsme zjistili, že operátorový přenos je podíl dvou polynomů (7). Z matematiky víme, že každý polynom lze rozložit na součin kořenových činitelů. Vyjádříme-li tedy oba polynomy pomocí součinu kořenových činitelů, dostaneme:

$$F(s) = \frac{Y(s)}{U(s)} = \frac{b_m}{a_r} \frac{(s-n_1)(s-n_2)...(s-n_m)}{(s-s_1)(s-s_2)...(s-s_r)} \tag{8}$$

Čísla n_i, kde i=1,2,…m jsou kořeny polynomu čitatele a nazývají se nuly. Kořeny jmenovatele jsou čísla p_i, kde i=1,2,…r, které se nazývají póly operátorového přenosu. Póly a nuly mohou být obecně komplexní čísla a jejich poloha lze vyjádřit graficky v komplexní rovině. Operátorový přenos je oběma polynomy jednoznačně určen.

2.4 Impulsní charakteristika systému

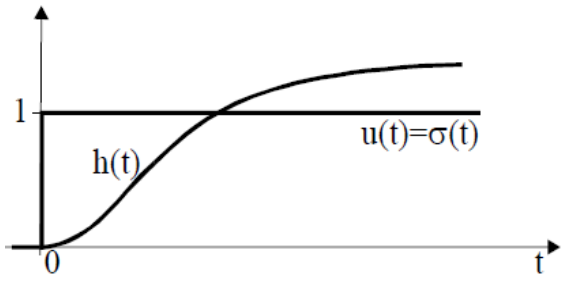
Impulsní charakteristika systému je odezva systému na jednotkový skok (Diracův impuls $\delta(t)$). Tuto odezvu značíme g(t) a příklad jejího typického průběhu je znázorněna na obrázku (Obrázek 3).



Obrázek 3 - Impulsní charakteristika

2.5 Přechodová charakteristika systému

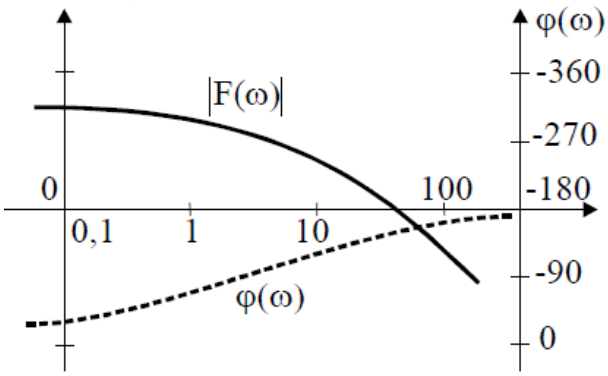
Jedná se o grafické znázornění přechodové funkce *h(t),* která je odezvou systému na jednotkovou změnu $\sigma(t)$ na vstupu. Příklad typického průběhu pro soustavu vyššího řádu je na Obrázku 4.



Obrázek 4 - Přechodová charakteristika

2.6 Frekvenční přenos

Tato forma vnějšího předpisu určuje, jak daný systém přenáší harmonický vstupní signál s měnícím se kmitočtem. Frekvenční přenos získáme tak, že na vstup systému přivedeme harmonický signál. Mezi typické harmonické signály patří sinusový průběh, který je dán předpisem $u(t)=u_0 sin(2\pi ft+\varphi)$, kde A značí amplitudu vstupního signálu, f frekvenci a φ je fázový posun. Přivedeme-li na vstup takovýto harmonický signál, na výstupu, po odeznění přechodového děje, obdržíme také sinusový průběh, ale s jinou amplitudou a posunutý o fázový posun oproti vstupnímu signálu.

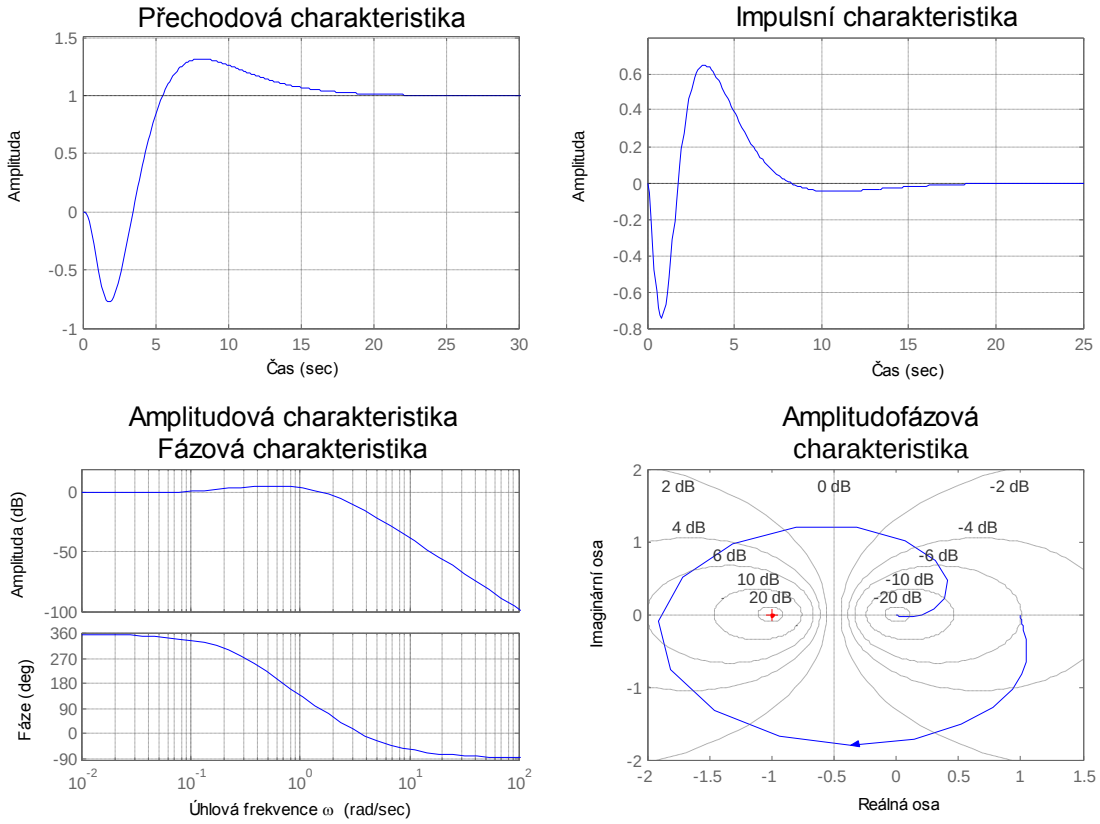


Obrázek 5 - Frekvenční přenos

Frekvenční přenos je funkce komplexního čísla $F(j\omega)$, kterou získáme dosazením $j\omega$ do přenosu (7) za operátor s . Frekvenční přenos se graficky znázorňuje dvěma způsoby.
V případě tzv. amplitudofázové frekvenční charakteristiky se vynáší hodnoty frekvenčního přenosu v závislosti na úhlové frekvenci ω do komplexní roviny. Amplitudová charakteristika je závislost amplitudy (absolutní hodnoty frekvenčního přenosu) na úhlové frekvenci ω a frekvenční charakteristika je závislost fáze (fázový úhel frekvenčního přenosu) na úhlové frekvenci ω. Např. pro systém popsaný přenosem:

$$F(s) = \frac{12.5531\left(-s + \frac{1}{2}\right)\left(s + \frac{1}{6}\right)}{\left(s + \frac{1}{2}\right)^3 \left(s + \frac{1}{2} + j\frac{2\pi}{5}\right)\left(s + \frac{1}{2} - j\frac{2\pi}{5}\right)} \tag{9}$$

jsou výše uvedené charakteristiky zobrazeny na obrázku



Obrázek 6 - Charakteristiky

3. Stavový popis lineárního dynamického systému

Stavový popis dynamického systému je soustava diferenciálních rovnic prvního stupně popisující závislost stavových veličin na vstupech doplněná o tzv. výstupní algebraickou rovnici popisující závislost výstupů na stavových veličinách a případně vstupech. Každou diferenciální rovnici stupně *n* lze převést na *n*-diferenciálních rovnic prvního stupně. Stavový model dynamického systému se skládá z popisu dynamické části, popisujících vývoj stavu (diferenciální rovnice pro *x*) a rovnice výstupu *y(t)*. Stavové rovnice se vyjadřují vazbou derivace stavové proměnné na libovolný vstup nebo výstup a vzájemným vztahem vektoru vstupu a vektoru stavu. Pro lineární dynamický systém pak můžeme stavový popis zapsat v maticovém tvaru:

