

# **Průvodce základy práce v Communications System Toolboxu v programu MATLAB**

Bc. Peter Javor

---

Diplomová práce  
2014



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

\*\*\*nascannované zadání s. 1\*\*\*

\*\*\*nascannované zadání s. 2\*\*\*

### **Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### **Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- Že odevzdaná verze diplomové/bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

## **ABSTRAKT**

Práca prináša a objasňuje najnovšie poznatky z oblasti používania programu MATLAB a Simulink, ako aj Communications System Toolbox. Vo svojej teoretickej časti sa zameriava na teoretický popis i charakteristiku týchto programov, ako aj ich možné použitie v reálnych situáciach. Obsahuje aj stručný manuál k jednotlivým blokom Communications System Toolboxu. Praktická časť popisuje možnosti využitia Communications System Toolbox v predmete Telekomunikačné systémy.

Kľúčová slova: MATLAB, Simulink, Communications System Toolbox, Telekomunikačné systémy

## **ABSTRACT**

The thesis presents and describes the latest knowledges of using MATLAB, Simulink and Communications System Toolbox. In the theoretical part is focused on the theoretical description with intrinsic characteristic of these programs as well as their usage in real situations. The theoretical part contains also short manual of all important blocks of the Communications System Toolbox. The practical part describes possibilities of usage of the Communications System Toolbox for Telecommunications systems subject.

Keywords: MATLAB, Simulink, Communications System Toolbox, Telecommunications systems

PodĎakovanie patrí vedúcemu mojej diplomovej práce Ing. Karlovi Perůtkovi, Ph.D. za odborné rady, za vedenie a za cenné pripomienky. Ďalej by som chcel poĎakovať prof. Ing. Karlovi Vlčkovi, CSc. za cenné rady v oblasti komunikačných systémov.

Veľká vĎaka patrí aj mojím rodičom, ktorí mi boli vždy oporou a podporovali ma pri písaní tejto práce, ako aj počas celého štúdia.

*Remember, happiness doesn't depend upon who you are or what you have.  
It depends solely on what you think.*

*Dale Carnegie*

# OBSAH

ÚVOD.....	9
<b>I TEORETICKÁ ČASŤ.....</b>	<b>10</b>
<b>1 MATLAB .....</b>	<b>11</b>
1.1 CHARAKTERISTIKA PROSTREDIA MATLAB.....	11
1.2 PREHĽAD PREMENNÝCH V MATLABE.....	16
1.2.1 Príklady premenných v MATLABe.....	17
1.3 VYHODNOCOVANIE VÝRAZOV V MATLABE .....	17
1.3.1 Skalárne operácie .....	18
1.3.2 Vektorové operácie .....	19
1.3.3 Maticové operácie .....	23
1.4 VYKRESĽOVANIE GRAFOV V MATLABE .....	26
1.4.1 Základný prehľad práce s grafmi .....	26
1.4.2 Vykresľovanie 2-D grafov .....	26
1.4.3 Vykresľovanie 3-D grafov .....	29
<b>2 SIMULINK .....</b>	<b>31</b>
2.1 HLAVNÉ SÚČASTI SIMULINKU .....	32
2.2 TOOLBOXY V SIMULINKU .....	32
2.2.1 Najpoužívanějšíe toolboxy pre prácu v MATLABe a Simulinku.....	32
2.3 PRÍKLAD SIMULÁCIE V SIMULINKU.....	34
2.3.1 Simulácia poskakujúcej loptičky.....	34
<b>3 COMMUNICATIONS SYSTEM TOOLBOX .....</b>	<b>37</b>
3.1 NÁVRH SYSTÉMU, CHARAKTERIZÁCIA A VIZUALIZÁCIA .....	37
3.2 HLAVNÉ SÚČASTI COMMUNICATIONS SYSTEM TOOLBOXU .....	37
3.3 POPIS ZÁKLADNÝCH BLOKOV COMMUNICATION SYSTEM TOOLBOXU.....	38
3.3.1 Knižnica Komunikačných Kanálov .....	38
3.3.2 Knižnica Kódovania Zdroja .....	39
3.3.3 Knižnica Filtrovania.....	39
3.3.4 Knižnica Comm Sink .....	40
3.3.5 Knižnica Comm Sources.....	41
3.3.6 Knižnica Equalizers .....	42
3.3.7 Knižnica Error Detection and Correction .....	44
3.3.8 Knižnica Interleaving .....	46
3.3.9 Multiple-Input Multiple-Output (MIMO).....	48
3.3.10 Analógová a Digitálna modulácia.....	49
3.3.11 RF poruchy.....	50
3.3.12 Opravy RF porúch.....	50
3.3.13 Sekvenčné operácie .....	50
3.3.14 Synchronizácia .....	51
3.3.15 Utility Blocks .....	53
<b>II PRAKTICKÁ ČASŤ .....</b>	<b>55</b>
<b>4 SIMULAČNÝ MODEL BLOKOVÉHO KÓDOVANIA .....</b>	<b>56</b>

4.1	REED-SOLOMONOV MODEL KÓDOVANIA PRIRODZENÝCH ČÍSEL .....	56
4.2	SIMULAČNÝ MODEL PRENOSU POMOCOU HAMMINGOVHO KÓDOVANIA .....	58
<b>ZÁVER .....</b>		<b>62</b>
<b>SEZNAM POUŽITÉ LITERATURY .....</b>		<b>64</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>		<b>66</b>
<b>SEZNAM OBRÁZKŮ .....</b>		<b>68</b>
<b>SEZNAM PŘÍLOH.....</b>		<b>70</b>

## ÚVOD

MATLAB je velmi populární skriptovací jazyk pre číselné výpočty používaný vedcami, technikami a študentmi na celom svete. Tento programovací balíček slúži na rýchle a prehľadné kalkulácie, vizualizácie a programovanie určené pre široké spektrum použitia. MATLAB využíva viac ako milión rôznych technikov a vedcov na celom svete. Obsahuje doslova stovky rozličných funkcií a množstvo toolboxov navrhnutých pre špeciálne výskumné disciplíny vrátane štatistiky, optimalizácie, riešenia parciálnych diferenciálnych rovníc a simulácie a analýzu dát.

Práve na spomínané simulácie, modelovanie a analýzu dát je určený softwarový balíček Simulink. Veľmi dobre je prepojený práve s MATLABom a všetkým jeho funkciám, čo umožňuje analýzu a ďalšie spracovanie nasimulovaných dát priamo zo Simulinku. Významná vlastnosť Simulinku je možnosť interaktívne meniť parametre jednotlivých nástrojov priamo za behu simulácie a sledovať zmeny výstupnej charakteristiky dát. Tento program premení každý počítač vo veľké laboratórium pre analýzu najrôznejších vlastností, ako odpor vzduchu lietadla, simulácia výrobných procesov či ekonomický plán podniku na nasledujúci rok.

Ďalší významný prvok pre analýzu, návrh a simuláciu prenosu dát je Communications System Toolbox. Tento toolbox umožňuje zdrojové kódovanie, kanálové kódovanie, prekladanie, moduláciu, porovnávanie dát a synchronizáciu. Taktiež obsahuje koncepty pre analógovo-digitálnu konverziu a kompresiu dát.

Cieľom tejto práce je charakterizovať MATLAB a Simulink, rozobrať ich prvky a nastavenia a uviesť najdôležitejšie nástroje pre prácu s MATLABom. Zvlášť klásť dôraz na Communications System Toolbox a popísať jednotlivé jeho bloky. Využitím všetkých týchto poznatkov by som mal byť schopný navrhnuť konkrétne simulácie určenú pre výučbu predmetu Telekomunikačné systémy.

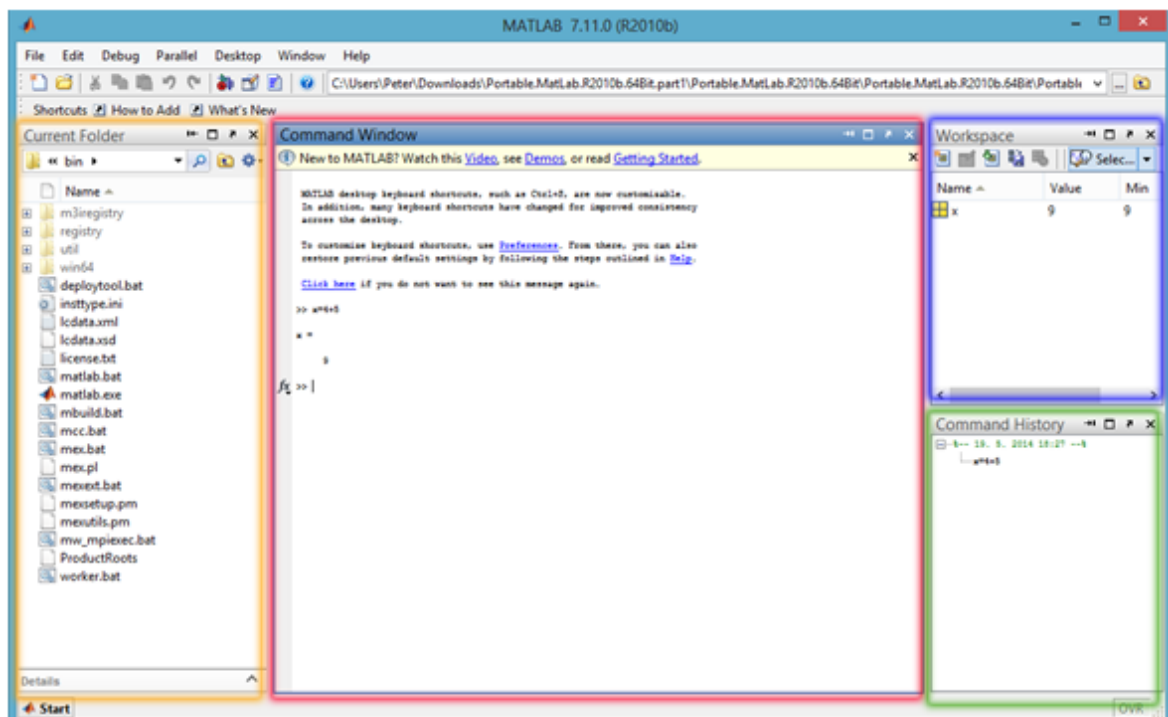
## **I. TEORETICKÁ ČASŤ**

## 1 MATLAB

Názov programu tvorí skratka MATrix LABoratory. Tento populárny dynamický skriptovací jazyk sa používa od roku 1970 a ako komerčný produkt spoločnosti MathWorks od roku 1984. Počet užívateľov presiahol v roku 2004 jeden milión. Jazyk MATLAB sa vyvíjal v priebehu niekoľkých rokov. Postupne sa zavádzali ďalšie a ďalšie funkcie. Okrem jeho použitia pri výpočtoch lineárnej algebry, algebrických a diferenciálnych rovníc, MATLAB obsahuje aj prepracované grafické nástroje pre zobrazovanie zaujímavých 2D i 3D grafov. Programovanie pomocou MATLABu je oveľa rýchlejšie než pomocou vyššieho programovacieho jazyku. [1]

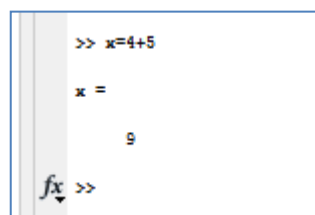
### 1.1 Charakteristika prostredia MATLAB

Po spustení prostredia MATLAB môžeme pozorovať niekoľko okien. Meniť ich veľkosť či usporiadanie môžeme podľa vlastných potrieb či uváženia. Môžeme ich dokonca zatvárať, otvárať, či pripínať k hlavnej ploche. Pre lepšiu navigáciu medzi jednotlivými oknami slúži menu Window v hornom paneli. Ich rozloženie sa postupne menilo aj v závislosti na verzii. Štandardné prostredie verzie R2010b vyzerá asi takto:



Obr. 1: Pracovné prostredie programu MATLAB R2010b

Hlavné okno sa nazýva príkazové (**Command Window**) je orientované v prostrednej časti obrazovky (ružová farba). Slúži na zadávanie príkazov, dát a pre zobrazenie výsledkov. Font písma môžeme samozrejme meniť podľa potreby. Môžeme v ňom vidieť príkazový riadok začínajúci `>>`. Ak vidíme tento príkazový riadok, MATLAB je pripravený na zadávanie príkazov, v opačnom prípade je zaneprázdnený predchádzajúcou úlohou. Predchádzajúci príkaz môžeme zobrazit' šípku hore. Urýchľujeme tak písanie a predchádzame chybám. Pre ukážku som do príkazového okna napísal operáciu  $x=4+5$ . Stiskom tlačidla Enter sa pod daným výrazom zobrazí výsledok.



```
>> x=4+5
x =
    9
fx >>
```

Obr. 2: Ukážkový matematický príkaz

V pravom hornom rohu si môžeme všimnúť okno **Workspace** (modré označenie). Zobrazuje nám všetky premenné ktoré sme doteraz v našom programe použili ako aj nejaké základné informácie o každej z nich. Ikony navrchu tohto okna umožňujú základné operácie s premennými. Dvojklikom na jednotlivé premenné sa nám otvorí nové okno Variable Editor. Každá premenná môže byť uložená pre ďalšie projekty pomocou **File**→**Save**

**Workspace As.** Na Obr. 1 si môžeme všimnúť už spomínanú premennú **x**. Hodnota tejto premennej zostane uložená a môžeme s ňou pracovať v ďalšom vývoji programu. Pri prvom spustení MATLABu bude toto okno pochopiteľne prázdne. K premenných sa ešte dostaneme neskôr.

Spomínaný **Variable Editor** otvára premenné v "excelovskom" formáte vo forme tabuľky. Môžeme tak skontrolovať ktorý riadok/stĺpec obsahuje akú hodnotu. Samozrejmosťou je možnosť meniť akékoľvek hodnoty.

V pravom spodnom rohu nájdeme zeleno zvýraznené okno **Command History**. Nájdeme tam chronologicky usporiadaný zoznam všetkých použitých príkazov v MATLABe. Vhodné pre vyhľadávanie a znovu použitie niektorých starších príkazov. Vo verzii R2014a je toto okno vyriešené veľmi prakticky a to formou vyskakovacieho okna pri stisku šípky hore, teda zobrazenia posledného príkazu. Vyskočí nám zoznam všetkých použitých príkazov, takže si môžeme listovať a vybrať práve ten ktorý potrebujeme. Pri ďalšom stlačení

Ľubovoľnej klávesy toto okno zase zmizne, takže nezaberá zbytočné miesto na obrazovke. Samozrejme nie je problém nastaviť toto okno aby bolo pevnou súčasťou pracovného prostredia.

Okno **Current Folder** nám zobrazuje obsah a umiestnenie aktuálneho pracovného adresára, s ktorým momentálne pracujeme. V tomto adresári nájdeme všetky súbory projektu, ktoré sme doteraz uložili. Všetky skripty ktoré chceme použiť musia byť uložené práve v tomto adresári. Inak sa nespustia. Pre prepínanie medzi adresármi nám slúži menu v hornej časti tohto okna. Nájdeme ho v ľavej časti obrazovky zvýraznené žltou farbou.

Veľmi dôležité a často používané okno v MATLABe je **Editor**. Slúži na editáciu m-súborov. Teda súborov ktoré obsahujú príkazy a funkcie ktoré sme preddefinovali. Otvárame ho pomocou príkazu `edit` v Command Window. Viaceré takéto súbory sa kvôli prehľadnosti štandardne otvárajú vo forme záložiek.

Ďalšie okno s ktorým prideme so styku je **Figure Editor**. Je to zvlášť okno určené pre vykresľovanie schém, grafov a podobne. Môžeme v ňom taktiež meniť mierku, farbu a formát vykresľovania.

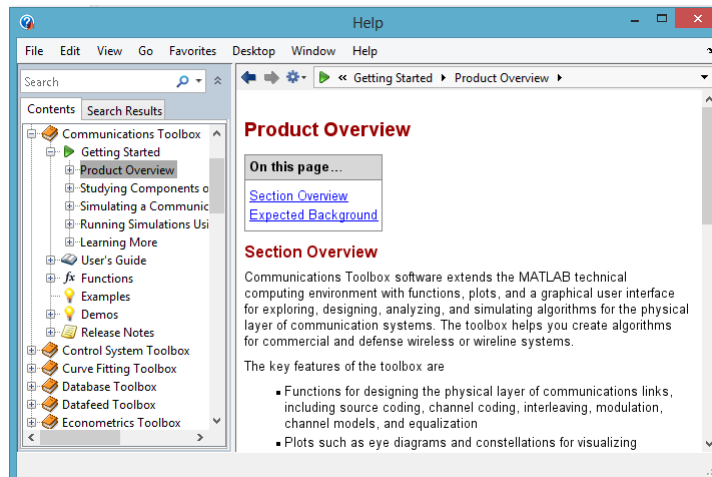
MATLAB má veľmi prepracovanú a prehľadnú nápovedu. Spúšťa sa v novom okne **Help**. Nájdeme v nej všetky informácie ktoré pri práci potrebujeme vedieť. Svojím vzhľadom a správaním pripomína webový prehliadač, pri ktorom nechýba možnosť vyhľadávania požadovaného problému. Otvoríme ju kliknutím na **Help** → **Product Help** v hornom menu.

