Moderné vzdelávanie pre vedomostnú spoločnosť/
Projekt je spolufinancovaný zo zdrojov EÚ

SIMULAČNÉ SYSTÉMY V KYBERNETIKE

Fakulta elektrotechniky a informatiky

Anna Jadlovská, Slávka Jadlovská

Európska únia
Európsky sociálny fond
Agentúra Ministerstva školstva, vedy, výskumu a športu SR pre štrukturálne fondy EÚ
Operačný program VZDELÁVANIE

Táto publikácia vznikla za finančnej podpory z **Európskeho sociálneho fondu** v rámci Operačného programu **VZDELÁVANIE.**

Prioritná os 1 Reforma vzdelávania a odbornej prípravy

Opatrenie 1.2 Vysoké školy a výskum a vývoj ako motory rozvoja vedomostnej spoločnosti.

Názov projektu: **Balík doplnkov pre ďalšiu reformu vzdelávania na TUKE**

ITMS 2611020093

NÁZOV: Simulačné systémy v kybernetike
AUTORI: doc. Ing. Anna Jadlovská, CSc, Ing. Slávka Jadlovská
VYDAVATEĽ: Technická univerzita v Košiciach
ROK: 2015
VYDANIE: prvé
NÁKLAD: 70 ks
ROZSAH: 240 strán
ISBN: 978-80-553-2011-3

Moderné vzdelávanie pre vedomostnú spoločnosť/
Projekt je spolufinancovaný zo zdrojov EÚ

SIMULAČNÉ SYSTÉMY V KYBERNETIKE

Fakulta elektrotechniky a informatiky

Anna Jadlovská, Slávka Jadlovská

Obsah

Úvod

Základným krokom k riešeniu mnohých vedeckých problémov býva vytvorenie modelu skúmaného problému. **Modelovanie** je činnosť vedúca k nájdeniu matematického alebo fyzikálneho modelu, ktorý svojimi vlastnosťami charakterizuje skúmaný systém.

Matematický model je sústava rovníc, ktoré s požadovanou presnosťou opisujú (zvolené) statické a dynamické vlastnosti systému. Pri zostavovaní matematických modelov obvykle uvažujeme s rôznymi zjednodušeniami s cieľom, aby výsledné vzťahy neboli neúmerne zložité. Zjednodušujúce predpoklady, ktoré použijeme pri zostavovaní matematického modelu skúmaného systému spôsobujú, že model je len aproximáciou reálneho systému. **Analytická identifikácia** dynamického systému je proces, ktorým získavame matematický model daného systému vo forme sústavy lineárnych/nelineárnych diferenciálnych, diferenčných alebo iných rovníc.

Simulácia je experimentovanie s modelom systému s cieľom analýzy jeho dynamických vlastností. Nakoľko simulačné experimenty sa realizujú hlavne na číslicových počítačoch, z toho vyplýva, že je nutné z matematického modelu skúmaného systému získať **simulačný model** implementáciou do vhodného programového prostredia**.**

Môžeme konštatovať, že modelovanie a simulácia svojou podstatou prirodzene predstavujú prienik mnohých vedeckých disciplín – matematiky, fyziky, teórie automatického riadenia a ďalších technických vied.

V predkladanom učebnom texte "**Simulačné systémy v kybernetike"** sú všetky odvodené statické/dynamické matematické modely systémov metódami analytickej identifikácie implementované do simulačného jazyka **MATLAB** alebo programového prostredia **Simulink** s cieľom riešiť definované úlohy Kybernetiky.

Učebný text "**Simulačné systémy v kybernetike**" je určený hlavne pre poslucháčov II. ročníka prvého stupňa štúdia odboru **Kybernetika** študujúcich na FEI TU v Košiciach.

Hlavným cieľom učebného textu, ktorý svojim obsahom pokrýva prednášky a cvičenia z predmetu "**Simulačné systémy**" je zoznámiť študentov so základmi **modelovania** a **simulácie** fyzikálnych systémov v programovom prostredí MATLAB/Simulink na PC a tiež poukázať na dôležitosť **modelovania** a **identifikácie** v úvodných fázach vedeckej práce, keď sa formulujú hypotézy, ktoré sa ďalej testovaním prijímajú alebo zamietajú.

Ku splneniu vytýčeného cieľa napomáhajú študentom ilustračné riešené príklady v simulačnom jazyku MATLAB/Simulink nachádzajúce sa v jednotlivých kapitolách učebného textu a tiež vypracované vzorové zadania (Z1, Z2, Z3, Z4, Z5), ktoré tvoria ucelený pohľad na riešenie aplikačných úloh z kybernetiky počnúc od analýzy úlohy, výberu vhodných metód riešenia, návrhu algoritmického riešenia problému, zostavenia simulačného modelu a verifikácie navrhnutého riešenia pomocou simulácie.

Štruktúra a obsahová náplň učebného textu je zvolená tak, že po jej preštudovaní by študenti mali vedieť:

- zostavovať matematické modely jednoduchých dynamických systémov na základe fyzikálnej analýzy (voľba vstupov/výstupov systému, výber vhodných fyzikálnych zákonov popisujúcich dynamický systém, **kap.5, kap.6**.,)
- zostaviť stavový opis dynamického systému (prepis nelineárnych/lineárnych diferenciálnych rovníc popisujúcich fyzikálny systém do substitučného kanonického tvaru, **kap.5, kap.6**, **kap.8**),
- navrhnúť algoritmické riešenie pre definované úlohy modelovania statických a dynamických systémov, (**kap.2, kap.3, kap.4**)
- navrhnúť a naprogramovať simulačné modely fyzikálnych systémov v programovom prostredí **MATLAB/Simulink** s cieľom overenia algoritmického riešenia definovaných úloh pomocou simulácie, (**kap.5, kap.6**, **kap.8**),)
- využívať na podporu tvorby matematických modelov a ich simulácie symbolické výpočty pomocou funkcií **Symbolic Math Toolbox-u**, (**kap. 10**)

- ovládať základné metódy pre analýzu lineárnych dynamických systémov s využitím funkcií **Control System Toolbox-u, (kap.7)**
- ovládať základné metódy pre návrh a simuláciu algoritmov riadenia lineárnych dynamických systémov v spätnoväzobnej riadiacej štruktúre v simulačnom jazyku MATLAB/Simulink a s využitím funkcií **Control System Toolbox-u**, (**kap.7, kap.8**)
- orientovať sa v programovom/simulačnom prostredí MaATLAB/Simulink a funkciách **Symbolic Math Toolbox**, ktoré sú vhodné na riešenie aplikačných úloh numerickej matematiky a grafickej prezentácii získaných výsledkov, (**kap.1, kap.2, kap.3, kap.4**. **kap.9**)
- rozumieť princípom analýzy, modelovania a simulácie systémov hromadnej obsluhy (**kap.11**)

Štúdium publikácie predpokladá základné vedomosti z fyziky, matematickej analýzy, algebry, numerickej matematiky a teórie automatického riadenia lineárnych dynamických systémov v rozsahu aký je bežne prednášaný na fakultách technických skôl.

Obsah učebných textov je rozdelený do jedenástich kapitol, z ktorých každá tvorí základ minimálne jednej prednášky a cvičenia pre predmet Simulačné systémy. V niektorych kapitolách učebného textu "**Simulačné systémy v kybernetike**" a v časti "**Prílohy**" sú použité modifikované riešené aplikačné úlohy z nasledujúcich bakalárskych prác:

- **Oravec, M.:** Simulácia modelov dynamických systémov s využitím webových aplikácií na báze .NET, *Bakalárska práca*, FEI TU, Košice, 2011, (*vedúca BP: doc.Ing. A. Jadlovská, PhD., konzultant: Ing. Š. Jajčišin*)
- **Tomčák, M**.: Spracovanie tutoriálov aplikačného využitia knižnice hydraulických systémov, Bakalárska práca, FEI TU, Košice, 2013, (*vedúca BP: doc. Ing. A. Jadlovská, PhD., konzultant: Ing. J. Čerkala*)
- **Novisedláková, D.**: Tvorba elektronických výukových materiálov pre predmet Simulačné systémy, Bakalárska práca, FEI TU, Košice, 2013, (*vedúca BP doc. Ing. A. Jadlovská, PhD., konzultant: Ing. S. Jadlovská*)
- **Dovcová, M.:** Vytvorenie výukových materiálov a testových modulov pre predmet Simulačné systémy v hospodárskej informatike, Bakalárska práca, FEI TU, Košice, 2013, (*vedúca BP: doc.Ing. A. Jadlovská, PhD., konzultant: Ing. Š. Jajčišin*).

Užitočné web stránky pre riešenie aplikačných úloh kybernetiky v programovom prostredí MATLAB/Simulink:

http://www.mathworks.com/ - web firmy MathWorks – výhradný predajca výpočtového prostredia MATLAB/Simulink

http://www.humusoft.cz/ - web firmy Humusoft - výhradný zástupca firmy MathWorks pre Českú a Slovenskú republiku

http://www-test.humusoft.cz/produkty/matlab/matlab-ve-skolstvi/ - programové prostredie MATLAB/Simulink v školstve

http://www.mathworks.com/matlabcentral/ - priestor pre vzájomnú komunikáciu medzi užívateľmi a priaznivcami systemu MATLAB/Simulink. Otvorená platforma pre prezentáciu vlastných aplikácií, výmenu súborov, názorov a skúseností

http://www.humusoft.cz/produkty/matlab/csmug/ - skupina užívateľov výpočtového prostredia MATLAB v Českej republike,

http://www.posterus.sk/?cat=7 – články venované systému MATLAB na stránkach internetového časopisu Posterus.sk

http://www.matlab.sk/ - diskusné forum so zameraním na vývojové a výpočtové prostredie MATLAB/Simulink

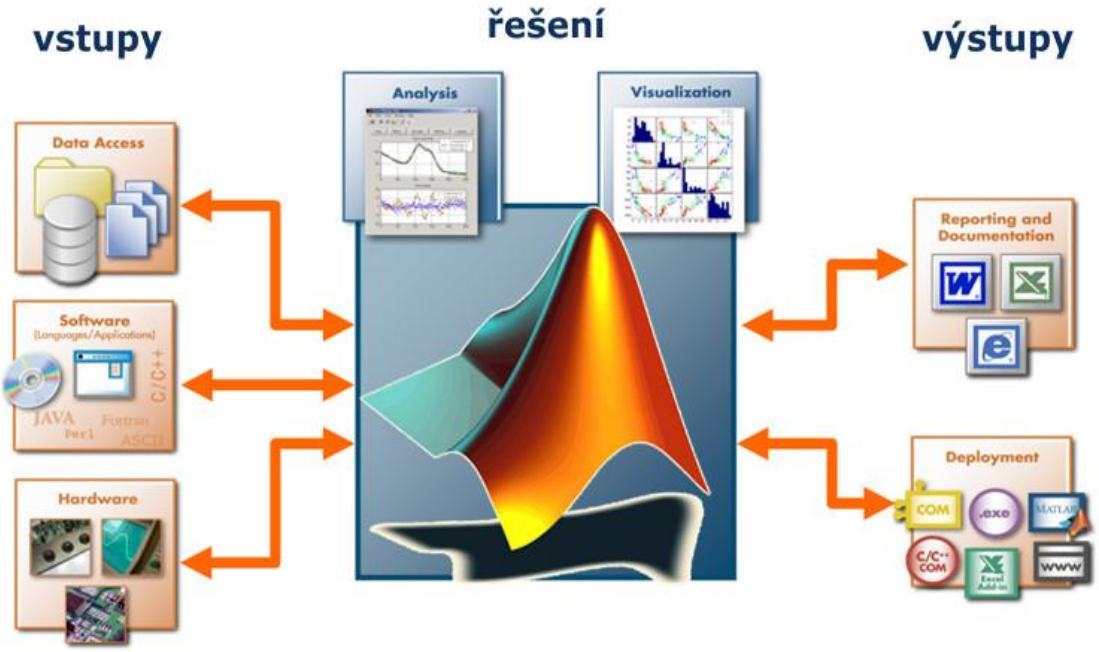
http://kyb.fei.tuke.sk/laboratoria/infdsr.php - web výskumného "Centra moderných metód a priemyselnej informatiky" (CMMRaPI),

http://kyb.fei.tuke.sk/laboratoria/predmet/sss.php - web stránka predmetu Simulačné systémy na KKUI FEI TU.

1 Modelovanie, simulácia a programové prostredie MATLAB/Simulink

- **Simulačný jazyk MATLAB**

⇨ **MATLAB** (MATrix LABoratory) je simulačný jazyk vyvinutý pre vedecko-technické výpočty, modelovanie, návrhy algoritmov, simuláciu, analýzu a prezentáciu dát, meranie a spracovanie signálov, návrhy riadiacich a komunikačných systémov.
⇨ Základ tvorí výpočtové jadro, ktoré je zamerané na operácie s maticami a preto je považované za najsilnejšiu stránku simulačného jazyka MATLAB s jeho optimálnymi algoritmami.
⇨ Jadro je rozšírené o množstvo nadstavieb (Toolboxov = aplikačné knižnice), ktoré sú určené na riešenie úloh z takmer všetkých oblastí ľudskej činnosti.



Obrázok 1-1 Výpočtový systém MATLAB/Simulink

⇨ Základné operácie simulačného jazyka MATLAB sú operácie s maticami
⇨ Umožňuje 2D a 3D vizualizáciu v grafoch s množstvom voliteľných a nastaviteľných parametrov
⇨ Umožňuje vytváranie vlastných funkcií a knižníc
⇨ Distribúcia užívateľských aplikácií napríklad do jazyka C
⇨ Obsahuje rozsiahlu dokumentáciu v PDF alebo help v on-line hypertextovej forme

- **Vývoj simulačného jazyka MATLAB**

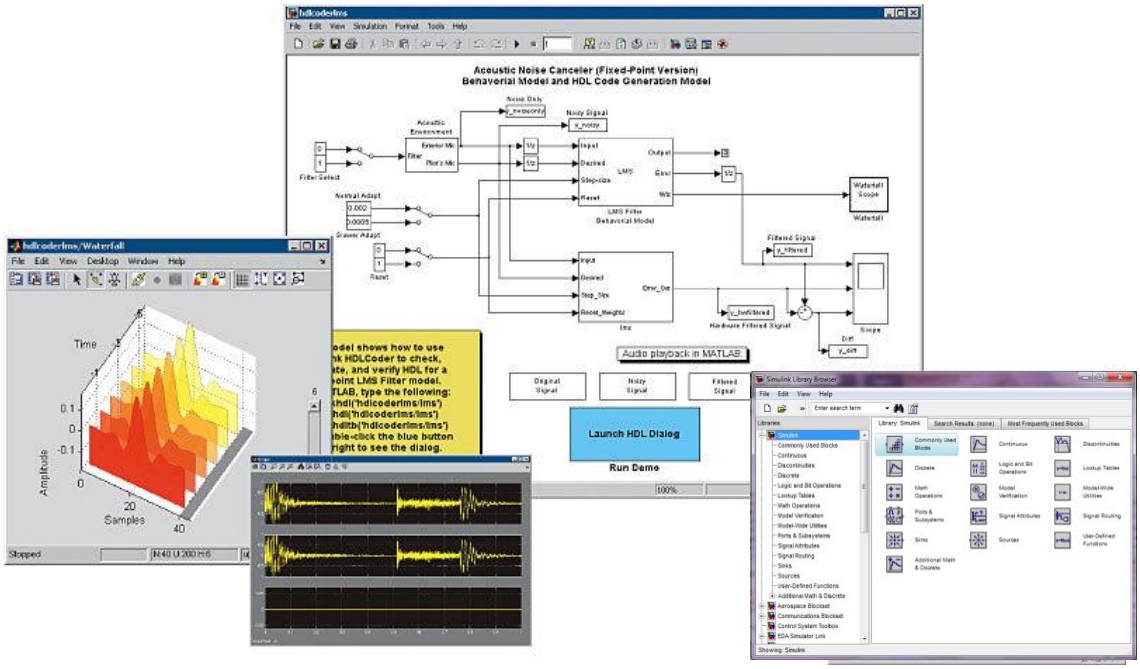
Korene MATLABu siahajú do 70-tych rokoch 20. storočia, kedy boli v jazyku Fortran naprogramované knižnice LINPACK (zastrešovala oblasť lineárnych rovníc) a EISPACK (vyvinutá pre riešenie vlastných čísel). V osemdesiatych rokoch sa prof. Cleve Moler rozhodol pre svojich študentov z univerzity v Novom Mexiku vytvoriť program, ktorý by zastrešoval subrutiny LINPACK a EISPACK s interaktívnym rozhraním bez nutnosti ovládať Fortran. Tento program nazval MATLAB (MATrix LABoratory), neskôr ho s Stevem Bangertom a Johnom Littleom preprogramovali do jazyka C a doplnili o grafiku a založili spoločnosť The MathWorks, ktorá MATLAB ďalej vyvíja a inovuje.

- **Typické úlohy riešené simulačným jazykom MATLAB**

⇨ matematické výpočty, maticové operácie, lineárna algebra, numerické výpočty
⇨ tvorba vlastných funkcií a skriptov
⇨ riešenie lineárnych a nelineárnych diferenciálnych rovníc
⇨ riešenie úloh z analýzy a syntézy lineárnych a nelineárnych DS
⇨ simulácia modelov fyzikálnych systémov

- **Programové prostredie Simulink**

Simulink je grafická nadstavba, zameraná na modelovanie a simuláciu dynamických systémov, ktorý využíva algoritmy jazyka Matlab, ako napríklad algoritmy pre numerické riešenie nelineárnych diferenciálnych rovníc.



Obrázok 1-2 Typy okien v programovom prostredí MATLAB/Simulink

⇨ nezávislé používateľské rozhranie
⇨ pomocou aplikácie Simulink a jeho grafického editora je možné vytvárať modely lineárnych, nelineárnych, v čase diskrétnych alebo spojitých systémov jednoduchým posúvaním funkčných blokov pomocou myši
⇨ obsahuje bloky, ktoré reprezentujú jednotlivé elementárne prvky systémov. Zo skupiny blokov je možné jednoducho vytvárať subsystémy
⇨ využitie napríklad v aplikovanej matematike, automatickom riadení a regulácii, modelovaní fyzikálnych systémov, spracovanie obrazu, zjednodušenie zložitých blokových schém a iné...

- **Aplikačné knižnice (Toolboxy)**

⇨ toolboxy sú tematický zamerané knižnice príkazov určitej vednej oblasti
⇨ časť je distribuovaná voľne prostredníctvom internetu, iné sú vyvíjané spoločnosťou The MathWorks sú predávané prostredníctvom regionálnych predajcov.
⇨ sú vzájomne previazané a používanie jedného neobmedzuje použitie iného.
⇨ všetky toolboxy pritom používajú základné výpočtové jadro simulačného jazyku MATLAB.

- **Control System Toolbox**

Control System Toolbox je aplikačná knižnica, ktorá rozširuje programové prostredie MATLAB o nástroje pre teóriu systémov a kybernetiku (riadiacu techniku).

⇨ obsahuje funkcie z oblasti analýzy a návrhu riadiacich systémov
⇨ okrem klasických prechodových charakteristík využíva aj popis systémov v stavovom priestore
⇨ obsahuje časovo invariantné objekty (LTI), čo sú štruktúry popisujúce jednorozmerné i viacrozmerné lineárne systémy
⇨ poskytuje nástroj na analýzu odozvy systému, ako napríklad prechodovú a frekvenčnú charakteristiku v logaritmických súradniciach a komplexnej rovine.

- **Symbolic Math Toolbox**

Symbolický Toolbox ako už aj jeho názov napovedá poskytuje nástroje pre riešenie a prácu so symbolickými výrazmi.

⇨ do simulačného jazyka Matlab prináša stovky symbolických funkcií, ako napríklad derivovanie, integrovanie, transformácia a riešenie algebraických a diferencialnych rovníc, zjednodušovanie výrazov ...
⇨ oblasť jeho využitia je v aplikovanej matematike a v finančnom modelovaní a analýze

- **Nápoveda v programovom prostredí MATLAB**

Súčasťou programového prostredia MATLAB je nápoveda, ktorá obsahuje kompletnú dokumentáciu k programovému systému aj s príkladmi pre konkrétne funkcie.

- **Nápoveda v okne Command Window**

Ľahko ju vyvoláme príkazom ***help*** za ktorý zapíšeme príkaz alebo okruh príkazov. O takejto nápovede sa ľahko dozvieme ak zadáme do príkazového okna ***help help***

⇨ Ak potrebujeme zistiť použitie a zápis nejakej funkcie (napríklad eye) použijeme príkaz ***help eye***

```
>> help eye
 EYE Identity matrix.
   EYE(N) is the N-by-N identity matrix.

   EYE(M,N) or EYE([M,N]) is an M-by-N matrix with 1's on
   the diagonal and zeros elsewhere.

   EYE(SIZE(A)) is the same size as A.

   EYE with no arguments is the scalar 1.

   EYE(M,N,CLASSNAME) or EYE([M,N],CLASSNAME) is an M
   of class CLASSNAME on the diagonal and zeros elsewhere
```

⇨ Ak potrebujeme zistiť príkazy z niektorého okruhu (napr. elementárne matematické funkcie) do príkazového okna zadáme ***help elfun***

```
>> help elfun
 Elementary math functions.

Trigonometric.
  sin       - Sine.
  sind      - Sine of argument in degrees.
  sinh      - Hyperbolic sine.
  asin      - Inverse sine.
  asind     - Inverse sine, result in degrees.
  asinh     - Inverse hyperbolic sine.
  cos       - Cosine.
  cosd      - Cosine of argument in degrees.
  cosh      - Hyperbolic cosine.
```

⇨ Ak potrebujeme nájsť niektorú funkciu ale nevieme jej presný názov, môžeme použiť funkciu ***lookfor***, ktorá prehľadáva celú dokumentáciu a hľadá v nej konkrétne slovo alebo reťazec slov.

- **MATLAB Help**

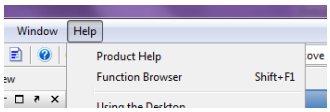
MATLAB Help je ďalšou z možností nápovedy v programovom prostredí MATLAB. Vyvoláme ju

⇨ stlačením tlačidla F1

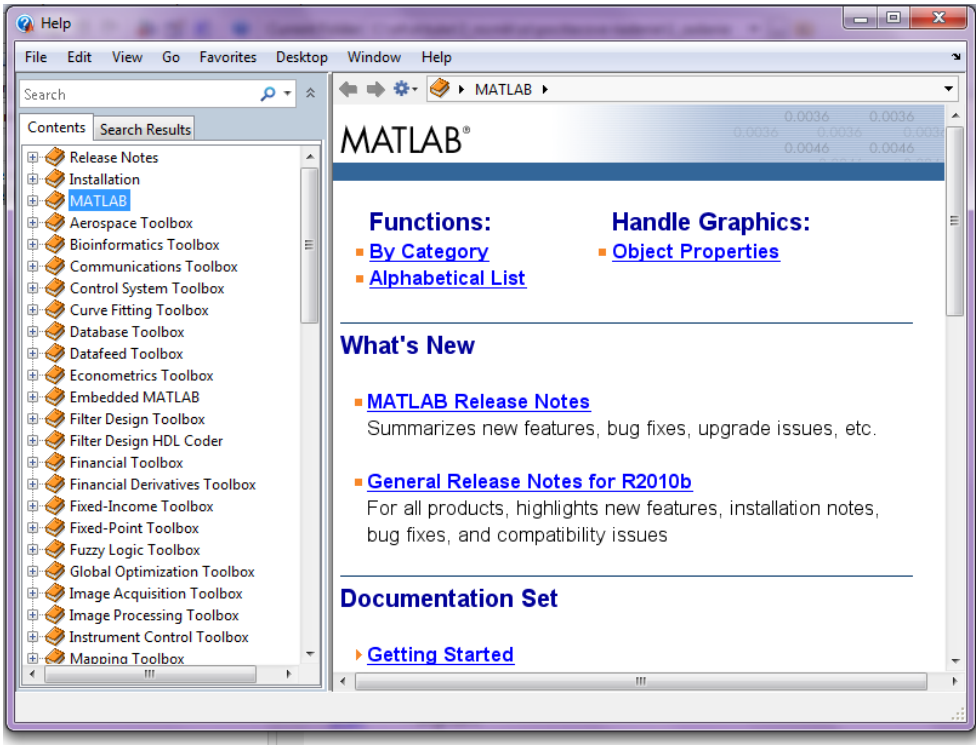
⇨ klepnutím na ikonu 

⇨ výberom možnosti z panelu nástrojov **Help -> Product Help**

⇨ alebo zadaním príkazu ***helpbrowser*** v príkazovom okne



Tieto možnosti otvoria nasledujúce okno:



Obrázok 1-3 Okno pre nápovedu v simulačnom jazyku MATLAB

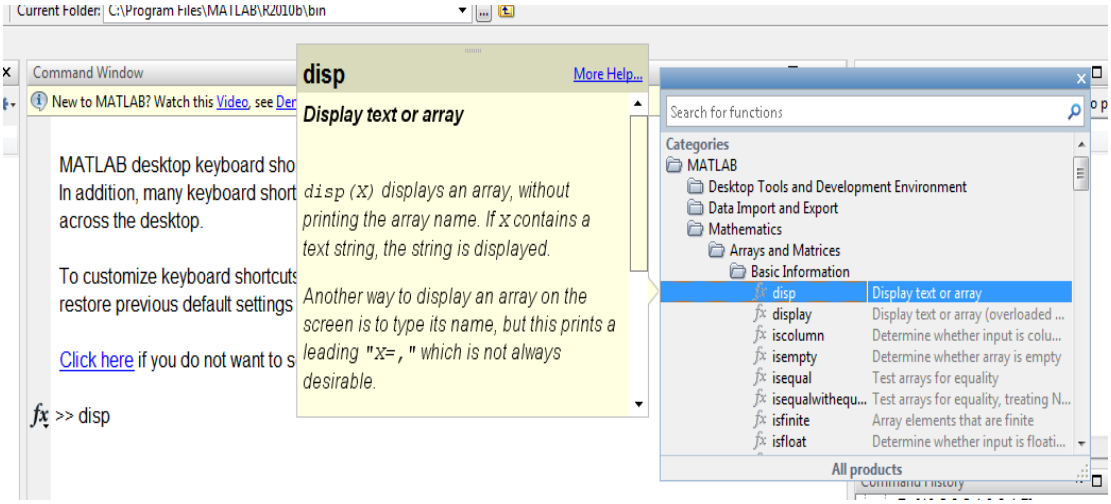
V tomto ***help-e*** sú všetky témy spracované ako www stránky. Súčasťou sú aj ukážkové programy (Demos), ktoré názorne ukazujú použitie jednotlivých funkcií na príkladoch.

- ***Okno nápovedy*** sa skladá z dvoch panelov. Ľavý panel umožňuje vyhľadávať v štruktúre nápovedy, v pravom paneli sa následne objaví aktuálna téma.
- ***Contents*** obsahuje knižnice k jednotlivým Toolboxom, ktorými je možné postupným prechádzaním po zložkách dostať sa až k jednotlivým funkciám.
- V záložke ***Search Results*** je možné pomocou kľúčového slova alebo funkcie možné vyhľadávanie.

- **Prehliadač funkcií**

Ďalšou alternatívou je prehliadač funkcií spúšťaný kombináciou kláves **Shift + F1** alebo výberom z panelu nástrojov z položky ***Help -> Function Browser***.

Prehliadač funkcií umožňuje prezerať jednotlivé záložky – toolboxy a ich podsekcie - v ktorých sú podľa účelu porozdeľované funkcie.
Pri podržaní kurzora nad danou funkciou je otvorená informačná bublina, ktorá nás informuje o použití danej funkcie.
Ak na danú funkciu dvakrát klikneme, dostane sa do *Command Window*, kde s ňou môžeme následne pracovať.



Obrázok 1-4 Okno pre prehliadač funkcií

- **The MathWorks**

The MathWorks je oficiálna stránka programového prostredia MATLAB, kde je taktiež možné nájsť mnoho zaujímavých noviniek o produkte MATLAB, funkcií a tutoriálov k nim, nové toolboxy a mnoho ďalších. http://www.mathworks.com

1.1 Porovnanie simulačného jazyka MATLAB a procedurálneho jazyka C

S programovacím jazykom C sa študenti oboznámili v 1. ročníku na odbore Kybernetika v predmete Programovanie. Na úvod tejto časti si porovnáme programovací jazyk C a programové prostredie Matlab, s ktorým budeme pracovať na predmetoch študijného programu Kybernetika ako napríklad Simulačné systémy , Základy automatického riadenia, Riadenie a vizualizačné systémy, Počitačové riadenie.

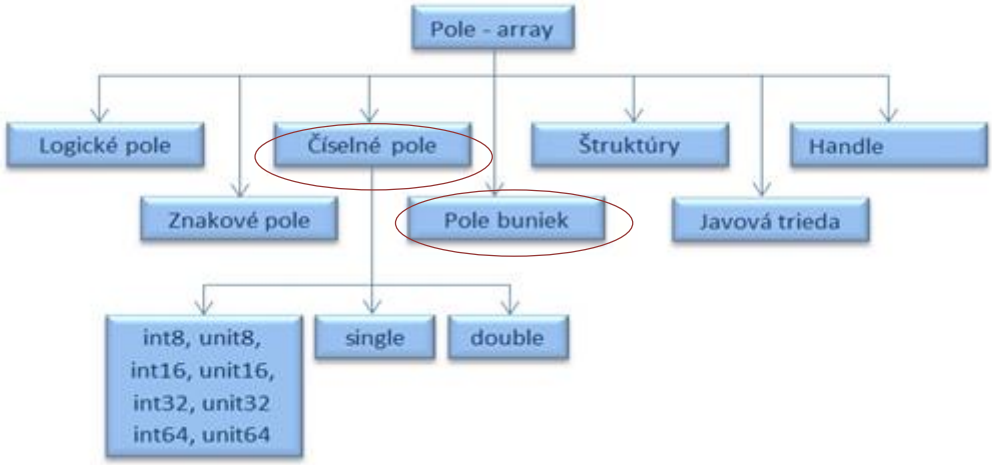
MATLAB (MATrix LABoratory) je vysokoúrovňový simulačný jazyk a interaktívne prostredie pre numerické výpočty, vizualizáciu a programovanie. Pomocou simulačného jazyka Matlab, môžete analyzovať dáta, vyvinúť algoritmy, a vytvárať modely aplikácií. Už samotný názov nám prezrádza, že bol navrhnutý predovšetkým pre prácu s maticami (matrix) a vektormi. Simulačný jazyk ďalej umožňuje prácu s rôznymi grafickými nástrojmi.
Otvorená architektúra prostredia MATLAB viedla k vzniku knižníc funkcií nazvaných toolboxy. Toolboxy predstavujú jednou z výhod jazyka MATLAB, rozširujú použitie programu v príslušných vedných disciplínach.

Jazyk C je problémovo orientovaný programovací jazyk, ktorý bol pôvodne vyvinutý pre operačný systém UNIX a až neskôr bol implementovaný aj do iných operačných systémov a stal sa jedným z najpoužívanejších programovacích jazykov. Jeho hlavnou výhodou je jednoduchosť a nezávislosť na počítači. Umožňuje vytvárať rozsiahle a výkonné programy.

	Matlab	**C**
Druh jazyka	Skriptovací / simulačný	Problémovo orientovaný
Deklarácia dát	Nie je potrebná	Potrebná
Reprezentácia dát	Matica, vektor	Podľa deklarácie
Indexovanie od	1	0

1.2 Dátové typy jazyka MATLAB

Dátový typ je určenie množiny hodnôt, ktoré môže daná premenná nadobúdať. Matlab obsahuje 15 základných dátových typov a každý z nich môže byť zapísaný vo forme matice alebo poľa. Všetky základné typy sú znázornené v nasledujúcom obrázku.



Obrázok 1-5 Dátové typy simulačného jazyka MATLAB

- **Číselné premenné**

Číselné premenné v MATLABe môžu byť zapísané ako znamienkové (signed), alebo neznamienkové (unsigned), ako celé čísla (integer) alebo ako reálne čísla a to buď s jednoduchou (single) alebo dvojitou (double) presnosťou.

Pri celých číslach MATLAB podporuje 8, 16, 32, 64 - bitový spôsob znamienkového/ neznamienkového zobrazenia. Rozsahy jednotlivých typov celých čísel si môžete pozrieť v nasledujúcej tabuľke.

Označenie	**popis**	**rozsah**
int8	znamienkový 8bitový integer	-2^7 – 2^7 -1
int16	znamienkový 16bitový integer	-2^{15} – 2^{15} -1
int32	znamienkový 32bitový integer	-2^{31} – 2^{31} -1
int64	znamienkový 64bitový integer	-2^{63} – 2^{63} -1
uint8	neznamienkový 8bitový integer	0 – 2^8 -1
uint16	neznamienkový 16bitový integer	0 – 2^{16}-1
uint32	neznamienkový 32bitový integer	0 – 2^{32}-1
uint64	neznamienkový 64bitový integer	0 – 2^{64}-1

Pri nezadaní konkrétneho typu číselnej premennej MATLAB ukladá všetky čísla ako reálne čísla s dvojitou presnosťou. Pokiaľ chceme reálne číslo uložiť len s jednoduchou presnosťou musíme to

spraviť pri jeho inicializácií. Na výpise kódu z programovacieho prostredia MATLAB je znázornený rôzny číselný zápis.

Pri použití funkcie „***whos***“ môžeme získať informácie o type premennej a jej veľkosti v bytoch.

```
>> a=uint8(136);
>> b=int32(2165);
>> c=5.214;
>> d=single(6.1247);
>> whos a b c d
  Name        Size              Bytes  Class     Attributes

  a           1x1                   1  uint8
  b           1x1                   4  int32
  c           1x1                   8  double
  d           1x1                   4  single
```

- **Formátovaný vstup a výstup**

Matlab ponúka dve možnosti pre vstup a to :

funkcia ***sscanf*** - čítanie dát z reťazca, po preformátovaní ho uloží do premennej
funckia ***fcanf*** - číta a formátuje dáta z textového súboru.

Pre formátovaný výstup sa používajú funkcie :

funkcia ***sprintf*** - vzťahuje sa na všetky prvky poľa, naformátuje ich a vráti výsledok
funkcia ***fprintf*** - zapisuje dáta do textového súboru alebo ich vypíše na obrazovku.

Pri používaní funkcií formátového vstupu a výstupu je potrebné použiť konverzné znaky. V nasledujúcej tabuľke sú uvedené najpoužívanejšie.

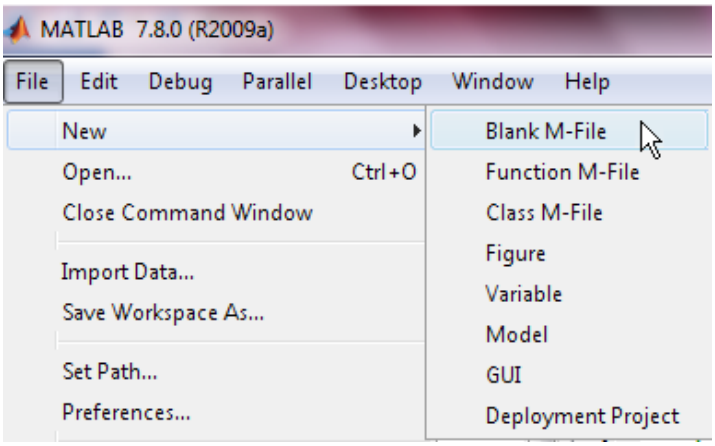
Celé číslo	signed	%d	Desiatkové číslo typu signed
		%ld	Desiatkové číslo typu signed
	unsigned	%u	Desiatkové číslo typu unsigned
		%lu	Desiatkové číslo typu unsigned long
		%o	Osmičkové číslo
		%x	Hexadecimetrálne číslo s malými písmenami
		%X	Hexadecimetrálne číslo s veľkými písmenami
Reálne číslo		%f	
Znaky		%c	Jeden znak
		%s	Reťazec

Príklad:

```
premenna=sscanf(x,'%o');
fprintf('Výsledok je: %f',premenna);
```

1.3 Riadiace štruktúry jazyka MATLAB

Pri tvorbe zložitejších programov, v ktorých sa využívajú riadiace štruktúry, je nevyhnutné vytvárať skripty a kód písať v nich. Nový skript vytvoríme nasledovne:

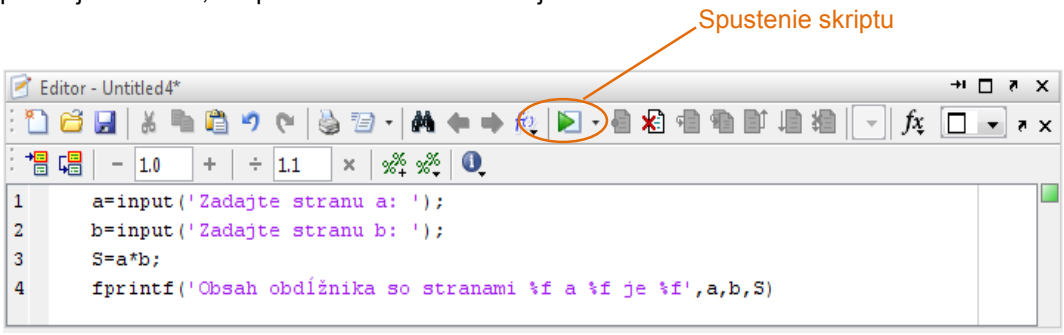


Obrázok 1-6 Vytvorenie nového *m-súboru* pre napísanie SKRIPT-u

- **Skripty** sú postupnosti príkazov uložené do súborov. Sú považované za najjednoduchšie m-fily z toho dôvodu, že nevyžadujú vstupné a výstupné argumenty. Pracujú v spoločnom prostredí programu s globálnymi premennými t.j. pracujú s premennými, ktoré sú definované vo Workspace, ale môžu vytvárať aj vlastné premenné. Hodnoty priradené premenným zostávajú v pamäti aj po vykonaní skriptu.

V čase písania skriptu sa jeho príkazy nevykonávajú. Na ich vykonanie stačí napísať do príkazového okna Command Window MATLABu názov skriptu bez prípony a odoslať na spracovanie.
Niekedy je vhodné nevykonávať určité časti príkazov v M-súbore. Na túto činnosť slúžia komentáre. Komentár začína znakom percenta a končí na konci aktuálneho riadku. Pre lepšiu identifikáciu sa komentáre zvýrazňujú štandardne zelenou farbou.

Príklad 4
Vytvorte postupnosť príkazov - skript, v ktorý vyzve užívateľa na zadanie strán obdĺžnika a následne vypočíta jeho obsah, skript uložte na disk a definujte meno m-súboru



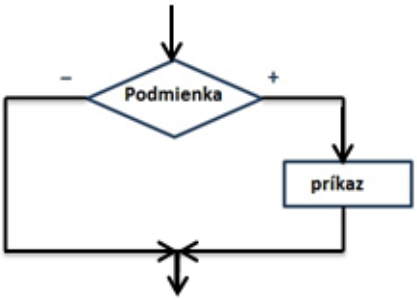
Obrázok 1-7 Vytvorenie skriptu v editore pre výpočet obsahu obdĺžnika

Riadiace štruktúry sú nevyhnutnou súčasťou programovacieho jazyka. Umožňujú riadiť chod programu, jeho vetvenie či opakovanie časti kódu.

- **Jednoduché vetvenie - príkaz if**

Príkaz ***if*** sa používa na jednoduché vetvenie, v prípade kedy je potrebné vykonať dané príkazy len za predpokladu splnenia určitej podmienke. Príkaz vetvenia umožňuje vyhodnotiť podmienku a na základe tohto vyhodnotenia vykonať respektíve nevykonať príkazy v danej vetve. Jeho základná syntax a štruktúra vetvenia znázorňujúca činnosť príkazu sú nasledovné :

```
if podmienka
    prikazy
end
```

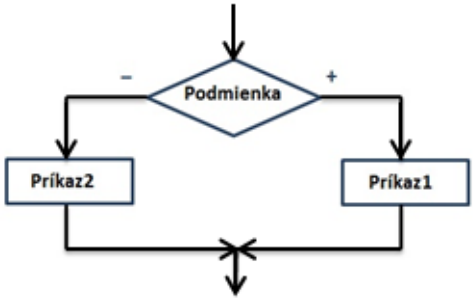


Príklad 5
Ak x < 10, potom vypíšte hlásenie „ číslo x je menšie ako 10“.

```
if x<10
    disp('číslo x je menšie ako 10')
end
```

V prípade potreby vykonania iných príkazov pri nesplnení podmienky sa používa príkaz ***if – else***.
Syntax a štruktúra takto definovaného vetvenia znázorňuje činnosť :

```
if podmienka
    prikaz1
else prikaz2
end
```

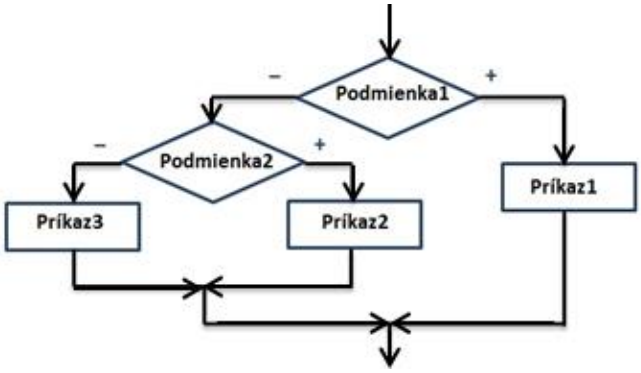


Príklad 6
Ak x < 10 potom vypíšte hlásenie „ číslo x je menšie ako 10“, v opačnom prípade vypíšte hlásenie „číslo x je väčšie alebo rovné číslu 10“.

```
if x<10
    disp('číslo x je menšie ako 10')
else disp('číslo x je väčšie alebo rovné číslu 10')
end
```

Ďalšou modifikáciou príkazu ***if*** je príkaz ***if – elseif – else***.
Syntax a štruktúra vnoreného vetvenia sú nasledovné:

```
if podmienka1
    prikaz1
elseif podmienka2
    prikaz2
else
    prikaz3
end
```



Pri splnení podmienky `podmienka1` sa vykoná príkaz `príkaz1`, a zvyšok príkazu ostane ignorovaný. Pri nesplnení prvej podmienky nasleduje testovanie podmienky `podmienka2`, pri jej kladnom vyhodnotení sa vykoná príkaz `príkaz2`, v opačnom prípade bude vykonaný príkaz `príkaz3`.

Príklad 7
Porovnajte číslo x s číslom 10 a vypíšte príslušné hlásenia.

```
if x<10
    disp('číslo x je menšie ako 10')
elseif x>10
    disp('číslo x je väčšie ako 10')
else disp('číslo x je rovné číslu 10')
end
```

Príklad 8
Zostavte program na určenie minima z 3 čísel zadaných z klávesnice a určte jeho poradie.

Kód v jazyku MATLAB pre Príklad 8

```
a=input('Zadajte číslo: ');
b=input('Zadajte číslo: ');
c=input('Zadajte číslo: ');
min=a;
pozm=1;
if min>b
    min=b;
    pozm=2;
end
if min>c
    min=c;
    pozm=3;
end
fprintf('Hodnota minima zo zadaných čísel je %f \n',min);
fprintf('Pozícia minima zo zadaných čísel je %d \n',pozm);
```

Príklad 9
Vytvorte program na výpočet minimálneho potrebného počtu jázd výťahom. Výťah má obmedzenú nosnosť a sú známe hmotnosti troch cestujúcich, tieto hodnoty zadá užívateľ z klávesnice.

Kód v jazyku MATLAB pre Príklad 9

```
h=input('Zadajte nosnosť výťahu: ');
m1=input('Zadajte hmotnosť 1.pasažiera: ');
m2=input('Zadajte hmotnosť 2.pasažiera: ');
m3=input('Zadajte hmotnosť 3.pasažiera: ');
n=3; %maximálny potrebný počet jázd
if m1+m2<=h | m1+m3<=h | m2+m3<=h %overenie-nájde sa dvojica, ktorá by mohla ísť spolu
   n=2; % nutné sú dve jazdy
end
if m1+m2+m3<=h  % overenie, či by mohli ísť všetci spolu
   n=1; % stačí jedna jazda
end
if m1>h | m2>h | m3>h   % overenie, či určitá hmotnosť nepresahuje nosnosť výťahu
   n=inf;   % nie je možné aby išli všetci výťahom
end
fprintf('Potrebný počet jázd výťahom je: %d \n',n)
```

- **Príkaz switch - case**

V prípade potreby väčšieho počtu vetiev pri jednej podmienke je príkaz vetvenia ***if***, nahradený príkazom ***switch***, ktorého syntax je nasledovná :

```
switch podmienka
    case {hodnota1}
        prikazy1
    case {hodnota2}
        prikazy2
    otherwise
        prikazy3
end
```

`Podmienka` predstavuje hodnotu, ktorú ideme porovnávať s obsahom premenných `hodnota1` a `hodnota2`. Po vyhodnotení zhodnosti s jednou z týchto hodnôt sa vykoná príslušný `príkaz`.
V prípade, že sa `vyraz` nebude zhodovať ani s jedným obsahom `hodnoty`, vykoná sa príkaz `príkaz3`.

Príklad 10
Nech číslo *a* je vstupom do programu. Zistite či zvyšok po delení číslom 7 je párny, nepárny, alebo číslo a je k-násobkom čísla 7.

Kód v jazyku MATLAB pre Príklad 10

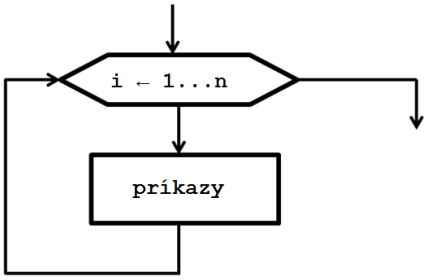
```
a=input('Zadaj celé číslo a : ')
b=mod(a,7)
switch b
    case {1, 3, 5}
        disp('zvyšok po delení číslom 7 je nepárny')
    case {2, 4, 6}
        disp('zvyšok po delení číslom 7 je párny')
    case {0}
        disp('vami zadané číslo je deliteľné číslom 7')
end
```

- **Cykly**

Cykly slúžia na opakované vykonávanie príkazu alebo skupiny príkazov. Cykly môžeme rozdeliť do dvoch skupín a to cykly s pevne daným počtom opakovaní a cykly, pri ktorých nepoznáme počet opakovaní.

- **Cyklus so známym počtom opakovaní - for**

Príkaz ***for*** nám slúži na vykonanie určitých príkazov známy definovaný počet krát. Počet opakovaní určujeme v deklarovaní premennej za príkazom ***for***, v našom prípade ***„i“.*** Určíme počiatočnú hodnotu pre premennú ***i*** od ktorej sa cyklus začne vykonávať, ***krok*** ,s ktorým sa bude cyklus vykonávať. V prípade vynechania časti ***„krok“*** sa za krok automaticky považuje číslo 1. Keď premenna ***i*** dosiahne hodnotu n = počet opakovaní cyklu príkazy v tele cyklu prebehnú posledný krát. Vývojový diagram a syntax príkazu sú na obr.:



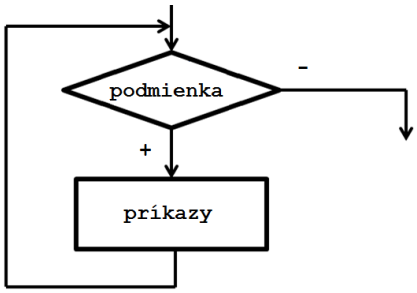

```
for i=zaciatok:krok:koniec
    prikaz
end
```

Príklad 11
Vytvorte vektor ***a*** ktorého prvky budú zodpovedať ich poradiu a dĺžka bude načítaná zo vstupu.

```
n=input('Zadaj dĺžku vektora : ')
for i=1:n
    a(i)=i;
end
disp(a)
```

- **Cyklus while**

Príkaz ***while*** sa využíva väčšinou na vykonanie príkazu, alebo skupiny príkazov s vopred neznámym počtom opakovaní. `Príkaz` sa vykonáva dovtedy, kým je splnená `podmienka`. Syntax príkazu:




```
while podmienka
    prikaz
end
```

Príklad 12
Je zadaná funkcia

$$f(a) = \frac{(a^2-3a-2)}{(a-1)^2(a+2)}$$

Vytvorte program pre výpočet hodnoty definovanej funkcie pre ručne zadaný vstup hodnoty *a*. Použite typ riadiacej štruktúry cyklus, ktorý výpočet vykoná až po zadaní vhodnej hodnoty ***a.***

```
a=input('Zadajte hodnotu a pre ktorú chcete vykonať výpočet výrazu: ');
while (a==1)| (a==-2)
   fprintf('\nZadali ste hodnotu, pre ktorú výraz nemá zmysel.\n')
   a=input('\nZadajte hodnotu a pre ktorú chcete vykonať výpočet výrazu:');
end
x=(a^2-3*a-2)/((a-1)^2*(a+2));
fprintf('Hodnota výrazu:(a^2-3*a-2)/((a-1)^2*(a+2)) pre zadanú hodnotu a=%d je %f',a,x)
```

Príklad 13
Vytvorte maticu s rozmermi 6x8. Použite cyklus, ktorý vytvorí novú maticu, ktorá bude obsahovať prvky pôvodnej matice nachádzajúce sa na pozícií, ktorej poradové číslo stĺpca aj riadku je párne.

Kód v jazyku MATLAB pre Príklad 13

```
M=rand(6,8)
for i=1:6   % cyklus pre prechádzanie riadkov
    for j=1:8   % cyklus pre prechádzanie stĺpcov
        if (rem(i,2)==0 & rem(j,2)==0)
% podmienka: zvyšok po delení čísla i číslom 2 je nula a zároveň zvyšok po
% delení čísla j číslom 2 je nudla
            a=i/2;  % zavedenia nového počítadla riadkov
            b=j/2;  % zavedenie nového počítadla stĺpcov
            A(a,b)=M(i,j);   % vkladanie prvkov z pôvodnej matice do novej
        end
    end
end
A
```

Príklad 14
Predpokladajme, že do „ideálnej banky“, ktorej ročná úroková miera je 5,5% sme vložili 30 000€. Banka úrok pripočítava na konci každého roka zo sumy, ktorá tam je s tou podmienkou, že počas roka sa peniaze nevyberajú. Vypočítajte nárast úroku v nasledujúcich piatich rokoch jednotlivo.

Kód v jazyku MATLAB pre Príklad 14

```
BH(1)=30000*(1+0.055);  %výpočet budúcej hodnoty po prvom roku
for i=2:5
    BH(i)=BH(i-1)*(1+0.055);  %výpočet budúcej hodnoty v ďalších rokoch
end
U(1)=BH(1)-30000; %výpočet úroku, kt. pribudol po prvom roku
for i=1:4
    U(i+1)=BH(i+1)-BH(i);   %výpočet úroku, kt. pribudol v ďalších rokoch
end
U
```

Príklad 15
Vytvorte skript, ktorý vypočíta aritmetický priemer troch najväčších zadaných čísel. Čísla bude zadávať užívateľ z klávesnice a zápis ukončí vložením čísla 0. Ošetrite kód tak, že v prípade, že užívateľ zadal menší počet čísel ako 3 program do aritmetického priemeru vloží „0“.

Kód v jazyku MATLAB pre Príklad 15

```
i=1;
a(1)=input('Zadajte číslo: ');
% kým zadané číslo nie je 0 pokračujte v zadávaní
while a(i)~= 0
    i=i+1;
    a(i)=input('Zadajte číslo: ');
end
% zníženie počítadla o 1  aby hodnota premennej i predstavovaa počet
% zadaných čísel bez čísla 0
i=i-1;
% overenie, či užívateI zadal menší počet čísel ako 3
% v opačno prípade sa vykoná vetva "else"
if i<3
    APmax=0;
else
    % načítanie prvých troch hodnôt do maxím
    for j=1:3
        max(j)=a(j);
    end
    % usporiadaie prvých "maxím" zostupne
    if max(1)<max(2)
        p=max(1);
        max(1)=max(2);
        max(2)=p;
    end
    if max(1)<max(3)
        p=max(1);
        max(1)=max(3);
        max(3)=p;
    end
    if max(2)<max(3)
        p=max(2);
        max(2)=max(3);
        max(3)=p;
    end
    % overenie či zvyšné čísla sú väčšie ako čísla v uložené v premenných
    % max(i)a následné nahradenie hodnoty
    for k=4:i
        if max(3)<a(k)
            max(3)=a(k);
            if max(3)>max(2)
                p=max(2);
                max(2)=max(3);
                max(3)=p;
                if max(2)>max(1)
                    p=max(1);
                    max(1)=max(2);
                    max(2)=p;
                end
            end
        end
    end
    APmax=(max(1)+max(2)+max(3))/3;
end
fprintf('Aritmetický priemer troch najväčších zadaných čísel je: %f',APmax)
```

Príklady na precvičenie

1. Vytvorte program na spravovanie bankového účtu. Užívateľ bude zadávať vklady a výbery prostredníctvom kladných a záporných čísel, ukončenie bude vložením hodnoty 0. Zisti počet výberov, počet príjmov, sumu výberov a sumu príjmov.
2. Naplňte maticu rozmerov 4x4 hodnotami zadanými z klávesnice. Využite pritom cykly.
3. Pomocou cyklu vypíšte sumu všetkých riadkov matice a sumu celej matice spolu.

1.4 Typy matíc a ich reprezentácia v programovom prostredí MATLAB

Matica je určitá množina čísel alebo iných matematických objektov (tzv. prvkov matice) usporiadaných do pravidelných riadkov a stĺpcov.
MATLAB vždy počíta s maticami (t.j. aj skalár je len matica typu 1x1)

- **Typy matíc**

Skalár

- ⇨ matica typu **1** x **1**
- ⇨ pre vytvorenie premennej použijeme priradenie pomocou " =" . Ak nezadáme názov premennej automaticky sa vytvorí premenná ***ans*** (od slovička answer), kde sa uloží hodnota.

```
>> a = 5

a =

     5
```

alebo

```
>> 5

ans =

     5
```

Treba si uvedomiť, že pri každom novom priradení je premenná (vrátane premennej ans) prepisovaná novou hodnotou. Pri novom priradení prídeme o hodnoty uložené v premennej!

Matica m x n

- ⇨ Maticu väčšiu ako 1x1 vždy zapisujeme do hranatých zátvoriek [ ]
- ⇨ Zapisuje sa po riadkoch, pričom jednotlivé elementy (prvky) matice sú oddelené čiarkou alebo medzerou
- ⇨ Nový riadok vytvoríme zadaním bodkočiarky alebo stlačením klávesy ***ENTER*** pričom v tomto prípade ak nie sú zátvorky ukončené, ***ENTER*** neodošle príkaz, iba nás posunie na ďalší riadok

```
>> A=[1,0
2,5
4,3]

A =

     1     0
     2     5
     4     3
```

alebo

```
>> B=[1 0; 2 5; 4 3]

B =

     1     0
     2     5
     4     3
```

alebo

```
>> C=[1 0
2 5
4 3]

C =

     1     0
     2     5
     4     3
```

alebo

```
>> D=[1, 0; 2, 5; 4, 3]

D =

     1     0
     2     5
     4     3
```

Riadkový vektor

- ⇨ matica typu **1** x ***n***
- ⇨ Vytvára sa podobne ako by sme vytvorili maticu ***n x m*** , ale použijeme len medzery alebo čiarky medzi jednotlivými prvkami vektora.
- ⇨ Vytvorenie riadkového vektora je možné aj „***dvojbodkovou konvenciou"*** a to tak, že do hranatých zátvoriek zapíšeme 3 hodnoty :
  počiatočná hodnota,
  hodnota kroku ,
  konečná hodnota.

Túto možnosť je výhodné využívať pri väčších vektoroch, ktoré majú byťinicializované s konštantným krokom.

```
>> E = [1 2 3]

E =

   1   2   3
```

alebo

```
>> E = [1, 2, 3]

E =

   1   2   3
```

alebo

```
>> E=[0:1.2:5]

E =

    0    1.2000    2.4000    3.6000    4.8000
```

⇨ V prípade ak neurčíme hodnotu kroku, bude automaticky rovná 1.

```
>> E=[0:5]

E =

   0   1   2   3   4   5
```

Stĺpcový vektor

⇨ matica typu ***m*** x **1**
⇨ stĺpcový vektor vytvoríme použitím bodkočiarky ";" alebo klávesy ENTER za každou číslicou

```
>> F = [1;2;3]

F =

   1
   2
   3
```

alebo

```
>> F = [1
2
3]

F =

   1
   2
   3
```

Jednotková matica

⇨ štvorcová matica, ktorá obsahuje na hlavnej diagonále samé ***jednotky*** a zvyšné prvky matice tvoria nuly
⇨ takúto maticu vieme vytvoriť dvoma spôsobmi a to tak ako by sme vytvárali maticu ***n x n***, a po jednom vypisovali každý prvok matice, čo je podstatne prácnejšie a náchylnejšie na pomýlenie sa ako druhá možnosť a to je vytvorenie pomocou príkazu :
G = ***eye (****rozmer_matice****)***

```
>> G = eye(3)

G =

   1   0   0
   0   1   0
   0   0   1
```

Nulová matica

⇨ matica, ktorej všetky prvky sú **nulové**

⇨ takúto maticu je tiež možné vytvoriť dvoma spôsobmi a to buď prácnym vypisovaním pre každý prvok nulu, alebo použitím príkazu pre štvorcovú maticu *H = **zeros(**n**)***, alebo príkazu *H =**zeros(**m,n**)*** pre maticu typu ***m x n***

```
>> H = zeros(3)

H =

     0     0     0
     0     0     0
     0     0     0
```

alebo

```
>> H = zeros(3,2)

H =

     0     0
     0     0
     0     0
```

Matica jednotiek

⇨ všetky prvky matice sú ***jednotky***
⇨ pre štvorcovú maticu: *I =**ones(**n**)*** pre maticu typu ***m x n*** *I = **ones(**m,n**)***

```
>> I = ones (3)

I =

     1     1     1
     1     1     1
     1     1     1
```

alebo

```
>> I = ones (3,2)

I =

     1     1
     1     1
     1     1
```

Generovanie vektora s ekvidistantným krokom

⇨ pre vytvorenie vektora môžeme použiť príkaz *J = **linspace(**poč_hodnota, koneč_hodnota**)*** pričom tento príkaz vytvorí vektor o 100 hodnotách v intervale <poč_hodnota, koneč_hodnota>, alebo ak chceme vytvoriť ***n-*** hodnotový vektor za konečnú hodnotu napíšeme počet prvkov

*J =**linspace(**poč_hodnota, koneč_hodnota, počet_prvkov**)***

```
>> J = linspace(1,15)

J =

  Columns 1 through 6

    1.0000    1.1414    1.2828    1.4242    1.5657    1.7071

  Columns 7 through 12

    1.8485    1.9899    2.1313    2.2727    2.4141    2.5556

  Columns 13 through 18

    2.6970    2.8384    2.9798    3.1212    3.2626    3.4040
```

```
>> J = linspace(1,15,10)

J =

  Columns 1 through 6

    1.0000    2.5556    4.1111    5.6667    7.2222    8.7778

  Columns 7 through 10

   10.3333   11.8889   13.4444   15.0000
```

Matica náhodných čísel

⇨ V programovom prostredí MATLAB existuje príkaz ***rand(**veľkosť_matice**)***. Tento príkaz vygeneruje maticu náhodných čísel z intervalu <0,1>

```
>> K = rand(2,3)

K =

    0.9572    0.8003    0.4218
    0.4854    0.1419    0.9157
```

⇨ Druhou alternatívou je použiť príkaz ***randn(****veľkosť_matice****)***, ktorý použije čísla z normálneho (Gaussovského) rozdelenia

```
>> K = randn(3,2)

K =

   -0.4336    2.7694
    0.3426   -1.3499
    3.5784    3.0349
```

1.5 Základné operácie s údajovým typom matica

Dáta v prostredí MATLAB, ako sme už spomínali, sú reprezentované maticami. Matica je dvojrozmerné / obdĺžnikové pole reálnych alebo komplexných čísel. Matica, ktorá má *m* riadkov a *n* stĺpcov je nazývaná „matica *m* x *n*“. Každý prvok matici je indexovaný dvojitým indexom. napr a_{ij}, kde *i* predstavuje poradie riadku a *j* poradie stĺpca, v ktorom sa prvok nachádza. Môžeme použiť aj skalárnu veličinu, avšak v Matlabe je tiež chápaná ako matica 1x1.

Maticu môžeme zadať niekoľkými spôsobmi, a to :
- ⇨ zadaním po prvkoch (prvky v riadku sú oddelené medzerou/čiarkou, v stĺpcoch + bodkočiarkou)
- ⇨ pomocou funkcie – špeciálne matice (***eye, ones, rand***...)
- ⇨ nahraním zo súboru

- **Operátor „ : “**

Často používaný operátor, dvojbodkovej konvencie sa používa pri tvorbe postupností s konštantným krokom.

Jeho syntax je :

```
v = začiatok : krok : koniec
```

Kde
- „krok“ je možné vynechať a vtedy sa bude chápať krok = 1
- „krok“ môže byť aj záporné číslo, vtedy sa vytvára zostupná postupnosť

Príklad 1
V príkazovom režime vytvorte vektor v1 s prvkami od 1 po 9 s krokom 2 a vektor v2 ktorého prvky budú zostupnou postupnosťou čísel od 15 po 3 s krokom 3 pomocou dvojbodkovej konvencie.

```
>> v1=1:2:9

v1 =

     1     3     5     7     9
```

```
>> v2=15:-3:3

v2 =

    15    12     9     6     3
```

- **Funkcia *linspace* na generovanie vektorov**

Funkcia ***linspace*** má podobnú funkciu ako predchádzajúci operátor „ : “ s tým rozdielom, že si presne vypočíta krok. Je výhodné použiť túto funkciu v prípadoch, kedy potrebujeme vytvoriť vektor s vopred známym počtom prvkov, ale neznámym krokom. Syntax funkcie je:

```
v = linspace(začiatok, koniec, počet_prvkov)
```

Príklad 2
V príkazovom režime vytvorte vektor v3, s počtom prvkov 5, ktoré budú z intervalu <0;π>. Príklad riešte s využitím funkcie ***linspace***.

```
>> v3=linspace(0,pi,5)

v3 =

         0    0.7854    1.5708    2.3562    3.1416
```

- **Výber prvkov matice a submatica**

Ako sme už spomenuli, každý prvok matice je indexovaný v tvare a_{ij}. Pri výbere konkrétneho prvku matice stačí zadať príkaz :

```
Matica(číslo_riadku, číslo_stĺpca)
```

Pri výbere časti matice – submatice nám pomáha operácia dvojbodkovej konvencie „ **:** “. Je to podobne ako pri výbere prvku s tým rozdielom, že namiesto „číslo_riadku“ sa dá rozmedzie riadkov, ktoré chceme vybrať. A rovnako aj namiesto „číslo_stĺpca“ dáme rozmedzie stĺpcov, ktoré má submatica obsahovať.
Príkaz:

```
A=M(x:y,x:y)
```

kde

x – poradie riadka/stĺpca, od ktorého chceme vytvoriť submaticu
y - poradie riadka/stĺpca, pri ktorom chceme ukončiť výber

Príklad 3
Vytvorte maticu M(4,4) pričom jej prvky a_{ij} sú celé čísla

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

a) hodnotu prvku 3.riadku a 2. stĺpca vložte do premennej *a*
b) vytvorte submaticu *B* z matice *M*, ktorá bude tvorená prvkami 1.-2. riadku, 2.-4. stĺpca.

```
>> M=[1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]

M =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16

>> a=M(3,2)

a =

    10

>> B=M(1:2,2:end)

B =

     2     3     4
     6     7     8
```

1.6 Príklady na riešenie

Riadiace štruktúry

1. Vytvorte vektor v, ktorý bude obsahovať 10 náhodných čísel z intervalu <-10 ; 10>.

2. Zistite koľko prvkov z vektora v je kladných, koľko záporných a koľko rovných nule.

3. Vytvorte vektory k a z. Do vektora k vložte všetky prvky z vektora v, ktoré sú kladné. Do vektora z vložte všetky prvky z vektora v, ktoré sú záporné. Vypíšte na obrazovku vektory v, k, z.

4. Vytvorte pomocou cyklu maticu rozmerov 4x4, ktorá bude obsahovať len párne čísla. Čísla do matice sa budú vkladať z klávesnice. Zadajte podmienku, aby sa do matice dali vložiť len párne čísla.

5. Vytvorte cyklus, ktorý vypíše minimálne hodnoty stĺpcov z ľubovoľnej matice, ktorú zadáte. Následne porovnajte výsledky, ktoré vyhodnotil vás cyklus s výsledkami, ktoré vygeneroval príkaz min().

6. Vytvorte maticu ľubovoľných hodnôt s rozmermi 5x5. Napíšte cyklus na vytvorenie vektora, ktorý bude obsahovať prvky hlavnej diagonálu zadanej matice. Vypočítajte sumu tohto vektora, výsledok vypíšte.

7. Predpokladajme, že do banky, ktorá má ročnú úrokovú mieru 7% vložíme 10 000€.
    a. zistime za koľko rokov budeme mať aspoň 20 000€
    b. ako sa nám bude meniť suma v jednotlivých rokoch

Matice a vektory

1. Do premennej *x* vložte 10 čísel v rozmedzí čísel od 1 do 5 pomocou funkcie linspace.
2. Do premennej *y* vložte logaritmy hodnôt nachádzajúcich sa vo vektore *x*.
3. Do premennej *z* vložte hodnoty premennej *x* umocnené na druhú.
4. Vytvorte vektor *s,* ktorého počet prvkov bude 8, prvky budú usporiadané zostupne a ich hodnoty budú v rozmedzí od 23 po 9 pomocou dvojbodkovej konvencie.
5. Vytvorte maticu M, ktorá bude mať v prvom riadku čísla od 3 po7, v druhom riadku bude tak isto päť prvkov ktoré budú v rozmedzí 10 – 23 a v treťom riadku nech sú čísla v zostupnom poradí od 19 do 5 s rovnomerným rozdelením.
    a. Vytvorte submaticu A matice *M,* ktorá bude obsahovať prvky 2-3riadku a 3-4stĺpca
    b. Vytvorte submaticu B matice *M*, ktorá bude obsahovať prvky 3-koncového riadku a 1-2 stĺpca, tak aby prvky boli v riadku usporiadané v opačnom poradí oproti matici *M*
6. Vytvorte ľubovoľnú maticu K s rozmermi 8x8. Vytvorte ľubovoľnú submaticu *C* matice *M*. Z matice *C* vyberte prvok nachádzajúci sa na pozícií [1,3] a vložte ho na pozíciu [1,4]. Zobrazte maticu *C* pred aj po presune prvku [1,3]
7. Vytvorte premennú *t ,* ktorá bude obsahovať 20 hodnôt funkcie *cos(x)* pričom x bude v rozmedzí od <-10:10>. Hodnoty ktoré získate vypíšte na obrazovku .

1.7 Operácie s maticami

- **Základné operátory**

Znak	Opis	Znak	Opis
+	Plus	**.**	Desatinná bodka
-	Mínus	**%**	Komentár
*	Maticové násobenie	**=**	Priradenie
.*	Násobenie po prvkoch	**==**	Zhodnosť
^	Umocnenie	**&**	Logický AND
. ^	Umocnenie po prvkoch	**|**	Logický OR
\	Ľavé delenie	**~**	Logická negácia
/	Pravé delenie		
./	Pravé delenie prvkov		

Sčítanie a odčítanie matíc

Uvažujme matice ***A*** *(m, n)* a ***B*** *(m, n)*. Ich súčet je definovaný nasledovne:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{11} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{pmatrix}$$

Riešenie v programovom prostredí MATLAB:

```
>> A = [1,2,3;4,5,6;7,8,9];
>> B=[7,8,9;4,5,6;1,2,3];
>> C=A+B

C =

    8   10   12
    8   10   12
    8   10   12
```

Pozn. Odčítanie dvoch matíc je definované ako pripočítanie opačnej matice, t.j. platia rovnaké podmienky pre rozmery matíc ako pri sčítaní.
Operácie sčítania a odčítania sú definované len pre matice rovnakých rozmerov, inak MATLAB vypíše chybu kvôli nezhode rozmerov matíc.

```
>> A=[1 0;2 5;4 3];
>> B=[2 3;5 8];
>> A-B
??? Error using ==> minus
Matrix dimensions must agree.

>> A+B
??? Error using ==> plus
Matrix dimensions must agree.
```

⇨ výnimkou je sčítanie a odčítanie skaláru od matice

```
>> A=[2 3;5 8];
>> B= A+2

B =

   4   5
   7  10

>> C=A-2

C =

   0   1
   3   6
```

Násobenie matíc

⇨ dve matice rozmerov ***m x n*** a ***p x r*** môžeme násobiť iba za predpokladu, že ***n = p***; pričom ako výsledok dostaneme maticu veľkosti ***m x r***

⇨ V prípade že neplatí podmienka ***n = p***, vypíše MATLAB chybu. Výnimku opäť tvorí skalár, ktorým vieme násobiť matice.

```
>> A=[1 0;2 5;4 3];
>> B=[2 3;5 8];
>> C=B*A
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Príklad 1
Vypočítajte súčin daných matíc A(m,n) , B(p,r) a overte v programovom prostredí MATLAB

$$A=\begin{pmatrix}1 & 0\\ 2 & 5\\ 4 & 3\end{pmatrix},\quad B=\begin{pmatrix}2 & 3\\ 5 & 8\end{pmatrix}.$$

Ručný výpočet súčinu matíc (podmienky pre násobenie matíc sú evidentne splnené):

$$\begin{pmatrix}1 & 0\\ 2 & 5\\ 4 & 3\end{pmatrix}*\begin{pmatrix}2 & 3\\ 5 & 8\end{pmatrix}=\begin{pmatrix}2+0 & 3+0\\ 4+25 & 6+40\\ 8+15 & 12+24\end{pmatrix}=\begin{pmatrix}2 & 3\\ 29 & 46\\ 23 & 36\end{pmatrix}$$

Súčin matíc v programovom prostredí MATLAB:

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

A =

   1   0
   2   5
   4   3

>> B=[2 3;5 8]

B =

   2   3
   5   8
```

...

```
>> C=A*B

C =

   2   3
  29  46
  23  36
```

Transponovaná matica

- ⇨ matica, ktorá vznikne výmenou jednotlivých riadkov za stĺpce a naopak, t.j. ak $\boldsymbol{B} = \boldsymbol{A}^T$, tak platí $b_{ij} = a_{ji}$

Príklad 2
Vytvorte transponovanú maticu k matici $\boldsymbol{A}(m,n)$, ktorá je zadaná v tvare :

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 \\ 2 & 5 \\ 4 & 3 \end{pmatrix}$$

a následne overte svoje riešenie v programovom prostredí MATLAB.
Transponovaná matica A:

$$\boldsymbol{A}^T = \begin{pmatrix} 1 & 2 & 4 \\ 0 & 5 & 3 \end{pmatrix}$$

Riešenie v programovom prostredí MATLAB:

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

A =

   1   0
   2   5
   4   3

>> D=A'

D =

   1   2   4
   0   5   3
```

1.8 Operácie po prvkoch matice (element-by-element)

Pri použití operácií element-by-element sa príslušný operátor aplikuje postupne na každú dvojicu prvkov uvažovaných matíc nezávisle od seba. Pre zápis takýchto operácií používame bodku ". " pred operátorom. Matice vystupujúce ako operandy musia mať rovnaké rozmery, inak MATLAB vypíše chybu.