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$$

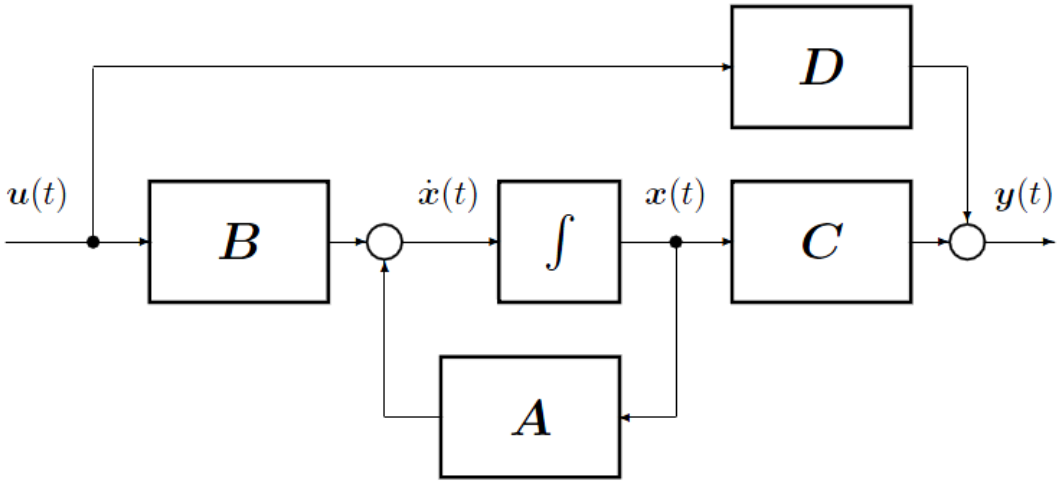
$$y(t) = Cx(t) + Du(t) \tag{10}$$

Systém s modelem (10) se nazývá lineární, časově invariantní systém a běžně se označuje zkratkou *LTI* (Linear Time Invariant system). Stavová rovnice je tvořena maticemi *A*, *B*, *C*, *D*, kde *A* je matice vnitřních vazeb systému, *B* je matice vazeb systému na vstup, *C* matice vazeb výstupu na stav a matice *D*, která se často bere nulová, protože má zanedbatelný vliv na vlastnosti dynamického systému. Pokud je matice D rovna nula, mluví se pak o ryzím dynamickém systému. Jednotlivé matic pak můžeme zapsat:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}$$

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rn} \end{bmatrix} D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ d_{r1} & d_{r2} & \cdots & d_{rm} \end{bmatrix} \tag{11}$$

Stavová rovnice je dále tvořena vektorem vstupů *u(t)*, vektorem výstupu *y(t)* a vektorem stavů *x(t)*. Ve všech třech případech se jedná o vektor sloupcový. Počet prvků vektoru stavu *n* udává řád diferenciální rovnice. Obecné stavové schéma je znázorněno na obrázku (Obrázek 7).



Obrázek 7 - Obecné stavové schéma

Stavová rovnice pro časově invariantní systémy, kde matice *A, B, C* jsou maticemi konstant a matice *D* je nulová, vypadá:

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) \tag{12}$$

Diferenciální rovnice lze řešit několika způsoby. Je na výběr z metod analytických, kam se řadí například variace konstanty, Laplaceova transformace nebo rozvoj v nekonečnou řadu. Na druhé straně můžeme použít metody numerické. Mezi numerické metody se řadí metoda Runge-Kutta a Adamsova metoda.

4. Převedení lineární diferenciální rovnice vyššího řádu na soustavu diferenciálních rovnic řádu prvního

4.1 Lineární diferenciální rovnice s derivací pravé strany

Lineární diferenciální rovnice řádu *n* s derivací pravé strany, která má maximální řád derivace o stupeň nižší s dopravním zpožděním T_d ve vstupní veličině *u(t)*, můžeme zapsat ve tvaru:

$$\begin{aligned} &y^{(n)} + a_{n-1}y^{(n-1)} + a_{n-2}y^{(n-2)},.....,+a_2y^{(2)} + a_1y^{(1)} + a_0y \;\; = \\ &b_{n-1}u^{(n-1)}(t-T_d) + b_{\acute{n}-2}u^{(n-2)}(t-T_d)+,.....,+b_1u^{(1)}(t-T_d) + b_0u\;\;(t-T_d) \end{aligned}$$

(13)

U diferenciální rovnice se zadávají i její počáteční podmínky. Pro diferenciální rovnici s derivací pravé strany (13) pak vypadají ve tvaru:

$$y(0) = y_0 \qquad y^{(1)}(0) = y_1 \qquad y^{(2)}(0) = y_2 \qquad ,\ldots, \qquad y^{(n-2)}(0) = y_{n-2} \qquad y^{(n-1)}(0) = y_{n-1}$$

$$u(0) = u_0 \qquad u^{(1)}(0) = u_1 \qquad u^{(2)}(0) = u_2 \qquad ,\ldots, \qquad u^{(n-2)}(0) = u_{n-2}$$

Uvažujeme-li, dle zadání bakalářské práce, lineární diferenciální rovnici maximálně 5. řádu se zadanými počátečními podmínkami pro výstupní veličinu *y(t)* a pro vstupní veličinu *u(t)*, pak její tvar a její počáteční podmínky zapisujeme jako:

$$\begin{aligned} &y^{(5)} + a_4y^{(4)} + a_3y^{(3)} + a_2y^{(2)} + a_1y^{(1)} + a_0y \;\; = \\ &b_4u^{(4)}(t-T_d) + b_3u^{(3)}(t-T_d) + b_2u^{(2)}(t-T_d) + b_1u^{(1)}(t-T_d) + b_0u\;\;(t-T_d) \end{aligned}$$

$$y(0)=y_0 \qquad y^{(1)}(0)=y_1 \qquad y^{(2)}(0)=y_2 \qquad y^{(3)}(0)=y_3 \qquad y^{(4)}(0)=y_4$$

$$u(0)=u_0 \qquad u^{(1)}(0)=u_1 \qquad u^{(2)}(0)=u_2 \qquad u^{(3)}(0)=u_3$$

(14)

Pro numerický výpočet zadané rovnice 5. řádu je dobré převést na ekvivalentní soustavu pěti lineárních diferenciálních rovnic prvního řádu. Tento převod není vždy stejný. Znamená to, že existuje nekonečně mnoho takových soustav. Jeden z možných převodů je např.:

$$\begin{aligned}
\frac{dx_1}{dt} &= x_2 & x_1(t=0) &= x_{10} \\
\frac{dx_2}{dt} &= x_3 & x_2(t=0) &= x_{20} \\
\frac{dx_3}{dt} &= x_3 & x_3(t=0) &= x_{30} \\
\frac{dx_4}{dt} &= x_4 & x_4(t=0) &= x_{40} \\
\frac{dx_5}{dt} &= -a_0x_1 - a_1x_2 - a_2x_3 - a_3x_4 + u(t-T_d) & x_5(t=0) &= x_{50}
\end{aligned} \qquad (15)$$

$$y(t) = b_0x_1 + b_1x_2 + b_2x_3 + b_3x_4 + b_4x_5$$

Vyjádření (15) vede na složitý přepočet původních počátečních podmínek diferenciální rovnice vyššího řádu na počáteční podmínky soustavy rovnic. Proto je vhodnější použít jinou formu převodu, která má jednodušší výpočet počátečních podmínek soustavy rovnic.

$$\begin{aligned}
\frac{dx_1}{dt} &= -a_4x_1 + x_2 + b_4u(t-T_d) & x_1(t=0) &= x_{10} \\
\frac{dx_2}{dt} &= -a_3x_1 + x_3 + b_3u(t-T_d) & x_2(t=0) &= x_{20} \\
\frac{dx_3}{dt} &= -a_2x_1 + x_4 + b_2u(t-T_d) & x_3(t=0) &= x_{30} \\
\frac{dx_4}{dt} &= -a_1x_1 + x_5 + b_1u(t-T_d) & x_4(t=0) &= x_{40} \\
\frac{dx_5}{dt} &= -a_0x_1 + b_0u(t-T_d) & x_5(t=0) &= x_{50}
\end{aligned} \qquad (16)$$

$$y(t) = x_1$$

4.2 Přepočet počátečních podmínek

Máme-li uskutečněný převod diferenciální rovnice ve tvaru (16), pak přepočet počátečních podmínek původní rovnice na počáteční stav je realizováno jako:

$$y(t) = x_1(t) \Rightarrow y_0 = x_{10}$$

$$\frac{dy}{dt} = \frac{dx_1}{dt} = -a_4 y + x_2 + b_4 u$$

$$\frac{d^2y}{dt^2} = -a_4 \frac{dy}{dt} + \frac{dx_2}{dt} + b_4 \frac{du}{dt} = -a_4 \frac{dy}{dt} - a_3 y + x_3 + b_3 u + b_4 \frac{du}{dt}$$