Pred začatím práce v MATLABe určite odporúčam si túto nápovedu prejsť. Veľmi užitočné sú aj Demo videá pri každej kapitole nápovedy.

Ďalší druh nápovedy získame priamo z príkazového riadka pomocou príkazu

```
>> help nazov_funkcie.
```

Zobrazí nám popis a správne použitie daného príkazu i podobné príkazy, ktoré by nás mohli zaujímať. [2]



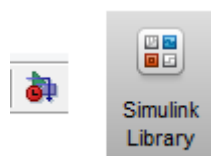
Obr. 3 Okno nápovedy v MATLABe

Pre zmenu usporiadania pracovných okien v MATLABe klikneme v hornom menu na odkaz **Desktop** → **Desktop Layout**. Vo verzii R2014a pomocou menu **Layout** v hlavnej záložke **Home**. Máme na výber z niekoľkých preddefinovaných štýlov.

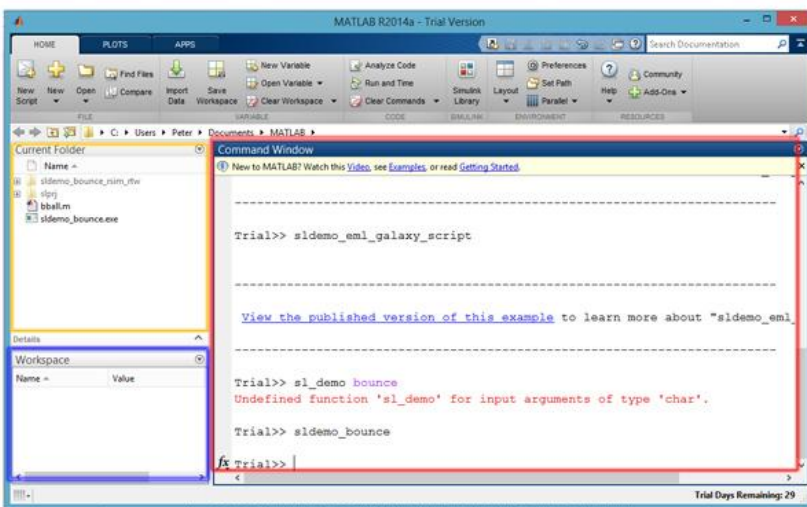
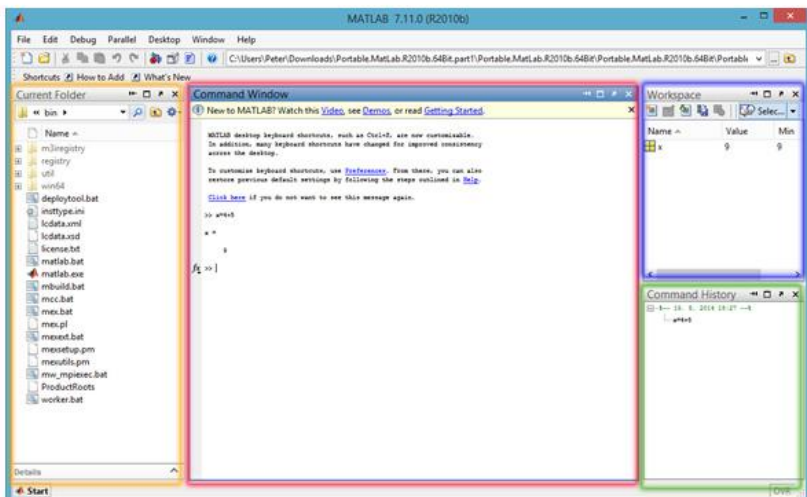
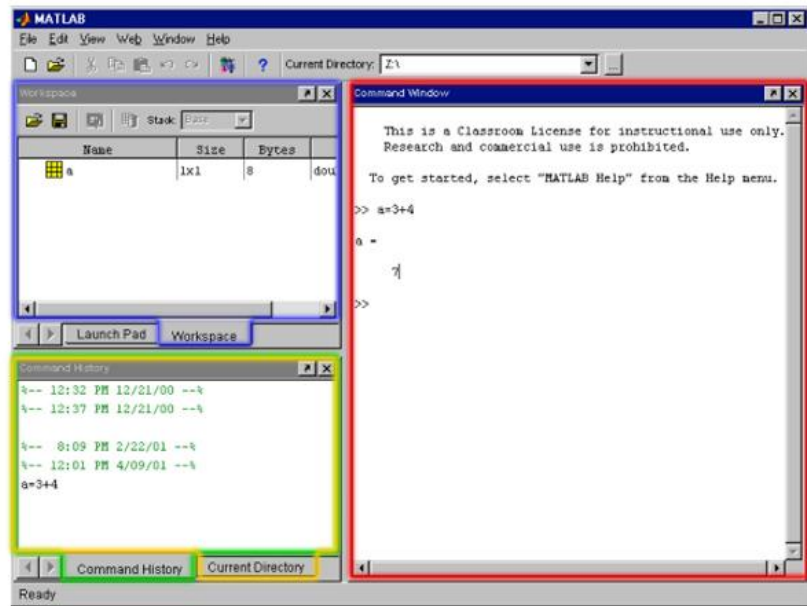
Pre ukážku vývoja pracovného prostredia som pripojil ukážku MATLABu verzie 6x z roku 2001 a najnovšej verzie R2014a. Pre sprehľadnenie som farebne zvýraznil rovnaké funkčné okná.

Na prvý pohľad je vidieť že verzia 2014b priniesla značne prepracované grafické prostredie. Oproti svojim predchodcom neobsahuje funkčné tlačidlo **Štart**. Celé menu je umiestnené v hornej časti obrazovky a rozdelené je do troch záložiek.

Zmenou dizajnu prešli aj všetky ikony, ktoré teraz omnoho lepšie zapadajú do celkového grafického konceptu. Práve k ikonám pribudli aj popisy, takže nie je až taký problém zvyknúť si na nové prostredie. Pre porovnanie uvediem ikonu Simulinku vo verziách R2010b a R2014a:



Obr. 4 Ikona Simulinku v prostredí R2010b a 2014a



Obr. 5 Porovnanie rozloženia okien verzie 6x, R2010b a R2014a

## 1.2 Prehľad premenných v MATLABe

Táto kapitola popisuje základné operácie spojené s vytváraním a používaním premenných v MATLABe.

Premenná sa v MATLABe vytvorí, ak sa objaví naľavo od znaku "rovná sa". Na pravej strane tohto výrazu určujeme hodnotu, ktorú táto premenná nadobúda. Premenné môžeme definovať kedykoľvek ich potrebujeme využiť, nemusia byť dopredu preddefinované. Každá premenná musí začínať písmenom. Ďalej sa v názve môžu vyskytovať aj čísla. Názov premenných môže mať až 31 znakov, nesmie však obsahovať medzeru.

```
>> premenna=hodnota
```

V skutočnosti platí, že ak chceme uložiť do premennej textový reťazec, musíme ho dať do apostrofov:

```
>> premenna='hodnota'
```

V takomto prípade sa do premennej "premenna" uloží textový reťazec "hodnota".

Všetky číselné premenné v MATLABe majú tvar matice a zodpovedajú dvojrozmernému poľu čísel. MATLAB presne dodržiava všetky pravidlá lineárnej algebry, čo umožňuje presné úkony medzi maticami, vektormi a skalármi.

V predošlej kapitole uvedená premenná  $x$  nadobudla hodnotu **9**. Môžeme teda hovoriť o skalárnej premennej w jedným riadkom a jedným stĺpcom.

Ak chceme do premennej uložiť vektor, uloží sa v maticovom tvare do práve jedného riadka, alebo práve jedného stĺpca. Rozdiel medzi riadkovým a stĺpcovým vektorom je zásadný. Pri práci s takýmito vektormi musíme správne chápať algebraické operácie medzi skalárom, vektorom a maticou.

Pri premenných vo forme matice sa dané hodnoty ukladajú do viacerých riadkov i stĺpcov.

Numerické prvky matice môžu byť reálne, alebo komplexné čísla. Typ čísla sa definuje pri vytváraní premennej.

Pri práci s MATLABom musíme mať na pamäti že MATLAB je "**case sensitive**". Znamená to že premenná  $x$  sa nerovná premennej  $X$ . Musíme teda dávať pozor na klávesu Caps Lock. [3]

Presné hodnoty akejkoľvek premennej máme možnosť vidieť vo Variable Editore.

### 1.2.1 Príklady premenných v MATLABe

Príklady platných premenných v MATLABe:

```
>> a=5  
>> rychlost=120  
>> meno= 'Jan Prochazka'  
>> b=cos(a)  
>> A=[1 2 3 4 5]  
>> B=[1 2 3 ; 4 5 6 ; 7 8 9]
```

Hodnoty premenných sa nám automaticky vypisujú na ďalší riadok pod daný príkaz.

Ak však nechceme hodnoty vypisovať, pri ich definovaní pridáme na koniec príkazu bodkočiarku ";".

```
>> d=sin(b);
```

Teraz uvediem pár príkladov neplatných premenných:

```
>> cislo jedna = 1  
>> 2pretekar = 'Bolt'  
>> auto = Porshe
```

### 1.3 Vyhodnocovanie výrazov v MATLABe

Operátory v MATLABe sú veľmi podobné klasickým matematickým, takže práca s nimi nerobí žiadny problém.

n, funcia alebo konštanta	MATLAB príkaz	n, funcia alebo konštanta	MATLAB príkaz
plus	+	sin x	sin(x)
mínus	-	cos x	cos(x)
krát	*	tan x	tan(x)
deleno	/	cot x	cot(x)
x  (absolútna hodnota x)	abs(x)	arcsin x	asin(x)
square root of x	sqrt(x)	arccos x	acos(x)
e <sup>x</sup>	exp(x)	arctan x	atan(x)
ln x (prirodzený logaritmus)	log(x)	arccot x	acot(x)
log <sub>10</sub> x (logaritmus o základe 10)	log10(x)	n! (n faktoriál)	gamma(n+1)
e (2.71828...)	exp(1)	i (imaginárna zložka, sqrt(-1))	i
π (3.14159265...)	pi		

Tabuľka 1 Základné matematické operácie v MATLABe

### 1.3.1 Skalárne operácie

Riešenie skalárnych rovníc je veľmi jednoduché, v tomto prípade slúži MATLAB ako prehľadná kalkulačka. Využiť môžeme všetky matematické formule. Medzery medzi jednotlivými operátormi nepredstavujú žiadny znak. Nasledujúce 2 formule majú ten istý výsledok:

```
>> 1+3 * 2-1 / 2*4
>> 1 + 3 * 2 - 1 / 2 * 4
```

Ďalej uvediem príklad goniometrickej rovnice:

```
>> x = pi^2;
>> X = sqrt(x);
>> x/X
ans =
3.1416
```

Pre každý výsledok môžeme nastaviť požadovaný formát, v ktorom chceme daný výsledok zobraziť. Máme na výber z troch možností:

- long - krátky (5 miestny)
- short - dlhý (15 miestny)

- rat - racionálny (zlomok).

Pre prehľadnosť ich aplikujem na výsledok predchádzajúceho príkladu.

```
>> format long; ans
```

```
ans =
```

```
3.141592653589793
```

```
>> format rat; ans
```

```
ans =
```

```
355/113
```

```
>> format short; ans
```

```
ans =
```

```
3.1416
```

### 1.3.2 Vektorové operácie

Spolu s maticovými operáciami patria medzi najvýznamnejšie operácie v MATLABe. Pracujeme s nimi ako s objektmi.

Čísla do vektoru môžeme priamo vypisovať. Ak chceme vektor 5 čísel od 1 do 5 tak do MATLABu zadáme:

```
>> r = [1 2 3 4 5];
```

Alebo môžeme požiť dvojbodku ":" kedy stačí zadať prvý a posledný prvok a rozstupy medzi nimi. Ak nezadáme parameter rozstupov, MATLAB tento ho považuje za 1. Predchádzajúci výraz by teda vyzeral:

```
>> r = 1:5;
```

Príklad s vektora s uvedeným prostredným parametrom:

```
>> s = 2:2:10
```

```
s =
```

```
2 4 6 8 10
```

Ďalší spôsob vytvorenia vektoru je pomocou funkcie **linspace**.

Zapisuje sa vo formáte linspace(začiatok, koniec, n-prvkov):

```
>> x=linspace(1,6,5)
```

```
x =
    1.0000    2.2500    3.5000    4.7500    6.0000
```

V prípade že by sme neuviedli posledný parameter - počet prvkov, MATLAB by vytvoril vektor o 100 prvkoch medzi začiatočnou a konečnou hodnotou.

Násobenie vektora skalárom je veľmi jednoduché. Keďže sa vektor správa ako objekt, stačí názov vektora vynásobiť ľubovoľným číslom a každý jeden člen  $x$  - členného vektora sa vynásobí práve týmto číslom.

Pri násobení vektora vektorom platí pravidlo násobenia matice maticou, čo však pri vektoroch nevieme zaručiť, keďže majú rovnakú dimenziu. Ak však chceme vynásobiť jednotlivé elementy 2 vektorov, použijeme príkaz `".*"`

```
>> a = [1 2 3];
>> b = [4 5 6];
>> a.*b
ans =
    4   10   18
```

MATLAB previedol operáciu  $[a_1b_1 \ a_2b_2 \ a_3b_3]$ .

Obidva vektory ale musia mať rovnakú veľkosť. Tú zistíme pomocou príkazu `"size"`. Tento príkaz nám vráti 2 čísla a to počet riadkov a počet stĺpcov.

```
>> size (s)
ans =
    1     5
```

Znamená to teda že vektor "s" má jeden riadok a 5 stĺpcov.

podrobnejšie informácie zistíme pomocou príkazu `"whos"`. Zadáva sa bez akéhokoľvek parametru. Jeho výstupom je prehľadná tabuľka všetkých použitých premenných, ich veľkosť, počet Bytov a trieda.

Tie isté pravidlá platia aj pri delení jednotlivých členov vektoru medzi sebou. Teda podľa zápisu  $\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$ .

Pre sprehládnenie doplním aj formátovanie výsledku do zlomku:

```
>> a = [1 2 3];  
>> b = [4 5 6];  
>> a./b;  
>> format rat; ans  
ans =  
1/4      2/5      1/2
```

Praktické využitie vektorov:

V nasledujúcom príklade si ukážeme ako pomocou MATLABu vypočítame goniometrickú rovnicu:

$$y(x) = \sin\left(\frac{x \cos x}{x^2 + 3x + 1}\right), \text{ pre hodnoty } x \text{ od } 1 \text{ do } 3 \text{ po krokoch } 0,02.$$

Do MATLABu postupne zadáme nasledujúce príkazy:

```
>> x= 1:0.02:3;  
>> f = x.*cos(x);  
>> g = x.^2+3*x+1;  
>> y = sin(f./g);
```

Pre prácu s polynómami nájdeme v MATLABe veľké množstvo príkazov, stačí si prezrieť nápovedu. Asi najviac užitočný je príkaz **roots**. Slúži na hľadanie koreňov polynomiálnej rovnice. Jeho tvar je veľmi jednoduchý.

Použitie: Budeme hľadať korene nasledujúcej polynomiálnej rovnice:

$$y = x^3 - 3x^2 + 2x$$

```
>> c = [1 -3 2 0];  
>> r = roots(c)  
r =  
0  
2  
1
```

V prípade, že chceme vypísať niektorý z prvkov vektoru, stačí za vektor napísať v zátvorke poradie požadovaného prvku. Použijem vektor z predchádzajúceho vektoru:

```
>> c(2)
```

```
ans =  
-3
```

Toto platí pri jednotlivých prvkoch. Ak chceme vypísať viacero prvkov z daného vektora, musíme pridať hranarú zátvorku, kde zadáme poradie prvkov, ktoré chceme vypísať:

```
>> c([3,4])  
  
ans =  
2 0
```

Použiť môžeme aj výraz pomocou dvojbodky, pri väčšom počte vypisovaných prvkov je toto riešenie rýchlejšie. Dokonca si môžeme z indexov vytvoriť premennú a potom sa odkazovať už len na ňu. Používa sa pri viacnásobnom vypisovaní tých istých prvkov:

```
>> c(3 : 4)  
  
ans =  
2 0  
  
>> i = 3:4;  
  
>> c(i)  
  
ans =  
2 0
```

Ak chceme vypísať viac prvkov bez toho aby sme každý zvlášť zadávali, môžeme použiť aj parameter "**end**". Môžeme takto zadať, od ktorého prvku máme začať vypisovať všetky ostatné až do konca. Príkaz je však možné upraviť podľa vlastnej potreby. Na ukážku vypíšeme niekoľko možností:

- **y = x(1:end);** - Vypis celého vektoru
- **y = x(1:end/2);** - Vypis prvej polovice prvkov
- **y = x(2:2:end);** - Vypis každého druhého prvku vektoru od druhého do konca
- **y = x(1:end-1);** - Vypis všetkých prvkov okrem posledného

### 1.3.3 Maticové operácie

Matice vytvárame veľmi podobne ako vektory, len pridávame druhý rozmer.