```
>> A=[1 2 3; 4 5 6];
>> B=[9 8 7 ; 6 5 4 ;3 2 1];
>> C=A.*B
??? Error using ==> times
Matrix dimensions must agree.
```

Násobenie po prvkoch – vynásobenie každého prvku matice *A* prvkom matice *B* na príslušnej pozícii v riadku a stĺpci

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} .* \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11} * b_{11} & a_{12} * b_{12} & \cdots & a_{1n} * b_{1n} \\ a_{21} * b_{21} & a_{22} * b_{22} & \cdots & a_{2n} * b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} * b_{m1} & a_{m2} * b_{m2} & \cdots & a_{mn} * b_{mn} \end{pmatrix}$$

Výsledok operácie násobenia po prvkoch v programovom prostredí MATLAB**:**

```
>> A=[1 2 3 ; 4 5 6; 7 8 9];
>> B=[9 8 7 ; 6 5 4 ;3 2 1];
>> C=A.*B

C =

    9   16   21
   24   25   24
   21   16    9
```

Z definície násobenia skalárom vyplýva, že rovnaký výsledok dostávame pri štandardnom násobení a násobení po prvkoch:

```
>> A=[1 2 3 ; 4 5 6; 7 8 9];
>> C= A.*5

C =

    5   10   15
   20   25   30
   35   40   45
```

```
>> C = A*5

C =

    5   10   15
   20   25   30
   35   40   45
```

Delenie po prvkoch – vydelenie každého prvku matice *A* prvkom matice *B* na príslušnej pozícii v riadku a stĺpci

Delenie tak ako násobenie po prvkoch musí byť vykonávané s maticami rovnakých rozmerov, inak programový system MATLAB zahlási chybu.

```
>> A=[5,10,15;20,25,30;35,40,45];
>> B=[1,2,3;4,5,6];
>> C=A./B
??? Error using ==> rdivide
Matrix dimensions must agree.
```

```
>> A=[5,10,15;20,25,30;35,40,45];
>> B=[1,2,3;4,5,6;7,8,9];
>> C=A./B

C =

     5     5     5
     5     5     5
     5     5     5
```

Umocňovanie po prvkoch – umocnenie každého prvku matice *A* prvkom matice *B* na príslušnej pozícii v riadku a stĺpci

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} .^{\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix}} = \begin{pmatrix} a_{11}{}^{b_{11}} & a_{12}{}^{b_{12}} & \cdots & a_{1n}{}^{b_{1n}} \\ a_{21}{}^{b_{21}} & a_{22}{}^{b_{22}} & \cdots & a_{2n}{}^{b_{2n}} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}{}^{b_{m1}} & a_{m2}{}^{b_{m2}} & \cdots & a_{mn}{}^{b_{mn}} \end{pmatrix}$$

Výsledok umocňovania po prvkoch v programovom prostredí MATLAB:

```
>> A=[2,3,5;4,5,2;1,3,2];
>> B=[2,2,1;2,1,2;3,1,3];
>> C=A.^B

C =

     4     9     5
    16     5     4
     1     3     8
```

1.9 Riešenie systémov algebraických rovnic v jazyku MATLAB

Nech je daný **systém lineárnych algebraických rovníc** *n*-tého rádu vo všeobecnom tvare:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{1}$$

kde $a_{11},a_{12},\ldots,a_{nn},b_1,\ldots,b_n$ sú reálne čísla a usporiadaná *n*-tica $\{x_1,x_2,...,x_n\}$ predstavuje riešenie systému.
Vzhľadom na pravidlá násobenia matíc môžeme systém (1) zapísať v maticovom tvare

$$\boxed{\boldsymbol{Ax} = \boldsymbol{b}} \tag{2}$$

ktorého matice koeficientov majú tvar $\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}$ (matica rozmeru *nxn*) a $\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ (stĺpcový vektor rozmeru *nx1*). Riešením systému je stĺpcový vektor $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ rozmeru *nx1*.

Pokiaľ je matica $\boldsymbol{A}$ **regulárna** (jej determinant je rôzny od 0, resp. matica $\boldsymbol{A}$ má plnú hodnosť), potom k nej **existuje inverzná matica** $\boldsymbol{A}^{-1}$, pre ktorú platí

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I} \tag{3}$$

kde $\boldsymbol{I} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$ je jednotková matica *n*-tého rádu.

Ak teda systém (2) vynásobíme **zľava** maticou $\boldsymbol{A}^{-1}$, naľavo sa osamostatní vektor $\boldsymbol{x}$ (pretože $\boldsymbol{Ix} = \boldsymbol{x}$):

$$\begin{aligned} \boldsymbol{A}^{-1}\boldsymbol{Ax} &= \boldsymbol{A}^{-1}\boldsymbol{b} \\ \boldsymbol{Ix} &= \boldsymbol{A}^{-1}\boldsymbol{b} \end{aligned} \tag{4}$$

a riešenie systému možno vyjadriť v tvare:

$$\boxed{\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}} \tag{5}$$

***Pomôcka**:* Princíp (5) je analogický princípu, podľa ktorého sa riešia lineárne rovnice. Ak je daná lineárna rovnica (t.j. 1 rovnica 1. rádu s 1 riešením), napr.

$$5x = 8\,, \tag{6}$$

tak riešenie spočíva v tom, že vynásobíme ľavú aj pravú stranu prevrátenou hodnotou ku koeficientu, ktorý sa nachádza pri *x*, v tomto prípade $\frac{1}{5} = 5^{-1}$. Keďže platí, že súčin čísla a jeho prevrátenej hodnoty je 1 (analógia s (3)), tak riešenie rovnice dostávame v tvare

$$\begin{gathered} \frac{1}{5}5x = \frac{1}{5}8 \\ \boxed{x = \frac{8}{5}} \end{gathered} \tag{7}$$

Inak povedané, riešenie sme dostali **vydelením pravej strany koeficientom pri *x***. Je to síce triviálne tvrdenie, ale ak si ho porovnáme so vzťahom (5), môžeme povedať, že inverzia je istou analógiou delenia. Tento záver využijeme nižšie pri popise operácií ľavého a pravého maticového delenia.

Príklad 1. *Použitím programového prostredia MATLAB riešte nasledujúci systém lineárnych algebraických rovníc 3. rádu metódou a) násobenia inverznou maticou zľava b) ľavého maticového delenia:*

$$\begin{aligned} 2x_1 + 5x_2 + 3x_3 &= 1 \\ 4x_1 + 6x_2 + 2x_3 &= 5 \\ x_1 - 5x_2 + 3x_3 &= 3 \end{aligned} \tag{8}$$

Riešenie: *a) V MATLABe si vytvoríme matice* ***A***, ***b*** *a využitím príkazu* **`inv`** *na výpočet inverznej matice určíme riešenie systému* ***x*** *zo vzťahu (5):*

```
>> A=[2 5 3; 4 6 2; 1 -5 3];
>> b=[1; 5; 3];
>> x=inv(A)*b

x =

    2.0278
   -0.4028
   -0.3472
```

b) Operácia vynásobenia stĺpcového vektora inverznou maticou zľava je ekvivalentná s ***operáciou ľavého maticového delenia*** *v MATLABe. (Pomôcka: Uvažujte s inverziou ako s analógiou delenia a vzťah (5) si predstavujte ako „****b*** *delené* ***A****“.)*

```
>> x=A\b

x =

    2.0278
   -0.4028
   -0.3472
```

Systém lineárnych algebraických rovníc (1) možno okrem štandardného maticového vyjadrenia (2) zapísať aj v nasledovnom ekvivalentnom maticovom tvare:

$$\boxed{\boldsymbol{x}_E \boldsymbol{A}_E = \boldsymbol{b}_E} \tag{9}$$

pričom matice koeficientov majú tvar $\boldsymbol{A}_E = \boldsymbol{A}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{bmatrix}$ (matica rozmeru *nxn*),

$\boldsymbol{b}_E = \boldsymbol{b}^T = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}$ (riadkový vektor rozmeru *nx1*). Riešením systému je riadkový vektor $\boldsymbol{x}_E = \boldsymbol{x}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ rozmeru *nx1*. (*Odporúčanie:* Roznásobením po prvkoch overte, že zápis (9) je ekvivalentný zápisu (1) aj (2).)

Riešenie $\boldsymbol{x}_E$ systému (9) dostaneme, ak obidve strany rovnice vynásobíme **sprava** maticou $\boldsymbol{A}_E^{-1}$, čím sa podobne ako v predchádzajúcom prípade naľavo osamostatní vektor $\boldsymbol{x}_E$:

$$\begin{aligned} \boldsymbol{x}_E \boldsymbol{A}_E \boldsymbol{A}_E^{-1} &= \boldsymbol{b}_E \boldsymbol{A}_E^{-1} \\ \boldsymbol{x}_E \boldsymbol{I} &= \boldsymbol{b}_E \boldsymbol{A}_E^{-1} \end{aligned} \tag{10}$$

a riešenie systému nadobudne tvar:

$$\boxed{\boldsymbol{x}_E = \boldsymbol{b}_E \boldsymbol{A}_E^{-1} = \boldsymbol{b}^T \left(\boldsymbol{A}^T \right)^{-1}} \tag{11}$$

resp. (pokiaľ chceme riešenie vyjadriť v tvare stĺpcového, nie riadkového vektora):

$$\boxed{\boldsymbol{x} = \boldsymbol{x}_E^T = \left(\boldsymbol{b}^T \left(\boldsymbol{A}^T \right)^{-1} \right)^T} \tag{12}$$

Príklad 2. Použitím programového prostredia MATLAB a maticového zápisu (9) riešte systém lineárnych algebraických rovníc 3. rádu z príkladu 1 metódou
a) násobenia inverznou maticou sprava,
b) pravého maticového delenia.

Riešenie:
a) Podobne ako v príklade 1 si v MATLABe vytvoríme matice ***A***, ***b*** systému a využitím príkazu `inv` na výpočet inverznej matice určíme riešenie systému buď v tvare riadkového vektora $\boldsymbol{x}_E$ zo vzťahu (11), alebo v tvare stĺpcového vektora $\boldsymbol{x}$ zo vzťahu (12):

```
>> A=[2 5 3; 4 6 2; 1 -5 3];
>> b=[1; 5; 3];
>> xe=b'*inv(A')

xe =

    2.0278   -0.4028   -0.3472

>> x=(b'*inv(A'))'

x =

    2.0278
   -0.4028
   -0.3472
```

b) Operácia vynásobenia riadkového vektora inverznou maticou sprava je ekvivalentná s **operáciou pravého maticového delenia** v MATLABe. (Pomôcka: Uvažujte s inverziou ako s analógiou delenia a vzťah (5) si predstavujte ako „$\boldsymbol{b}^T$ delené $\boldsymbol{A}^T$“.)

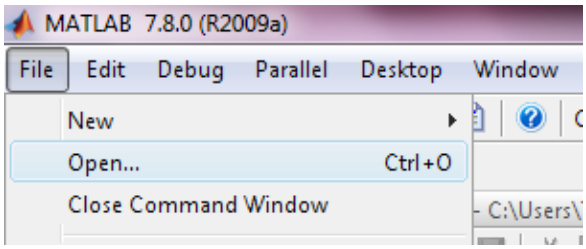
```
>> xe=b'/A'

xe =

    2.0278   -0.4028   -0.3472

>> x=(b'/A')'

x =

    2.0278
   -0.4028
   -0.3472
```

2 Riešenie úloh v simulačnom jazyku MATLAB s využitím skriptov a funkcií

2.1 Práca s m-súborom v programovom prostredí MATLAB

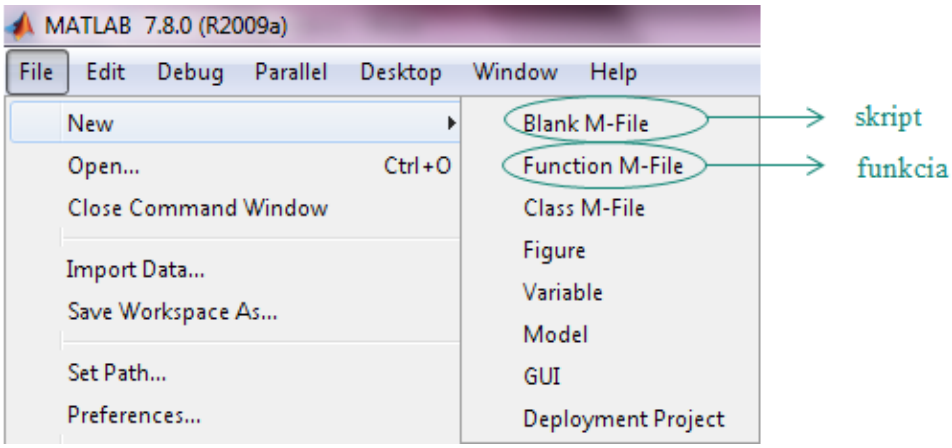
- **M – file / M – súbor** je textový súbor s príponou „ .m “ , ktorý slúži na ukladanie postupnosti príkazov – ***skripty***, alebo na ukladanie užívateľských funkcií – ***funkcie***. Využívajú sa predovšetkým vtedy, keď je potrebné zadať väčšie množstvo príkazov, čo by v príkazovom okne bolo neprehľadné, zdĺhavé a zložité meniť. Programové prostredie MATLAB obsahuje vlastný textový editor M – súborov, avšak môže sa použiť aj iný textový editor. Takto je k nim zabezpečený priamy prístup a môžu sa kedykoľvek upravovať nezávisle od hlavného programu.

Editor prostredia MATLAB sa spustí v novom okne po otvorení ***M- súboru***



Obrázok 2-1 Spustenie Editora po otvorení M - súboru

alebo po vytvorení nového ***M- súboru***

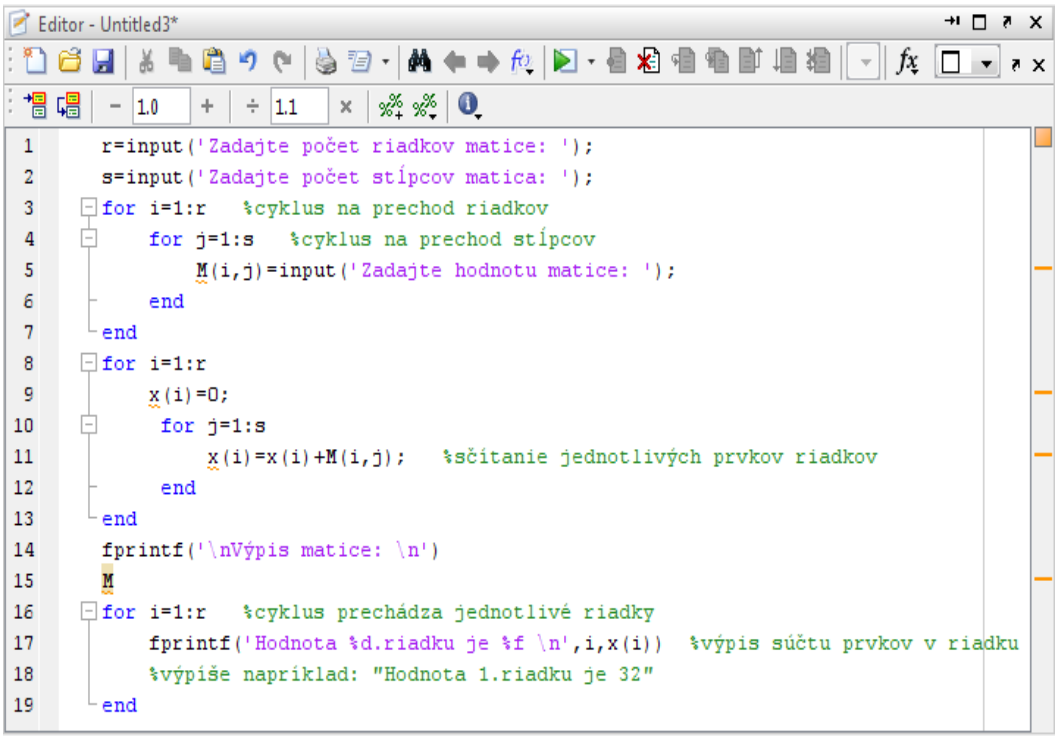


Obrázok 2-2 Spustenie Editora po vytvorení nového súboru funkcie

Ako už bolo spomenuté, existujú dva ***typy M- súborov*** a to : ***súbor so skriptom*** a ***súbor s funkciou***, v nasledujúcej tabuľke je ich porovnanie.

Skript	**Funkcia**
Nezadávajú sa vstupné argumenty a ani sa nevypisujú výstupné	Môže obsahovať vstupné a vracať výstupné premenné
Pracuje s premennými vo Workspace	Vnútorné premenné sú lokálne pre funkciu, Workspace „nevidí“
Vhodné pre automatizáciu volania tých istých príkazov viac krát	Vhodné pre rozšírenie funkčnej základne matlabovských vstavaných funkcií

Príklad 1
Vytvorte postupnosť príkazov - skript, v ktorom vyzvete užívateľa na zadanie rozmerov matice a jej následné naplnenie. Vypíšte súčet riadkov matice.



2.2 Tvorba vlastných funkcií v jazyku MATLAB

V prostredí MATLAB existujú *vstavané funkcie* (bult-in functions), ako napr. ***sin***, ***sqrt*** či ***abs***, ktoré bežne používame pri matematických výpočtoch. Ich názov hovorí na čo slúžia, ale užívateľ nemusí vedieť nič o výpočte vykonávajúcom sa vo vnútri funkcie. Užívateľ jednoducho zadá *vstupný parameter* a následne mu je vrátený *výstupný parameter*.
Výhodou využívania funkcií je „skrytie“ algoritmu vypočítavajúceho danú funkciu pod názov funkcie. Nie však všetky funkcie, ktoré používateľ potrebuje, sú vopred naprogramované, preto existujú ***používateľské funkcie***.

- **Funkcie** sú M-súbory s presne definovanou štruktúrou, ktoré obsahujú postupnosti príkazov a zároveň vstupné a výstupné premenné.

```
function [ output_args ] = Untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here


end
```

Charakteristické znaky funkcie :

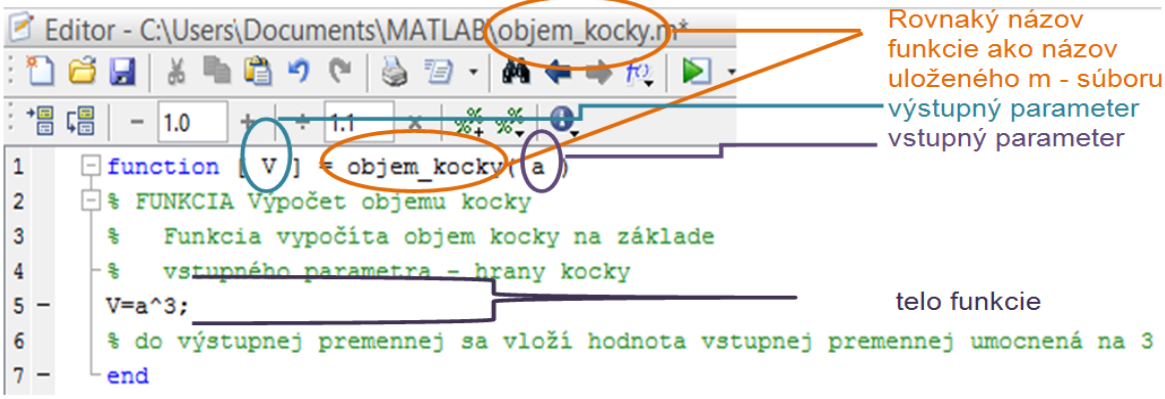
1. definícia funkcie – názov funkcie, vstupné a výstupné parametre
2. prvý riadok nápovedy – stručný popis funkcie, je prehľadávaný funkciou ***lookfor***
3. nápoveda – popis funkcie
4. vlastný kód

Poznámka :

⇨ nápoveda (body 2, 3) je nepovinná časť,
⇨ názov funkcie sa musí zhodovať s názvom súboru bez prípony - „.m“

Vstupné parametre môžu byť pri každom spustení funkcie iné. Výhodou funkcií je použitie jedného postupu - algoritmu pre rôzne hodnoty vstupných parametrov. V takomto prípade nie je vhodné použitie skriptov, keďže by sa hodnoty museli stále prepisovať.
Všetky premenné vo funkcii sú ***lokálne***, to znamená, že existujú len počas spustenia funkcie. Vo funkcii nie je možné použiť iné premenné ako tie, ktoré vložíme do vstupných parametrov, alebo tie, ktoré vo funkcii vytvoríme. Taktiež platí, že všetky premenné po skončení funkcie zaniknú. Vstupné parametre sú zadavané hodnotou, takže ak zmeníme ich hodnotu v tele funkcie nezmení sa tým hodnota premennej, s ktorou bola funkcia volaná.

Príklad 2
Vytvorte funkciu, ktorej vstupným parametrom bude strana kocky a výstupným objem kocky. Do tela funkcie napíšte kód, ktorým zrealizujete výpočet objemu kocky.



Obrázok 2-3 Zápis funkcie v Editore pre „výpočet objemu kocky“

V príkazovom riadku si pri spustení funkcie s názvom ***objem_kocky*** do premennej *a* vložíme hodnotu hrany kocky 12 a pomocou vytvorenej ***funkcie objem_kocky*** bude vypočitaný jej objem ***V***, ktorý uložíme do premennej ***X***.

```
Command Window
>> X=objem_kocky(12)

X =

        1728

fx >>
```

Obrázok 2-4 Spustenie funkcie objem_kocky pre definovaný parameter strany kocky *a* = 12

Príklad 3

Vytvorte funkciu, ktorej výstupná premenná bude obsahovať vypočitaný n – faktoriál čísla pričom vstupný parameter bude číslo, ktorého faktoriál sa počíta.

Editor - C:\Users\ \Documents\MATLAB\faktorial.m

```
1  function [ nfaktorial ] = faktorial( x )
2  %FUNKCIA výpočet n - faktoriálu
3  %   vypočíta faktoriál vstupného argumentu x
4  nfaktorial=1;
5  for i=1:x
6      nfaktorial=nfaktorial*i;
7  end
8
9  end
```

Obrázok 2-5 Výpočet *n* – faktoriálu pre zadané číslo – riešené pomocou *funkcie*

V príkazovom riadku si do premennej *N* vložíme vypočitanú hodnotu faktoriálu čísla 5 (5!), ktorý bol vypočitaný pomocou vytvorenej funkcie ***faktorial*** pre zvolený parameter funkcie $x = 5$.

```
Command Window
>> N=faktorial(5)

N =

   120
```

Obrázok 2-6 Spustenie funkcie *faktorial(5)*

Príklad 4

Vytvorte funkciu na výpočet obsahu trojuholníka. V hlavnom programe vyzvite užívateľa na zadanie strany a výšky pre vytvorenú funkciu.

```
% HLAVNÝ PROGRAM
b=input('Zadajte stranu trojuholníka: ');
vb=input('Zadajte výšku na zadanú stranu: ');
St=obsah_trojuholnika(b,vb);
fprintf('Obsah trojuholníka je : %f',St);
```

```
% FUNKCIA
function [ S ] = obsah_trojuholnika( a,va )
S=a*va/2;
end
```

Príklad 5
Vytvorte funkciu, ktorá vypočíta hodnotu zlomku

$$\frac{(a^2-3a-2)}{(a-1)^2(a+2)}.$$

Túto funkciu použite v hlavnom programe, pomocou ktorého budete môcť vypočítať hodnotu funkcie pre premennú ***a*** zadanu z klávesnice. Skontrolujte či pre danú hodnotu ***a*** má výraz riešenie. Ak nie výsledok bude 0. Výstup zobrazte.

```
function [ H ] = zlomok( a )
if (a==1 || a==-2)
    H=0;
else
    H=(a^2-3*a-2)/((a-1)^2*(a+2));
end
end
```

Príklad 6
Vytvorte funkciu, ktorá bude využívať Pytagorovu vetu pre výpočet chýbajúcej strany trojuholníka. V hlavnom programe (skripte) ponúknite možnosť výberu chýbajúcej strany - prepona/odvesna a zohľadnite to aj vo funkcii. Dĺžky zvyšných dvoch strán zadá užívateľ na vstupe.

Kód v jazyku MATLAB

```
function [ x ] = pytagoras( o,a,b )
switch o
    case {1}
        x=sqrt(a*a+b*b);
    case {2}
        if a>b
            x=sqrt(a*a-b*b);
        else x=sqrt(b*b-a*a);
        end
end
end
```

Kód hlavného programu :

```
o=input('Pre výpočet prepony zadaj "1" \nPre výpočet odvesny zadaj "2"\n');
while (o>2 | o<1)
o=input('Pre výpočet prepony zadaj "1" \nPre výpočet odvesny zadaj "2"\n');
end
a=input('Zadajte hodnotu prvej strany: ');
b=input('Zadajte hodnotu druhej strany: ');
c=pytagoras(o,a,b)
```

Príklad 7
Vytvorte funkciu, ktorá bude vracať súčet vektorov, súčin vektorov po prvkoch a súčet absolútnych hodnôt jednotlivých prvkov vektorov. Vektory budú vstupnými argumentami funkcie.

Kód v jazyku MATLAB

```
function [ sucin, sucet, SAH ] = vektory( v1,v2 )
% sucin - výstupný vektor, ktorý predstavuje súčin vstupných vektorov
% sucet - výstupný vektor, ktorý predstavuje súčet vstupných vektorov
% SAH - výstupná hodnota - súčet absolútnych hodnôt jednotlivých prvkov vstupných vektorov
if length(v1)~=length(v2)
    error('vektory nemajú rovnakú dĺžku')
else
    sucet=v1+v2;
    SAH=0;
    for i=1:length(v1)
        SAH=SAH+abs(v1(i))+abs(v2(i));
        sucin(i)=v1(i)*v2(i);
    end
end
end
```

Príklad 8
Vytvorte funkciu ktorej vstupnými parametrami budú dve ľubovoľné matice. Funkcia porovná ich rozmery, v prípade nerovnosti rozmerov vypíše chybu, v opačnom prípade bude hodnotiť prvky matíc na rovnakých pozíciách na základe ich párnosti. Výsledky zapíše do výslednej matice, ktorá bude výstupným argumentom, a to tak, že do nej zapíše počet párnych prvkov na danej pozícii(0,1 alebo 2)

Kód v jazyku MATLAB

```
function [ V ] = matice( A,B )
if size(A)~=size(B)
    error('Zadané matice nemajú rovnaké rozmery')
else
    a=size(A);
    for i=1:a(1)    %cyklus na prehliadanie riadkov matíc
        for j=1:a(2)    %cyklus na prehliadanie stĺpcov matíc
            if rem(A(i,j),2)==0 %overenie párnosti prvku matice A na pozícii (i,j)
                if rem(B(i,j),2)==0 %overenie párnosti prvku matice B na pozícii (i,j)
                    V(i,j)=2;   %prípad, že obidve podmienky sú vyhodnotené ako pravdivé
                else
                    V(i,j)=1;   %prvá podmienka platí a druhá nie
                end
            else
                if rem(B(i,j),2)==0 %overenie párnosti prvku matice B na pozícii (i,j)
                                    %v prípade,že prvok matice A na danej pozícii nie je párny
                    V(i,j)=1;   %prvok matice A nepárny a prvok matice B párny
                else
                    V(i,j)=0;   %prípad, že obidve podmienky sú vyhodnotené ako nepravdivé
                end
            end
        end
    end

end
```

2.3 Príklady na riešenie

1. Vytvorte funkciu, ktorej výstupný parameter bude štvrtá mocnina vstupného parametra.

2. Vytvorte funkciu, ktorá vypočíta obsah trojuholníka so vstupnými premennými základňou a výškou.

3. Vytvorte funkciu, ktorá vráti skalárny súčin zadaných vektorov.

4. Máme funkciu výnosov $f(b)$=895b+64-2^b. Vytvorte funkciu v Matlabe, ktorá vytvorí vektor hodnôt funkcie výnosov na základe vstupného vektora a vyhľadá bod, v ktorom sú výnosy najvyššie. Použite funkciu v hlavnom programe, v ktorom si vyžiadate ručný vstup hodnôt, ktoré budú vstupnými hodnotami pre funkciu výnosov. Na vstup dajte kontrolu, či ste zadali vektor. Ak nie vytvorte cyklus, aby sa funkcia vykonala až po správnom zadaní hodnôt. Vo výsledku vypíšte hodnoty funkcie a aj najvyššiu hodnotu a bod.

5. Vytvorte funkciu, ktorá vynásobí 2 vektory medzi sebou po prvkoch a zo súčinu vypíše všetky čísla deliteľné 3 alebo 4. Vektory zadajte z klávesnice a výsledky vypíšte.

6. Vytvorte funkciu, ktorej vstupné parametre budú dva vektory rovnakej dĺžky. Funkcia má umocniť každý *i*-ty prvok 1.vektora *i*-tym prvkom 2.vektora.

7. Vstupnými parametrami funkcie nech sú dve matice. Vytvorte funkciu na porovnanie týchto matíc. Funkcia zistí či sú matice rovnakých rozmerov. V prípade že nie sú výsledkom bude 0. V opačnom prípade bude funkcia postupne porovnávať prvky matíc na rovnakých pozíciách. Výsledky porovnania zapíše do výslednej matice nasledovne. Ak je prvok prvej matice väčší ako prvok druhej matice do výslednej matice sa zapíše 1, ak je prvok prvej matice menší ako prvok druhej matice do výslednej matice sa zapíše 2. V prípade, že sa prvky rovnajú zapíše sa 0.

2.4 Postup pre vytvorenie m-súboru, skriptu a funkcie v Editore programového prostredia MATLAB

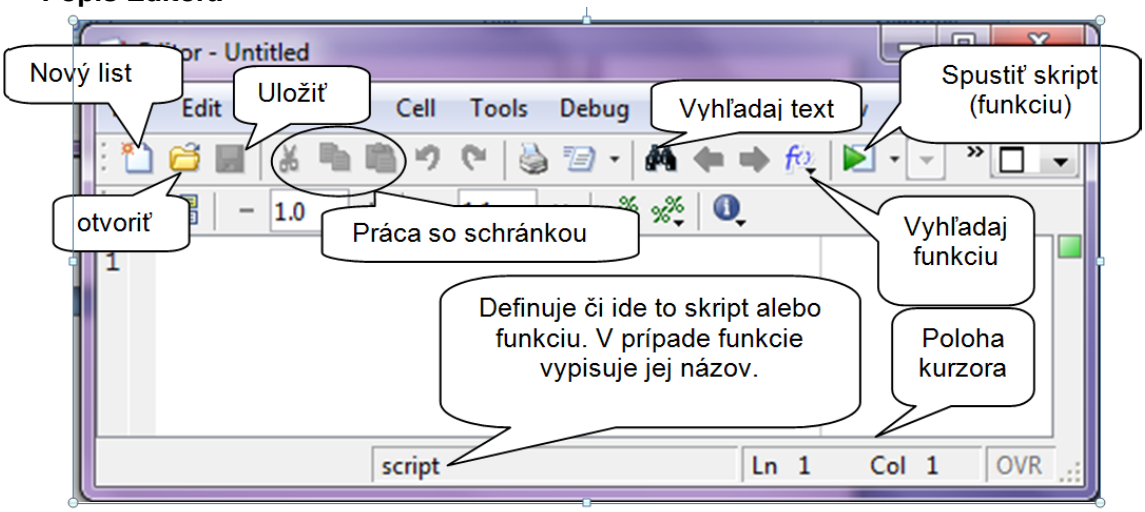
- **M-súbory**

⇨ ***m-súbory*** sú zapisované do editora, ktorý je súčasťou programového prostredia MATLAB

⇨ otvorenie nového m-súboru v prostredí MATLAB je možné po kliknutí na ikonu 

⇨ ***m-súbory*** môžu byť ***skripty*** alebo ***funkcie***

- **Popis Editora**



Obrázok 2-7 Editor programového prostredia MATLAB a práca v ňom

⇨ zvýrazňuje kľúčové slová simulačného jazyka MATLAB
⇨ umožňuje krokovať obsah *m-súborov*

- **Skripty**

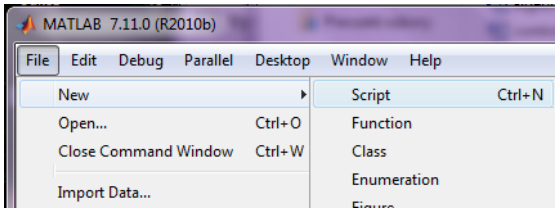
⇨ skript je po obsahovej stránke postupnosť príkazov simulačného jazyka MATLAB zapísaných do súboru pod určitým menom
⇨ skripty neprijímajú vstupné a nevracajú výstupné argumenty, pracujú s dátami uloženými v pamäťovom okne *Workspace* (okno na vizualizáciu obsahu používaných premenných)
⇨ v skripte použité funkcie pracujú s údajmi v základnom pracovnom priestore
⇨ súbory sú ukladané s jedinečným **menom** a príponou **.m** ***(meno_suboru.m)***
⇨ premenné, ktoré sú pred použitím skriptu definované, môžeme v skripte použiť
⇨ premenné, ktoré sú vytvorené počas vykonávania skriptu, zostanú po ukončení skriptu zachované

- **Tvorba skriptu**

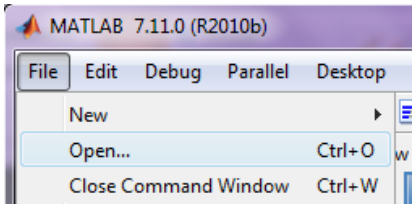
a. Najskôr si musíme v pracovnom priestore MATLAB-u nastaviť cestu k pracovnému adresáru

Otvorenie nového skriptu prebieha v Editore ikonou 

alebo



ak meníme existujúci skript, použijeme  alebo



b. do prázdneho listu editora sa napíše postupnosť príkazov, ktoré sa v tomto prípade nevykonávajú hneď po napísaní a stlačení klávesy ENTER

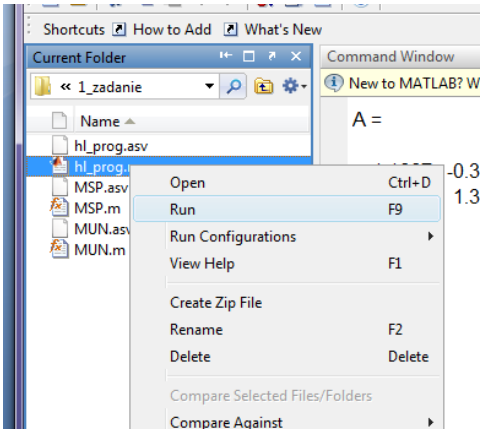
c. napísanú postupnosť príkazov je potrebné uložiť pod nejakým menom na disk, pre uloženie môžeme použiť ikonu  alebo *File/Save As...*

Ak sa pokúsime spustiť neuložený skript, vyskočí nám rovno okno pre uloženie, t.j. neuložený skript nevieme spustiť

d. Volanie/spustenie skriptu sa vykonáva prostredníctvom zápisu mena skriptu v príkazovom okne (Command Window),

V tomto prípade sa musí skript nachádzať v adresári, v ktorom sa aktuálne nachádzame alebo tam musi byť nastavená cesta .

Spustenie skriptu sa môže vykonať kliknutím na ikonu  v Editore, alebo kliknutím na skript, ktorý chceme otvoriť pravým tlačidlom myši a vybrať možnosť *Run*



Obrázok 2-8 Spustenie vytvoreného skriptu

- **Funkcie**

⇨ funkcie sú najefektívnejším nástrojom pre modulárne programovanie a automatizáciu úlohy
⇨ prijímajú vstupné a vracajú výstupné argumenty

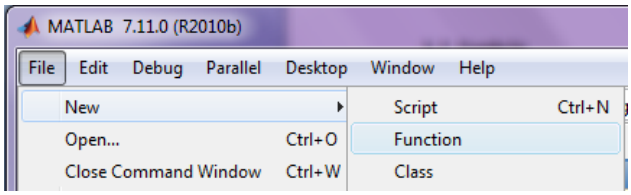
⇨ premenné novo vytvorené pri behu funkcie sú lokálne a po ukončení posledného príkazu zanikajú (ak nechceme, aby zanikli, musíme ich definovať ako globálne – príkazom ***global nazov_premennej***)

- **Tvorba vlastných funkcií**

a. Najskôr si musíme v pracovnom priestore Matlab-u nastaviť cestu k pracovnému adresáru

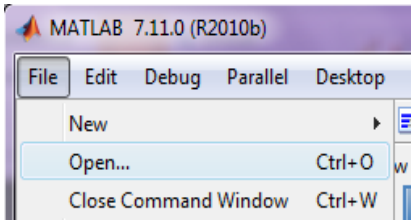
b. Otvorenie nového listu Editora ikonou 

alebo



ak chceme zmeniť už existujúcu funkciu použijeme 

alebo



c. Ako prvé musíme napísať hneď na začiatku deklaráciu funkcie a to :

function [výstup_1, výstup_2, ...] = meno_funkcie (vstup1, vstup2).

- funkcie nemusia mať žiaden vstup ani výstup
- ak existuje len jeden výstup nemusí byť v zátvorkách
- ak existuje viac vstupov (výstupov), oddeľujeme ich čiarkou
- názov funkcie by mal vystihovať jej funkčnosť
- názov sa nesmie zhodovať s názvom už existujúcej funkcie vstavanej v Matlabe alebo uživateľom predtým vytvorenej funkcie.

d. Ďalej môžeme pokračovať podobne ako pri tvorení skripu postupnosťou príkazov, ktoré však nie sú vykonávané hneď a samostatne ale až po **zavolaní funkcie**. xxx

e. Takúto funkciu musíme uložiť pod rovnakým názvom ako je ***meno_funkcie***. Ak ju uložíme aj s malým rozdielom, pri spustení ju nebude vedieť nájsť a systém nám vypíše chybu.

f. Volanie funkcie sa vykonáva v pracovnom priestore, kde funkciu zavoláme jej menom, alebo ju použijeme a voláme v skripte.

g. Ak pri tvorení funkcie si ju popíšeme pomocou komentárov, začínajúcich za deklaráciou funkcie a končiacich prvým príkazom, tieto komentáre si vieme zobraziť po zadaní príkazu ***help meno_funkcie*** alebo ***lookfor kľúčove_slovo*** v príkazovom riadku

```
>> help MSP
  Funkcia pre výpočet metódou slučkových prúdov
```

Príklad 1

a. Vytvor v simulačnom jazyku MATLAB skript pre výpočet odvesny pomocou Pytagorovej vety $a^2 + b^2 = c^2$.

```
a=input('Zadaj dĺžku prvej odvesny:  ');
b=input('Zadaj dĺžku druhej odvesny:  ');
c = sqrt(a^2 + b^2)
```

b. Vykonajte ten istý výpočet avšak s použitím funkcie pre výpočet odvesny Pytagorovou vetou.

prepona.m

```
function c=prepona(a,b)

c = sqrt(a^2 + b^2);
```

hl_prog.m **%skrip hlavného programu**

```
a=input('zadaj hodnotu prvej odvesny:  ');
b=input('zadaj hodnotu druhej odvesny:  ');
c=prepona(a,b)
```

2.5 Zadanie č. 1: Riešenie lineárnych algebraických rovníc s aplikáciou na elektrické obvody metódou slučkových prúdov a uzlových napätí v prostredí MATLAB

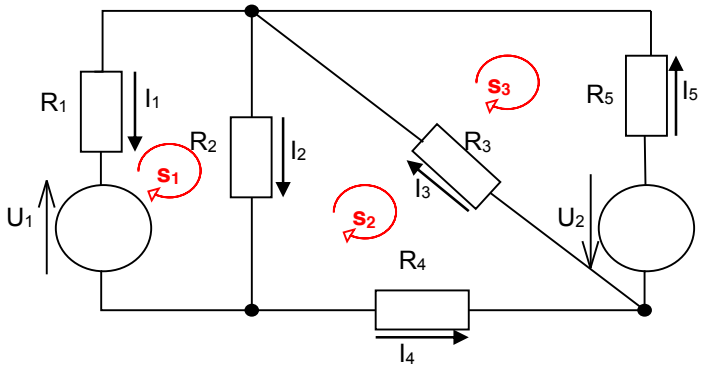
*Nasledujúce uvedené poznatky z oblasti riešenia elektrických obvodov pomocou **metódy slučkových prúdov** a **uzlových napätí** je potrebné využiť pri riešení Zadania č. 1. Vzorové zadanie číslo 1 sa nachádza nižšie.*

Metóda slučkových prúdov

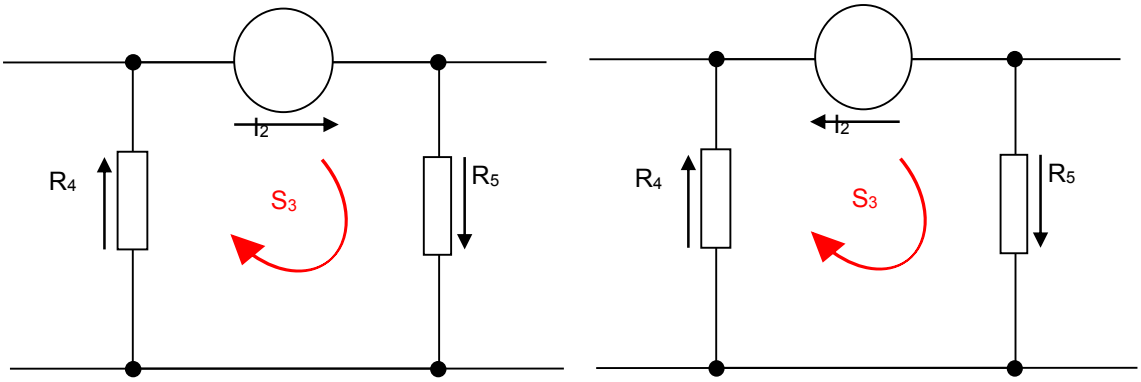
- je založená na poznatku, že prúdy vo vetvách stromu grafu sú jednoznačne určené prúdmi v nezávislých vetvách grafu
- spočíva v aplikácii 2. Kirchhoffovho zákona na všetky základné slučky grafu za predpokladu, že nimi tečie fiktívny, tzv. slučkový prúd, čím získame podmienkové rovnice pre daný obvod.

Postup:

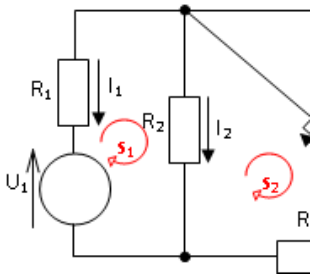
1. Zvolíme si smery slučkových prúdov v jednotlivých slučkách, smery napätia na zdrojoch a prúdy pretekajúce rezistormi.



2. Zistíme či sa v niektorej vetve nenachádza prúdový zdroj, potom sa hodnota slučkového prúdu bude rovnať prúdu zdroja s kladným alebo záporným znamienkom závisiacim od orientácie týchto prúdov.



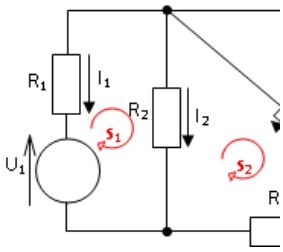
3. Rovnice zostavujeme nasledovne: slučkovým prúdom danej slučky vynásobíme súčet odporov danej slučky, ak niektorý rezistor susedí s ďalšími slučkami, potom odčítame ich súčin (odporu a slučkového prúdu), a nakoniec ak sa nachádza v slučke aj napäťový zdroj pričítame ho s kladným alebo záporným znamienkom podľa jeho smeru prúdenia.



$$\Rightarrow I_{s1} * (R_1 + R_2) - I_{s2} * R_2 + U_1 = 0$$

4. Za jednotlivé odpory a napätia a prúdy zdrojov dosadíme ich hodnoty a zapíšeme do rozšírenej matice (prúdy, ktoré sme dostali v 2.bode už do tejto matice nezapisujeme), z ktorej vypočítame hodnoty zvyšných slučkových prúdov.Konštanty dávame na pravú stranu matice. Ak sme zostavili rovnice správne matica by mala byť zrkadlová vzhľadom na hlavnú diagonálu.

5. Vytvoríme si podmienkové rovnice, z ktorých dostaneme konkrétne prúdy na jednotlivých rezistoroch.

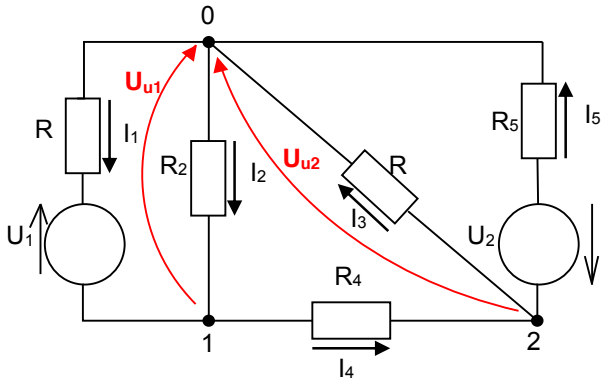


$$\Rightarrow I_1 = -I_{s1}$$
$$\Rightarrow I_2 = I_{s1} - I_{s2}$$
$$\Rightarrow \dots$$

Metóda uzlových napätí

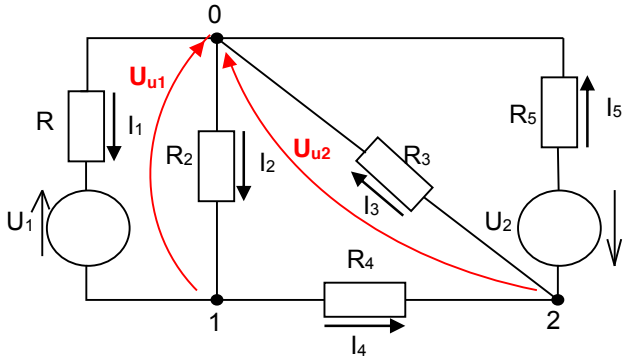
- je založená na poznatku, že napätia na nezávislých vetvách grafu sú jednoznačne určené napätiami na vetvách stromu grafu.
- spočíva v aplikácii 1. Kirchhoffovho zákona na všetky nezávislé uzly grafu za predpokladu, že na vetvách stromu grafu sú fiktívne, tzv. uzlové napätia, čím získame podmienkové rovnice pre daný obvod

1. Zvolíme si smery napätia na zdrojoch, prúdy pretekajúce odpormi a referenčný uzola označíme si aj zvyšné uzly.



2. Rovnice pre metódu uzlového napätia tvoríme nasledovne: pre každý uzol okrem referenčného vytvoríme rovnicu tak, že napätie od uzla vynásobíme súčtom prevrátených hodnôt odporov na prislúchajúcich vetvách a od toho odčítame súčin napätí a prevrátených

hodnôt z susedných uzlov. Nakoniec ešte pričítame súčin napätia na napäťovom zdroji a súčet prevrátených hodnôt na spoločnej vetve. Vo vetva s ideálnym napäťovým zdrojom je uzlové napätie rovné napätiu na zdroji.



$$U_{u1} * \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_4}\right) - U_{u2} * \frac{1}{R_4} - U_1 * \frac{1}{R_1} = 0$$

3. Zo zostavených rovníc zostavíme rozšírenú maticu dosadením za odpory a známe napätia ich hodnoty. Konštanty dávame na pravú stranu matice. Ak sme správne zostavili rovnice mala by byť matica zrkadlová vzhľadom na hlavnú diagonálu.

4. Vypočítaním matice dostaneme hodnoty uzlových napätí z ktorých si následne vytvoríme podmienkové rovnice pre výpočet jednotlivých prúdov.

$$I_1 = \frac{U_1 - U_{u1}}{R_1}, \quad I_3 = \frac{U_{u2}}{R_3}$$

ZADANIE 1:

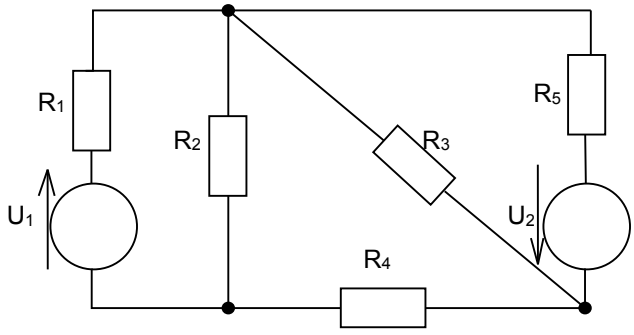
Z navrhnutej topológie elektrického obvodu vypočítajte prúdy vo vetvách metódou slučkových prúdov (MSP) a metódou uzlových napätí (MUN).

1. *Zvoliť topológiu obvodu a vykonať analytický výpočet pre obidve metódy a skúšku správnosti.*
2. *Algoritmicky vyriešiť a simulačne overiť v programovom prostredí MATLAB výpočet prúdov s využitím funkcií (MSP a MUN), [minimálne požiadavky na elektrický obvod: 3 slučky, 5 rezistorov, 2 zdroje]*

ÚLOHA:

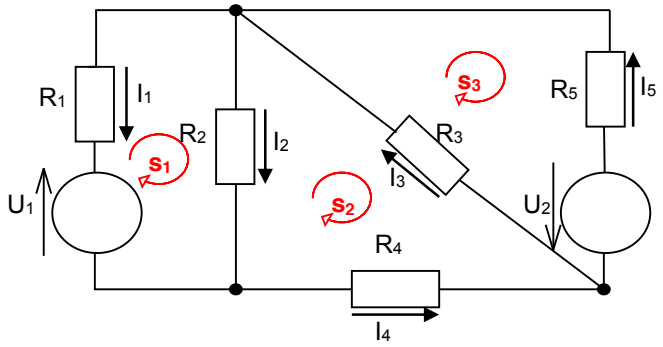
Vyriešiť analyticky zadaný obvod pomocou metódy slučkových prúdov a uzlových napätí a vytvor v programovom prostredí MATLAB program pre výpočet prúdov tohto obvodu.

$R_1 = 2\Omega$, $R_2 = 3\Omega$, $R_3 = 2\Omega$, $R_4 = 3\Omega$, $R_5 = 2\Omega$, $U_2 = 10$ V, $U_1 = 10$ V



1.a) **Metóda slučkových prúdov**

S1: $I_{s1} * (R_1 + R_2) - I_{s2} * R_2 = -U_1$
S2: $I_{s2} * (R_2 + R_3 + R_4) - I_{s1} * R_2 - I_{s3} * R_3 = 0$
S3: $I_{s3} * (R_3 + R_5) - I_{s2} * R_3 = -U_2$



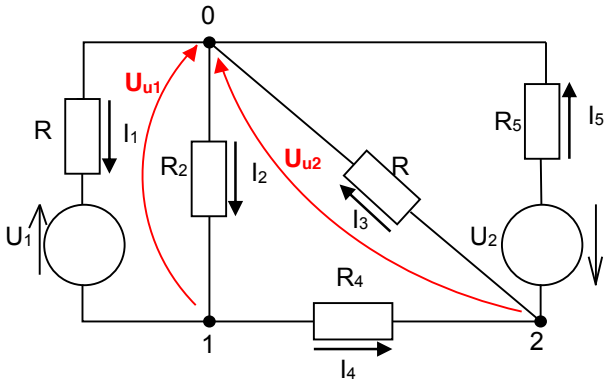
Zo získaných algebraických rovníc ***Ax = B*** vieme zostaviť maticu odporov ***A*** vektor naätí ***B***, a vypočítať prvky vektora x čo sú neznáme slučkové prúdy.

$$\begin{pmatrix} 5 & -3 & 0 \\ -3 & 8 & -2 \\ 0 & -2 & 4 \end{pmatrix} \begin{vmatrix} -10 \\ 0 \\ -10 \end{vmatrix} \Rightarrow \begin{matrix} I_{s1} = -\frac{85}{26} \\ I_{s2} = -\frac{55}{26} \\ I_{s3} = -\frac{185}{52} \end{matrix} \Rightarrow \begin{matrix} I_1 = -I_{s1} = \frac{85}{26} \\ I_2 = I_{s1} - I_{s2} = -\frac{15}{13} \\ I_3 = I_{s3} - I_{s2} = -\frac{75}{52} \\ I_4 = -I_{s2} = \frac{55}{26} \\ I_5 = -I_{s3} = \frac{185}{52} \end{matrix}$$

1.b) **Metóda uzlových napätí**

1: $U_{u1} * \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_4}\right) - U_{u2} * \frac{1}{R_4} = U_1 * \frac{1}{R_1}$

2: $U_{u2} * \left(\frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_5}\right) - U_{u1} * \frac{1}{R_4} = -U_2 * \frac{1}{R_5}$



Upravíme na tvar ***Ax = B***:

$$\left(\begin{array}{cc|c} \frac{6}{7} & -\frac{1}{3} & 5 \\ -\frac{1}{3} & \frac{4}{3} & -5 \end{array}\right) \Rightarrow \begin{array}{l} U_{u1} = \frac{45}{13} \\ U_{u2} = -\frac{75}{26} \end{array} \Rightarrow \begin{array}{l} I_1 = \frac{U_1 - U_{u1}}{R_1} = \frac{85}{26} \\ I_2 = \frac{U_{u1}}{R_2} = \frac{15}{13} \\ I_3 = \frac{U_{u2}}{R_3} = \frac{75}{52} \\ I_4 = \frac{U_{u1} - U_{u2}}{R_4} = \frac{55}{26} \\ I_5 = \frac{U_{u2} + U_2}{R_5} = \frac{185}{52} \end{array}$$

Pre jednotlivé výpočty si vytvoríme samostatné funkcie a pre hlavný program vytvoríme skript, v ktorom naše vytvorené funkcie použijeme.

2a. Programový návrh funkcie pre výpočet slučkových prúdov a prúdov v jednotlivých vetvách metódou slučkových prúdov

MSP.m

```
function I= MSP(U,R)
% Funkcia pre výpočet prúdov metódou slučkových prúdov

% vytvorte maticu A, ktorá bude obsahovať hodnoty ľavých strán rovníc
A = [R(1)+R(2)                -R(2)                              0;
-R(2)            R(2)+R(3)+R(4)            -R(3);
0                      -R(3)                      R(3)+R(5)]
% vytvorte maticu B, ktorá bude obsahovať hodnoty pravých strán rovníc
B = [ - U(1);
0;
- U(2)]

%výpočet slučkových prúdov realizujte ľavým delením
IS = A\B

%dopočítajte zvyšné hodnoty prúdov
I(1) = -IS(1);
I(2) = IS(1) - IS(2);
I(3) =  IS(3) -  IS(2);
I(4) =  -IS(2);
I(5) = - IS(3);
```

2b Programový návrh funkcie pre výpočet uzlových napätí a prúdov v jednotlivých vetvách

MUN.m

```
function I=MUN(U,R)
%metóda uzlových napätí

%vytvorte maticu A, ktorá bude obsahovať hodnoty ľavých strán rovníc
A = [(1/R(1)+1/R(2)+1/R(4)) -1/R(4); -1/R(4)  (1/R(3)+1/R(4)+1/R(5)) ]

%vytvorte maticu B, ktorá bude obsahovať hodnoty pravých strán rovníc
B = [ U(1)*(1/R(1));         -U(2)*(1/R(5))]
```

```
% výpočet uzlových napätí realizujte ľavým delením
Uu = A\B

%dopočítajte zvyšné hodnoty prúdov
I(1) = (U(1)-Uu(1))/R(1);
I(2) = Uu(1)/R(2);
I(3) = Uu(2)/R(3);
I(4) = (Uu(1)-Uu(2))/R(4);
I(5) = (Uu(2)+U(2))/R(5);
```

Skúška správnosti výpočtu

skuska.m

```
function skuska(I)

if I(1)-I(2)-I(4)<1e-6
if I(4)+I(3)-I(5)<1e-6
if I(2)-I(1)-I(3)+I(5)<1e-6
            disp('Prúdy vo všetkých vetvách vyhovujú 1.KZ.')
else disp('Prúdy vo všetkých vetvách nevyhovujú 1KZ. ')
end
else disp('Prúdy vo všetkých vetvách nevyhovujú 1KZ. ')
end
else disp('Prúdy vo všetkých vetvách nevyhovujú 1KZ. ')
end
```

Hlavný program – načítanie hodnôt odporov a napätí a použitie vytvorených uživateľských funkcií MUN a MSP

hl_prog.m

```
% zadávanie hodnôt odporov
R1 = input ('Zadaj hodnotu R1 = ');
R2 = input ('Zadaj hodnotu R2 = ');
R3 = input ('Zadaj hodnotu R3 = ');
R4 = input ('Zadaj hodnotu R4 = ');
R5 = input ('Zadaj hodnotu R5 = ');
disp('************************************************************')
% zadávanie hodnôt prúdov
U1 = input ('Zadaj hodnotu napäťového zdroja U1 = ');
U2 = input ('Zadaj hodnotu napäťového zdroja U2 = ');

disp('************************************************************')
disp('')
disp('')
R = [R1, R2, R3, R4, R5 ];
U = [U1, U2 ];
disp ('metoda sluckovych prudov')
I=MSP(U,R)
disp('***************skúška*****************')
skuska(I)
disp('************************************************************')
disp('metoda uzlovych napati')
I=MUN(U,R)
disp('***************skúška*****************')
skuska(I)
```

3 Riešenie úloh regresnej analýzy v prostredí MATLAB s využitím dátových súborov

3.1 Práca s binárnymi a textovými súbormi

MATLAB umožňuje spracovanie súborov rôznych formátov. Nemusíme mať dáta uložené priamo v prostredí MATLAB. V tomto programovom prostredí dokážeme exportovať a importovať dáta vo formáte, aký si zvolíme. V tejto kapitole sa budeme venovať súborom typu mat, textovým súborom a xls súborom a prácou s nimi.

- **Ukladanie a načítavanie dát do/z *mat* súborov**

⇨ **Ukladanie dát**

Na ukladanie dát do súboru bola v Matlabe vytvorená funkcia ***save***. Táto funkcia ukladá premenné do binárneho súboru. Meno súboru do ktorého sa dáta uložia vieme ovplyvniť tým, že za názvom funkcie dopíšeme nami zvolený názov súbor. V prípade, že názov nezadáme, Matlab nám premenné uloží do súboru *subor.mat*. Keď chceme do súboru uložiť len niektoré premenné, je potrebné ich špecifikovať za názvom súboru, v opačnom prípade Matlab uloží do súboru všetky premenné Workspace.

Uloženie do súboru	subor.mat	nazov.mat	nazov.mat
Uloženie premenných	všetkých	všetkých	b, c, d
Príkaz	`a=1; b=4; c=3.8; d=b*c;` `save`	`a=1; b=4; c=3.8; d=b*c;` `save nazov.mat`	`a=1; b=4; c=3.8; d=b*c;` `save nazov.mat b c d`

Pri zadávaní premenných, ktoré chceme uložiť je možné použiť aj tzv. žolíky označované znakom hviezdičky (*). Premenné, ktoré majú časť svojho názvu rovnakú nemusíme vypisovať jednotlivo, ale využijeme spomínaného žolíka.

```
a=1; prem1=4; prem2=2.1; b=6; prem3=0.3; prem4=6;
save nazovsuboru.mat prem*
```

Takto zadaný príkaz nám uloží do súboru ***nazovsuboru.mat*** len premenné *prem1, prem2, prem3, prem4*.

MATLAB nám takto zadaným príkazom uloží dáta v **binárnom formáte**. Ďalšími príkazmi vieme zmeniť aj formát a to nasledovne :

```
save nazovsuboru.mat % štandardný binárny formát
save nazovsuboru.mat -ascii % 8-bitový textový výstup
save nazovsuboru.mat -ascii -tabs % 8-bitový textový výstup oddelený tabulátormi
save nazovsuboru.mat -ascii -double % 16-bitový textový výstup
save nazovsuboru.mat -ascii -double -tabs % 16-bitový textový výstup oddelený tablátormi
```

⇨ **Načítavanie dát**

Import dát z Mat súborov je veľmi jednoduchý pomocou príkazu ***load***. Tak ako pri ukladaní aj pri načítavaní môžeme načítať celý súbor, len určité premenné alebo využiť žolíka.

```
load nazovsuboru.mat
load nazovsuboru.mat prem1 prem2
load nazovsuboru.mat prem*
```

Predtým ako si dáta zo súboru načítame, môžeme sa pozrieť aké dáta v danom súbore sú uložené. Príkaz ***whos*** s prepínačom *file* a názvom súboru nám zobrazí názov premenných, veľkosť, počet bytov a triedu.

```
>> whos -file nazovsuboru.mat
  Name        Size              Bytes  Class     Attributes

  prem1       1x1                   8  double
  prem2       1x1                   8  double
  prem3       1x1                   8  double
  prem4       1x1                   8  double
```

- **Ukladanie a načítavanie dát do/z textových súborov**

⇨ **Ukladanie dát**

MATLAB má niekoľko vstavaných funkcií pre ukladanie textových súborov. Ich použitie závisí od množstva ukladaných dát a od formátu súboru, do ktorého chceme dáta uložiť. Pri binárnych súboroch MAT sme využívali najjednoduchšiu funkciu ***save***. Táto funkcia taktiež umožňuje ukladanie do ASCII súborov. Pri takomto uložení dát sa ako oddeľovač používa medzera.

```
a=2; b=3.1; C=[1,2,3;6,8,0];
save nazovsuboru.out -ASCII
```

Funkcia ***dlmwrite*** tiež slúži na ukladanie dát do textového súboru. Na rozdiel do funkcie ***save*** si pri tejto funkcii môžeme sami zvoliť znak slúžiaci na oddeľovanie dát. Oddeľovač sa do príkazu pridáva v úvodzovkách.

```
A=[1 2 8; 6 3 7];
dlmwrite('data.out',A, ';')
```

Ďalšou funkciou slúžiacou na ukladanie textových súborov je funkcia ***csvwrite***, ktorá je primárne určená pre dáta využívajúce tabuľkový procesor. Pri tejto funkcii sa oddeľovač nešpecifikuje, je to vždy čiarka.

```
A=[1 2 8; 6 3 7];
csvwrite('csvdata.out',A)
```

Poslednou funkciou na ukladanie dát do textového súboru je ***diary***. Ukladá výstup príkazového riadku MATLABus.

```
diary data.out
A=[1 5 8 ; 2 4 7; 2 0 1];
A
diary off
type data.out
```

Príkazom uvedeným v poslednom riadku vypíšeme obsah súboru *data.out* do príkazového riadku.

⇨ **Načítavanie dát**

Na načítavanie dát taktiež existuje niekoľko vstavaných funkcií. Výber funkcie, ktorá sa na import dát použije závisí od naformátovanie dát v súbore. Textové dáta musia byť naformátované do rovnakého počtu stĺpcov v jednotlivých riadkoch. Ako oddeľovač dát slúži tzv. oddeľovací znak.

Najjednoduchší spôsob načítania dát je pomocou funkcie ***load***. Táto funkcia sa využíva v prípade keď sú dáta uložené do obdĺžnika (v každom riadku je rovnaký počet stĺpcov). Dáta budú vložené do ***Workspace*** s menom totožným ako je názov súboru bez koncovky. Dáta je možné uložiť aj s názvom, ktorý si užívateľ zvolí sám (prípad v 2. riadku – dáta sa uložia do premennej A)

```
load data.txt
A=load('data.txt')
```

V prípade, že máme ako oddeľovač použitý iný znak ako je medzera, môžeme na načítanie dát použiť funkciu ***dlmread***. Táto funkcia ignoruje medzery medzi dátami. Ako druhý parameter je potrebné zadať znak oddeľovača. Nepovinnými parametrami je posun, od ktorého chceme začať načítavanie. Indexovať sa začína od 0,0. V druhom riadku príkladu je uvedený príkaz na načítanie dát od prvku nachádzajúceho sa v 3. riadoku 2. stĺpci.

```
A = dlmread('data.txt', ';');
A = dlmread('data.txt', ';', 3, 2);
```

Čítanie dát z textových súborov uložených v ***csv*** súboroch sa realizuje funkciou ***csvred***. V takýchto súboroch ako oddeľovač slúži čiarka, preto sa v príkaze parameter oddeľovača nezadáva. Zvyšné parametre zostávajú také isté ako pri funkcii ***dlmread***.

```
A=csvread('data.txt', 2, 3);
```

V prípade, že textový súbor obsahuje hlavičky k dátam, je potrebné použiť funkciu ***textscan***, v tejto funkcii sa môže využiť parameter ***headerlines,*** ktorý umožňuje ignorovanie zadaného počtu riadkov od začiatku súboru.

Majme súbor *data.txt*, v ktorom sú nasledovné dáta :

```
Stĺpec1     Stĺpec2     Stĺpec3     Stĺpec4     Stĺpec5
2.154       1.256       3.785       24.125      2.536
3.256       1.401       2.869       36.254      1.258
3.987       1.569       3.028       12.548      3.685
```

V takomto prípade je potrebné najskôr otvoriť súbor na čítanie. Identifikátor súboru si vložíme do premennej ***idenif***. Funkciou ***textscan*** načítame 3 riadky, znakom ***%f*** udávame, že chceme aby sa údaje uložili v desatinnom formáte. Ďalej využijeme parameter ***headerlines*** a za ním udáme číslo, ktoré nám označuje počet riadkov, ktoré sú v súbore uvedené ako hlavičky. Súbor je potrebné zavrieť a to príkazom ***fclose***.

```
identif = fopen('data.txt','r');
data = textscan(identif, '%f %f %f %f %f',3,'headerlines',1);
fclose(identif);
```

Nie všetky textové súbory musia obsahovať výhradne číselné premenné. Uvedieme si príklad súboru v ktorom sa nachádzajú premenné rôzneho dátového typu. Majme súbor ***data.txt*** s nasledujúcim obsahom:

```
Vysoký      elektrikár    512.65      2         12        Áno
Chladná     lekárka       1254.35     1         10        Áno
Hravý       účtovník      845.23      2         15        Nie
Nováková    učiteľka      658.12      3         20        Áno
```

Takto usporiadané údaje je možné načítať pomocou funkcie ***textread*** takto:

```
>> [meno, profesia, mzda, deti, odpracovane_roky, odpoved]=...
textread('data.txt','%s %s %f %d %d %s', 4)
```

Údaje zo súboru sa rozdelia do jednotlivých premenných – meno, profesia, mzda.... Typy týchto premenných určujú prepínače (*%s* – reťazec znakov, *%f* – reálne číslo, *%d* – celé číslo). Posledný parameter udáva počet riadkov, ktoré sa majú spracovať.

- **Ukladanie a načítavanie dát do/z xls súborov**

⇨ **Ukladanie dát**

Funkcia ***xlswrite*** slúži na ukladanie dát vo formáte *xls*. Pri použití funkcie je potrebné zadať názov súboru, do ktorého sa budú dáta ukladať a maticu ***m x n*** dát, ktorá sa bude ukladať. Matlab má obmedzenie pre veľkosť takejto matice. Rozmer *m* nesmie prekročiť hodnotu 65536 a rozmer *n*

nesmie prekročiť hodnotu 256. Matica môže obsahovať číselne a reťazcové hodnoty. Údaje sa zapíšu do prvého hárka na pozíciu A1. Pridaním parametrov vieme ovplyvniť názov hárka, do ktorého sa údaje uložia a pozíciu buniek od ktorej po ktorú sa údaje budú zapisovať (prvá bunka označuje ľavý horný roh, druhá označuje pravý dolný roh). Obidva spomínané parametre zadávame ako reťazec.

```
M={'júl','teplota vzduchu','teplota vody';...
1,31,23;2,29,22;3,33,24;4,31,25;5,30,21;6,34,24};
xlswrite('Zosit.xls',M);
xlswrite('Zosit.xls',M,'júl');
xlswrite('Zosit.xls',M,'júl','B3:D8');
```

⇨ **Načítavanie dát**

Na načítavanie dát zo súboru xls bola vytvorená funkcia ***xlsread***. Táto funkcia je určená na čítanie číselných údajov. Funkcia vracia maticu čísel dátového typu double.

```
A=xlsread('Názov_súboru.xls');
% načítajú sa údaje zo súboru Názov_súboru
% z prvého hárku od bunky A1
B=xlsread('Názov_súboru.xls','názov_hárku');
% načítajú sa údaje zo súboru Názov_súboru
%z hárka názov_hárku  od bunky A1
C=xlsread('Názov_súboru.xls','názov_hárku','B3:D8');
% načítajú sa údaje zo súboru Názov_súboru
%z hárka názov_hárku  od bunky B3 po bunku D8
```

3.2 Príklady práce so súbormi v prostredí MATLAB

- **Načítavanie výstupných dát z programového prostredia MATLAB**

a) Export premenných z pracovného priestoru do binárneho súboru.

*T*ento úkon je možný použitím funkcie ***save.*** MATLAB ukladá premenné do špeciálneho binárneho súboru s príponou .mat.

```
save('názov súboru','premenná1 ','premenná2 ',...,'premennáN ');
save  názov_súboru premenná1  premenná2 ...premennáN;
```

Ak nie je špecifikovaná cesta, potom je súbor uložený do aktuálneho adresára.

Príklad 1
Vytvorte 3 premenné, prvá bude skalár, druhá vektor a tretia matica. Tieto premenné uložte do súboru s názvom „premenne.mat“.

Riešenie v programovom prostredí MATLAB:

```
>> x=1;
>> y=5:2:25;
>> A=[1 2 3; 4 5 6; 7 8 9];
>> save('premenne','x','y','A')
```

b) Export dát do textového súboru v ASCII formáte

S týmito súbormi je možné neskôr pracovať v akomkoľvek textovom editore.
Funkcia ***save*** ukladá vybrané premenné do textového súboru

```
save('názov súboru','premenná1 ','premenná2 ',...,'premennáN ','-ascii' );
save  názov súboru premenná1  premenná2 ...premennáN  '-ascii';
```

Príklad 2
Premenné vytvorené z príkladu 1 uložte do súboru s názvom „ascii“, vo formate ascii kódu.

Riešenie v programovom prostredí MATLAB:

```
>> save('ascii','x','y','A','-ascii')
```

Funkcia ***dlmwrite –*** ukladanie numerický dát do textového súboru, pričom je možné špecifikovať typ oddeľovača medzi jednotlivými prvkami.