$$\frac{d^3y}{dt^3} = -a_4 \frac{d^2y}{dt^2} - a_3 \frac{dy}{dt} + \frac{dx_3}{dt} + b_3 \frac{du}{dt} + b_4 \frac{d^2u}{dt^2} = -a_4 \frac{d^2y}{dt^2} - a_3 \frac{dy}{dt} - a_2 y + x_4 + b_2 u + b_3 \frac{du}{dt} + b_4 \frac{d^2u}{dt^2}$$

$$\frac{d^4y}{dt^4} = -a_4 \frac{d^3y}{dt^3} - a_3 \frac{d^2y}{dt^2} - a_2 \frac{dy}{dt} + \frac{dx_4}{dt} + b_2 \frac{du}{dt} + b_3 \frac{d^2u}{dt^2} + b_4 \frac{d^3y}{dt^3} =$$
$$= -a_4 \frac{d^3y}{dt^3} - a_3 \frac{d^2y}{dt^2} - a_2 \frac{dy}{dt} - a_1 y + x_5 + b_1 u + b_2 \frac{du}{dt} + b_3 \frac{d^2u}{dt^2} + b_4 \frac{d^3u}{dt^3}$$

$$\frac{d^5y}{dt^5} = -a_4 \frac{d^4y}{dt^4} - a_3 \frac{d^3y}{dt^3} - a_2 \frac{d^2y}{dt^2} - a_1 \frac{dy}{dt} - a_0 y + b_0 u + b_1 \frac{du}{dt} + b_2 \frac{d^2u}{dt^2} + b_3 \frac{d^3u}{dt^3} + b_4 \frac{d^4u}{dt^4}$$

$$x_{10} = y_0$$

$$x_{20} = y_1 + a_4 y_0 - b_4 u_0$$

$$x_{30} = y_2 + a_4 y_1 + a_3 y_0 - b_3 u_0 - b_4 u_1$$

$$x_{40} = y_3 + a_4 y_2 + a_3 y_1 + a_2 y_0 - b_2 u_0 - b_3 u_1 - b_4 u_2$$

$$x_{50} = y_4 + a_4 y_3 + a_3 y_2 + a_2 y_1 + a_1 y_0 - b_1 u_0 - b_2 u_1 - b_3 u_2 - b_4 u_3 \qquad (17)$$

U lineárních diferenciálních rovnic nižšího řádu se používá stejný postup, jako je popsán výše (17). Pro 4. řád, soustava rovnic pro přepočet počátečních podmínek vypadá následovně:

$$x_{10} = y_0$$

$$x_{20} = y_1 + a_3 y_0 - b_3 u_0$$

$$x_{30} = y_2 + a_3 y_1 + a_2 y_0 - b_2 u_0 - b_3 u_1$$

$$x_{40} = y_3 + a_3 y_2 + a_2 y_1 + a_1 y_0 - b_1 u_0 - b_2 u_1 - b_3 u_2 \qquad 18)$$

U diferenciálních rovnic 3. řádu dostaneme soustavu:

$$x_{10} = y_0$$

$$x_{20} = y_1 + a_2 y_0 - b_2 u_0$$

$$x_{30} = y_2 + a_2 y_1 + a_1 y_0 - b_1 u_0 - b_2 u_1 \qquad (19)$$

Obdobně pak pro rovnici 2. řádu:

$$x_{10} = y_0$$

$$x_{20} = y_1 + a_1 y_0 - b_1 u_0 \qquad (20)$$

4.3 Maticové vyjádření soustavy rovnic

Vhodné je pak soustavu diferenciálních rovnic prvního řádu převést na maticové vyjádření. Převedení se dělá zejména kvůli jednoduššímu zápisu a řešení v programu Matlab.

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t - T_d)$$

$$y(t) = Cx(t) \qquad (21)$$

Stavový popis pro diferenciální rovnice řádu n (13) pak vypadá:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} \quad A = \begin{bmatrix} -a_{n-1} & 1 & 0 & 0 & 0 \\ -a_{n-2} & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -a_1 & 0 & 0 & 0 & 1 \\ -a_0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} b_{n-1} \\ b_{n-2} \\ \vdots \\ b_1 \\ b_0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

Pro vyjádření (16), pak matici *A* a vektory *B, x ,C* můžeme zapsat ve tvaru:

$$x=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \qquad A=\begin{bmatrix} -a_4 & 1 & 0 & 0 & 0 \\ -a_3 & 0 & 1 & 0 & 0 \\ -a_2 & 0 & 0 & 1 & 0 \\ -a_1 & 0 & 0 & 0 & 1 \\ -a_0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad B=\begin{bmatrix} b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} \qquad C=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Uvedeme-li si i stavový popis pro vyjádření, které vede na složitý přepočet počátečních podmínek (15).

$$x=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \qquad A=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 & -a_4 \end{bmatrix} \qquad B=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C=\begin{bmatrix} b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix}$$

4.4 Numerické metody

Diferenciální rovnice lze řešit pomocí numerických metod, díky kterým můžeme vyřešit problémy, které nelze vyřešit přímo, nebo by řešení bylo příliš složité, časově a ekonomicky náročné. Výsledky řešení jsou přibližné. Při numerickém řešení vzniká nepřesnost tzn. chyba. Je udávána pouze jako odhad chyby.

4.5 Řešení metodou Runge-Kutta

Jedná se o numerickou metodu pro výpočet diferenciálních rovnic. Vychází z Taylorova rozvoje a počítá i s členy vyšších řádů. Je nazývána také metodou jednouzlovou, protože k výpočtu hodnoty y_{n+1} stačí znát pouze hodnotu y_n. Potřebné derivace funkce počítá složitější diferenční metodou pomocí dalších pomocných bodů mezi sousedními uzly v síti. Metod Runge-Kutta je více, nejpoužívanější je tzv. klasická metoda 4. řádu, která pro naše potřeby postačí. Diskretizační chyba této metody je $O(h^5)$.

Pro numerický výpočet diferenciální rovnice vyššího řádu lze využít metody Runge-Kutta 4. řádu (RK4). V našem případě použijeme řešitele *ode45* z programu Matlab, která používá modifikovanou metodu RK4. Pro jeden časový krok výpočtu $t = t_o + h$ můžeme pro soustavu diferenciálních rovnic ve tvaru (13) s vektorem hodnot $x(t_0) = x_0$ maticově zapsat jako:

$$
\begin{aligned}
k_1 &= A.x(t_0) + b.u(t_0 - T_d) \\
k_2 &= A.x(t_0 + \frac{1}{2}hk_1) + b.u(t_0 + \frac{1}{2}h - T_d) \\
k_3 &= A.x(t_0 + \frac{1}{2}hk_2) + b.u(t_0 + \frac{1}{2}h - T_d) \\
k_4 &= A.x(t_0 + hk_3) + b.u(t_0 + h - T_d)
\end{aligned} \tag{22}
$$

$$
\begin{aligned}
x(t_0 + h) &= x(t_0) + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\
y(t_0 + h) &= c.x(t_0 + h)
\end{aligned}
$$

Předchozí tvar (22) lze zjednodušit, vzhledem k tomu, že hodnoty vstupní veličiny *u(t)* se během intervalu *h* nemění, budou hodnoty uloženy s konstantním krokem *h* a velikost dopravního zpoždění bude v násobcích kroku *h*.

$$
\begin{aligned}
k_1 &= A.x(t_0) + b.u(t_0 - T_d) \\
k_2 &= A.x(t_0 + \frac{1}{2}hk_1) + b.u(t_0 - T_d) \\
k_3 &= A.x(t_0 + \frac{1}{2}hk_2) + b.u(t_0 - T_d) \\
k_4 &= A.x(t_0 + hk_3) + b.u(t_0 - T_d)
\end{aligned} \tag{23}
$$

$$
\begin{aligned}
x(t_0 + h) &= x(t_0) + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\
y(t_0 + h) &= c.x(t_0 + h)
\end{aligned}
$$

Pomocné hodnoty k_1, k_2 k_3 k_4 představují derivace stavu systému ve speciálních bodech na začátku, konci a uprostřed intervalu $<t_n , t_{n+1}>$.

5. Matlab

5.1 Úvod

Jedná se o prostředí a skriptovací programovací jazyk pro modelování, simulace, paralelní výpočty, vědeckotechnické výpočty, měření a zpracování signálů a návrhy algoritmů. Umožňuje vytváření aplikací včetně grafického uživatelského rozhraní, vykreslování jak 2D tak i 3D grafů funkcí, počítání s maticemi a implementaci algoritmů. Matlab má slabou dynamicko-typovou kontrolu tzn. po deklaraci nemají proměnné určený datový typ.