Hodnoty udávame tak isto v hranatých zátvorkách, oddelené medzerou alebo čiarkou.

Jednotlivé riadky matice ale oddeľujeme bodkočiarkou. Musíme dbať na to, aby sme mali vo všetkých riadkoch rovnaký počet znakov. Pre príklad uvediem maticu **M** o troch riadkoch a troch stĺpcoch:

```
>> M=[1 2 3 ; 4 5 6 ; 7 8 9]
```

```
M =
```

```

1     2     3
4     5     6
7     8     9
```

Matice môžeme vytvárať spojovaním z takzvaných submatic:

```
>> b = [M 10*M; -M [1 0 0; 0 1 0; 0 0 1]]
```

```
b =
```

```

1     2     3    10    20    30
4     5     6    40    50    60
7     8     9    70    80    90
-1    -2    -3     1     0     0
-4    -5    -6     0     1     0
-7    -8    -9     0     0     1
```

Ďalší spôsob ako spojovať matice je pomocou funkcie "**repmat**". Táto funkcia nám zreplicuje zadanú maticu do požadovaných rozmerov ako dlaždice. Zadáva sa v tvare: `repmat(matica, pocet_vertikalnych_opakovani, pocet_horizontalnych_opakovani)` Uvediem príklad:

```
>> a = [1 2 ; 3 4]
```

```
a =
```

```

1     2
3     4
```

```
>> repmat(a,1,2)
```

```
ans =
```

```

1     2     1     2
3     4     3     4
```

Ak potrebujeme vygenerovať ľubovoľnú maticu o **m** riadkoch a **n** stĺpcoch, MATLAB nám poskytuje niekoľko jednoduchých spôsobov ako ju vytvoriť:

- `>> zeros(m,n)` ; - matica naplnená nulami
- `>> ones(m,n)` ; - matica naplnená jednotkami
- `>> rand(m,n)` ; - matica naplnená náhodnými kladnými číslami
- `>> randn(m,n)` ; - matica naplnená náhodnými číslami

- `>> eye(m,n)`; - jednotková matice

Invertovanú maticu vytvoríme pomocou príkazu "inv(a)", alebo  $a^{-1}$ .

Pri maticiach indexujeme na jednotlivé prvky tým istým spôsobom ako pri vektoroch. Musíme ale zadať dve dimenzie miesto jednej. Pre lepšie pochopenie uvediem príklad 3-rozmernej matice:

```
>> a = [1 2 3 ; 4 5 6 ; 7 8 9]
```

```
a =
```

```
     1     2     3
     4     5     6
     7     8     9
```

```
>> a(2,3)
```

```
ans =
```

```
     6
```

```
>> a(2,2:3)
```

```
ans =
```

```
     5     6
```

```
>> a(3,:)
```

```
ans =
```

```
     7     8     9
```

```
>> a(:,1)
```

```
ans =
```

```
     1
```

```
     4
```

```
     7
```

Ak potrebujeme odstrániť niektorý riadok alebo stĺpec z vopred vytvorenej matice, priradíme mu prázdnu maticu. V príklade vychádzam z matice **a** z predošlej ukážky:

```
>> a(:,1)=[]
```

```
a =
```

```
2 3
```

```
5 6
```

```
8 9
```

Ak chceme sčítať, či odčítať matice, musia mať podobne ako vektory rovnakú veľkosť. Potom medzi sebou odčítame jednotlivé prvky s rovnakými indexmi.

Pre vytvorenie transpozičnej matice  $\mathbf{a}^T$  z pôvodnej matice používame apostrof ""

```
>> a'
```

```
ans =
```

```
2 5 8
3 6 9 [4]
```

Nadobudnuté vedomosti o maticiach, vektoroch a skalároch môžeme využiť v nasledovných príkladoch:

$$2a + 3b + 1c + 6d = 4$$

$$4a - 2b + .5c + 1d = 2$$

$$6a + 1b + 0c - 1d = -1$$

$$2a + 2b - 1c + 3d = 4$$

Riešenie:

```
>> A=[2 3 1 6 ; 4 -2 .5 1 ; 6 1 0 -1 ; 2 2 -1 3]
```

```
A =
```

```
2 3 1 6
```

```
4 -2 1/2 1
```

```
6 1 0 -1
```

```
2 2 -1 3
```

```
>> B = [4 ; 2 ; -1 ; 4]
```

```
B =  
  
      4  
  
      2  
  
     -1  
  
      4  
  
>> x = inv(A)*B  
  
x =  
  
    0.1166 = a  
   -0.5544 = b  
   -1.4404 = c  
    1.1451 = d
```

## 1.4 Vykresľovanie grafov v MATLABe

### 1.4.1 Základný prehľad práce s grafmi

MATLAB umožňuje jednoduchú vizualizáciu dát. Ak chceme použiť funkciu pre vykresľovanie grafov v MATLABe "**plot**", musíme sa ubezpečiť, že matice, prípadne vektory, ktoré potrebujeme vykresliť majú rovnakú veľkosť. Funkcia "**plot**" vykresľuje graf zadaný pomocou hodnôt vo vektoroch. Pomocou príkazu "**figure**" si otvorí nové grafické okno do ktorého graf vykreslí.

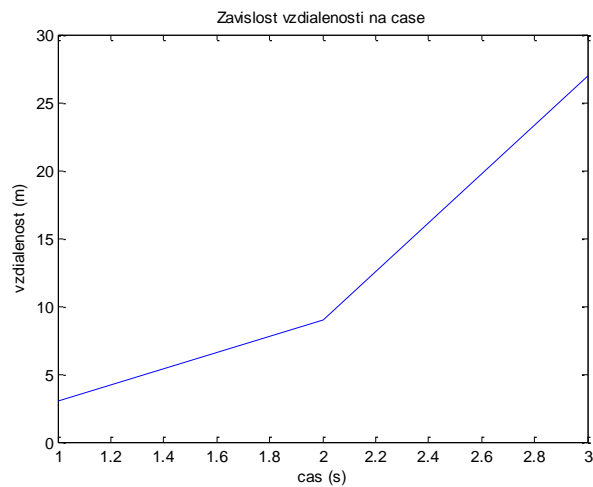
### 1.4.2 Vykresľovanie 2-D grafov

Ak chceme vykresliť graf vektoru o veľkosti 1x3 v závislosti na čase, tak vektor času musí mať tak isto veľkosť 1x3:

```
>> x = [3 9 27];  
>> t = [1 2 3];  
  
>> figure                                - Vytvorí novú schému  
  
>> plot (t, X)                            - v danej schéme vykreslí graf  
  
>> title('Zavislost vzdialenosti na case')  
  
>> ylabel('vzdialenost (m)')              - Popisok osi y
```

```
>> xlabel('cas (s)')
```

- Popisok osi x



Obr. 6 Vykreslenie závislosti 1 premennej

Ak však chceme vykresliť viacero vektorov do jedného grafu, musíme preddefinovať rozdielne farby pre každú charakteristiku. Pre sprehľadnenie môžeme vložiť i legendu.

Do predchádzajúceho grafu teda pridáme ďalšie 2 závislosti a zadefinujeme farby i legendu:

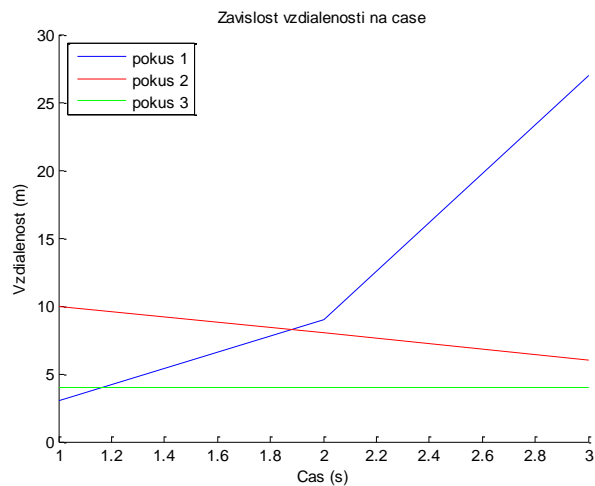
```
>> clear all
>> X = [3 9 27];
>> Y = [10 8 6];
>> Z = [4 4 4];

>> t = [1 2 3];

>> figure
>> hold on

>> plot (t, X, 'blue', t,Y,'red', t,Z,'green')
>> title('Zavislost vzdialenosti na case')
>> ylabel('Vzdialenost (m)')
>> xlabel('Cas (s)')

>> legend('pokus 1', 'pokus 2', 'pokus 3')
>> legend('Location','NorthWest')
```



Obr. 7 Vykreslenie závislosti 3 premenných

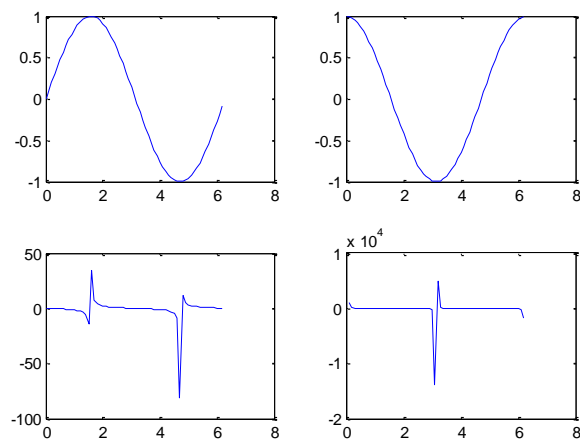
Pokiaľ potrebujeme do jedného grafického okna umiestniť viac navzájom nezávislých grafov, použijeme funkciu "**subplot**", ktorá vymedzí konkrétnu časť grafického okna a aktivuje ju pre vykresľovanie: `subplot(m,n,c)`. Funkcia vytvorí v grafickom okne maticu  $m \times n$  a každú jednu bunku matice očísľuje hodnotou **c**. V čase zadávania príkazu aktivuje práve jedno **c**-te okno, do ktorého vykreslí graf. Ak sa v zadanej bunke už nejaký graf nachádza, tak ho prepíše novým. Funkcia `subplot` sa volá pri každom okne. [5]

```
>> clear all
>> x = 0:.1:2*pi;
>> subplot(2,2,1);
>> plot(x,sin(x));

>> subplot(2,2,2);
>> plot(x,cos(x));

>> subplot(2,2,3);
>> plot(x,tan(-x));

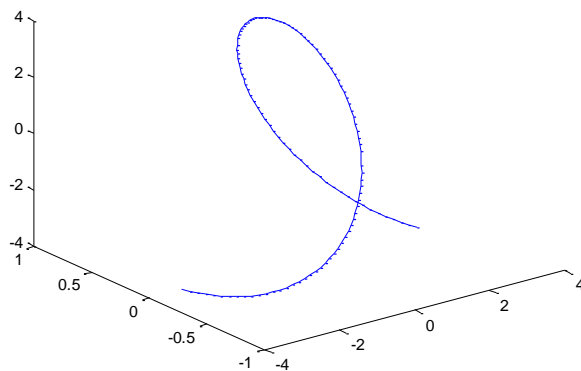
>> subplot(2,2,4);
>> plot(x,cot(x).^3);
```



Obr. 8 Použitie funkcie subplot

### 1.4.3 Vykresľovanie 3-D grafov

Najjednoduchší prípad vykresľovania 3-D grafov v MATLABe je v prípade ak máme 3 vektory premenných rovnakých rozmerov. Hovoríme vtedy o **čiarových** grafoch, pri ktorých používame funkciu `plot3`.

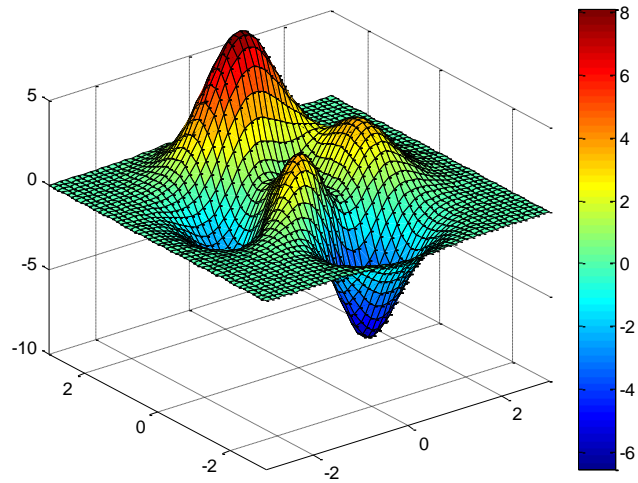


Obr. 9 Použitie funkcie plot3 - čiarový graf

Ďalší typ 3-D grafov v MATLABe sú takzvané plošné grafy, ktoré nám zobrazia celú 3-D plochu. Najskôr potrebujeme vygenerovať 2-D plochu x-y, na ktorú potom aplikujeme tretiu funkciu ako rozmer **Z**. [6]

```
>> clear all
>> [x,y] = meshgrid([-3:.125:3]);
>> Z= peaks(-x,y);
```

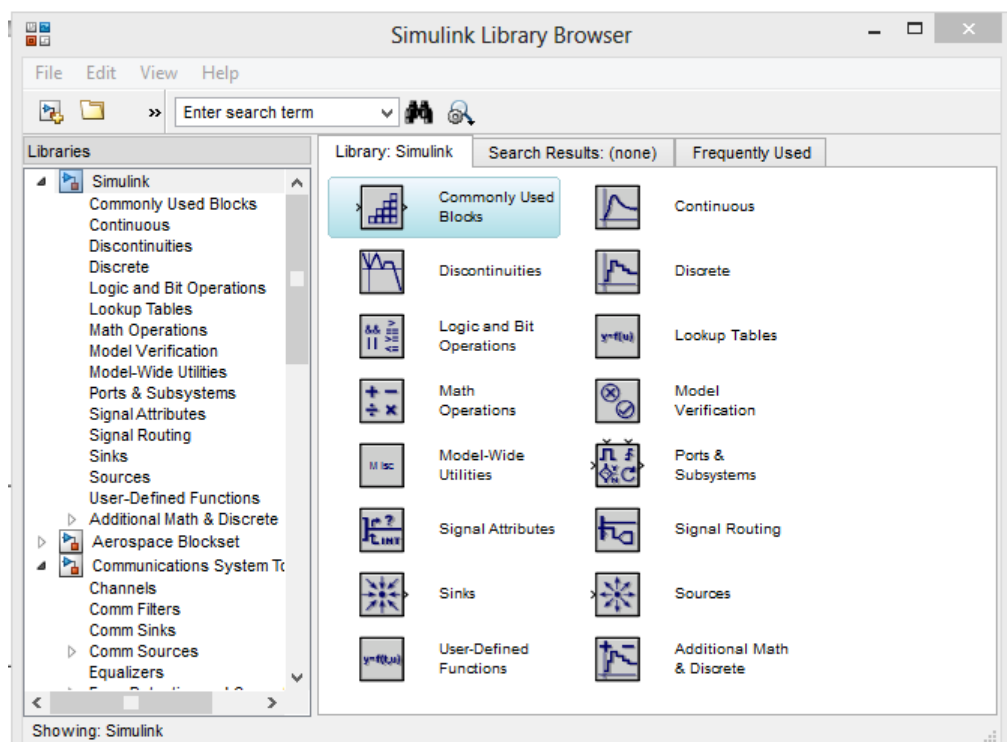
```
>> figure(2)
>> surf(-x,y,Z);
>> axis([-3 3 -3 3 -10 5])
>> colorbar
```



Obr. 10 Příklad plošného grafu pomocí příkazu surf.