```
dlmwrite ('názov súboru',X,'Oddeľovač')
```

Názov súboru je zadávaný spolu s príponou (napríklad „.txt“, „.dat“). Oddeľovačom môže byť akýkoľvek znak(obvykle medzera, tabulátor a pod.)

Príklad 3
Vektor z príkladu 1 uložte do textového súboru s názvom *pokus.txt*, pričom ako oddeľovače použite tabulátor:

Riešenie v programovom prostredí MATLAB:

```
>> dlmwrite('pokus.txt', y, '\t')
```

Funkcia ***fprintf –*** ukladanie formátovaných dát do textových súborov
Funkcia ***fopen, fclose –*** otvorenie / zatvorenie súboru

```
F = fopen(`nazov suboru`,`w`);  % otvorenie súboru funkciou
fprintf(F, format, X);          % uloženie dáta z matice
fclose(F);                      % zatvorenie súboru
```

Názov súboru je zadávaný aj s príponou napríklad „.txt“ , „.dat“.
Parameter formát je reťazec znakov, ktorý charakterizuje formátovanie dát. Reťazec musí začínať znakom **%**, ďalšie znaky určujú počet zobrazených číslic resp. desatinných miest, formát zápisu a pod.

formát zápisu	
znak	***popis***
%d	*desatinné číslo*
%e	*exponenciálny tvar*
%f	*pevný počet miest*
%s	*reťazec znakov*
špeciálne znaky	
\n	*posun na nový riadok*
\r	*znak nového riadku*
\t	*tabulátor*
%%	*percento*

Príklad 4
Otvorte súbor **pokus.txt,** ktorý sme si vytvorili v príklade 3, pre zápis, vložte do súboru reťazec „toto je skalár: x“, kde x bude predstavovať skalár vytvorený v príklade 1. Nakoniec súbor zatvorte.

Riešenie v programovom prostredí MATLAB:

```
>> f=fopen('pokus.txt','w');
>> fprintf(f,'toto je skalár: %d', x);
>> fclose(f);
```

c) Export dát do tabuliek

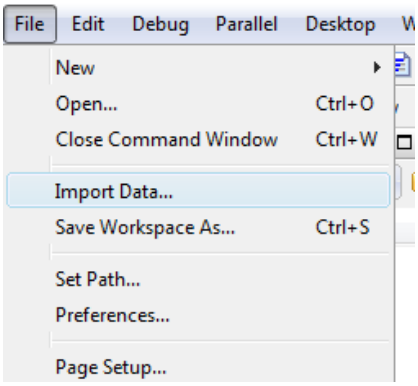
Funkcia ***xlswrite*** vykonáva export dát do tabuliek Excelu

xlswrite('nazov suboru',X,list,'pozicia');

Príkaz exportuje je dáta v matici ***X*** do tabuľky v Exceli do súboru so zadaným názvom (***názov súboru***), do zadaného ***listu*** a na zadanú ***pozíciu***. Pokiaľ súbor neexistuje, tak je najprv vytvorený.

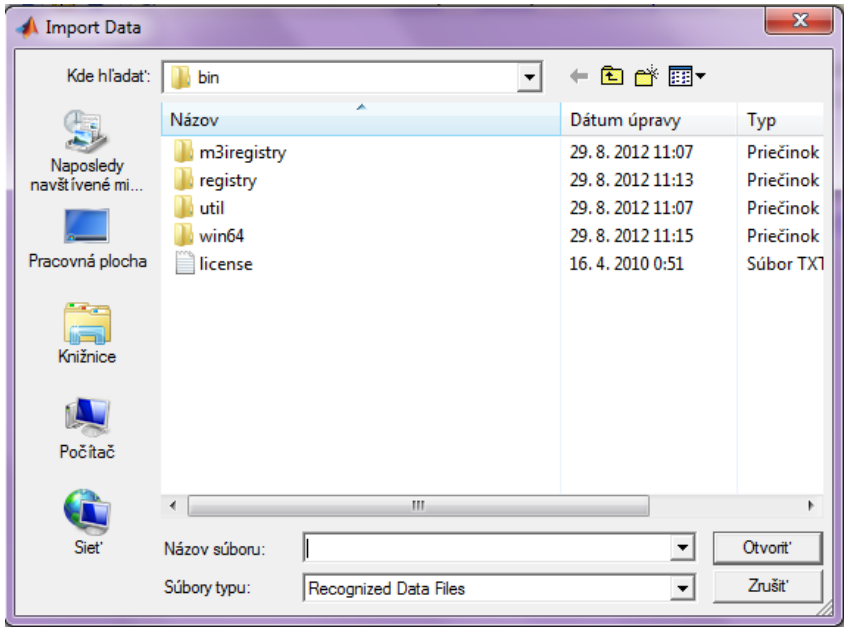
- **Načítavanie vstupných dát do programového prostredia MATLAB**

Jednou z najjednoduchších možností ako načítať vstupné dáta je použiť ***automatický nástroj pre import dát***. Tento nástroj je možné spustiť ***File → Import Data.***



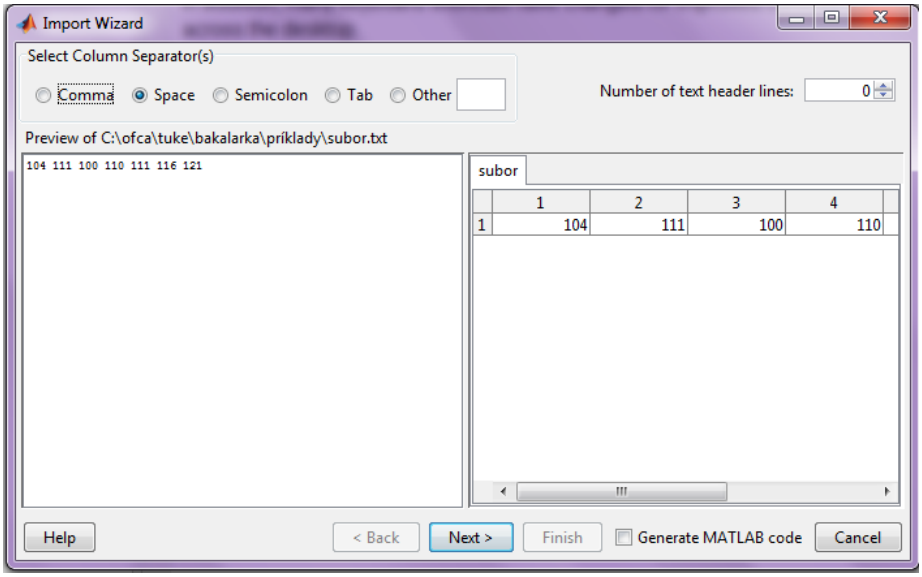
Obrázok 3-1 Spustenie nástroja pre načítanie vstupných dát do MATLABu : File -> Import Data

1) Zobrazí sa dialógové okno pre výber súboru, ktorý chceme importovať.



Obrázok 3-2 Načítanie vstupných dát do MATLABu

2) Po výbere importovaného súboru sa MATLAB automaticky pokúsi rozoznať formát súboru a dáta previesť do odpovedajúcich typov premenných.
3) V prípade textových súborov sa objaví ďalšie okno, kde je možné špecifikovať oddeľovače medzi dátami.
4)



Obrázok 3-3 Špecifikácia oddeľovačov dát

Funkcia pre načítanie dát zo súboru : ***importdata('nazov suboru')***

⇨ automatické načítavanie dát zo súboru, kde parameter ***názov súboru*** musí obsahovať aj príponu,

⇨ pokiaľ sa automaticky na základe typu prípony nepodarí rozpoznať typ dát, potom sú dáta načítane ako text s oddeľovačmi.

*a) Import dát z **mat** súborov*

$$load\,nazov\_suboru;$$

Funkcia ***load*** načíta premenné uložené v danom súbore do pracovného priestoru programového prostredia MATLAB.

b) Import dát z textových súborov

$$X = csvread(\text{´}nazov\ suboru\text{´})$$
$$X = dlmread(\text{´}nazov\ suboru\text{´}, D)$$
$$X = load(\text{´}-ascii\text{´}, \text{´}nazov\ suboru\text{´})$$

c) *Import dát z tabuľkových súborov realizuje funkcia **: xlsread***

$$[X,T] = xlsread(\text{´}nazov\ suboru\text{´}, list, \text{´}pozicia\text{´});$$

⇨ funkcia ***xlsread*** načíta dáta zo zadanej ***pozície a listu*** **zo** zadaného **súboru**, pričom numerické dáta budú načítané do **matice** *X* a textové dáta do ***poľa buniek*** *T*.

Príklad 6
Načítajte data uložené do súboru ***premenne.mat*** do programového prostredia MATLAB.
Riešenie v programovom prostredí MATLAB*:*

```
>> load premenne.mat        % pred použitím funkcie load súbor musí obsahovať data
```

3.3 Operácie s polynómami s využitím funkcií jazyka MATLAB

Polynóm je výraz (súčet alebo rozdiel jednočlenov) v tvare

$$P(x) = \sum_{i=0}^{n} a_i x^{n-i} = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_n x^0 ,$$

kde $a_n \neq 0$ a $a_0, a_1, \ldots, a_n$ sa nazývajú koeficienty polynómu

- **Základné vlastnosti polynómov**

a. dva polynómy sa rovnajú, ak sú rovnakého stupňa a majú rovnaké koeficienty pri každej mocnine
b. dva polynómy sa rovnajú, ak nadobúdajú rovnaké hodnoty pre každé x,
c. nech polynómy $P_n(x)$a $Q_m(x)$; kde $Q_m(x) = b_0 x^n + b_1 x^{n-1} + \cdots + b_{n-1} x^1 + b_n$, kde $n \geq m$ a nech $P_n(x)$a $Q_m(x)$ sú rôzneho stupňa, potom existujú dva jednoznačne určené polynómy $R(x)$ a $Z(x)$, pre ktoré platí $P_n(x) = Q_m(x) * R(x) + Z(x)$, ppolynóm $R(x)$ sa nazýva čiastočný podiel a polynóm $Z(x)$ sa nazýva zvyšok po delení. Stupeň polynómu $Z(x)$ je vždy menší ako stupeň polynómu $R(x)$

V programovom prostredí MATLAB je polynóm zadávaný v podobe vektora koeficientov počnúc koeficientom pri najvyššej mocnine.

- **Operácie s polynómami**

Pri násobení polynómov P, Q platí pravidlo vynásobenia každého s každým prvkom polynómu

$$P_n(x) * Q_m(x) =$$
$$= (a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x^1 + a_n) * (b_0 x^m + b_1 x^{m-1} + \cdots + b_{m-1} x^1 + b_m) =$$
$$= a_0 x^n * b_0 x^m + a_0 x^n * b_1 x^{m-1} + \cdots + a_0 x^n * b_m + a_1 x^{n-1} * b_0 x^m + \cdots$$

Funkcia ***conv*** slúži na vynásobenie dvoch polynómov **P a Q** (zadaných ako vektory):

conv(P,Q)

Príklad 1
Vynásobte dva polynómy, ktoré su dané nasledovne::

$$P(x) = x^3 + 4x^2 + 3,\ Q(x) = 8x^2 - 2$$

Numerický výpočet:

$$P(x) * Q(x) = (x^3 + 4x^2 + 3) * (8x^2 - 2)$$
$$P(x) * Q(x) = 8x^5 + 31x^4 - 4x^3 + 24x^2 - 3x$$

Riešenie v programovom prostredí MATLAB :

```
>> P=[1 4 0 3];
>> Q=[8 -1 0];
>> vysledok = conv(P,Q)

vysledok =

    8   31   -4   24   -3    0
```

Funkcia ***deconv*** slúži na delenie polynómu iným polynómom

[*vysledok, zvysok] = deconv(P,Q)*

pričom polynómy ***P,Q*** musia byť dopredu inicializované. Výstupom sú polynómy vo forme vektorov prísl

Príklad 2
Vydeľte polynómy ***P,Q*** a overte výsledok výpočtu v programovom prostredí MATLAB.

Numerický výpočet :

$(x^4 - 3x^3 + 6x^2 - 3x^1 + 5x^0) : (x^2 - 3x^1 + 5x^0) = x^2 + x^0$

$-(x^4 - 3x^3 + 5x^2)$

$0x^4 + 0x^3 + x^2 - 3x^1 + 5x^0$

$-(x^2 - 3x^1 + 5x^0)$

Riešenie v programovom prostredí MATLAB:

```
>> P=[1 -3 6 -3 5];
>> Q=[1 -3 5];
>> [vysledok,zvysok] = deconv(P,Q)

vysledok =

   1   0   1


zvysok =

   0   0   0   0   0
```

Príklad 3
Numericky vypočítajte a následne overte v programovom prostredí MATLAB **delenie dvoch polynómov *P,Q*** so zvyškom.

Numerický výpočet :

$$(x^3 - 2x^2 + x - 1) : (x^2 - 3x + 2) = x + 1 + \frac{2x - 3}{x^2 - 3x + 2}$$

$-(x^3 - 3x^2 + 2x)$

$x^2 - x - 1$

$-(x^2 - 3x + 2)$

$2x - 3$

Riešenie v programovom prostredí MATLAB:

```
>> P=[1 -2 1 -1];
>> Q=[1 -3 2];
>> [vysledok, zvysok] = deconv(P,Q)

vysledok =

    1    1


zvysok =

    0    0    2   -3
```

Funkcia ***roots*** sa používa na výpočet koreňov polynómu,

roots(polynom)

Príklad 4
Majme zadaný polynóm $P(x) = x^3 + 4x^2 + x - 6$, nájdite jeho korene pomocou funkcie ***roots***.
Po inicializácii polynómu P= [1 4 1 -6] ako vektora, môžeme volať funkciu ***roots*** na výpočet koreňov

```
>> koren = roots(P)

koren =

   -3.0000
   -2.0000
    1.0000
```

Funkcia ***poly*** vytvára polynóm z zadaných koreňov (vypočíta koeficienty polynómu v klesajúcich mocninách ***x)***

polynom = poly(A)

pričom ***A*** je stĺpcový vektor obsahujúci korene polynómu

Príklad 5
Nájdite koeficienty polynómu ktorého korene sú: $x_1 = -3$, $x_2 = 1$, $x_3 = -2$

Numericky postupujeme tak, že vykonáme súčin koreňových činiteľov:

$$A(x) = (x+3) * (x-1) * (x+2) = x^3 + 4x^2 + x - 6$$

Riešenie v programovom prostredí MATLAB:

```
>> A=[-3;-2;1];
>> polynom = poly(A)

polynom =

    1    4    1   -6
```

Funkcia **polyval -** vypočíta hodnotu polynómu daného stupňa pre dané *x*

polyval(polynom, x);

pričom prvky vektora *polynom* musia byť známe a tiež musíme špecifikovať hodnotu *x*.

Príklad 6
Vypočítajte a následne overte v jazyku MATLAB hodnotu zadaného polynómu ***P(x)*** pre *x* = 2

$$P(x) = x^3 + 4x^2 + x - 6\,, \quad pre \quad x=2$$

$$P(x=2) = 2^3 + 4*2^2 + 2 - 6 = 20$$

Riešenie v programovom prostredí MATLAB:

```
>> P=[1 4 1 -6];
>> vysledok = polyval(P,2)

vysledok =

   20
```

- Funkcia ***polyder*** **-** derivácia polynómu
  a. ***polyder(****P****)*** derivácia polynómu ***P***
  b. ***polyder(****A,B****)*** alebo ***polyder(conv(****A,B****))*** derivácia súčinu polynómov ***A*B***
  c. ***polyder(****B,A****)*** derivácia podielu polynómov ***B/A***

Príklad 7
Vypočítajte deriváciu zadefinovaného polynómu ***P(x)*** . Výsledok overte v jazyku MATLAB s využitím funkcie ***polyder.***

$$P(x) = 9x^5 - 6x^3 + x^2 - 18$$

$$P(x)' = (9x^5 - 6x^3 + x^2 - 18)' = 45x^4 - 18x^2 + 2x$$

Riešenie v programovom prostredí MATLAB:

```
>> P=[9  0 -6 1 0 -18];
>> vysledok = polyder(P)

vysledok =

   45    0  -18    2    0
```

Príklad 8
Vypočítajte deriváciu súčinu polynómov ***A(x)*** a ***B(x)*** pričom sú zadefinované nasledovne :

$$A(x) = x^3 + 4x^2 + 3$$
$$B(x) = 8x^2 - 2$$

Numericky vypočítame súčin polynómov:

$$A(x) * B(x) = (x^3 + 4x^2 + 3) * (8x^2 - 2) = 8x^5 + 31x^4 - 4x^3 + 24x^2 - 3x$$

Výsledok súčinu polynómov ***A(x), B(x)*** následne zderivujeme:

$$\big(A(x) * B(x)\big)' = (8x^5 + 31x^4 - 4x^3 + 24x^2 - 3x)' =$$
$$= 40x^4 + 124x^3 - 12x^2 + 48x - 3$$

Riešenie v programovom prostredí MATLAB:

```
>> A = [1 4 0 3];
>> B = [8 0 -2];
>> vysledok = polyder(A,B);

vysledok =

  40   124   -12   48   -3
```

Funkcia ***residue*** – umožňuje rozklad racionálne lomenej funkcie na parciálne zlomky, pričom parametre ***b,a*** majú význam vektorov v ktorých sú uložené koeficienty polynómov čitateľa a menovateľa racionálne lomenej funkcie v klesajúcich mocninách.

[R,P,K] = residue(b,a)

kde :

R – zosilnenia výsledných parciálnych zlomkov
P – stĺpcový vektor koreňov
K – zosilnenie

Príklad 9

Rozložte funkciu $G(x) = \frac{-x+10}{x^2-4}$ na parciálne zlomky, riešte samostatne numericky a výsledok overte v jazyku MATLAB.

$$G(x) = \frac{b(x)}{a(x)} = \frac{-x+10}{(x-2)(x+2)} = \frac{r_1}{x-p_1} + \frac{r_2}{x+p_2} = \frac{r_1}{x-2} + \frac{r_2}{x+2}$$

Riešenie v programovom prostredí MATLAB:

```
>> b=[-1 10];
>> a=[1 0 -4];
>> [R,P,K]=residue(b,a)

R =

    2
   -3


P =

   2.0000
  -2.0000


K =

     []
```

3.4 Príklady na riešenie

1. Naplňte premenné *prem1, prem2, prem3, prem4, prem5, a, b, c, d* číselnými hodnotami. Uložte vytvorené premenné *prem1, prem2, prem3, prem4, prem5, b, d* do súboru *uloha1.mat*. Pri zadávaní premenných využite žolíka.
2. Vynulujte Workspace. Zo súboru *úloha1.mat* načítajte všetky premenné.
3. Vynulujte Workspace. Zo súboru *úloha1.mat* načítajte nasledovné premenné – a,c,d,prem1.
4. Vytvorte maticu 4x2. Uložte ju do textového súboru , v ktorom ako oddeľovač bude slúžiť čiarka.
5. Vynulujte Workspace. Do premennej A načítajte obsah súboru, ktorý ste vytvorili v predošlej úlohe.
6. Vytvorte si textový súbor, ktorý bude obsahovať „tabuľku“ údajov o spolužiakoch, a to : meno, evidenčné číslo, počet zapísaných predmetov v minulom semestri, počet úspešne zvládnutých predmetov v minulom semestri, odpoveď na otázku, či sú spokojný so svojím výberom vysokej školy. Načítajte tieto údaje do Workspace.
7. Vytvorte súbor v programe Microsoft Excel, do ktorého vložíte informácie o meniacom sa kurze americký dolár vzhľadom na eura v čase od 1.2.2013-15.2.2013 (1.stĺpec – poradové číslo dňa v mesiaci február, 2.stĺpec – predaj, 3.stĺpec – nákup). Načítajte tieto údaje do Workspace.
8. Pomocou ľubovoľného príkazu na zobrazovanie grafov zobrazte funkciu $z=\sqrt{(2x^2-y^3)}$, pričom x=3:20 a y=-3:-20. Graf popíšte.
9. POTREBNÉ PRE ĎALŠIE PRÍKLADY
   A) Vytvorte súbor *data1*, kde uložíte premenné:
   x =[-5, -4.5, -3, -2, 0.5, 2.5, 6, 10, 12.5, 16, 20.5, 22]
   y =[0.257, 0.968, 1.254, 1.685, 2.002, 2.685, 2.986, 3.574, 3.867, 4.012, 3.986, 4.876].
   B) Vytvorte súbor *data2*, kde uložíte premenné:
   x=[-4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 21]
   y=[-0.112, -0.058, 0.218, 0.294, 0.351, 0.486, 0.687, 0.788, 1.007, 1.315, 1.419, 1.225, 1.254, 1.321, 1.457]
10. Aproximujte údaje zo súboru *data1* polynómami 2., 3. a 4. stupňa. Vykreslite do grafu príslušné priebehy. Vypočítajte sumu štvorcov odchyliek pre každú aproximáciu.
11. Aproximujte údaje zo súboru *data1* polynómom *n*- tého stupňa, pričom *n* sa zadá z klávesnice. Aproximované údaje uložte do súboru. Vykreslite do grafu priebeh aproximácie.
12. Interpolujte namerané dáta zo súboru *data2*. Použite tretinový krok vzorkovania a na interpolovanie použite metódu ‘**cubic‘** a **‘linear‘**. Výsledky zobrazte v dvoch grafoch ktoré budú vedľa seba.
13. Načítajte dáta z súboru *data1*. Tieto hodnoty interpolujte zo štvrtinovým krokom vzorkovania a použite na to funkciu interp. Následne interpolujte dáta pomocou metódy ‘spline‘ a zobrazte výsledok v grafe.

3.5 Tvorba grafických objektov v jazyku MATLAB s využitím základných funkcií

Grafické funkcie v programovom prostredí MATLAB automaticky otvoria nové grafické okno, v prípade ak už je nejaké grafické okno otvorené, prekresľujú pôvodné grafické zobrazenie.

- Funkcia ***figure*** sa používa na otvorenie nového okna grafického obrázku. Ak za funkciu ***figure*** zapíšeme do zátvoriek číslo - vytvorí sa grafické okno s číslom, ktoré sme zadali,

  figure;
  figure(x);

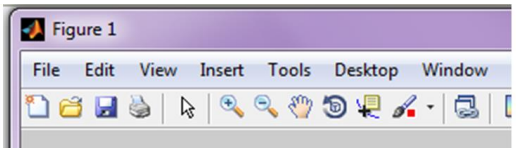
  Syntax

  x = figure;

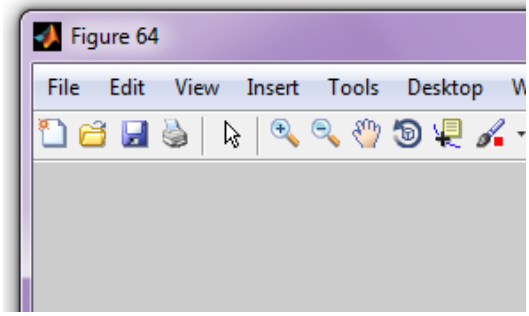
  umožní užívateľovi po otvorení okna uložiť jeho číslo

Funkcionalita funkcie ***figure*** v prostredí MATLAB je znázornená na grafických oknách

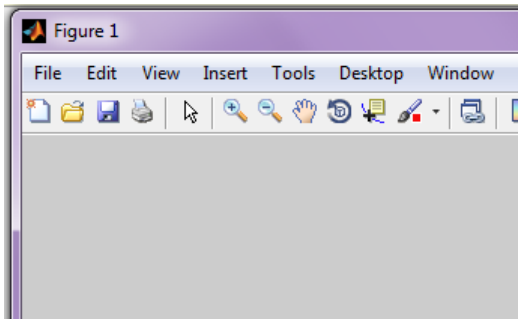
```
>> figure
```




```
>> figure(64)
```




```
>> x =figure

x =

     1
```



- Funkcia ***close*** sa používa uzatváranie grafických okien. Ak použijeme funkciu close bez parametrov, uzatvára sa momentálne aktívne okno. Ak zatvoríme okno, aktuálnym sa stane okno vytvorené pred uzatvorením okna. Ak otvoríme nové okno, považujeme ho automaticky za aktívne.

  close(x), close(´nazov´),close all

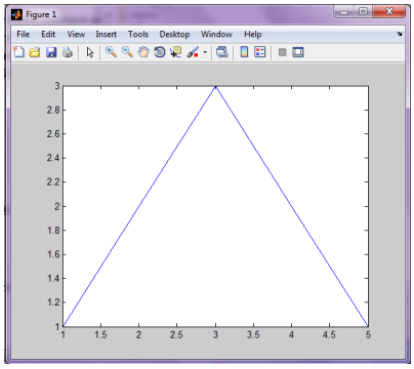
  pričom :

x - číslo aktualného grafického okna v ktorom pracujeme,
nazov - názov aktuálneho grafického okna v ktorom pracujeme,
all – funkcia ***close all*** uzatvára všetky grafické okna v ktorých sme pracovali.

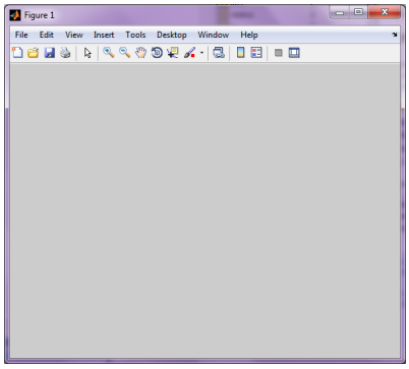
- Funkcia ***gcf*** (*Get Handle to Current Figure*) sa používa na zistenie aktuálneho okna (ak aktuálne nie je otvorené žiadne grafické okno, funkcia otvorí grafické okno)

```
>> x=gcf

x =

     1
```

- Funkcia ***clf*** **(Clear Current Figure)** sa používa na zmazanie aktuálneho obrázku z aktuálneho grafického okna.



```
>> clf
```



- Funkcia ***plot*** – vykreslí graf, ak hodnoty zavislej a nezavislej premennej boli vopred zadané vo vektoroch. Funkcia otvorí nové grafické okno, za predpokladu, že doposiaľ nebolo otvorené a ak bolo, pracuje s aktívnym oknom a vykreslí graf.

  plot(x,y) %funkcia ***plot*** vykreslí funkčnú závislosť *y = f(x)*

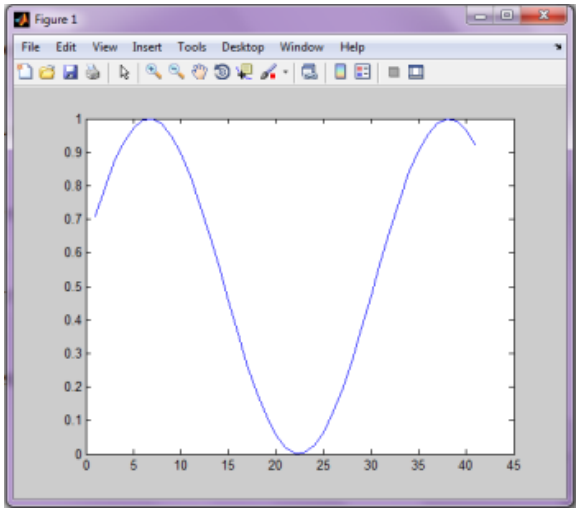
  pričom vektory hodnôt ***x,y*** musia byť dopredu zadefinované

Príklad 1

Je daná funkcia $y = (sin(x))^2$ Znázornite graficky danú funkciu pre hodnoty vektora ***x*** z intervalu : $x \in \langle 1,5 \rangle$ s krokom 0,1

Riešenie v jazyku MATLAB :

```
>> x=[1:0.1:5];
>> y=sin(x).^2;
>> plot(x,y)
```



Funkcia ***plot*** volaná nasledovne

plot (x,y, ´vlastnosti´)

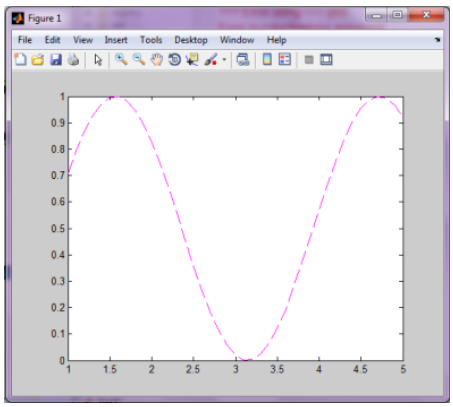
vykreslí hodnoty ***y*** v závislosti na hodnotách ***x*** a s atribútmi , ktoré sú nastavené textovým reťazcom uvedeným v apostrofoch. Môžeme použiť nasledujúce atribúty:

farby		**body**		**čiary**	
r	červená	**.**	bod	**-**	plná (default)
y	žltá	**o**	kruh	**:**	bodkovaná
k	čierna	**x**	krížik	**-.**	bodkočiarkovaná
b	modrá	**+**	plus	**--**	čiarkovaná
g	zelená	*****	hviezda	**none**	bez čiary
c	azúrová	**s**	štvorček		
m	purpurová	**d**	diamant	**LineWidth**	šírka čiary
w	biela	**v**	trojuholník (dolu)	**Color**	farba popísaná v RGB
		^	trojuholník (hore)	**MarkerEdgeColor**	farba obrysu ukazovateľa
		<	trojuholník (vľavo)	**MarkerFaceColor**	farba vypne ukazovateľa
		>	trojuholník (vpravo)	**MarkerSize**	veľkosť ukazovateľa
		p	pentagram		
		h	hexagram		

Príklad 2

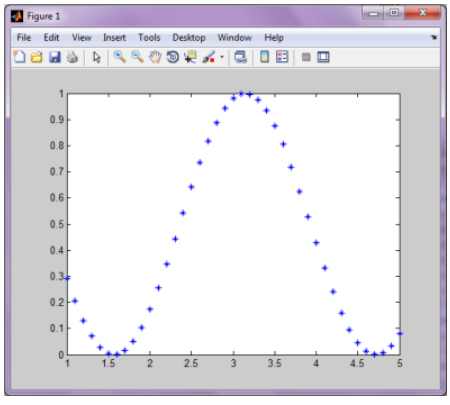
Vykreslite funkciu $y = sin(x^2)$ ***y = sin(x²)*** na intervale x ∈ <1,5> s krokom 0,1 ,čiarkovanou purpurovou čiarou

```
>> x=[1:0.1:5];
>> y=sin(x).^2;
>> plot(x,y,'m--')
```



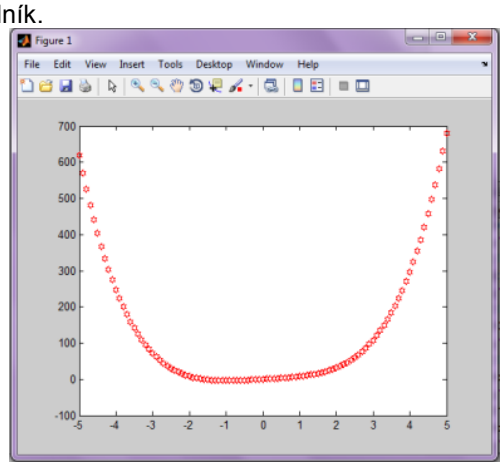
- Vykreslite funkciu ***y = (cos(x))²*** na intervale x ∈ <1,5> s krokom 0,1 , pričom pixle sú označené modrými hviezdičkami

```
>> x=[1:0.1:5];
>> y=cos(x).^2;
>> plot(x,y,'b*')
```



- Vykreslite funkčnú závislosť $y = x^4 + x^2 + 6 * x$ ***y = x⁴ +x² +6.x*** na intervale $x \in$ <-5,5> s krokom 0,1 , použite červený šesťuholník.

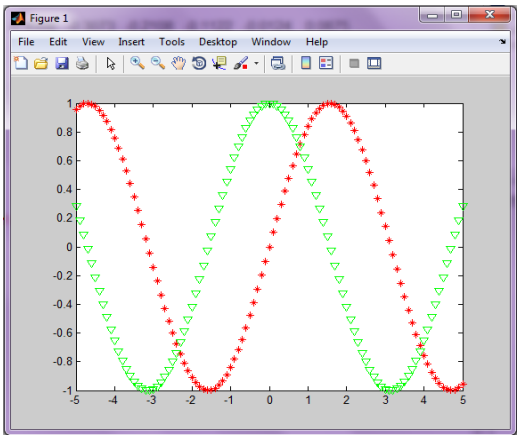
```
>> x=[-5:0.1:5];
>> y=x.^4+x.^2+6*x;
>> plot(x,y,'rh')
```



Pozn. Viacparametrovú funkciu ***plot(x,y₁,x,y₂,....)*** používame na vykreslenie viacerých funkčných závislostí vzhľadom na jediný vektor ***x*** nezávislej premennej.

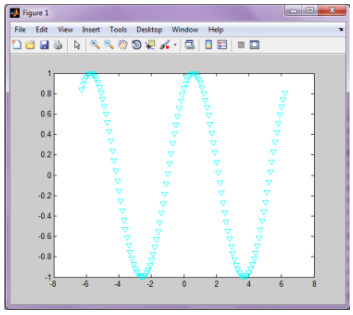
Príklad 3
Vykresli funkcie $y_1 = sin(x)$ a $y_2 = cos(x)$ na intervale <-5,5> s krokom 0,1 , pričom y_1 je vykreslená červenými hviezdičkami a y_2 je vykreslené zelenými trojuholníkmi.

```
>> x= [-5:0.1:5];
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,'r*',x,y2,'gv')
```

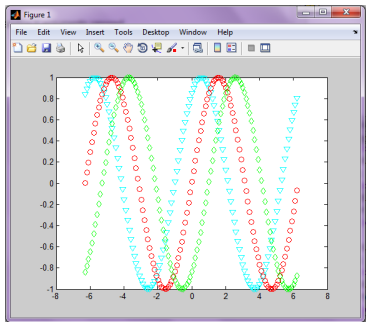


- funkcia ***hold –*** umožňuje ponechať otvorené aktuálne okno grafického zobrazenia a zakresľovať ďalšie grafické zobrazenia (***hold on/off***)

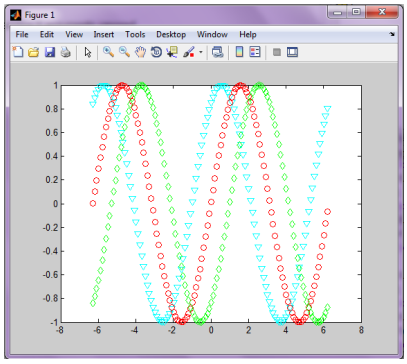
```
>> x=[-2*pi:0.1:2*pi];
>> y=sin(x+1);
>> plot(x,y,'cv')
```



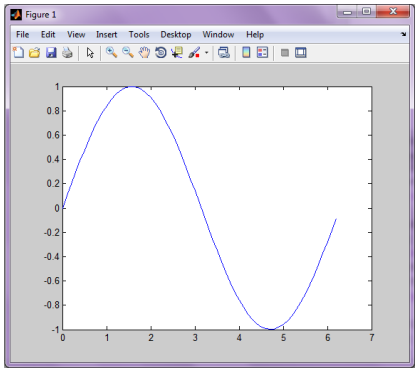

```
>> hold on
>> x=[-2*pi:0.1:2*pi];
>> y=sin(x);
>> plot(x,y,'ro')
>> x=[-2*pi:0.1:2*pi];
>> y=sin(x-1);
>> plot(x,y,'gd')
```



Príkaz ***hold off*** prepína späť do implicitného stavu, v ktorom nové príkazy grafiky vždy prepíšu starý graf a pred kreslením resetujú všetky vlastnosti objektu.



```
>> hold off
>> x=[0:0.1:2*pi];
>> y=sin(x);
>> plot(x,y)
```



- Funkcia ***subplot*** – umožňuje rozdelenie grafického okna na podokná. Syntax v tvare

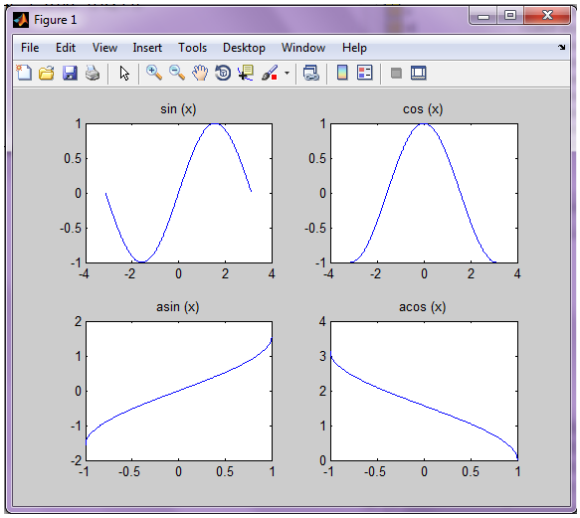
  subplot(m,n,p)

  vytvorí grafické okno v podobe matice $m \times n$ ***m x n*** a aktivuje práve jedno podokno ***p***, takže ju musíme volať pre každé podokno samostatne**,**

Príklad 4
Použitím funkcie ***subplot*** vytvorte grafické okno v ktorom budú 4 podokná a v nich budú zobrazené funkcie ***sin(x)*** a ***cos(x)*** pre $x \in \langle -\pi, \pi \rangle$ a ***acrsin(x)*** a ***arccos(x)*** v intervale $x \in \langle -1,1 \rangle$.

Riešenie v jazyku MATLAB:

```
>> x1=-pi:0.01:pi;
>> x2=-1:0.01:1;
>> y1=sin(x1);
>> y2=cos(x1);
>> y3=asin(x2);
>> y4=acos(x2);
>> subplot(2,2,1); plot(x1,y1); title('sin (x)')
>> subplot(2,2,2); plot(x1,y2); title('cos (x)')
>> subplot(2,2,3); plot(x2,y3); title('asin (x)')
>> subplot(2,2,4); plot(x2,y4); title('acos (x)')
```



- Funkcia ***title*** – umožňuje pomenovať graf (vypíše názov grafu daný textovým reťazcom na stred horného okraja grafu)

  `title('nazov')` ***title(´nazov´)***

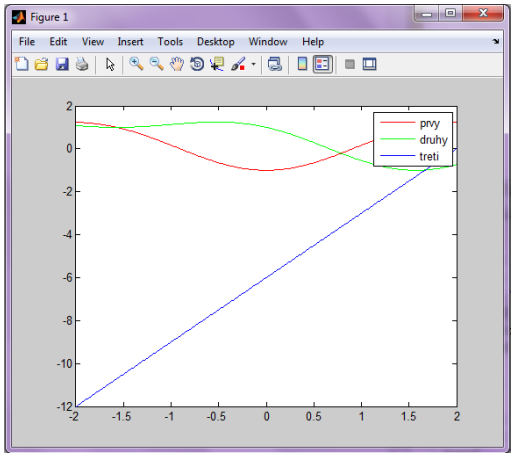
- Funkcia ***label*** – umožňuje popísať osi grafu

`xlabel('popisosix')`	***xlabel(´popisosix´)***	%pomenuje x-ovú os
`ylabel('popisosiy')`	***ylabel(´popisosiy´)***	%pomenuje y-ovú os

- Funkcia ***legend*** – umožňuje vytvorenie popisu jednotlivých objektov(grafických závislostí) – priradí k jednotlivým grafickým závislostiam ich popis, farby a prispôsobí ich poradie v akom boli zakreslené

  `legend('nazov1','nazov2','nazov3',...)` ***legend(´nazov1´, ´nazov2´, ´nazov3´, ...)***

```
>> x=[-2:0.01:2];
>> y1=sin(x).^2-cos(x);
>> y2=cos(x).^2-sin(x);
>> y3=3*x-6;
>> plot(x,y1,'r')
>> hold on
>> plot(x,y2,'g')
>> plot(x,y3,'b')
>> legend('prvy', 'druhy', 'treti')
```



- Funkcia ***text*** – umožňuje priradenie textu do aktuálneho grafického okna vytvorením objektu text, čím vloží na súradnice *(x,y)* resp. *(x,y,z)* zadaný *reťazec*

  text(x,y,´retazec´), text(x,y,y, ´retazec´),

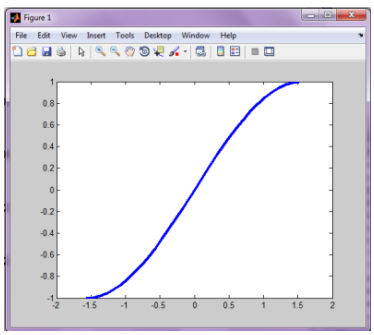
- **Funkcia *axis* -** umožňuje zmenu rozsahu a vzhľadu osi. Syntax

  axis([xmin, xmax, ymin, ymax])

  nastaví osi *x,y* podľa nami zadaných minimálnych a maximálnych hodnôt a zápis

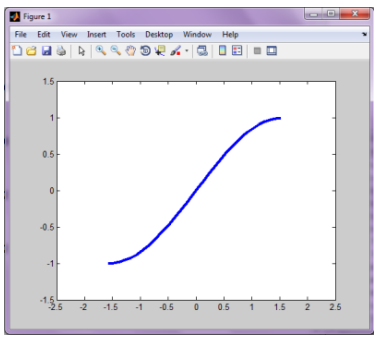
  axis(´auto´)

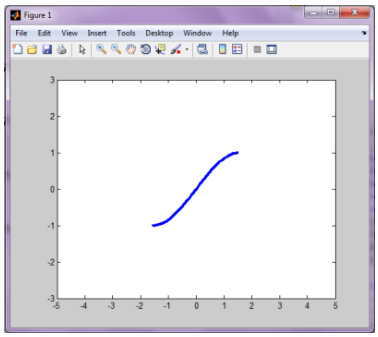
  umožní automatické nastavovanie osi.



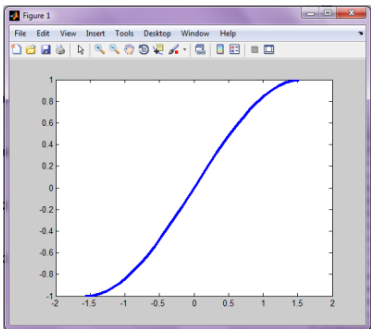

```
>> axis([-2.5, 2.5, -1.5, 1.5])
```

```
>> x=[-pi/2:0.1:pi/2];
>> y=sin(x);
>> plot(x,y,'LineWidth',3)
```





axis('auto')



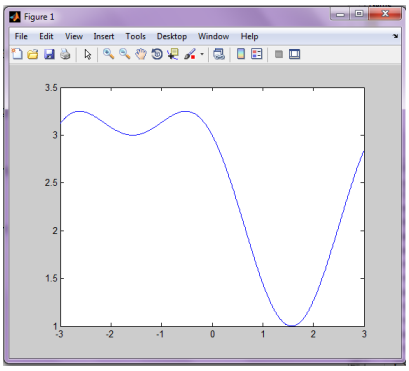
- Funkcia ***grid –*** sa používa na vytvorenie mriežky v grafickom zobrazení, pričom

  grid ongrid on

  zapína kreslenie mriežky do grafického okna a

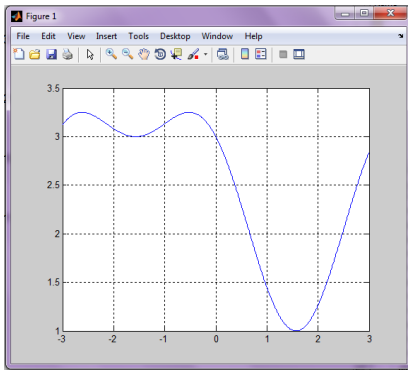
  grid offgrid off

  vypína kreslenie mriežky v grafickom okne.



>> grid on

>> grid off



Zápis ***grid*** slúži ako prepínač na zakreslenie mriežky, ak nebola zakreslená a naopak

3.6 Riešenie príkladov na regresnú analýzu v jazyku MATLAB

Budeme skúmať funkčný vzťah (priebeh závislosti), $y_i = f(x_i)$ podľa ktorého sa mení závislá premenná y pri zmenách nezávislých veličín $x_1, x_2, ..., x_n$. Cieľom aproximácie je nájsť vhodnú regresnú funkciu – funkčný predpis. Z nameraných hodnôt x_i / y_i
Regresná analýza pomáha :

- ⇨ riadiť a predpovedať chovanie sledovaných premenných
- ⇨ predpovedať hodnoty výstupných premenných aj tam, kde na výpočet nebolo dostatočné množstvo dát
- ⇨ zistiť body, ktoré sa výrazne odlišujú od očakávaného výsledku

Metóda najmenších štvorcov

Metóda najmenších štvorcov spočíva v nájdení funkcie $y = f(x)$, ktorá čo najlepšie aproximuje namerané hodnoty.

- **Aproximácia nameraných dát priamkou s využitím metódy najmenších štvorcov**

Majme súbor, ktorý obsahuje n nameraných hodnôt v tvare usporiadaných dvojíc $[x_i, y_i]$. Hľadáme lineárnu závislosť (priamku – polynóm 1.stupňa) v tvare

$$y(x) = a_1 x + a_0$$

s takými parametrami priamky a_0 a a_1, aby súčet druhých mocnín (štvorcov) vzdialeností jednotlivých bodov od tejto priamky bol čo najmenší. Túto podmienku je možné zapísať vzťahom:

$$\sum_{i=1}^{n} (a_1 x_i + a_0 - y_i)^2 \rightarrow min$$

Pomocou metódy najmenších štvorcov dostaneme priamku využitím nasledujúcich rovníc pre aproximáciu:

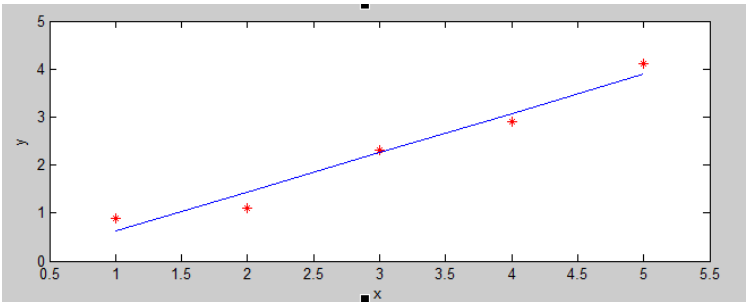
$$a_0(n+1) + a_1 \sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i$$

$$a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} x_i y_i$$

Z tejto podmienky vieme vypočítať parametre priamky a_0 , a_1 nasledovne :

$$a_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i x_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i}$$

$$a_0 = \left(\sum_{i=1}^{n} y_i - a_1 \sum_{i=1}^{n} x_i \right) / n$$



Obrázok 3-4 Názorna ukážka aproximácie nameraných údajov x_i, y_i priamkou

Korelačný koeficient určuje do akej miery lineárny vzťah $y = a_1 x + a_0$ aproximuje namerané hodnoty y_i, x_i. Korelačný koeficient je vždy číslo z intervalu <-1,1> a vypočítame ho podľa vzorca :

$$r_{x,y} = \frac{s_{xy}}{s_x s_y} = \frac{\frac{1}{n}\sum(x_i - \bar{x}).(y_i - \bar{y})}{s_x s_y}.$$

- **Aproximácia nameraných dát polynómom *n* - tého rádu v prostredí MATLAB**

Niektoré závislosti *x* a *y* nie je vhodné aproximovať priamkou, ale je potrebné ich aproximovať polynómom vyššieho rádu. Napr. pri aproximácii funkcie polynómom druhého stupňa aplikujeme metódu najmenších štvorcov na základe vzťahov

$$a_0(n+1) + a_1 \sum_{i=0}^{n} x_i + a_2 \sum_{i=0}^{n} x_i^2 = \sum_{i=0}^{n} y_i,$$

$$a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 + a_2 \sum_{i=0}^{n} x_i^3 = \sum_{i=0}^{n} x_i y_i,$$

$$a_0 \sum_{i=0}^{n} x_i^2 + a_1 \sum_{i=0}^{n} x_i^3 + a_2 \sum_{i=0}^{n} x_i^4 = \sum_{i=0}^{n} x_i^2 y_i.$$

Vo všeobecnosti je v MATLABe je možné zadané (namerané) údaje aproximovať polynómom n-tého stupňa s využitím metódy najmenších štvorcov.

- Funkcia ***polyfit*** realizuje výpočet koeficientov aproximujúceho polynómu *n*-tého stupňa metódou najmenších štvorcov, ktoré zabezpečia funkčné hodnoty polynómu čo najbližšie k funkčným hodnotám aproximovanej funkcie.

P = polyfit(x,y,n)

kde:

x – vektor hodnôt nezávislej premennej
y – vektor hodnôt závislej premennej
n – stupeň polynómu (závisí na fyzikálnej podstate problému, napr. ak predpokladáme aproximáciu priamkou, tak n = 1, ak parabolou, tak n = 2)
P – obsahuje vypočítané koeficienty výsledného polynómu

$$P(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_1 x^1 + p_0,$$

- Funkcia ***polyval*** slúži na výpočet hodnôt aproximačného polynómu v zvolených bodoch:

y_aproximovane = polyval(P,x)

kde:

x je vektor hodnôt nezávislej premennej
P je vektor koeficientov aproximovaného polynómu
y_aproximovane je vektor hodnôt aproximovaného polynómu

Chyby aproximácie

Chybu aproximácie vieme vypočítať pomocou funkcie prostredia MATLAB ***sum.***

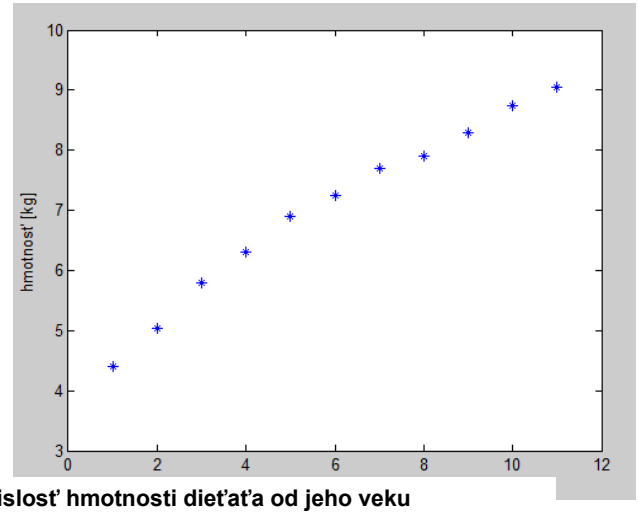
$$S = sum((y\_aproximované - y).\hat{}2)$$

pričom p_1 je približná hodnota funkcie v bode x.

Príklad 1

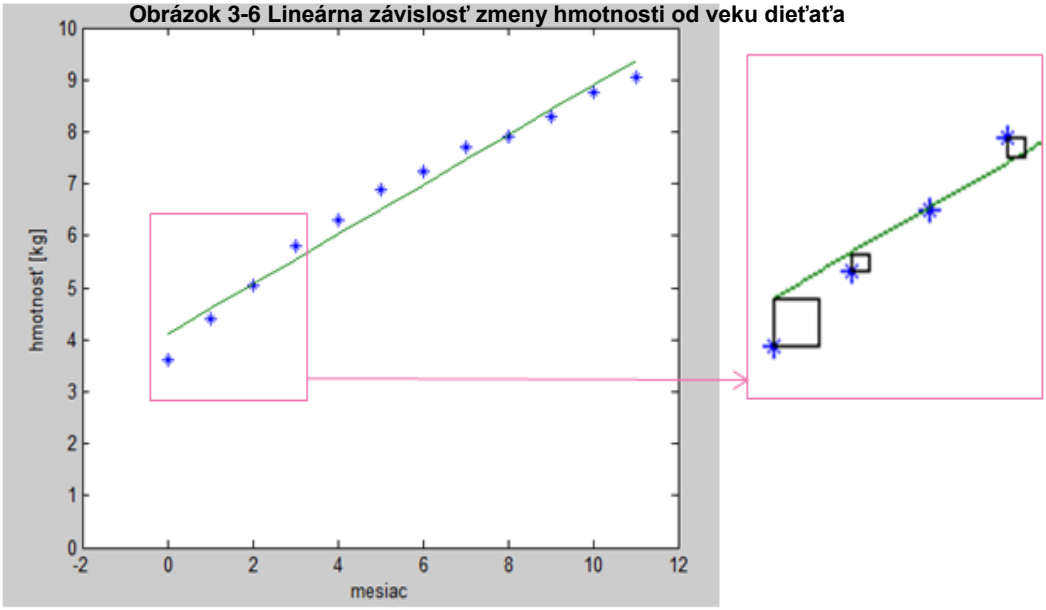
V tabuľke je uvedená meniaca sa hmotnosť dieťaťa vzhľadom na vek dieťaťa v prvých 12-tich mesiacoch. Úlohou je zistiť, aká je závislosť medzi hmotnosťou [kilogram] dieťaťa a jeho vekom [mesiac]. Tabuľkové hodnoty vykreslíme v grafe.

Mesiac (x)	Hmotnosť dieťaťa (y)
0.	3,6
1.	4,4
2.	5,03
3.	5,8
4.	6,3
5.	6,9
6.	7,25
7.	7,7
8.	7,9
9.	8,3
10.	8,75
11.	9,05



Obrázok 3-5 Závislosť hmotnosti dieťaťa od jeho veku

Obrázok 3-6 Lineárna závislosť zmeny hmotnosti od veku dieťaťa



Riešenie

$$12a_0 + 67a_1 = 80{,}98$$
$$67a_0 + 580{,}1784a_1 = 513{,}91$$

Riešením systému rovníc dostávame hľadané hodnoty koeficientov a_0 = 4,11295 , a_1 =0,4791 a výsledná regresná priamka je teda v tvare:

$$y = 0{,}4791x + 4{,}11295$$

Overíme vypočítané parametre priamky a_0, a_1 s využitím príkazu

```
polyfit(x,y,1)
```

```
Command Window
>> x=0:11;
>> y=[3.6 4.4 5.03 5.8 6.3 6.9 7.25 7.7 7.9 8.3 8.75 9.05];
>> f=polyfit(x,y,1)

f =

    0.4792    4.1129
```

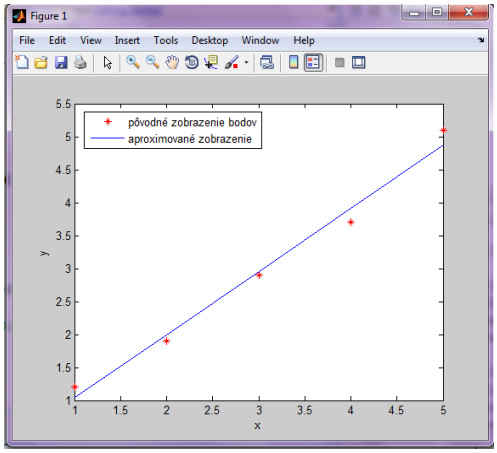
Príklad 2

Napíšte program v simulačnom jazyku MATLAB pre aproximáciu nameraných hodnôt x_i, y_i , priamkou, pričom n = 5, namerané hodnoty x_i, y_i sú uvedené v tabuľke:

x_i	1	2	3	4	5
$y_i = f(x_i)$	1.2	1.9	2.9	3.7	5.1

Riešenie v jazyku MATLAB:

```
>> x=[1:5];
>> y=[1.2,1.9,2.9,3.7,5.1];
>> P=polyfit(x,y,1)

P =

   0.9600   0.0800

>> y_aprox = polyval(P,x)

y_aprox =

   1.0400   2.0000   2.9600   3.9200   4.8800

>> S=sum((y_aprox-y).^2) %výpočet sumy štvorcov odchyliek

S =

   0.1360
```

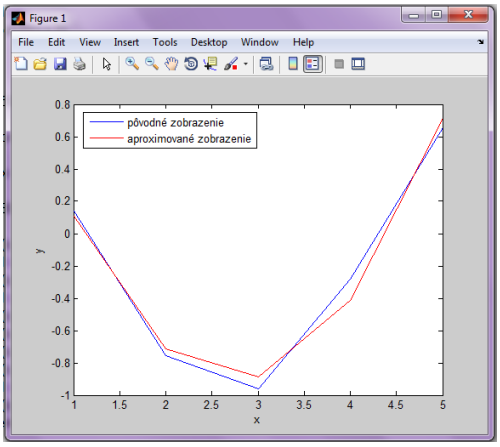



```
>> plot(x,y,'r*')
>> hold on
>> plot(x,y_aprox)
>> legend('pôvodné zobrazenie bodov', 'aproximované zobrazenie')
```

Príklad 3

Napíšte program pre aproximáciu bodov x_i, y_i polynómom 2. rádu pričom $x_i = 1,2,\ldots,5$ a hodnoty y_i vypočítate podľa predpisu $y_i = sin(x_{i+2} + 2)$. Zistite chybu aproximácie

```
>> x=[1:5];
>> y=sin(x+2);
>> P=polyfit(x,y,2)

P =

   0.3250  -1.7992   1.5830

>> y_aprox = polyval(P,x)

y_aprox =

   0.1088  -0.7153  -0.8894  -0.4135   0.7125

>> S=sum((y_aprox-y).^2) %výpočet sumy štvorcov odchyliek

S =

   0.0286
```

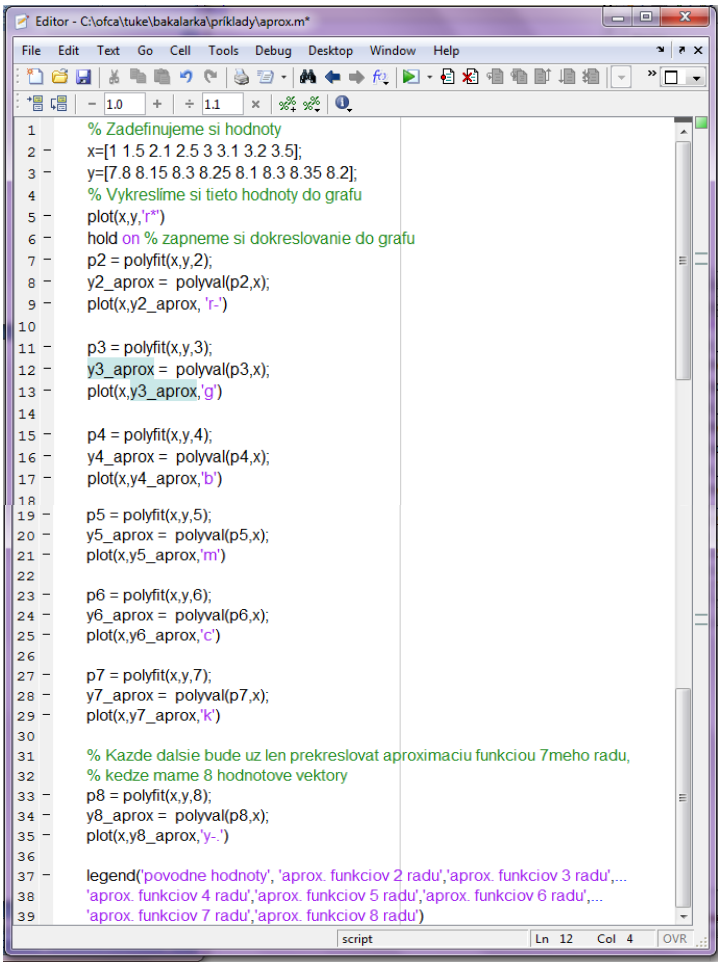



```
>> plot(x,y)
>> hold on
>> plot(x,y_aprox,'r')
>> legend('pôvodné zobrazenie', 'aproximované zobrazenie')
```

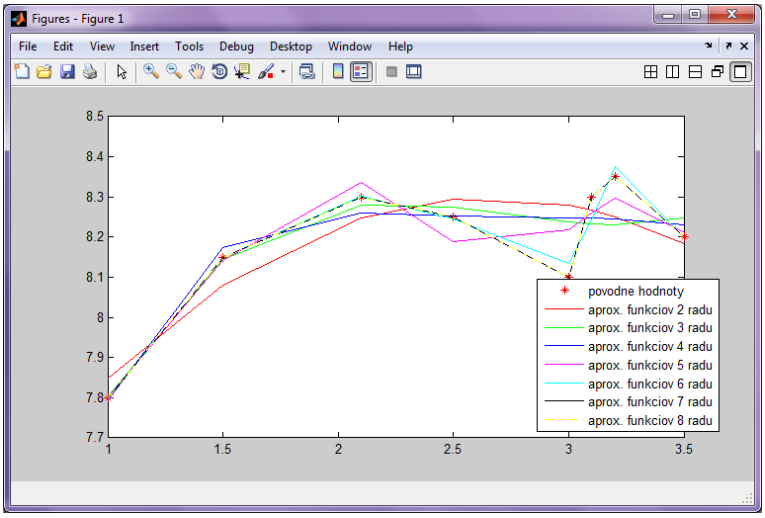
Príklad 4

Aproximujte namerané hodnoty x_i, y_i v tabuľke funkciou (polynómom) 2, 3, ... 8 rádu

x_i	1	1,5	2,1	2,5	3	3,1	3,2	3,5
y_i	7,8	8,15	8,3	8,25	8,1	8,3	8,35	8,2



Obrázok 3-7 Riešenie Príkladu 3 – aproximácia nameraných bodov x_i, y_i polynómom

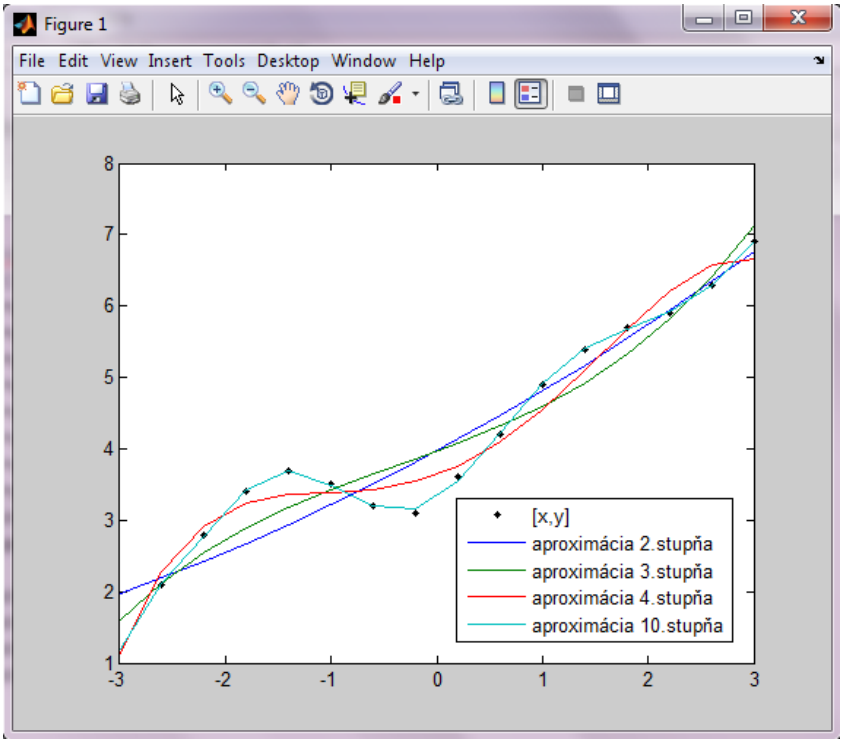


Obrázok 3-8 Grafický priebeh aproximácie funkcie zadanej tabuľkou z Príkladu 3

Príklad
Nech x=-3:0.4:3, y=[1.2 2.1 2.8 3.4 3.7 3.5 3.2 3.1 3.6 4.2 4.9 5.4 5.7 5.9 6.3 6.9]. Aproximujte údaje x_i a y_i polynómami n=2., 3., 4. a 5. stupňa. Vykreslite do grafu príslušné priebehy. Vypočítajte chybu odchyliek pre každú aproximáciu.

Riešenie v prostredí MATLAB:

```
x=-3:0.4:3;
y=[1.2 2.1 2.8 3.4 3.7 3.5 3.2 3.1 3.6 4.2 4.9 5.4 5.7 5.9 6.3 6.9];
a1=polyfit(x,y,2);
a2=polyval(a1,x);
b1=polyfit(x,y,3);
b2=polyval(b1,x);
c1=polyfit(x,y,4);
c2=polyval(c1,x);
d1=polyfit(x,y,10);
d2=polyval(d1,x);
plot(x,y,'k.',x,a2,x,b2,x,c2,x,d2)
legend('[x,y]','aproximácia 2.stupňa','aproximácia 3.stupňa',...
    'aproximácia 4.stupňa','aproximácia 10.stupňa')
odchylka2=sum(sum(y-a2).^2);
odchylka3=sum(sum(y-b2).^2);
odchylka4=sum(sum(y-c2).^2);
odchylka10=sum(sum(y-d2).^2);
```



Obrázok 3-9 Aproximácia polynómu pre n = 2, 3, 4, 5

3.7 Riešenie úlohy interpolácie z nameraných dát v programovom prostredí MATLAB

Interpolácia je postup pre odhad hodnôt bodov, ktoré neboli priamo namerané, ale ležia medzi nameranými uzlovými bodmi. Polynóm $P_n(x)$ je interpolačným polynómom množiny bodov $[x_i, y_i]$, i = 1, 2,...,k práve vtedy, ak pre jeho hodnoty v uzloch interpolácie platí $P_n(x_i) = y_i$, $i = 1, 2, \dots, k$.
Stupeň n interpolačného polynómu $P_n(x)$ je menší nanajvýš rovný $k - 1$.

- ⇨ V praxi je metóda interpolácie používaná pokiaľ poznáme danú funkciu $f(x)$ v určitých diskrétnych bodoch x_i a požadujeme , aby s aproximovanou funkciou $\varphi(x)$ súhlasila vo všetkých bodoch x_i, t. j. platí **interpolačná podmienka**:
  - $f(x_i) = \varphi(x_i), \quad i = 0,1, \dots, n$
- ⇨ Interpolovať dáta teda znamená nájsť z danej sady takú závislosť, ktorá vyhovuje všetkým dátam.
- ⇨ Túto úlohu často potrebujeme v prípadoch, keď poznáme hodnotu funkcie v určitých diskrétnych bodoch $x_i \in (a, b)$ a chceme určiť hodnotu v iných bodoch intervalu *(a,b)*.
- ⇨ Pri interpolacií sa výsledná aproximovaná funkcia φ volí ako lineárna kombinácia čiastkových funkcií φ_i a teda je riešená úloha hľadania koeficientov c_i,*i*=1, ... , *n* tak , aby platilo:

$$\varphi(x_k) = \sum_{i=1}^{n} c_i \varphi_i(x_k) = y_k = f(x_k).$$

Funkcie φ_i sú dopredu zvolené uživateľom.

- **Interpolácia pomocou polynómov**

- ⇨ Všeobecne platí, že $(n+1)$ bodov je možné jednoznačne preložiť polynómom n-tého stupňa. Pre danú množinu čísel x_i a pre danú množinu funkcií φ_i je možné vytvoriť množinu funkcií v_i ortogonálych na množine x_ia generujúcich rovnaký priestor ako množina φ_i. Príkladom je tzv. ***Lagrangeov interpolačný polynóm***.
- ⇨ Ak označíme l_k *k*-tým Lagrangeových polynómom na množine x_i, $k = 1, \dots, n$, potom môžeme zapísať:

$$l_k = \frac{(x - x_1) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}{(x_k - x_1) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)}$$

potom Lagrangeov interpolačný polynóm $L_n(x)$ pre i=0, 1, ..., n má tvar:

$$L_n(x) = \sum_{i=0}^{n} \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)} f(x_i)$$

Príklad
Majme zadané hodnoty funkcie *f* : <1;4> tabuľkou:

x	1	2	4
f(x)	1	4	16

Nahradením funkcie pomocou polynómu $L_2(x)$ sa vypočíta približná hodnota funkcie *f* v bode *x*=3 pomocou $L_2(3)$

$$L_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \cdot f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \cdot f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \cdot f(x_2).$$

Dosadením konkrétnych hodnôt do predchádzajúceho vzťahu dostaneme :

$$L_2(3) = \frac{(3-2)(3-4)}{(1-2)(1-4)} \cdot 1 + \frac{(3-1)(3-4)}{(2-1)(2-4)} \cdot 4 + \frac{(3-1)(3-2)}{(4-1)(4-2)} \cdot 16 = 9.$$

Pre interpoláciu funkčných hodnôt *y* v závislosti na *x (y=f(x))* existuje v prostredí MATLAB funkcia ***interp1***. Pre interpoláciu funkčných hodnôt *z* v závislosti od *x,y* ***(z=f(x,y))*** sa využíva funkcia ***interp2***. Pre závislosť *v=f(x,y,z)* je k dispozícii funkcia ***interp3***.

Funkcia ***interp1*** rieši úlohu interpolácie pomocou vhodne zvolených aproximačných funkcií pre zadefinované body x_i. Syntax funkcie je:

```
interp1(x,y,xi,'metoda'),
```

kde :

x – vektor nameraných hodnôt nezávislej premennej
y – vektor nameraných hodnôt závislej premennej
$\boldsymbol{x_i}$ – body interpolácie – hodnoty pre ktoré nepoznáme funkčnú hodnotu y_i
„metoda“ – zvolená konkrétna aproximačná funkcia pre interpoláciu
$\boldsymbol{y_i}$ – vektor funkčných hodnôt interpolačnej krivky v bodoch x_i

Parameter **‚metóda‘** nie je povinný, predstavuje akou metódou sa má interpolácia vykonať:

‚nearest‘ – interpolácia susedných bodov
‚linear‘ – lineárna interpolácia
‚spline‘ – kubická splajnová interpolácia
‚cubic‘ – štvorcová interpolácia

Syntax funkcie ***interp2*** je:

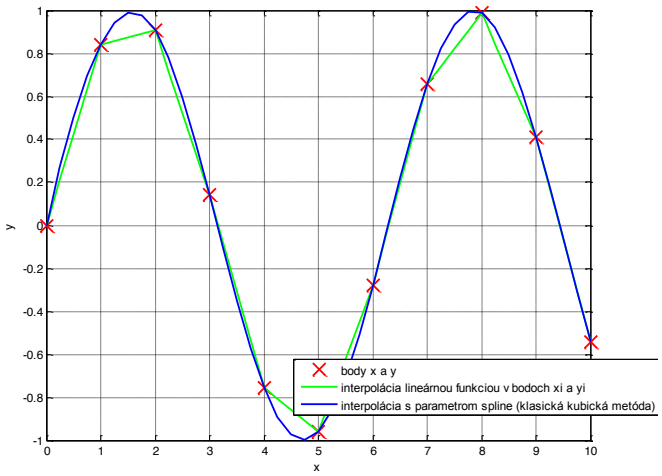
```
interp1(x,y,xi,'metóda')
```

kde *z* je funkčná hodnota pre dvojrozmernú interpoláciu, *x* a *y* sú súradnice interpolácie, *xi* a *yi* súbor bodov interpolácie. Pre parameter **‚metóda‘** platí to, čo pri funkcii *interp1.*

Budeme sa zaoberať možnosťami interpolácie v rovine (2D) a v priestore (3D).

Príklad 1
Vytvorte vektor $x \in <0,10>$ s krokom 1. Vypočítajte v programovom prostredí MATLAB hodnoty $y_i = \sin(x)$ a následne tieto dvojice x_i,y_i vykreslite. Aplikujte metódu interpolácie s využitím funkcie ***interp1*** a **lineárnej /kubickej** funkcie pre aproximáciu hodnôt $x_i \in <0,10>$ s krokom 0,25.