5.2 Dynamický systém v programu Matlab

Pro práci s dynamickými systémy typu LTI nabízí Matlab několik jednoduše použitelných nástrojů. Snadná použitelnost je dána tím, že všechny formy popisu dynamického systému jsou reprezentovány jedním typem objektu. Jejich velká nevýhoda je použití pro výpočet pouze dynamických systémů s nulovými počátečními podmínkami. Mezi takové nástroje patří například funkce *sys = ss(A, B, C, D)*. Tato funkce slouží pro vytvoření objektu na základě zadaného stavového popisu dynamického systému. Stavový popis byl již výše uveden, tedy pod parametry *A,B,C,D* funkce *ss*, si můžeme představit jednotlivé matice stavového popisu viz. *kapitola 4.3*. Po zavolání funkce, získáme v návratové hodnotě objekt, na který dále můžeme aplikovat další příkazy. Například pro přechodovou charakteristiku příkaz *step(sys)*, pro impulsní charakteristiku příkaz *impulse(sys)* a pro amplitudo-fázovou charakteristiku *nyquist(sys)*.

Obdobně je to i při zadání dynamického systému pomocí obrazového přenosu, jen s tím rozdílem, že musíme použít jinou funkci pro vytvoření objektu. Tato funkce má syntaxi *sys = tf(Bs, As)*. Funkce čeká pod parametrem *Bs*, vektor koeficientů polynomu čitatele přenosu *b, [b_{n-1},..., b_0]* a pod parametrem *As* tentokrát jmenovatel obrazového přenosu *a, [a_{n-1},..., a_0]*. V poslední řadě, máme-li systém vyjádřený pomocí pólů a nul, použijeme pro výpočet v Matlabu příkaz *sys = zpk(N, P, K)*, kde *N* vyjadřuje vektor nul přenosu, *P* vektor pólů přenosu a *K* zesílení, nebo-li podíl $\frac{b_0}{a_0}$.

Převod mezi jednotlivými druhy zadání je znázorněn v tabulce (Tabulka 1).

Command Function	Converting From	Converting To	Input Arguments	Output Arguments
`ss2tf`	state space	transfer function	`[A,B,C,D]`	`[num,den]`
`ss2zp`		zero pole gain		`[z,p,k]`
`tf2ss`	transfer function	state space	`[num.den]`	`[A,B,C,D]`
`tf2zp`		zero pole gain		`[z,p,k]`
`zp2ss`	zero pole gain	state space	`[z,p,k]`	`[A,B,C,D]`
`zp2tf`		transfer function		`[num,den]`

Tabulka 1 - Funkce

Nyní jsme si popsali nástroje, které umí pracovat s dynamickými systémy, ale pouze mají-li nulové počáteční podmínky. Platí-li, že dynamický systém je vyjádřen formou diferenciální rovnice *n*-tého řádu, můžeme v programu Matlab využít funkce *ode45*. Ode45 je funkce, která počítá numerické řešení soustavy obyčejných diferenciálních rovnic. Funkce vrací dva parametry. První je sloupcový vektor časů řešení a druhý je matice, kde každý řádek odpovídá příslušnému času řešení a jednotlivé sloupce jednotlivým proměnným. Volání této funkce je ve tvaru:

```
[t,y]=ode45('fce',[t0,tf],y0);
```

kde *fce*, je uživatelem definovaný m-soubor (textový soubor s příponou *.m*), obsahující funkci, která popisuje soustavu řešených diferenciálních rovnic.. Parametry *t0, tf* udávající začátek respektivně konec intervalu výpočtu. Jak všichni už tušíme, parametr *y0* obsahuje počáteční podmínky rovnice. M-soubor s definovanou funkcí, musí mít speciální tvar a to *function dY = fce(t,y)*, kde *dY* je sloupcový vektor obsahující hodnoty derivací všech proměnných v čase *t* a pro aktuální hodnoty proměnných obsažených ve vektoru *y*.

V případě, že pro výpočet derivací v uživatelské funkci jsou potřebné další informace, je nutné základní syntaxi funkce *ode45* modifikovat, aby funkce *ode45* zajistila předání parametrů. Dostáváme tak např.:

```
[t,x]=ode45(@(t,x) fce(t,x,A,B),[t0,tf],x0);
```

Nyní soubor *fce*, je rozšířen o další parametry *A, B* jedná se o matice stavového popisu. Očekávají se také počáteční podmínky rovnice *x0*, které jsou jak jsme se dozvěděli v předchozích kapitolách přepočítány viz. *kapitola 4.2*. Pro případ dynamického systému popsaného stavovým modelem (10) a konstantními vstupem u=1 můžeme funkce *fce* zapsat např. jako:

```
function dx  = fce(t, x, A, B)
     u=1;
     dx = A*x + B*u;
```

6. Možnosti využití knihoven Matlab v jazyce Java

Možnost využití výpočetních algoritmů Matlabu v cizím programovém jazyku, v našem případě Java, je velice rozsáhlý a záleží jen na našich dovednostech a představě k čemu a jaké knihovny chceme využít. Asi nejjednodušší cestou je zapsat řešení příslušné úlohy ve forme funkce Matlabu, které se předají potřebné parametry a která vrací vypočtené hodnoty. Jedná se o soubory s koncovkou **.m*. Tyto soubory mohou obsahovat definice funkcí, skriptů nebo tříd. Náplň těchto souborů definuje uživatel a sám si určuje, co bude daná funkce provádět za výpočet, jestli bude vykreslovat pouze graf ze zadaných parametrů nebo bude vracet výsledek určitého výpočtu. Příklad takovéto m-funkce, která vrací součet dvou čísel, má tvar:

```
function [s] = soucet(a,b)
     s = a+b;
```

Poté co si uživatel nadefinuje vlastní funkce, musí ještě provést kompilaci skrze Matlab Compile Runtime do jazyka, se kterým chce dále tyto knihovny využívat. V případě jazyka Java se využívá Matlab Compileru spolu s toolboxem MATLAB Builder for Java. Pomocí toho toolboxu se z m-souborů vytváří programové části v jazyce Java.

Pro komunikaci (předávání parametrů funkcí tj. typové konverze) mezi Javou a Matlabem je poskytnuto API, které je implementováno jako balík *MWArray*. Tento balík obsahuje několik tříd, kde každá třída reprezentuje jeden datový typ Matlabu. Mezi takovéto třídy patří:

- *MWNumericArray*
- *MWLogicalArray*
- *MWCharArray*
- *MWCellArray*
- *MWStructArray*

Příklad použití MWNumericArray a kompilace je uvedeno v Praktické části.

Praktická část

1. Praktická část v programu Matlab

1.1 Tvorba m-souborů

Jednou z možností, jak volat knihovny Matlabu, je jak už bylo popsáno v předchozí kapitole, pomocí m-souborů s námi definovanou funkcí, které slouží k určité práci s knihovnami. První, mnou vytvořená funkce, s názvem *„fce“* nese informace o zadané diferenciální rovnici a je tvaru *dx = fce(t,x,p,A,B,k).* Některé parametry jsou stejné jako v další popsané funkci.

Druhá funkce nese název *„Volani“*. Tato funkce má všechny potřebné parametry, mezi které patří volba řádu (*n)*, parametry levé a pravé strany diferenciální rovnice (*a, b*), počáteční podmínky vstupních a výstupních hodnot (*ppU, ppY*), výběr vstupního signálu (*k)* a jeho parametrů (*p),* a časový interval (*min, max)* pro vykreslení průběhu řešení. Funkce *„Volani“* je tvaru *Volani(a, b, ppY, ppU, k, p, min, max, n)* a využívá funkci *ode45* pro výpočet obyčejných diferenciálních rovnic a funkci *„fce“*, která jelikož jde o diferenciální rovnici se speciální pravou stranou, poskytuje diferenciální rovnici v upraveném tvaru.

Vektor *p* nese parametry vstupního signálu. Na prvním místě vektoru je velikost, následuje perioda, poté je tam umístěn fázový posun a nakonec dopravní zpoždění. Dalším důležitým parametrem pro upřesnění je parametr *k.* Jestliže předáme funkci parametr k=1 jedná se o dynamický systém se zpožděním $u(t-T_d)$, je-li k=2 vstupní signál je obdélníkového průběhu, je-li k=3 vstupní signál je sinusového průběhu $u = a*sin(\omega t+\varphi)$ ve všech ostatních případech se jedná o dynamický systém bez zpoždění, kde $u=1$.