## 2 SIMULINK

Simulink predstavuje grafickú nadstavbu programu MATLAB určenú pre modelovanie a simuláciu a analýzu doménových dynamických systémov. Obsahuje funkcie, ktoré môžu byť použité na reprezentáciu lineárnych i nelineárnych systémov s kontinuálnou, či vzorkovacou časovou charakteristikou. Simulink dokáže bezproblémovo komunikovať s pracovným prostredím i s funkciami MATLABu ako aj programami naprogramovanými užívateľom v iných jazykoch. Môže byť teda riadený, alebo písaný priamo z MATLABu. Významná je jeho schopnosť automaticky generovať zdrojový kód jazyka C pre implementáciu do real-time systémov. Je možné ho použiť s množstvom hardvéru či softvéru firmy MathWorks. [7]



Obr. 11 Základné okno nástroja Simulink

Systémy sú v Simulinku vykreslené ako blokové diagramy. Tieto blokové diagramy popisujú rovnice a vzťahy ich komponentov. K dispozícii sú aj zobrazovacie funkcie pre priame vykresľovanie (už za behu simulácie). Okrem štandardných úloh dovoľuje Simulink rýchlo a presne simulovať i rozsiahle "stiff" systémy s efektívnym využitím pamäti počítača. Simulink tiež umožňuje spúšťať určité časti simulačnej schémy na základe výsledkov logickej podmienky. Tieto spúšťané a povolané subsystémy umožňujú použitie programu v náročných simulačných experimentoch. Samozrejmosťou je otvorená architektúra, ktorá

dovoľuje užívateľovi vytvárať vlastné funkčné bloky a rozširovať už tak bohatú knižnicu Simulinku. Hierarchická štruktúra modelov umožňuje koncipovať i veľmi zložité systémy do prehľadnej sústavy subsystémov prakticky bez obmedzenia počtu blokov. [8]

## 2.1 Hlavné súčasti Simulinku

- Grafický editor pre vytváranie a úpravu hierarchických blokových diagramov
- Knižnice preddefinovaných blokov pre modelovanie časovo spojených i diskretných systémov
- Simulačný prostriedok s pevným i variabilným krokom
- Rozsahy a dátové výstupy pre zobrazenie výsledkov simulácie
- Nástroje pre správu dát a modelových súborov projektu.
- Nástroje správy modelu pre zdokonalenie architektúry modelu a zvýšenie rýchlosti simulácie
- Blok funkcie MATLABu pre importovanie algoritmov z MATLABu
- Nástroj **Legacy Code** pre importovanie C a C++ kódu

## 2.2 Toolboxy v Simulinku

Simulink obsahuje množstvo toolboxov pre široké využitie. V skratke popíšem tie najdôležitejšie. Rozdelím ich na 3 skupiny z hľadiska dôležitosti a využitia: **najpoužívanejšie, menej používané**, ale užitočné a tzv. "**nice to have**", čiže ktoré sa môžu hodiť, ale nie sú nevyhnutné pre prácu. V krátkosti popíšem pár najdôležitejších toolboxov pre prácu s MATLABom a Simulinkom:

### 2.2.1 Najpoužívanejšie toolboxy pre prácu v MATLABe a Simulinku

- **Communications system toolbox** - poskytuje algoritmy a nástroje pre dizajn, simuláciu a analýzu komunikačných systémov. K tomuto toolboxu sa ešte vrátíme v ďalšej kapitole.
- **Control System Toolbox** - poskytuje priemyselné štandardy, algoritmy a nástroje systematické analyzovanie, návrh a ladenie lineárnych kontrolných systémov.
- **Filter Design Toolbox** - vychádza z výpočtového prostredia MATLABu a System Processing Toolboxu. Obsahuje rad pokročilých konštrukčných filtrov, ktoré podporujú návrh simulácií, a analýzu pevného bodu a voliteľné filtre pohyblivej desatinnej čiarky pre širokú škálu presnosti.[9]

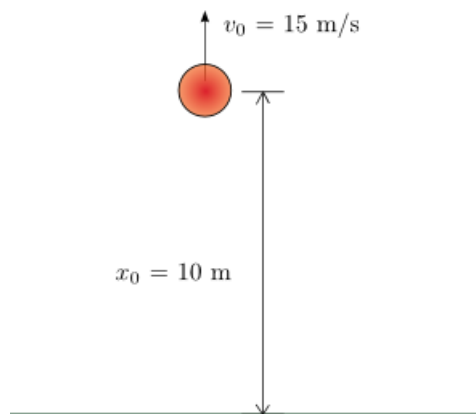
- **Image Processing Toolbox** - poskytuje rozsiahly súbor referenčne - štandardných algoritmov a grafických nástrojov pre spracovanie obrazu, analýzu, vizualizáciu a vývoj algoritmu.
- **MATLAB Compiler** - konvertuje užívateľom napísané MATLAB skripty do samostatných aplikácií. K spusteniu teda nepotrebujú prostredie MATLAB. Vďaka tomu budú fungovať na viacero počítačoch aj bez licencovaného MATLABu.
- **MATLAB Runtime Server** - obsahuje všetky výpočtové i grafické možnosti komerčného MATLABu, ale je určený pre prevádzku samostatných aplikácií, ktoré sú založené na MATLABe. Koncoví užívatelia takýchto aplikácií nepotrebujú vlastniť MATLAB, ani nepotrebujú vlastniť nejaké vedomosti o MATLABe. Do zdrojového kódu takýchto aplikácií však prístup nemajú.
- **Neural Network Toolbox** - poskytuje nástroje pre návrh, tréning, tvorbu, simuláciu a vizualizáciu neurónových sietí. Neurónové siete nachádzajú uplatnenie hlavne v oblastiach, kde je použitie formálnych analytických nástrojov zložité či dokonca nemožné, ako napríklad vzorové rozpoznávanie či riadenie a identifikácia nelineárnych sústav. Neural Network Toolbox ponúka grafické užívateľské rozhrania, ktoré uľahčujú návrh a prácu s neurónovými sieťami.
- **Optimization Toolbox** - rozširuje výpočtové prostredie MATLAB o nástroje a široko používané algoritmy pre štandardné a rozsiahle optimalizačné úlohy. Tieto algoritmy riešia podmienené i nepodmienené, spojité i diskrétné problémy. Toolbox obsahuje funkcie pre lineárne programovanie, kvadratické programovanie, nelineárnu optimalizáciu, nelineárnu metódu najmenších štvorcov, riešenie nelineárnych rovníc a binárne celočíselné programovanie.
- **Partial Differential Equation Toolbox** - obsahuje nástroje pre riešenie parciálnych diferenciálnych rovníc (PDE) v dvojdimenzionálnych rozmeroch (2-D) a v čase. Súbor funkcií do príkazového riadka a grafické užívateľské rozhranie umožňujú pripraviť, riešiť a dodatočne spracovať riešenie obyčajných dvojrozmerných parciálnych diferenciálnych rovníc pre širokú škálu vedeckých aplikácií.
- **Signal Processing toolbox** - rovnako aj väčšina ostatných knižníc MATLABu je aj táto vybavená grafickým rozhraním, ktoré zjednodušuje frekvenčnú analýzu signálov i jej úpravy, napríklad filtráciu, moduláciu, alebo demoduláciu. K najdôležitejším schopnostiam tohto toolboxu patrí návrh a analýza filtrov vrátane interaktívneho zobrazovania fázových a amplitúdových charakteristík.

- **Statistics Toolbox** - ponúka rozsiahly súbor nástrojov pre prácu s dátami. zahŕňa funkcie a interaktívne nástroje pre modelovanie dát, vývoj algoritmov pre štatistiku, analýzu trendov a simuláciu stochastických systémov.
- **Symbolic Math Toolbox** - poskytuje nástroje pre riešenie a prácu so symbolickými výrazmi a aritmetiku s premenlivou presnosťou. Do MATLABu prináša stovky symbolických funkcií. [10]

## 2.3 Príklad simulácie v Simulinku

V Simulinku, ako aj na webových stránkach nájdeme veľké množstvo vzorových príkladov a simulácií. Môžeme tak lepšie pochopiť fungovanie celej simulácie a princíp fungovania jednotlivých blokov použitých pri simulácii.

### 2.3.1 Simulácia poskakujúcej loptičky



Obr. 12 Loptička je vyhodená rýchlosťou 15 m/s z výšky 10 m.

Táto simulácia je klasický príklad **hybridného dynamického systému**. V tomto prípade ide o dynamiku danú jednoduchými vzťahmi:

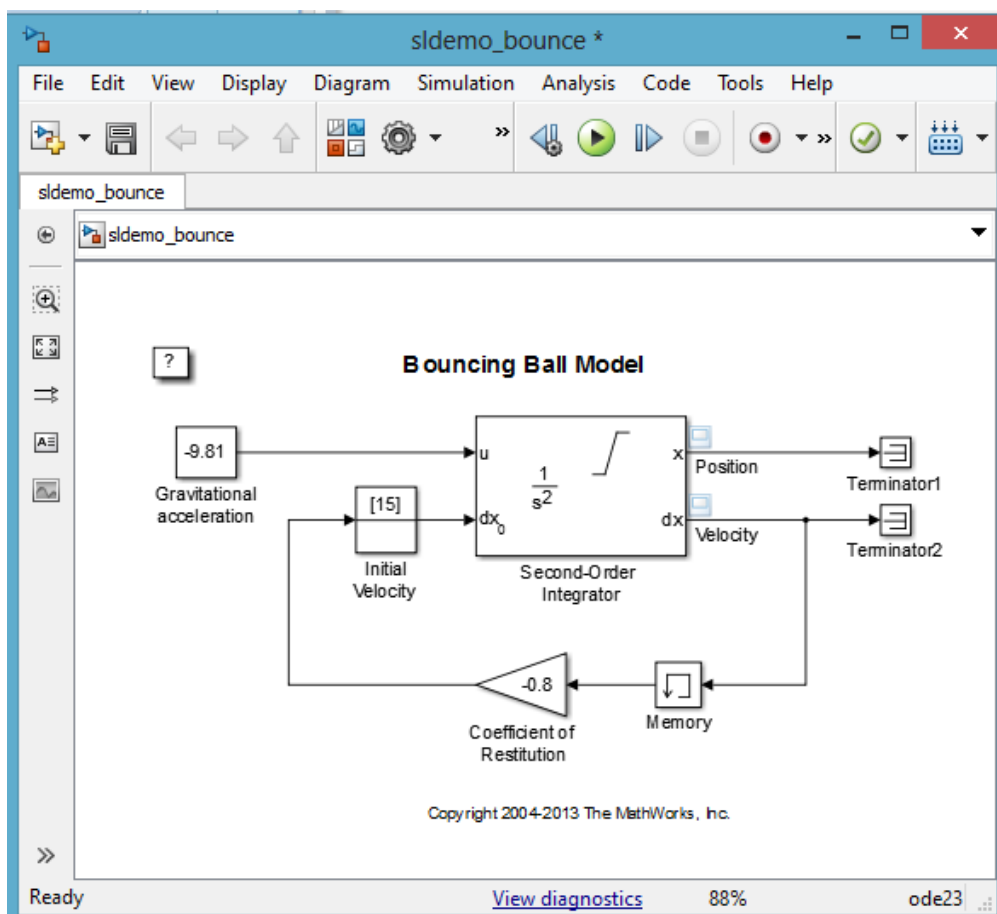
$$\frac{dv}{dt} = -g,$$
$$\frac{dx}{dt} = v,$$

kde  $g$  je zrýchlenie spôsobené gravitáciou,  $x(t)$  je pozícia loptičky a  $v(t)$  je rýchlosť. Preto má tento systém 2 priebehové stavy: pozíciu  $x$  a rýchlosť  $v$ .

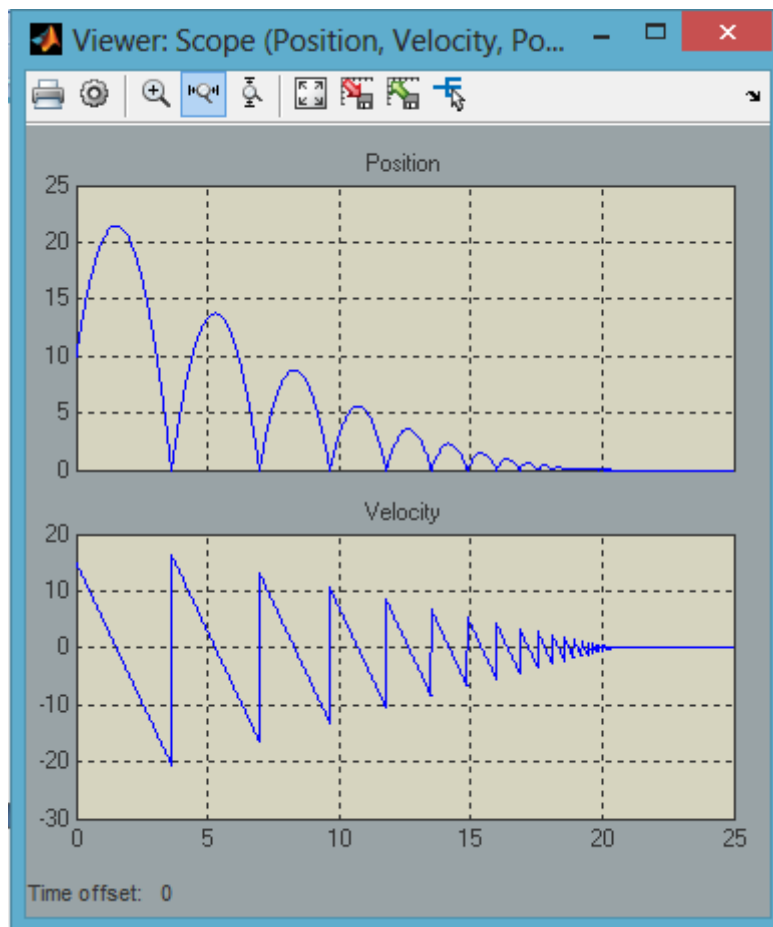
Keďže loptička niekoľkokrát koliduje s rovinou podlahy, pri každej kolízii sa mení vektor rýchlosti  $v^-$  pred a  $v^+$  po odraze. Tento aspekt vyriešime koeficientom odrazu loptičky  $k$ .

$$v^+ = -kv^-, \quad x = 0$$

Na modelovanie tohto systému môže použiť jednoduchší blok **Second-Order Integrator**.



Obr. 13 Realizácia simulácie poskakujúcej loptičky



Obr. 14 Výsledné charakteristiky pozície  $x$  a rýchlosti  $v$  poskakujúcej loptičky

/ <http://www.mathworks.com/help/simulink/examples/simulation-of-a-bouncing-ball.html> /

### 3 COMMUNICATIONS SYSTEM TOOLBOX

Tento toolbox poskytuje algoritmy a nástroje pre návrh, simuláciu a analýzu komunikačných systémov. Algoritmy sú vo forme funkcií MATLABu, systém objektov a Simulink blokov. Communications System Toolbox obsahuje algoritmy pre kódovanie kanálov, prekladanie, moduláciu, ekvalizáciu, synchronizáciu a modelovanie kanálov.

Nástroje umožňujú analýzu bit-error-rate, tvorbu diagramov a vizualizáciu charakteristík kanálov. Toolbox tiež obsahuje adaptívne algoritmy, ktoré umožňujú modelovanie dynamických komunikačných systémov využívajúcich techniky OFDM, OFDMA či MIMO. Algoritmy podporujú aritmetiku pevného bodu a generovanie C i HDL kódu. [11]

#### 3.1 Návrh systému, charakterizácia a vizualizácia

Návrh a simulácia komunikačných systémov vyžaduje analýzu reakcie na šum a rušenie vznikajúce v reálnych podmienkach a sledovanie ich správania pomocou grafických a kvantitatívnych prostriedkov. Z týchto hodnôt treba následne určiť či výsledný výkon spĺňa štandardy prijateľnosti.

Communications System Toolbox realizuje celý rad úloh pre návrh a simuláciu komunikačných systémov. Mnoho funkcií, systémových objektov a blokov v toolboxe vykonáva kalkulácie spojené s konkrétnymi zložkami komunikačných systémov ako je ekvalizér či demodulátor. Ďalšie funkcie sú navrhnuté pre vizualizáciu a analýzu. [12]

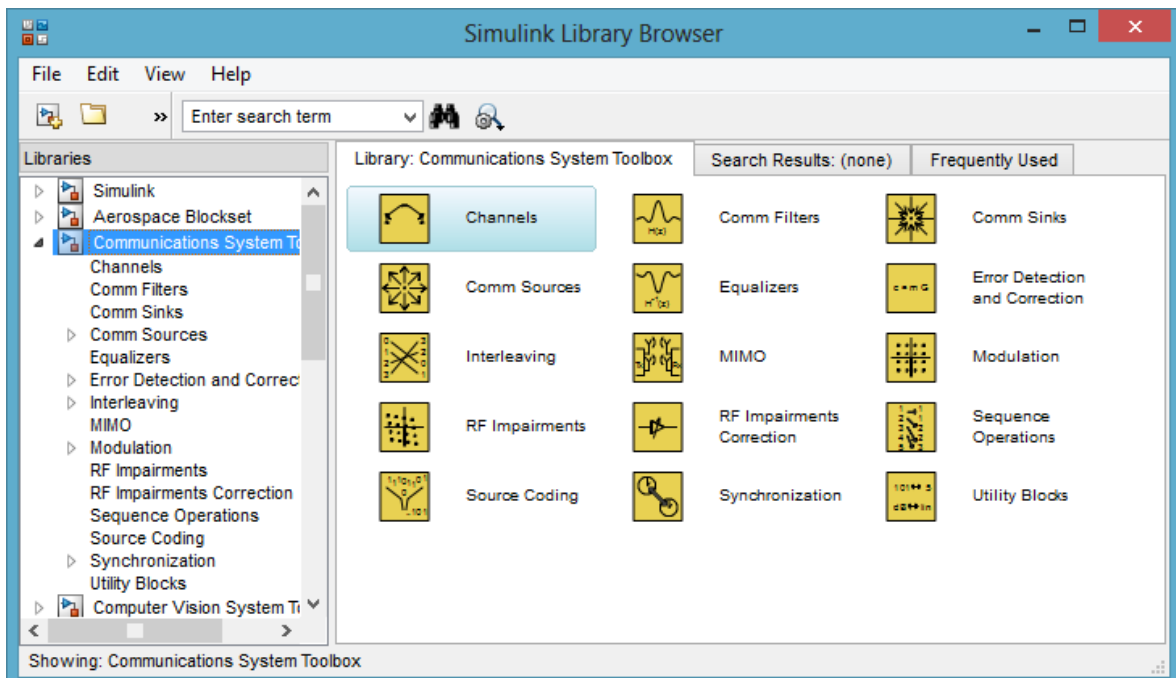
#### 3.2 Hlavné súčasti Communications System Toolboxu

- Algoritmy pre dizajnovanie fyzickej časti komunikačných systémov, obsahujúce kódovanie zdroja, kódovanie kanálu, prekladanie (interleaving), moduláciu, modely kanálov, MIMO, ekvalizáciu a synchronizáciu
- Systémové objekty pre výpočtovo náročné algoritmy ako sú Turbo, LDPC a Viterbi dekodéry
- Aplikácia rozsahu diagramu oka a vizualizačné funkcie pre rozptyl kanálu
- Aplikácia bitovej chybovosti pre porovnávanie simulovanú bitovú chybovosť systému s analytickými výsledkami
- Kanálové modely obsahujúce AWGN, Multipath Rayleigh Fading, Rician Fading, MIMO Multipath Fading a LTE MIMO Fading

- Algoritmy dostupné ako MATLAB funkcie, MATLAB systémové objekty a Simulink bloky
- Podpora pre modelovanie pevného bodu a generovanie C a HDL kódu.