```
>> x=0:10;
>> y=sin(x);
>> plot(x,y,'rx');
>> xi=0:0.25:10;
>> yi=interp1(x,y,xi,'linear');
>> plot(xi,yi,'g');
>> yi=interp1(x,y,xi,'spline');
>> plot(xi,yi,'b');
>> hold on
>> grid on
>> legend('body x a y','interpolácia lineárnou funkciou v
bodoch xi a yi', 'interpolácia s parametrom spline (klasická
kubická metóda)')
```

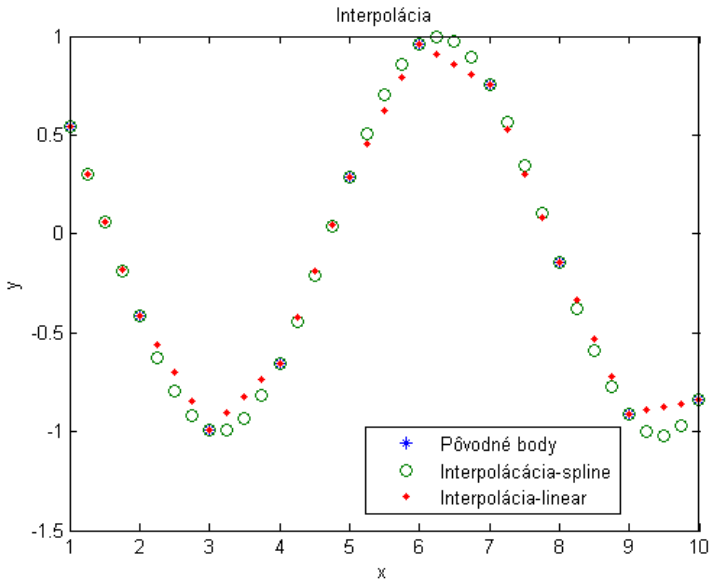


Obrázok 3-10 Interpolácia lineárnou funkciou a splinom v bodoch x_i , y_i

Príklad 2

Interpolujte dáta *x* a *y* so štvornásobnou periódou vzorkovania (so štvrtinovým krokom). Výsledok vykreslite v grafe, kde *x*=0:10, *y*=cos(x).

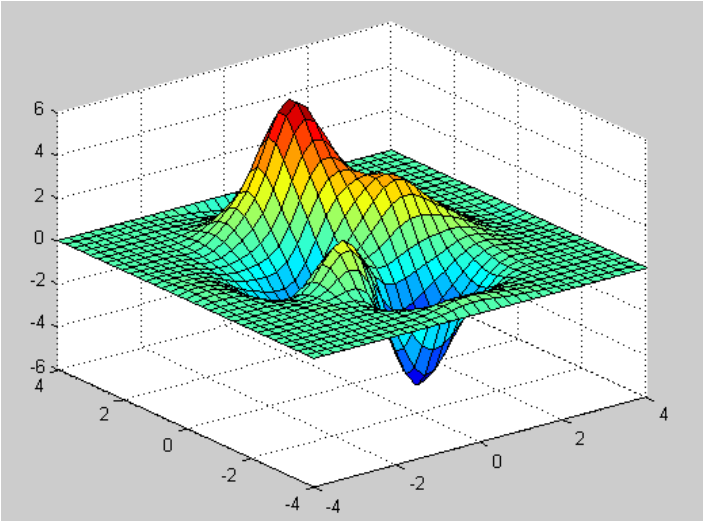
```
x=1:10;
y=cos(x);
xi=1:0.25:10;
yi1=interp1(x,y,xi,'spline');
yi2=interp1(x,y,xi,'linear');
plot (x,y,'*',xi,yi1,'o',xi,yi2,'.')
title ('Interpolácia')
xlabel ('x')
ylabel('y')
legend('Pôvodné body', 'Interpolácácia-spline','Interpolácia-linear')
```



Obrázok 3-11 Interpolácia lineárnou funkciou a splinom v bodoch x_i , y_i

Príklad 3
Vytvorte matice *X* a *Y* ktoré budú obsahovať sieť hodnôt v rozmedzí od <-3;3>. Sieť pre interpoláciu vytvorte s krokom 0,25. Dáta interpolujte druhým stupňom interpolácie. Výsledok zobrazte v grafe pomocou ***surf().***

```
[x,y]=meshgrid(-4:1:4);
z=peaks(x,y);
% Vytvorenie siete pre interpoláciu:
[xi,yi]=meshgrid(-4:0.25:4);
% Interpolácia štvorcovou metódou
zi1=interp2(x,y,z,xi,yi,'cubic');
surf(xi,yi,zi1)
```



Obrázok 3-12 Výsledok interpolácie funkcie z = f(x,y)

4 Aplikácie metód numerickej matematiky

4.1 Numerické a algoritmické riešenie určitého integrálu

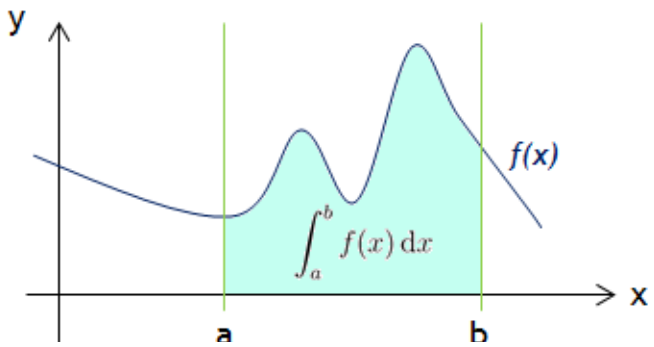
V tejto kapitole sa budeme zaoberať numerickým riešením integrálu. Rozoberieme si dve metódy jeho riešenia a to :

obdĺžnikovú metódu,
lichobežníkovú metódu.

V tejto kapitole sa budeme zaoberať výpočtom integrálu funkcie zadanej tabuľkou.

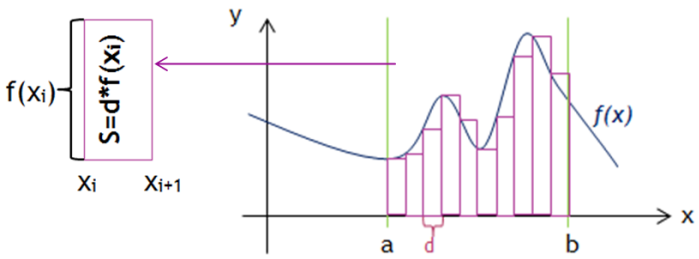
Určitý integrál $\int_a^b f(x)\,\mathrm{d}x$ predstavuje obsah plochy ohraničenej nerovnicami

$a \le x \le b,\ 0 \le y \le f(x)$



Obrázok 4-1 Grafické znázornenia určitého integrálu funkcie *f(x)*

- **Obdĺžniková metóda**



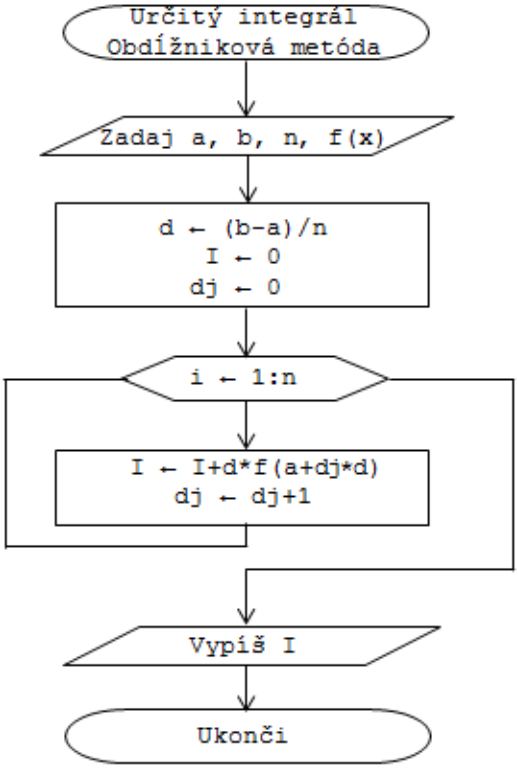
Obrázok 4-2 Obdĺžniková metóda výpočtu určitého integrálu

Podstatou obdĺžnikovej metódy je rozdelenie intervalu *<a,b>* na *n* častí (v našom prípade 10). Plochu rozdelíme na *n* obdĺžnikov, ktorých jednu stranu bude tvoriť premenná $d=(b-a)/n$ alebo $d=x_{i+1}-x_i$ a druhú stranu tvorí funkčná hodnota v bode x_i.
Obsah obdĺžnika potom vieme vypočítať :

$S=d*f(x_i)$

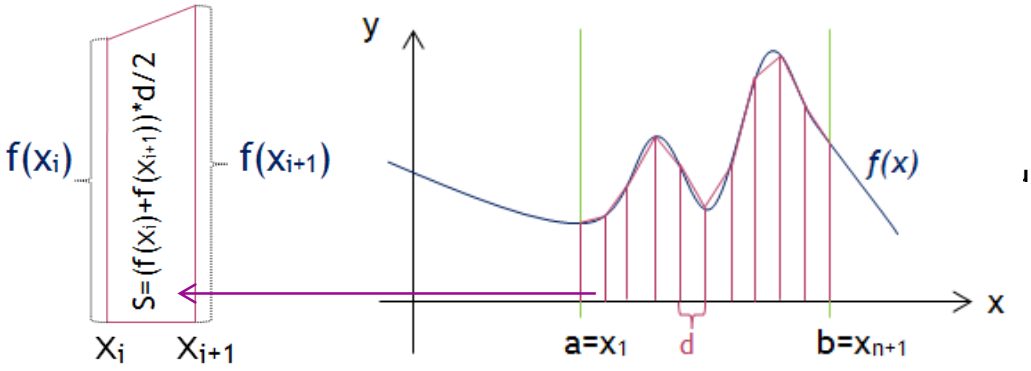
Integrál potom vieme vyjadriť ako súčet obsahov jednotlivých obdĺžnikov.

Algoritmické riešenie určitého integrálu obdĺžnikovou metódou.



Obrázok 4-3 Vývojový diagram výpočtu určitého integrálu obdĺžnikovou metódou

- **Lichobežníková metóda**



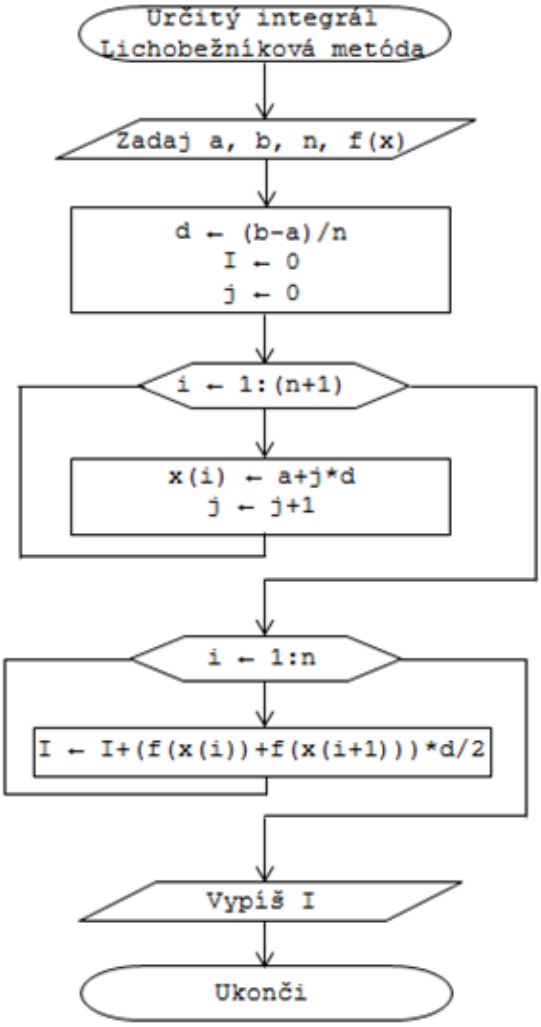
Obrázok 4-4 Lichobežníková metóda výpočtu určitého integrálu

Podstatou lichobežníkovej metódy je rozdelenie intervalu *<a,b>* na *n* častí (v našom prípade 10). Plochu rozdelíme na *n* lichobežníkov, ktorých výšku tvorí $d=(b-a)/n$ alebo $d=x_{i+1}-x_i$. Základňami jednotlivých lichobežníkov sú funkčné hodnoty v bodoch výšky – $f(x_i)$ a $f(x_{i+1})$.
Pre obsah lichobežníka s takto danými stranami platí :

$$S=(f(x_i)+f(x_{i+1}))*d/2$$

Výpočet integrálu sa realizuje sčítaním jednotlivých obsahov lichobežníkov.

Algoritmické riešenie určitého integrálu lichobežníkovou metódou

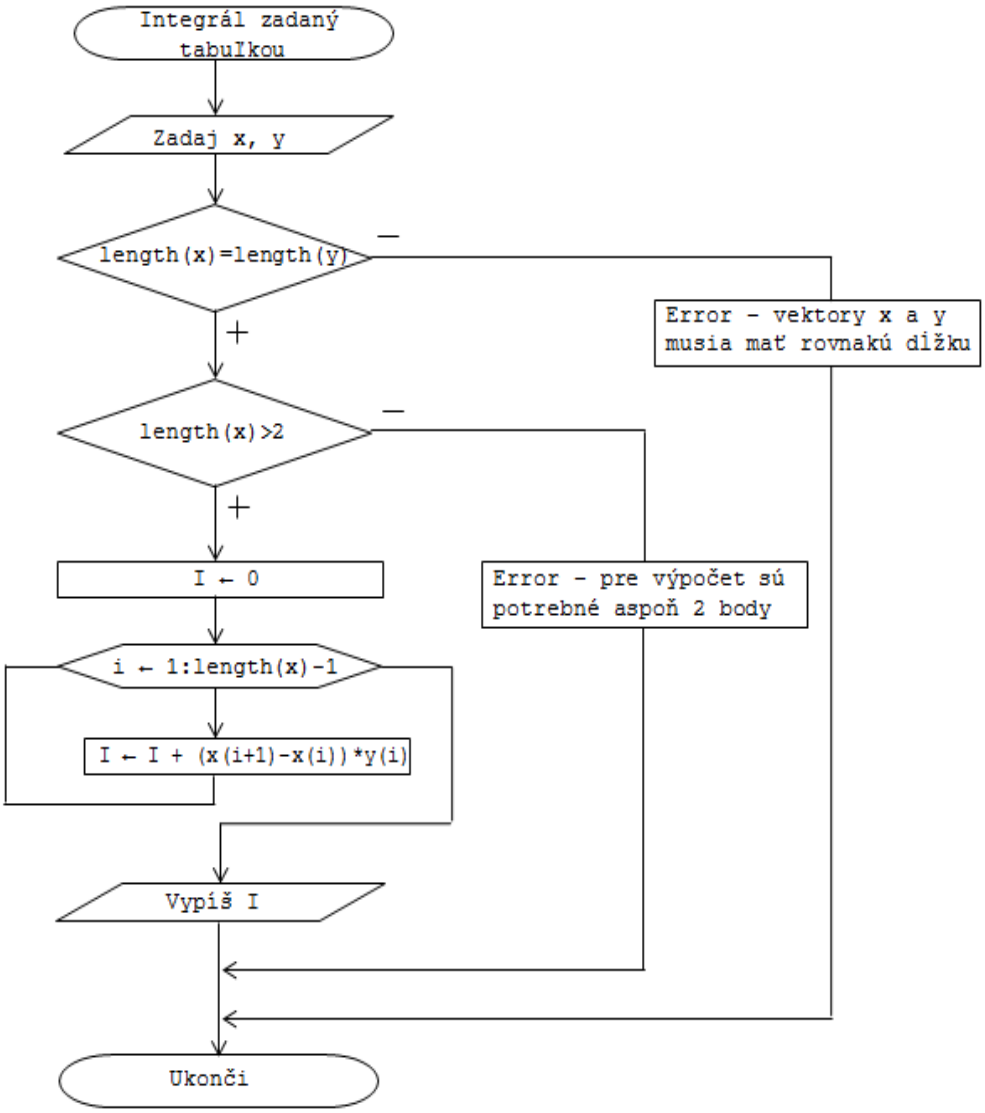


Obrázok 4-5- Vývojový diagram výpočtu určitého integrálu lichobežníkovou metódou

- **Výpočet integrálu z funkcie zadanej tabuľkou – algoritmické riešenie**

V prípade, že predpis funkcie, ktorej integrál máme počítať, nepoznáme a máme len jej body zadané tabuľkou využijeme všetky tieto body $[x_i, y_i]$ *a* výpočet vykonáme po nahradení určitého integrálu súčtom, ktorý vyjadruje nasledujúci vzorec:

$$\sum_{i=1}^{n-1} (x_{i+1} - x_i).y_i$$



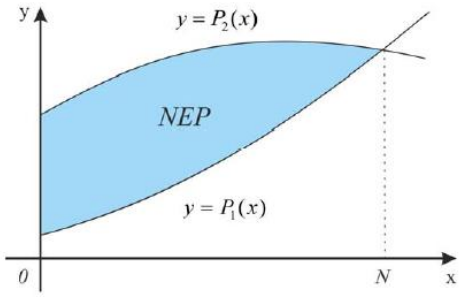
Obrázok 4-6 Vývojový diagram výpočtu určitého integrálu obdĺžnikovou metódou

4.2 Aplikácie určitého integrálu v ekonómii

- **Čistý prebytok zisku**

Majme dva projekty, ktorých rýchlosti ziskov sú dané funkciami $P_1(x)$ a $P_2(x)$, kde x predstavuje počet rokov. Nech funkcia $P_2(x) > P_1(x)$ počas nasledujúcich N rokov.
Čistý prebytok zisku – **NEP** vypočítame nasledovne:

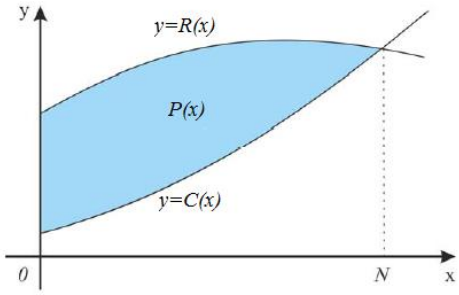
$$NEP = \int_0^N [P_2(x) - P_1(x)]\,dx$$



- **Zisk z výrobného zariadenia**

Výrobné zariadenie vytvára príjem rýchlosťou $R(x)$ a náklady na jeho prevádzku rastú rýchlosťou $C(x)$, kde x predstavuje počet rokov. Zariadenie je ziskové, pokiaľ platí nerovnosť $R(x) > C(x)$.
Zisk z výrobného zariadenia $\boldsymbol{P}$, za N rokov vypočítame podľa vzorca:

$$P(x) = \int_0^N [R(x) - C(x)]\,dx$$



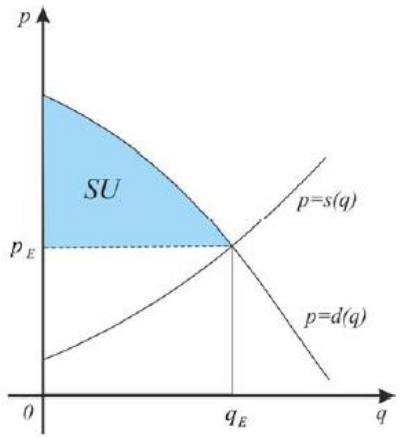
- **Spotrebiteľská úspora, podnikateľský prebytok**

Majme danú funkciu dopytu ***p = d(q)*** a funkciu ponuky ***p = s(q)*** a súradnice rovnovážneho bodu

Obrázok 4-7 Vývojový diagram výpočtu určitého integrálu pre funkciu zadanú tabuľkou

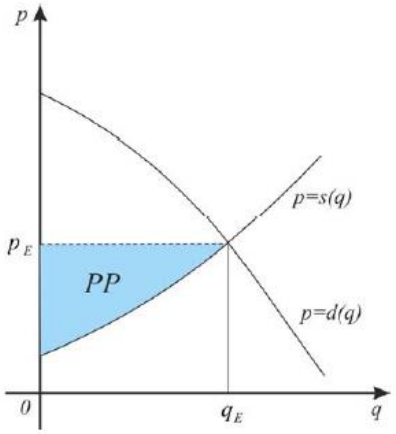
Spotrebiteľská úspora – **SU** predstavuje rozdiel medzi množstvom peňazí, ktoré sú spotrebitelia ochotní minúť za q_E výrobkov a množstvom peňazí, ktoré by minuli pri cene p_E.

$$SU = \int_0^{q_E} d(q)\,dq - p_E q_E$$



Podnikateľský prebytok – **PP** je rozdiel medzi množstvom peňazí, ktoré výrobcovia dostanú za q_E výrobkov predávaných za cenu p_E a množstvom peňazí, za ktoré boli ochotní predať q_E výrobkov.

$$PP = p_E q_E - \int_0^{q_E} s(q)\,dq$$

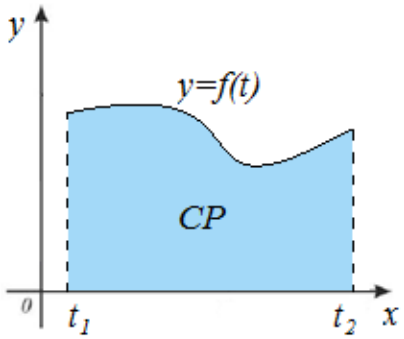


- **Celkový príjem**

Nech je funkcia hustoty toku príjmu *f(t)* v čase *t* spojitá na intervale $<t_1,t_2>$

Celkový príjem – **CP** v čase $<t_1,t_2>$ určíme podľa vzorca:

$$CP = \int_{t_1}^{t_2} f(t)\,dt$$



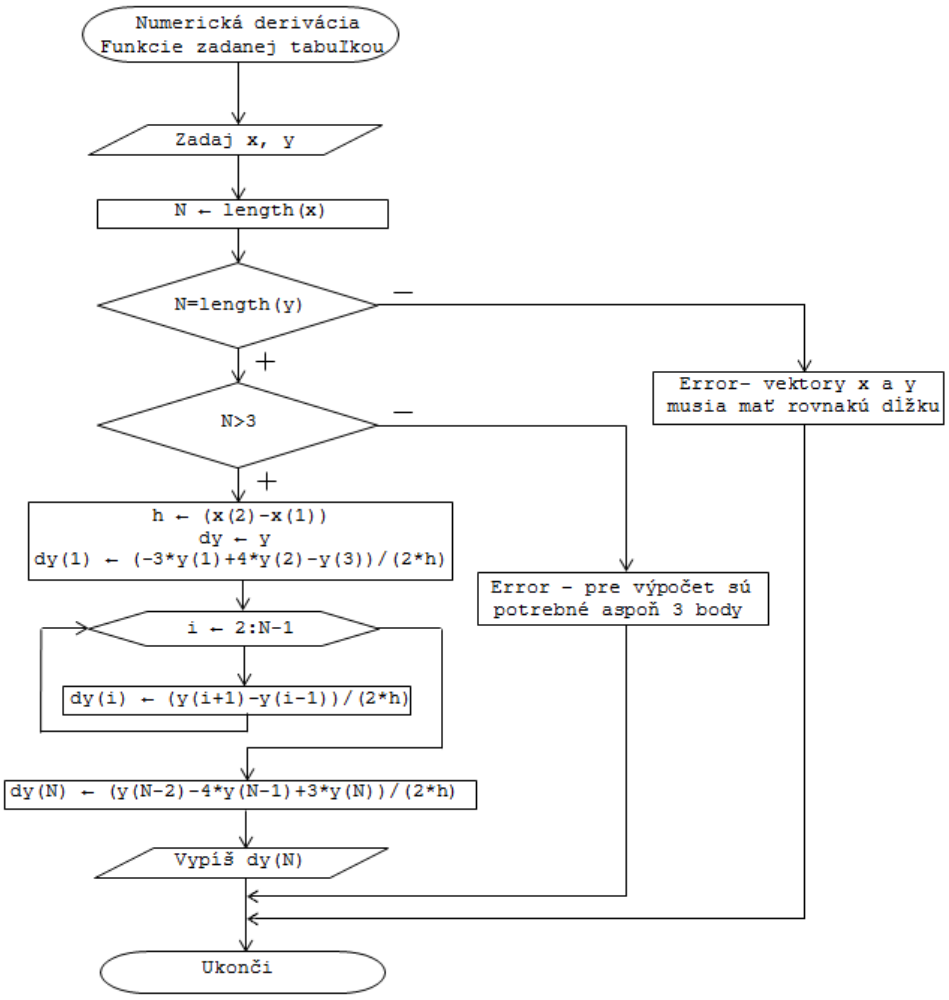
4.3 Algoritmické a numerické riešenie derivácie

Derivácia predstavuje smernicu ku krivke v určitom bode. Pri odhade derivácie funkcie *f(x)* môžeme vychádzať z definície :

$$f'(x) = \lim_{h \to 0}(f(x+h) - f(x))/h$$

kde *h* je z prstencového okolia bodu 0. Po zvolení „malého“ *h*, dostaneme odhad derivácie v bode *x*.

Tento postup však nie je možné použiť pri funkciách zadaných tabuľkou.
V prípade, že je funkcia zadaná tabuľkou, s rovnomerným rozdelením bodov $x_1, x_2, x_3, \ldots, x_{n-1}, x_n$, môžeme použiť interpoláciu 2. stupňa a získame vzťahy pre výpočet derivácie v daných bodoch – deriváciu vo vnútorných bodoch $x_2, \ldots, x_{n-1}$ určíme podľa symetrického vzorca a okraje dopočítame z vnútorných bodov.



4.4 Príklady na samostatné riešenie

1. Vypočítajte integrál $\int_{0,1}^{1,5} \frac{\cos(x)}{x^2}\,\mathrm{dx}$ obdĺžnikovou metódou, pre *n* zadané z klávesnice, riešte pomocou funkcií v MATLAB-e a vlastných naprogramovaných funkcií.
2. Vypočítajte integrál $\int_{0,5}^{2,3} \frac{dx}{\sqrt{(1+3x^2)}}$ obdĺžnikovou metódou pre *n*=5, 10 a 15. Výpočty porovnajte.
3. Vypočítajte integrál $\int_{0}^{2\pi} \sin(x)\,dx$, na výpočet použite lichobežníkovú metódu.
4. Majme funkciu danú bodmi uvedenými v tabuľke. Vytvorte funkciu na numerický výpočet určitého integrálu obdĺžnikovou metódou. Aplikujte túto funkciu na výpočet integrálu funkcie danej nasledujúcou tabuľkou

x	1	2.12	3.24	4.36	5.48	6.6	7.72	8.84
y	0.265	0.369	0.985	1.356	1.452	0.874	1.256	2.012

4.5 Pojem funkcie funkcií v jazyku MATLAB

Hľadanie minima funkcie viacerých premenných, numerický výpočet koreň rovnice *f(x)* = 0, riešenie sústavy diferenciálnych rovníc, numerická integrácia sú často vyskytujúce sa problémy pri riešení inžinierskych úloh.
Simulačný jazyk MATLAB s využitím svojich vstavaných funkcií podporuje riešenie napr. týchto problémov:

a) nájdenie nulovej hodnoty funkcie jednej premennej (riešenie nelineárnych rovníc)
b) nájdenie minima funkcie jednej alebo viacerých premenných,
c) numerický výpočet hodnoty určitého integrálu jednej premennej,
d) riešenie sústavy diferenciálnych rovníc (lineárnych,nelineárnych DR)

Funkcie funkcií sú funkcie, ktoré pracujú s funkciami programového prostredia MATLAB v úlohe ich parametra umožňujú prácu s matematickými funkciami namiesto číselných premenných.
Podmienkou pre používanie štandardných vstavaných funkcií simulačného jazyka MATLAB je programatorska zručnosť pri vytváraní ***m-funkcií*** : funkcie funkcií majú ako prvý parameter meno ***novovytvorenej*** funkcie, ďalšie parametre sú dané syntaxou vstavanej funkcie.
Vstavané funkcie jazyka MATLAB pracujúce s matematickými funkciami sú umiestnené v adresári ***funfun***.

funkcia		***popis***
fmin	→	*minimalizácia funkcie s jednou premennou*
fmins	→	*minimalizácia funkcie s niekoľkými premennými*
fplot	→	*zobrazenie priebehu funkcie*
fzero	→	*nájdenie núl funkcie s jednou premennou*
ode23	→	*riešenie DR* $R-K$ *3.rádu*
ode45		$R-K$ *5.rádu*
quad	→	*numerický integrál* *v tvare nižšieho rádu*
quad8		*v tvare vyššieho rádu*
quadl		

- **Zápis matematických funkcií**

V simulačnom jazyku MATLAB môžeme zapísať funkcie pomocou ***m-súborov*** typu ***funkcia*** alebo s využitím priamých objektov ~ s využitím funkcie ***inline.*** Definícia funkcie ***inline*** je dočasné a pri novom spustení programového systému MATLAB je funkcia nedostupná

Príklad 1
Uvažujeme matematickú funkciu s nasledujúcim predpisom

$$f(x) = \frac{1}{(x-0{,}3)^2+0{,}01} + \frac{1}{(x-0{,}9)^2+0{,}04} - 6$$

Táto funkcia môže byť použitá ako vstup pre už vyššie uvedené funkcie. To si vyžaduje vykonať zápis funkcie ***f(x)*** do m-súboru napr. s názvom ***humps.m***:

Riešenie v programovom prostredí MATLAB:

```
function y = humps(x)          %x je vstupná premenná, y je výstupná premenná
y = 1./((x - 0,3).^2 + 0,01) + 1./((x / 0,9).^2 + 0,04) / 6;
```

Príklad 2
S využitím funkcie ***inline*** vytvorte v simulačnom jazyku Matlab zápis funkie ***f(x)*** z príkladu 1.

Riešenie v programovom prostredí MATLAB:

```
>> f=inline('1./((x - 0.3).^2 + 0.01)+ 1./((x - 0.9).^2 + 0.04)- 6')

f =
      Inline function:

      f(x) = 1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6
```

Pri výpočte funkčnej hodnoty v danom bode je postup rovnaký nezávisle na spôsobe, ako bola funkcia zadefinovaná (m-súbor resp. inline).

Príklad 3
Vypočítajte hodnotu funkcie ***f(x)*** pre $x = 2$.

Riešenie v programovom prostredí MATLAB:

```
>> f(2)

ans =

      -4.8552
```

Vytvorenie funkcie viacerých premenných prebieha pomocou zápisu:

$$inline('funkcia','arg1','arg2',\ldots)$$

kde:
funkcia – reťazec znakov vyjadrujúcich funkciu (vstavanú alebo zadefinovaú užívateľom)
arg1, **arg2** – argumenty – parametre funkcie

Príklad 4
S využitím MATLAB funkcie ***inline*** vytvorte funkciu viacerých premenných

$$f(x,y) = y * sin(x) + x * cos(y).$$

Riešenie v programovom prostredí MATLAB:

```
>> f=inline('y*sin(x)+x*cos(y)', 'x', 'y')

f =
      Inline function:
      f(x,y) = y*sin(x)+x*cos(y)
```

- **Grafické zobrazenie funkcií**

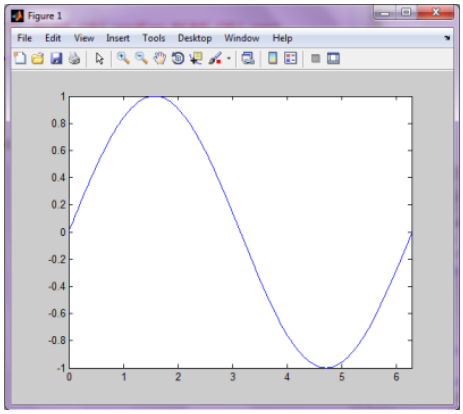
Funkcia ***fplot*** umožňuje zobraziť matematické funkcie, ktoré sú buď vstavané funkcie jazyka MATLAB alebo funkcie vytvorené uživateľom.

$$fplot('fmeno', limit)$$

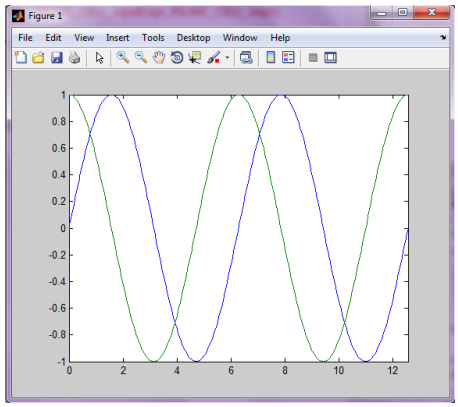
kde:
fmeno – názov funkcie v jazyku MATLAB
limit – interval na ktorom budeme zobrazovať funkciu s názvom ***fmeno***

Príklad 5
Príklad s využitím funkcie ***fplot***. Graficky znázornite goniometrické funkcie na definovanom intervale.

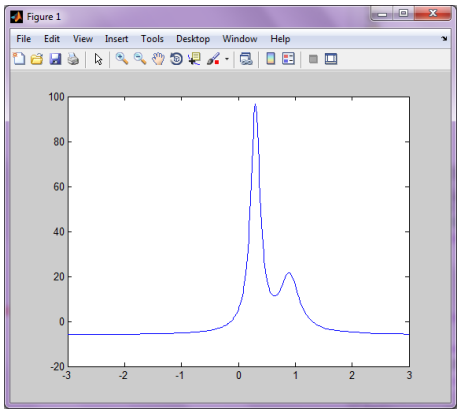
```
>> fplot('sin',[0,2*pi]);
```




```
>> fplot('[sin(x),cos(x)]',[0,4*pi]);
```

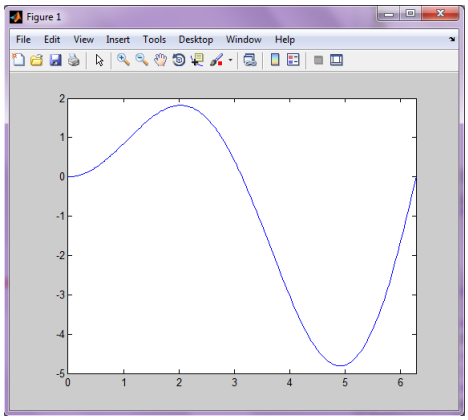



```
>> fplot('humps',[-3,3]);
```



Podobne je možné použiť ako vstupný argument funkcie ***plot*** aj funkciu definovanú ako ***inline*** objekt

```
>> f=inline('x.*sin(x)','x');
>> fplot(f,[0,2*pi])
```



- **Minimum funkcie a hľadanie nulových bodov – funkcia *fminbnd***

Na minimalizáciu zadefinovanej funkcie simulačný jazyk MATLAB využíva vstavané funkcie ktoré umožňujú:

- minimalizácia funkcie s jednou premennou;
- minimalizáciu funkcie s viacerými premennými;
- hľadanie nulového bodu funkcie s jednou premennou:

Minimalizácia funkcie s jednou premennou :

$$fminbnd('fun', x_1, x_2, options)$$

kde:

fun – reťazec znakov, pomocou ktorého je zadefinová funkcia alebo názov premennej, v ktorej je funkcia zadefinová pomocou príkazu ***inline***

x_1, x_2 **–** začiatok a koniec intervalu na ktorom hľadáme minimum

options **–** voľby pre hľadanie minima

Príklad 6

Nájdite minimum funkcie sin na intervale <0,2;π>, využite pritom vstavanú funkciu ***fminbnd.***

Riešenie v programovom prostredí MATLAB:

```
>> k=fminbnd('sin',0,2*pi)
k =
    4.7124
>> min=sin(k)
min =
   -1.0000
```

- **Práca s rovnicami s jednou premennou – funkcia *fzero***

Na riešenie koreňov algebraických rovníc môžeme využiť funkciu ***roots*** (ľavá strana polynómu), avšak na hľadanie numerického riešenia rovnice, ktorá nie je algebraická, používame funkciu ***fzero*** - nájdenie núl funkcie(nulových bodov), ktorú voláme nasledovne:

$$fzero('fun', x_0)$$

kde :

fun – zápis funkcie, ktorá predstavuje ľavú stranu funkcie

x_0 – počiatočný odhad riešenia, alebo interval kde sa ma nachádzať koreň

Príklad 7

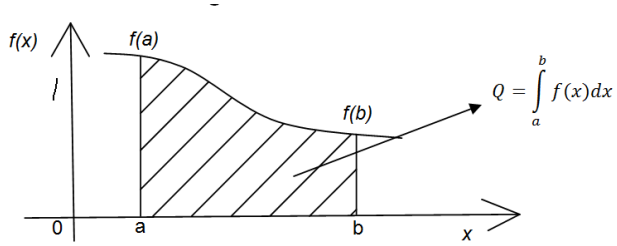
Nájdite riešenie rovnice $\cos(2x) * \sin(3x) = 0$ s využitím funkcie ***fzero***

```
>> x=fzero(inline('cos(2*x)*sin(3*x)'),2)
x =
    2.0944
>> x=fzero(inline('cos(2*x)*sin(3*x)'),[1.5 2.2])
x =
    2.0944
```

Samostatne riešte nasledujúce rovnice s využitím funkcie ***fzero***, s ich úpravou na tvar *f(x)* = 0

1. $x * e^x = 1$
2. $x^2 + x + 1 = 0$
3. $\sin(x) = \frac{x}{10}$

- **Numerická integrácia**



Na numerické integrovanie funkcie jednej premennej v tvare $\int_a^b f(x)dx$ ponúka MATLAB dve štandardné vstavané funkcie ***quad, quadl.***

- ***quad*** – implementuje adaptívne Simpsonovo pravidlo

Pre výpočet Simpsonovým pravidlom rozdelíme interval <*a,b*> na párny počet podintervalov a v každom intervale vykonávame náhradu pôvodnej funkcie parabolou. Vzorec pre výpočet Simpsonovho pravidla je:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(x_0)+f(x_{2n})+4(f(x_1)+f(x_3)+\ \dots\ +f(x_{2n-1}))+2(f(x_2)+f(x_4)+\cdots+f(x_{2n-2}))\right).$$

- ***quadl*** – implementuje adaptívne Lobattovo pravidlo

Funkcie quad,quadl majú analogickú syntax:

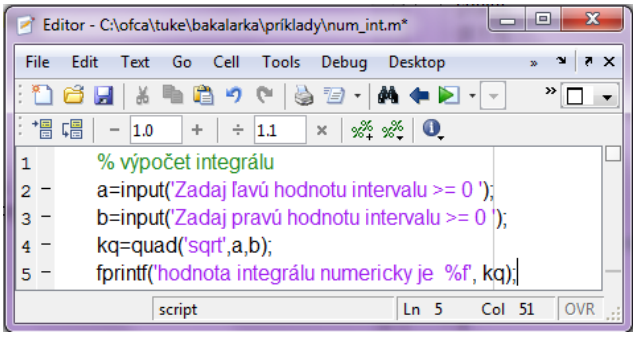
$$hodnota_{int} = quad('fname', dolna\_hr, horna\_hr);$$

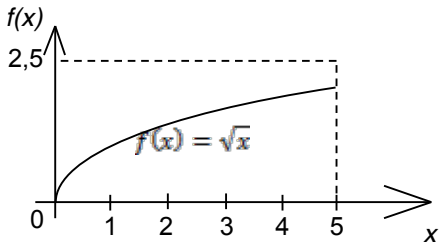
kde:

fname – meno užívateľom vytvorenej funkcie alebo aj meno funkcie programového prostredia MATLAB, (*built-in-funkcie)*, ktoré sú napríklad: sin, sqrt ...
dolna_hr, horna_hr – hranice intervalu, ktorých počítame integrál z definovanej funkcie

Príklad 8
Vypočítajte s využitím funkcie ***quad*** v simulačnom jazyku MATLAB hodnotu integrálu $\int_0^5 \sqrt{x}\,dx;$ na intervale <0,5> .

Riešenie v programovom prostredí MATLAB:





Medzi ďalšie volitelné parametre pre funkciu ***quad*** :

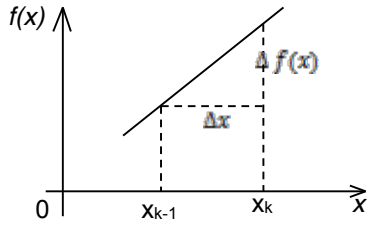
$$Q = quad('fun', a, b, tol, trace)$$

kde:

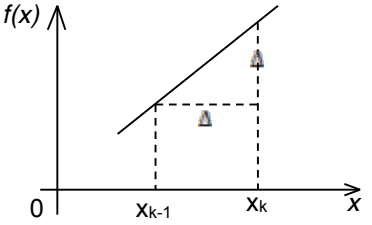
tol- presnosť integrálu ($1e^{-6}$)
trace – nenulový výpis výpočtovej rekurzie

- **Numerická derivácia - funkcia diff**

Funkcia ***diff*** vypočíta diferencie medzi hodnotami vo vektore a vytvára nový vektor:



$f'(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$ spätná diferencia



$f'_{KM}(x_k) = \frac{f(x_{KM}) - f(x_K)}{x_{KM} - x_K}$ dopredná diferencia

Príklad 9
Predpokladajme, že funkcia *f(x)* má tvar polynómu:

$$f(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$$

Vypočítajte deriváciu tejto funkcie na intervale [-4,5] s využitím funkcie ***diff***.

Riešenie v programovom prostredí MATLAB:

```
>> x=-4:0.1:5;%generovanie hodnôt nezávislej premennej
>> f=x.^5-3*x.^4-11*x.^3+27*x.^2+10*x-24;
>> df=diff(f)./diff(x);
```

Príklad na samostatné riešenie:

Vypočítajte integrál $\int_0^{2\pi} \sqrt{4\cos(2t)^2 + \sin(t)^2 + 1}\, dt$ pomocou vytvorenia funkcie ako ***m-súbor*** a ako objekt.

[t,y]=**solver**('odefun', cas_int, poc_pod)

kde***:***
solver – riešiteľ , napríklad vstavané funkcie *ODE45, ODE23* ...
'odefun' – meno ***m-súboru*** funkcie , ktorá obsahuje definovaný predpis systému DR
.

4.6 Analytické a numerické riešenie diferenciálnych rovníc

Fyzikálne systémy (elektrické, mechanické, tepelné, hydraulické) ,ktoré sú vo všeobecnosti popísané systémom lineárnych alebo nelineárnych diferenciálnych rovníc (lineárne dynamické systémy sú popísané LDR s konštantnými koeficientmi).

Analytické riešenie LTI (Linear Time Invariant) systému DR v časovej oblasti získame:

- riešením DR bez a s pravou stranou a sčítaním homogenného a partikulárneho riešenia
- riešením v Laplaceovej transformácii a časový originál riešenia získame spätnou Laplaceovou transformáciou ($\mathcal{L}^{-1}$).

DR môžeme riešiť **numerickými metódami**. Funkciu, ktorú dostávame po každom kroku pri numerickom výpočte nazývame aproximácia analytického riešenia.

Pri riešení diferenciálnych rovníc v sa budeme vžívať dva postupy a to :

- na riešenie DR použijeme vstavanú funkciu **ode45 (využíva metódu Runge-Kutta 4.stupňa),**
- a vlastnú funkciu využívajúcu algoritmus metódy **Runge – Kutta.**

Je nutné si uvedomiť, že simulačný jazyk MATLAB neumožňuje riešiť DR vyššieho rádu ako 1. rádu, t.j. DR s definovanou počiatočnou podmienkou v tvare

$$\frac{dy(t)}{dt} = f(t,y), \quad PP: y(0) = y_0$$

Z tohto dôvodu je pri oboch postupoch potrebné vykonať transformáciu DR *n*-tého rádu na systém *n* diferenciálnych rovníc 1. rádu. (prepísanie DR do substitučného kanonického tvaru). Princíp spočíva v vysvetlený na konkrétnych príkladoch DR.

- **Prepis diferenciálnej rovnice na substitučný kanonický tvar**

Prepis lineárnej DR 2.rádu do substitučného kanonického tvaru:

Uvažujme DR, ktorá ma nasledujúci tvar:

$$y''(t) + 3y'(t) + y(t) = 1$$

Na prepis do substitučného kanonického tvaru zvolíme substitúciu :

$$y = x_1(t),\ y' = x_2(t)$$

$y'(t) = x_1'(t) = x_2(t)$
$y''(t) = x_2'(t) = 1 - 3x_2(t) + x_1(t)$ → Substitučný kanonický tvar

V každom kroku výpočtu je dôležité poznať hodnoty stavových premenných ktoré sú reprezentované vektorom [*x1; x2*]**.** V prostredí MATLAB si vytvoríme funkciu, ktorú nazveme ***„dif.m“***. Jej výstupný parameter (***xout***) predstavuje v každom kroku derivácie stavových premenných: **xout =** [*x1'; x2'*].

```
function [ xout ] = dif(t,x)
x1=x(2);
x2=x(1)-3*x(2)+1;
xout=[x1;x2];
end
```

Prepis lineárnej DR 3.rádu do substitučného kanonického tvaru:

Uvažujme všeobecnú DR 3.rádu s pravou stranou.

$$y'''(t) + a_2 y''(t) + a_1 y'(t) + a_0 y(t) = b_0 * u(t)$$

Definujme stavové premenné: $y = x_1(t)$, $y' = x_2(t)$, $y'' = x_3(t)$

Prepis do substitučného kanonického tvaru:

$$y'(t) = x_1'(t) = x_2(t)$$
$$y''(t) = x_2'(t) = x_3(t)$$
$$y'''(t) = x_3'(t) = b_0 u(t) - a_0 x_1 - a_1 x_2 - a_2 x_3$$

DR n = 3 je prepísaná do substitučného kanonického tvaru troch DR 1.rádu.

- **Riešenie DR s využitím vstavanej funkcie *ode45* jazyka MATLAB**

Funkcia ***ode45*** vypočíta riešenie DR *n*-tého rádu, ktorá bola zapísana do substitučného kanonického tvaru(súbor ***„dif.m“***), funkcia ***ode45*** používa názov funkcie ***„dif“*** ako parameter. Funkcie na riešenie diferenciálnych rovníc teda považujeme za ***„funkcie funkcií“***. Základná syntax funkcie ode45 je nasledovná:

```
[t,x] = ode45(@dif, [t0,tf], [P])
```

kde:

dif – funkcia, ktorá obsahuje definovaný predpis riešenej DR v tvare systému diferenciálnych rovníc 1. rádu

[t0,tf] – začiatočná a koncová hodnota intervalu, na ktorom vykonávame riešenie DR

[P] – vektor počiatočných podmienok

Výstup z funkcie ***ode45***:

t – stĺpcový vektor časových bodov
y – matica riešenia (jednotlivé stĺpce obriešenie DR a jeho derivácie)

Výsledok riešenia funkcie ***ode45*** je možné vykresliť pomocou funkcie ***plot*** nasledovne :

```
plot(t,y(:,1))          %riešenie DR x1(t)
plot(t,y (:,2))         %derivácia riešenia DR x2(t)
```

Pozn. Namiesto funkcie ode45 je možné použiť aj iné solvery (riešitele), napr. ode23 a

Príklad
Uvažujeme nelineárnu diferenciálnu rovnicu 2. rádu ,ktorú reprezentuje Van-der-Polov oscilátor. Namodelujte riešenie tejto DR v simulačnom jazyku MATLAB.

$$y''(t) = g(t, y, y') = y'(1 - y^2) - y$$

$$y''(t) - y'(t) * (1 - y^2(t)) + y(t) = 0$$

Substitúcia: $y = x_1(t)$, $y' = x_2(t) y(t) = x_1(t)$

Vytvorenie substitučného kanonického tvaru:

$$y'(t) = x_1'(t) = x_2(t)$$
$$y''(t) = x_2' = x_2(t) * (1 - x_1^2(t)) - x_1(t)$$

vander.m:

```
% Matematicky zápis DR2.rádu prepísaný
%do stavového priestoru
function xder = vander(t,x)
xder=[x(2);x(2).*(1-x(1).^2)-x(1)];
return
```

Pozn. `xder = [x(2);x(2).*(1 - x(1).^2) / x(1)]` - maticový zápis

Hlavný program :

difrov2r.m :

```
% program na riešenie nelineárnej diferenciálnej rovnice
% 2.rádu -  Van - Der - Pol
% oscilátor - zadaný v stavovom priestore
x0=[0 0.25]'; % inicializácia počiatočných podmienok
t0=0; tf=10;  % definícia časového intervalu
[t,x]=ode23('vander',[t0,tf],[0;0.25]);
[t,x]=ode45('vander',[t0,tf],[0;0.25]);
subplot(211); plot(t,x(:,1)); ...
title('riešenie y(t)'); xlabel('t'); grid; ...
subplot(212); plot(t,x(:,2)); ...
title('prvá derivácia y(t)'); xlabel('t'); grid; ...
plot(t,x); % dva grafy v jednom obrázku
title('Van-der-Pol rovnica - časová história ');
pause
```

```
plot(t,x(:,1),'k-',t,x(:,2),'k--')
```
%využitie viacparametrovej funkcie na grafický výstup

- **Metóda Runge – Kutta 4.rádu**

Metóda Runge – Kutta je jednou z najznámejších numerických metód na riešenie DR. Podstata spočíva v aproximácii lineárnych kombinácií funkčných hodnôt v zvolených bodoch. Vyjadrenie nového stavu systému pomocou predchádzajúceho stavu je dané vzťahmi :

$$k_1 = T * f(t_n, x_n)$$

$$k_2 = T * f(t_n + \tfrac{1}{2}T, x_n + \tfrac{1}{2}Tk_1)$$

$$k_3 = T * f(t_n + \tfrac{1}{2}T, x_n + \tfrac{1}{2}Tk_2)$$

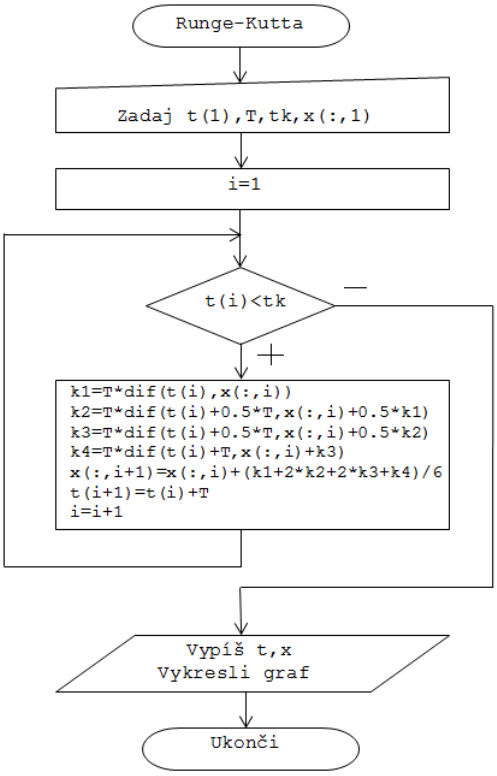
$$k_4 = T * f(t_n + T, x_n + Tk_3)$$

$$x_{n+1} = x_n + (k_1+2k_2+2k_3+k_4)/6$$

pre $n = 1, 2, 3...$

Zavedieme si premenné:

t_k – hodnota času t, pri ktorej chceme výpočet ukončiť
T – integračný krok, s ktorým sa bude výpočet realizovať
t(1) – počiatočná hodnota t, pri ktorej sa výpočet odštartuje ak $i = 1$
x(:,1) – počiatočný vektor x závislej premennej



Obrázok 4-8 Vývojový diagram pre metódu Range – Kulta – vlastná naprogramovaná funkcia

4.7 Riešenie lineárnych diferenciálnych rovníc analyticky a s využitím vstavaných funkcií v jazyku MATLAB

Diferenciálne rovnice nazývame rovnice, v ktorých sa vyskytujú funkcie a ich derivácie. K diferenciálnym rovniciam nás vedú fyzikálne a matematické úlohy.

- **Druhy diferenciálnych rovníc**

a) **Obyčajné diferenciálne rovnice** – ako neznáma vystupuje reálna funkcia jednej reálnej premennej a tiež derivácie tejto funkcie.
   - Hovoríme, že diferenciálna rovnica je *n*-tého rádu, ak v nej vystupuje *n*-tá derivácia neznámej funkcie *y* a ak už v nej nevystupuje žiadna jej derivácia vyššieho rádu ako *n*.
     - Lineárne diferenciálne rovnice
     - Nelineárne diferenciálne rovnice

b) **Parciálne diferenciálne rovnice** – ako neznáma vystupuje funkcia dvoch alebo viac premenných a v ktorej vystupujú parciálne derivácie tejto neznámej funkcie.

- **Lineárne diferenciálne rovnice 1.rádu**

V úlohách, ktoré vedú na riešenie diferenciálnych rovníc 1. rádu je obyčajne vopred známa hodnota hľadanej funkcie v nejakom bode t_0.
Úlohou je nájsť medzi všetkými riešeniami diferenciálnej rovnice $y' = f(t, y)$ také riešenie $y = y(t)$, ktoré spĺňa počiatočnú podmienku $y(t_0) = t_0$, kde y_0 je dané číslo. Takýto typ úlohy sa nazýva *Cauchyho úloha.*

- **Lineárne diferenciálne rovnice n-tého rádu**

Homogénna lineárna diferenciálna rovnica (DR bez pravej strany) je DR tvaru

$$a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \cdots + a_1 y'(t) + a_0 y(t) = 0,$$

kde a_i, i =0,1, ..., *n* sú koeficienty DR.

Nehomogénna lineárna diferenciálna rovnica (DR s pravou stranou) je DR tvaru

$$a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \cdots + a_1 y'(t) + a_0 y(t) = u,$$

kde a_i, $i = 0,1,\dots,n$ sú koeficienty DR a u je budiaca funkcia.

Všeobecné riešenie y DR je súčtom homogenného riešenia $\bar{y}$ a partikulárneho riešenia y^* , tj. $y = \bar{y} + y^*$.

- **Riešenie LDR Laplaceovou transformáciou**

Laplaceova transformácia nám ponúka jednoduché riešenie lineárnych diferenciálnych rovníc s konštantnými koeficientami.

1) DR v časovej oblasti pretransformujeme do LT so zohľadnením počiatočných podmienok
2) Vypočítame obrazový prenos systému $G(s) = \frac{Y(s)}{U(s)}$
3) Z obrazového prenosu systému vyjadríme obraz riešenia DR *Y(s)*

Y(s) = G(s) . U(s)

kde

$U(s)$ predstavuje vstup do systému
$Y(s)$ predstavuje výstup zo systému

4) Použitím vzorcov pre spätnú LT získame riešenie DR v časovej oblsati.

Y(s) → y(t)

PRÍKLAD 1

Majme LDR tvaru:

$$y''(t) - 2y'(t) - 3y(t) = 7 * \cos(3t).$$

Nájdite všeobecné riešenie LDR pre nulové počiatočné podmienky analytickým výpočtom a s využitím vstavanej funkcie ode45 v jazyku MATLAB.

Riešenie DR v časovej oblasti:

1. Najprv vyriešime homogénnu DR
   $y''(t) - 2y'(t) - 3y(t) = 0$
   Pre výpočet homogénnej DR potrebujeme vypočítať charakteristickú rovnicu a jej korene:

   $s^2 - 2s - 3 = 0$
   $(s+1)(s-3) = 0$
   $s_1 = -1$
   $s_2 = 3$
   Na základe koreňov charakteristickej rovnice všeobecné riešenie $\bar{y} = C_1 * e^{-t} + C_2 * e^{3t}$

2. Následne vypočítame partikulárne riešenie DR s uvažovaním špeciálnej pravej strany
   V našom prípade je pravá strana $f(t) = 7 * \cos(3t)$.
   Z všeobecného partikulárneho riešenia $y^* = x^k * e^{ax} * (A * \cos(bx) + B * \sin(bx))$ odhadneme $s = 0$, $a = 0$, $b = 3$.
   Keďže komplexne združené číslo $a + ib = 3i$ nie je v tomto prípade žiadnym koreňom, preto $k = 0$.
   A teda partikulárne riešenie tejto LDR je
   $y^* = x^0 * e^{0t} * (A * \cos(3t) + B * \sin(3t)) = A * \cos(3t) + B * \sin(3t)$

3. Získané partikulárne riešenie zderivujeme:

   $(y^*)' = -3A * \sin(3t) + 3B * \cos(3t)$
   $(y^*)'' = -9A * cos(3t) - 9B * \sin(3t)$

4. Dosadíme partikulárne riešenia do pôvodnej LDR

   $(y'' - 2y' - 3y = 7 * \cos(3t))$
   $-9A * \cos(3t) - 9B * \sin(3t) - 2 * (-3A * \sin(3t) + 3B * \cos(3t)) -$
   $-3 * (A * cos(3t) + B * \sin(3t)) = 7 * \cos(3t)$

Upravíme:

$$-9A * \cos(3t) - 9B * \sin(3t) + 6A * \sin(3t) - 6B * \cos(3t) - 3Acos(3t) - 3B * \sin(3t) = 7 * \cos(3t)$$
$$(-12A - 6B) * \cos(3t) + (-12B + 6A) * \sin(3t) = 7 * \cos(3t)$$

$\cos(3t)$: $\quad -12A - 6B = 7 \quad\quad A = -\frac{7}{15}$

$\sin(3t)$: $-12B + 6A = 0 \quad\quad B = -\frac{7}{30}$

Partikulárnym riešením DR je $y^* = -\frac{7}{15} * \cos(3t) - \frac{7}{30} * \sin(3t)$

5. Všeobecné riešenie danej nehomogénnej LDR hľadáme v tvare $y = \bar{y} + y^*$

$$y = C_1 * e^{-t} + C_2 * e^{3t} - \frac{7}{15} * \cos(3t) - \frac{7}{30} * \sin(3t)$$

6. Určenie integračných konštanty z počiatočných podmienok:

$y(0) = 0$:

$$y(t) = C_1 * e^{-t} + C_2 * e^{3t} - \frac{7}{15} * \cos(3t) - \frac{7}{30} * \sin(3t)$$

$$y'(t) = -C_1 * e^{-t} + 3C_2 e^{3t} + \frac{7}{5}\sin(3t) - \frac{7}{10}\cos(3t)$$

$$C_1 * e^{-1*0} + C_2 * e^{3*0} - \frac{7}{15} * \cos(3 * 0) - \frac{7}{30} * \sin(3 * 0) = 0$$

$$C_1 + C_2 - \frac{7}{15} + 0 = 0$$

$$C_1 = \frac{7}{15} - C_2$$

$$C_1 = \frac{7}{40}$$

$y'(0) = 0$:

$$-C_1 * e^{-t} + 3C_2 e^{3t} + \frac{7}{5}\sin(3t) - \frac{7}{10}\cos(3t) = 0$$

$$-C_1 + 3 * C_2 + 0 - \frac{7}{10} = 0$$

$$4 * C_2 = \frac{7}{6}$$

$$C_2 = \frac{7}{24}$$

7. Celkové analytické riešenie DR :

$$y = \frac{7}{40} * e^{-t} + \frac{7}{24} * e^{3t} - \frac{7}{15} * \cos(3t) - \frac{7}{30} * \sin(3t)$$

<u>Nájdenie riešenia DR pomocou Laplaceovej transformácie</u>

$$y''(t) - 2 * y'(t) - 3 * y(t) = 7 * cos(3t),\ PP: y(0) = y'(0) = 0$$

1. Prevedieme LDR do Laplaceovej transformácie

$$s^2Y(s) - sy(0) - y'(0) - 2 * \big(sY(s) - y(0)\big) - 3Y(s) = \frac{7s}{s^2 + 3^2}$$

$$s^2Y(s) - 2sY(s) - 3Y(s) = \frac{7s}{s^2 + 3^2}$$

2. Vyjadríme obraz riešenia DR v LT : Y (s) :

$$Y(s) * (s^2 - 2s - 3) = \frac{7s}{s^2 + 3^2}$$

$$Y(s) = \frac{7s}{(s^2 + 3^2) * (s^2 - 2s - 3)}$$

3. Rozklad prenosovej funkcie na parciálne zlomky

$$\frac{7s}{(s^2+3^2)*(s^2-2s-3)} = \frac{A}{s+1}+\frac{B}{s-3}+\frac{Cs+D}{s^2+9}$$

$$7s = A*(s^2+9)*(s-3)+B*(s^2+9)*(s+1)+(Cs+D)*(s-3)*(s+1)$$

$$\begin{aligned} s^3&: \quad 0 = A+C+D \\ s^2&: \quad 0 = -2A+B-3C+D \\ s^1&: \quad 7 = -3A-2B+9C+9D \\ s^0&: \quad 0 = -3B-27C+9D \end{aligned}$$

$$\left(\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 0 \\ -2 & 1 & -3 & 1 & 0 \\ -3 & -2 & 9 & 9 & 7 \\ 0 & -3 & -27 & 9 & 0 \end{array}\right) \Rightarrow \begin{aligned} A &= \tfrac{7}{40} \\ B &= \tfrac{7}{24} \\ C &= -\tfrac{7}{15} \\ D &= -\tfrac{7}{10} \end{aligned}$$

$$\frac{7s}{(s^2+3^2)*(s^2-2s-3)} = \frac{\frac{7}{40}}{s+1}+\frac{\frac{7}{24}}{s-3}-\frac{\frac{7}{15}s+\frac{7}{10}}{s^2+9}$$

4. Následne použijeme spätnú laplaceovú transformáciu, aby sme získali analytické riešenie DR y(t) v časovej oblasti.

$$\frac{7}{40}*\frac{1}{s+1} \div \frac{7}{40}*e^{-t}$$

$$\frac{7}{24}*\frac{1}{s-3} \div \frac{7}{24}*e^{3t}$$

$$\frac{-\frac{7}{15}s-\frac{7}{10}}{s^2+9} = -\frac{7}{15}*\frac{s}{s^2+3^2}-\frac{7}{10}*\frac{1}{s^2+3^2}$$

$$-\frac{7}{10}*\frac{1}{s^2+3^2} = -\frac{7}{30}*\frac{3}{s^2+3^2} \div -\frac{7}{30}*\sin(3t)$$

$$-\frac{7}{15}*\frac{s}{s^2+3^2} \div -\frac{7}{15}*\cos(3t)$$

5. Riešenie y(t) pomocou spätnej Laplaceovej transformácie

$$y = \frac{7}{40}*e^{-t}+\frac{7}{24}*e^{3t}-\frac{7}{15}*\cos(3t)-\frac{7}{30}*\sin(3t)$$

Riešenie v jazyku MATLAB :

Pri riešení funkciami DR v simulačnom jazyku MATLAB je nutné si uvedomiť, že MATLAB neumožňuje riešiť DR vyššieho rádu, ale iba systém diferenciálnych rovníc 1.rádu. Preto používame transformáciu diferenciálnych rovníc *n*-tého rádu na diferenciálne rovnice 1-ho rádu.

Prepis do substitučného kanonického tvaru

$$y(t) = x_1$$

$$y'(t) = x_1' = x_2$$

$$y''(t) = x_2' = \frac{1}{a_2}*(7\cos(3t)-a_1*x_2-a_0*x_1) = 7\cos(3t)+2x_2+3x_1$$

Program v simulačnom jazyku MATLAB

difrov.m

```
function xder = difrov(t,x)
%Zápis diferenciálnej rovnice 2.radu pomocou 2rovnic 1.radu
global  a0 a1 a2;
xder=[x(2) ; (7*cos(3*t)-a1.*x(2)-a0.*x(1))./a2];
return
```

analyt.m

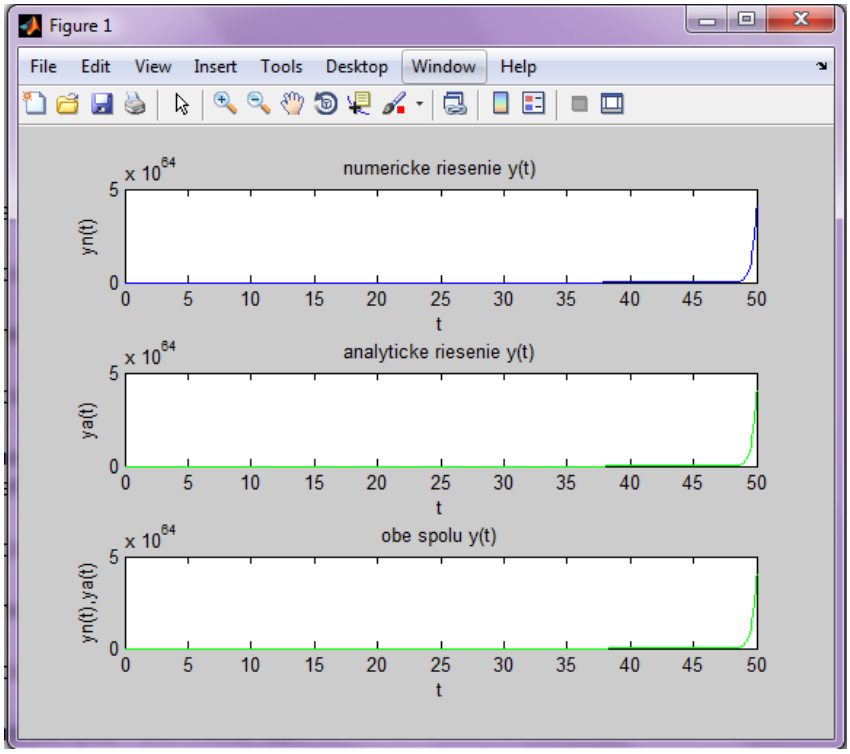
```
%analytické riešenie
function d=analyt(t)
d=(7/40)*exp(-t)+(7/24)*exp(3*t)-(7/15)*cos(3*t)-(7/30)*sin(3*t);
return
```

chyba.m

```
%odhad chyby:
function chyba = rozdiel(d,y)
rozdiel = abs(d-y(:,1));
chyba = max(rozdiel);
fprintf('Maximálna odchýlka = %d \n', chyba)
return
```

Hlavný program

```
global a;
a = input('Zadaj koeficienty LDR: [a(1) a(2) a(3)]\n');
T(1) = input('Zadaj počiatočnú hodnotu časového intervalu pre riešenie LDR:
 \n');
T(2) = input('Zadaj konečnú hodnotu časového intervalu pre riešenie LDR:
 \n');
PP = input('Zadaj počiatočné podmienky: [P(0) P(1)]\n');

 %riešenie LDR pomocou funkcie ode45:
[t,y]=ode45('difrov',T,PP);

%analytické riešenie:
d = analyt(t);

%odhad chyby:
chyba = rozdiel(d,y);
%vykreslenie vyriešených priebehov:
subplot(3,1,1)
plot(t,y(:,1))
title('numerické riešenie y(t)'),xlabel('t'),ylabel('yn(t)')
subplot(3,1,2)
plot(t,d,'g')
title('analytické riešenie y(t)'),xlabel('t'),ylabel('ya(t)')
subplot(3,1,3)
plot(t,y(:,1),t,d,'g')
title('obe spolu y(t)'),xlabel('t'),ylabel('yn(t),ya(t)');
```



Obrázok 4-9 Porovnanie analytického a numerického riešenia LDR v prostredí MATLAB s využitím funkcie odr45

PRÍKLAD 2

$y''(t) + 2y'(t) + y(t) = t \qquad PP: y(0) = y'(0) = 0$

Riešenie homogénnej diferenciálnej rovnice

$y'' + 2y' + y = 0$

1. Charakteristická rovnica: $s^2 + 2s + 1 = 0$
   $(s+1)(s+1) = 0$

Korene charakteristickej rovnice : $s_1 = -1 \qquad s_2 = -1$

Všeobecné riešenie vyplývajúce z charakteristickej rovnice: $\bar{y} = C_1 e^{-t} + C_2 t e^{-t}$

2. Riešenie špeciálnej pravej strany :

   $f(t) = t$

   Partikulárne riešenie : $y^*(t) = A_1 t + A_0$, pričom

   $A_0 = -2 \qquad A_1 = 1$

3. Všeobecné riešenie DR $y = \bar{y} + y^*$ má tvar:

   $$y = C_1 e^{-t} + C_2 t e^{-t} + t - 2$$

4. Výpočet integračných konštánt C_1, C_2 z počiatočných podmienok:

$$y(0) = 0: \qquad C_1 = 2$$

$$y'(0) = 0: \qquad C_2 = 1$$

Celkové riešenie lineárnej DR získane analyticky má tvar:

$$y(t) = t - 2 + 2e^{-t} + te^{-t}$$

Riešenie lineárnej DR pomocou Laplaceovej transformácie

$$y''(t) + 2y'(t) + y(t) = t\,, počiatočné\ podmienky:\ PP: y(0) = y'(0) = 0$$

1. Prepis do LT LDR: $s^2Y(s) + 2sY(s) + Y(s) = \frac{1}{s^2}$

2. Lapl. obraz riešenia lineárnej DR Y(s) má tvar: $Y(s) = \frac{1}{s^2 * (s^2 + 2s + 1)}$

3. Rozklad Laplacovho obrazu riešenia DR (prenosovej funkcie) Y(s) na parciálne zlomky

$$\frac{1}{s^2 * (s+1)^2} = \frac{A}{s^2} + \frac{B}{s} + \frac{C}{(s+1)^2} + \frac{D}{s+1}$$

Výpočtom získame : $A = 1$, $B = -2$, $C = 1$, $D = 2$

4. Celkové riešenie DR v časovej oblsati získame spätnou Laplaceovou transformáciou :

$$y(t) = t - 2 + t * e^{-t} + 2 * e^{-t}$$

Príklad:
Vytvorte funkciu v prostredí MATLAB, ktorá numericky vyrieši zadanú diferenciálnu rovnicu s využitím vstavanej funkcie ***ode45***. Uvažovanú DR prepíšte do substitučného kanonického tvaru.

4.8 Zadanie č.2: Riešenie LDR (*n* = 2) s konštantnými koeficientami analyticky a algoritmicky v prostredí MATLAB

ZADANIE:

Riešenie LDR 2. alebo 3. rádu s konštantnými koeficientami analyticky a algoritmicky v programovom prostredí MATLAB.

OBSAH ZADANIA:

A. *Analyticky vyriešiť a nájsť celkové riešenie LDR v časovej oblsati.*
B. *Analyticky vyriešiť celkové riešenie LDR v Laplaceovej transformácií s uvažovaním nájdenia časového originálu riešenia y(t) aplikovaním inverznej Laplaceovej transormácie.*
C. *Programové riešenie v simulačnom jazyku MATLAB (s využitím vstavanej funkcie **ode45** a vlastnej funkcie RK).*

ÚLOHA:

A. Riešenie v časovej oblasti:

Nájdite celkové riešenie LDR s využitím metódy špeciálnej pravej strany pre DR tvaru :

$y''(t) - 2y'(t) - 3y(t) = 3e^{2t}$ *s počiatočnými podmienkami:* $y(0) = y'(0) = 0$

Riešenie LDR so špeciálnou pravou stranou:

1. Charakteristická rovnica:

$$s^2 - 2s - 3 = 0$$
$$(s-3)*(s+1) = 0$$

Korene charakteristickej rovnice : $s_1 = 3$ $s_2 = -1$

Celkové riešenie vyplývajúce z charakteristickej rovnice:

$$\bar{y} = C_1 e^{-t} + C_2 e^{3t}$$

2. Partikulárne riešenie predpokladáme v tvare $y^* = Ae^{2t}$ nakoľko prava strana DR má tvar :

$$f(t) = 3e^{2t}$$

Derivovaním partikulárneho riešenia a následným výpočtom získame : $A = -1$

3. Celkové riešenie LDR : $y = C_1 e^{-t} + C_2 e^{3t} - e^{2t}$

4. Výpočet integračných konštánt C_1, C_2 z počiatočných podmienok: $C_1 = \frac{1}{4}$, $C_2 = \frac{3}{4}$

5. Celkové analytické riešenie LDR má tvar:

$$y = \frac{1}{4}e^{-t} + \frac{3}{4}e^{3t} - e^{2t}$$

B. Riešenie LDR pomocou Laplaceovej transformácie:

Nájdite celkové riešenie LDR pomocou priamej a inverznej LT:

$y''(t) - 2y'(t) - 3y(t) = 3e^{2t}$ $\qquad$ $PP: y(0) = y'(0) = 0$

1. Zápis LDR do LT: $s^2Y(s) - 2sY(s) - 3Y(s) = \frac{3}{s-2}$

2. Riešenie Y(s) vyjadrenie v Laplaceovej transformácií: $Y(s) = \frac{3}{(s-2)(s-3)(s+1)}$

3. Rozklad prenosovej funkcie Y(s) (riešenia LDR v LT) na parciálne zlomky:

$$\frac{3}{(s-2)(s-3)(s+1)} = \frac{-1}{s-2} + \frac{\frac{3}{4}}{s-3} + \frac{\frac{1}{4}}{s+1}$$

4. Použitím inverznej LT získame výsledné riešenie DR v časovej oblasti:

$$y(t) = -e^{2t} + \frac{3}{4}e^{3t} + \frac{1}{4}e^{-t}$$

C. Riešenie v programovom prostredí MATLAB:

1. Prepis do kanonického substitučného tvaru LDR:

$$a_2y'' + a_1y' + a_0y(t) = u(t) \quad \rightarrow \quad y''(t) - 2y'(t) - 3y(t) = 3e^{2t}$$

Zvolíme substitúciu : $y(t) = x_1$

$$y'(t) = x_1' = x_2$$

$$y''(t) = x_2' = \frac{1}{a_2} * (3e^{2t} - a_1 * x_2 - a_0 * x_1) = 3e^{2t} + 2x_2 + 3x_1$$

2. Riešenie v jazyku MATLAB

difrov.m