Příklad výpočtu diferenciální rovnice 3. řádu, po zavolání funkce *„Volani“* s naplněnými parametry je následující. Nejprve dojde k přepočtu počátečních podmínek původní diferenciální rovnice na počáteční podmínky soustavy rovnic *x0*. Poté se vytvoří stavový model, konkrétně matice vstupu *A* a matice buzení *B.* Nejdůležitější je zavolání funkce *ode45* se souborem s diferenciální rovnicí, dosazenou mezí výpočtu a počátečními podmínkami. K vykreslení grafického výsledku je použit nástroj *plot.*

```
x0 = [ppY(1),
      ppY(2)+a(3)*ppY(1)-b(3)*ppU(1),
      ppY(3)+a(3)*ppY(2)+a(2)*ppY(1)-b(2)*ppU(1)-b(3)*ppU(2)];
B = [b(3);b(2);b(1)];
A = [-a(3) 1 0;-a(2) 0 1;-a(1) 0 0];
[T X] = ode45(@(t,x) fce(t,x,p,A,B,k),[min max], x0);
Y=X(:,1);
plot(T,Y);
```

Pro vykreslení nejen výstupního průběhu *y(t)*, ale také vstupního průběhu *u(t)* jsem využil nástroj *plot* pro více průběhů. Nejdříve si spočítám průběh vstupního průběhu, v tomto případě sinusového signálu, a následně použiji nástroj *plot*. Jak je patrné z náhledu, vstupní průběh bude vykreslen barvou červenou a výstupní průběh barvou modrou. Graf bude dále obsahovat popisek osy x a legendu.

```
u = p(1)*sin((2*pi/p(2))*T+p(3));
plot(T,Y,'b',T,u,'r');
legend('y(t)', 'u(t)');
xlabel('t(sec)')
```

1.2 Kompilace

Jsou dvě varianty kompilace souborů. První z nich je provést kompilaci přímo z příkazové řádky v *Command Window* v Matlabu a nebo mnou využitý nástroj *Deployment Tool*. Ten se spouští buď pomocí zápisu *deploytool* do příkazové řádky, anebo se nachází v menu pod položkou *Desktop*. První krok, který je v tomto nástroji potřebný je vytvořit si nový projekt a určit pro jaký programovací jazyk budeme kompilovat. V mém případě jde o aplikaci založenou na programovacím jazyku Java, proto jsem vybral kompilaci jako *Java Package* a zadal jméno projektu „*MatlabBakalarka“*. Jelikož jde o programovací jazyk Java, kompilace bude probíhat pomocí *Java Builderu*. Dále se už jen pojmenuje nová třída v projektu „*TridaMatlab“*, ke které se přidají námi vytvořené m-soubory funkcí „*fce“* a *„Volani“*. Nakonec se určí cesta kam se má kompilace uložit a provede se *Build*.

Všechny námi zadané názvy, projektu, třídy, funkcí a všechny parametry našich vytvořených funkcí v Matlabu je dobré si pamatovat, protože prostřednictvím těchto názvů budeme v programovacím jazyku Java komunikovat s knihovnami Matlabu.

2. Prostředí Netbeans

2.1 Propojení s knihovnami Matlabu

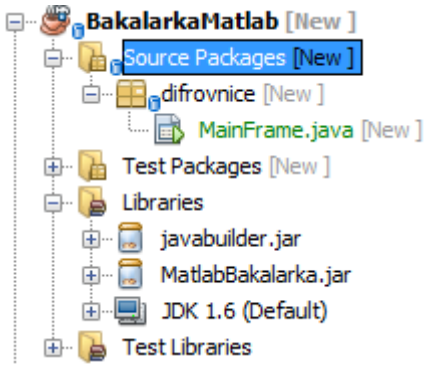
Mezi nejrozšířenější vývojové prostředí pro programovací jazyk Java patří volně šiřitelný software *Netbeans*. Velice oblíbený je zejména kvůli svému přístupu při vytváření grafického rozhraní aplikace. *Netbeans* má již mnoho verzí, které se však liší jen detaily. Mnou využitá verze je *Netbeans 6.7.1*.

Ještě před prvním programováním je dobré dostat knihovny, vytvořené v předešlé části, do našeho projektu v *Netbeans*. Slouží k tomu nástroj *Add Libraries* ve vlastnostech projektu. Přidáme tedy knihovnu *MatlabBakalarka*, to nám však nestačí, ještě potřebujeme přidat knihovnu *Java Builder,* která nám poskytne nástroje pro úpravu dat. Nejčastější cesty k oběma souborům jsou:

```
„<Matlab>\MCR\v714\toolbox\javabuilder\jar\javabuilder.jar“
„<Matlab>\R2010b\ MatlabBakalarka\src\MatlabBakalarka.jar“
```

kde <*Matlab*> znamená cesta k programu Matlab.

Celou stromovou hierarchii vytvořenou v projektu v Netbeans vidíme na následujícím obrázku (Obrázek 8). Projekt obsahuje mimo výše uvedených knihoven také třídu *MainFrame*, jedná se o speciální třídu pro programování grafického rozhraní, která bude popsána v další kapitole.



Obrázek 8 - Projekt v Netbeans

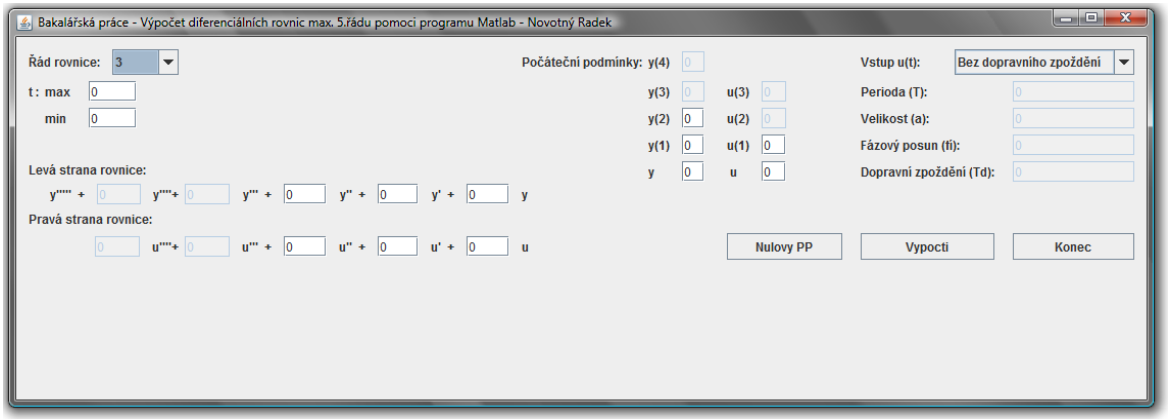
2.2 Návrh uživatelského rozhraní

Abychom mohli využít grafických prvků prostředí Netbeans, nelze vytvořit v projektu pouze třídu, ale tzv. *JFrame form.* Jedná se o již předpřipravenou třídu, která nám umožní využívat grafických nástrojů jazyka Java. Poznáme to z toho, že naše třída dědí od knihovny *javax.swing.JFrame*. Takto vytvořená třída nám umožňuje nejenom samotné programování, ale také přidávání grafických komponentů jako jsou např. tlačítka textová pole, výběrová menu, textové popisky, zaškrtávací políčka atd.

Uživatelské rozhraní se skládá z mnoha prvků, které sou nutné pro zadání všech parametrů diferenciální rovnice. Pro výběr řádu diferenciální rovnice je využit objekt *JComboBox*, který obsahuje rozmezí od 1 do 5. Stejně tak je využit i pro volbu vstupního signálu. Zde je možné volit mezi různými typy vstupního signálu – konstantní vstup bez zpoždění a se zpožděním, a s obdélníkovým nebo sinusovým průběhem na vstupu. Po výběr jedné z možností je k dispozici i zadání parametrů vstupního průběhu od periody až po fázový posun v případě sinusového průběhu. Na levé straně je dále místo pro zadání intervalu výpočtu do připravených objektů *JTextField*. Další textová pole jsou připraveny pro zadání parametrů levé a pravé strany diferenciální rovnice. K zadání počátečních podmínek slouží další textová pole uprostřed aplikace. Chce-li si uživatel ulehčit psaní

s nulovými počátečními podmínkami, stačí kliknout na objekt *JButton Nulovy PP*. Tlačítko, *Vypočti,* slouží k zahájení výpočtu a vykreslení grafického výpočtu. Tlačítko *Konec* obsluhuje uzavření celé aplikace. Po prvním spuštění aplikace je grafické rozhraní nastaveno tak, aby všechny textová pole měla hodnotu nula.

Celá aplikace funguje na základě uzamčení některých nepotřebných částí v případě, kdy nejsou potřeba k vyplnění. Například výběrem 3. řádu diferenciální rovnice, není potřeba vyplňovat parametry jako a_3, a_4, b_3, b_4, proto je aplikace uzavře a nedá se je vyplnit.