### 3.3 Popis základných blokov Communication System Toolboxu

Communication System toolbox obsahuje veľké množstvo blokov pre, návrh, simuláciu a analýzu komunikačných systémov. Všetky bloky sú rozdelené do 15 knižníc.



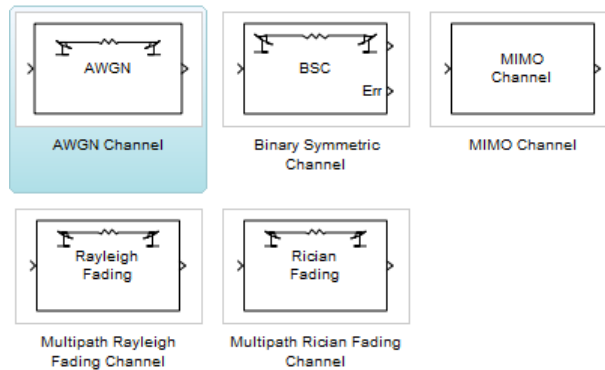
Obr. 15 Communications System Toolbox a jeho bloky.

#### 3.3.1 Knižnica Komunikačných Kanálov

Communications System Toolbox poskytuje algoritmy a nástroje pre modelovanie šumu, zoslabovanie signálu, rušenia a ostatných skreslení, ktoré nájdeme v reálnych komunikačných kanáloch. Tento toolbox podporuje nasledujúce typy kanálov:

- AWGN Channel - aditívny biely Gaussov šum (AWGN)
- MIMO Channel - zoslabenie viacnásobného vstupu, viacnásobného výstupu
- Multipath Rayleigh Fading Channel

- Multipath Rician Fading Channel
- Binary Symmetric Channel - binárne symetrické kanály

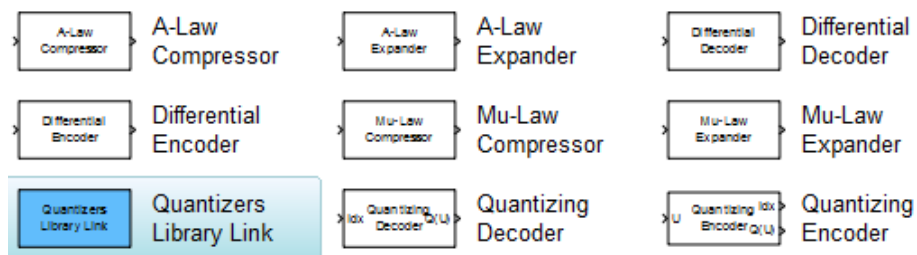


Obr. 16 Bloky určené pre simuláciu podmienok v komunikačnom kanáli

### 3.3.2 Knižnica Kódovania Zdroja

Svojevoľne konvertuje informácie reálneho sveta na informácie reprezentované prijateľne pre komunikačné systémy. Tieto nástroje poskytujú základné techniky ako napríklad riešenie problémov kódovania zdroja použitím Simulinku a MATLABu. Tento toolbox obsahuje techniky signálnej kvantizácie a diferenciálnej pulznej kódovej modulácie.

Source Coding taktiež zahŕňa tzv. **kompander** techniku. Názov kompander vznikol kombináciou kompresora a expanderu. Dátová kompresia je dôležitá pre premenu signálov s rôznym výkonovým stupňom transformácie.



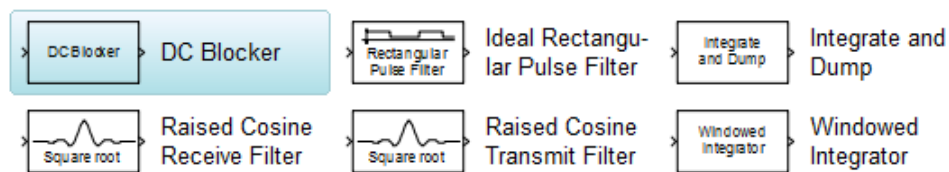
Obr. 17 Bloky obsiahnuté v skupine Source Coding

### 3.3.3 Knižnica Filtrovania

Communications System Toolbox obsahuje zopár funkcií, objektov a blokov, ktoré nám môžu pomôcť s návrhom a používaním filtrov. Ďalšie filtračné prostriedky nájdeme v Sig-

nal Processing Toolboxe a v DSP System Toolboxe. Tento toolbox obsahuje nasledujúce filtračné bloky:

- Ideal Rectangular Pulse Filter - tvarovanie vstupného signálu pomocou ideálne obdĺžnikových pulzov
- Integrate and Dump- integrácia diskretného signálu, pravidelné resetovanie na nulu
- Raised Cosine Receive Filter - filter vstupného signálu, pravdepodobné prevzorkovanie pomocou kosínusového FIR filtra
- Raised Cosine Transmit Filter - vzorkovanie a filter vstupného signálu použitím kosínusového FIR filtra



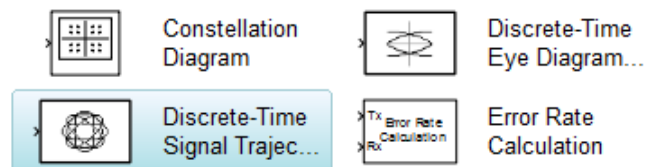
Obr. 18 Skupina filtračných blokov komunikačných systémov

### 3.3.4 Knižnica Comm Sink

Obsahuje 4 základné bloky pre kvalitatívne vizuálne pochopenie správania systému:

- Constellation Diagram - Vykresľuje hodnoty signálu a jeho pevné body. V najlepšom bode by pevné body signálu mohli byť práve v čase, keď je oko pri diagramoch signálov oka otvorené. Tento blok vytvára konštelačný diagram z diskretných systémov.
- Discrete-Time Eye Diagram - pre vytvorenie diagramu oka zobrazuje viacero stôp modulovaného signálu. Môžeme ho použiť na odhalenie vlastností modulácie signálu, ako pulzové formátovanie alebo skreslenia kanálu.
- The Discrete-Time Signal Trajectory Scope - Rozsah trajektórie diskretného signálu. Zobrazuje trajektóriu modulovaného signálu a to vykresľovaním jeho fázovej zložky oproti kvadrátúrnej zložke. Tento Blok má jeden vstupný port, pomocou ktorého prijíma komplexný skalárny alebo stĺpcový vektor vstupného signálu. Prijíma signál s nasledujúcimi typmi dát: double, single, základné prirodzené čísla a vstup pevného bodu, ten ale bude reprezentovaný ako double typ.

- Error Rate Calculation - výpočet chybovosti. Porovnáva vstupné dáta od vysielača so vstupnými dátami z prijímača. Počíta chybovosť ako priebehovú charakteristiku a to podelením celkového množstva chybných párov s celkovým množstvom vstupných dát zo toho istého zdroja. Tento blok sa využíva na výpočet drobnej chybovosti, lebo neberie do úvahy veľké rozdiely medzi elementmi vstupných dát.



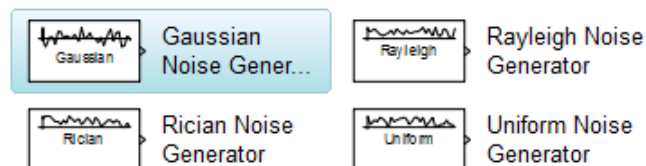
Obr. 19 Bloky knižnice Comm Sink

### 3.3.5 Knižnica Comm Sources

Obsahuje bloky rozdelené do 3 ďalších podskupín:

**Noise Generators**, čiže generátory šumu. Bloky v tejto podknižnici generujú náhodné dáta pre simuláciu šumu. Bloky v tejto skupine môžeme použiť na generovanie náhodných reálnych čísel závislých na type prenosu. Nájdeme tu nasledujúce bloky:

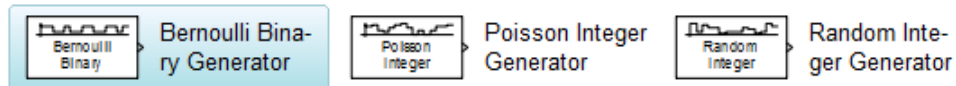
- Gaussian Noise Generator - generuje diskretný biely Gaussov šum. Musíme špecifikovať vstupný vektor simulácie.
- Rayleigh Noise Generator - generuje Rayleighov distribuovaný šum
- Rician Noise Generator - generuje Ricianov distribuovaný šum
- Uniform Noise Generator - generuje rovnomerne rozložený šum medzi hornou a dolnou hranicou.



Obr. 20 Bloky generujúce šum v knižnici Noise Generators

**Random Data Source** obsahuje 3 bloky pre generátora náhodných dát zdroja.

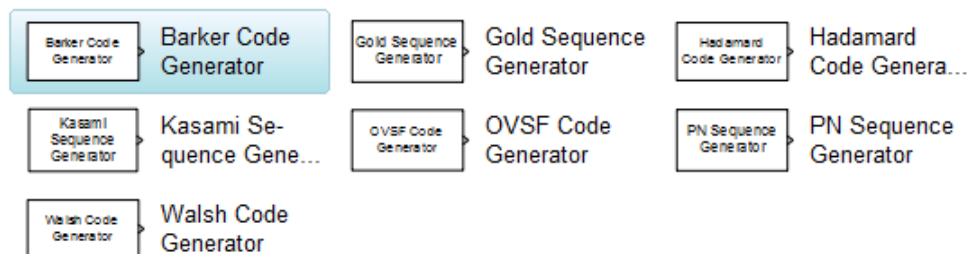
- Bernoulli Binary Generator - generuje náhodné binárne čísla použitím Bernoulliho distribúcie.
- Poisson Integer Generator - generuje vektory obsahujúce náhodne nezáporné prirodzené hodnoty
- Random Integer Generator - generuje náhodne rozdelené prirodzené hodnoty v rozmedzí 0 až M-1, kde hodnota M môže byť buď skalár alebo vektor



Obr. 21 Bloky obsiahnuté v knižnici Random Data Source

**Sequence Generators** obsahuje bloky určené na generovanie sekvencií pre prenos alebo synchronizáciu v komunikačnom systéme. Bloky obsiahnuté v tejto podknižnici generujú **pseudonáhodné sekvencie**, **synchronizačné kódy** a **ortogonálne kódy**. Do tejto skupiny patria bloky generátorov:

- Barker Code Generator
- Gold Sequence Generator
- Hadamard Code Generator
- Kasami Sequence Generator
- OVVSF Code Generator
- PN Sequence Generator
- Walsh Code generator



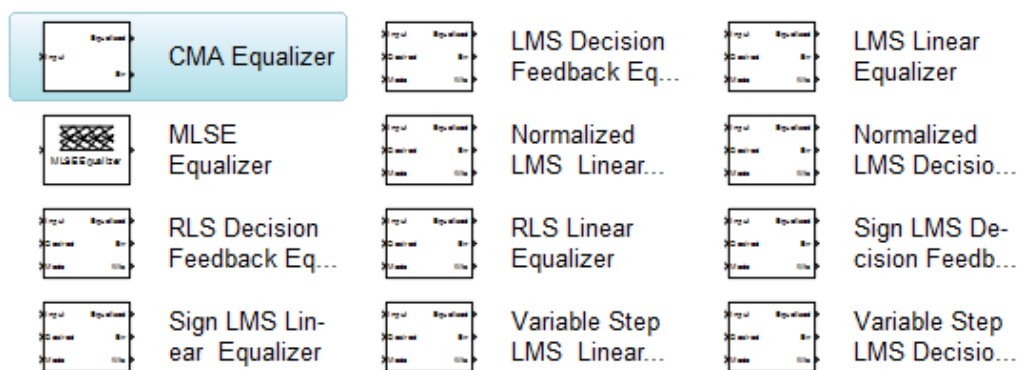
Obr. 22 Bloky obsiahnuté v knižnici Sequence Generators

### 3.3.6 Knižnica Equalizers

Communications System Toolbox nám umožňuje preskúmať i ekvalizačné a synchronizačné techniky komunikačného kanálu. Návrh a charakterizácia takýchto techník je vo vše-

obecnosti náročné. Tento toolbox poskytuje algoritmy a nástroje, ktoré nám umožnia rýchlo vybrať vhodnú techniku pre náš komunikačný systém. Communications System Toolbox poskytuje na vyhodnocovanie rozdielnych prístupov ekvalizácie adaptívne algoritmy ako:

- CMA Equalizer - používa konštantný model algoritmu
- LMS Decision Feedback Equalizer - upravuje váhy s LMS algoritmom
- LMS Linear Equalizer - lineárne vyhodnocuje váhy s LMS algoritmom
- MLSE Equalizer - Využíva Viterbiho algoritmus
- Normalized LMS Linear Equalizer - upravuje váhy s normalizovaným LMS algoritmom
- Normalized LMS Decision Feedback - upravuje váhy s normalizovaným LMS algoritmom
- RLS Decision Feedback Equalizer - upravuje váhy pomocou RLS algoritmu
- RLS Linear Equalizer - upravuje váhy pomocou RLS lineárneho algoritmu
- Sing LMS Decision Feedback Equalizer - upravuje váhy pomocou podpísaného LMS algoritmu
- Sign LMS Linear Equalizer - upravuje váhy pomocou podpísaného LMS algoritmu
- 
- Variable Step LMS Linear Equalizer - využíva premennú veľkosť kroku LMS algoritmu
- Variable Step LMS Feedback Decision Equalizer - využíva premennú veľkosť kroku LMS algoritmu



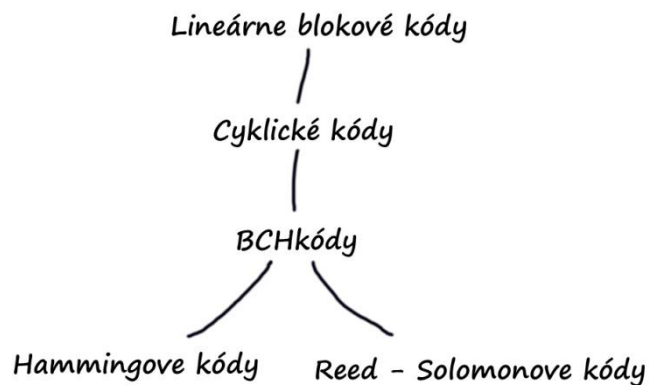
Obr. 23 Ekvalizačné bloky Communications System Toolboxu

### 3.3.7 Knižnica Error Detection and Correction

Obsahuje bloky určené na skenovanie systému, rozpoznanie chýb a s ním spojené i opravenie chýb. Celú knižnicu rozdeľujeme do troch častí: **blokové kódy**, **CRC kódy** a **konvolučné kódy**.

**Blokové kódovanie** je špeciálny prípad kódovania kontroly chýb. Blokové kódovacie techniky mapujú pevný počet znakov správy na pevný počet kódovacích symbolov. Blokový kodér spracúva každý blok dát nezávisle a to bez toho aby ho uložil do pamäti.

Trieda blokových kódovacích techník obsahuje kategórie uvedené v nasledujúcom diagrame.