```
function xder = difrov(t,x)
%Zápis diferenciálnej rovnice 2.radu pomocou 2rovnic 1.radu
global a0 a1 a2;
xder=[x(2); (3*exp(2*t) - a1*x(2)- a0*x(1))./a2];
return
```

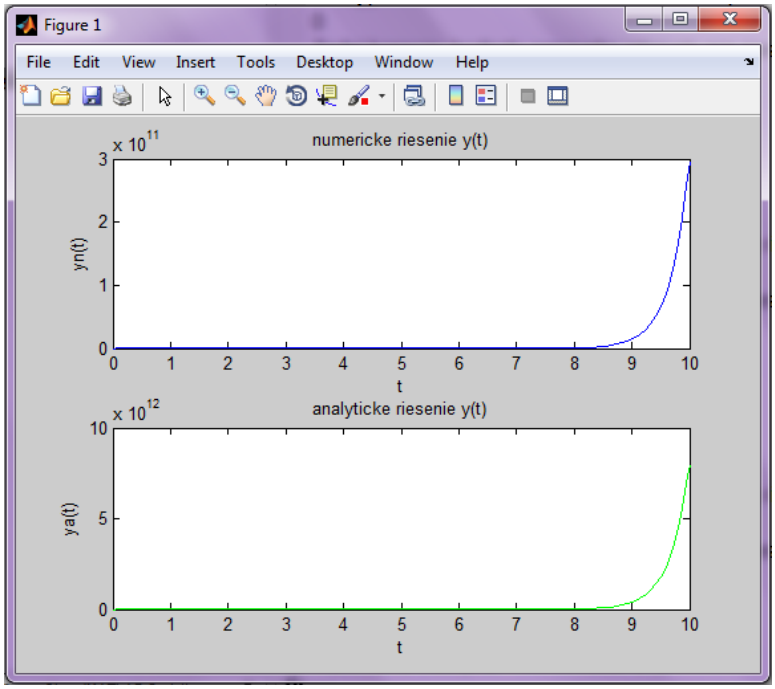
analyt.m

```
%analytické riešenie
function d = analyt(t)
d=(1/4)*exp(-t)+(3/4)*exp(3*t)-exp(2*t);
return
```

Úloha
Naprogramujte hlavný program, ktorého výsledkom bude :

a) funkcia na výpočet riešenia LDR numericky pomocou funkcie ***ode45*** a vlastnej naprogramovanej funkcie R-K,
b) funkcia na výpočet chyby medzi analytickým riešením LDR a jej numerickým riešením získaným metódou R-K (***ode45***/R-K),
c) riešenie LDR numerické a analytické znázornite graficky

Výstupom hlavného programu je nasledujúci obrázok :



Obrázok 4-10 Porovnanie analytického a numerického riešenia LDR v MATLAB-e

5 Modelovanie fyzikálnych systémov v prostredí MATLAB

5.1 Modelovanie a simulácia RLC obvodu – nabíjanie kondenzátora v prostredí MATLAB

Úloha:
Vypočítajte a znázornite priebeh prúdu i(t) a napätia $u_c(t)$ v RLC obvode s parametrami $R = 5\Omega$, $L = 0{,}1H,\ C = 100\mu F$ po pripojení na napätie $U_{DC} = 10V$. Zostavte matematický a simulačný model RLC obvodu pre nabíjanie kondenzátora.

a) modelovanie RLC obvodu – analytická identifikácia → matematický model

Systém RLC obvodu je popísaný diferenciálno- integrálnou rovnicou vyplývajúcou z 2. Kirchhoffovho zákona:

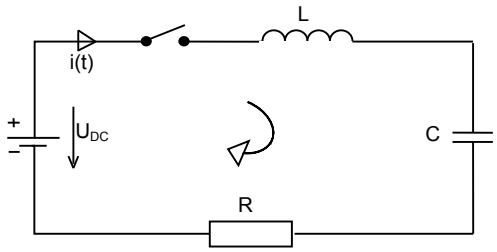
$$L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int i(t)dt = U_{DC} \quad (1)$$

Počiatočné podmienky riešenia uvažujeme:

$$i(0) = 0;\ u_c(0) = 0 \quad (2)$$

pre prúd pretekajúci obvodom platí vzťah:

$$i = C * \frac{du_c}{dt} \quad (3)$$



Obrázok 5-1 Nabíjanie kondenzátora cez technickú cievku

Úpravou integrálu v rovnici (1) získame výslednú diferenciálnu rovnicu popisujúcu nabíjanie kondenzátora, ktorá reprezentuje matematický model RLC obvodu:

$$CL * \frac{d^2u_c(t)}{dt^2} + CR\frac{du_c(t)}{dt} + u_c(t) = U_{DC} \quad (4)$$

$$CL * u_c''(t) + CR * u_c'(t) + u_c(t) = U_{DC} \quad (4a)$$

V prípade, že RLC obvod je pripojený na jednosmerné napätie U_{DC} pri nulových počiatočných podmienkach, potom za stavové veličiny je vhodné zvoliť napätie na kondenzátore a prúd pretekajúci obvodom: u_{Ct} a i_t. Volíme teda substitúciu

$$x_1(t) = u_{Ct},\ \ x_2(t) = i_t$$

a substitučný kanonický tvar systému má tvar

$$x_1' = u_{Ct}' = \frac{x_2}{c} \qquad x_2' = i'(t) = C * u_{Ct}'$$

Dostávame systém dvoch DR 1.rádu:

$$\begin{aligned} x_1'(t) &= \frac{1}{c}x_2(t) \\ x_2' &= \frac{1}{L}(U_{DC} - R * x_2(t) - x_1(t)) \end{aligned} \quad (5)$$

b) Simulácia nabíjania kontenzátora v RLC obvode v jazyku MATLAB

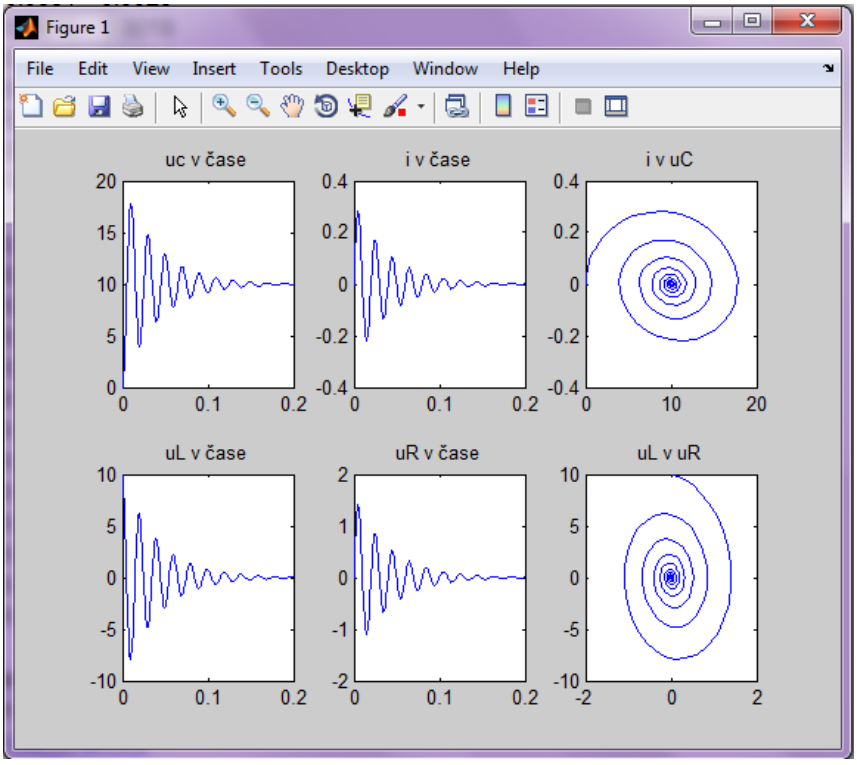
Simulujte v prostredí MATLAB časový priebeh napätia $u_C(t)$ pri prechodovom deji nabíjania kondenzátora, časový priebeh prúdu $i(t)$ v obvode, úbytok napätia na odpore $u_R(t)$ a na cievke $u_L(t)$.

Riešenie v programovom prostredí MATLAB:
funkcia **nabikon.m**

```
% súbor nabikond.m - popis pre nabíjanie
%kondenzátora v RLC obvode
function xdot = nabikond(t,x) % vracia st. derivácie
R=5;
L=0.1;
C=1e-4;
Udc=10;
xdot = [x(2)/C; (1/L)*(Udc-R*x(2)-x(1))];

```

Hlavný program: **RLC45nab.m**

```
% súbor RLC45nab.m na vyhodnotenie súboru nabikond.m
t0=0;  tfin=0.2;
x0=[0 0]';
tol=1e-5; trace=0;

[t,x]=ode45('nabikond',t0,tfin,x0,tol,trace)

subplot(231), plot(t,x(:,1)), title('uc v čase');
subplot(232), plot(t,x(:,2)), title('i v čase');

uC=x(:,1);  iC=x(:,2);
uR=iC*5;  uL=(10-uR-uC);

subplot(233), plot(uC,iC), title('i v uC');
subplot(234), plot(t,uL), title('uL v čase');
subplot(235), plot(t,uR), title('uR v čase');
subplot(236), plot(uR,uL), title('uL v uR');
```



Obrázok 5-2 Grafické znázornenie $u_C(t)$, $u_L(t)$, $u_R(t)$ a i(t) v RLC obvode

5.2 Modelovanie a simulácia RLC obvodu – vybíjanie kondenzátora v prostredí MATLAB

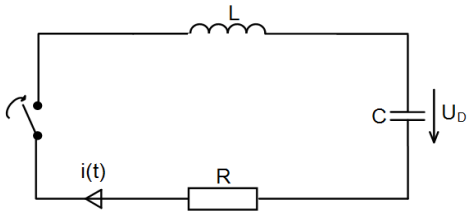
Úloha:
Vypočítajte a znázornite priebeh prúdu a napätia v RLC obvode s parametrami $R = 5\Omega,\ \mathrm{L} = 0{,}1\mathrm{H},\ \mathrm{C} = 100\mu\mathrm{F}$, ak na kondenzátore bolo v čase $t = 0$ napätie $u_{DC} = 10V$. Zostavte matematický a simulačný model RLC obvodu pre výbíjanie kontenzátora.

a) Modelovanie RLC obvodu – analytická identifikácia → matematický model

Uvažovaný RLC obvod vieme popísať na základe 2. Kirchhoffovho zákona lineárnou diferenciálnou rovnicou 2: rádu:

$$L\frac{d^2i(t)}{dt^2} + R\frac{di(t)}{dt} + \frac{1}{C}i(t) = 0,$$

pričom pre počiatočné podmienky platí: $i(0) = 0\ , \frac{di(0)}{dt} = \frac{u_{DC}}{L}$



Obrázok 5-3 Vybíjanie kondenzátora cez technickú cievku

Pre prepis lineárnej diferenciálnej rovnice popisujúcej systém do substitučného kanonického tvaru zvolíme substitúciu $x_1(t) = i(t)$.

$$x_1'(t) = \frac{di(t)}{dt};$$

$$x_2'(t) = \frac{1}{L}\left(-Rx_2(t) - \frac{1}{C}x_1(t)\right),$$

s uvažovaním počiatočných podmienok $i(0) = 0\ , \frac{di(0)}{dt} = \frac{u_{DC}}{L}$

b) Simulácia vybíjania kondenzátora RLC obvodu v jazyku MATLAB

Simulujte v prostredí MATLAB časový priebeh napätia $u_C(t)$ pri vybíjaní kondenzátora, časový priebeh prúdu $i(t)$ tečúceho v obvode, úbytok napätia na odpore $u_R(t)$ a na cievke $u_L(t)$.

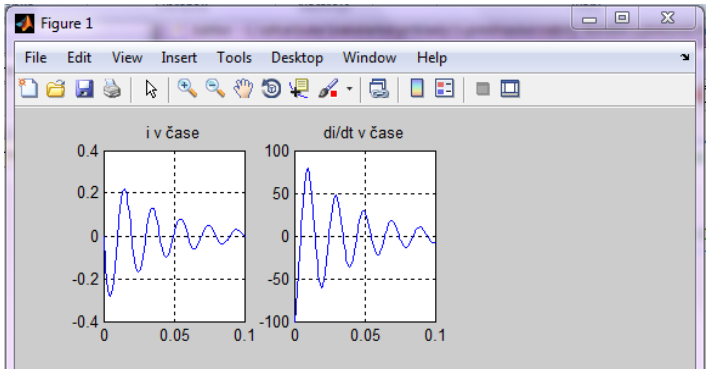
Riešenie v programovom prostredí MATLAB:

funkcia: **vybikond.m**

```
1   %vybikond - vybíjanie kondenzátora v obvode s cievkou a odporom
2   function xdot=vybikond(t,x); % vracia stavové ch.
3   R=5;  L=0.1; C=1e-4;
4   xdot=[x(2); (1/L)*(-R*x(2)-(1/C)*x(1))]; %maticový zápis oboch rovníc
```

hlavný program:

```
1 -  t0=0; tfin=0.1;
2 -  x0=[0; -100];
3 -  tol=1e-3;
4
5 -  [t,x]=ode23('vybikond',t0,tfin,x0,tol);
6
7 -  subplot(231), plot(t,x(:,1)), grid, title('i v čase');
8 -  subplot(232), plot(t,x(:,2)), grid, title('di/dt v čase');
9
10 - ic=x(:,1);  di=x(:,2);
11 - uR=ic*5;  uL=di*0.1;
12 - uC=(-uR-uL);
```



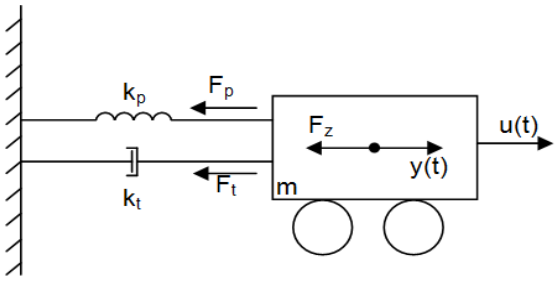
Obrázok 5-4 Vybíjanie kondenzátora – časový priebeh prúdu tečúceho v obvode a jeho derivácia

5.3 Modelovanie a simulácia systému "pružina-tlmič"

Úloha:
Zostavte matematický a simulačný model systému „pružina-tlmič".

a) Modelovanie systému „pružina – tlmič" – analytická identifikácia

$u(t)$ - budiaca sila
$F_p(t)$ – direktívna sila pružiny
$F_t(t)$ – sila viskózneho trenia
$F_z(t)$ - sila zotrvačnosti
$y(t)$ – poloha vozíka
$y'(t)$ - rýchlosť vozíka



Uvažujeme, že pre jednotlivé sily znázornené na obrázku platí :

$$F_z(t) = m * \frac{dy^2(t)}{dt^2} \qquad F_t(t) = k_t * \frac{dy(t)}{dt} \qquad F_p(t) = k_p * y(t)$$

Na základe zákona o rovnováhe síl (súčet síl pôsobiacich na teleso v ťažisku je rovný nule), t.j.

$$\sum_{i=1}^{n} F_i = 0$$

môžeme napísať :

$$F_z(t) + F_p(t) + F_t(t) = u(t)$$

Po dosadení za jednotlivé sily $F_Z(t)$, $F_T(t)$, $F_P(t)$ do zákona o rovnováhe síl dostavame lineárnu diferenciálnu rovnicu s konštatnými koeficientami, ktorá predstavuje matematický model systému „pružina-tlmič":

$$m * \frac{d^2y(t)}{dt} + k_t * \frac{dy(t)}{dt} + k_p * y(t) = u(t)$$

Pre prepis LDR do substitučného kanonického tvaru vykonáme substitúciu: $y(t) = x_1(t)$;

$$x_1'(t) = x_2(t)$$

$$x_2'(t) = \frac{u(t)}{m} - \frac{k_p}{m} * x_1(t) - \frac{k_t}{m} x_2(t)$$

b) Simulácia časových priebehov polohy y(t),rýchlosti y´(t) a zrýchlenia y´´(t) systému „pružina – tlmič" pri pôsobení vstupnej sily u(t)

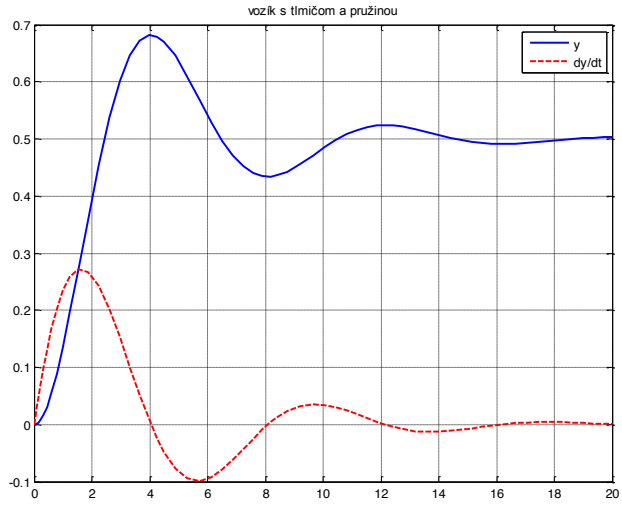
Simuláciu vykonajte pre nasledujúce parametre: *m=30kg* ,$k_t = 20$,$k_p = 15$, $u(t) = 10N$

funkcia **vozik.m**

```
1    % funkcia 'vozik.m' - vozík s tlmičom a pružinou
2    function xdot=vozik(t,x)
3    global u m kt kp
4    xdot=[x(2); u/m-(kp*x(1))/m-(kt*x(2))/m];
```

hlavný program

```
1 -   t0=0;
2 -   tfin=20;
3 -   global u m kt kp
4 -   u=input('zadaj vonkajsiu silu u = ');
5 -   m=input('zadaj hmotnost vozika m = ');
6 -   kp=input('zadaj konstantu pruziny kp = ');
7 -   kt=input('zadaj konstantu tlmica kt = ');
8 -   x0=input('zadaj pociatocne podmienky x0 = ');
9
10 -  [t,x]=ode23('vozik', t0,tfin, x0);
11
12 -  plot(t,x(:,1),'k-',t,x(:,2),'k--')
13    % alebo plot(t,x) pricom x je dvojrozmerny vektor
```



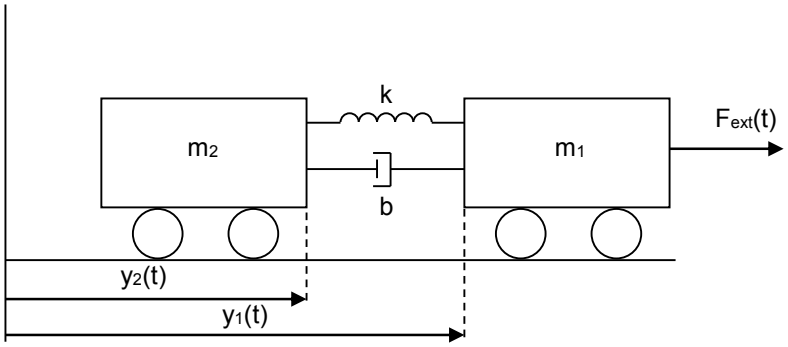
Obrázok 5-5 Simulácia dynamického systému „vozik-pružina“
časove priebehy polohy y(t) a rýchlosti y´(t)

Pozn. Každá funkcia v simulačnom jazyku MATLAB definovaná ako m-file má svoje vlastné lokálne premenné, ktoré sú neviditelné pre ostatné funkcie. Ak chceme, aby lokálne premenné v používanej funkcii boli viditelné aj mimo funkcie, musíme ich deklarovať pomocou príkazu ***„global“***

5.4 Modelovanie a simulácia systému "dva vozíky v interakcii" v jazyku MATLAB

Úloha:
Zostavte matematický a simulačný model systému „dva vozíky v interakcii".

Vozová súprava (dva vozíky v interakcii) predstavuje model dynamického správania sa dvoch telies prepojených pružinou a tlmičom. Tento typ systému sa často používa v kybernetike pri modelovaní kmitavých systémov.



Obrázok 5-6 Fyzikálny systém, dva vozíky v interakcii

a) Modelovanie systému „dva vozíky v interakcii" – analytická identifikácia

Vstup do systému:
$F_{ext}(t)$ – sila pôsobiaca na vozík s hmotnosťou m_1 a následne aj na vozík s hmotnosťou m_2

Výstup zo systému:
$y(t)$- vzdialenosť medzi vozíkmi:
$y(t) = y_1(t) - y_2(t)$

Systém je popísaný dvoma lineárnymi diferenciálnymi rovnicami 2. rádu:

1. Prvá LDR popisuje sily pôsobiace na vozík s hmotnosťou m_1 :

$$m_1 y_1''(t) = -k[y_1(t) - y_2(t)] - b[y_1'(t) - y_2'(t)] + F_{ext}(t)$$

2. Druhá LDR popisuje pôsobenie síl na vozík s hmotnosťou m_2:

$$m_2 y_2''(t) = k[y_1(t) - y_2(t)] + b[y_1'(t) - y_2'(t)]$$

pričom parameter *b* predstavuje koeficient tlmenia a parameter *k* predstavuje tuhosť pružiny.

Model systému v substitučnom kanonickom tvare:

$$x_1'(t) = x_2(t)$$

$$x_2'(t) = -\frac{k}{m_1}x_1(t) + \frac{k}{m_1}x_3(t) - \frac{b}{m_1}x_2(t) + \frac{b}{m_1}x_4(t) + \frac{1}{m_1}F_{ext}(t)$$

$$x_3'(t) = x_4(t)$$

$$x_4'(t) = \frac{k}{m_2}x_1(t) - \frac{k}{m_2}x_3(t) + \frac{b}{m_2}x_2(t) - \frac{b}{m_2}x_4(t)$$

Systém štyroch DR prvého rádu predstavuje matematický model systému dva vozíky v interakcii v substitučnom kanonickom tvare, ktorý je takto pripravený na programovú implmentáciu do simulačného jazyka MATLAB

Výstup systému *y(t)* odpovedá rozdiel vzdialeností jednotlivých vozíkov od zvislej osi:

$y(t) = x_1(t) - x_3(t)$→rovnica pre výstup

b) Simuláciou v jazyku MATLAB zistite časový priebeh vzdialenosti medzi dvoma vozíkmi *y(t)* pri pôsobení vonkajšej sily $F_{ext}(t)$ na daný systém. Simuláciu vykonajte pre vhodnú voľbu parametrov systému *m1, m2, b, k* a budiaceho vstupného signálu vstupnej sily *F.*

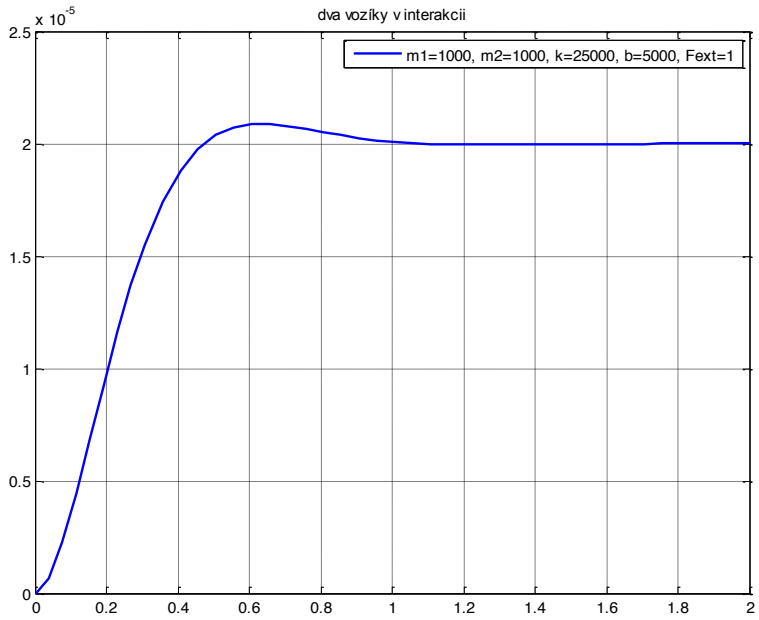
Riešenie v programovom prostredí MATLAB:

funkcia : vozíky.m

```
function xder=voziky(t,x)
global m1 b k F m2

xder=[x(2); (-/m1)*x(1)+(k/m1)*x(3)(b/m1)*x(2)+(b/m1)*x(4)+(1/m1)*F;
x(4);(k/m2)*x(1)-(k/m2)*x(3)+(b/m2)*x(2)-(b/m2)*x(4)]

return
```

Hlavný program

```
global m1 b k F m2   %parametre systému

m1=1000;
m2=1000;
b=5000;
k=25000;
F=1;

[t,y]=ode45('voziky',[0 2],[0 0])

plot(t,y(:,1))
grid on;
title('dva vozíky v interakcii')
legend('m1=1000, m2=1000, k=25000, b=5000, Fext=1')
```

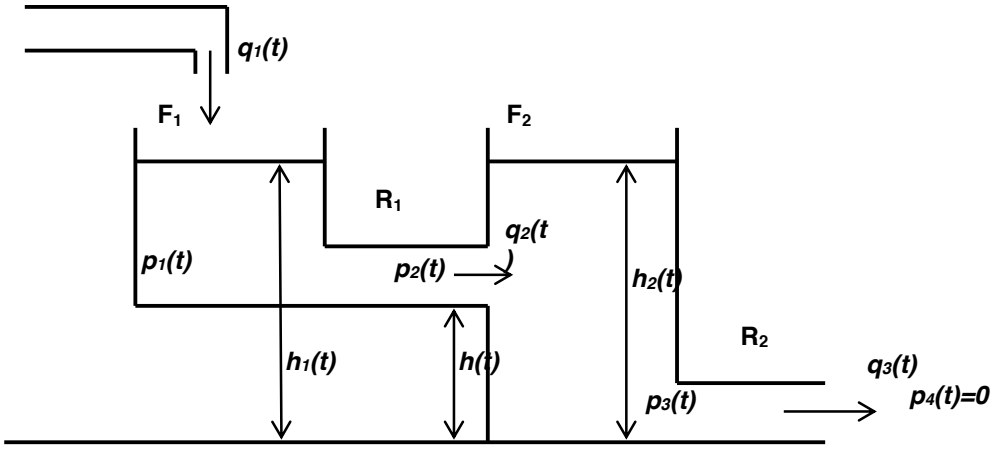


Obrázok 5-7 Simulácia vzdialenosti medzi dvoma vozíkmi v interakcii y(t) pri pôsobení sily $F_{ext}(t)$

5.5 Modelovanie a simulácia hydraulického systému - dve nádoby v interakcii v jazyku MATLAB

Úloha:
Zostavte matematický model hydraulického systému dvoch nádob v interakcii, pričom uvažujete, že vstupom do systému je prítok do prvej nádoby $q_1(t)$ a výstup hydraulického systému je výtok z druhej nádoby $q_3(t)$

Na riešenie danej úlohy uvažujeme hydraulický systém pozostávajúci z dvoch vodorovne prepojených nádob s prítokom aj odtokom v dvojrozmernom súradnicovom systéme.



Obrázok 5-8 Hydraulický systém – dve nádoby v interakcii

a) Modelovanie hydraulického systému – matematický model

Hydrostaticky tlak v jednotlivých nádobách a potrubí je popísaný rovnicou

$$p = \rho g h$$

Linearizovaná závislosť prietoku v prietokovej trubici z nádrže 1 do nádrže 2 je popísaná rovnicou:

$$q_2(t) = \frac{1}{R_1}\left(p_1(t) - p_2(t)\right)$$

a linearizovaná závislosť prietoku v odtokovej trubici z nádrže 2. je popísaná rovnicou:

$$q_3(t) = \frac{1}{R_2}\left(p_3(t) - p_4(t)\right)$$

Diferenciálne rovnice opisujúce dynamiku, t.j. zmenu objemu kvapaliny v obidvoch nádobách v závislosti od vstupného a výstupného prietoku z konkrétnej nádoby majú tvar:

$$F_1 \frac{dh_1(t)}{dt} = q_1(t) - q_2(t)$$

$$F_2 \frac{dh_2(t)}{dt} = q_2(t) - q_3(t)$$

Úpravou týchto rovníc získame výslednú diferenciálnu rovnicu pre matematický model hydraulického systému kde $q_3(t)$ je hľadané riešenie LDR pri budení $u = q_1(t)$:

$$\frac{F_1 R_2}{\rho g}\frac{dq_3(t)}{dt}+\frac{F_1 R_1 F_2 R_2}{(\rho g)^2}\frac{d^2 q_3(t)}{dt^2}+\frac{F_1 R_1}{\rho g}\frac{dq_3(t)}{dt}+\frac{F_2 R_2}{\rho g}\frac{dq_3(t)}{dt}+q_3(t)=\ q_1(t)$$

Pre prehľadnejší zápis LDR zvolímenasledujúce označenie:

$$T_1=\frac{F_1 R_1}{\rho q} \qquad T_2=\frac{F_2 R_2}{\rho q} \qquad T_3=\frac{F_1 R_2}{\rho q}$$

Lineárna diferenciálna rovnica popisujúca prechodový dej odtoku z druhej nádoby má tvar :

$$T_1 T_2 \ddot{q_3}(t)+\ (T_1+T_2+T_3)\dot{q_3}(t)+q_3(t)=q_1(t)$$

Prepis do substitučného kanonického tvaru :

ak zvolíme substitúciu $x_1(t) = q_3(t)$ dostaneme

$$x_1\dot{}(t)=\frac{1}{T_1 T_2}x_1(t)$$

$$x_2\dot{}(t)=\frac{-1}{T_1 T_2}x_1(t)+\frac{-(T_1+T_2+T_3)}{T_1 T_2}x_2(t)+\frac{q_1(t)}{T_1 T_2}$$

b) **Úloha na samostatné riešenie**

Zostavte simulačný model HS v jazyku MATLAB s využitím funkcie *„ode45“,* ktorého výsledkom bude časový priebeh fyzikálnej veličiny *„$q_3(t)$“* čo je výtok z druhej nádoby pri uvažovanom prítoku *„$q_1(t) = 1\ m^3s^{-1}$*

Parametre simulačného výpočtu:
Počiatočné podmienky: $q_3(0) = 0$, $q\dot{}_3(0) = 0$
prierezy nádob :_ $F_1 = 0.25m^2$ $F_2 = 0.2m^2$,
odpor potrubia : $R_1 = 20kPa$ $R_2 = 20kPa$
hustota kvapaliny : $\rho = 998\ kgm^{-3}$
gravitačné zrýchlenie : $g = 9.81ms^{-2}$
budiaci signál – prítok $u(t) = q_1(t)$, $q_1 = 1m^3s^{-1}$

6 Modelovanie a simulácia nelineárnych fyzikálnych systémov v prostredí MATLAB

6.1 Modelovanie a simulácia matematického kyvadla v prostredí MATLAB

Úloha: Zostavte matematický a simulačný model pre NDS matematického kyvadla.

Uvažujeme matematické kyvadlo zobrazené na obrázku, ktoré je tvorené závažím o hmotnosti m, zavesenom na šnúre o dĺžke l, ktorej hmotnosť môžeme zanedbať.

a) Modelovanie nelineárneho DR – matematické kyvadlo

Popis parametrov matematického kyvadla na obrázku:

m – hmotnosť závažia

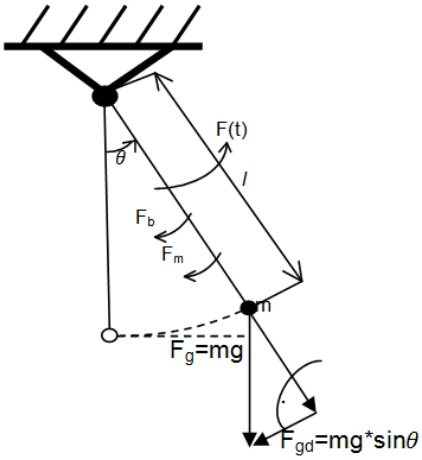
l - dĺžka závesu

g - gravitačná konštanta

B - koeficient tlmenia

Význam skúmanej fyzikálnej veličiny:

θ- uhol vychýlenia kyvadla



Význam síl v dynamickom modeli matematického kyvadla je nasledovný

$F(t)$- budiaca sila (vyvedie dynamický systém z rovnovážnej polohy)
F_m- sila zotrvačnosti
F_b- brzdná sila (odpor vzduchu + tangensiálna zložka tiažovej sily F_{gd})
F_g - tiažová sila, $F_g = m.g$

Na zostavenie matematického modelu dynamického systému matematické kyvadlo použijeme d'Alembertov princíp:

$$F_m + F_b + F_{gd} - F(t) = 0$$

kde

- $F_m = m * l * \frac{d^2\theta(t)}{dt^2} = m * l * \theta''(t)$ - výpočet sily zotrvačnosti
- $F_b = B * l * \frac{d\theta}{dt} = B * l * \theta'(t)$ - výpočet brzdiacej sily pre obvodovú rýchlosť kyvadla
- $F_{gd} = m * g * sin\theta$ - tangensiálna zložka tiažovej sily
- $F(t) = \frac{M(t)}{l}$ - externá budiaca sila
- $M(t)$ - vonkajší moment – vstup do systému
- $\theta(t)$ - uhol vychýlenia kyvadla – výstup systému

Dosadením jednotlivých síl do d'Alembertov princíp dostaneme nelineárnu diferenciálnu rovnicu 2. rádu s pravou stranou, ktorá reprezentuje matematický model kyvadla

$$m * l * \theta''(t) + B * l * \theta'(t) + m * g * sin\,\theta = \frac{M(t)}{l}$$

Po normovaní uvedenej DR dostanem NDR

$$\theta''(t) + \frac{B}{m}\theta'(t) + \frac{g}{l}\sin\theta = \frac{1}{ml^2}M(t)$$

Prepis do substitučného kanonického tvaru

Substitúcia: $x_1(t) = \theta(t)$,

$$x_1'(t) = x_2(t)$$

$$x_2'(t) = -\frac{B}{m}x_2(t) - \frac{q}{l} * \sin(x_1(t)) + \frac{1}{ml^2}M(t)$$

b) Simuláciou v MATLAB-e zistite časový priebeh výchylky kyvadla *θ(t)* a jeho rýchlosti *θ´(t)* bez pôsobenia vstupného signálu (počiatočné podmienky : $x_1(0) = 1$, $x_1'(0) = 0$)a pri pôsobení vstupného budiaceho signálu *M(t)*

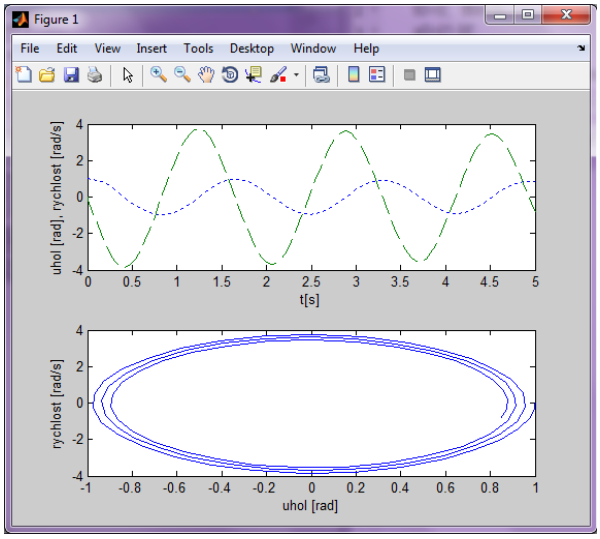
Riešenie v programovom prostredí MATLAB:

funkcia : **kyvadlo.m**

```
1     % funkcia 'kyvadlo.m' - kmitanie vychyleneho kyvadla s tlmenim
2
3     function xdot=kyvadlo(t,x)
4 -     l=0.6; B=1e-2;g=9.81;m=0.2038;
5 -     xdot=[x(2);-B/m*x(2)-g/l*sin(x(1))];
```

hlavný program : **kyv23sol.m**

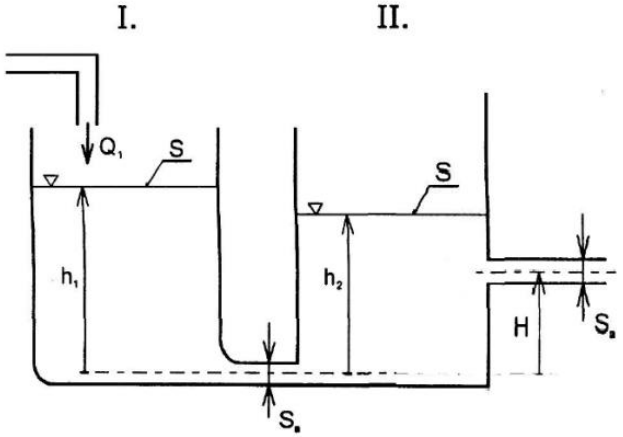
```
1     % subor 'kyv23sol.m' na riesenie DR kyvadla
2 -   t0=0;   tfin=5;
3 -   x0=[1 0]';
4 -   tol=1e-3; trace=0;
5 -   [t,x]=ode45('kyvadlo',t0,tfin,x0,tol,trace);
6 -   subplot(211), plot(t,x(:,1),':',t,x(:,2),'--');
7 -   xlabel('t[s]'), ylabel('uhol [rad], rychlost [rad/s]');
8 -   subplot(212), plot(x(:,1),x(:,2))
9 -   title ...
```



Obrázok 6-1 Simulácia pohybu matematického kyvadla – časový priebeh výchylky kyvadla *θ(t) , rýchlosti pohybu θ´(t) pri M(t) = 0 a PP = [1,0]*

6.2 Modelovanie a simulácia hydraulického systému v jazyku MATLAB

Úloha:
Zostavte matematický a simulačný model hydraulického systému dvoch nádob v interakcii.



a) Zostavenie matematického modelu hydraulického systému

Sledovanou veličinou hydraulického systému bude zmena výšky hladiny kvapaliny v prvej nádobe $h_1(t)$ od prítoku $Q_1(t)$do prvej nádoby.
Zmenu výšky hladiny vieme popísať pomocou DR v tvare

$$S * \frac{dh_1(t)}{dt} = Q_1 - S_0 * v_1$$

Pre výtokovú rýchlosť kvapaliny z nádoby platí Toricelliho vzťah $v = \sqrt{2g(h_1(t) - h_2(t))}$.

Teda zmena výšky hladiny vody $h_1(t)$ môže byť vyjadrená DR v prvej nádobe bude popísaná diferenciálnou rovnicou

$$h_1'(t) = \frac{1}{S} * Q - \frac{S_0}{S}\sqrt{2g(h_1(t) - h_2(t))}.$$

Podobným spôsobom vieme popísať aj zmenu výšky hladiny vody $h_2(t)$ v druhej nádobe:

$$h_2'(t) = \frac{S_0}{S}\sqrt{2g(h_1(t) - h_2(t))} - \frac{S_0}{S}\sqrt{2g(h_2(t) - H)}$$

b) Úloha na samostatné riešenie: Naprogramujte simulačný model hydraulického systému pre získanie časových priebehov výšky hladiny kvapaliny $h_1(t)$ a $h_2(t)$ pri zvolených počiatočných podmienkach a vstupnom prítoku $Q_1(t)$

Simuláciu vykonajte pre zvolených počiatočných podmienkach: $h_1(0) = 1{,}2$ a $h_2(0) = 0{,}7$.
a parametroch:

$$S = 0{,}25m^2 \qquad S_0 = 0{,}01m^2 \qquad H = 0{,}2m$$

Pri vstupnom prítoku :

$$Q = 0{,}025m^3s^{-1}$$

6.3 Príklad na simuláciu nelineárneho dynamického systému van der Polov oscilátor

Zostavte simulačný model nelineárneho dynamického systému van der Polovho oscilátora v jazyku MATLAB na základe matematického modelu, ktorý je vyjadrený nelineárnou DR:

$$y''(t) + A\left(1 - y^2(t)\right)y'(t) - y(t) = u(t)$$

Nelineárna diferenciálna rovnica je druhého rádu potrebujeme,kvôli vytvoreniu simulačného modelu zostaviťsystém NDR prvého rádu a to prepisom NDR oscilátora do substitučného kanonického tvaru.

Prepis do substitučného kanonického tvaru:

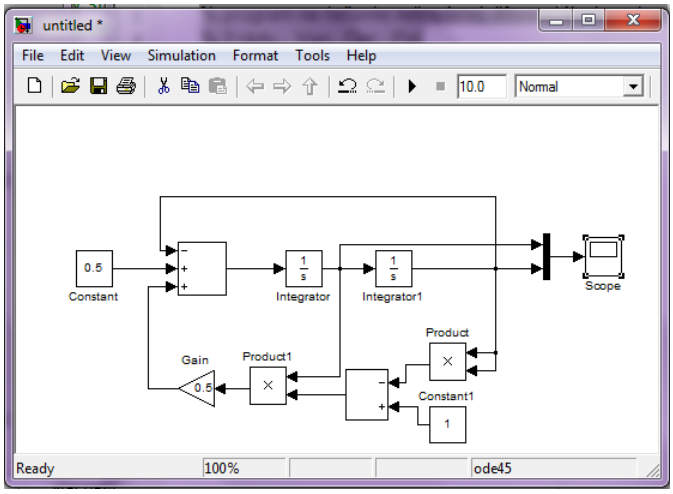
Substitúcia: $y(t) = x_1(t)$

$$x_1'(t) = x_2(t)$$

$$x_2'(t) = -A * \left(1 - x_1^2(t)\right)x_2(t) + x_1(t) + u(t)$$

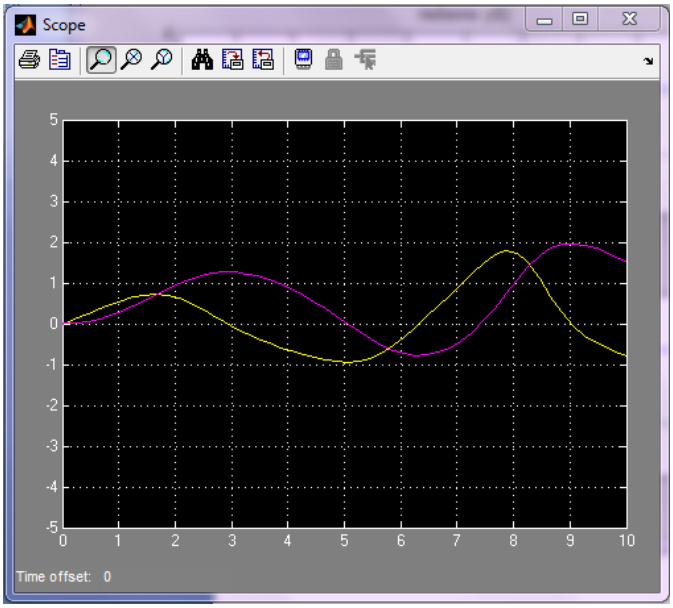
Zostavenie simulačného modelu v programovom prostredí Simulink :

Simulačný model van der-Polovho oscilátora zostavíme v programovom prostredí Simulink, metódou postupného znižovania radu derivácie.
Pre simuláciu modelu použijeme voľbu parametra $A = 0{,}5$ a vstupný budiaci signál $u(t) = 0{,}5$.



Obrázok 6-2 Programová Schéma simulačného modelu – Vander –Polov oscilátor

Pred spustením simulácie užívateľ musí zvoliť metódu pre riešenie NDR (solver: napr. metóda R-K), je nutné zvoliť integračný krok pre výpočet, počiatočné podmienky a začiatočný a koncový čas pre riešenie simulácie.



Obrázok 6-3 Grafické znazornenie riešenia $x_1(t)$ a $x_2(t)$ pre NDR Vander-Polov oscilatora

6.4 Príklad na simuláciu nelineárneho dynamického systému "dravec-korisť"

Úloha:
Zostavte simulačný model v prostredí MATLAB pre systém "dravec-korisť" (systém Lotka-Volterra) s cieľom študovať efekt interakcie koeficientov α,β , ktorého matematický model je tvorený dvoma NDR prvého rádu

$$y_1'(t) = y_1(t) - \alpha . y_1(t) . y_2(t)$$

$$y_2'(t) = -y_2(t) + \beta . y_1(t) . y_2(t)$$

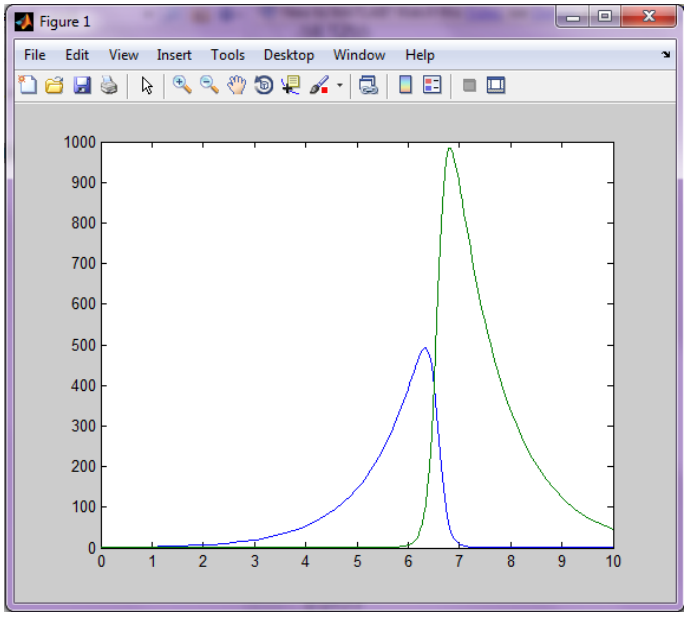
Simulačný model systému Lotka-Volterra v jazyku MATLAB

funkcia: **lotka.m**

```
1     % funkcia 'lotka.m' - interakcia koeficientov alpha a beta
2     function yp=lotka(t,y)
3     global ALPHA BETA
4     yp = [y(1)-ALPHA*y(1)*y(2); -y(2)+BETA*y(1)*y(2)]
```

hlavná funkcia

```
1  global ALPHA BETA
2  ALPHA = 0.01
3  BETA = 0.02
4  [t,y]=ode23('lotka',0,10,[1,1]);
5  plot(t,y)
```



Obrázok 6-4 Časové priebehy $y_1(t)$ a $y_2(t)$ simulačného modelu Lotka-Volterra pri α = 0,01 , β = 0,02

6.5 Zadanie č.3: Riešenie nelineárnej diferenciálnej rovnice v prostredí MATLAB s využitím naprogramovanej metódy Runge-Kutta 4. rádu

ZADANIE:

Riešenie nelineárnej diferenciálnej rovnice (NDR) numericky so zvolenou numerickou technikou a algoritmicky v programovom prostredí MATLAB.

OBSAH ZADANIA:
Zvolenú nelineárnu diferenciálnu rovnicu s konštantnými koeficientmi a s počiatočnými podmienkami riešte v programovom prostredí MATLAB.

Majme nelineárnu diferenciálnu rovnicu 2. rádu tvaru:

$$5y''(t) + 4cos(y'(t))y(t) - \sqrt{y(t)} = sin(2\pi)$$

Prepíšeme do substitučného kanonického tvaru pri substitúcii: $y(t) = x_1(t)$:

$$x_1' = x_2$$

$$x_2' = \frac{sin(2\pi) - 4cos(x_2)x_1 - \sqrt{x_1}}{5}$$

Riešenie nelineárnej diferenciálnej rovnice v programovom prostredí MATLAB s využitím funkcie ode45 a vlastnej funkcie R-K:

funkcia **NDR.m**

```
function xder=NDR(t,x)

xder=[x(2);(sin(2*pi)-4*cos(x(2))*x(1)-sqrt(x(1)))/5]

return
```

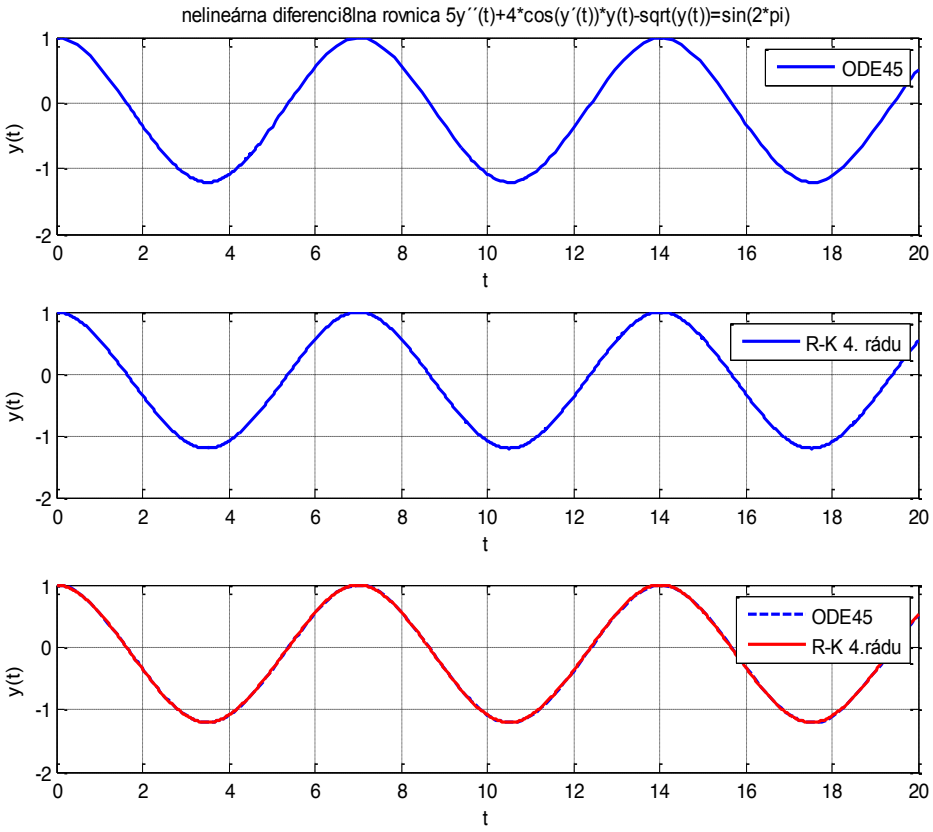
Časť kódu v jazyku MATLAB pre vlastnú funkciu Runge-Kutta 4.rádu

funkcia **rk.m**

```
function [t,y]=rk(f,T,PP)

⋮

for i=1:length(t)
    K1=h.*f(x(i),y(i,:));
    K2=h.*f(x(i)+h./2,y(i,:)+K1'./2);
    K3=h.*f(x(i)+h./2,y(i,:)+K2'./2);
    K4=h.*f(x(i)+h,y(i,:)+K3');

if i~=(length(t))
        y(i+1,:)=y(i,:)+(K1'+2*(K2'+K3')+K4')/6;
end;
end;
```

Časť kódu hlavného programu

```
% hlavnýprogram pre riešenie nelineárnej diferenciálnej rovnice pomocou funkcie
ODE45 a vlastnou funkciou R-K 4.  rádu
...

[t,y]=ode45('NDR',[0 20],[1 0])
[t1,y1]=rk(@NDR,[0 20], [1 0])
subplot(311)
plot(t,y(:,1))
...
```



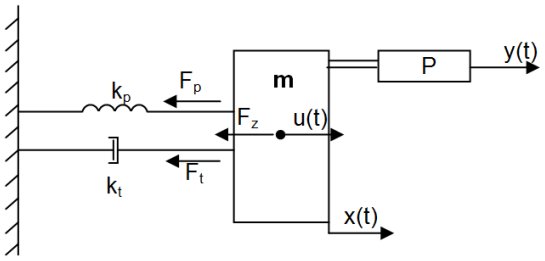
Obrázok 6-5 Riešenie NDR s využitím vstavanej funkcie ode45 a vlastnej funkcie rk 4. rádu a ich vzajomné porovnanie

7 Modelovanie a analýza lineárnych dynamických systémov s využitím funkcií Control System Toolbox-u

- Aplikačná knižnica **Control System Toolbox** je zameraná na riešenie úloh analýzy a syntézy lineárnych časovo-invariantných dynamických systémov (Linear-Time-Invariant - LTI)
- Základným predpokladom použitia jednotlivých modulov (funkcií) je znalosť matematických modelov riadených systémov opísaných v stavovom priestore (v časovej oblasti); pomocou prenosových funkcií (v s- oblasti)
- Riešenie úloh pomocou funkcií **Control System Toolbox**-u môžeme riešiť úlohy tvorby modelov rôznymi spôsobmi zápisu a konverzie medzi matematickými modelmi pre systémy SISO v prostredí MATLAB
- **Control System Toolbox** využíva modely v tvare prenosových funkcií alebo systémov diferenciálnych rovníc v stavovom priestore
- Obidva typy modelov môžu byť vyjadrené v spojitom tvare (continuous time) a v diskrétnom tvare (discrete time)

7.1 Spôsoby zadefinovania lineárnych dynamických systémov s využitím funkcií Control Toolboxu

Rôzne spôsoby modelovania LDS budeme ilustrovať na príklade modelu DS – „pružina- tlmič“



$$F_z(t) = m * \frac{dx^2(t)}{dt^2}$$

$$F_t(t) = k_t * \frac{dx(t)}{dt}$$

$$F_p(t) = k_p * x(t)$$

$$y(t) = p * x(t)$$

$$\sum_{i=1}^{n} F_i = 0$$

Na zostavenie matematického modelu dynamického systému využijeme d´Alembertov princíp, výsledkom ktorého je lineárna DR 2.rádu.

$$m.\frac{dx^2(t)}{dt^2} + k_t.\frac{dx(t)}{dt} + k_p.x(t) = u(t)$$

$$m.x''(t) + k_t.x'(t) + k_p.x(t) = u(t)$$

Matematický model LDS prepíšeme do Laplaceovej transformácie

$$(ms^2 + k_t s + k_p)X(s) = U(s)$$

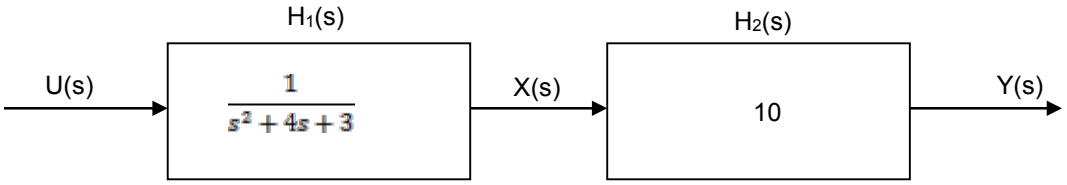
Vytvorenie prenosovej funkcie $H_1(s)$:

$$H_1(s) = \frac{X(s)}{U(s)} = \frac{1}{ms^2 + k_t s + k_p}$$

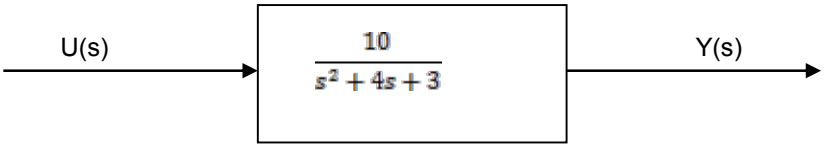
Bloková schéma poukazuje na súvislosť vstupných a výstupných premenných pomocou prenosovej funkcie $H_1(s)$ a $H_2(s)$.

Predpokladajme, že v modeli „pružina-tlmič“ sú zvolené parametre nasledovne :

$$m = 1;\ k_t = 3;\ k_p = 4;\ p = 10$$



$H = H_1(s).H_2(s)$



systém

Rôzne spôsoby zápisu prenosovej funkcie *H(s):*

a) $H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$, $m < n$ - racionálne lomená funkcia

b) $H(s) = K * \frac{(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_n)}$, - póly/nuly (z/p/k)

c) $H(s) = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \cdots + \frac{r_n}{s-p_n} + K$ - reziduálna forma

1) Zadávanie prenosovej funkcie modelu *H(s)* v tvare racionálne / lomenej funkcie (RLF)

- $sys = tf(num, den)$

V simulačnom jazyku MATLAB sa pre vytvorenie prenosovej funkcie v tvare RLF požíva funkciu ***tf***

$$H(s) = \frac{B(s)}{A(s)}$$

kde,

B(s) – num(čitateľ prenosovej funkcie) , *A(s)* – den(menovateľ prenosovej funkcie)

PRÍKLAD 1

```
>> num=[1,2];
>> denum=[1,2,1];
>> sys=tf(num,denum)
```

Výsledok :

Transfer function:

```
     s + 2
-------------
s^2 + 2 s + 1
```

2) Zadávanie prenosovej funkcie modelu *H(s)* v tvare *nuly/póly/zosilnenie (z/p/k)*

- $sys = zpk(z, p, k)$
  kde
  z – nuly
  p – póly
  $k = \frac{b_m}{a_n}$ *- zosilnenie*

PRÍKLAD 2

```
>> z=[-2];
>> p=[-1,-1];
>> k=[1];
>> sys=zpk(z,p,k)
```

Výsledok:

```
Zero/pole/gain:
(s+2)
-------
(s+1)^2
```

3) Zadávanie prenosovej funkcie modelu *H(s)* v reziduálnom tvare

- $[r, p, k] = residue(num, denum)$

PRÍKLAD 3

Rozložte na parciálne zlomky obrazový prenos $H(s) = \frac{s+2}{s^2+2s+1}$

```
>> num=[1,2];
>> denum=[1,2,1];
>> [r,p,k]=residue(num,denum)
```

```
r =
     1
     1
p =
    -1
    -1
k =
     []
```

$$H(s) = \frac{B(s)}{A(s)} = \frac{1}{s+1} + \frac{1}{s+1}$$

Reprezentácia LDS v stavovom priestore (STATE SPACE MODELS**)** predstavuje významný prístup k modelovaniu tohto druhu systémov.

Majme diferenciálnu rovnicu, ktorá vyjadruje model systému

„pružina – tlmič“ *:* $mx''(t) + k_p x'(t) + k_t x(t) = u(t)$

Vytvorte substitučný kanonický tvar v stavovom priestore, pričom uvažujeme substitúciu : *x(t)* = $x_1(t)$

$$\dot{x_1}(t) = x_2(t)$$

$$\dot{x_2}(t) = -\frac{k_t}{m}x_1(t) - \frac{k_p}{m}x_2(t) + \frac{1}{m}u(t)$$

Rovnica pre meraný výstup má tvar

$$y = g(x,u) = 10x_1(t)$$

Konkrétny tvar LDS v stavom priestore pri zadaných parametroch:

$$\dot{x_1}(t) = x_2(t)$$
$$\dot{x_2}(t) = -3x_1 - 4x_2 + u(t)$$

- Systém DR v substitučnom kanonickom tvare prepíšeme do stavového priestoru, ktorý má tvar*:*

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$
$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

- Prepis systému pružina-tlmič do stavového popisu

$$\begin{bmatrix}\dot{x_1}(t)\\ \dot{x_2}(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -3 & -4\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}u$$

$$y(t) = [10 \quad 0]\begin{bmatrix}x_1\\ x_2\end{bmatrix} + [0]u$$

4) Zadávanie dynamického systému pomocou stavového opisu a matíc *A, B, C, D*

Ak je stavový popis DS definovaný *:*

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$
$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)'$$

potom na zápis systému do stavového priestoru v simulačnom jazyku MATLAB používame funkciu ***„ss“***

- $\boldsymbol{sys = ss(A,B,C,D)}$

PRÍKLAD 4

Majme zadané matice systému v stavovom priestore : $\boldsymbol{A} = \begin{bmatrix}-2 & -1\\ 1 & 0\end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix}1\\ 0\end{bmatrix}$, $\boldsymbol{C} = [1 \quad 2]$, $\boldsymbol{D} = 0$. Vytvorte v programovom prostredí MATLAB systém v stavovom popise.

```
>> A=[-2 -1;1 0];
>> B=[1;0];
>> C=[1 2];
>> D=[0];
>> sys=ss(A,B,C,D)
```

výsledok:

```
A =
       x1  x2
   x1  -2  -1
   x2   1   0
B =
       u1
   x1   1
   x2   0
C =
       x1  x2
   y1   1   2
```

5) **V ďalšom budeme ilustrovať funkcie Control System Toolbox-u, ktoré umožňujú prepis lineárnych modelov DS z časovej oblasti do s-oblastí s využitím LT*:***

Function		***Význam***
c2d	→	*zo spojitého stavového priestoru na diskrétny tvar*
tf2zp	→	*prenosová funkcia $H(s) \rightarrow H(s)$ v tvare z_i/p_i*
zp2tf	→	*$H(s)$ v tvare $z_i/p_i \rightarrow$ prenosová funkcia $H(s)$*
tf2ss	→	*prenosová funkcia $H(s) \rightarrow$ stavový priestor*
ss2tf	→	*stavový priestor $\rightarrow$ prenosová funkcia $H(s)$*
zp2ss	→	*$H(s)$ v tvare $z_i/p_i \rightarrow$ stavový priestor*
ss2zp	→	*stavový priestor $\rightarrow H(s)$ v tvare z_i/p_i*
tfdata	→	*obrazový prenos – identifikácia dát*
pzmap	→	*vypísanie pólov a núl v*

Popis funkcií s príkladmi:

- ***c2d function*** **-** konvertuje spojitý popis modelu v stavovom priestore na diskrétny stavový popis modelu

$$[A_d B_d] = c2d(A, B, T_s)$$

kde:

A,B – matice DS v spojitom stavovom popise
T_S – perióda vzorkovania modelu
A_d, B_d – matice diskretného popisu modelu

PRÍKLAD 5

Majme zadaný spojitý systém v stavovom priestore

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} \text{ a } \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Vypočítajte matice diskrétného modelu systému ***A_d, B_d*** v stavovom priestore pomocou funkcie ***c2d*** systém s periódou vzorkovania 0,1.

```
>> A=[0,1;-3,-4];
>> B=[0; 1];
>> [Ad,Bd]=c2d(A,B,0.1)
```

Hodnoty matíc vypočítame c2d

```
Ad =
    0.9868    0.0820
   -0.2460    0.6588

Bd =
    0.0044
    0.0820
```

Stavový popis diskretného modelu má tvar

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0{,}9868 & 0{,}0820 \\ -0{,}2464 & 0{,}6568 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0{,}0044 \\ 0{,}0820 \end{bmatrix} u(k)$$

- ***tf2zp function*** **-** funkcia konvertuje polynomiálny tvar ***H(s)*** na tvar poly/nuly

$$[z, p, k] = tf2zp(num, den)$$

kde :

num/den – koeficienty čitateľa/menovateľa prenosovej funkcie *H(s)*
z – vektor núl
p – vektor polov
k - zosilnenie

PRÍKLAD 6

Majme zadaný systém v tvare obrazového prenosu *:* $H(s) = \frac{Y(s)}{U(s)} = \frac{10}{s^2+4s+3}$.
Vytvorte systém popísaný prenosovou funkciou v tvare póly/nuly.

Riešenie v jazyku MATLAB:

```
>> num = 10;
>> den=[1 4 3];
>> [z p k] = tf2zp(num,den)
```

```
z =
   Empty matrix: 0-by-1
p =
    -3
    -1
k =
    10
```

- ***zp2tf* function -** prevod ***H(s)*** v tvare póly/nuly na polynomiálny tvar

  $[num\ den] = zp2tf(z,p,k)$

  kde :

  z – stĺpcový vektor núl
  p – stĺpcový vektor pólov
  k – statické zosilnenie(pre MIMO systém každý stĺpec je určený pre jeden výstup MIMO systému)
  num/den – vektor koeficientov čitateľa/menovateľa *H(s)*

PRÍKLAD 7

Majme lineárny systém zadaný v tvare prenosovej funkcie *H(s)* póly/nuly. Pričom zosilnenie k = 10, póly systému sú $p_1 = -3$, $p_2 = -1$ a systém nemá nuly.

```
>> z=[];
>> p=[-3 -1];
>> k=10;
>> [num,den]=zp2tf(z,p,k)
```

```
num =
     0     0    10
den =
     1     4     3
```

- ***tf2ss* function –** prenosová funkcia ***H(s)*** v polynomiálnom tvare sa transformuje do podoby stavového popisu linearného dynamického systému

  $[A,B,C,D] = tf2ss(num,den)$

  kde:

  num/den – vektory obsahujúce koeficienty pri mocninách s klesajúcim exponentom operátora *s* prenosovej funkcie *H(s)*
  A,B,C,D – matice stavového popisu LDS

PRÍKLAD 8

Majme systém zadaný prenosovou funkciou v polynomiálnom tvare*:* $H(s) = \frac{10}{s^2+4s+3}$.
Vytvorte pomocou funkcii Control System Toolbox-u systém popísaný v stavovom priestore.

Riešenie v jazyku MATLAB:

```
>> num=10;
>> den=[1 4 3];
>> [A,B,C,D]=tf2ss(num,den)
```

```
A =
    -4    -3
     1     0
B =
     1
     0
C =
     0    10
D =
     0
```

Stavový popis systému:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -4 & -3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

- ***ss2tf function*** **-** konvertuje systém v tvare spojitého popisu na polynomiálny tvar prenosovej funkcie ***H(s)***

$$[num, den] = ss2tf(A, B, C, D, iu)$$

kde:

A,B,C,D – matice stavového popisu LDS
Iu – pre SISO systém iu = 1
num/den –vektory koeficientov čitateľa/menovateľa prenosovej funkcie ***H(s)***

PRÍKLAD 9

Majme zadaný systém v stavovom priestore:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 10 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u \ .$$

Vytvorte systém popísaný prenosovou funkciou v polynomiálnym tvarom.

Riešenie v jazyku MATLAB:

```
>> A=[0,1;-3,-4];
>> B=[0 1]';
>> C=[10 0];
>> D=0;
>> iu=1;
>> [num,den]=ss2tf(A,B,C,D,iu)
```

```
num =
     0     0    10
den =
     1     4     3
```

- ***zp2ss function*** **–** konvertuje prenosovú funkciu ***H(s)*** v tvare póly/nuly do stavového priestoru

$$[A, B, C, D] = zp2ss(z, p, k)$$

kde:

z – matica korešpondujúca s rozložením núl, ktorá má jeden stĺpec z MIMO systému(ak uvažujeme SISO systém ***z*** je stlpcový vektor)

p – stĺpcový vektor rozloženia pólov
k - statické zosilnenie
A, B, C, D – matice stavového priestoru

PRÍKLAD 10
Majme systém zadaný prenosovou funkciou v tvare poly/nuly: $H(s) = \frac{10}{(s+2)(s+1)}$.
Pretransformujte tento systém pomocou funkcie *Control System Toolbox*-u na systém popísaný v stavovom priestore.

```
>> z=[];
>> p=[-3,-1];
>> k=10;
>> [A,B,C,D]=zp2ss(z,p, k)
```

```
A =
  -4.0000   -1.7321
    1.7321         0
B =
     1
     0
C =
     0    5.7735
D =
     0
```

- ***ss2zp function -*** konvertuje systém v stavovom priestore na prenosovú funkciu ***H(s)*** v tvare poly/nuly

$$[z, p, k] = ss2zp(A, B, C, D, iu)$$

kde:
A, B, C, D – matice dynamického systému opisaného v stavovom priestore, korešpondujúce s „*iu*-tým“ vstupom pre MIMO systém, iu = 1 pre SISO systém
z – vektor núl
p – vektor pólov
k – statické zosilnenie

PRÍKLAD 11
Majme systém „pružina/tlmič“ popísaný v stavovom priestore maticami:

$$A = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad C = [10 \quad 0] \qquad a \qquad D = 0.$$

S využitím funkcie Control System Toolbox-u transformujte popis tohto systému z časovej oblasti do Laplaceovej transformácie v tvare prenosovej funkcie – poly/nuly.

```
>> A=[0 1; -3 -4];
>> B=[0 1]';
>> C=[10 0];
>> D=[0];
>> iu=1;
>> [z p k]=ss2zp(A,B,C,D,iu)
```

```
z =
   Empty matrix: 0-by-1
p =
    -1
    -3
k =
    10
```

- ***tfdata function -*** zistenie hodnôt parametrov prenosovej funkcie *H(s)*

$$[num, den] = tfdata(sys)$$

kde:
sys – zadefinovaný systém v tvare prenosovej funkcie *H(s)*
num/den – vektory čitateľa/menovateľa prenosovej funkcie *H(s)* obsahujúce koeficienty (parametre) pri klesajúcich mocninách operátora ***s***

PRÍKLAD 12

Majme systém popísaný v tvare: $H(s) = \frac{s+2}{s^2+2s+1}$.

Pomocou funkcie Control System Toolbox-u zistite hodnoty parametrov prenosovej funkcie.

```
>> sys=tf([1 2], [1,2,1]);
>> [num,den]=tfdata(sys)
```

```
num =    [1x3 double]
den =    [1x3 double]
```

- ***pzmap function -*** vypísanie hodnôt pólov a núl v komplexnej rovine

$\boldsymbol{pzmap(sys)}$

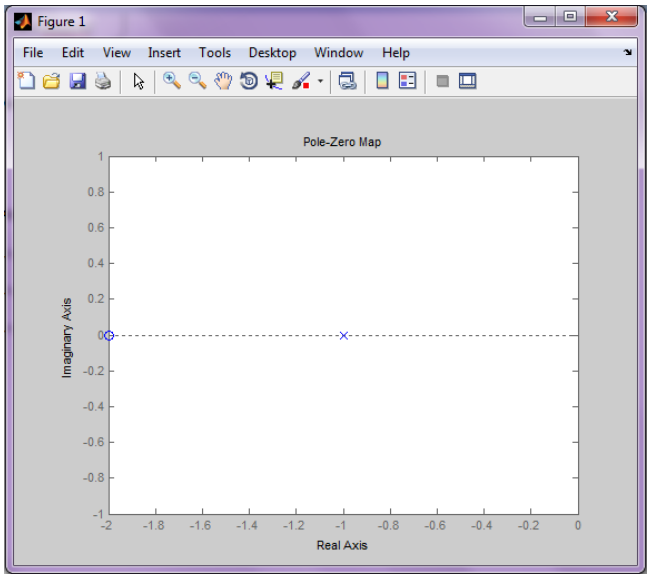
kde:

sys – definovaný LDS, ktorého poly/nuly chceme vykresliť

PRÍKLAD 13

Vykreslite póly a nuly LDS zadaného prenosovou funkciou v polynomiálnom tvare : $\mathbf{H(s)} = \frac{s+2}{s^2+2s+1}$.