Obrázek 9 - Uživatelské rozhraní

2.3 Volání funkcí vytvořených v programu Matlab

Náš balíček *„MatlaBakalarka“* zkompilovaný v Matlabu a přidaný do našeho prostředí obsahuje třídu *„TridaMatlab“*, kterou musíme nejprve inicializovat. Poté můžeme využít funkci *„Volání“* k výpočtu a zobrazení grafického výsledku. Samotná inicializace a zavolání funkce vypadá:

```
TridaMatlab novy = null;
novy = new TridaMatlab();
novy.Volani(…);
```

Základním problémem, který musí řešit každý programátor, který chce nějakým způsobem využívat ve své aplikaci knihovny Matlabu, je transformace dat do podoby, které bude Matlab rozumět. Matlab pracuje zejména s vektory, naneštěstí žádný takový datový typ Java nezná, proto musíme využít metody třídy *MWNumericArray,* kterou nám poskytla přidaná knihovna *Java Builder* v předchozí kapitole. Tato třída má jako parametr konstruktoru pole objektů, ze kterých umí udělat vektor. Máme funkci *„Volání“*, které potřebujeme předat její parametry. Chceme-li funkci naplnit např. parametr *A*, který je vektorem parametrů vstupu diferenciální rovnice, kde $A = [a_0,a_1,a_2,a_3,a_4]$, musíme v tomto případě využít třídy *MWNumericArray* takto:

```
Object[] poleA={a0,a1,a2,a3,a4};
MWNumericArray A = new MWNumericArray(poleA);
```

Třídu *MWNumericArray* využijeme pro všechny parametry funkce, které mají být vektory. Ty parametry, které jsou pouze čísla, necháme typu *float.* Poslední možností, která se musí udělat před zavoláním funkce, je ošetřit ji tzv. výjimkou. To se dělá v prostředí *Netbeans* velice snadno, protože *Netbeans* samo pozná, kde je dobré mít ošetření pomocí výjimek a nabídne sám přidání. Výsledek pak vypadá následovně.

```
try {
    novy.Volani(A,B,ppY,ppU,k,p,min,max,n);
} catch (MWException ex) {}
```

2.4 Vytvoření spustitelné aplikace

Přece jen projekt, který je vyvíjen v prostředí *Netbeans* není moc přenositelný. Uživatel, který by si chtěl danou aplikaci vyzkoušet, by si musel nahrát minimálně toto prostředí. Nevýhodou je, že v tom okamžiku dostává uživatel do rukou upravitelný kód, do kterého může kdykoli a jakkoli zasáhnout. Proto se musí zajistit kompilace do spustitelné aplikace, která poskytne pouze uživatelské prostředí a výpočty na základě zadaných dat. Taková aplikace, vytvořená programovacím jazykem Java, má příponu souboru **.jar*. Aby se prováděla kompilace, musí být ve vlastnostech projektu nastavena volba *Compress JAR file* a volba *Build JAR after Compiling*. Prostředí *Netbeans* pak vytváří spustitelnou aplikaci v okamžiku prvního spuštění a následnou aktualizaci při spouštěních s upraveným kódem. Spustitelný soubor se nachází v adresáři projektu ve složce */dist*. Pouze obsah této složky pak poskytneme uživatelům, kteří chtějí využívat aplikaci. Složka */dist* kromě spustitelné aplikace, která má název projektu a koncovku *jar,* musí obsahovat další složku */lib*. Ta v sobě ukrývá námi přidané knihovny *„MatlabBakalarka.jar“* a *„javabuilder.jar“*.

3. Spouštění na straně uživatele

Vytvořená aplikace je založena na propojení jazyka Java s knihovnami Matlabu. Jak je patrné, ke spuštění aplikace bude potřeba nástroj, který umí pracovat s Javou. Takový nástroj se dnes vyskytuje skoro v každém počítači, jedná se o *Java Virtual Machine* dále jen JVM. Nejlepším řešením je mít v počítači nainstalovanou verzi *Java JDK*.

Každému uživateli, který chce používat tuto aplikaci, nestačí mít k dispozici jen *JVM,* ale také něco co umožní aplikaci komunikovat s knihovnami Matlabu. Uživatel rozhodně nechce instalovat celé prostředí Matlab, které je ke spuštění aplikace zásadní. Musel by vynaložit tisíce korun na pořízení licence a to je v dnešní době přítěží, když může být volně na internetu k dispozici mnoho dalších podobných aplikací.

Společnost MathWorks, která stojí za vývojem softwaru Matlab, přišla s nástrojem zvaným *Matlab Compiler*, který umožňuje vytvářet nejen samostatně spustitelné aplikace **.exe* nebo knihovny pro různé programovací jazyky, ale také umí do našeho počítače dostat knihovny, které jsou pro spouštění aplikace potřebné. Proto pro správný chod

aplikace stačí nainstalovat MCR(*Matlab Compiler Runtime*) pomocí volně šiřitelného instalačního souboru *MCRinstaller.exe*.

Mezi největší neúspěchy při spouštění aplikací, které používají pro určité výpočty právě knihovny Matlab, je použití jiné verze *Matlab Compiler Runtime* než verze *Matlab Compiler* v jaké byla samotná aplikace kompilována. V případě rozdílných verzí to vede k neúspěchu aplikaci spustit. Proto uživatel, který chce tuto aplikaci spustit, musí mít nainstalovaný *MCR verze 7.14*.

4. Samotný průběh výpočtu zadané diferenciální rovnice

4.1 Stanovení zadání

Prvním krokem, který musíme udělat, je výběr nějakého dynamického systému zadaného ve formě diferenciální rovnice nebo jiného přenosu. Musíme zvolit parametry levé a pravé strany diferenciální rovnice a počáteční podmínky celého systému. Dalším důležitým krokem je zvolení vstupního signálu s jeho parametry. Poslední krok je určení intervalu pro, který se má dynamický systém počítat. Nyní přejdeme k vybranému dynamickému systému, jedná se o SISO soustavu 3. řádu neminimálně fázovou, zadanou ve formě přenosu:

$$F(s)=\frac{Y(s)}{U(s)}=\frac{(T_1s+1)(T_2s+1)}{(\tau_1s+1)(\tau_2s+1)(\tau_3s+1)}= \\ =\frac{T_1T_2s^2+(T_1+T_2)s+1}{\tau_1\tau_2\tau_3s^3+(\tau_1\tau_2+\tau_1\tau_3+\tau_2\tau_3)s^2+(\tau_1+\tau_2+\tau_3)s+1} \tag{24}$$

Takovéto soustavě odpovídá přenos s obrazy vstupu *U(s)* a výstupu *Y(s)* (25). Přenos (24) můžeme zobrazit i pomocí Laplaceovi transformace s uvažováním nulových počátečních podmínek (26)

$$\left[\tau_1\tau_2\tau_3s^3+(\tau_1\tau_2+\tau_1\tau_3+\tau_2\tau_3)s^2+(\tau_1+\tau_2+\tau_3)s+1\right]Y(s)=\left[T_1T_2s^2+(T_1+T_2)s+1\right]U(s) \tag{25}$$

$$\tau_1\tau_2\tau_3\left(s^3Y-y''_{t0}\right)+(\tau_1\tau_2+\tau_1\tau_3+\tau_2\tau_3)\left(s^2Y-y'_{t0}\right)+(\tau_1+\tau_2+\tau_3)\left(sY-y_{t0}\right)+Y= \\ =T_1T_2\left(s^2U-u'_{t0}\right)+(T_1+T_2)\left(sU-u_{t0}\right)+U \tag{26}$$

Aplikace však vyžaduje přenos ve formě diferenciální rovnice. Nezbývá než najít diferenciální rovnici, která odpovídá původnímu přenosu (24). Taková diferenciální rovnice je tvaru:

$$y''' + \frac{\tau_1\tau_2 + \tau_1\tau_3 + \tau_2\tau_3}{\tau_1\tau_2\tau_3} y'' + \frac{\tau_1 + \tau_2 + \tau_3}{\tau_1\tau_2\tau_3} y' + \frac{1}{\tau_1\tau_2\tau_3} y = \frac{T_1T_2}{\tau_1\tau_2\tau_3} u'' + \frac{T_1 + T_2}{\tau_1\tau_2\tau_3} u' + \frac{1}{\tau_1\tau_2\tau_3} u$$

Nyní dosadíme parametry levé a pravé strany, za předpokladu, že hodnoty parametrů T_1, T_2, τ_1, τ_2, τ_3 jsou: $T_1 = -1.5$s, $T_2 = 12$s, $\tau_1 = 3$s, $\tau_2 = 3.5$s, $\tau_3 = 4$s. Po dosazení parametrů do rovnice a menších matematických úpravách dostáváme rovnici:

$$y''' + 0{,}869y'' + 0{,}25y' + 0{,}024y = -0{,}428u'' + 0{,}25u' + 0{,}024u \tag{27}$$