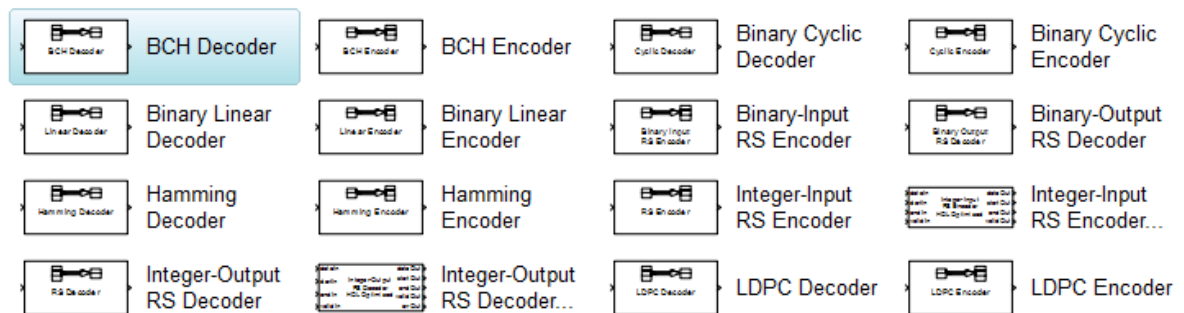


Obr. 24 Blokové kódovacie techniky

V tejto podknížnici blokového kódovania nájdeme nasledujúce kodéry a dekodéry:

- Binary Linear Decoder
- Binary Linear Encoder
- Binary Cyclic Decoder
- Binary Cyclic Encoder
- Hamming Decoder
- Hamming Encoder
- BCH Decoder
- BCH Encoder
- Binary-Input RS Encoder
- Binary-Output RS Decoder
- Integer-Input RS Encoder

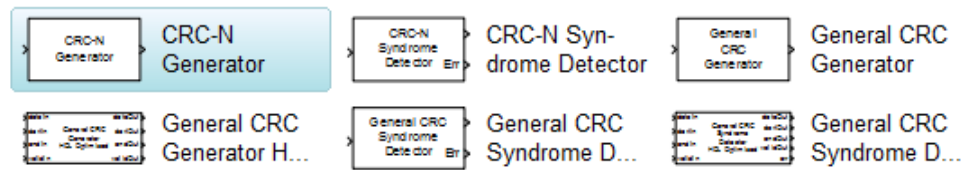
- Integer-Output RS Decoder
- Integer-Input RS Encoder HDL Optimized
- Integer-Output RS Decoder HDL Optimized
- LDPC Decoder
- LDPC Encoder



Obr. 25 Blokové kódovacie techniky

**CRC (Cyclic Redundancy Check) kódovanie** je kódovacia technika pre detekciu chýb vzniknutých pri prenose správy. Na rozdiel od blokových, alebo konvolučných kódov CRC kódy nemajú vstavanú schopnosť opravy chýb. Namiesto toho ak komunikačný systém zistí prítomnosť chyby v prijatej správe, príjemca požiada odosielateľa o opätovné prepísanie správy. Vysielač pri CRC kódach pridáva ku každému slova správy extra bity, tzv. kontrolný súčet. Ten potom pripojí ku kódovanému slovu. Príjemca po obdržaní preneseného slova vykoná tie isté pravidlá pre vytvorenie kontrolného súčtu. Ak sa obe hodnoty nerovnajú, požiada vysielač aby kódované slovo znovu preposlal. Communications System Toolbox obsahuje 6 blokov ktoré implementujú CRC algoritmus.

- CRC-N Generator - generuje CRC kód a pripojí ho k dátovému rámcu
- CRC-N Syndrome Detector - detekuje chyby vo vstupných dátových rámcoch
- General CRC Generator - generuje CRC bity podľa generujúceho polynómu a pripojí ich k dátovým rámcem
- General CRC Syndrome Detector - detekuje chyby vo vstupných dátových rámcoch podľa generujúceho polynómu
- General CRC Generator HDL Optimized - Generuje CRC bity a pridáva ich do vstupných dát optimalizovaných pre HDL kódy
- General CRC Syndrome Detector HDL Optimized - detekuje chyby vo vstupných dátach pomocou CRC

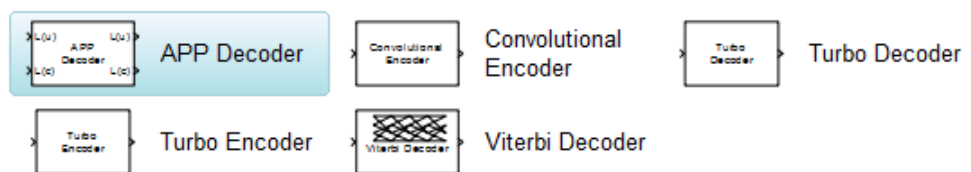


Obr. 26 Bloky implementujúce CRC kódovací algoritmus

**Konvolučné kódy** predstavujú zvláštny prípad kódovania proti chybám. Konvolučný kódér obsahuje na rozdiel od blokového kódéra vnútornú pamäť. Aj keď konvolučný kódér prijíma pevný počet symbolov správy a vytvára pevný počet kódovaných symbolov, jeho výpočty sú závislé nielen ma aktuálnej množine vstupných symbolov ale aj na niektorých predchádzajúcich vstupných symbolov.

V tejto knižnici nájdeme 5 blokov riešiacich konvolučné kódy:

- APP Decoder - dekóduje konvolučné kódy pomocou metódy pravdepodobnosti (APP)
- Convolutional Encoder - vytvára konvolučný kód pre binárne dáta
- Turbo Decoder - dekóduje vstupný signál použitím paralelne zreťazenej dekódovacej schémy
- Turbo Encoder - kóduje binárny signál použitím paralelne zreťazenej dekódovacej schémy
- Viterbi Decoder - dekóduje konvolučne zakódované dáta pomocou Viterbiho algoritmu.



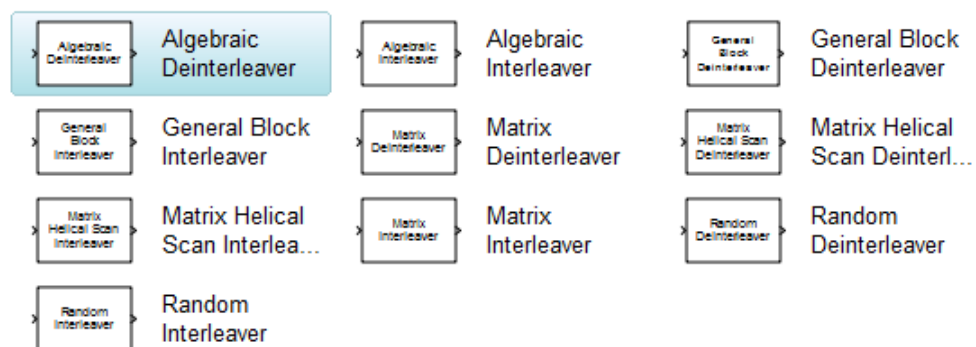
Obr. 27 Bloky určené na riešenie konvolučných algoritmov

### 3.3.8 Knižnica Interleaving

Podobne ako v predchádzajúcom prípade, obsahuje podknížnice ako **blokové** a **konvolučné** prekladanie.

**Blokové prekladanie** - prekladačí blok prijme súbor symbolov a usporiada ich bez toho žeby nejaké sa symboly opakovali, či dokonca vynechali. Počet symbolov v každom súbore je pre každý prekladač presne daný. Samotná operácia prekladania jedného súboru symbolov nie je nijako závislá na ostatných súboroch symbolov. Patria sem nasledovné bloky:

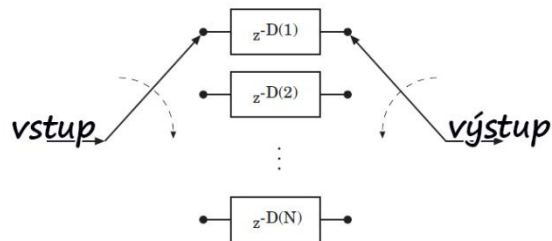
- Algebraic Deinterleaver
- Algebraic Interleaver
- General Block Deinterleaver
- General Block Interleaver
- Matrix Deinterleaver
- Matrix Helical Scan Deinterleaver
- Matrix Helical Scan Interleaver
- Matrix Interleaver
- Random Deinterleaver
- Random Interleaver



Obr. 28 Bloky implementujúce blokové prekladanie

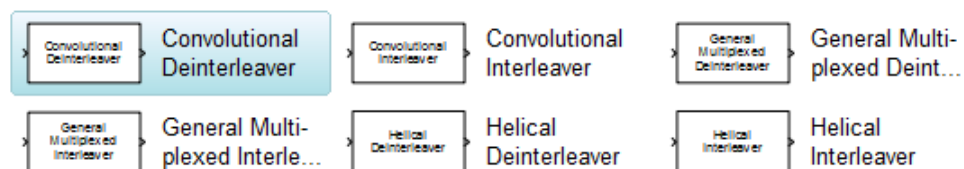
**Konvolučné prekladacie funkcie** sa skladajú zo sady posuvných registrov, každý z nich má pevné oneskorenie. V typickom konvolučnom prekladaní sú doby meškania nezáporné prirodzené násobky fixných celých čísel. Každý nový symbol zo vstupného vektora sa zaradi do ďalšieho posuvného registra a najstarší symbol tohto registra sa stáva časťou výstupného vektora. Konvolučné spôsoby prekladania používajú vnútornú pamäť, to znamená že operácia závisí nielen na aktuálnych symboloch ale aj na predchádzajúcich. Nasledujúca schéma znázorňuje štruktúru všeobecného konvolučného prekladania. Môžeme vidieť sady posuvných registrov a ich hodnoty oneskorenia  $D(1)$ ,  $D(2)$ , ...,  $D(N)$ .  $K$ -ty posun registra drží  $D(k)$  symbol, kde  $k = 1, 2, 3, \dots, N$ .

Konvolučná prekladacia funkcia má v tomto toolboxe vstupné argumenty, ktoré ukazujú počet posuvných registrov a oneskorenie pre každý z nich.



Obr. 29 Schéma konvolučne prekladacích blokov

- Convolutional Deinterleaver
- Convolutional Interleaver
- General Multiplexed Deinterleaver
- General Multiplexed Interleaver
- Helical Deinterleaver
- Helical Interleaver



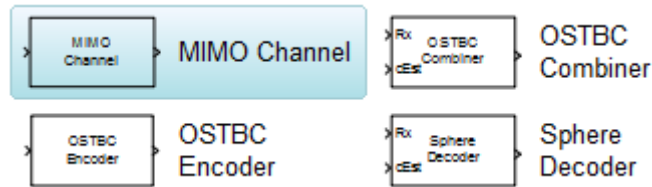
Obr. 30 Základné bloky konvolučného prekladania

### 3.3.9 Multiple-Input Multiple-Output (MIMO)

Používanie MIMO technológií spôsobilo revolúciu bezdrôtových komunikačných systémov s potenciálnym ziskom kapacity používaním viacerých antén ako na vysielači, tak i na prijímači. Technológia MIMO sa začala používať v niekoľkých bezdrôtových systémoch vrátane WiFi, WiMAX, LTE a je navrhnutá i pre budúce štandarty ako LTE-Advanced a IMT-Advanced. V Communications system toolboxe s touto technológiou pracujú 4 bloky:

- MIMO Channel - filter vstupného signálu cez MIMO viaccestný zoslabujúci kanál

- OSTBC Combiner - kombinuje vstupy pre prijaté signály a kanály pre ortogonálny časo-priestorový blok kódu
- OSTBC Encoder - kóduje vstupné správy použitím OSTBC kódu
- Sphere Decoder - dekóduje vstupné správy pomocou sphere dekodéru.



Obr. 31 MIMO bloky

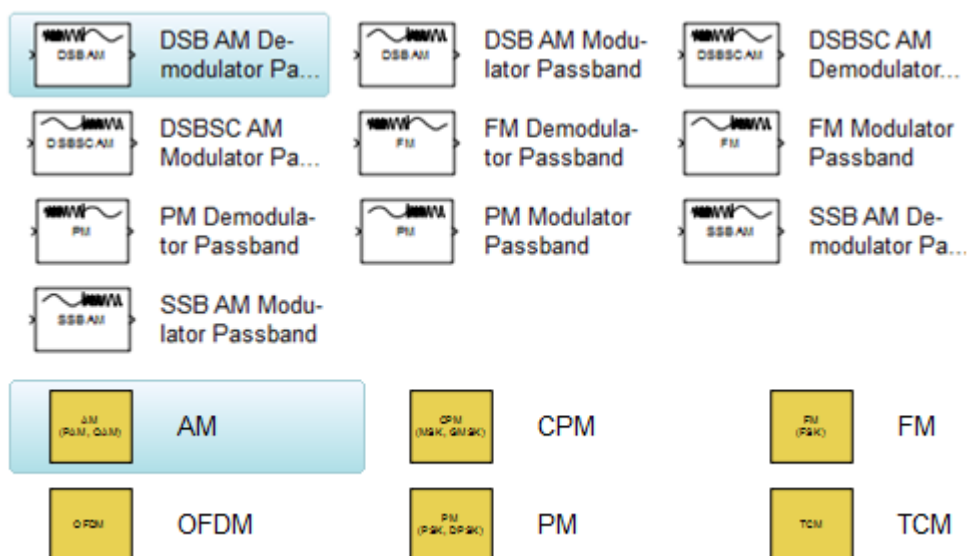
### 3.3.10 Analógová a Digitálna modulácia

Analógové a digitálne modulačné techniky kódujú prúd informácií na signál, ktorý je vhodný pre prenos. Communications System Toolbox poskytuje množstvo modulačných a k nim i príslušných demodulačných schopností. Tieto schopnosti sú dostupné ako funkcie a objekty MATLABu a bloky Simulinku.

Modulačné typy v komunikačnom toolboxe sú:

Analógové: AM, FM, PM, SSB, a DSBSC

Digitálne: FSK, PSK, BPSK, DPSK, OQPSK, MSK, PAM, QAM, a TCM

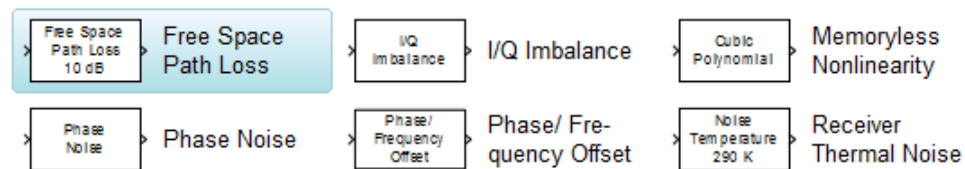


Obr. 32 Analógové a digitálne modulačné bloky

### 3.3.11 RF poruchy

Pozostávajú zo 6 základných blokov:

- Free Space Path Loss - redukuje amplitúdu vstupného signálu o stanovenú časť
- I/Q Imbalance - Vytvára komplexne pásmový model poruchy signálu spôsobenej medzi fázovou a kvadratúrnou zložkou prijímača.
- Memoryless Nonlinearity - používa bezpamäťové nelinearity pri komplexných baseband signáloch
- Phase Noise - používa fázový šum prijímača na komplexný baseband signál
- Phase/Frequency Offset - používa fázové a frekvenčné posuny
- Receiver Thermal Noise - používa tepelný šum prijímača

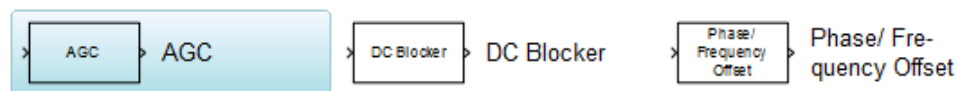


Obr. 33 Bloky RF porúch

### 3.3.12 Opravy RF porúch

Táto knižnica obsahuje 3 bloky:

- AGC - adaptívne nastavuje zisk pre konštantný výstupný signál
- DC Blocker - odstraňuje jednosmerné komponenty vstupného signálu
- Phase/Frequency Offset - aplikuje základný fázový a frekvenčný posun pre komplexný baseband signál

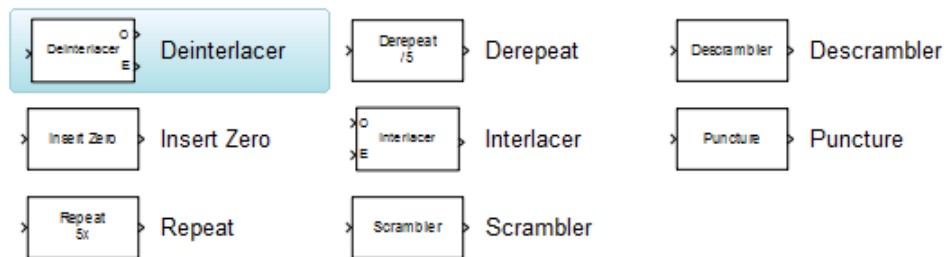


Obr. 34 Bloky opravujúce RF poruchy

### 3.3.13 Sekvenčné operácie

obsahujú nasledujúce bloky:

- Deinterlacer - striedavo rozdeľuje prvky vstupného vektora medzi dva výstupné vektory
- Derepeat - znižuje vzorkovaciu frekvenciu priemerovaním po sebe idúcich vzoriek
- Descrambler - dekóduje vstupný signál
- Insert Zero - rozdeľuje vstupné prvky výstupného vektora
- Interlacer - striedavo vyberá prvky z dvoch vstupných vektorov na generovanie výstupného vektora
- Puncture - výstupné prvky, ktoré korešpondujú 1s v binárnom Puncture vektore
- Repeat - prevzorkovanie vstupu na vyššiu frekvenciu opakovaním hodnôt
- Scrambler - kóduje vstupný signál



Obr. 35 Bloky sekvenčných operácií

### 3.3.14 Synchronizácia

Pre správne interpretovanie informácií musí byť komunikačný prijímač synchronizovaný s príslušným vysielačom. Digitálny prijímač musí vzorkovať signál vo vhodnom okamihu periódy signálu a musí vedieť odhadnúť nosnú fázu. Prípadne môžu analógové komponenty, ako napät'ovo riadený osciloskop (VCOs) a spät'no-väzobná slučka (PLLs) povoliť prijímaču prispôbiť svoje správanie na základe parametrov vstupných či požadovaných signálov.