```
>> num=[1,2];
>> den=[1,2,1];
>> sys=tf(num,den);
>> pzmap(sys)
```



Obrázok 7-1 Znázornenie polov/nul prenosovej funkcie *H(s)* LDS

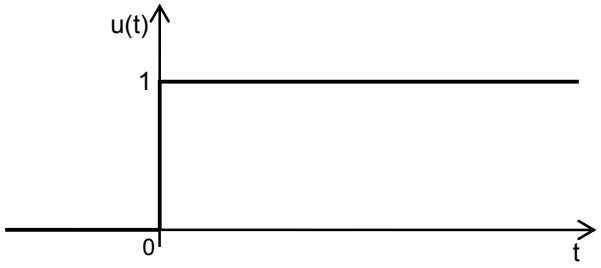
7.2 Analýza lineárnych dynamických systémov (LDS) s využitím funkcií *Control System Toolbox*-u

- **Funkcie pre analýzu lineárnych dynamických systémov v časovej oblasti**

Jednotkový skok a prechodová charakteristika

- prechodová charakteristika je odpoveď dynamického systému na jednotkový skok pri nulových počiatočných podmienkach.
- analytické vyjadrenie prechodovej charakteristiky sa nazýva prechodová funkcia
- jednotkový skok (Heavisidov skok) sa označuje ako $1(t)$ a jeho matematické vyjadrenie:

$$1(t) < \begin{matrix} 0\ ak\ t < 0 \\ 1\ ak\ t \geq 0 \end{matrix}$$



Vstupný budiaci signál u(t) = 1(t) (jednotkový skok)

- funkcia $\boldsymbol{step(system)}$ **-** umožňuje vypočítať odozvu LDS

Váhová funkcia a impulzná charakteristika

- Impulzná charakteristika je odozva na výstupe z LDS na Diracov impulz (jednotkový impulz).
- Diracov impulz je definovaný ako derivácia jednotkového skoku,
- váhová funkcia je analytické vyjadrenie odozvy nenabudeného sústavu na Diracov impulz,
- jednotkový impulz označujeme $\delta(t$) a má nasledujúce matematické vyjadrenie:

$$u(t) = \delta(t) < \begin{matrix} 0\ ak\ t > 0, t < 0 \\ nie\ je\ definované\ ak\ t = 0 \end{matrix}$$

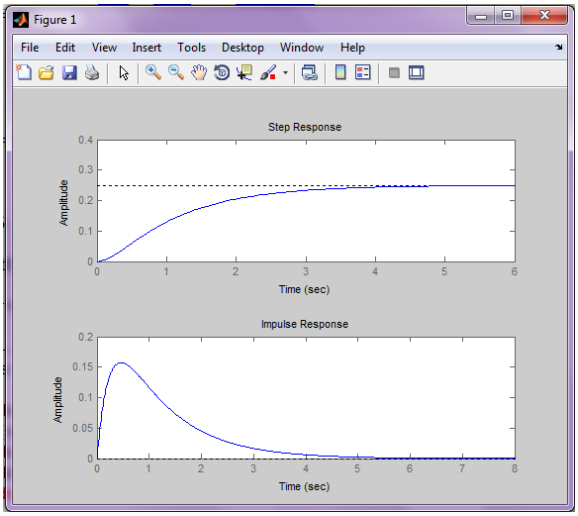
Vstupný budiaci signál $u(t) = \delta(t)$ (Diracov impulz)

- funkcia $\boldsymbol{impulse(systém)}$ - umožňuje vypočítať odozvu LDS na vstupný signál $u(t) = \delta(t)$

PRÍKLAD
Majme obrazový prenos LDS prenosovou funkciou $F(s) = \frac{1}{s^2+5s+4}$.

Vypočítajte a vykreslite v simulačnom jazyku MATLAB a pomocou Control System Toolbox-u prechodovú a impulznú charakteristiku systému.

```
>> num =1; % definícia čitateľa obrazového prenosu
>> den = [1 5 4]; % definícia menovateľa obrazového prenosu
>> sys=tf(num,den) % vytvorenie systému pomocou num, den

Transfer function:
  1
-------------
s^2 + 5 s + 4

>> subplot(211)
>> step(sys) % vykreslenie prechodovej charakteristiky
>> subplot(212)
>> impulse(sys) % vykreslenie impulznej časovej charakteristiky
```



Obrázok 7-2 Prechodová a impulzná charakteristika LDS ako odozva na u(t) = 1(t)/ $\delta(t)$

- **Funkcie pre analýzu lineárnych dynamických systémov vo frekvenčnej oblasti**

Nyquistova charakteristika
Nyquistova charakteristika je znázornenie komplexných hodnôt frekvenčného prenosu v Gaussovej rovine komplexných čísel, pričom sa vychádza z algebraickeho tvaru frekvenčného prenosu (funkcie):

$$F(i\omega) = Re[F(i\omega)] + i\,Im[F(i\omega)]$$

- funkcia $nyquist(system)$ – umožňuje vypočítať a vykresliť Nyquistovú charakteristiku

Nicholsova charakteristika
Nicholsova charakteristika je znázornenie prirodzeného logaritmu frekvenčného prenosu v Gaussovej rovine komplexných čísel. Vychádza z exponenciálneho tvaru frekvenčného prenosu:

$$\ln F(i\omega) = \ln|F(i\omega)|e^{i\varphi(\omega)} = \ln|F(i\omega)| + i\varphi(\omega)$$

- funkcia $nyquist(system)$ – umožňuje vypočítať a vykresliť Nicholsovu charakteristiku

Bodeho charakteristika

Bodeho frekvenčné charakteristiky sú parametrickým vyjadrením Nicholsovej frekvenčnej charakteristiky v závislosti od logω (amplitudová charakteristika, fázová charakteristika)

Funkcie $\boldsymbol{bode(system)}$ umožňuje vypočítať a vykresliť Bodeho charakteristiky vykresliť pomocou

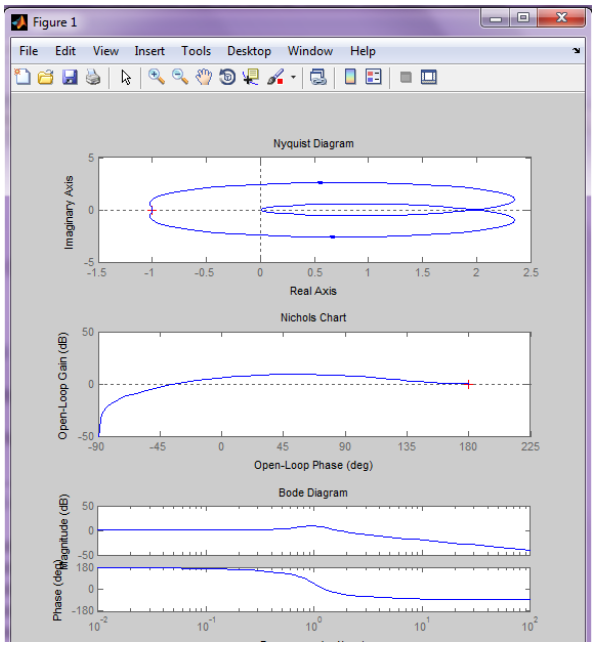
PRÍKLAD

Majme definovaný obrazový prenos v tvare

$$F(s) = \frac{s-1}{s^2+0.5s+1},$$

pomocou funkcií *Control System Toolbox*-u. Vykreslite Nyquistovu, Nicholsovu a Bodeho frekvenčné charakteristiky.

```
>> num= [1 -1];  % vytvorenie num - čitateľ obrazového prenosu
>> den = [1 0.5 1]; % vytvorenie denum - menovateľ obrazového prenosu
>> sys=tf(num,den) % vytvorenie obrazového prenosu pomocou num,den

Transfer function:
 s - 1
---------------
s^2 + 0.5 s + 1

>> subplot(311)
>> nyquist(sys) %nyquistová frekvenčná charakteristika
>> subplot(312)
>> nichols(sys) %nicholsová frekvenčná charakteristika
>> subplot(313)
>> bode(sys) % bodeho frekvenčné charakteristiky
```

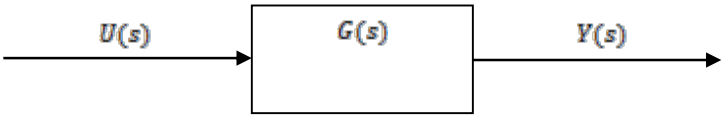


Obrázok 7-3 Frekvenčné charakteristiky LDS (Nyquist, Nichols, Bode)

7.3 Úprava blokových schém regulačných obvodov v prostredí MATLAB

Cieľom cvičenia je naučiť sa aplikovať príkazy pre úpravu blokových schém pre rôzne typy regulačných obvodov.

Základomblokovej schémy riadiaceho systému je blokové zobrazenie všeobecného lineárneho systému so vstupom U(s) a výstupom Y(s):



ktorý je popísaný prenosovou funkciou v tvare

$$G(s) = \frac{Y(s)}{U(s)}$$

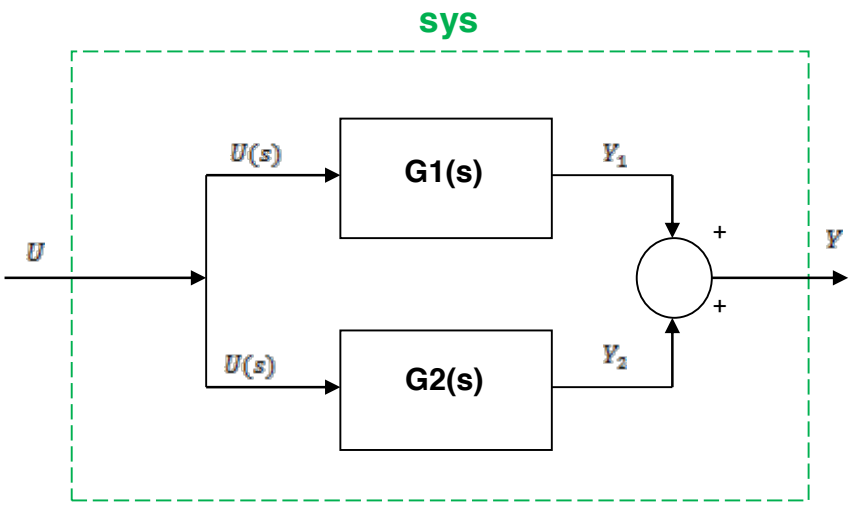
- **Redukcie blokových schém v simulačnom jazyku MATLAB**

Control Toolbox obsahuje príkazy na redukciu blokových schém na jednoduchšie schémy. Tieto príkazy (*cloop, feedback, series, paralel, connect, blkbuild)* vedú k zjednotenému opisu komplexnejšieho systému.

- **Paralelné zapojenie:**

function parallel - generuje celkový prenos dvoch systémov radených paralelne

```
sys = parallel (sys1, sys2)
```



Obrázok 7-4 Paralelné zapojenie prenosov

Z obrázka pre výstup systému vyplýva:

$$Y = Y_1 + Y_2 = G_1 U + G_2 U = U(G_1 + G_2)$$

Výsledný prenos dostávame v tvare:

$$\frac{Y(s)}{U(s)} = G_1 + G_2$$

PRÍKLAD 1

Nech je daný systém pozostávajúci z dvoch paralelne zapojených prenosových funkcií:
$H_1(s) = \frac{1}{s+2}$; $H_2(s) = \frac{s+3}{s+10}$;

Vypočítajte výsledný prenos $H(s)$ s využitím funkcie ***parallel***.

```
sys1 = tf([1],[1 2]);
sys2 = tf([1 3],[1 10]);
[numh,denh] = parallel(sys1,sys2);
sys=parallel(sys1,sys2);
printsys(numh,denh,'s')
```

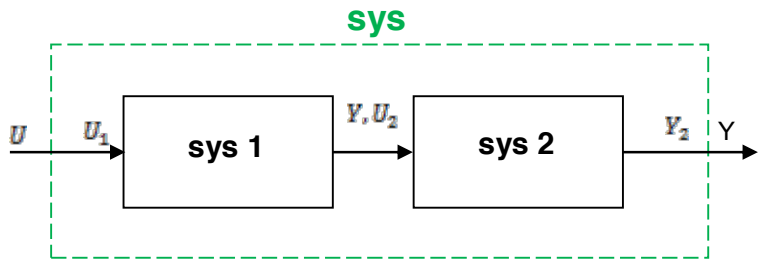
Výsledok získaný v programovom prostredí MATLAB:

```
Transfer function:

s^2 + 6 s + 16
--------------
s^2 + 12 s + 20
```

- **Sériové zapojenie:**

  function series - generuje celkový prenos dvoch systémov zapojených do série

  ```
  sys = series (sys1, sys2)
  ```



Obrázok 7-5 Sériové zapojenie prenosov

Z definície prenosov jednotlivých systémov je zrejmé, že:

$$\frac{Y_1}{U_1} = G_1 \Rightarrow Y_1 = G_1 U_1$$

$$\frac{Y_2}{U_2} = G_2 \Rightarrow Y_2 = G_2 U_2$$

odkiaľ

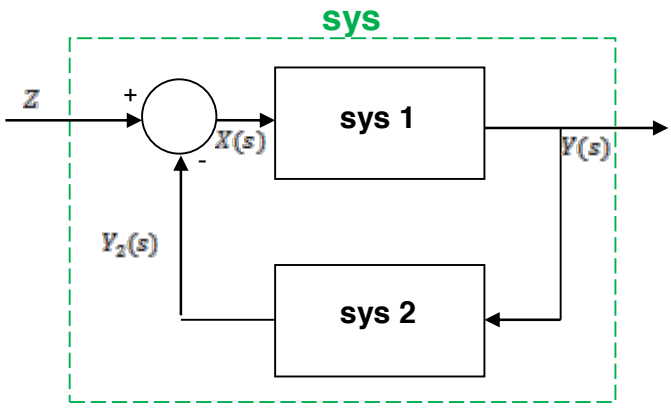
$$U_2 = Y_1 \Rightarrow Y_2 = G_2 G_1 U_1$$

Výsledný prenos má tvar:

$$\frac{Y_2}{U_1} = G_2 G_1$$

- **Spätnoväzobné zapojenie:**

```
sys = feedback (sys1, sys2)
```



Obrázok 7-6 Spätnoväzobná riadiaca štruktúra – prenos na poruchu

Takto volaná funkcia reprezentuje v teórii automatického riadenia výpočet prenosu na poruchu (kde sys1 = $G_S(s)$ – LDS, sys2 = $G_R(s)$ – zvolený prenos regulátora):

$$\frac{Y(s)}{Z(s)} = \frac{Y(s)}{Y_2(s)+X(s)} = \frac{G_s(s)X(s)}{X(s)G_s(s)G_R(s)+X(s)} = \frac{G_s(s)}{G_s(s)G_R(s)+1}$$

PRÍKLAD 2
Napíšte program pre výpočet výsledného prenosu spätnoväzobného usporiadania, ak sys1 = $G_0(s)$, sys2 = H(s) a platí

$$G_0 = \frac{s+1}{(s+3)(s+5)} \text{ a } H(s) = \frac{s+6}{s+10}$$

v spätnej väzbe.

Funkcia ***feedback*** vypočíta výsledný prenos spätnoväzobného usporiadania podľa vzorca:

$$G_{CL}(s) = \frac{Y(s)}{Z(s)} = \frac{G_0(s)}{(1+G_0(s)\cdot H(s))};$$

ktorý v teórii riadenia odpovedá prenosu URO na poruchu (Z(s))

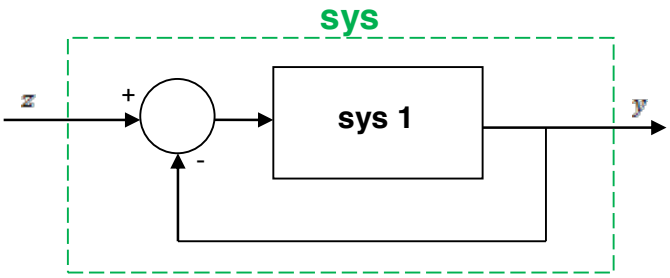
```
numgo = [1 1]; dengo = conv([1 3],[1 5]);
sys1=tf(numgo,dengo);
sys2=tf([1 6],[1 10]); %prenos H(s) v spatnej vazbe
sys=feedback(sys1,sys2); % vysledny prenos URO
printsys(sys,'s'); % nejde potrebujem num, den
```

Poznámky k funkcii *feedback*:

- ⇨ funkcia ***feedback*** defaultne uvažuje zapojenie so *zápornou spätnou väzbou*
- ⇨ vyjadrenie *kladnej spätnej väzby* v programovom prostredí MATLAB v Control System Toolboxe vykonáme pomocou príkazu:

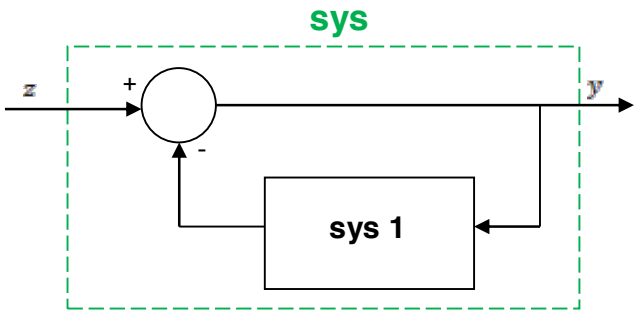
$$sys = feedback(sys1, -sys2)$$

⇨ ak v niektorej vetve nie je zapojený systém s daným prenosom, uvažujeme prenos rovný 1



$$sys = feedback(sys1, 1)$$

Obrázok 7-7 Prenos nachádzajúci sa v priamej vetve

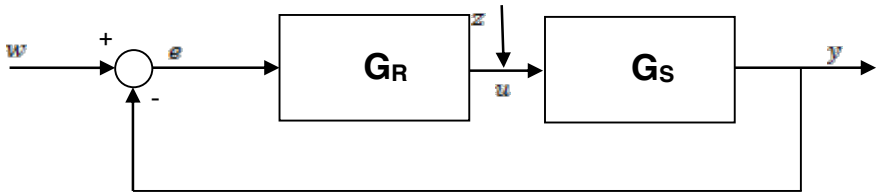


$$sys = feedback(1, sys1)$$

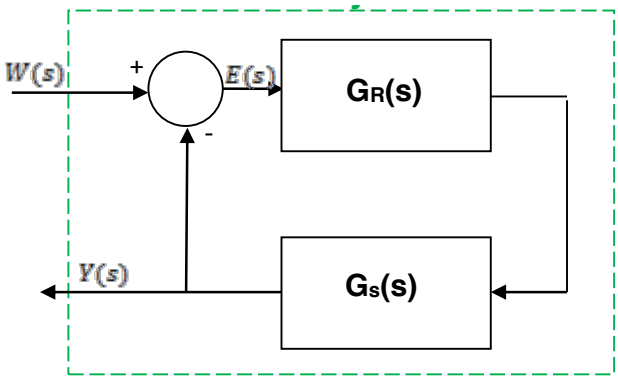
Obrázok 7-8 Prenos nachádzajúci sa v spätnej väzbe

⇨ ak je sys1 v priamej vetve tvorený sériovým zapojením prenosov $G_S(s)$ a $G_R(s)$, kde $G_S(s)$ – LDS, $G_R(s)$ – zvolený prenos regulátora, výsledný prenos vypočítame volaním funkcie feedback v tvare

```
sys = feedback(series(Gr,Gs),1)
```



Obrázok 7-9 Spätnoväzobná riadiaca štruktúra – prenos na požadovanú hodnotu



Obrázok 7-10 Spätnoväzobná riadiaca štruktúra – prenos na požadovanú hodnotu – alternatívne zobrazenie

čo reprezentuje v teórii automatického riadenia výpočet prenosu na požadovanú hodnotu:

$$\frac{Y(s)}{W(s)} = \frac{Y(s)}{E(s)+Y(s)} = \frac{E(s)G_R(s)G_s(s)}{E(s)+E(s)G_R(s)G_s(s)} = \\ = \frac{G_R(s)G_s(s)}{1+G_R(s)G_s(s)}$$

PRÍKLAD 3

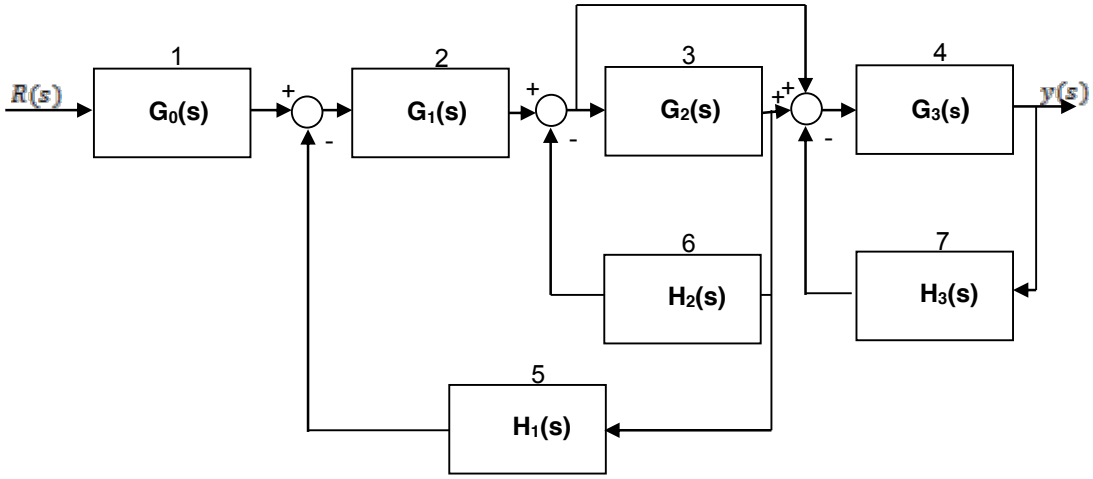
Uvažujeme bloky $G_0(s), H(s)$ z predošleho príkladu, ale zaradené v sérií v priamej vetve. Vytvorte prenosovú funkciu $G_{c3}(s)$ zloženú z blokov $G_0(s), H(s)$ v priamej vetve, kt. su uzatvorené do štruktúry s jednotkovou SV

```
% redukcia blokových schém
ngo=[1 1]; dgo=conv([1 3],[1 5]);
sys1=tf(ngo,dgo); sys2=tf([16],[1 10]);
sys12=series(sys1,sys2);
sys=feedback(sys12,1); % G_0H=G_0*H
printsys(sys,'s'); % G_c3 = G_0H/1+G_oH = Y/W
```

Riešenie komplikovaného systému zahŕňa niekoľko krokov. Nasledujúce príkazy obsahuje programové prostredie MATLAB v Control Toolbox. Tieto príkazy si ukážeme na nasledujúcom probléme.

PRÍKLAD 4

Zaoberajme sa štruktúrou na Obr. 8 . Úlohou je vypočítať výsledný prenos tejto štruktúry.



Obrázok 7-11 Bloková schéma zložitého systému

Bloková štruktúra zložitého systému $\frac{Y(s)}{R(s)} = ?$

Predpokladajme, že:

$$G_0(s) = 1;\ G_1(s) = \frac{1}{s+1};\ G_2(s) = \frac{1}{s+2};\ G_2(s) = \frac{1}{s+3};\ H_1(s) = 4;\ H_1(s) = 8;\ H_1(s) = 12.$$

Použijeme príkazy ***bbuild***, ***connect*** pre vytvorenie m-file-u buidsys.m

- ⇨ Každému systému priradíme číslo (viď obr.)
- ⇨ Nakoľko pracujeme so 7 prenosovými funkciami, vložíme ich do programového priestoru MATLAB ako polynóm n_i, d_i alebo pomocou funkcie „*tf*“.

```
% redukcia zložitej blokovej schémy
n1=1; d1=1; n2=1; d2= [1 1]; n3=1; d3=[1 2]; n4=1; d4=[1 3];
n5=4; d5=1; n6=8; d6=1; n7=12; d7=1;
nblocks = 7;      % počet subsystémov
blkbuild    % používa premennú nblocks na výstavbu systému, vykoná sa to v
stavovom priestore použitím "tf2ss";
```

```
% vytvára jeden blokovo-diagonálny stavový model použitím
% funkcie "append"
q=[2 1 -5 0 0      % vytvoríme maticu q
    3 2 -6 0 0      % identifikuje prepojenia medzi subsystémami
    4 2 -6 3 -7     % každý riadok odpovedá samostatnému subsystému
    5 3 0 0 0        % Prvé číslo je pridelene subsystému
    6 3 0 0 0        % ostatne čísla určujú, ktoré bloky majú svoj výstup
    7 4 0 0 0];      % pripojený na vstup tohto systému.

input = 1;  % určený vstupný blok štruktúry
output = 4;       % určený výstupný blok štruktúry
[Ad,Bd,Cd,Dd]=connect(A,B,C,D,q,input, output)
      % redukcia systému po vykonaní prepojení na 1st.m.
        [num,den]=ss2tf(Ad,Bd,Cd,Dd);
printfsys(num,den,'s')
```

Poznámka

1. ***nblocks*** – špecifikácia 7 subsystémov
2. ***blkbuild***– funkcia konvertuje vsetky opisy prenosovými funkciami na modeli v stavovom priestore →***tf2ss***→ a tvorí jeden z blokovo diagonálny stavový model obsahujúci A,B,C,D opakovaným použitím ***append***.
3. Vytvorenie matice q → definuje prepojenia medzi systémami. Každý riadok q odpovedá jednému subsystému. Prvé číslo je číslo subsystému, ostatné čísla určujú, ktoré bloky majú výstupy pripojené na vstup subsystému. (1.riadok ~ subsystém 2 ($G_1(s)$) :$G_0(s)\sim 1$, $H_1(s)\sim -5$
4. Funkcia ***connect*** vykonáva prepojenia a redukuje celý systém na jeden stavový model.

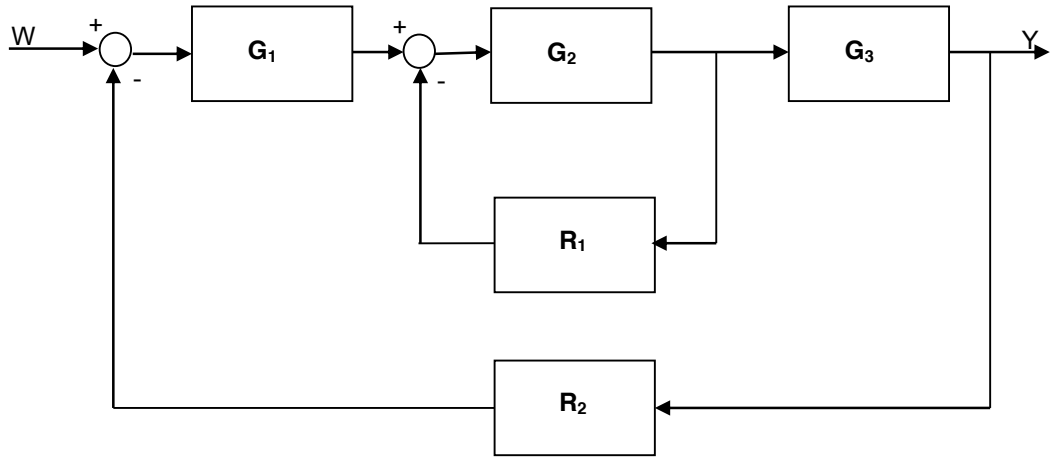
Testovanie:

$$\frac{num}{den} = \frac{-7.105e-15s^2+1s+3}{s^3+26s^2+179s+210} \cong 0$$

Kontrola porovnať polohy pólov a núl pre opis stavovým modelom a opisom prenosovou funkciou ~ musia byť identické!

Funkcia ***minreal***→ funkcia na získanie minimálnej formy
Programové prostredie MATLAB pri operáciách spájania nevykompenzuje spoločné korene činiteľa a menovateľa. Prenosovej funkcie → odporúča sa použiť príkaz funkcie minreal!

PRÍKLAD 5
Pre blokovú schému regulačného obvodu na Obr. 9 vypočítajte výslednú prenosovú funkciu $G_{\frac{Y}{W}}(s)$. Pri spájaní jednotlivých blokov použite príkazy prebraté skôr.



Obrázok 7-12 Bloková schéma zložitého systému

$$G_1(s)=\frac{0,5}{2s};\ G_2(s)=\frac{0,1}{s(0,15s+0,5)};\ G_3(s)=\frac{10}{1,2s+1};\ R_1(s)=2;\ R_2(s)=\frac{s+0,15}{10s}$$

Na základe pravidiel blokovej algebry:

$$G_{\frac{Y}{W}}(s)=\frac{Y(s)}{W(s)}=\frac{G_1(s)*G_2(s)*G_3(s)*R_1(s)*R_2(s)}{1+G_2(s)*R_1(s)+G_1(s)*G_2(s)*G_3(s)*R_2(s)*R_1(s)}$$

```
g1s=tf(0.5,[2 0]) % inicializácia G1(s)
g2s = tf(0.1,[0.5 0.5 0])%inicializácia G2(s)
g3s=tf(10, [1.2 1]) % inicializácia G3(s)
r1s=tf(2); r2s=tf([1 0.15],[10 0])% inicializácia R1 a R2
numgyw=g1s*g2s*g3s*r1s*r2s% čitateľ G_Y/W -> URO
dengyw=1+g2s*r1s+g1s*g2s*g3s*r2s*r1s % menovateľ G_Y/W
gyw=numgyw/dengyw % celková prenosová funkcia URO
gywpn =zpk(gyw) % vyjadrenie pomocou pólov a núl
gywm=minreal(gyw) % min. realizácia pr. funkcie URO
```

zjednodušený postup → najprv spočítame vnútornú slučku a následne potom celú funkciu pomocou funkcie feedback

```
spv1=feedback(g2s,r1s) % vnútorná SV G2(s) a R1(s)
```

transfer function: $\frac{0,1}{0,15s^2+0,5s+0,2}$

```
spv2=feedback(g1s*spv1*g3s,r2s) % 2. SV tvorená G1(s) ,SPV1, G3(s) a R2(s)
```

transfer function: $\frac{5s}{36s^5+15s^4+4,8s^3+4s^2+0,5s+0,075}$;

Prevod do formy „póly-nuly-zosilnenie“: ***zpk(spv2)***

Z/P/Gain: $\frac{1,3889s}{(s+2,859)(s+0,096)(s+0,297)(s^2+0,04s+0,02)}$

Zjednodušenie prenosovej funkcie: ***minreal(spv2)***

transfer function: $\frac{1,389s}{s^5+4,16s^4+4,11s^3+1,11s^2+0,13s+0,02}$

riešenie cez *connect*→ q

Použitím prebratých príkazov nájdite výslednú prenosovú funkciu ak: regulovaný proces je opísaný $G_p(s)=\frac{5}{(40s+1)^2}e^{-40s}$ a regulátor PID je vyjadrený prenosovou funkciou $G_R(s)=8\left(1+\frac{1}{80s}+20s\right)=\frac{160s^2+80s+1}{10s}$

7.4 Zadanie č. 4. : Modelovanie a analýza modelu fyzikálneho systému jednosmerný motor v prostredí MATLAB s využitím funkcií Control Toolboxu

ZADANIE:
Uvažujte fyzikálno- matematický model dynamického systému, ktorý je popísaný lineárnou diferenciálnou rovnicou (LDR) 2. a vyššieho rádu.

ÚLOHA: *Navrhnite m-file v simulačnom jazyku Matlab, ktorý umožní:*

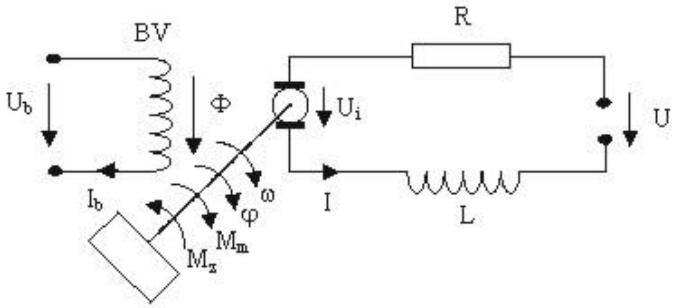
1. *Zadefinovanie prenosovej funkcie opisujúcej LDS v s-oblasti (v polynomiálnom tvare, v tvare póly/nuly) a v stavovom priestore pomocou matíc A,B,C,D.*
2. *Konverziu modelov zo stavového priestoru do tvaru prenosovej funkcie a naopak.*
3. *Analýzu LTI DS v časovej (prechodová charakteristika, impulzná charakteristika, odozva na ľubovoľný vstupný signál) a frekvenčnej oblasti (Nyquistová, Nicholsová a Bodeho charakteristiky).*
4. *Vyhodnotenie stability uvažovaného LTI dynamického systému na základe získaných odoziev na rôzne typy vstupných signálov.*

Parametre:
$B\ [N.m.s^{-1}]$ – *koeficient viskózneho trenia*

R [Ω] – *odpor*

L [H] – *indukcia*

$C_u\ [N.m\ /\ Amp]$ – *momentová konštantamotora*



Pôsobiace momenty:

$J\ [kg.m^2]$ – *moment zotrvačnosti*

$M_m\ [Nm]$– *krútiaci moment*

$M_{dyn}\ [kg.m^2s^{-2}]$ – *dynamický moment*

Vstup:
$U\ [V]$– *zdroj napätia*
Výstup:
$\omega[s^{-1}]$- *uhlová rýchlosť hriadeľa*
$\varphi\ [rad]$ - *uhlová poloha hriadeľa*

Na motore sa vytvára indukované napätie: $U_i = C_u * \omega$

Získame dve rovnice. Prvá je diferenciálna rovnica vyplývajúca z 2.Kirchhoffovho zákona :

$$L\frac{dI(t)}{dt} + RI(t) + U_i(t) = U(t)$$

druhá rovnica popisuje mechanické deje – pohyb (vyplýva z 2.Newtonovho zákona):

$$M_m(t) - M_z(t) = M_{dyn}(t).$$

Pre krútiaci moment hriadeľa približne platí:

$$M_m(t) = C_u I(t)$$

Záťažový moment (typický tvar) :
$M_z(\omega) = B * \omega$

Dynamický moment je daný rovnicou:
$M_{dyn}(t) = J\frac{d\omega}{dt} = J\frac{d^2\varphi}{dt^2}$

Do 2. rovnice dosadíme rovnice pre $M_m(t),\ M_z(t),\ M_{dyn}(t)$ z ktorých následne dostaneme:

$$M_m(t) = M_{dyn}(t) + M_z(t) = J\frac{d\omega}{dt} + B * \omega = C_u I(t)$$

Z tejto rovnice si vyjadríme prúd $I(t)$:

$$I(t) = \frac{J}{C_u}\frac{d\omega}{dt} + \frac{B}{C_u} * \omega$$

Pre dosadenie do 1.diferenciálnej rovnice potrebujeme aj prvú deriváciu prúdu:

$$\frac{dI(t)}{dt} = \frac{J}{C_u}\frac{d^2\omega}{dt^2} + \frac{B}{C_u} * \frac{d\omega}{dt}$$

Teraz dosadíme rovnice do prvej diferenciálnej rovnice, pričom za indukované napätie na motore dosadíme $U_i = C_u * \omega$:

$$L * \left(\frac{J}{C_u}\frac{d^2\omega}{dt^2} + \frac{B}{C_u} * \frac{d\omega}{dt}\right) + R * \left(\frac{J}{C_u}\frac{d\omega}{dt} + \frac{B}{C_u} * \omega\right) + C_u * \omega = U(t)$$

$$\frac{L * J}{C_u} * \frac{d^2\omega}{dt^2} + \left(\frac{L * B}{C_u} + \frac{R * J}{C_u}\right) * \frac{d\omega}{dt} + \left(\frac{R * B}{C_u} + C_u\right) * \omega = U(t)$$

resp.
$$L * J * \frac{d^2\omega}{dt^2} + (L * B + R * J) * \frac{d\omega}{dt} + \left(R * B + {C_u}^2\right) * \omega = U(t) * C_u$$

Použitím Laplaceovej transformácie získame diferenciálnu rovnicu v oblasti laplaceových obrazov:

$$L * J * s^2 * Y(s) + (L * B + R * J) * s * Y(s) + \left(R * B + {C_u}^2\right) * Y(s) = C_u * U(s)$$

Z ktorej tejto rovnice získam prenosovú funkciu:

$$G(s) = \frac{C_u * U(s)}{s^2(JL) + s(BL + JR) + ({C_u}^2 + BR)}$$

Pre zápis v programovom prostredí MATLAB s použitím Control Math Toolbox:

$$num = C_u U(t)$$

$$denum = JL + (BL + JR) + ({C_u}^2 + BR)$$

Prepis do stavového popisu v maticovom tvare:

Stavový popis tvoria dve rovnice v maticovom tvare, kde prvá stavová rovnica je diferenciálna,predstavujúca rovnicu stavu:

$x'(t) = Ax(t) + Bu(t)$,
kde $x(t)$ je stavový vektor,$u(t)$ je vektor vstupov, A je matica vnútorných väzieb systému a B je matica väzieb systému na vstup. Druhá stavová rovnica je algebraická rovnica predstavujúca rovnicu výstupu:

$$y(t) = Cx(t) + Du(t),$$

kde$y(t)$ je vektor výstupov, $u(t)$ je vektor vstupov,C je matica väzieb výstupu na stav a D matica väzieb vstupu na výstup.

Stavový vektor:

$$\underline{x} = \begin{bmatrix} I \\ \omega \end{bmatrix}$$

Treba vyjadriť prvú deriváciu oboch stavových veličín :

Dosadením do $M_m - M_z = M_{dyn}$ rovnice :

$$M_m(t) = C_u I$$
$$M_z(\omega) = B * \omega$$
$$M_{dyn}(t) = J\frac{d\omega}{dt}$$

Dostaneme :

$$J\frac{d\omega}{dt} = C_u I - B * \omega$$

z ktorej si jednoducho vyjadrime prvú deriváciu uhlovej rýchlosti:

$$\frac{d\omega}{dt} = \frac{C_u}{J} I - \frac{B}{J} * \omega$$

A z rovnice

$$L\frac{dI}{dt} + RI + U_i = U$$

kde za indukované napätie na motore dosadíme :

$$U_i = C_u * \omega$$

jednoducho vyjadríme prvú deriváciu prúdu :

$$\frac{dI}{dt} = \frac{U}{L} - \frac{R}{L} I - \frac{C_u}{L} * \omega$$

$$\begin{bmatrix} I \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{C_u}{L} \\ \frac{C_u}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} I \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} * U$$

$$y = \omega = [0 \quad 1] \begin{bmatrix} I \\ \omega \end{bmatrix}$$

Hodnoty parametrov pre modelovanie a analýzu jednosmerného motora v programovom prostredí MATLAB:

$J = 0{,}01\ kg.m^2$,$B = 0{,}1\ N.m.s$, $R = 1\ \Omega$,$L = 0{,}5H$, $C_u = 0{,}01\ ^{N.m}/_{Amp}$, $U = 1V$

Riešenie úloh modelovania a analýzy v programovom prostredí MATLAB s využitím vlastnej aplikácie s menu rozhraním

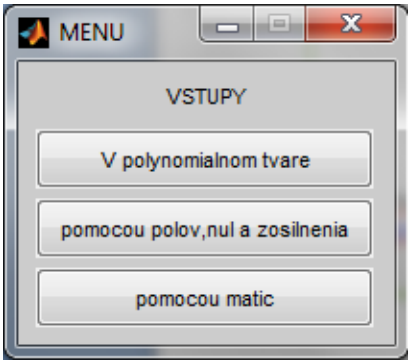
```
voľba=menu('','definovanie prenosovej funkcie','konverzia','časové
charakteristiky',' frekvenčné charakteristiky','stabilita','koniec');
% tvorba menu a jeho obsahu, prvý parameter je názov, ostatné možnosti
(tlačidlá) menu

switch volba
case 1,
        vstup
case 2,
        konverzia
case 3,
        casova
case 4,
        frekvencna
case 5,
        stabilita
case 6,
end
```

Obrázok 7-13 Grafické MENU vytvorené pomocou funkcie menu

Definovanie vstupov v simulačnom jazyku MATLAB:

```
function [sys]=vstup()

global sys;
global a b c d;

voľba = menu('VSTUPY','V polynomiálnom tvare ','pomocou pólov ,núl
a zosilnenia ','pomocou matíc');

switch volba
case 1,
        num=input('Zadaj citatel ... num = ');
        den=input('zadaj menovatela... den = ');
        sys=tf(num,den) % tvorba systému v polynomiálnom tvare
    case 2,
        z=input('zadaj nuly : ');
```

```
        p=input('zadaj poly : ');
        k=input('zadaj zosilnenie : ');
        sys=zpk(z,p,k) % tvorba systému pomocou pólov a núl
case 3,
        a=input('zadaj maticu A: ');
        b=input('zadaj maticu B: ');
        c=input('zadaj maticu C: ');
        d=input('zadaj maticu D: ');
        sys=ss(a,b,c,d) % tvorba systému pomocou matíc stavového popisu
end
hlavny % skok späť do menu
return
```



Obrázok 7-14 Výber možnosti zadania systému

```
Zadanie systému v polynomiálnom tvare:
Zadaj čitateľa ... num = 0.01
Zadaj menovateľa... den = [0.005 0.06 10.01]
 Transfer function:
        0.01
--------------------------
0.005 s^2 + 0.06 s + 10.01
```

Ďalšie príkazy ktoré je potrebné použiť pri riešení tohto zadania v Control Toolbox:

Funkcie pre konverziu:

```
[num,den]=tfdata(sys,'v')
[num,den]=ss2tf(a,b,c,d,1)
[a,b,c,d]=tf2ss(num,den)
[z,p,k]=ss2zp(a,b,c,d)
[z,p,k]=tf2zp(num,den)
[num,den]=zp2tf(z,p,k)
[a,b,c,d]=zp2ss(z,p,k)
```

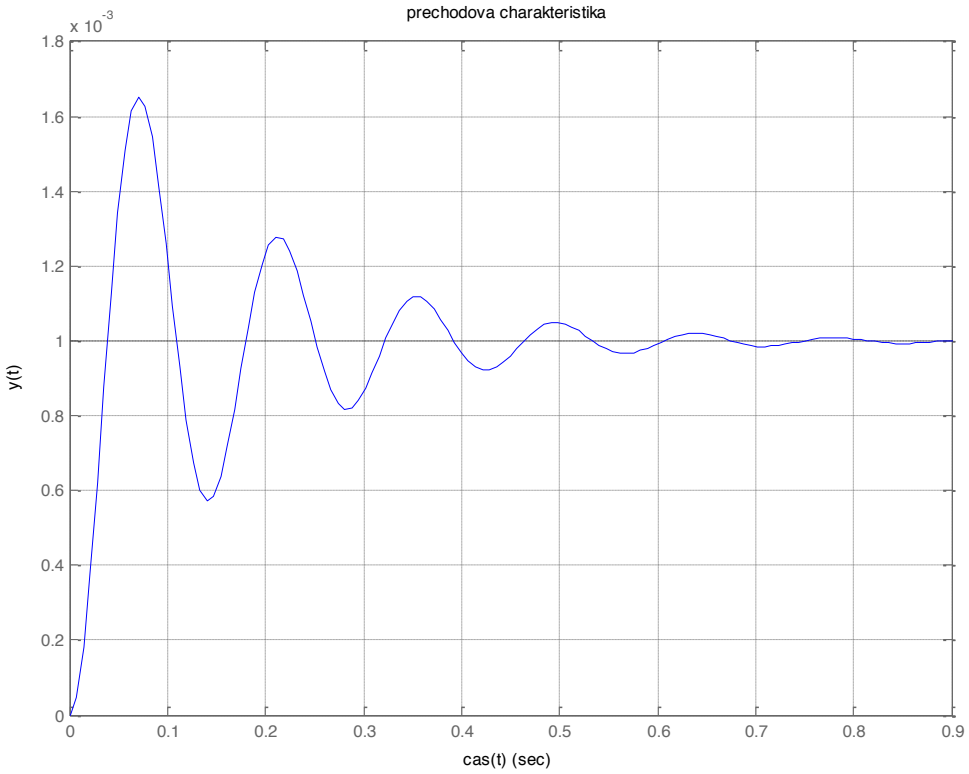
Časové charakteristiky:

```
step(sys);
impulse(sys)
```

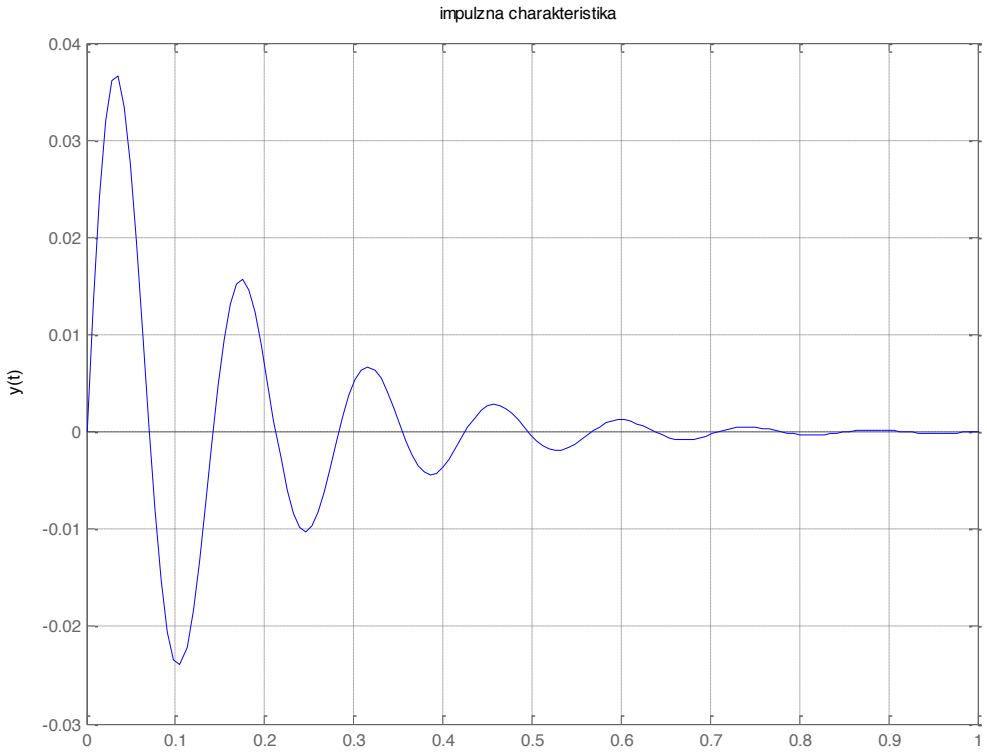
Frekvenčné charakteristiky:

```
nyquist(sys);
bode(sys);
nichols(sys);
```

Časové charakteristiky modelu otáčok motora:

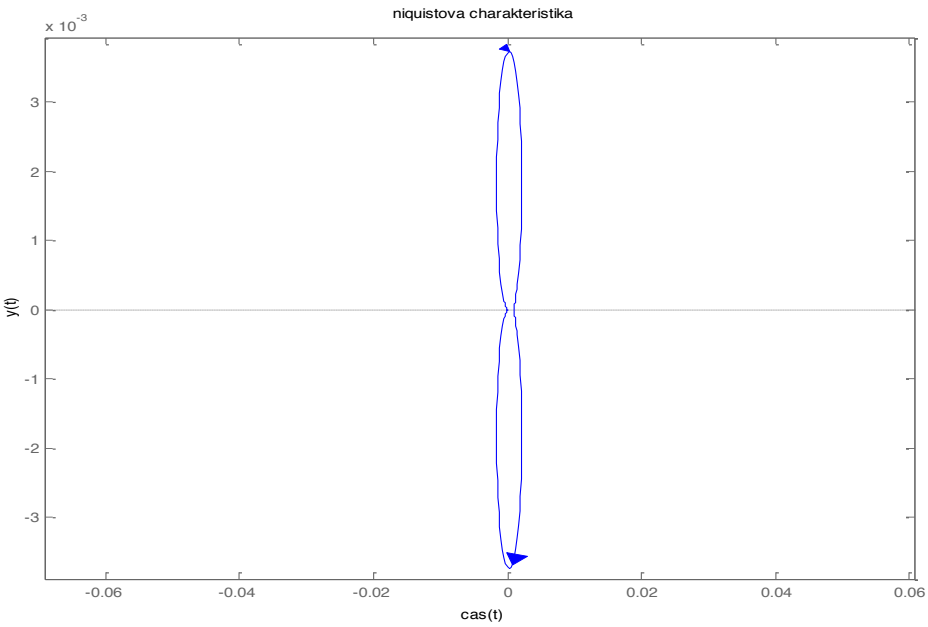


Obrázok 7-15 Prechodová charakteristika modelu otáčok motora

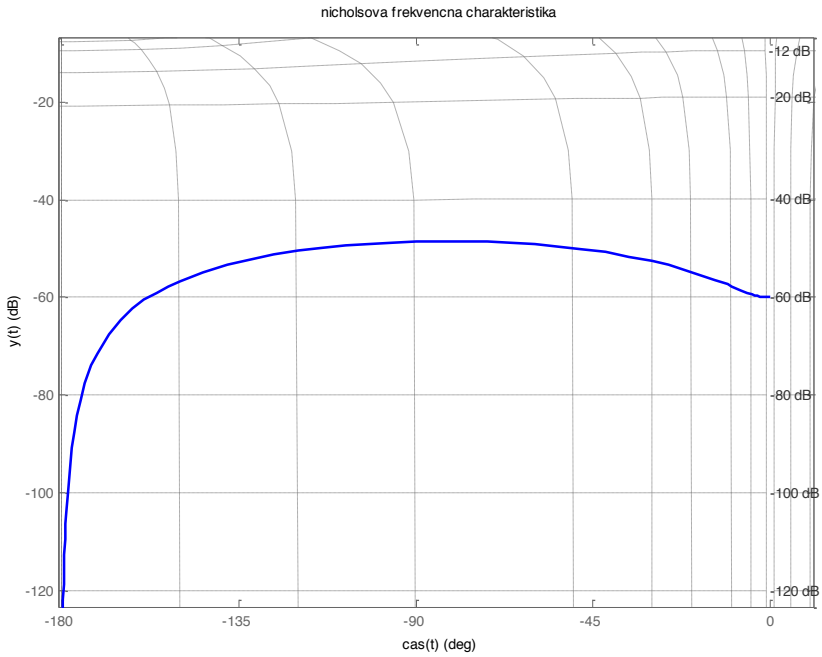


Obrázok 7-16impulzná charakteristika modelu otáčok motora

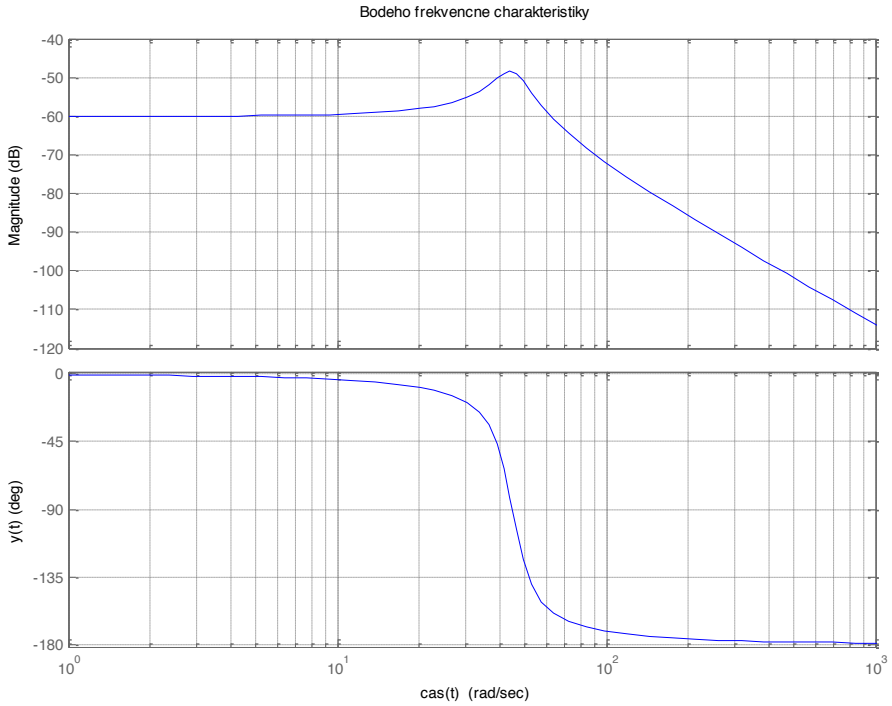
Frekvenčné charakteristiky modelu otáčok motora:



Obrázok 7-17 Nyquistova charakteristika



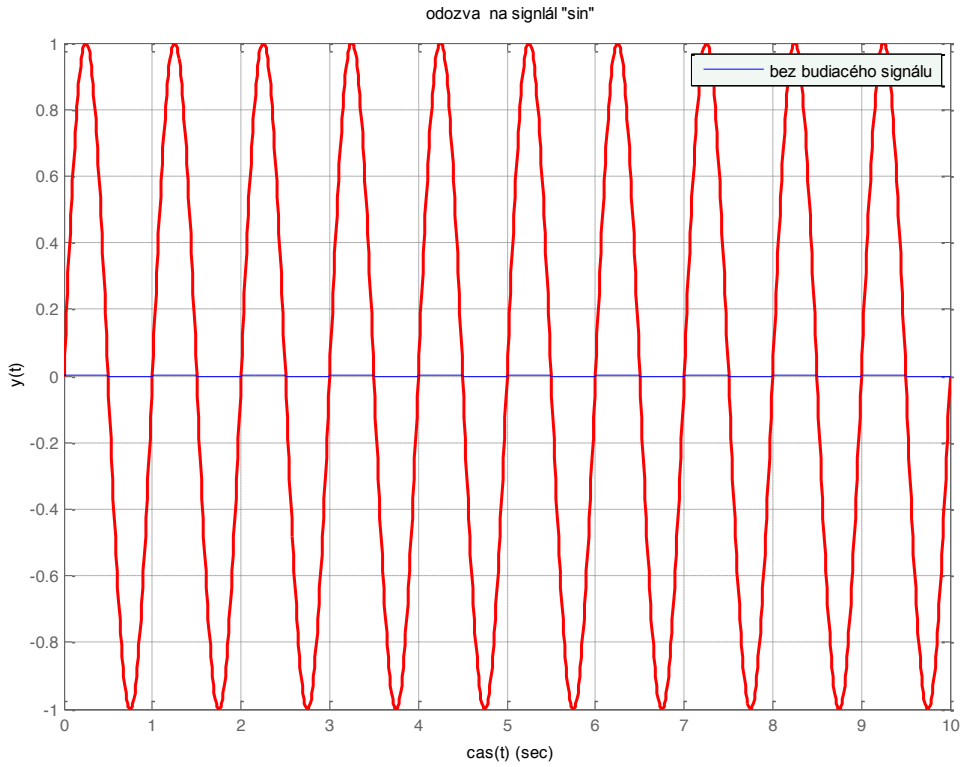
Obrázok 7-18 Nicholsova charakteristika



Obrázok 7-19 Bodeho frekvenčné charakteristiky

Odozva na ľubovoľný signál:

```
function []=odozva()
global sys ;

type=input('Zadaj typ signalu napr. sin, pulse, square: ');
Ton=input('zadaj periodu vzorkovania : ');
Tf=input('zadaj celkovu dobu simulacie : ');
Ts=input('Zadaj vzorkovaci cas Ts : ');

[u,t]=gensig(type,Ton,Tf,Ts)
lsim(sys,u,t)
grid;
title('odozva na lubovolny vstupny signal');
xlabel('cas(t)'); ylabel('y(t)');

hlavny
return
```



Obrázok 7-20 Odozva na sínusový signál

Vyhodnotenie stability systému:

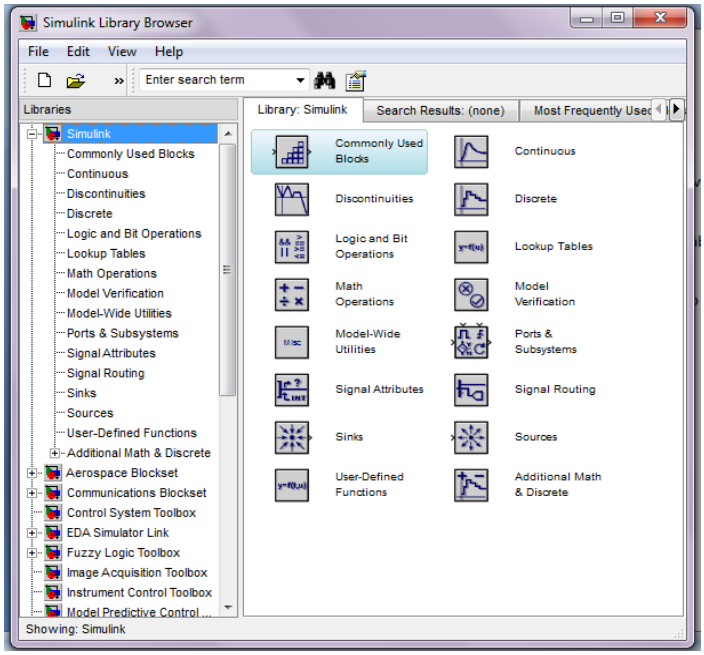
```
%vyhodnotenie stability
function []=stabilita()
global sys;
[num,den]=tfdata(sys,'v');

r=roots(den)
max=size(r);
test=1;