Právě jsme dostali základní tvar diferenciální rovnice, ze které lze vyčíst jednotlivé parametry $a_0 \ldots . a_{n-1}$ na levé straně diferenciální rovnice a $b_0 \ldots . b_{n-1}$ na pravé straně diferenciální rovnice:

$$\begin{aligned} a_2 &= 0{,}869 & b_2 &= -0{,}428 \\ a_1 &= 0{,}250 & b_1 &= 0{,}250 \\ a_0 &= 0{,}024 & b_0 &= 0{,}024 \end{aligned} \tag{28}$$

Matice stavového popisu pak jsou:

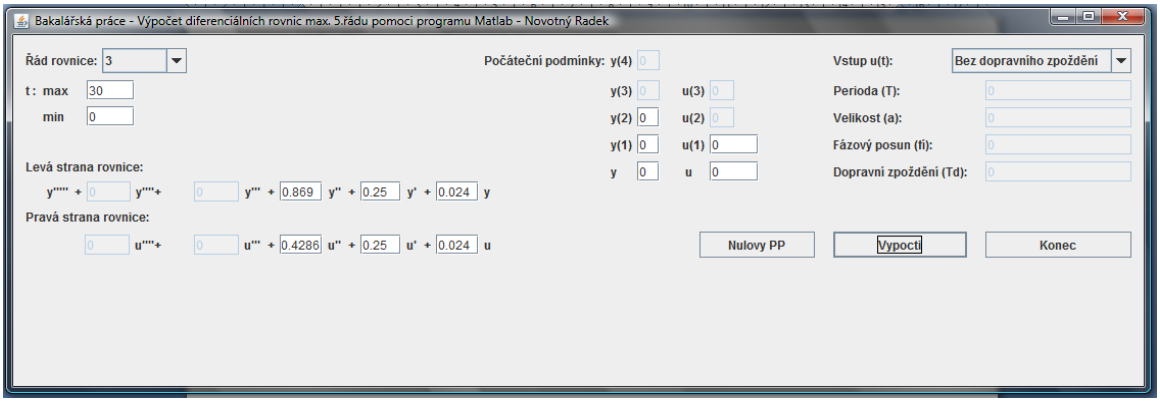
$$A = \begin{bmatrix} -0{,}869 & 1 & 0 \\ -0{,}25 & 0 & 1 \\ -0{,}024 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -0{,}428 \\ 0{,}25 \\ 0{,}024 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Při výpočtu budeme uvažovat nejdříve nulové počáteční podmínky. Poté si ukážeme výsledek výpočtu i pro nenulové počáteční podmínky.

4.2 Nastavení uživatelského rozhraní

Před finálním výpočtem, musíme všechny námi zjištěné parametry, zadaného dynamického systému, zadat do aplikace. Nejprve musíme zadat řád diferenciální rovnice $n = 3$. Dále vyplnit parametry levé a pravé strany z vyjádření výše (28). Předpokládáme také interval výpočtu v rozmezí $t = \langle 0{,}30 \rangle$, dynamický systém bez zpoždění a s nulovými počátečními podmínkami:

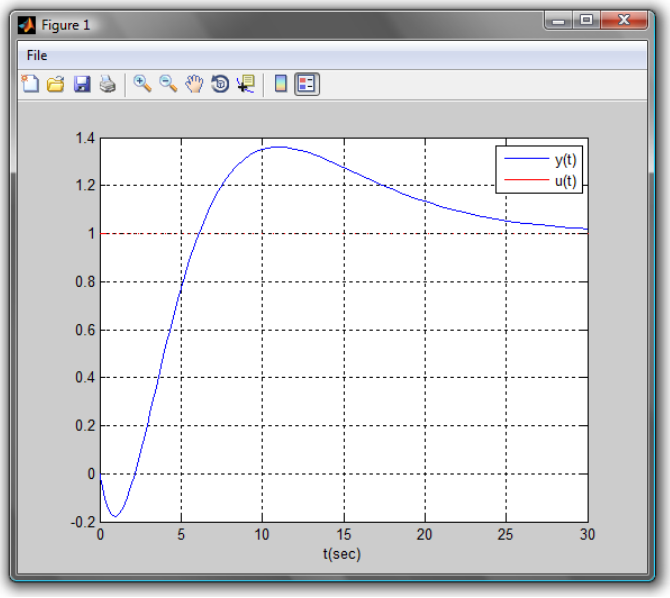
$$y(0) = 0 \qquad y^{(1)}(0) = 0 \qquad y^{(2)}(0) = y0 \qquad u(0) = 0 \qquad u^{(1)}(0) = 0$$



Obrázek 10 - Rozhraní pro výpočet

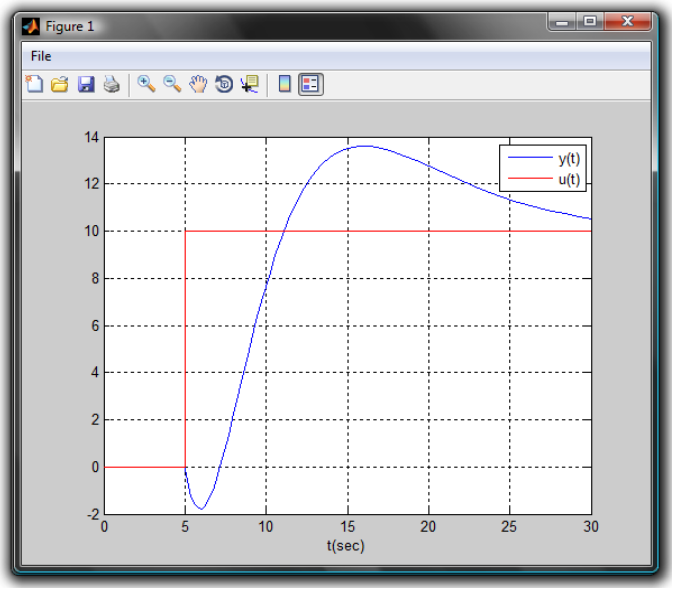
4.3 Výsledky výpočtu

Pro diferenciální rovnici (27), s nulovými počátečními podmínkami a bez dopravního zpoždění, vypadá grafický výsledek:



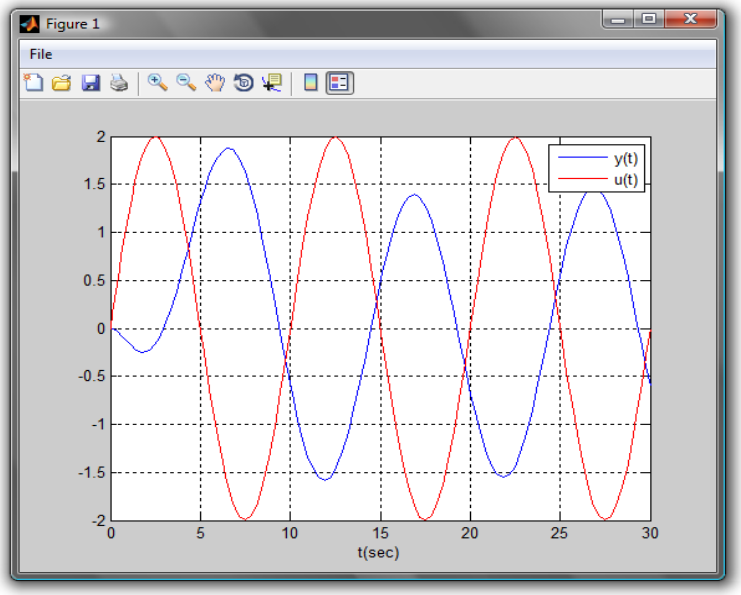
Obrázek 11 - Nulové PP, bez dopravního zpoždění

Nyní předpokládáme dynamický systém, kde na vstupu dojde ke skokové změně s velikostí $a=10$ a s posunem $T_d=5$. Jak je patrné z obrázku (Obrázek 12), dojde k posunutí grafu o zadanou hodnotu, v místech nulového vstupního průběhu je vidět i nulový výstupní průběh.



Obrázek 12 - Nulové PP, skoková změna

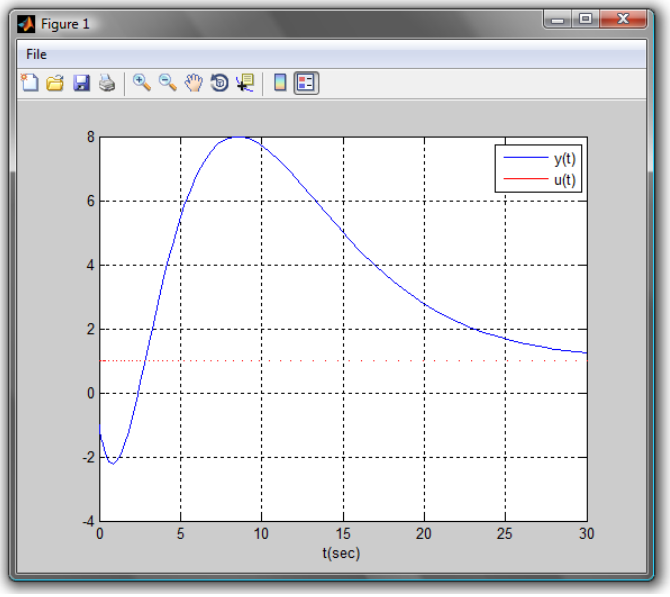
Ukážeme si ještě totožný dynamický systém, ale s volbou sinusového průběhu na vstupu. Sinusový průběh má amplitudu $a = 2$, periodu $T = 10$ a fázový posun φ je nulový (Obrázek 13).