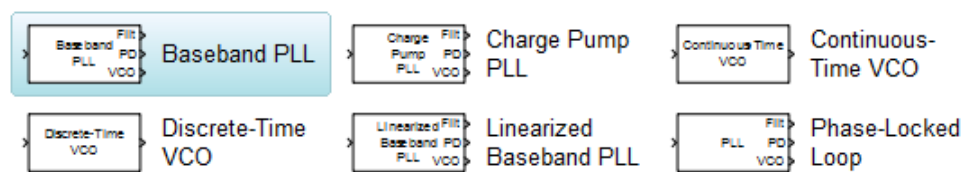
Knižnica synchronizácia sa ďalej delí na 3 podskupiny : **obnovenie nosnej fázy, komponenty** a **obnovenie časovej fázy**.

**Nosná fáza oživenia** (Carrier Phase Recovery) - spadajú pod ňu tieto bloky:

- CPM Phase Recovery - obnovuje nosnú fázu použitím 2P-Power metódy
- M-PSK Phase Recovery - obnovuje nosnú fázu použitím M-Power metódy

**Komponenty** - táto podknížnica obsahuje modely ako napät'ovo riadený oscilátor (VCO) ako aj spätno-väzobnú slučku (PLL).

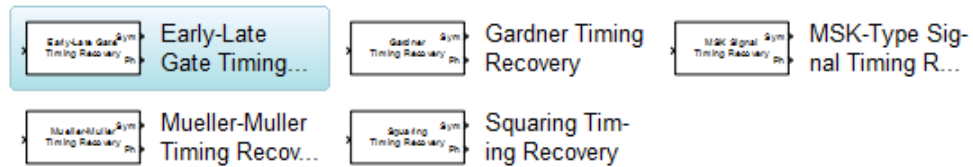
- Baseband PLL - implementuje spätnú slučku základného pásma
- Charge Pump PLL - implementácia charge pump na PLL
- Continuous-Time VCO - implementácia napät'ovo riadeného oscilátora
- Discrete-Time VCO - implementácia VCO v diskretnom čase
- Linearized Baseband PLL - implementuje linearizovanú verziu Basebanded PLL
- Phase-Locked Loop - implementuje PLL na obnovenie fázy vstupného signálu



Obr. 36 Synchronizačné VCO a PLL komponenty

**Obnovenie časovej fázy** - táto knižnica obsahuje bloky imlementujúce rôzne algoritmy pre zistenie najlepšieho okamihu vzorkovania signálu na prijímači. Napríklad pre PSK modulovaný signál je najlepší okamih na vrchole krivky pulzu. Vzorkovanie v správnych okamihoch zlepšuje výkon prijímača i pri zašumenom signáli. Zvyčajne by sme mali bloky na obnovu časovej fázy ihneď za prijímací filter, ktorý je prispôsobený prenosovému pulzu a pred demodulátor. Táto knižnica obsahuje 5 blokov:

- Early-Late Gate Timing Recovery - obnovuje symbol časovej fázy pomocou early-late gate metódy
- Gardner Timing Recovery - obnovuje symbol časovej fázy pomocou Gardnerovej metódy
- MSK-Type Signal Timing Recovery - obnovuje symbol časovej fázy pomocou štvrtej v poradí nelineárnej metódy
- Mueller-Muller Timing Recovery - obnovuje symbol časovej fázy pomocou Muller=Mullerovej metódy
- Squaring Timing Recovery - obnovuje symbol časovej fázy pomocou metódy porovnávania

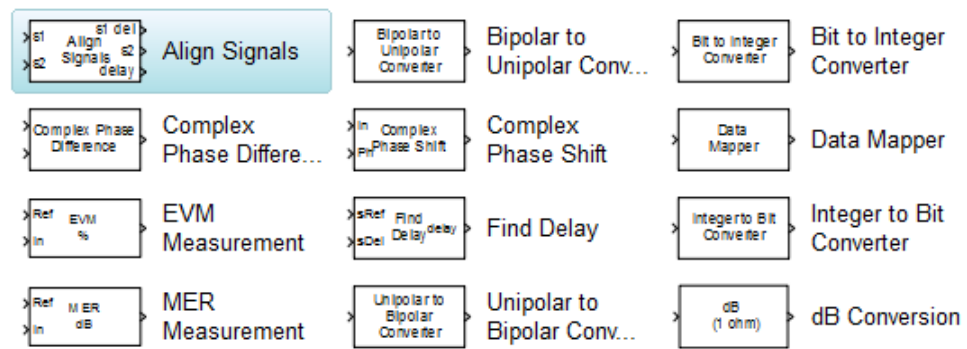


Obr. 37 Bloky obnovenia časovej fázy

### 3.3.15 Utility Blocks

Posledná knižnica, ktorá obsahuje niektoré ďalšie užitočné bloky, ktoré neboli zaradené v žiadnej z predchádzajúcich knižníc. Patrí sem:

- Align Signals - usporiada 2 signály a hľadá rozdiely medzi nimi. Je to veľmi užitočné pri porovnávaní odosielaného a prijatého signálu a zisťovanie chybovosti prenosu
- Bipolar to Unipolar Converter - premieňa bipolárny signál na unipolárny v rozmedziach od 0 po M-1
- Bit to Integer Converter - konvertuje vektor bitov na zodpovedajúci vektor prirodzených čísel
- Complex Phase Difference - výstupnú fázu rozdeľuje medzi 2 komplexné vstupné signály
- Complex Phase Shift - posúva fázu komplexného vstupného signálu druhou vstupnou hodnotou
- Data Mapper - mapuje prirodzené symboly z jednej kódovacej schémy na inú
- EVM Measurement - počíta rozdiel veľkosti vektoru medzi ideálnym referenčným signálom a meraným signálom
- Find Delay - hľadá oneskorenia medzi dvomi signálmi
- Integer to Bit Converter - premieňa vektor prirodzených čísel na vektor bitov
- MER Measurement - meria pomer signálu k šumu (SNR) v digitálnych modulačných aplikáciach
- Unipolar to Bipolar Converter - premieňa unipolárny signál s hodnotami od 0 po M-1 na bipolárny signál
- dB Conversion - prevádza rozsah dát na decibely (dB alebo dBm) [13]



Obr. 38 Další užitočné bloky použiteľné v komunikačných systémoch

## **II. PRAKTICKÁ ČÁST**

## 4 SIMULAČNÝ MODEL BLOKOVÉHO KÓDOVANIA

Každá správa kódového slova je usporiadaná do jednotlivých blokov symbolov. Každý blok predstavuje v každom časovom kroku jedno slovo. Reed-Solomonova kódovacia metóda umožňuje kódovať ako binárne, tak dáta prirodzených čísel.

Pri binárnom formáte prijaté správy a kódované slová predstavujú binárne vektory. Vektor binárnej správy musí mať veľkosť  $\mathbf{K}$  a vektor kódového slova musí mať veľkosť  $\mathbf{N}$ . Pri Reed-Solomonových vstupných kódoch tvoria symboly kódu binárne sekvencie dĺžky  $\mathbf{M}$ . Vektor správy teda musí mať veľkosť  $\mathbf{M}*\mathbf{K}$  a zodpovedajúci kódovaný vektor  $\mathbf{M}*\mathbf{N}$ .

Ak vstupný blok tvorí rámcový signál, tento vektor musí byť stĺpcový.

Aj pri kódovaní formátu prirodzených čísel obsahuje vektor správy  $\mathbf{M}*\mathbf{K}$  bitov, ktoré predstavujú hodnoty medzi 0 a  $2^{\mathbf{M}}$ . Štruktúra správy i kódové slová zostávajú vo formáte prirodzených čísel. Vstupný signál tvoria tak isto stĺpcové vektory.

Pre vytvorenie vzorkovanej správy formátu prirodzených čísel môžeme použiť blok **Random Integer Generator**. Môžeme nakonfigurovať parametre ako M-hodnotu i vstupný prvok, ktoré predstavujú vektory s požadovanou dĺžkou.

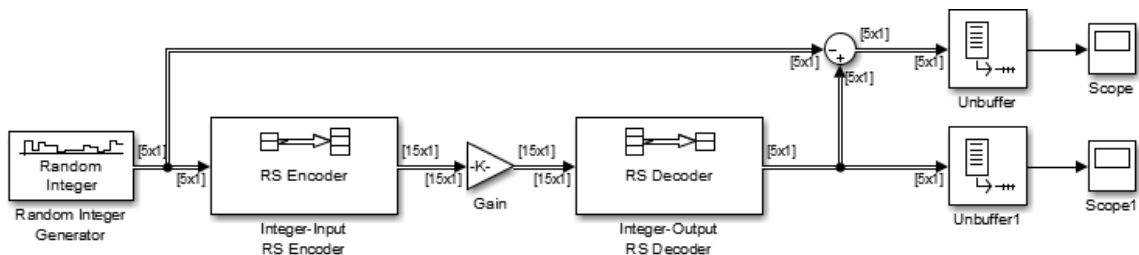
### 4.1 Reed-Solomonov model kódovania prirodzených čísel

Pre vytvorenie nového modelu simulácie otvoríme knižnicu Simulinku. V pravom hornom rohu klikneme na ikonu "New model", čím sa nám otvorí nové prázdne okno, do ktorého budeme vkladat' jednotlivé bloky. Dvojklikom na každý blok nastavíme jeho vlastnosti. Pre tento model budeme potrebovať nasledovné bloky:

- **Random Integer Generator** - knižnica Comm Sources
  - do poľa M-ary zadáme hodnotu 15
  - do Initial seed zadáme ľubovoľné kladné číslo
  - zaškrkneme check box Frame-based outputs
  - Samples per frame nastavíme na 5
- **Integrer-Input RS Encoder** - knižnica Error Detection and Correction/Block
  - Codeword length N nastavíme na 15
  - Message length K nastavíme na 5
- **Gain** - Simulink / knižnica Commonly Used Blocks
  - do Gain zadame stĺpcový vektor [0; 0; 0; 0; 0; ones(10,1)]

- **Integer-Output RS Decoder** - knižnica Error Detection and Correction/Block
  - Codeword length N nastavíme na 15
  - Message length K nastavíme na 5
- **Sum** - Simulink / knižnica Commonly Used Blocks
  - List of sings nastavíme na |-+
- **Unbuffer** x2 - DSP System Toolbox / knižnica Signal Management / Buffers
- **Scope** x2 - Simulink / knižnica Commonly Used Blocks

Bloky rozostavíme podľa a pospájame nasledovnej schémy. Ak podržíme kurzor myši na vstupných / výstupných portoch každého bloku, zobrazí sa nám znak "+". Kliknutím si daný port označíme a potiahneme kurzor na požadovaný ďalší port, s ktorým chceme predchádzajúci port prepojiť.

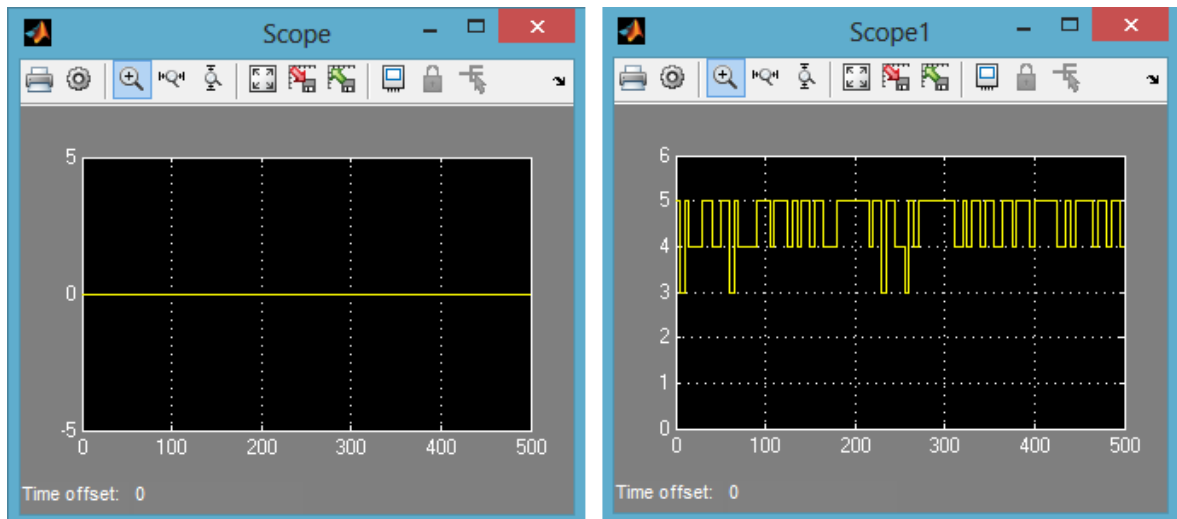


Obr. 39 Reed-Solomonov model kódovania prirodzených čísel

Pomocou menu **Simulation** → **Model Configuration Parameters** nastavíme **Stop Time** na hodnotu 500. Ak si chceme zobrazit veľkosť vektorov pri jednotlivých blokoch, použijeme menu **Display** → **Signal & Ports** a označíme **Signal Dimensions**.

Simuláciu spustíme pomocou zeleného tlačidla "**Play**" uprostred horného menu. Kliknutím na bloky **Scope** zobrazíme jednotlivé vykreslené priebehy.

Druhý graf znázorňuje počet chýb, ktoré detektor objavil. Vidíme že najviac hodnôt zodpovedá hodnote 5. Je to spôsobené blokom **Gain**, v ktorom sme nastavili prvých 5 symbolov v kódovom slove na hodnotu 0. Ak však správne kódové slovo obsahovalo jednu, alebo viac núl, číslo chýb bolo nižšie. Z nasledujúcich grafov môžeme vidieť že počet chýb osciloval medzi hodnotami 3 až 5.



Obr. 40 Výsledná charakteristika chybovosti prenosu Reed-Solomonovho kódu

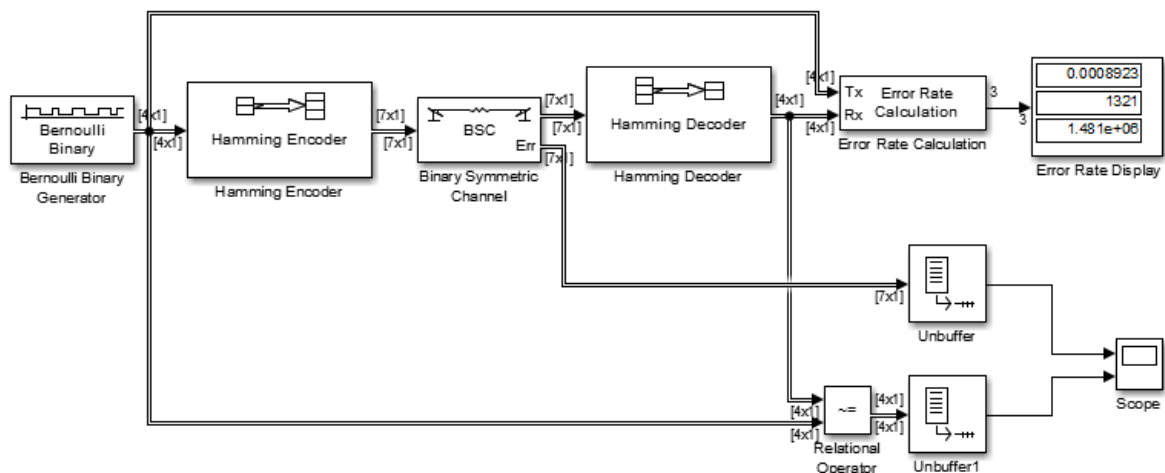
## 4.2 Simulačný model prenosu pomocou Hammingovho kódovania

V ďalšom príklade si ukážeme simulačný model blokového kódovania pomocou **Hammingovho kódu**. Ide o jednoduché zapojenie kodéra i dekodéra. Ukážeme si vhodné dĺžky vektorov správy i kódových slov. Keďže blok výpočtu chybovosti prenosu akceptuje len skalár, alebo rámcový signál, podobne ako v predchádzajúcom príklade použijeme stĺpcové vektory.

Pre urýchlenie práce môžeme otvoriť už preddefinovaný model prenosového kanálu, do ktorej vložíme kódovacie i dekodovacie bloky.

V príkazovom riadku MATLABu zadáme príkaz `doc_channel`. Otvorí sa nám nové okno s prenosovým modelom. Bloky si trochu poposúvame, čím si vytvoríme viac miesta pre kódovacie a dekodovacie prvky. V Simulink knižnici blokov si vyhľadáme sekciu Error Detection and Correction / Block, odkiaľ si do nášeho prenosového bloku vložíme **Hamming Encoder** a **Hamming Decoder**. Blok Hamming Encoder vložíme medzi **Bernoulliho Generátor** a **Binary Symmetric Channel**. Hamming Decoder potom medzi **Binary Symmetric Channel** a **Error Rate Calculation**.