for a= 1:max(:,1)
if r(a) > 0
        test=0;
end
end

if test==0
    disp('nestabilny')
else disp('stabilny alebo na hranici')
end
hlavny
return
```

8 Programové prostredie Simulink

Simulink je nadstavba programového prostredia MATLAB, ktorá využíva :

- prácu s blokmi (z vopred definovaných knižníc)
- vyšetruje správanie DS -> je určený na riešenie (simuláciu prechodových dejov v dynamických lineárnych a nelineárnych systémoch)
- predpoklad znalosti matematického popisu dynamických systémov

Simulink používa pre prácu štandardné menu, pomocou ktorého vieme vytvoriť simulovaný model z blokov, ktoré sú vyberané z knižníc.
Simulink je možné otvoriť len ak máme otvorené programové prostredie MATLAB. Pre aktivovanie je potrebné kliknúť na ikonu  alebo zadať príkaz ***simulink*** do príkazového riadku MATLABu. Po tomto úkone sa nám otvorí samostatné okno (Simulink Library Browser).



Obrázok 8-1 Okno programového prostredia Simulink

(Na ľavom paneli sa nachádzajú knižnice, z ktorých základná sa nazýva Simulink, po rozkliknutí sa nám objavia jednotlivé podknižnice, ktoré sú k dispozícií.)
Programovanie v prostredí Simulink pozostáva:

- výber blokov z knižníc (libraries)
- pripájanie vstupov a výstupov odpovedajúcich blokov (signály modelu)
- zadávanie parametrov blokov
- vytváranie subsystémov

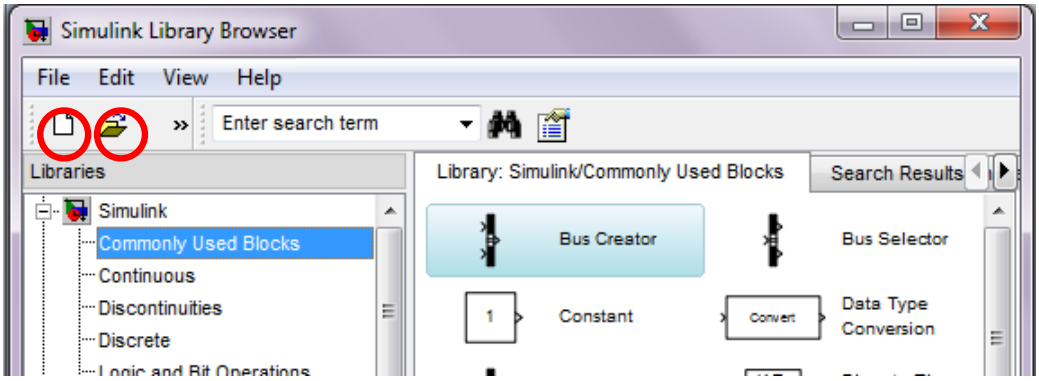
VSTUPNE SIGNÁLY vyberáme:

- z knižnice blokov generujúcich základné typy signálov,
- zo súboru,
- z matíc vopred pripravených v programovom prostredí MATLAB
- z merania v reálnom čase (meracia karta + Real Time Toolbox)

VÝSTUPNÉ SIGNÁLY získavame:

- z blokov typu osciloskop, xy graf ...
- do pracovného priestoru programového prostredia MATLAB
- do súboru

Modely vytvárame po kliknutí na ikonu NEW MODEL alebo OPEN MODEL, ak chceme pokračovať v už začatom modeli. **Modely** sú vytvárané (editované) pomocou myšou riadiacich príkazov -> pre kvalitné a rýchle vytváranie modelu je nutné orientačne poznať všetky typy blokov používaných pre danú triedu systémov.



Obrázok 8-2 Najpoužívanejšie bloky programového prostredia Simulink

Knižnice v Simulink-u:

Sources (zdroje) – generátor vstupov – obsahuje bloky, ktoré nemajú vstupy, pretože predstavujú vstupy vytváraného systému.


Step

blok generujúci skokovú funkciu (až po určitej dobe)

Constant

zdroj konštantnej hodnoty


Clock

zdroj času


Signal Generator

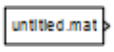
generátor rôznych funkcií, napr. sinus, obdĺžnik, píla, ...


Pulse Generator

blok simulujúci pulzný generátor

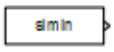

Sine Wave

generátor sínusového signálu



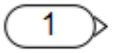
From File

blok pre načítanie údajov zo špecifického súboru *.mat



From Workspace

blok pre načítanie údajov z matice v pracovnom priestore



In1

blok pre tvorbu subsystému, vstupný blok


Band-Limited White Noise

aproximácia bieleho šumu (náhodný signál, ktorý ma rovnaký výkon na všetkých frekvenciách)

Sinks (bloky sledovania výstupov) – bloky, ktoré nemajú výstupy. Slúžia k sledovaniu a záznemu zvolených výstupov modelu pri simulačných experimentoch (ďalšie spracovanie)

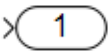
ekvivalent osciloskopu, zobrazenie signálu počas simulácie

numerické zobrazenie hodnôt signálu

ukladanie simulovaných údajov do pracovného priestoru

ukladanie simulovaných údajov do súboru *.mat

ukončenie výpočtu modelu pri dosiahnutí zvolenej hodnoty



blok používaný pri tvorebe subsystému ako výstupný blok

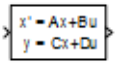
grafické znázornenie signálov t-parametrov

Bloky operácií - bloky, ktoré predstavujú V/V operácie

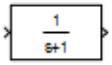
Continuous – obsahuje bloky pre vytváranie spojitých modelov z diferencialnych rovníc

derivačný blok

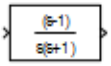
integračný blok



blok pre implementáciu stavového modelu systému



blok pre implementáciu prenosovej funkcie v polynomiálnom tvare



prenosová funkcia v tvare poly/nuly

spojité dopravné oneskorenie

premenlivé dopravné oneskorenie

Discrete – bloky pre vytvorenie diskretnych dynamických modelov

hodnota z minulého integračného kroku

Discontinuities – nespojité systémy, ktorých výstup je nespojitou funkciou vzhľadom na daný vstup (bloky typických nelinearít)

obmedzenie signálu

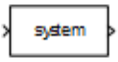
blok modelujúci relé

mrtvá zóna

blok modelujúci hysterézu

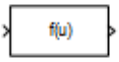
obmedzenie rýchlosti zmeny signálu

User-Defined Functions – používateľom definované funkcie

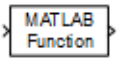


vytváranie vlastnej S-funkcie

príklady vlastných S-funkcií



blok používaný na vytvorenie vlastnej funkcie v programovacom jazyku C



blok odvolavajúci sa na matlabovské funkcie

Math Operations – zápis algoritmickej časti modelu

absolútna hodnota

zosilňovací blok, vynásobenie výstupného signálu konštantou

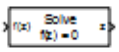
simulačný blok pre 1 ÷ n signálov

odmocnina

súčin vstupných signálov

preddefinovaná matematická funkcia

preddefinovaná trigonometrická funkcia



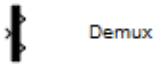
algoritmus slučky, hodnota signálu pre ktorý je f‘ = 0

Logic and Bit Operations

logická operácia AND

Signal Routing

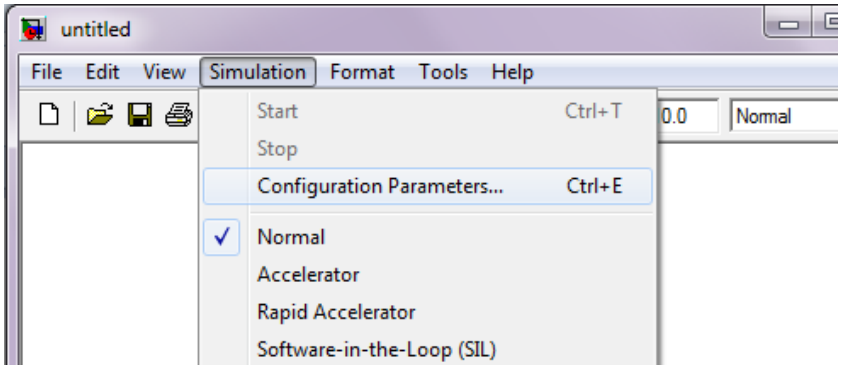
blok spájajúci niekoľko skalárnych/vektorových signálov na jeden vektorový signál



blok rozkladajúci vektorový signál na niekoľko skalárnych/ vektorových Signálov

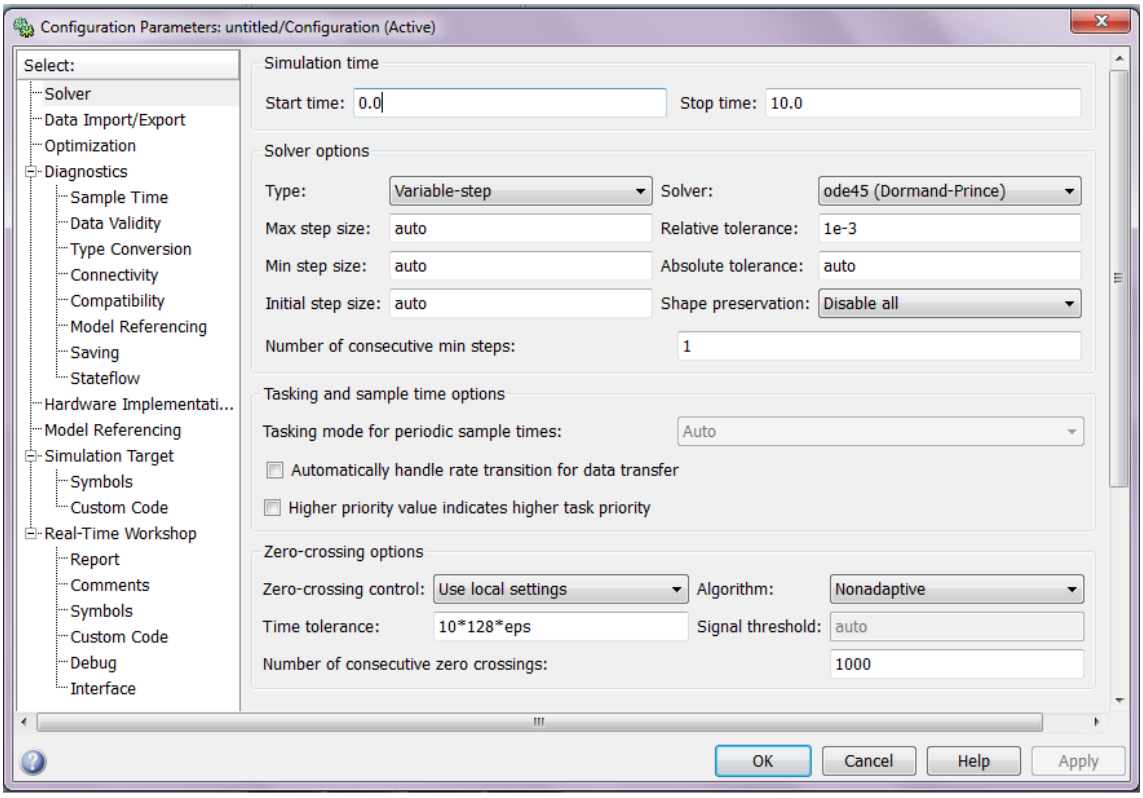
Commonly Used Blocks – najčastejsie používateľom používané bloky
Subsystems – samostatne vyčlenená knižnica. Obsahuje bloky súvisiace s tvorbou subsystému a dovoľujú do simulovaného modelu zahŕňať štandardné programové vybavenie

Pre nastavenie parametrov simulácie vyberieme záložku SIMULATION -> COMFIGURATION PARAMETERS



Obrázok 8-3 Nastavenie parametrov simulácie

alebo použijeme klávesovú skratku ctrl+E, ktorou otvoríme okno pre nastavenie simulácie



Obrázok 8-4 Okno pre nastavenie parametrov simulácie

V položke Solver môžeme nastaviť čas simulácie, voľba metódy riešenia a pod. ...

8.1 Simulácia riešenia LDR v prostredí SIMULINK

Majme LDR 2 rádu : $2y'' + 4y' + 2y = 1(t)$

1. Normovanie LDR

$1y'' + 2y' + 1y = 0{,}5(t)$

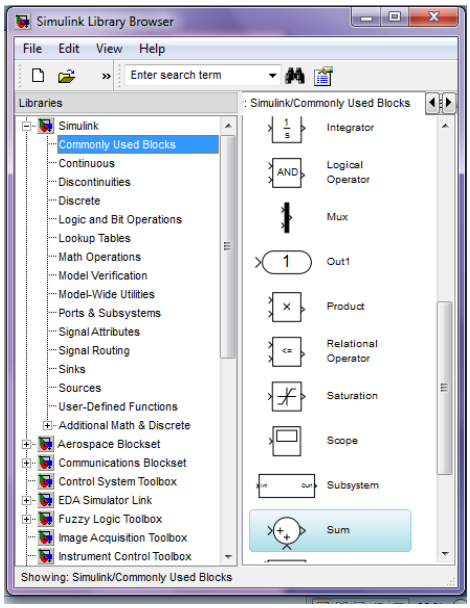
2. Prepis do substitučného kanonického tvaru:

$y' = x_1' = x_2$

$y'' = x_2' = (0{,}5 - 2x_2 - x_1)$

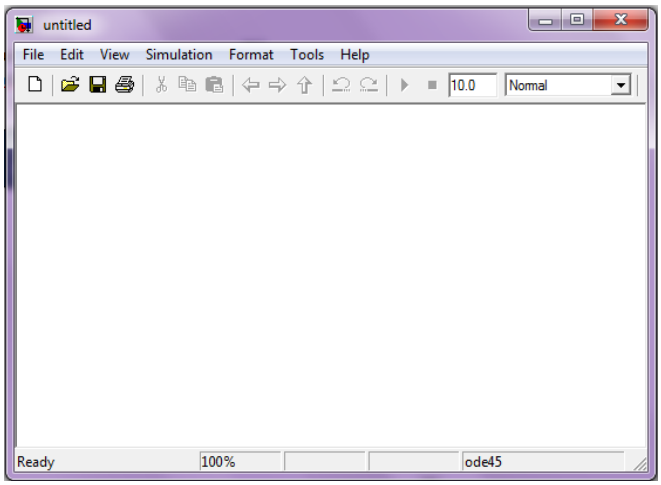
V Simulink-u budeme pracovať iba s $y'' = x_2' = (0{,}5 - 2x_2 - x_1)$

3. Spustíme si grafickú nadstavbu Simulink pomocou ikony  alebo pomocou príkazu ***simulink*** v command window. Otvorí sa následujúce okno:



Obrázok 8-5 Okno programového prostredia Simulink

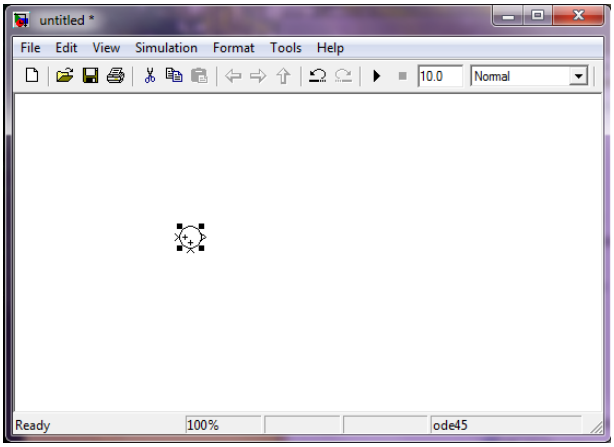
4. Vytvorenie nového podsystému: File -> New -> Model.



Obrázok 8-6 Okno pre vytvorenie modelu v Simulinku

Do tohto okna budeme postupne presúvať a spájať jednotlivé bloky, a následne nastavovať ich parametre.

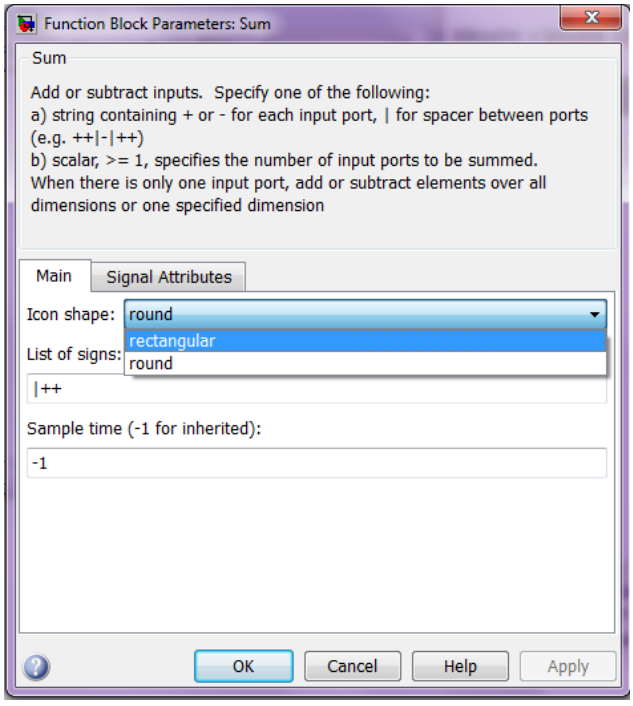
5. Vložíme blok **sumátor** jednoduchým kliknutím v ***Simulink Library Browser*** na konkrétny blok a potiahneme ho do nového okna pre model.



Obrázok 8-7 Blok sumátor

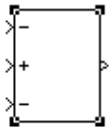
6. Blok **sumátor** je možné zmeniť na štvoruholníkový kliknutím na **sumátor** pravým tlačidlom myši a následným vybratím položky ***Sum Parameters***, alebo dvojklikom na blok.

Následne sa zobrazí :



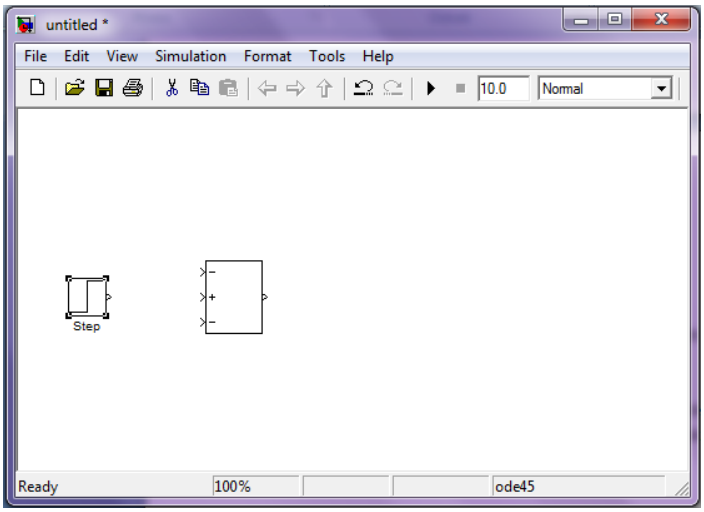
Obrázok 8-8 Definovanie vstupov do sumátora

V tomto prípade vyberieme **rectangular** a vstupy budú - + - a dostaneme



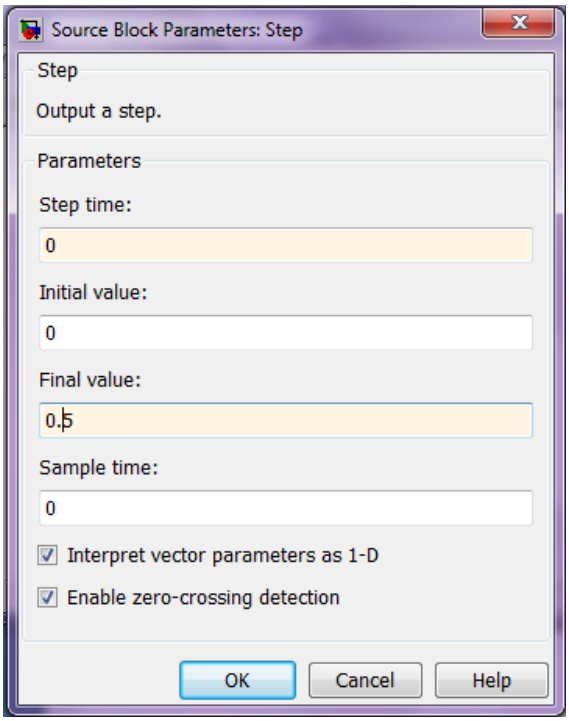
Výber potvrdíme tlačidlom OK.

7. Druhý blok, ktorý si pridáme je **vstup do systému**, ktorý nájdeme pod knižnicou **Sources** a vyberieme blok simulujúci jednotkový skok (step).



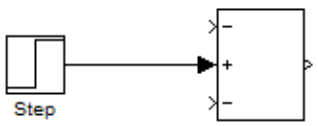
Obrázok 8-9 Blok pre vstup do systému (step)

8. Dvojklikom na blok **step** alebo kliknutím pravým tlačidlom na blok a výber ***Step Parameters*** otvoríme nastavenia parametrov tohto bloku.

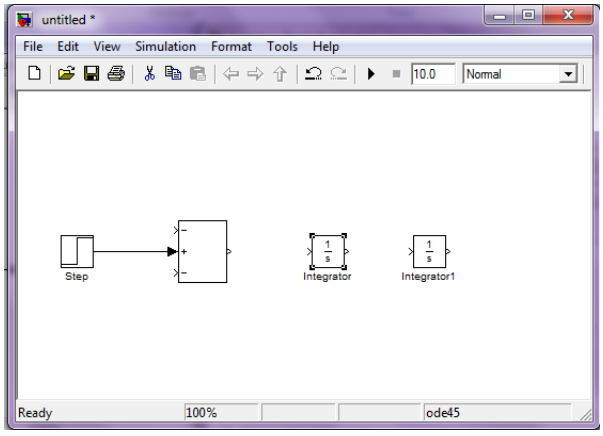


Obrázok 8-10 Inicializácia parametrov pre funkciu step

9. Bloky spojíme kliknutím na výstup bloku step a vstup sumátora a spojíme čiarou.

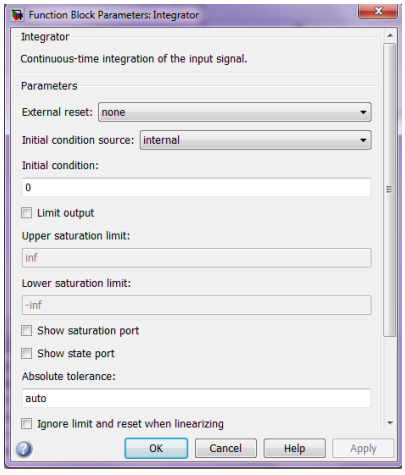


10. Ďalšie bloky, ktoré pridáme, budú dva integrátory. Integrátory nájdeme v položke ***Continuous*** alebo v položke ***Commonly Used Blocks.***
    - ⇨ Prvý integrátor použijeme pre integráciu x_2' na $x_2 = x_1'$
    - ⇨ Druhý na integráciu získaného x_2 na x_1



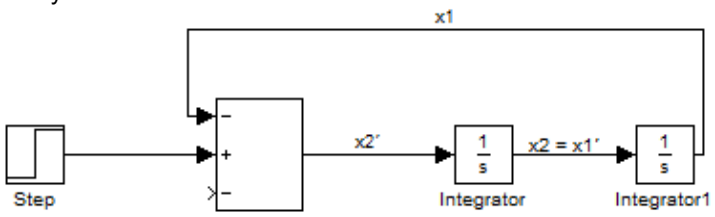
Obrázok 8-11 Integrátory pre zostavenie modelu v Simulinku

11. V jednotlivých integrátoroch buď dvojklikom na integrátor alebo kliknutím pravým tlačidlom myši na integrátor a následným výberom položky ***Block Properties*** sa otvorí okno, kde môžeme nastaviť napríklad počiatočné podmienky pre integráciu.

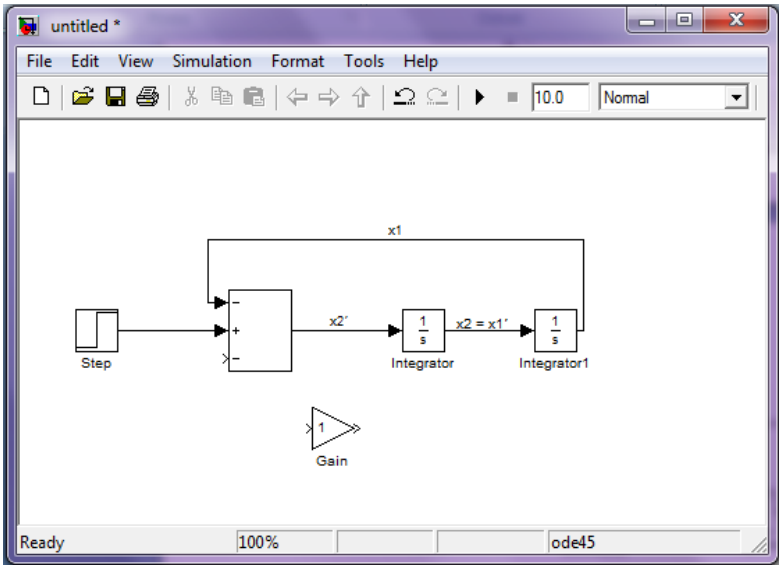


Obrázok 8-12 Nastavenie počiatočných podmienok pre integrátory

12. Pospájame bloky



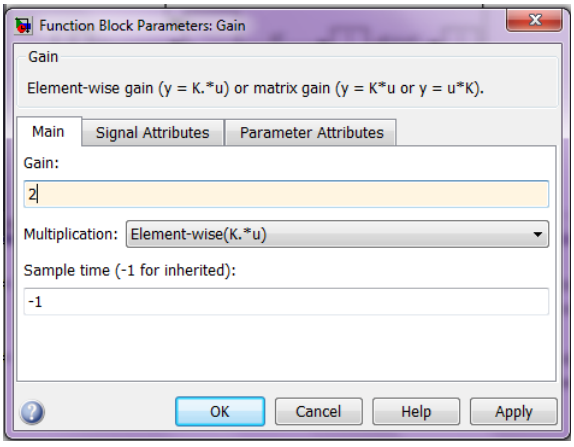
13. Potrebujeme ešte zosilniť signál x_2 a preto pridáme blok **Gain** z položky ***Math Operations*** alebo ***Commonly Used Blocks.***



Obrázok 8-13 Pospájanie blokov v modeli pre riešenie LDR

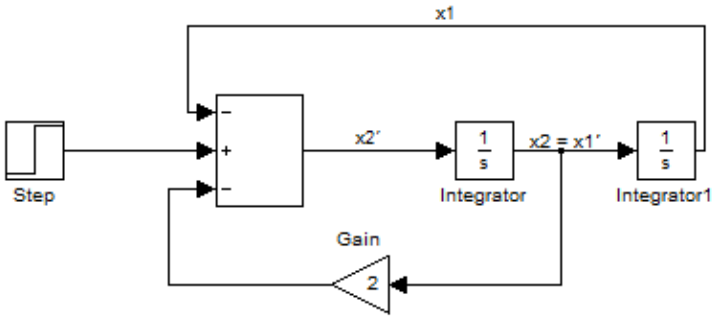
⇨ Pootočenie jednotlivých blokov je možné po kliknutí na blok ktorý chceme otočiť a stlačení kláves **CTRL+R** alebo kliknutím pravým tlačidlom na objekt a výberom položky **Format -> Rotate Block -> Clockwise**

⇨ Dvojklikom na zosilnenie **gain**, resp. kliknutím pravým tlačidlom myši na objekt a výberom položky ***Gain Properties*** sa otvorí okno, kde je možné napríklad zmeniť zosilnenie.

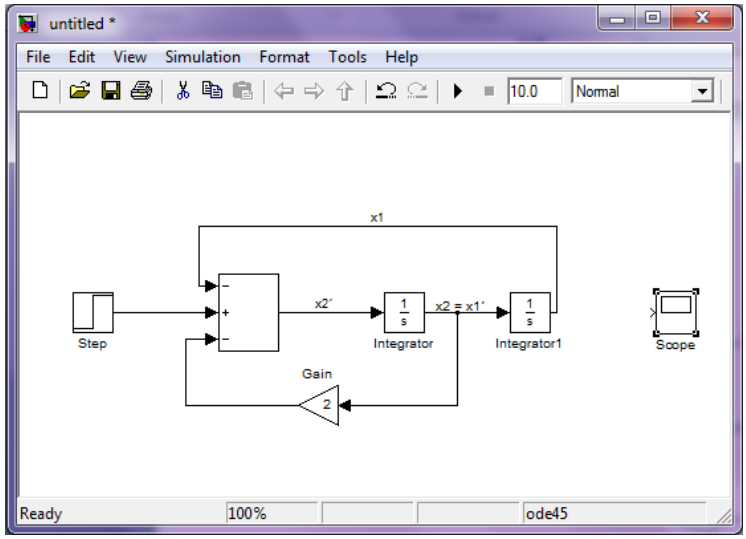


Obrázok 8-14 Nastavenie parametrov bloku gain

Po pospájaní jednotlivých blokov dostávame model v Simulinku na riešenie LDR:

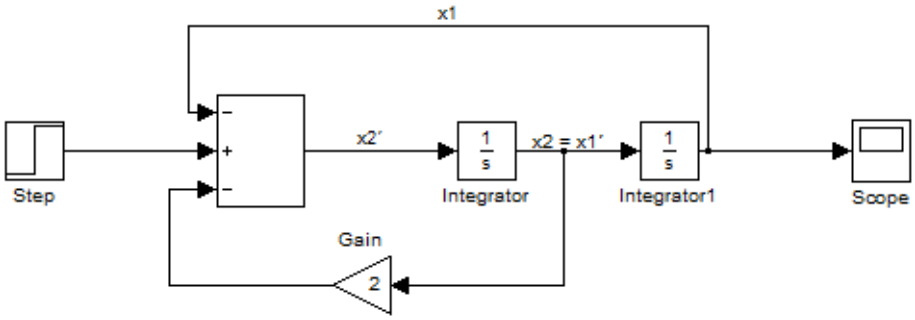


14. Nakoniec ak chceme vykresliť priebeh riešenia zadanej diferenciánej rovnice musíme pridať blok **Scope**. Blok Scope nájdeme v položke ***Sinks*** alebo ***Commonly Used Blocks.***

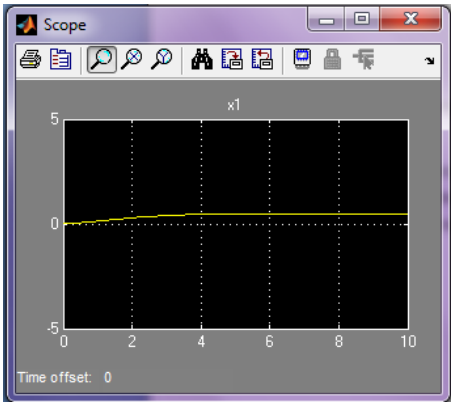


Obrázok 8-15 Model pre riešenie LDR s pridaním bloku Scope

⇨ Pripojenie výstupu integrátora integrator1 na vstup bloku **Scope** s cieľom vykresliť priebeh riešenia LDR:

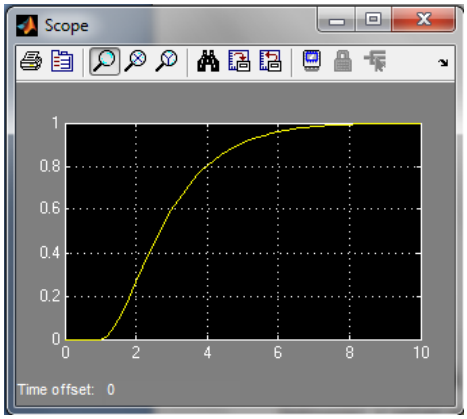


15. Pre vykreslenie priebehu riešenia musíme spustiť simuláciu a to pomocou ikony  .
16. Následne dvojklikom na Scope sa otvorí okno **figure** a v ňom priebeh simulovaného systému (lineárnej diferenciálnej rovnice)



Obrázok 8-16 Časový priebeh riešenia LDR

17. V bloku Scope kliknutím na ikonu  nám automatický priblíži nábeh vykreslenej simulácie (automatické škálovanie). V našom prípade to bude vyzerať:



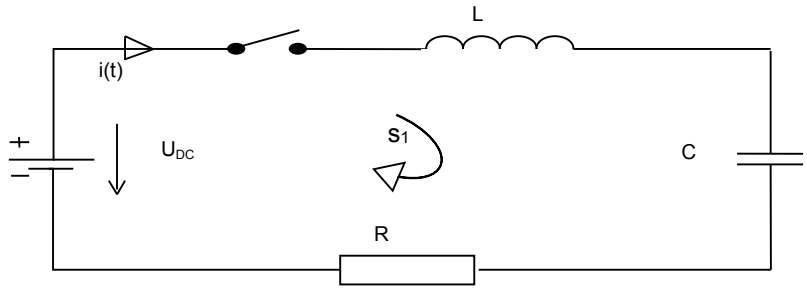
Obrázok 8-17 Škálovanie časového priebehu riešenia LDR v Simulinku

Ak chceme toto škálovanie použiť aj pre ďalšie simulácie klikneme na ikonu  , ktorá nám uloží aktuálne nastavenie osí. Teraz pri nasledujúcom spustení simulácie nám automaticky otvorí s uloženými nastaveniami osi. Opakom, teda na zrušenie uloženia osí slúži ikona  .

8.2 Simulácia modelov fyzikálnych systémov v prostredí SIMULINK

PRÍKLAD 1 – NABÍJANIE KONDENZÁTORA

Zostavte na základe matematického modelu RLC obvodu simulačný model v programovom prostredí Simulink a vypočítajte časový priebeh prúdu i(t) a časový priebeh napätia $u_C(t)$.



$$u_R(t) = Ri(t)$$

$$u_L(t) = L\frac{di(t)}{dt}$$

$$u_C(t) = \frac{1}{C}\int_0^t i(\tau)d\tau$$

Na základe vyššie uvedených vzťahov pre napätia na jednotlivých elektronických prvkoch obvodu vieme matematický model podľa 2.Kirchhoffovho zákona, ktorý hovorí, že súčet úbytkov napätí v uzavretej slučke obvodu je rovný nule. Znázornený RLC obvod obsahuje práve jednu slučku s_1, pre ktorú platí:

$$L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}\int i(t)dt = u_{DC}(t)$$

Zavedieme substitúciu pre prúd pretekajúci obvodom:

$$i(t) = C * \frac{du_c(t)}{dt}$$

Pomocou tejto substitúcie eliminujene integrál na pravej strane a získame výslednú diferenciálnu rovnicu 2. rádu popisujúcu nabíjanie kondenzátora cez technickú cievku:

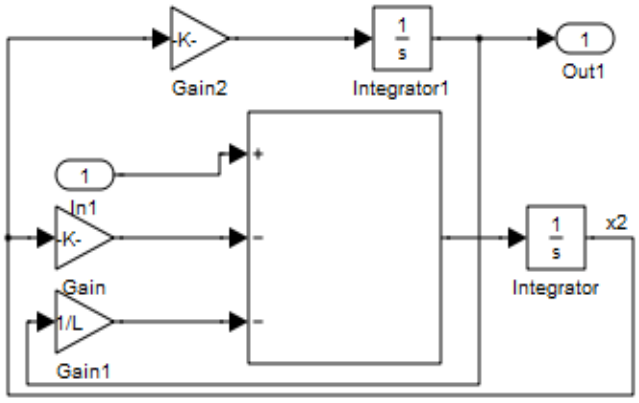
$$CL * \frac{d^2u_C(t)}{dt^2} + CR\frac{du_C(t)}{dt} + u_C(t) = u_{DC}(t)$$

Túto diferenciálnu rovnicu zapíšeme do substitučného kanonického tvaru a následne zostavíme model v Simulinku:

$$x_1'(t) = x_2(t)$$

$$x_2'(t) = \frac{1}{CL}u_{DC}(t) - \frac{1}{CL}x_1(t) - \frac{R}{L}x_2(t)$$

Vytvorený model pre nabíjanie kondenzátora cez technickú cievku v prostredí Simulink



PRÍKLAD 2 - HYDRAULIKA

Z nádoby o priereze $S = 2\,m^2$a výške $h = 2\,m$, vyteká voda otvorom na dne nádoby o priereze $S_0 = 0{,}001\ m^2$. Hydraulický súčiniteľ je $\alpha = 0{,}94$. Výška hladiny na začiatku sledovania je $h = 0{,}1\,m$. Zistite, ako sa bude meniť v čase výška hladiny $h(t)$ a vytekajúce množstvo Q_0, keď nebude vždy 1 minútu nič pritekať $Q = 0$ a následne 2 minúty bude pritekať množstvo $0{,}017\ m^3 s^{-1}$.

Hmotnostá bilancia systému: $\rho Q = \rho Q_0 = \frac{d(\rho S_N h(t))}{dt}$

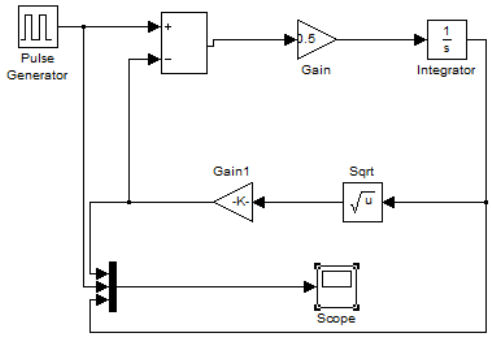
Torriceliho vzťah : $v_0 = \alpha\sqrt{2gh} \rightarrow \rho Q_0 = \rho v_0 S_0$

Výsledná nelineárna diferenciálna rovnica - závislosť h(t) na prítokového množstva Q:

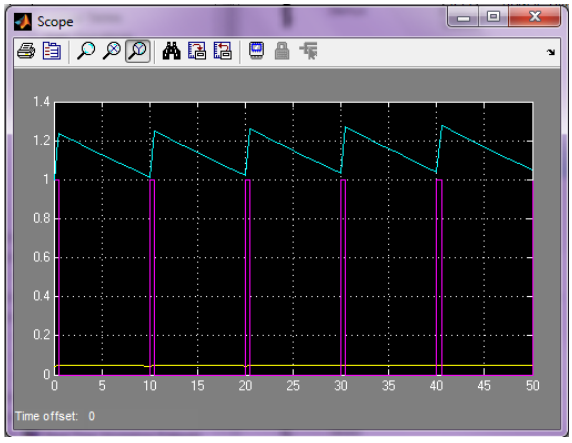
$$S_N \frac{dh(t)}{dt} + \alpha S_0\sqrt{2gh} = Q \qquad \text{h(t-0) = 1; h } \in <0,2)$$

$$h(t) = \int_0^t \frac{1}{s}(Q - \underbrace{\alpha S_0\sqrt{2g}}_{\text{konšt.}}\sqrt{h(t)})dt$$

Riešenie modelu hydrauliky – jednej nádoby v prostredí Simulink:



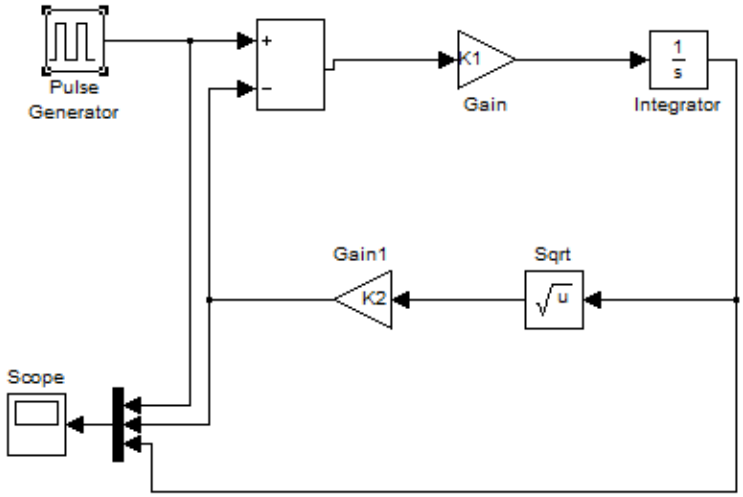
Obrázok 8-18 Model hydraulického systému



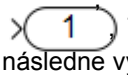
Obrázok 8-19 Časový priebeh výšky hladiny *h₁(t)*

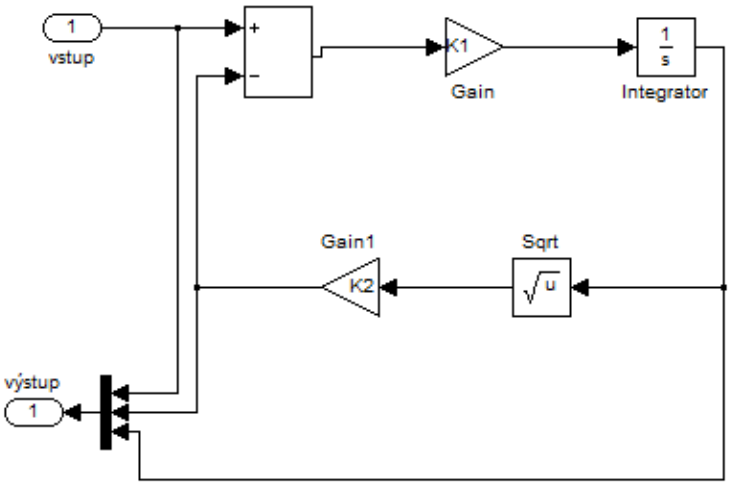
8.3 Tvorba subsystémov a maskovanie

Uvažujme simulačný model vytvorený pomocou blokov v Simulink-u. V tomto prípade je to naprogramovaný matematický model nádoby, z ktorej vyteká prerušovane kvapalina otvorom na dne (vždy 1 minúty nič nepriteká a následne 2 minúty kvapalina priteká).

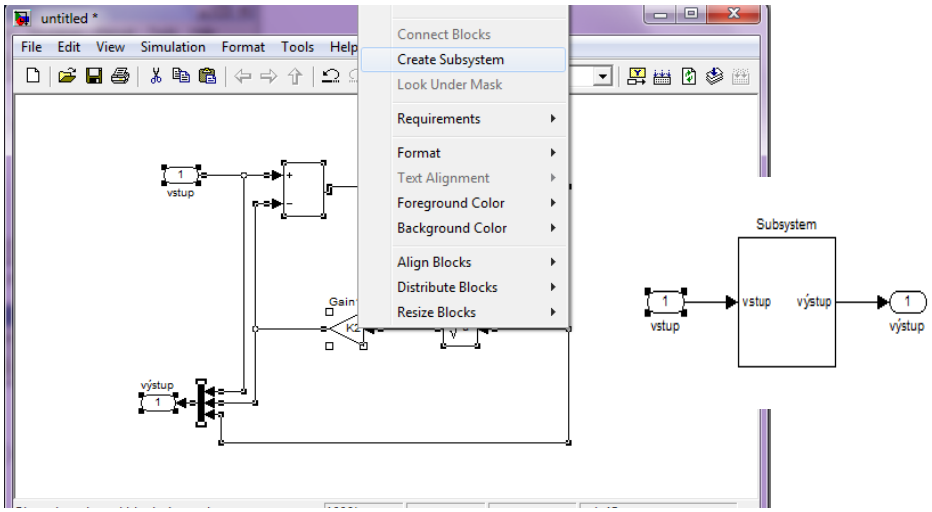


Úloha: Vytvorte subsystém z daného simulačného modelu v Simulink-u.

1. Všetky vstupy zameníme blokmi In  a všetky výstupy zameníme blokmi Out  Tieto bloky môžeme pomenovať a názvy sa následne prenesú aj do subsystému, ktorý následne vytvoríme .



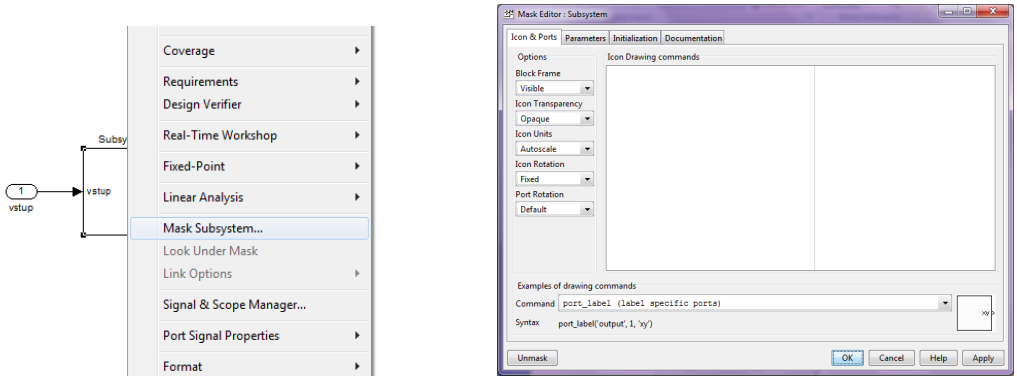
2. Teraz označíme všetky bloky systému, stlačíme pravé tlačidlo myši a vyberieme položku **Create Subsystem**



Obrázok 8-20 Vytvorenie subsystému

Úloha: Vytvorte masku daného subsystému.

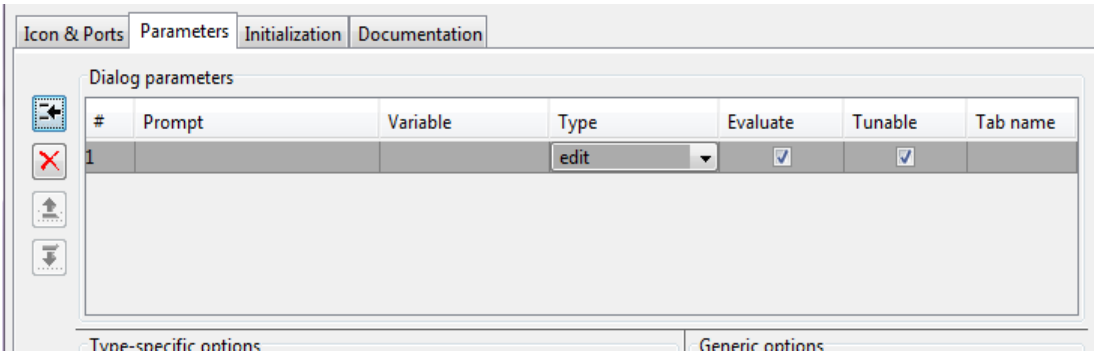
1. Kliknutím pravým tlačidlom myši na **subsystém** a následným výberom položky **Mask podsystem** sa otvorí okno:



2. V záložke ***parameters*** je možné parametre, ktoré chceme meniť zapísať, a následne pri kliknutí na **subsystém** sa otvorí okno, kde si tieto parametre budeme môcť nastavovať.

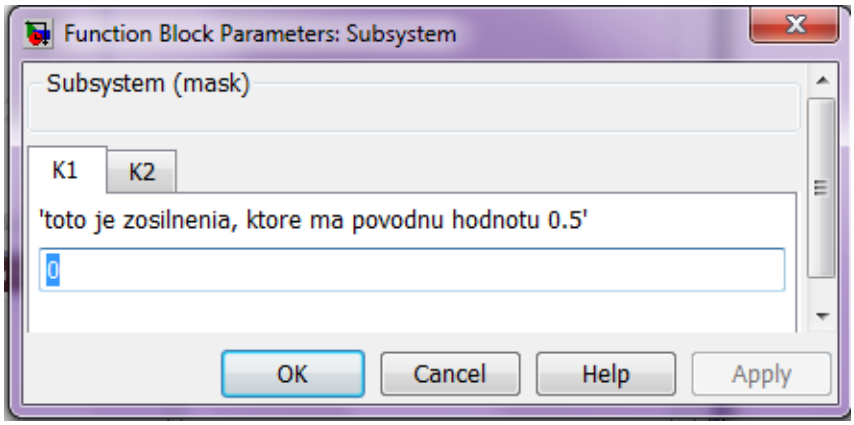
Napríklad v našom systéme si môžeme meniť **zosilnenia**, ak pri tvorení **subsystému** sme nekonkretizovali **hodnotu zosilnenia** ale zadali napríklad len **K**, potom postupujeme nasledovné:

a) V záložke *parameters* klikneme na ikonu  , a vytvorí sa nám 1 riadok v tabuľke.



Prompt – obsahuje popis tejto hodnoty, pozor musí byť zapísaný v apostrofoch, pretože je to reťazec
Variable – do tejto položky zapíšeme názov z systému, v našom prípade K
type – vyberieme zobrazenie
Tab name – vytvorí záložky s názvami pre jednotlivé premenné

Následne klikneme na *Apply*, ak chceme pridať ďalšiu položku zopakujeme postup, inak klikneme na *OK*.
Ak sme správne vytvorili premenné po kliknutí na subsystém sa nám zobrazí okno



V tomto okne si nastavíme požadované hodnoty a s nimi vykonáme simuláciu.

8.4 Zadanie č.5 : Simulácia LDR/NDR a modelu fyzikálneho systému v prostredí SIMULINK

ZADANIE: *Naprogramujte simulačnú schému (model) v prostredí Simulink na riešenie:*

a) *lineárnej diferenciálnej rovnice (zadanie č. 2) s uvažovaním definovaného budiaceho signálu,*
b) *nelineárnej diferenciálnej rovnice (zadanie č. 3) s uvažovaním definovaného budiaceho signálu,*
c) *odozvy fyzikálneho modelu (zadanie č. 4) na definovaný budiaci signál. So simulačným modelom pracujte ako so subsystémom a parametre nech sú zadávané v maske.*

- **Majme zadanú lineárnu diferenciálnu rovnicu tvaru:**

$$y''(t) + 2y'(t) + y(t) = t$$

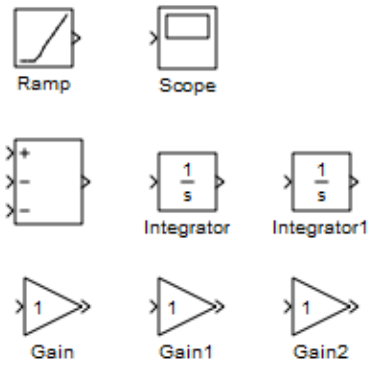
Počiatočné podmienky pre riešenie tejto lineárnej diferenciálnej rovnice sú $y(0) = y'(0) = 0$.

1. Zadanú diferenciálnu rovnicu 2. rádu rozdelíme na dve diferenciálne rovnice 1. rádu – teda prevedieme diferenciálnu rovnicu 2. rádu do substitučného kanonického tvaru:

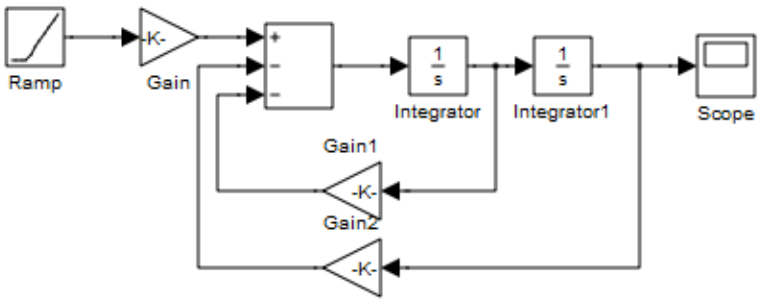
$$x_2 = x_1' = y'(t)$$
$$x_2' = y''(t) = t - 2 * x_2 - x_1$$

Substitúcia: $x_1 = y(t)$

2. Ako je vidieť zo substitučného kanonického tvaru, pre vytvorenie tejto diferenciálnej rovnice potrebujeme 2xintegrátor,1x zosilnenie, 1x súčtový člen , vstupný signál, v tomto prípade použijeme RAMP a pre vykreslenie získaného priebehu potrebujeme osciloskop. Ak však chceme vytvoriť všeobecné riešenie, kde používateľ má možnosť zadať vlastné hodnoty koeficientov a_0, a_1 a a_2 potom použijeme blok zosilnenia 3x.

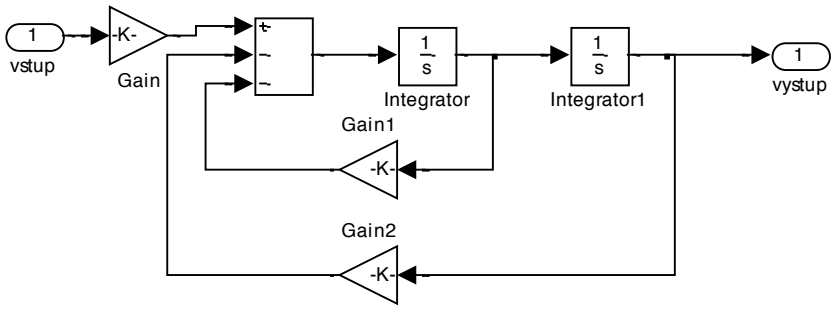


3. Pospájaním týchto blokov podľa vytvoreného substitučného kanonického tvaru získame riešenie zadanej lineárnej diferenciálnej rovnice v grafickom prostredí Simulink.

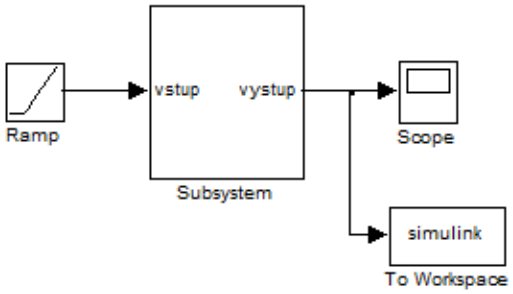


V tejto schéme Gain 1 predstavuje hodnotu a1/ a2 a Gain 2 hodnotu a0/a2 a Gain 1/a2.

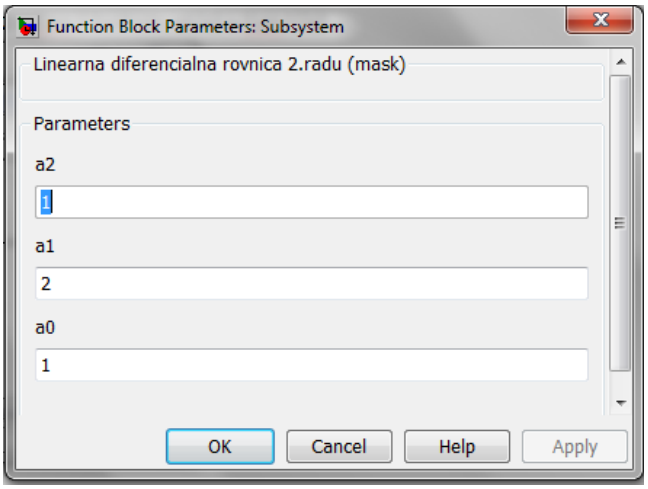
4. Zo získaného modelu vytvoríme subsystém tak, že za vstup (ramp) a výstup (scope) použijeme bloky IN a OUT.



5. Z tejto schémy modelu si následne vytvoríme subsystém a bloky IN a OUT zmeníme späť na Ramp a Scope, prípadne môžeme pridať aj blok To Workspace, ktorý vypočítané hodnoty zapíše do Workspace, kde s nimi môžeme ďalej pracovať .

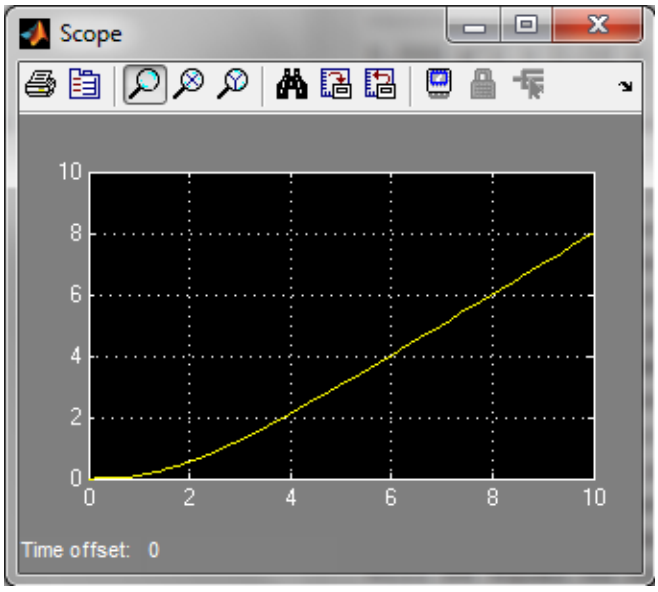


6. Parametre vytvoreného subsystému zamaskujeme, čím pri kliknutí na subsystém sa nám otvorí okno, v ktorom je možné nastavovať parametre a0, a1,



Obrázok 8-21 Nastavenie príslušných parametrov pre LDR v Simulinku

7. Výsledné grafické riešenie bude vyzerať:



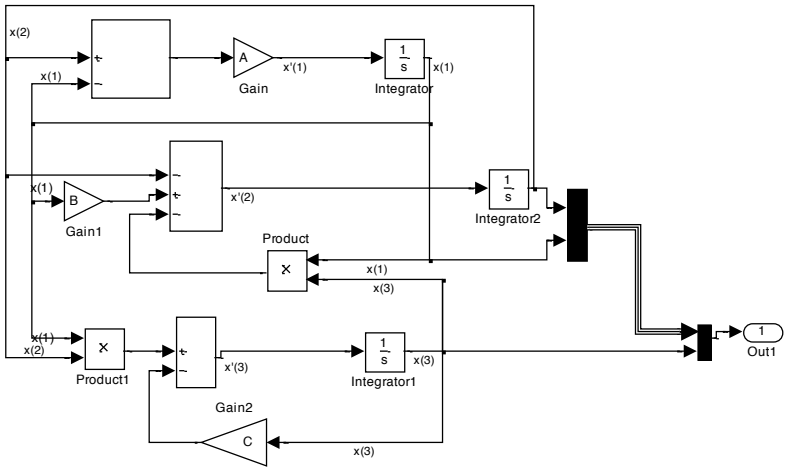
Obrázok 8-22 Časový priebeh riešenia $x_1(t)$ ako záznam z bloku Scope

- **Majme sústavu 3 diferenciálnych rovníc 1. rádu, ktoré predstavujú turbulencie v kvapalinách:**

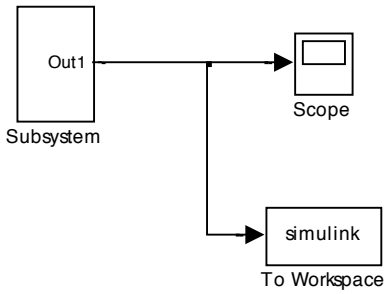
$$x_1'(t) = A(x_2(t) - x_1(t))$$
$$x_2'(t) = Bx_1(t) - x_2(t) - x_1(t) * x_3(t)$$
$$x_3'(t) = x_1(t)x_2(t) - Cx_3(t)$$

1. Pretože všetky rovnice sú 1. rádu, nepotrebujeme vytvárať substitučný kanonický tvar a môžeme začať rovno vytvárať model v prostredí Simulink:

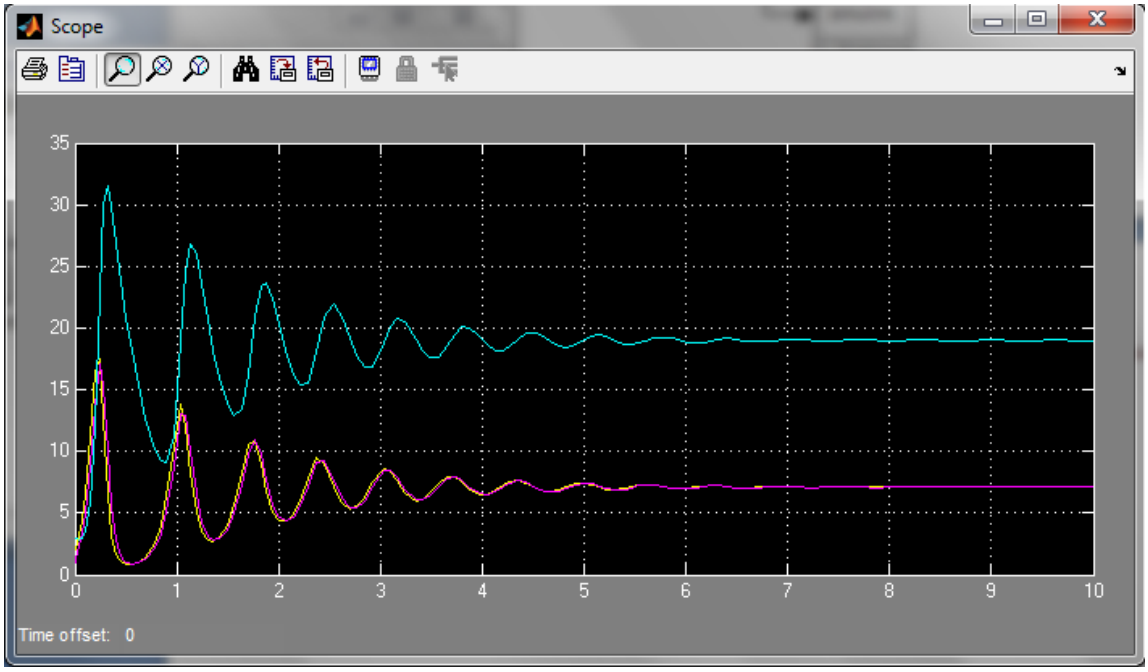
Postup pri vytváraní modelu bude zhodný z postupom vytvárania modelu lineárnej diferenciálnej rovnice. Pre tvorbu subsystému si zameníme iba blok výstupu (Scope) za blok Out, vstup do systému v tomto prípade neexistuje, pretože do týchto diferenciálnych rovníc nevstupuje žiadna budiaca sila:



2. Z takto vytvoreného modelu vytvoríme subsystém a ten následne zamaskujeme:



3. Spustením simulácie a nahliadnutím na oscilátor (scope) uvidíme:



Obrázok 8-23 Riešenie NDR z modelu v Simulinku

- **Majme matematický model otáčok motora. Našou úlohou je vytvoriť simulačný model otáčok motora pomocou programového nástroja Simulink.**

1. Tento model je popísaný diferenciálnou rovnicou 2. rádu tvaru:

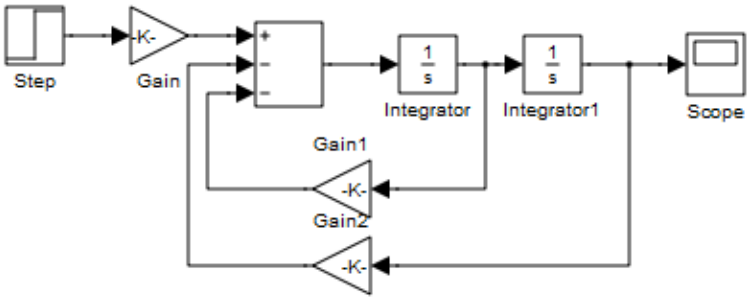
$$L * J * \frac{d^2\omega}{dt^2} + (L * B + R * J) * \frac{d\omega}{dt} + (R * B + C_u{}^2) * \omega = U(t) * C_u$$

Pre riešenie tohto modelu potrebujeme diferenciálnu rovnicu prepísať do substitučného kanonického tvaru, ktorý bude mať nasledovný tvar:

$$x_1{}' = \omega' = x_2$$
$$x_2{}' = \omega'' = \frac{1}{LJ}\left(C_u U(t) - (L * B + R * J)x_2 - (RB + C_u{}^2)x_1\right)$$

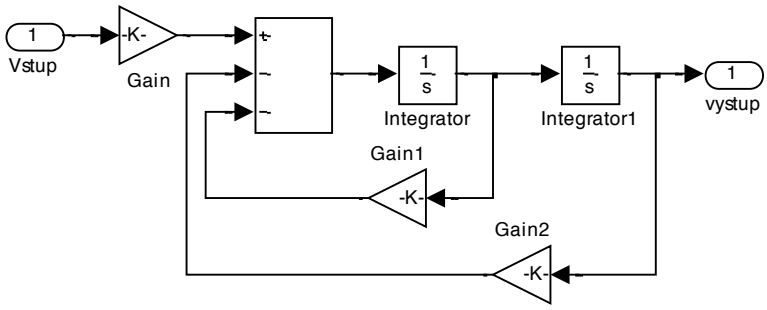
Substitúcia:
$x_1 = \omega$

2. Následne vytvoríme schému modelu, ktorá bude mať nasledovný tvar:

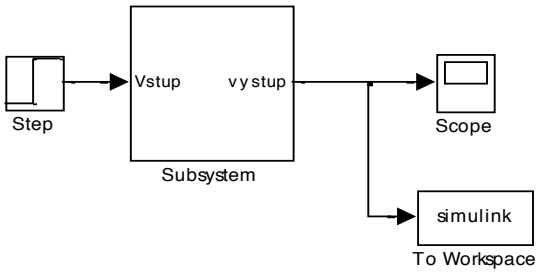


V tejto schéme Gain predstavuje hodnotu Cu/(J*L), Gain 1 hodnotu (B/J)+(R/L) a hodnota Gain 2 je daná ako ((Cu*Cu)/(J*L))+((B*R)/(J*L)).

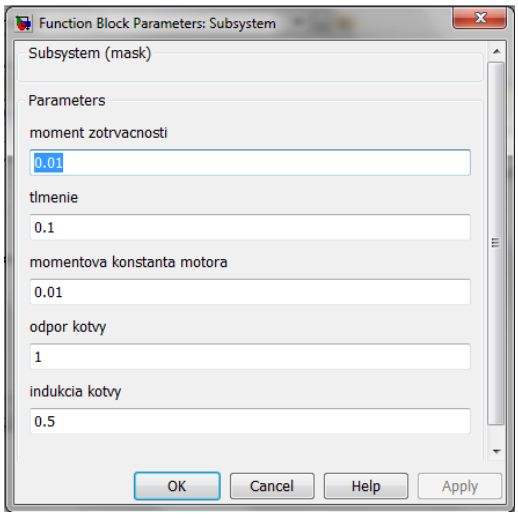
3. Pre vytvorenie subsystému potrebujeme dosadiť za vstup (Step) a výstup (Scope) bloky In a Out.



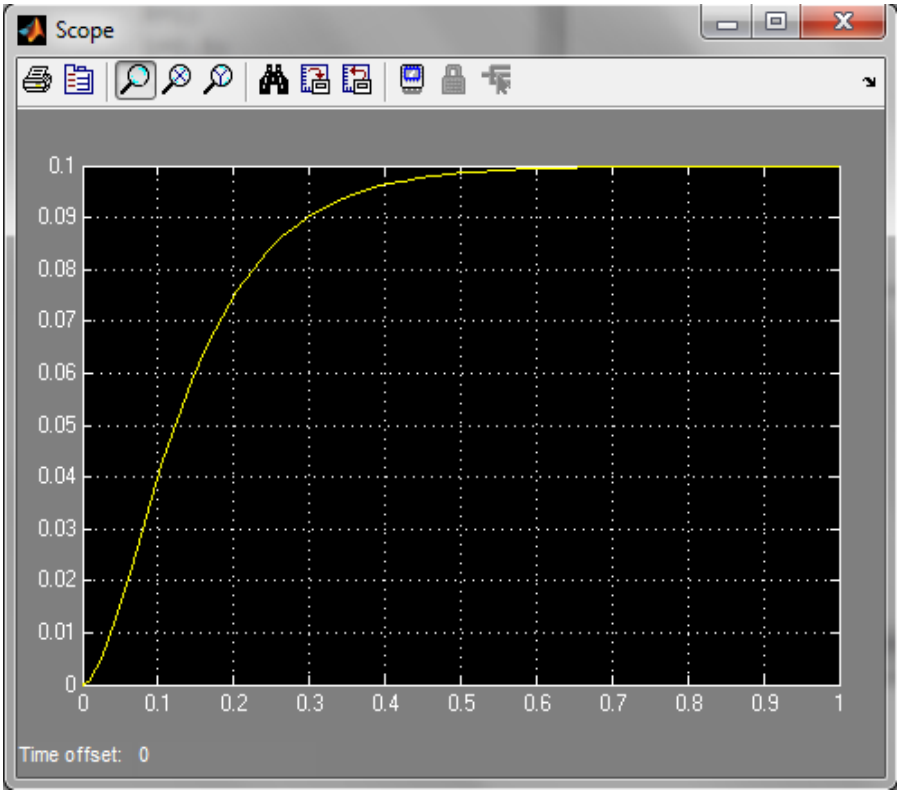
4. Takto vytvorený subsystém zamaskujeme a máme výsledný model otáčok motora:



Po dvojkliku na subsystém sa objaví blok pre nastavenie jednotlivých parametrov:



5. výstup získaný z oscilátora (scope)



Obrázok 8-24 Časový priebeh otáčok motora získaný zo simulačného modelu v Simulinku pri nastavených parametroch modelu a simulácie

8.5 Zadanie č. 6: Návrh a simulačné overenie algoritmu PID pre riadenie otáčok jednosmerného motora

ZADANIE: Navrhnite algoritmus riadenia (PI, PD, PID) dvoma metódami syntézy pre Vami zvolený fyzikálny model dynamického systému (viď zadanie č. 4), pričom rád dynamického systému je $n \leq 3$.

Navrhnutý algoritmus riadenia overte v spätnoväzobnej štruktúre uzavretého regulačného obvodu (URO) pomocou blokov knižníc Simulink, kde model vyskladajte z blokov namiesto použitia bloku Transfer Fanction pri skokovej zmene vstupov URO: $w(t) = 1(t) pre\ t = 0(s)$ a $z(t) = 0.39(t)\ pre\ t = 0.5(s)$.

OBSAH ZADANIA:

1. Analytický výpočet parametrov PID algoritmu zvolenými metódami syntézy pre simulačný model dynamického systému.
2. Overenie PID algoritmu v spätnoväzobnej štruktúre URO v Simulinku.
3. Grafické priebehy regulovanej veličiny:

$y(t)\ pre\ w = 1, z = 0$

$y(t)\ pre\ w = 0, z = 1(t)$

$y(t)\ pre\ w(t) = 1(t), z = 0.3(t) pre\ t = 0.5s$

4. Graf odozvy systému na jednotkový skok.

Majme zadanú prenosovú funkciu otáčok motora
:

$$F(s) = \frac{C_u}{s^2(JL) + s(BL + JR) + ({C_u}^2 + BR)}$$

Ak dosadíme hodnoty $J = 0{,}01\ kg.m^2$, $B = 0{,}1\ N.m.s$, $R = 1\ \Omega$, $L = 0{,}5H$, $C_u = 0{,}01\ {}^{N.m}/_{Amp}$

Získame obrazový prenos sústavy:

$$F_s(s) = \frac{1}{0{,}5s^2 + 6s + 10{,}0{,}1}$$

Pre riadenie zadanej sústavy si zvolíme PI regulátor, ktorého obrazový prenos vyzerá nasledovne:

$$F_R(s) = K * \left(1 + \frac{1}{T_i s}\right)$$

<u>Zvolíme si riešenie metódou Graham-Lathrop</u>

1. Potrebujeme získať charakteristickú rovnicu, ktorá má tvar 1+$F_R(s)F_s(s)$:

$$1 + \frac{T_i K s + K}{T_i s} * \frac{1}{0{,}5s^2 + 6s + 10{,}0{,}1} = 0$$

Po roznásobení a úprave získame charakteristickú rovnicu v tvare:

$$\frac{0{,}5T_i s^3 + 6T_i s^2 + 10{,}01T_i s + KT_i s + K}{0{,}5T_i s^3 + 6T_i s^2 + 10{,}01T_i s} = 0$$

Menovateľ je rovný nule. Zavedieme substitúciu:

$$\frac{K}{T_i} = I$$

A následne ešte čitateľa charakteristickej rovnice upravíme do tvaru:

$$s^3 + 12s^2 + 2 * (10{,}01 + K)s + 2I = 0$$

Z tabuľky štandardných tvarov charakteristických rovníc podľa Graham-Lathropa vyčítame odpovedajúcu charakteristickú rovnicu, ktorá musí obsahovať tak ako naša charakteristická rovnica 3 mocninu pri najvyššom laplaceovom operátorovi :

$$s^3 + 1{,}75s^2\omega_0 + 2{,}15s{\omega_0}^2 + {\omega_0}^3$$

Z týchto dvoch rovníc potrebujeme získať neznáme hodnoty ω_0, K, I

$$s^3 : 1 = 1$$

$$s^2 : 12 = 1{,}75\omega_0$$

$$\omega_0 = 5{,}857$$

$$s : 20{,}02 + 2K = 2{,}15{\omega_0}^2$$

$$K = 40{,}535$$

$$s^0 : \ 2I = {\omega_0}^3$$

$$I = 100{,}46$$

Ďalšou možnosťou pre získanie hodnou K a I systému je použiť Naslinovú metódu

Opäť začíname získaním charakteristickej rovnice, ktorá bude teda vyzerať $1+F_R(s)F_s(s)$. Výsledná charakteristická rovnica teda bude vyzerať:

$$0{,}5s^3 + 6s^2 + (10{,}01 + K)s + \frac{K}{T_i} = 0$$

Zavedieme substitúciu:
$$\frac{K}{T_i} = I$$

Podľa vzťahu ${a_j}^2 \geq \alpha a_{j-1}a_{j+1}$ pričom α získame z tabuľky:

α	1.75	1.8	1.9	2.0	2.2	2.4
$\sigma(\%)$	16	12	8	5	3	1

Pre náš výpočet si určíme maximálne preregulovanie 5%.
A teda získame dve nerovnice, ktorý úpravou získame hodnoty K a I

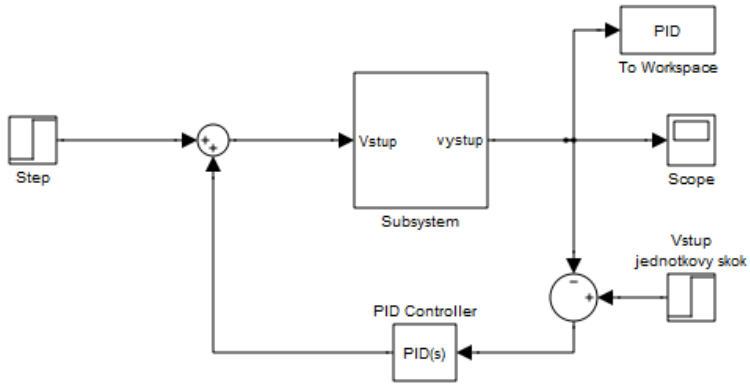
$$j = 1: \quad (10{,}01 + K)^2 \geq 2 * I * 6$$

$$j = 2: \quad 6^2 \geq 2 * 0{,}5 * (10{,}01 + K)$$

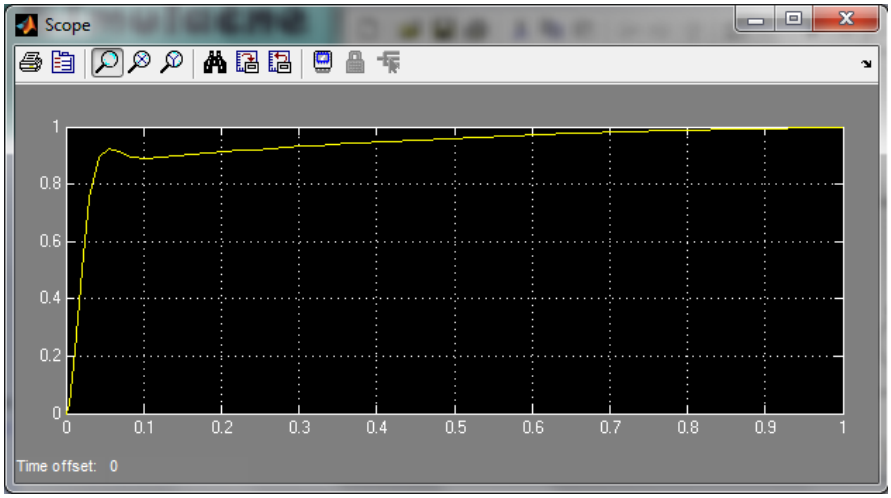
$$K \leq 25{,}99$$
$$I \leq 108$$

RIEŠENIE:

1.

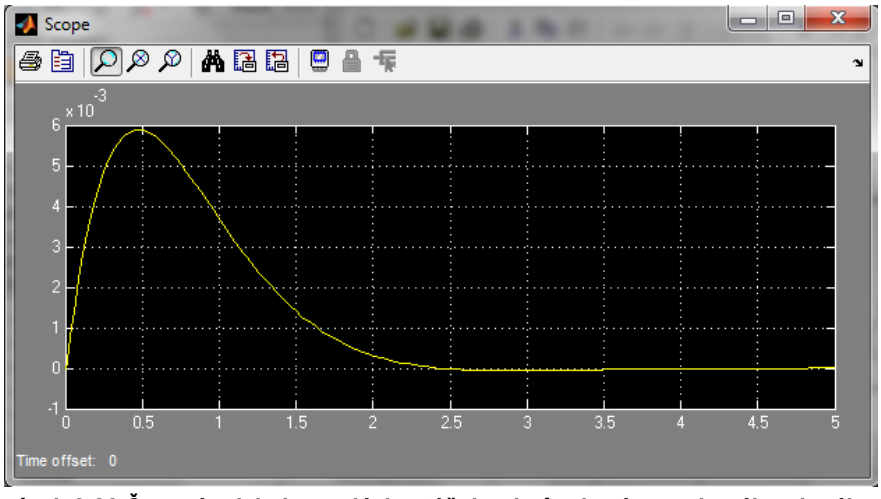


2.



Obrázok 8-25 Časový priebeh regulácie otáčok pri zmene vstupného signálu riadiacej veličiny w(t)

3.



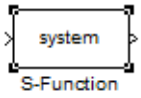
Obrázok 8-26 Časový priebeh regulácie otáčok pri pôsobení poruchového signálu z(t)

8.6 Postup pri tvorbe s-funkcií

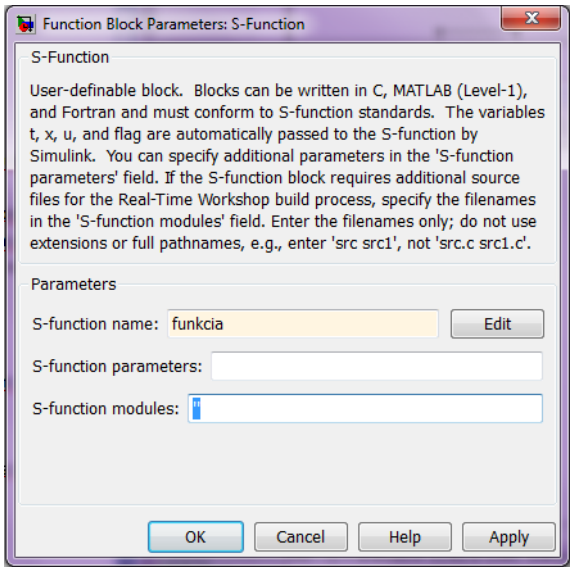
- ⇨ S-funkcia je dynamicky blok v Simulinku, ktorého "opis" je definovaný vo funkcií-súbore , ktorý je vytvorený v simulačnom jazyku MATLAB, v jazykoch C, C++, Ada alebo Fortran.
- ⇨ S-funkcie vytvorené v jazykoch C, C++, Ada a Fortran sú kompilované na MEX-súbory
- ⇨ S-funkcia umožňuje užívateľovi vytvárať vlastné bloky do modelu v Simulink-u. V m-súbore s-funkcie môžu byť definované vlastné diferenciálne rovnice, rovnice diskrétneho systému a/alebo ľubovoľný typ algoritmu.

Použitie S-funkcie v modeloch

Blok S-funkcion sa nachádza v knižnici Simulink → User- Defined Function →S-Function



Po presunutí do bloku modelu sa po dvojkliku na systém otvorí okno Block Parameters: S-Function.



- ⇨ Parametre sú oddeľované čiarkou a môžu to byť konštanty (vektory, matice), názvy premenných definovaných v pracovnom priestore programového prostredia MATLAB alebo výrazy v programovom prostredí MATLAB.
- ⇨ Parametre t, x, u (čas, stavy a vstupy) sú Simulink-om automaticky prenášané do S-funkcie.

Význam S-funkcií

- ⇨ S-funkcia reprezentuje dynamický prvok (u → [x] → y) , kde výstupy sú funkciou periódy vzorkovania, vstupov a stavov.
- ⇨ Matematické vzťahy medzi vstupmi, výstupmi a stavmi môžu byť vyjadrené nasledujúcimi rovnicami:

$y = f_y(t, x, u)$ (výstup)

$x_s = f_s(t, x, u)$ (derivácia – spojitý systém)

$x_d(t+1) = f_d(t, x, u)$ (diferencia – diskrétny systém)

$x = \{x_s, x_d\}$

⇨ Simulácia S-funkcie prebieha v niekoľkých etapách:

1. *flag = 0* - inicializácia (funkcia *mdlInitializeSizes*)
2. *flag = 4* - výpočet periódy vzorkovania (funkcia *mdlGetTimeOfNextVarHit*)
3. *flag = 3* - výpočet výstupu y(funkcia *mdlOutputs*)
4. *flag = 2* - výpočet diskrétnych stavov x_d (funkcia *mdlUpdate*)
5. *flag = 1* - výpočet spojitých stavov x_s (funkcia *mdlDerivatives*)
6. *flag = 9* - koniec (funkcia *mdlTerminate*)

S-funkcia v tvare m-súboru je definovaná ako obyčajná funkcia v simulačnom jazyku MATLAB:

$$\boldsymbol{function\ [sys,x0,str,ts] = meno(t,x,u,flag,p1,p2,\ldots)}$$

kde ***meno*** je názov S-funkcie, ***t*** je aktuálny čas, ***x*** je vektor stavov daného bloku (S-funkcie), ***u*** sú vstupy do bloku, ***flag*** udáva vykonávanú úlohu a ***p1,p2, a ďalšie*** sú parametre bloku.

⇨ Počas simulácie modelu, Simulink opakovane vyvoláva S-funkciu ***meno*** a na základe ***flag***-ov sa vykonávajú jednotlivé etapy S-funkcie.

PRÍKLAD 1
Vytvorte s-funkciu pre model Van der Pólovho oscilátora, ktorý je popísaný diferenciálnou rovnicou:

$$y'' - A(1-y^2)y' + y = u$$

Po prepise do substitučného kanonického tvaru, za podmienky, že stavová premenná $x_1 = y$, získame dve diferenciálne rovnice 1.rádu:

$$x_1' = x_2$$
$$x_2' = u - x_1 + x_2 * A * (1 - x_1^2)$$

Postup:

1. Otvoríme si Simulink a z knižnice User-Defined Functions vyberieme blok S-Function a vložíme ho do vytváraného systému alebo do nového súboru.
2. Po dvojkliku na tento blok sa otvorí okno v ktorom vieme zmeniť názov tohto bloku a pridať parametre.
3. Otvoríme si nový m-file, ktorý bude predstavovať funkciu a jej názov bude rovnaký ako názov nami vytvorenej s-funkcie.
4. Ak máme vytvorenú funkciu začneme s definovaním príznakov, ktoré budeme využívať. V našom prípade bude táto časť vyzerať nasledovne:

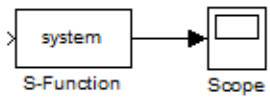
```
switch flag,
  case 0 % inicializácia
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1 % výpočet spojitých stavov
    sys=mdlDerivatives(t,x,u);
  case 3 % príznak výstupu
    sys=mdlOutputs(t,x,u);
  case {2, 4, 9}
    sys = [];
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag',     num2str(flag));     %
chybový oznam, v pprípade ak sa vyskytné iný príznak ako sú definované.
end
```

pokračujeme definovaním konkrétnych príznakov:

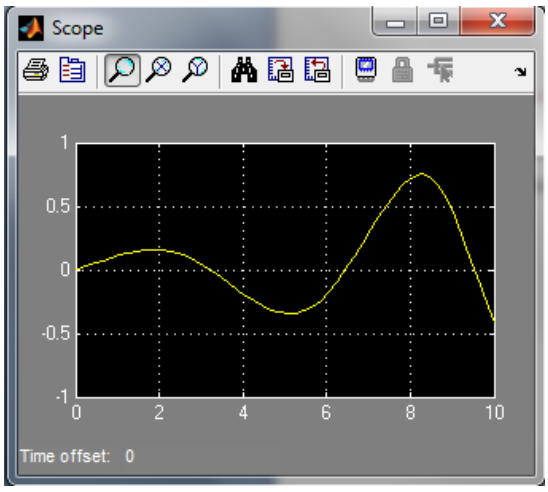
```
%=============================================================================
% mdlInitializeSizes
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
```

```
sizes = simsizes;


sizes.NumContStates  = 2; %počiatočné stavy
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1; %výstup
sizes.NumInputs      = 1; %vstup
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [0; 0]; % počiatočné podmienky
str = [];
ts  = [0 0]; % spojitý systém

%=============================================================================
% mdlDerivatives
%=============================================================================
function sys=mdlDerivatives(t,x,u)
 A=0.5; u=0.1   % definovanie vnútorných premenných
sys = [x(2); u-x(1)+x(2)*A*(1-x(1)*x(1))];

%=============================================================================
% mdlOutputs
%=============================================================================
function sys=mdlOutputs(t,x,u)
sys = x(2); % Výstupom je druhý stav
```

5. Do schémy pridáme ostatné bloky ktoré sú potrebné pre spustenie simulácie a systém je vytvorený.



Získaná simulácia vyzerá nasledovne:



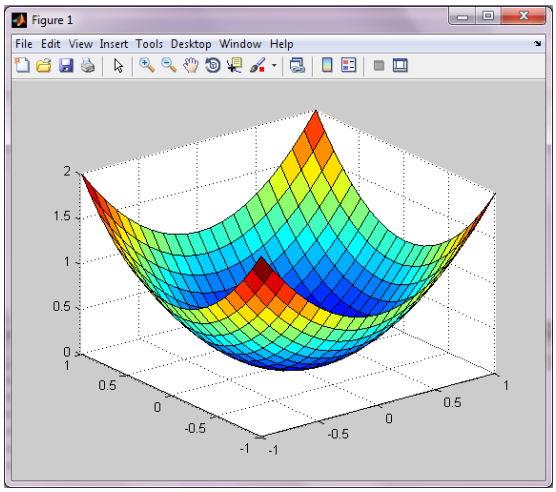
Obrázok 8-27 Časový priebeh riešenia NDR (Van-der-Polov oscilátor) získaný z S-funkcie

9 Využitie grafických možností jazyka MATLAB

9.1 Úvod do práce s grafickým prostredím

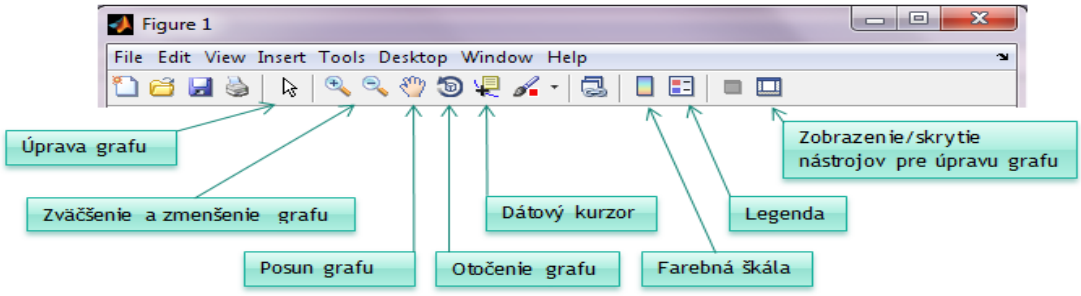
MATLAB ponúka rýchlu a kvalitnú reprezentáciu funkcií vo forme grafov. Disponuje pokročilou grafikou v oblasti 2D a 3D. Vizualizácia v MATLABe umožňuje prehľadné zobrazovanie výsledkov. Môžeme povedať, že grafika je jednou z najsilnejších stránok MATLABu.

Grafický výstup MATLABu sa realizuje v novom okne – **FigureWindow**. Toto okno sa skladá z hlavnej ponuky, lišty nástrojov a plochy v ktorej je graf vykreslený.



Obrázok 9-1 Figure Window v prostredí MATLAB

Na nasledujúcom obrázku je zobrazená a popísaná lišta nástrojov.



Obrázok 9-2 Lišta nástrojov v grafickom okne

Grafy vytvorené v prostredí MATLAB je možné uložiť do súboru ako:

A. Obrázok

1. Menu File – **Export Setup**
2. Vyplnenie dialógového okna **Export**
3. Aktivácia tlačidlom **Save**

B. *.fig

1. Menu – **Saveas**
2. Zadanie cieľového priečinka a mena súboru
3. Aktivácia tlačidlom Save

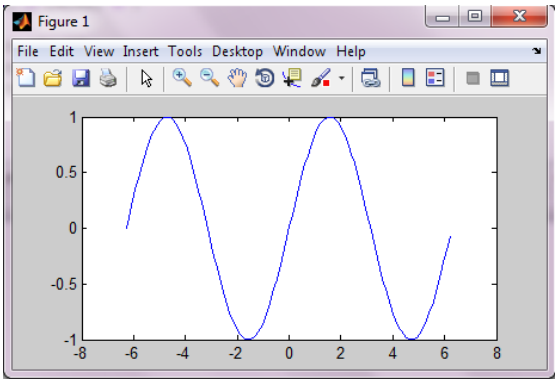
9.2 Funkcie pre 2D grafiku

Na vykreslenie 2D grafov bolo vytvorených niekoľko funkcií. Najpoužívanejšia je ***plot***. Ďalšie funkcie slúžiace na vykreslenie 2D grafov sú ***stairs***, ***bar***, ***stem***. Pomocou 2D grafov zvyčajne vykresľujeme závislosť jednej veličiny od druhej. Takúto závislosť je možné zapísať nasledovne *y=f(x)*.
Grafické možnosti Matlabu budeme znázorňovať na funkcii danej predpisom:

$$y = \sin(x)$$

Do premennej *x* vložíme vektor čísel od -2π do 2π. Premenná *y* je závislá na premennej *x*, vložíme do nej vektor *sin(x)*. Funkcia *plot* nám vytvorí graf funkcie f(x): y1 = sin(x)

```
x=-2*pi:0.1:2*pi;
y1=sin(x);
plot(x,y1)
```



Obrázok 9-3 Grafické znázornenie y=sin(x)

Matlab ponúka možnosť popísania a modifikovania daného grafu. Umožňuje pridanie názvu grafu, popísanie osi, vloženie legendy, zmeniť štýl a farbu čiary. V nasledujúcich tabuľkách sú vypísané znaky pre zmenu farby a štýlu grafu.

Symbol	Farba
b	modrá (blue)
g	zelená (green)
r	červená (red)
c	tyrkysová (cyan)
m	purpurová (magenta)
y	žltá (yellow)
k	čierna (black)
w	biela (white)

Symbol	Značka
.	bod
o	kruh
x	krížik
+	plus
*	hviezda
s	štvorec
d	diamant
v	trojuholník dole
^	trojuholník hore
>	trojuholník vpravo
<	trojuholník vľavo
P	pentagram
h	hexagram

Symbol	Štýl čiary
-	plná čiara
:	bodkovaná čiara
-.	bodko – čiarkovaná čiara
--	čiarkovaná čiara

Popis grafu realizujeme nasledovne:

```
title('Názov grafu')
xlabel('názov x-ovej osi')
ylabel('názov y-ovej osi')
legend('názov 1. funkcie','názov 2.funkcie','...')
```

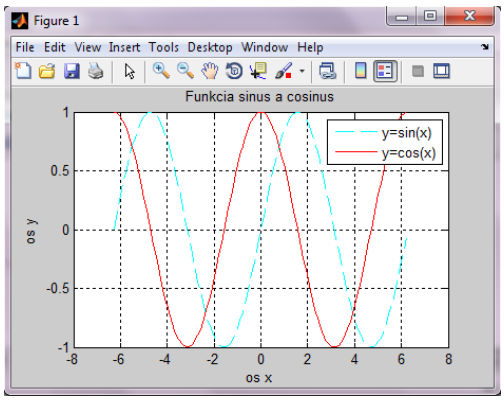
Graf môžeme obohatiť aj o mriežku, ktorá nám pomôže lepšie sa v ňom orientovať. Zapnutie mriežky sa vykoná príkazom ***grid on*** a vypnutie opačným príkazom a to ***gridoff.***

Do jedného grafu môžeme vložiť viac funkcií. Môžeme to vykonať dvomi spôsobmi. Keď ešte nemáme vytvorený graf, do funkcie plot za parametre vytvárajúce jeden priebeh zadáme ďalšie parametre takým istým spôsobom. V prípade, že už máme vytvorený graf použijeme funkciu ***hold on,*** ktorá nám už vytvorený graf zafixuje.

Riešený príklad 1:

Vytvorte graf s názvom „Funkcia sínus a cosínus“ vložte doňho priebeh funkcie sínus a cosínus. Popíšte x-ovú aj y-ovú os, vložte legendu a zapnite mriežku. Ľubovoľne meňte štýl a farbu vykreslenia.