Obrázek 13 - Nulové PP, sinusový průběh

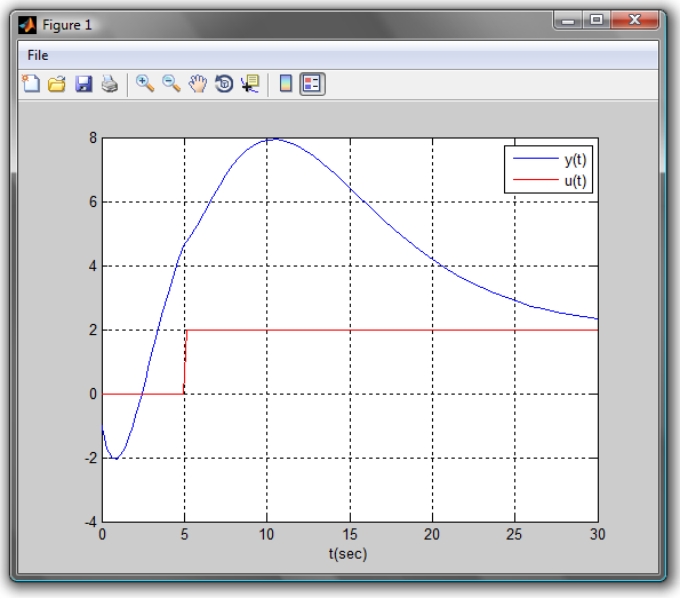
Posledním výpočtem, který tady uvedu, je výpočet dynamického systému (27), který ovšem nebude mít nulové počáteční podmínky. Jinak všechny další parametry budou stejné. Jeho počáteční podmínky nastavíme na hodnoty:

$$y(0) = -1 \qquad y^{(1)}(0) = -2 \qquad y^{(2)}(0) = 3 \qquad u(0) = -2 \qquad u^{(1)}(0) = 1$$

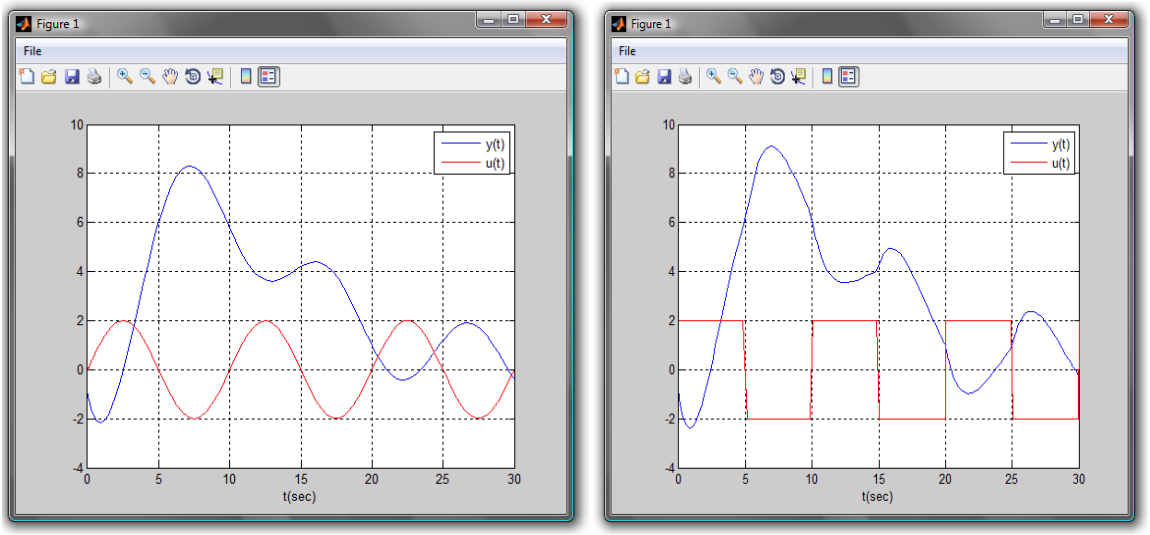


Obrázek 14 - Nenulové PP, bez dopravního zpoždění

Na následujícím průběhu (Obrázek 15), s volbou skokové změny na vstupu, je k vidění, že výstupní hodnota, v době nulového vstupního signálu, není nulová. Jedná se o ovlivnění dynamického systému nenulovými počátečními podmínkami. Další příklady grafických výsledků, při změnách vstupního signálu jsou na obrázku níže (Obrázek 16).



Obrázek 15 - Nenulové PP, skoková změna



Obrázek 16 - Nenulové PP, sinusový průběh, obdélníkový průběhu

Závěr

1. Nástroje a kroky potřebné k vytvoření a spuštění aplikace

Mezi první kroky při vytváření této aplikace je instalace nutných prostředí pro její vývoj. První software nainstalovaný je Java Development Machine (JDK) ver. 1.6.0 nebo vyšší, zároveň s ním také kompatibilní Java Runtime Enviroment (JRE). Tento software naleznete zdarma na stránkách Sun Microsystems nebo na přiloženém CD. Nyní můžeme provést další krok a to instalaci programového prostředí pro jazyk Java a to Netbeans 6.7.1 a prostředí Matlab, v mém případě verzi 2010. Po instalaci všech potřebných nástrojů jsou jednotlivé klíčové činnosti při vytváření aplikace následující:

- Napsání vlastních m-funkcí v programu Matlab (1.1)
- Kompilace m-funkcí pomocí Java Builder (1.2)
- Vytvoření projektu v Netbeans (2.2)
- Naprogramování grafického prostředí v projektu (2.2)
- Přidání zkompilovaných funkcí do projektu (2.1)
- Do programování předávání parametrů výpočtu přidaným m-funkcím (2.3)
- Finální kompilace projektu (2.4)

Na druhé straně pro úspěšné spuštění aplikace jsou jednotlivé kroky:

- Instalace Java Development Machine (JDK)
- Instalace Matlab Compiler Runtime (MCR) ver. 7.14
- Existence adresáře */dist* se spouštěcím souborem (*BakalarkaMatlab.jar*) a složkou */lib* s potřebnými knihovnami (2.4)

Příloha A

K této práci je přiložené CD, na kterém je uložena funkční aplikace, projekt i samotný zdrojový kód programu.

Příloha B

V této příloze, která je přiložena k hlavní práci, jsou k dispozici nejen zdrojové kódy obou funkcí, vytvořených v programu Matlab, ale i nejdůležitější části zdrojového kódu z prostředí Netbeans.

Literatura

BALÁTĚ, J. *Automatické řízení*. Praha : BEN, 2004. 654 s. ISBN 80-7300-148-9.

ŠTECHA, J; HAVLENA, V. *Teorie dynamických systémů*. ČVUT Praha, 1996. 247 s.

KINDLER, E; KŘIVÝ, I. *Simulace a modelování.* Univerzita Ostrava, 2001. 146 s. Učební Text. Univerzita Ostrava.

DUŠEK, F; HONC, D. *Matlab a Simulink - Úvod do používání.* Univerzita Pardubice, 2005. 172 s. ISBN 80-7194-776-8.

HORÁČEK, P. *Systémy a modely,* ČVUT Praha 1999.

ŠVARC, I; ŠEDA, M; VÍTEČKOVÁ, M. *Automatické řízení.* Brno, 2003. 90 s.

HEROUT, P. *Učebnice jazyka Java.* České Budějovice : Kopp, 2001. 349 s. ISBN 80-7232-115-3.

Stavový popis systému. In *Wikipedia* : *the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 11. 9. 2009, last modified on 29. 5. 2011 [cit. 2011-07-18]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Stavov%C3%BD_popis_syst%C3%A9mu>.

MATHWORKS. *Matlab Builder JA 2.* MathWorks, 2010. 292 s.

MATHWORKS. *Matlab 7 External Interfaces.* MathWorks, 2010. 776 s.

MATHWORKS. *Matlab Reference Guide.* MathWorks, 2010.

MathWorks [online]. 1984, 2011 [cit. 2011-07-18]. *Matlab online dokumentac*e. Dostupné z WWW: <http://www.mathworks.com/help/techdoc/>.