Do nášho modelu si môžeme vložiť i blok **Scope** pre detailné vykreslenie výslednej chybovosti. Okrem toho vložíme aj 2x **Unbuffer** a 1x **Relational Operator**. Budeme mať teda 2 výstupné zariadenia - **Display** a **Scope**. Všetky bloky si zoradíme a pospájame podľa nasledujúceho modelu:



Obr. 41 Simulačný model prenosu pomocou Hammingovho kódu

Nastavovanie parametrov pre použité bloky modelu:

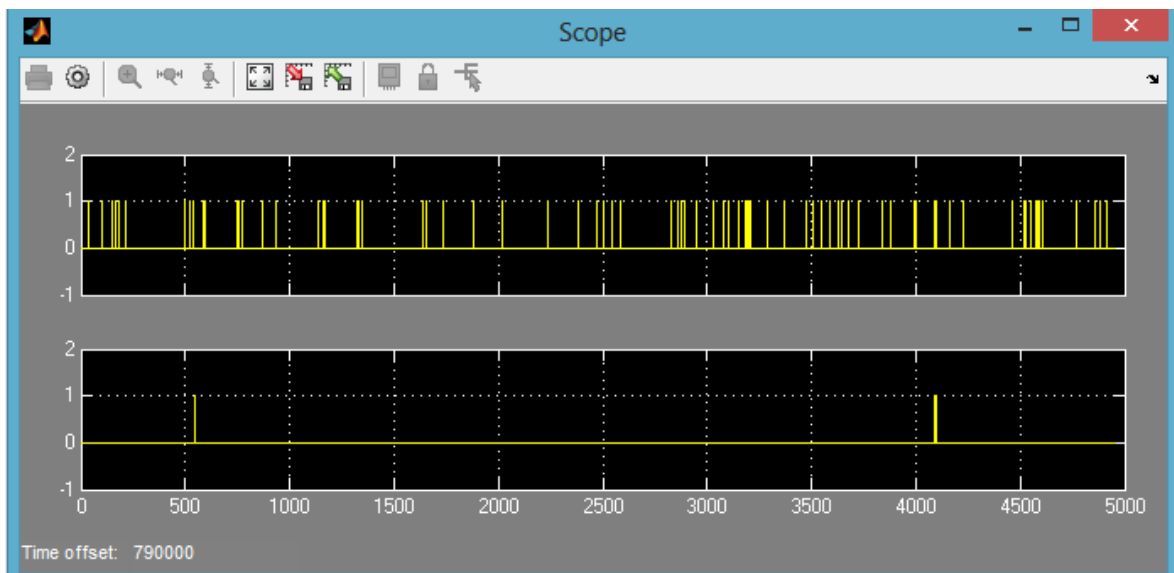
- **Bernoulli Binary Generator**
  - Označíme check box Frame-based outputs
  - Samples per frame nastavíme na 4
- **Binary Symmetric Channel**
  - Označíme check box Output error vector - zobrazí sa nám ďalší výstupný port, ktorý prepojíme s blokom Unbuffer
- **Scope**
  - klikneme na menu Parameters a nastavíme Number of axes na 2 - zobrazia sa nám 2 vstupné porty, ktoré prepojíme s výstupmi oboch Unbufferov
  - V menu Parameters nastavíme Time range na 5000
  - V záložke Data history zadáme do poľa Limit data points to last hodnotu 30000
  - klikneme na Ok a zavrieme menu Parameters, vidíme 2 vykresľovacie okná.
  - Klikneme na vertikálnu os pravým tlačidlom a otvoríme Axes properties
  - Hodnotu Y-min nastavíme na -1 a Y-max na 2.
  - Postup zopakujeme aj pri druhom vykresľovacom okne

- **Error Rate Calculation**
  - Odznačíme check box Stop simulation
- **Relational Operator**
  - Relational Operator nastavíme na " ~= "

Na záver si môžeme ešte zobrazit' veľkosť prenášaných vektorov pomocou menu **Display** → **Signal & Ports** a označíme **Signal Dimensions**.

Popisy jednotlivých blokov si zmeníme jednoducho kliknutím myšou. Editujú sa ako klasické textové polia.

Pred spustením simulácie odporúčam otvoriť si blok Scope a sledovať ako sa v priebehu simulácie vykresľuje chybovosť.

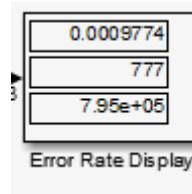


Obr. 42 Výsledná charakteristika chybovosti prenosu Hammingovho kódu

Na spodnom priebehu vidíme signál z Relation Operatoru, ktorý porovnáva priamo vygenerovaný signál so signálom z Hammingovho dekodéru. Ak sú oba signály totožné, vykreslí sa nula. Ak totožné nie sú, čiže v prenose nastala chyba, ktorá nebola opravená, vykreslí sa 1.

Vrchný graf vykresľuje chyby prenosu generované blokom Binary Symmetric Channel. Z nášho grafu vidíme že pri tom množstve chýb z vrchného grafu len dve neboli opravené.

Presné číselné hodnoty sa nám zobrazujú v bloku Display:



Obr. 43 Výsledné hodnoty chybovosti prenosu

Z tohoto bloku je zřejmé že chybovosť nie je ani 0,001, konkrétne má hodnotu **0,0009774**.

Ak by sme si simuláciu spustili viackrát, chybovosť prenosu by vždy kolísala okolo hodnoty 0,001. Pravdepodobnosť vzniku 2 alebo viac chýb počas prenosu v kódovom slove o veľkosti 7 je:

$$1 - (0,99)^7 - 7(0,99)^6(0,01) = 0,002$$

Ak sú kódové slová s dvoma alebo viacero chybami náhodne dekódované, môžeme očakávať že asi polovica bitov dekódovaného slova je chybná. Znamená to že práve 0,001 je rozumná hodnota chybovosti.

## ZÁVER

Cieľom tejto práce bolo oboznámiť sa s programovým balíkom MATLAB, jeho pracovným prostredím, základnými operáciami a funkciami. Z nadobudnutých poznatkov vytvoriť akýsi užívateľský manuál. Zamerať sa na Communications System Toolbox, a popísať všetky jeho bloky. Na základe získaných znalostí z tejto problematiky i z problematiky komunikačných systémov navrhnúť možnosti použitia tohto toolboxu na pre potreby výučby predmetu Telekomunikačné systémy.

V teoretickej časti som objasnil pracovné prostredie MATLABu s jeho jednotlivými oknami. Mal som možnosť pracovať ako s verziou R2010b, tak i R2014a, takže som mohol porovnať jednotlivé prostredia a vykresliť základné rozdiely. Pre porovnanie som priložil i obrázok MATLABu verzie 6x z roku 2001.

V ďalších kapitolách som rozoberal prácu v prostredí MATLAB. Popísal som premenné a možnosti ich použitia i s nimi súvisiace operácie. Začal som pri skalárnych a postupne cez vektorové som prešiel až k maticovým operáciám. Stručne som zhrnul i možnosti vykresľovania výstupných dát použitím 2-D i 3-D grafov.

V rámci teoretickej časti som pokračoval charakterizovaním programu Simulink a popísal som jeho najpoužívanejšie toolboxy. Jedným z nich je práve Communications System Toolbox, určený na simulácie prenosov v komunikačnom kanáli. Zvláštnu pozornosť som venoval jednotlivým jeho blokom. Všetky bloky sú rozdelené do 15 hlavných knižníc, pričom každá z nich plní špecifickú funkciu v komunikačnom kanáli.

V praktickej časti som vytvoril simulačné modely na skúmanie chybovosti prenosu. Obidva modely vychádzajú z blokového kódovania komunikačného kanála, pričom prvý model využíva Reed-Solomonov kód a druhý Hammingov kód. Pre lepšie pochopenie daných modelov som najskôr popísal vlastnosti takéhoto kódovania. V každom modeli som vypísal potrebné bloky a ich zapojenie podľa priloženej schémy. Všetky bloky som upravil podľa požadovaných parametrov a začal simuláciu. Následne výsledky som zobrazil pomocou bloku Display, alebo Scope pre grafický výstup.

V priebehu vypracovávania tejto diplomovej práce som nadobudol cenné teoretické i praktické znalosti. Pracovné prostredie MATLABu je veľmi príjemné. Aj keď je navrhnutý pre zložité vedecké projekty, práca v ňom je jednoduchá a veľmi prehľadná. Communications System Toolbox obsahuje množstvo nástrojov pre simulovanie najrôznejších komunikač-

ných modulov. Ja som uviedol dva moduly, ale podobné príklady by sa dali vytvoriť pre každý druh kódovania popisovaný v predmete Telekomunikačné systémy. Študenti by získali väčší prehľad o jednotlivých komunikačných kanáloch ich blokoch. Na základe výstupnej charakteristiky by navyše mohli porovnať priebehy a určiť účinnosť kódovania.

**SEZNAM POUŽITÉ LITERATURY**

- [1] KNIGHT, Andrew. Basics of MATLAB and beyond. Boca Raton, Fla: Chapman, 2000. ISBN 978-142-0048-162
- [2] KALNINS, Lara M. MATLAB Basics: Navigation and Tools. MATLAB Basics: Navigation and Tools [online]. 2010 [cit. 2014-05-20]. Dostupné z: <http://www.earth.ox.ac.uk/~larak/practicals/MatlabBasics.pdf>
- [3] MATLAB hypertext reference. MATLAB hypertext reference [online]. 1995 [cit. 2014-05-20]. Dostupné z: <http://web.cecs.pdx.edu/~gerry/MATLAB/masterOutline.html>
- [4] Simple Calculations with MATLAB. An Introduction to Programming and Numerical Methods in MATLAB [online]. London: Springer-Verlag, 2005, s. 1 [cit. 2014-05-20]. DOI: 10.1007/1-84628-133-4\_1. Dostupné z: [http://link.springer.com/10.1007/1-84628-133-4\\_1](http://link.springer.com/10.1007/1-84628-133-4_1)
- [5] MAJEROVA. MATLAB 2D grafika [online]. 2008 [cit. 2014-05-20]. Dostupné z: <http://uprt.vscht.cz/majerova/matlab/lekce5.html>
- [6] MAJEROVA. MATLAB 3D grafika [online]. 2008 [cit. 2014-05-20]. Dostupné z: <http://uprt.vscht.cz/majerova/matlab/lekce8.html>
- [7] HUMUSOFT. SIMULINK: Simulace a modelování dynamických systémů, Model-Based Design [online]. 2014 [cit. 2014-05-20]. Dostupné z: <http://www.humusoft.cz/produkty/matlab/simulink/>
- [8] MATHWORKS. SIMULINK: Key Features [online]. 2014 [cit. 2014-05-20]. Dostupné z: <http://www.mathworks.com/products/simulink/features.html>
- [9] MATHWORKS, Inc. Simulink: Model-based and System-based Design [online]. 2002 [cit. 2014-05-20]. Dostupné z: <http://books.google.sk/books?id=OtRQAAAAMAAJ>
- [10] HUMUSOFT. Aplikační knihovny pro MATLAB, Simulink [online]. 2014 [cit. 2014-05-20]. Dostupné z: <http://www.humusoft.cz/produkty/matlab/aknihovny/>
- [11] MATHWORKS, Inc. Communications System Toolbox: Design and simulate the physical layer of communication systems [online]. 2014 [cit. 2014-05-20]. Dostupné z: <http://www.mathworks.com/products/communications/>

- [12] PROAKIS, JOHN G, Masoud SALEHI a Gerhard BAUCH. Contemporary communication systems using MATLAB and Simulink. 2nd ed. Belmont: Brooks/Cole-Thompson Learning, 2004, 487 s. ISBN 05-344-0617-3.
- [13] MATHWORKS, Inc. Blocks in Communications System Toolbox [online]. 2014 [cit. 2014-05-20]. Dostupné z:  
<http://www.mathworks.com/help/comm/blocklist.html>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

AGC	Adaptively adjust gain for constant
AM	Amplitúdová modulácia
APP	A posteriori probability
atd.	A tak ďalej
AWGN	Add White Gaussian Noise
BCH	Bose - Chaudhuri - Hocquenghem
BPSK	Binary Phase-shift keying
CPM	Continuous phase modulation
CRC	Cyclic redundancy check
CRC-N	Cyclic redundancy check - Nth degree
DSBSC	Double-sideband suppressed-carrier
DSP	Digital signal processing
EVM	Error Vector Magnitude
FIR	Finite Impulse Response
FM	Frekvenčná modulácia
FSK	Frequency-shift keying
HDL	Hardware description language
LDPC	Low-density parity-check code
LMS	Least mean squares
LTE	Long Term Evolution
MER	Modulation Error Ratio
MIMO,	Multiple Input, Multiple Output
MLSE	Význam třetí zkratky.
M-PSK	Maximum likelihood sequence estimate

---

MSK	Minimum shift keying
OFDM	Orthogonal frequency division modulation
OFDMA	Orthogonal Frequency-Division Multiple Access
OQPSK	Offset Quadrature Phase-shift keying
OSTBC	Orthogonal space-time block code
PAM	Pulse amplitude modulation
PLL	Phase-locked loop
PM	Phase modulation
PSK	Phase-shift keying
QAM	Quadrature amplitude modulation
RS	Reed-Solomon
SISO	Single Input, Single Output
SSB	Single-sideband
TCM	Trellis-coded modulation
tzv.	Takzvané
VCO	Voltage-controlled oscillator
WiFi	Wireless fidelity
WiMAX	Worldwide Interoperability for Microwave Access

**SEZNAM OBRÁZKŮ**

Obr. 1: Pracovné prostredie programu MATLAB R2010b .....	11
Obr. 2: Ukázkový matematický príkaz .....	12
Obr. 3 Okno náповedy v MATLABe .....	14
Obr. 4 Ikona Simulinku v prostredí R2010b a 2014a .....	14
Obr. 5 Porovnanie rozloženia okien verzie 6x, R2010b a R2014a.....	15
Obr. 6 Vykreslenie závislosti 1 premennej .....	27
Obr. 7 Vykreslenie závislosti 3 premenných .....	28
Obr. 8 Použitie funkcie subplot .....	29
Obr. 9 Použitie funkcie plot3 - čiarový graf .....	29
Obr. 10 Príklad plošného grafu pomocou príkazu surf.....	30
Obr. 11 Základné okno nástroja Simulink .....	31
Obr. 12 Loptička je vyhodená rýchlosťou 15 m/s z výšky 10 m.....	34
Obr. 13 Realizácia simulácie poskakujúcej loptičky .....	35
Obr. 14 Výsledné charakteristiky pozície $x$ a rýchlosti v poskakujúcej loptičky .....	36
Obr. 15 Communications System Toolbox a jeho bloky.....	38
Obr. 16 Bloky určené pre simuláciu podmienok v komunikačnom kanáli .....	39
Obr. 17 Bloky obsiahnuté v skupine Source Coding.....	39
Obr. 18 Skupina filtračných blokov komunikačných systémov .....	40
Obr. 19 Bloky knižnice Comm Sink.....	41
Obr. 20 Bloky generujúce šum v knižnici Noise Generators .....	41
Obr. 21 Bloky obsiahnuté v knižnici Random Data Source .....	42
Obr. 22 Bloky obsiahnuté v knižnici Sequence Generators .....	42
Obr. 23 Ekvalizačné bloky Communications System Toolboxu.....	43
Obr. 24 Blokované kódovacie techniky .....	44
Obr. 25 Blokované kódovacie techniky .....	45
Obr. 26 Bloky implementujúce CRC kódovací algoritmus .....	46
Obr. 27 Bloky určené na riešenie konvolučných algoritmov .....	46
Obr. 28 Bloky implementujúce blokované prekladanie .....	47
Obr. 29 Schéma konvolučne prekladacích blokov .....	48
Obr. 30 Základné bloky konvolučného prekladania.....	48
Obr. 31 MIMO bloky.....	49
Obr. 32 Analógové a digitálne modulačné bloky .....	49

---

Obr. 33 Bloky RF porúch .....	50
Obr. 34 Bloky opravujúce RF poruchy.....	50
Obr. 35 Bloky sekvenčných operácií .....	51
Obr. 36 Synchronizačné VCO a PLL komponenty .....	52
Obr. 37 Bloky obnovenia časovej fázy .....	53
Obr. 38 Ďalšie užitočné bloky použiteľné v komunikačných systémoch.....	54
Obr. 39 Reed-Solomonov model kódovania prirodzených čísel .....	57
Obr. 40 Výsledná charakteristika chybovosti prenosu Reed-Solomonovho kódu .....	58
Obr. 41 Simulačný model prenosu pomocou Hammingovho kódu .....	59
Obr. 42 Výsledná charakteristika chybovosti prenosu Hammingovho kódu .....	60
Obr. 43 Výsledné hodnoty chybovosti prenosu .....	61

## SEZNAM PŘÍLOH

PI Použité grafické súbory

PII Simulačné modely použité v praktickej časti