```
x=-2*pi:0.1:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'c--',x,y2,'r-');
grid on
title('Funkcia sinus a cosinus')
xlabel('os x')
ylabel('os y')
legend('y=sin(x)','y=cos(x)')
```

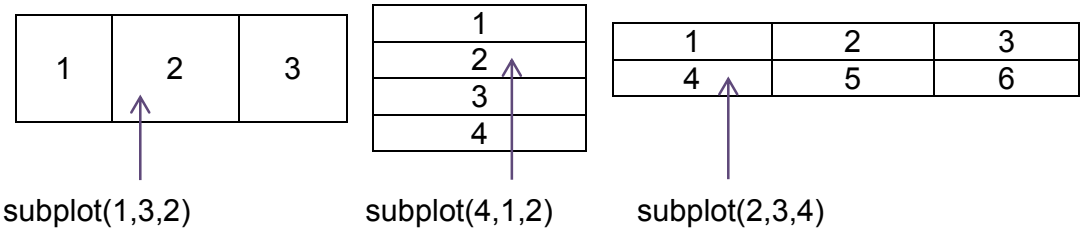


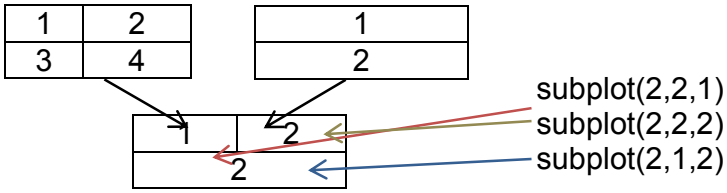
Obrázok 9-4 Grafický priebeh goniometrických funkcií

Zvýraznenú časť je možné zameniť za nasledujúcu časť kódu a to bez zmeny výsledku.

```
plot(x,y1,'c--')
hold on
plot(x,y2,'r-')
```

Okno **FigureWindow** je možné rozdeliť do niekoľkých častí pomocou príkazu ***subplot***(m,n,p). Tento príkaz rozdelí okno na mriežku okien s rozmermi *m*x*n* a parameter *p* určuje poradové číslo aktívneho poľa.

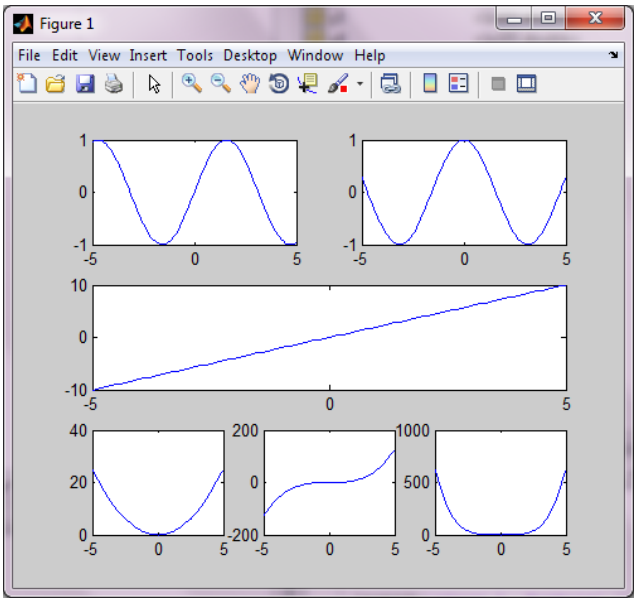




Riešený príklad 2

Pomocou funkcie **subplot** vytvorte grafické okno rozdelené do troch riadkov, v 1.riadku budú 2 stĺpce a v jednotlivých oknách funkcie sínus a cosínus, v 2. riadku 1 stĺpec a v okne funkcia 3*x, v 3. riadku 3 stĺpce a v jednotlivých oknách funkciex^2, x^3, x^4.

```
x=-5:0.1:5;
ysin=sin(x);
ycos=cos(x);
yp=3*x;
y2=x.^2;
y3=x.^3;
y4=x.^4;
subplot(3,2,1)
plot(x,ysin)
subplot(3,2,2)
plot(x,ycos)
subplot(3,1,2)
plot(x,yp)
subplot(3,3,7)
plot(x,y2)
subplot(3,3,8)
plot(x,y3)
subplot(3,3,9)
plot(x,y4)
```

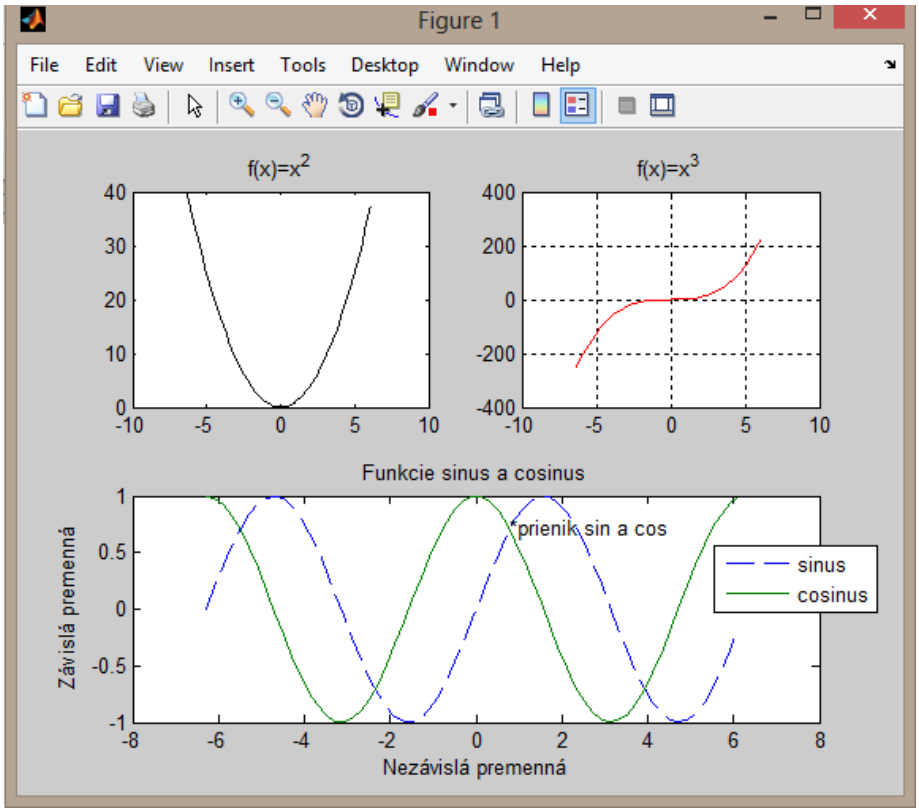


Obrázok 9-5 Delenie grafického okna s využitím funkcie subplot

Riešený príklad 3

Vytvorte komplexný príklad v ktorom použijete funkcie: **plot**, **subplot**, funkcie na popis grafu a vloženie mriežky.

```
x=-2*pi:0.2:2*pi;
y1=sin(x);
y2=cos(x);
y3=x.^2;
y4=x.^3;
subplot(2,2,1);
plot(x,y3,'k')
title('f(x)=x^2')
subplot(2,2,2);
plot(x,y4,'r')
grid on
title('f(x)=x^3')
subplot(2,1,2);
plot(x,y1,'--',x,y2)
title('Funkcie sinus a cosinus')
xlabel('Nezávislá premenná')
ylabel('Závislá premenná')
legend('sinus','cosinus')
text(pi/4,sin(pi/4),'*prienik sin a cos')
```



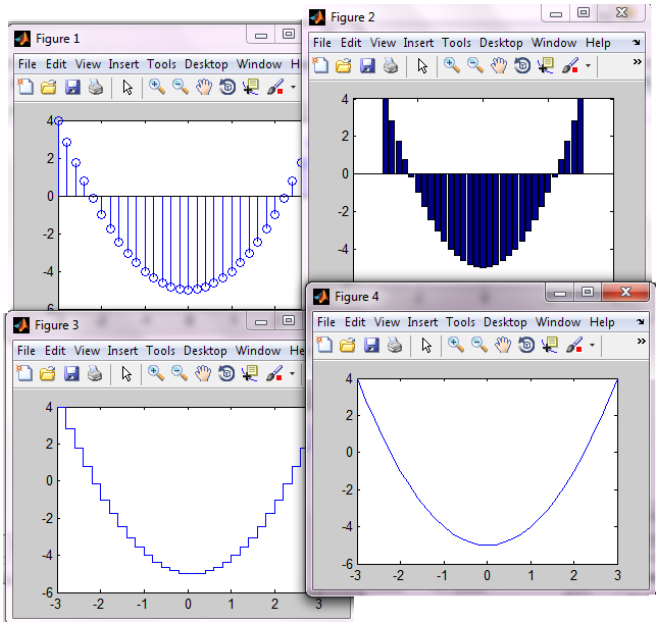
Obrázok 9-6 Grafický výstup matematických funkcií vo viacerých podoknách

Pre prípad kedy potrebujeme viac grafov, no nie v jednom okne bola vytvorená funkcia ***figure***.

Riešený príklad 4

Porovnajte grafy vytvorené pomocou funkcií plot, bar, stem, stair pre funkciu $y=x^2-5$. Každý graf vytvorte v novom okne.

```
x=-3:0.2:3;
y=x.^2-5;
stem(x,y)
figure
bar(x,y)
figure
stairs(x,y)
figure
plot(x,y)
```



Obrázok 9-7 Grafické znázornenie funkcie $y= x^2-5$

MATLAB ďalej ponúka možnosť vykresľovania takzvaných diskrétnych grafov, ktoré vyjadrujú hodnotu v určitom diskrétnom okamihu. Ide o stĺpcové grafy a koláčové diagramy. V nasledujúcej tabuľke sú uvedené počty prijatých študentov v jednotlivých odboroch. S údajmi uvedenými v tabuľke budeme pracovať .

Rok	2008	2009	2010	2011	2012
Hospodárska informatika	150	190	92	85	70
Kybernetika	32	30	35	30	25
Aplikovaná informatika	25	27	32	30	29
Informatika	350	320	390	370	385

Tieto údaje je možné vložiť do matice, ktorej stĺpce budú predstavovať jednotlivé odbory a riadky rok.

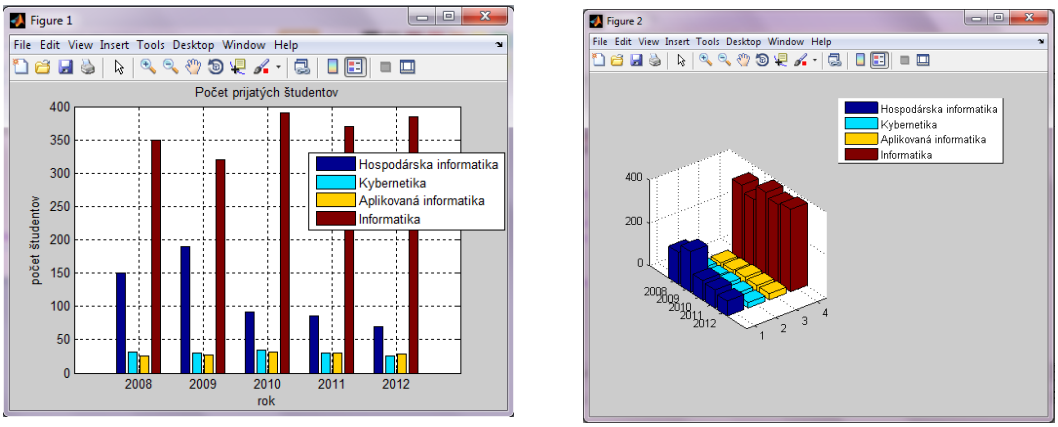
```
S=[150 32 25 350;190 30 27 320;92 35 32 390;85 30 30 370;70 25 29 385]
```

Základným príkazom pre vytvorenie stĺpcového grafu je ***bar***. Otočia grafu do horizontálnej roviny a zobrazovanie dát v 3D je možné pomocou príkazov *barh, bar3, bar3h*.

Riešený príklad 5

Vytvorte stĺpcový graf. Údaje čerpajte z predchádzajúcej tabuľky. Popíšte osi grafu, pridajte názov a legendu.

```
rok=[2008 2009 2010 2011 2012];
bar(rok,S)
grid on
title('Prijatie študentov')
xlabel('rok')
ylabel('počet študentov')
legend('Hospodárska informatika','Kybernetika','Aplikovaná informatika','Informatika')
figure
bar3(rok,S)
legend('Hospodárska informatika','Kybernetika','Aplikovaná informatika','Informatika')
```



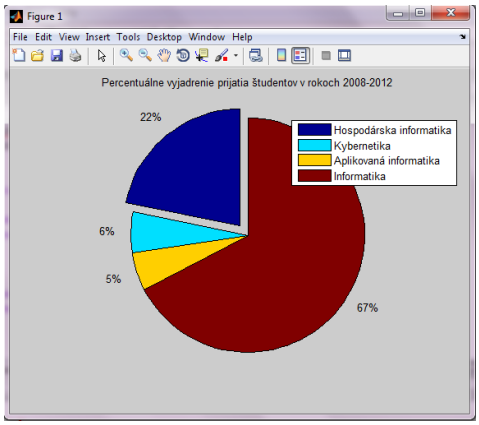
Obrázok 9-8 Grafické znázornenie počtu prihlásených študentov na odbory FEI stĺpcovým grafom

Koláčový, alebo kruhový, diagram sa vytvára pomocou príkazu ***pie,*** tak ako príkaz *bar,* tak ajpríkaz *pie* umožňuje zobrazenie v 3D a to príkazom *pie3.* Tento príkaz ponúka možnosť vysunutia určitej časti. Vysunutie vieme docieliť druhým parametrom príkazu. Druhý parameter je reprezentovaný vektorom rovnako dlhým ako je počet zobrazovaných hodnôt. V tomto vektore zadávame nulové a nenulové hodnoty. Nenulové hodnoty vysunú príslušnú časť z grafu.

Riešený príklad 6

Vytvorte kruhový graf, v ktorom budú udané počty prijatých študentov na jednotlivé odbory. Odbor hospodárskej informatiky vysuňte z grafu.

```
posun=[1,0,0,0];
pie3(sum(S),posun)
```



Obrázok 9-9 Grafická reprezentácia kruhového grafu

9.3 Funkcie pre 3D grafiku

Trojrozmerná grafika slúži na vykreslenie funkcie dvoch premenných. Vo funkcii sú dve premenné nezávislé a jedna závislá. Funkčné hodnoty závislej premennej sú odvodené od hodnôt nezávislých premenných. Takúto závislosť môžeme zapísať vzťahom :

$$z=f(x,y)$$

kde premenná *z* je závislá a premenné *x, y* sú nezávislé.
Výsledkom 3D grafov je plocha. Potrebujeme definovať plochu, ktorá bude definičným oborom našej funkcie. Výpočet závislej premennej sa vykonáva v každom priesečníku nezávislých premenných. Použitím funkcie ***meshgrid*** sa vytvoria dve matice, ktoré sú na seba navzájom kolmé a budú predstavovať definičný obor funkcie. Vstupom tejto funkcie sú vektory *x* a *y*.
Predpokladajme, že chceme vytvoriť definičný obor na intervale <-1;1> pre súradnicu *x* aj *y*. Nech je tento obor tvorený mriežkou s krokom 0,5.

```
>> [X,Y]=meshgrid(-1:0.5:1,-1:0.5:1)

X =

   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000
   -1.0000   -0.5000         0    0.5000    1.0000


Y =

   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -0.5000   -0.5000   -0.5000   -0.5000   -0.5000
         0         0         0         0         0
    0.5000    0.5000    0.5000    0.5000    0.5000
    1.0000    1.0000    1.0000    1.0000    1.0000
```

Majme funkciu $z=x^3+y^2$. Maticu Z dostaneme výpočtom :

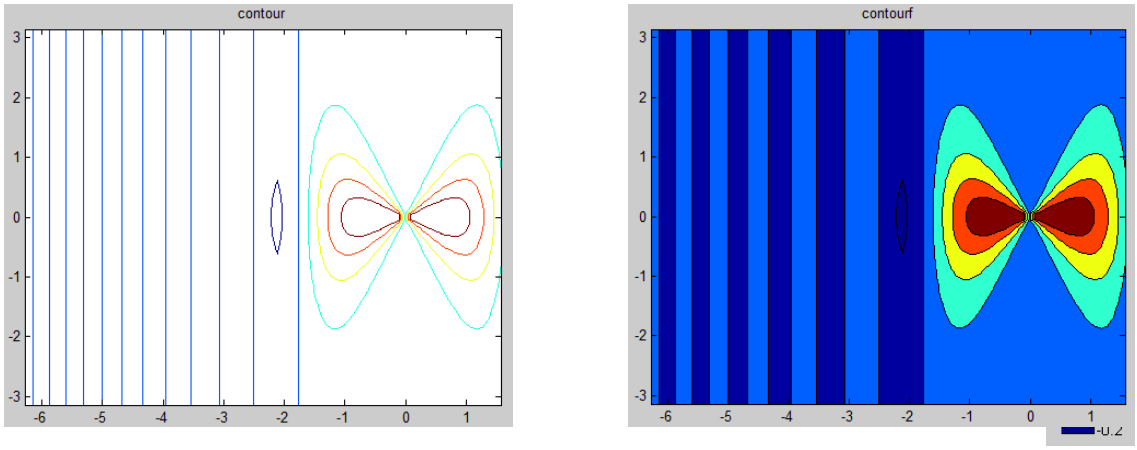
```
>> Z=X.^3+Y.^2

Z =

         0    0.8750    1.0000    1.1250    2.0000
   -0.7500    0.1250    0.2500    0.3750    1.2500
   -1.0000   -0.1250         0    0.1250    1.0000
   -0.7500    0.1250    0.2500    0.3750    1.2500
         0    0.8750    1.0000    1.1250    2.0000
```

Základné príkazy pre tvorbu 3D grafov sú ***plot3, mesh, surf, contour.*** Príkaz *contour* zobrazuje izočiary, príkaz je možné modifikovať na *contourf* a graf zobrazí izoplochy. Príkazy *mesh* a *surf* môžeme modifikovať na *meshc* a *surfc.* Takto zmeneným príkazom sa obohatí graf o izočiary. Príkaz *mesh* je možné modifikovať na *meshz*, ktorý do grafu pridá nulovú hladinu. Všetkým týmto príkazom je potrebné zadať tri parametre a to prvé dva sú matice definičného oboru a tretím parametrom sú funkčné hodnoty funkcie.

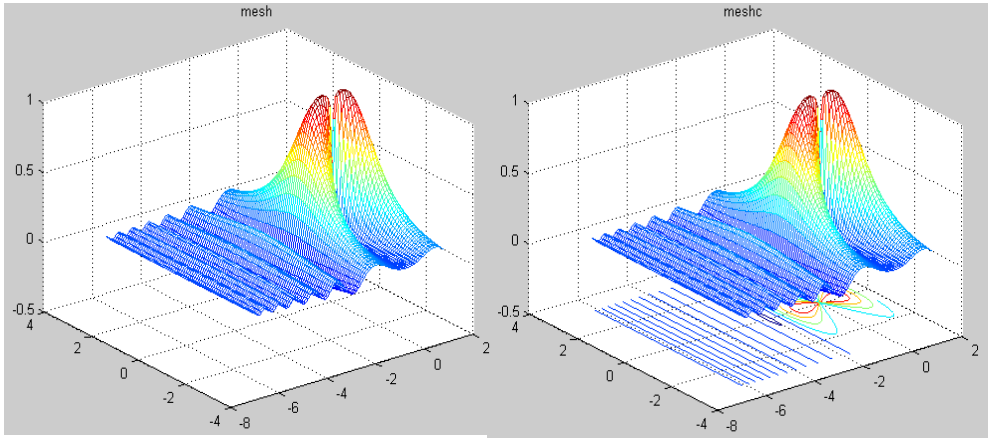
Riešený príklad 7

Demonštrujte priebeh funkcie $z = \sin(x^2)/(x^2+y^2)$ na jednotlivých typoch grafov.

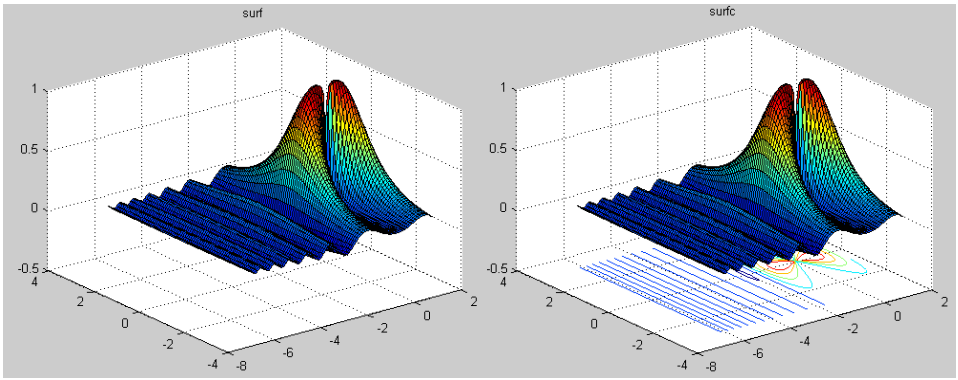
```
x=linspace(-2*pi,pi/2,80);
y=linspace(-pi,pi,80);
[X,Y]=meshgrid(x,y);
Z=sin(X.^2)./(X.^2+Y.^2);
contour(X,Y,Z)
title('contour')
figure
contourf(X,Y,Z)
title('contourf')
figure
surf(X,Y,Z)
title('surf')
figure
surfc(X,Y,Z)
title('surfc')
figure
mesh(X,Y,Z)
title('mesh')
figure
meshc(X,Y,Z)
title('meshc')
```

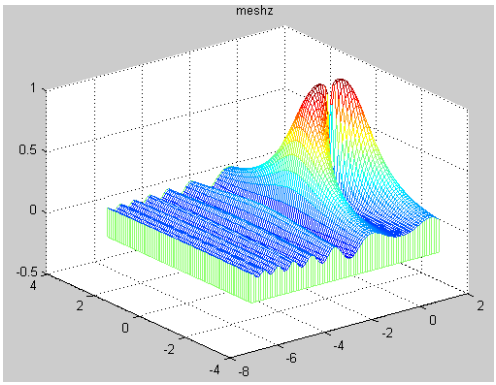


Obrázok 9-10 Grafické znázornenie funkcie z pomocou príkazu contour/contourf



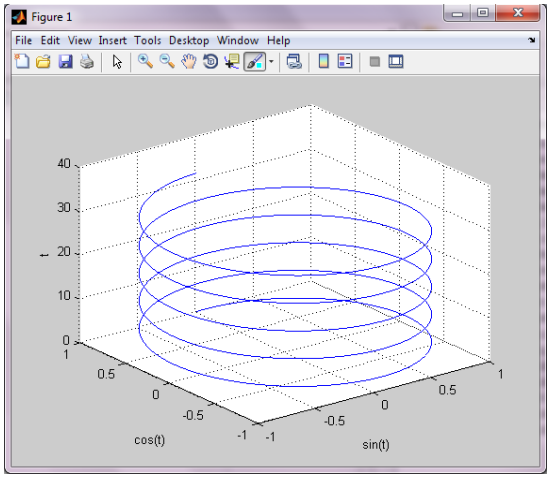
Obrázok 9-11 Grafické znázornenie funkcie z pomocou príkazu surf/surfc





Vykreslenie spojnicových grafov v trojrozmernom priestore umožňuje funkcia ***plot3***. Syntax príkazu je podobná ako pri funkcii *plot*s tým rozdielom, že sa neudáva usporiadaná dvojica bodov, ale trojica . táto funkcia spája definované body v priestore. Ako príklad použijeme kód uvedený v heple príkazu plot3. Kód vykreslí závitnicu.

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t);
grid on;
xlabel('sin(t)');
ylabel('cos(t)');
zlabel('t');
```

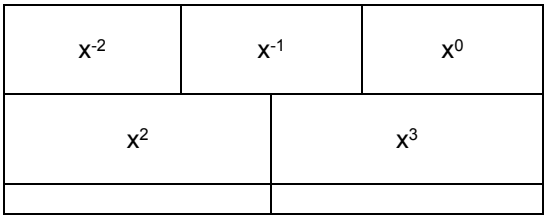


Obrázok 9-13 Grafické znázornenie závitnice funkciou plot3

9.4 Príklady na riešenie

1. Vytvorte graf s názvom „Exponenciálne funkcie“ vložte doňho priebeh funkcie x^{-2}, x^{-1},x^0,x^2 a x^3, . Hodnoty na x-ovej osi nech sú z intervalu <-10; 10> s krokom 0,2. Popíšte x-ovú aj y-ovú os, vložte legendu a zapnite mriežku. Ľubovoľne meňte štýl a farbu vykreslenia.

2. Z vytvorených funkcií v predošlom príklade vytvorte pomocou funkcie subplot grafické okno nasledujúceho tvaru:



9.5 Vytváranie trojrozmerných grafických zobrazení v prostredí MATLAB a interaktívne úpravy grafov

⇨ Pre zobrazenie obrazca potrebujeme mať dve matice, napr. X,Y , ktorých prvky $X[i,j]$ a $Y[i,j]$ budú obsahovať súradnice bodov v rovine.

⇨ Na generovanie týchto dvojrozmerných polí slúži funkcia meshgrid

$[X,Y] = meshgrid(v1,v2)$, pričom vektore $v1$ určuje interval na osi x s daným krokom a vektor $v2$ určuje interval na osi y s daným krokom.
$[X,Y] = meshgrid(v)$, pričom v je vektor, ktorý určuje interval na osi x aj osi y.

```
>> v1=[-5:0.1:1];  % inicializácia vektora v1, s potlačením výpisu
 >> v2=[4:0.1:12]; % inicializácia vektora v2, s potlačením výpisu
>> [X,Y]=meshgrid(v1, v2); % výpočet matíc X a Y, s potlačením výpisu

 >> v=[-5:0.1:5]; % inicializácia vektora v, s potlačením výpisu
>> [X,Y]=meshgrid(v); % výpočet matíc X a Y, s potlačením výpisu
```

⇨ ak mame vygenerované nezávislé premenné X a Y potrebujeme k vykresleniu ešte dvojrozmerné pole funkčných hodnôt napr. Z, do ktorej vložíme funkčnú závislosť

PRÍKLAD 1
Vytvorte maticu Z z funkcie $f(x,y) = \sin(5x) - \cos(y^2)$, pre interval na osi $x \in$ <-2,3>
a osi $y \in$ <0,6>

```
>> x=[-2,3]; % inicializácia vektora x, s potlačením výpisu
>> y=[0:6]; % inicializácia vektora y, s potlačením výpis
>>[X,Y]=meshgrid(x,y) %  výpočet matíc X a Y z vektorov x a y
      >> Z=sin(5*X)-cos(Y.^2); % výpočet matice Z pre matice
            nezávislých premenných X a Y
```

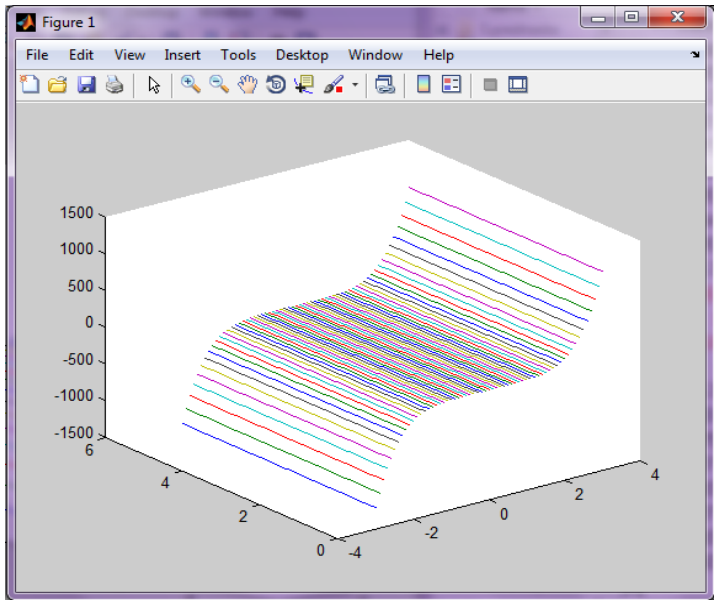
⇨ Pre vykreslenie 3D obrazca môžeme použiť nasledujúce funkcie:

- $plot3(X,Y,Z)$ – zobrazuje čiary a body v 3D priestore

PRÍKLAD 2
Vykreslite čiarový graf pre funkciu $f(x,y) = 5x^5 - \cos(y^2)$, pre intervalom pre os x $\in$ <-3,3>

a y $\in$ <0,5>

```
>> x=[-3:0.1:3];   % inicializácia vektora x, s potlačením výpisu
>> y=[0:0.1:5]; % inicializácia vektora y, s potlačením výpisu
>> [X,Y]=meshgrid(x,y); % výpočet matíc nezávislých premenných X a Y
>> Z=5*X.^5-cos(Y.^2); % výpočet matice Z pre matice nezávislých premenných
X a Y
>> plot3(X,Y,Z) % funkcia pre zobrazenie čiarových 3D grafov
```



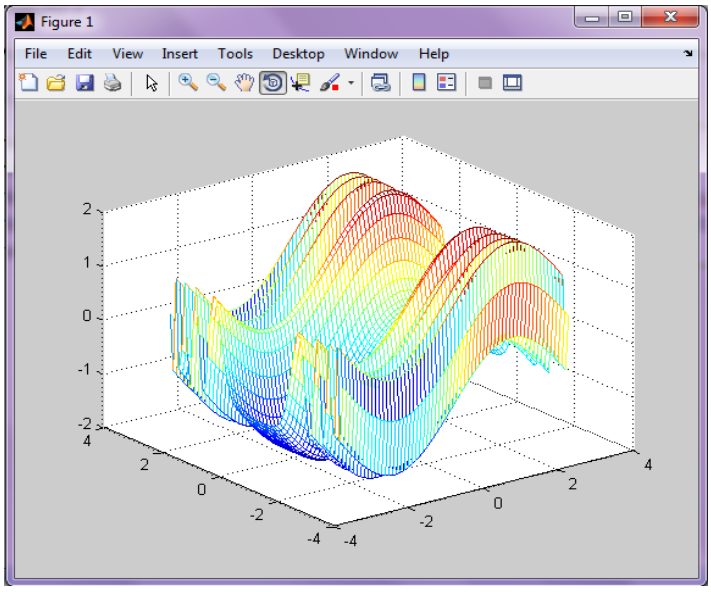
Obrázok 9-14 Grafické zobrazenie funkcie $f(x,y) = 5x^5 - \cos(y^2)$

- $mesh(X,Y,Z)$ - vytvára sieťové (inak nazývané aj drôtové) zobrazenie

PRÍKLAD 3

Vykreslite sieťový graf pre funkciu $f(x,y) = \sin(x) - \cos(y^3)$, pre intervalom pre os x ϵ <-3,3> a y ϵ <-3,3>

```
>> x=[-3:0.1:3];  % inicializácia vektora x, s potlačním výpisom
>> y=[-3:0.1:3];   % inicializácia vektora y, s potlačeným výpisom
>> [X,Y]=meshgrid(x,y); % výpočet matíc nezávislých premenných X a Y
>> Z=sin(X)-cos(Y.^3); %  výpočet matice Z pre matice nezávislých
premenných X a Y
>> mesh(X,Y,Z) % vykreslenie sieťového 3D grafu
```



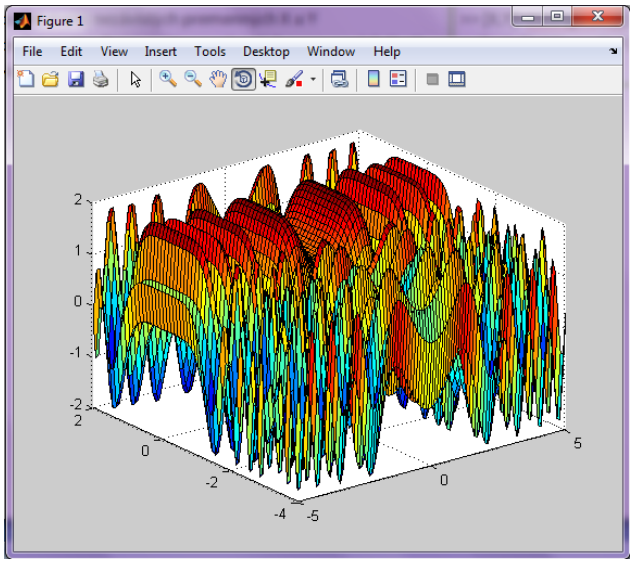
Obrázok 9-15 Grafické zobrazenie funkcie $f(x,y) = \sin(x) - \cos(y^3)$

- $surf(X,Y,Z)$ – vytvára plošný graf

PRÍKLAD 4

Vykreslite sieťový graf pre funkciu $f(x,y) = \sin(x^2) + \cos(y^3)$, pre intervalom pre os x ϵ <-5,5> a y ϵ <-4,2>

inicializácia vektora x, s potlačeným výpisom	>> x=[-5:0.1:5];
inicializácia vektora y, s potlačeným výpisom	>> y=[-4:0.1:2];
výpočet matíc nezávislých premenných X a Y	>> [X,Y]=meshgrid(x,y);
výpočet matice Z pre matice nezávislých premenných X a Y	>>Z=sin(X.^2)+cos(Y.^3);
vykreslenie sieťového 3D grafu	>> surf(X,Y,Z)



Obrázok 9-16 Grafické zobrazenie funkcie f(x,y)=sin(x^2)+cos(y^3)

10 Aplikačné využitie funkcií Symbolic Math Toolbox-u pri riešení jednoduchých úloh

- **Využitie Symbolic Math Toolboxu**

⇨ výpočet limít, derivácií a integrálov
⇨ riešenie algebraických a diferenciálnych rovníc
⇨ prepočet numerického tvaru polynómu na symbolický a naopak
⇨ vykresľovanie grafov funkcií

Rozoberané funkcie, ktoré tvoria základne jadro programového prostredia MATLAB
⇨ popis symbolického toolbox-u a jeho funkcií nájdeme po zadaní príkazu ***help symbolic***

- **Tvorba symbolických premenných**

⇨ $sym('premenná')$ - vytvorenie jednej symbolickej premennej
⇨ $syms\ premenná1\ premenná2\ \ldots\ premenná\,n$

Funkcie v Symbolic Math Toolboxe nie je potrebné zadávať v podobe inline funkcií alebo ako samostatných m-súborov. Ak máme zadefinovanú symbolickú premennú v prostredí MATLAB, môžeme s ňou pracovať ako s numerickou premennou.

- Výsledok symbolického výpočtu môžeme pretransformovať do premenných programového prostredia MATLAB $double(symbolická\ premenná)$
- Úprava výsledku príkazom $pretty(premenná)$

PRÍKLAD 1
Vytvorte symbolický výraz $\varphi = 1 + \frac{\sqrt{5}}{2}$ a pomocou tejto premennej vypočítajte hodnotu $f = \varphi^2 - \varphi - 1$.

Riešenie príkladu 1 v programovom prostredí MATLAB:

```
>> phi=sym('(1+sqrt(5))/2') % definícia symbolickej
premennej φ
phi = 5^(1/2)/2 + 1/2
>> f=phi^2-phi-1 % výpočet hodnoty symbolickej funkcie f
f = (5^(1/2)/2 + 1/2)^2 - 5^(1/2)/2 - 3/2
```

PRÍKLAD 2
Zadefinujte maticu A veľkosti 3 x 4, ktorá bude obsahovať symbolické premenné a, b, c, d. Z tejto matice vyberte 2 stĺpec a sčítajte jeho hodnoty.

Riešenie príkladu 2 v programovom prostredí MATLAB:

```
>> syms  a b c d % definícia symbolických premenných a, b, c, d
>> A=[a b c d; d c b a; c b c b] % vytvorenie matice A z symbolických
premenných a, b, c, d
 A =
[ a, b, c, d]
[ d, c, b, a]
[ c, b, c, b]
 >> sum(A(:,2)) % funkcia pre spočítanie prvkov 2 stĺpca matice A
ans =
2*b + c
```

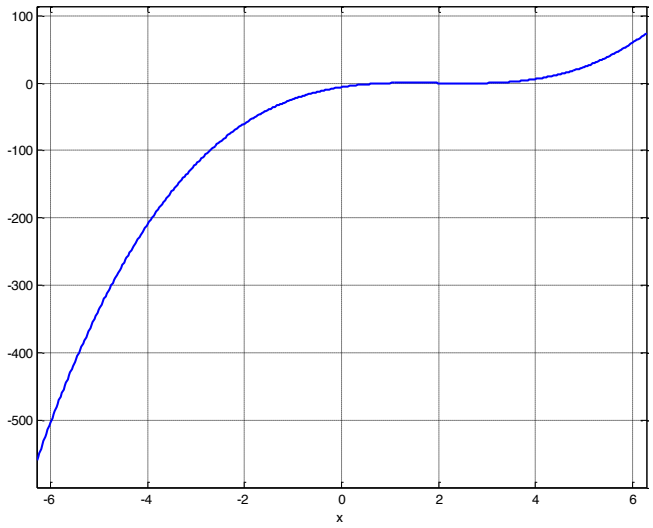
- **Vykresľovanie funkcií pomocou Symbolic Math Toolbox**

$ezplot(f)$ – vykresľovanie funkcie f v 2D priestore

PRÍKLAD 3
Vykreslite funkcie $x^3 - 6x^2 + 11x - 6$ v 2D priestore.

Riešenie v programovom prostredí MATLAB:

```
>> syms x
>>ezplot(x^3 - 6*x^2 + 11*x - 6)
```



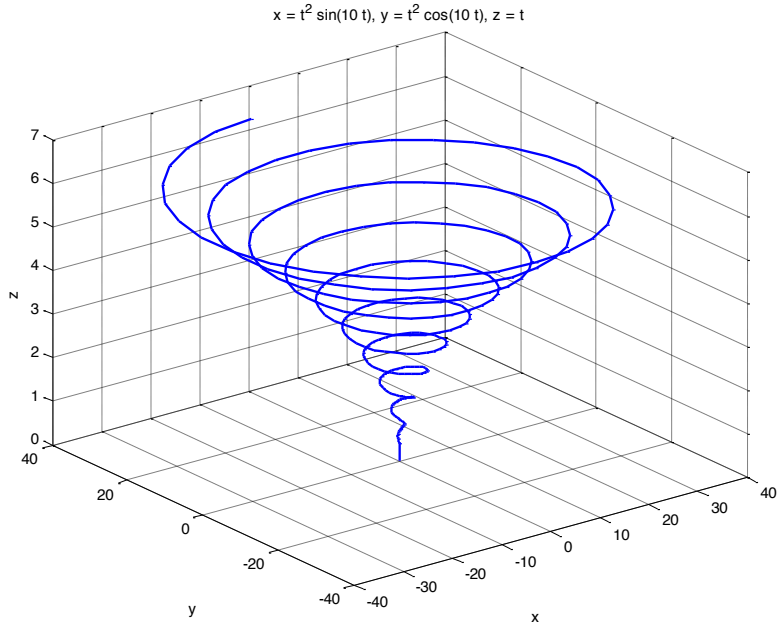
Obrázok 10-1 Grafické znázornenie funkcie $x^3 - 6x^2 + 11x - 6$

$ezplot3(f)$ - vykresľovanie funkcie f v 3D priestore

PRÍKLAD 4
Vykreslite funkcie $t^2 \sin(10t)$, $t^2 \cos(10t)$ a t v 3D priestore.

Riešenie v programovom prostredí MATLAB:

```
>> syms t
>>ezplot3(t^2*sin(10*t),t^2*cos(10*t), t)
```



Obrázok 10-2 Vykreslenie špirály v 3D

- **Riešenie algebraických rovníc príkazom- *solve*, výpočet derivácií a integrálov**

PRÍKLAD 5
Riešte rovnicu $x^2+5x+6=0$ použitím funkcie $solve('rovnica')$.

Riešenie príkladu 5 v programovom prostredí MATLAB:

```
>> sym('x');
>>  xx=solve('x^2+5*x+6=0')    %  je  možné  použiť  aj  zápis
xx=solve('x^2+5*x+6')
      xx =
      -3
      -2
```

PRÍKLAD 6
Vyriešte sústavu rovníc $x^2+xy+y=3$ a $x^2-4x+3=0$ pomocou funkcie $solve('rovnice')$.

Riešenie príkladu 6 v programovom prostredí MATLAB:

```
>> syms x y;
>> [x,y]=solve('x^2+x*y+y=3', 'x^2-4*x+3=0')
x =
1
3
y =
1
-3/2
```

⇨ $diff('f(x)','x')$ - derivácia funkcie

PRÍKLAD 7
Vypočítajte deriváciu funkcie $f(x)=x^2sin^2(x)$ pomocou funkcie symbolického toolboxu $diff('f(x)','x')$

Riešenie príkladu 7 v programovom prostredí MATLAB:

```
>> diff('x^2*sin(x^2)','x')
ans =
2*x*sin(x^2) + 2*x^3*cos(x^2)
```

PRÍKLAD 8
Vypočítajte parciálnu deriváciu funkcie $f=sin(x)^2+cos(y)^2$.

a) podľa x b) podľa y c) druhú deriváciu podľa y

Riešenie príkladu 8a) v programovom prostredí MATLAB:

```
>> syms x y % definícia symbolických premenných
>> f=sin(x)^2+cos(y)^2; % definícia funkcie
>> diff(f) % výpočet derivácie funkcie f
 ans = 2*cos(x)*sin(x)
```

Programové prostredie MATLAB vypočíta prvú deriváciu symbolickej premennej, ktorú vo výraze nájde ako prvú. V tomto prípade je to x. Ak by sme chceli derivovať podľa inej symbolickej premennej, musíme do výrazu $diff(f)$ pridať symbolickú premennú, podľa ktorej chceme derivovať.

Riešenie príkladu 8b) v programovom prostredí MATLAB:

```
>> syms x y % definícia symbolických premenných
>> f=sin(x)^2+cos(y)^2; % definícia funkcie
>> diff(f,y) % výpočet parciálnej derivácie podľa y
 ans = -2*cos(y)*sin(y)
```

Riešenie príkladu 8c) v programovom prostredí MATLAB

```
>> syms x y % definícia symbolických premenných
>> f=sin(x)^2+cos(y)^2; % definícia funkcie
>>  diff(f,y,2)  %  výpočet  druhej  parciálnej
derivácie podľa y
 ans = 2*sin(y)^2 - 2*cos(y)^2
```

⇨ $int('f(x)','x',a,b)$ – integrál funkcie, pričom **a** je dolná a **b** je horná hranica integrálu

PRÍKLAD 9
Vypočítajte integrál funkcie $f(x) = x * e^x$ pomocou funkcie symbolického toolboxu $int('f(x)','x',a,b)$

Riešenie príkladu 9 v programovom prostredí MATLAB:

```
syms x; % definícia symbolických premenných
>> int('x*exp(x)','x') % výpočet integrálu
ans =
exp(x)*(x - 1)
```

PRÍKLAD 10
Vypočítajte integrál funkcie $f(x) = x * e^x$ pomocou funkcie symbolického toolboxu $int('f(x)','x',a,b)$ v hraniciach [0, 5]

Riešenie príkladu 10 v programovom prostredí MATLAB

```
>>syms x % definícia symbolickej premennej
>>a=int('x*exp(x)','x',0,5)  %  výpočet  integrálu  v  hraniciach
<0,5>
a = 4*exp(5) + 1
>> double (a) % prepočet do symbolického výrazu na numerický
ans = 594.6526
```

- **Symbolické riešenie obyčajných diferenciálnych rovníc pomocou $dsolve$**

PRÍKLAD 11
Riešte diferenciálnu rovnicu

$$\frac{dy}{dt} = -ay$$

prvého rádu pomocou funkcie symbolického toolboxu $dsolve$

Riešenie príkladu 11 v programovom prostredí MATLAB:

```
>> syms a % definícia symbolickej premennej
ans =
 a
>> y=dsolve('Dy=-a*y') % výpočet zadanej diferenciálnej
rovnice
y =
C3/exp(a*t)
```

Pozn. Konštanty je možné dopočítať na základe počiatočných podmienok

PRÍKLAD 12
Riešte diferenciálnu rovnicu

$$\frac{d^2y}{dt^2} = -a^2y$$

pri počiatočných podmienkach $y(0) = 1$, $y'(0) = 0$ pomocou funkcie symbolického toolboxu ***dsolve***

Riešenie príkladu 12 v programovom prostredí MATLAB:

```
>> sym a; % definícia symbolickej premennej
>>  y=dsolve('D2y=-a^2*y','y(0)=1,  Dy(0)=0')  %  výpočet  diferenciálnej
premennej s počiatočnými podmienkami
y = (1/exp(a*t*i))/2 + exp(a*t*i)/2
```

- **Polynómy a symbolický toolbox**

Uvažujeme polynóm $P(x) = x^3 + 14x^2 + 53x + 40$ po zadefinovaní symbolickej premennej x:

```
>> x = sym('x');
>> P=x^3+14*x^2+53*x+40
 P =
 x^3 + 14*x^2 + 53*x + 40
```

⇨ ***poly2sym*** - vracia symbolickú reprezentáciu polynómu, ktorého koeficienty tvoria číselný vektor c

```
r = poly2sym(c)
```

poly2sym([1 3 2])

returns

```
ans =

x^2 + 3*x + 2
```

⇨ ***sym2poly*** - vracia riadkový vektor obsahujúci číselné koeficienty polynómu symbolicky

```
c = sym2poly(s)
```

syms x
sym2poly(x^3 – 2*x - 5)
returns

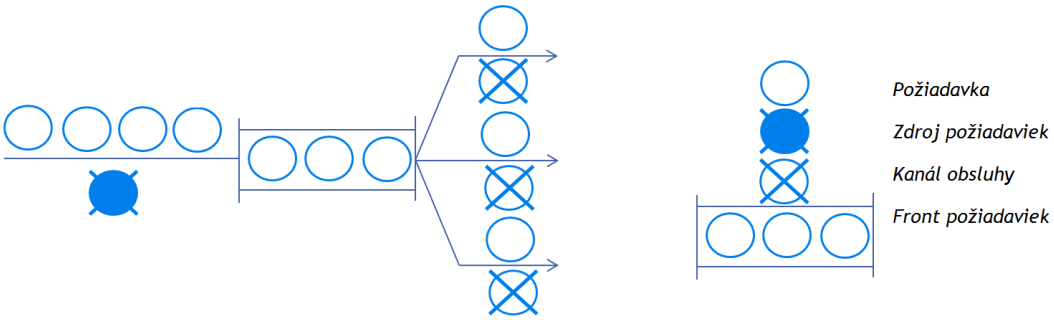
```
ans =

  1     0    -2    -5
```

11 Modelovanie systémov hromadnej obsluhy v prostredí MATLAB

11.1 Hromadná obsluha

Teória hromadnej obsluhy skúma kvantitatívnu stránku procesov súvisiacich s hromadnou obsluhou požiadaviek rôzneho charakteru. Ako obslužný systém môžeme chápať telefónnu ústredňu, lekársku ordináciu, čerpaciu stanicu pohonných hmôt, centrálnu jednotku počítača a pod. Vo všetkých týchto prípadoch je tu existencia jedného alebo viacerých kanálov, ktoré uspokojujú určité požiadavky. V prípade, že je v systéme aspoň jeden obslužný kanál voľný, požiadavka, ktorá vstúpi do systému je obslúžená a opúšťa ho. V prípade, že všetky kanály sú obsadené požiadavka sa zaradí do frontu a čaká, alebo opúšťa systém neobslúžená. Cieľom teórie hromadnej obsluhy je optimalizovať parametre obslužného systému napr. priemernú dĺžku frontu v systéme hromadnej obsluhy je možné ovplyvniť zmenou počtu obslužných kanálov, alebo zmenou intenzity obsluhy.

Systém hromadnej obsluhy tvorí súbor zariadení (osôb) schopných vykonávať obsluhu požiadavky – *kanály obsluhy* a súbor požiadaviek čakajúcich v systéme na obsluhu - *front požiadaviek*. Potenciálne požiadavky predstavujú *zdroj požiadaviek*. Systém hromadnej obsluhy možno schematický znázorniť nasledovne:



Obrázok 11-1 Systém hromadnej obsluhy

- **Klasifikácia systémov hromadnej obsluhy**

1. Podľa počtu obslužných kanálov:
   a. Jednokanálové
   b. Viackanálové
2. Podľa zdroja požiadaviek:
   a. Uzavretý systém (zdroj požiadaviek konečný)
   b. Otvorený systém (zdroj požiadaviek nekonečný)
3. Podľa charakteru vstupného prúdu:
   a. Stacionárnym alebo nestacionárnym vstupným prúdom
   b. Poissonovským alebo nepoissonovským vstupným prúdom
4. V prípade plného obsadenia kanálov sa delia na systémy:
   a. S čakaním
      i. Ohraničeným
      ii. Neohraničeným
   b. Bez čakania – s odmietnutím

V príkladoch budeme uvažovať o vstupnom prúde požiadaviek tvorenom Poissonovským procesom, hlavne z dôvodu, že mnoho reálnych procesov obsluhy spĺňa tento predpoklad a taktiež Poissonovský proces je podrobne rozpracovaný.

Poissonov proces je prúd javov, ktorý má tieto vlastnosti:

1. Nezávislosť prírastkov – počet javov, ktoré sa vyskytnú v určitom intervale, nezávisí od počtu javov v iných intervaloch
2. Stacionárnosť (homogenita v čase) – počet javov v ľubovoľných rovnako dlhých časových intervaloch je konštantný
3. Regulárnosť (ordinárnosť) –pravdepodobnosť výskytu viac než jedného javu v dostatočne malom časovom intervale Δtje zanedbateľne malá. Z toho vyplýva, že v intervale $(t, t+ \Delta t)$ môže nastať len jeden z prípadov:
    a. Vyskytne sa práve jeden javy s pravdepodobnosťou $\lambda \Delta t$
    b. Nevyskytne sa žiadny jav s pravdepodobnosťou $1- \lambda \Delta t$

Z čoho je jasné, že v Poissonovskom procese je možný len prechod systému od najbližšieho „vyššieho“ stavu, alebo zotrvanie v tom istom stave.

11.2 Príklady použitia hromadnej obsluhy

- **Otvorený jednokanálový systém hromadnej obsluhy bez čakania**

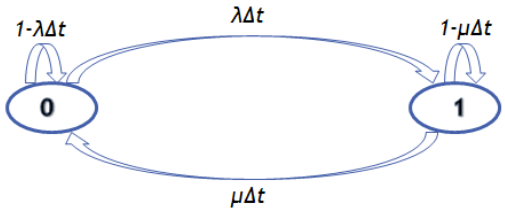
Pri modelovaní systému sa vychádza z nasledujúcich predpokladov:

1. Vstupný prúd je Poissonov proces s intenzitou vstupu λ
2. Čas obsluhy má exponenciálne rozdelenie so strednou hodnotou $\bar{t} = \frac{1}{\mu}$; μ je intenzita obsluhy.
   Koeficient zaťaženia systému $\Psi=\lambda/\mu$
3. Počet kanálov $n=1$
4. Front sa nevytvára; ak je systém obsadený, požiadavka bude odmietnutá
5. Zdroj požiadaviek je neohraničený

Systém má 1 obslužný kanál, do úvahy prichádzajú 2 možné stavy: S_0 – systém je voľný a S_1 – v systéme je jedna požiadavka, ktorá je práve obsluhovaná. Vychádzajúc z vlastností Poissonovho vstupného prúdu a exponenciálneho rozdelenia času obsluhy, možno pravdepodobnosti zmeny stavu systému popísať nasledovne (mocniny veličiny Δt sa zanedbávajú):

$S_0 \rightarrow S_0$	$1-\lambda\Delta t$	Do systému nevstúpi žiadna požiadavka
$S_0 \rightarrow S_1$	$\lambda\Delta t$	Do systému vstúpi jedna požiadavka
$S_1 \rightarrow S_0$	$(1-\lambda\Delta t)\mu\Delta t \doteq \mu\Delta t$	Do systému nevstúpi žiadna a práve jedna obslúžená požiadavka opustí systém
$S_1 \rightarrow S_1$	$1-\mu\Delta t$	Žiadna požiadavka neopustí systém alebo jedna požiadavka vstúpi do systému a jedna obslúžená systém opustí

Pravdepodobnosti zmeny stavu systému možno graficky znázorniť pomocou grafu:



Z toho vyplývajú vzťahy pre výpočet pravdepodobností:
$p_0(t+\Delta t)=p_0(t)(1-\lambda\Delta t) + p_1(t)\mu\Delta t$
$p_1(t+\Delta t)=p_0(t)\lambda\Delta t + p_1(t)(1-\mu\Delta t)$

Z čoho po úpravách dostávame diferenciálne rovnice:

$$\frac{dp_0(t)}{dt} = -\lambda p_0(t) + \mu p_1(t)$$
$$\frac{dp_1(t)}{dt} = \lambda p_0(t) - \mu p_1(t)$$

Počiatočné podmienky : $p_0(0)=1$, $p_1(0)=0$

Riešený príklad
Telefón na ústredni zvoní v priemere každých 12minút pričom jeden hovor trvá priemerne 6 minúty. Vstupný prúd možno považovať za Poissonovský proces a čas obsluhy za exponenciálne rozdelený. Určte:

a. pravdepodobnosť, že linka je obsadená – pravdepodobnosť odmietnutia
b. aké percento hovorov bude vybavené – relatívnu kapacitu
c. pomocou diferenciálnych rovníc znázornite priebeh ??? a porovnajte s vypočítanými hodnotami

Riešenie

λ = 1 volanie za 12 minút = 5 volaní/h

$\bar{t}$ = 6 minút na požiadavku = 1/10 h/pož.

$\mu = \frac{1}{\bar{t}}$ = 10 volaní/h

$\Psi = \frac{\lambda}{\mu}$ = 5/10 = 0,5

$P_{st} = p_1 = 1 - p_0 = 1/3$

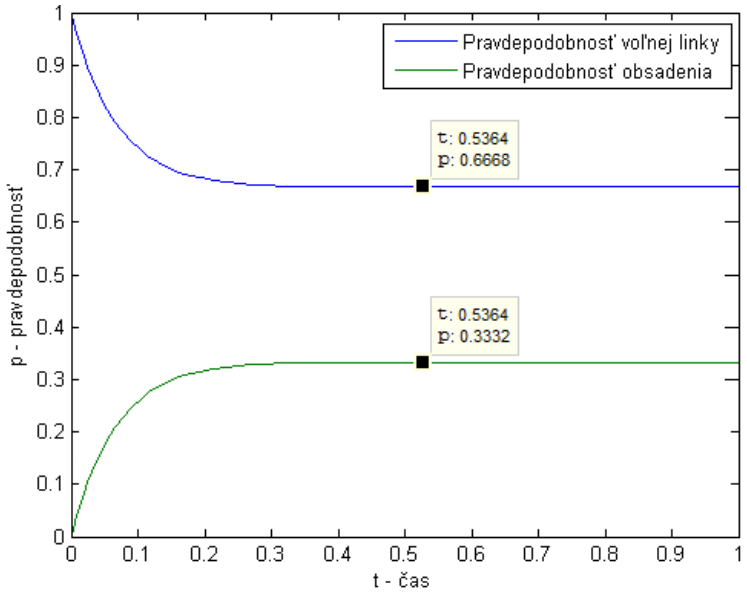
$K_r = p_0 = \frac{\mu}{\lambda+\mu} = 2/3$

Vytvoríme si funkciu :

```
function [pder] = ustredna(t,p)

pder = [-lambda*p(1)+mi*p(2)
        lambda*p(1)-mi*p(2)];


return
```

Hlavný program :

```
pp = [1 0];      % počiatočné podmienky
t =  [0 1];
lambda = 6;
mi = 12;

[t,pder] = ode45(@ustredna,t,pp);

plot(t,pder)
legend('Pravdepodobnosť voľnej linky','Pravdepodobnosť obsadenia')
```



Obrázok 11-2 Grafické znazornenie pravdepodobností voľnej a obsadenej linky

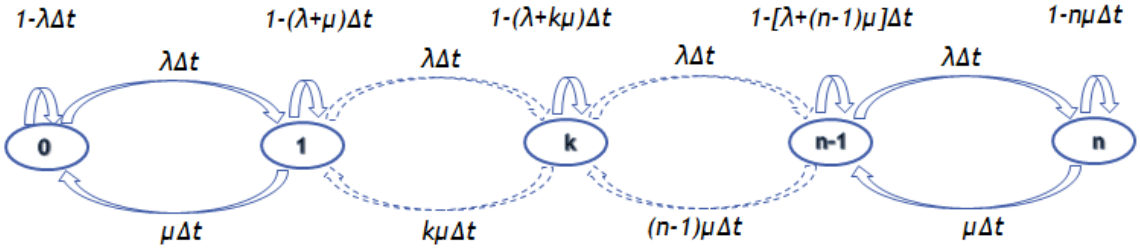
Z grafu vidíme, že priebeh pravdepodobností sa po určitom čase ustáli na tých hodnotách, ktoré nám vyšli aj z analytického výpočtu.

- **Otvorený viackanálový systém hromadnej obsluhy bez čakania**

Predpoklady modelu sú zhodné s predpokladmi otvoreného jednokanálového systému hromadnej obsluhy bez čakania okrem bodu 3., kde platí, že n>1. Takto systém sa môže nachádzať v jednom z *n+1*stavov:

S_0 – všetky kanály sú voľné
S_1 – jeden kanál je obsadený
...
S_n – všetkých *n* kanálov je obsadených

Graf prechodov otvoreného viackanálového systému bez čakania:



Sústava diferenciálno- diferenčných rovníc systému má nasledovný tvar :

$$\frac{dp_0(t)}{dt} = -\lambda p_0(t) + \mu p_1(t)$$

$$\frac{dp_1(t)}{dt} = \lambda p_0(t) - (\lambda + \mu)p_1(t) + 2\mu p_2(t)$$

.....

$$\frac{dp_k(t)}{dt} = \lambda p_{k-1}(t) - (\lambda + k\mu)p_k(t) + (k+1)\mu p_{k+1}(t)$$

.....

$$\frac{dp_n(t)}{dt} = \lambda p_n(t) - n\mu p_n(t)$$

Začiatočné podmienky sú: $p_0(0)=1$, $p_1(0)=p_2(0)=...=p_n(0)=0$

Riešený príklad

Do kancelárie informačnej služby so štyrmi linkami prichádza priemerne 1 výzva za 72 sekúnd. Priemerná dĺžka hovoru je 3 minúty. Za predpokladu Poissonovho vstupného prúdu a exponenciálneho času obsluhy vypočítajte :

a. pravdepodobnosť odmietnutia a pravdepodobnosť obslúženia (relatívna kapacita)

b. pomocou diferenciálnych rovníc znázornite priebeh stavov jednotlivých pravdepodobností a porovnajte s vypočítanými hodnotami.

Riešenie

n = 4

λ = 1 volanie za 72 sekúnd = 50 volaní/h

$\bar{t}$ = 3 minúty na požiadavku = 1/20 h/pož.

μ = 1/$\bar{t}$ = 20 volaní/h

Ψ = λ/μ = 50/20 = 2,5

K	Ψ^k	$k!$	$\frac{\Psi^k}{k!} = \frac{p_k}{p_0}$	p_k	kp_k
0	1	1	1	0,0921	0
1	2,5	1	2,5	0,2303	0,2303
2	6,25	2	3,125	0,2878	0,5756
3	15,625	6	2,6042	0,2399	0,7197
4=n	39.0625	24	1,6276	0,1499	0,5996
Σ			10,8568=1/p_0	1	2,125

p0=1/10,8568=0,0921

Pravdepodobnosť odmietnutia: ps t = pn = Ψ^k/n! p_0 = 0,1499

Relatívna kapacita: Kr = 1 - pst = 0,8501

Z diferenciálnych rovníc dostávame maticu:

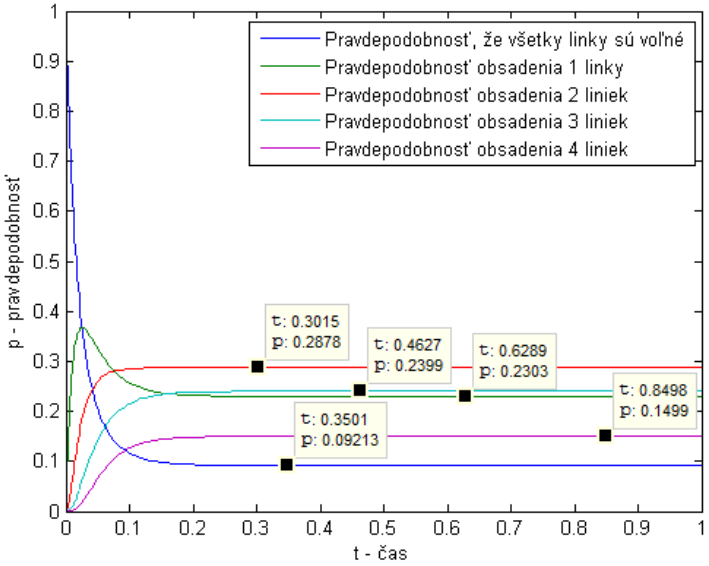
S_0:	-λ	μ	0	0	0
S_1:	λ	-λ-μ	2μ	0	0
S_2:	0	λ	-λ-2μ	3μ	0
S_3:	0	0	λ	-λ-3μ	μ
S_4:	0	0	0	λ	-4μ

Vytvoríme si funkciu :

```
function [pder] = ustredna4(t,p)

lambda = 50;
mi = 20;

pder = [-lambda*p(1)+mi*p(2);
        lambda*p(1)-mi*p(2)-lambda*p(2)+2*mi*p(3);
        lambda*p(2)-lambda*p(3)-2*mi*p(3)+3*mi*p(4);
        lambda*p(3)-lambda*p(4)-3*mi*p(4)+4*mi*p(5);
        lambda*p(4)-4*mi*p(5)];

return
```

Hlavný program:

```
pp = [1 0 0 0 0];      % počiatočné podmienky
t =  [0 0.5];

[t,pder]=ode45(@ustredna4,t,pp);
plot(t,pder)
legend('Pravdepodobnosť, že všetky linky sú voľné',...
    'Pravdepodobnosť obsadenia 1 linky','Pravdepodobnosť obsadenia 2 liniek',...
    'Pravdepodobnosť obsadenia 3 liniek','Pravdepodobnosť obsadenia 4 liniek')
```



Obrázok 11-3 Riešenie systému diferenciálnych rovníc modelujúcich SHO

Príklady na riešenie

1. Na benzínovú čerpaciu stanicu prichádza priemerne 25 vozidiel za hodinu. Stanica má tri čerpacie stojany, z ktorých každý obslúži zákazníka priemerne za 5 minút. Na čerpacej stanici nie je miesto na čakanie. Za predpokladu poissonovského vstupného toku a exponenciálnej doby obsluhy vypočítajte pravdepodobnosť straty zákazníka a relatívnu kapacitu.

Literatúra

[1] **Bartová, S. – Kukal, J. – Panek, M.:** Algoritmizace v systému Matlab, Výpočetní technika II, Ústav počítačové a řídící techniky VŠCHT, Praha

[2] **Kozák, Š. – Kajan, Š.:** MATLAB/SIMULINK 1, 2, FEI STU, Bratislava, 1999

[3] **Zaplatílek, K. – Donař, K**.: Matlab pro začátečníky, BEN, Praha 2003

[4] **Novák, J. – Pultarová, I. – Novák, P.:** Základy informatiky, Počítačové modelování v Matlabu, ČVUT Praha, 2005

[5] **Karban, P.:** Výpočty a simulace v programech Matlab a Simulink, Computer Press, a.s., Brno, 2006

[6] **Žáková, K.:** Základy práce v Matlabe, FEI STU Bratislava, 2006

[7] **Moore, H.:** Matlab for Engineers, third edition, Pearson, 2012

Prílohy

Príloha A, Príloha B