SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLOGIE

ÚSTAV INFORMATIZÁCIE, AUTOMATIZÁCIE A MATEMATIKY

TVORBA GUI PRE SYNTÉZU ROBUSTNÝCH PI REGULÁTOROV

BAKALARÁSKA PRÁCA

FCHPT-5415-50920

2010 **Martin Klaučo**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLOGIE

ÚSTAV INFORMATIZÁCIE, AUTOMATIZÁCIE A MATEMATIKY

TVORBA GUI PRE SYNTÉZU ROBUSTNÝCH PI REGULÁTOROV

BAKALARÁSKA PRÁCA

FCHPT-5415-50920

Študijný program: Automatizácia, informatizácia a manažment v chémii a potravinárstve
Číslo a názov študijného odboru: 5.2.14 automatizácia, 5.2.25 priemyselné inžinierstvo
Školiace pracovisko: Radlinského 9, 812 37 Bratislava
Vedúci záverečnej práce/školiteľ: Ing. Katarína Matejičková

2010 **Martin Klaučo**

Slovenská technická univerzita v Bratislave
Oddelenie informatizácie a riadenia procesov

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2009/2010
Evidenčné číslo: FCHPT-5415-50920

S T U
F C H P T

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Martin Klaučo**
ID študenta:	50920
Študijný program:	automatizácia, informatizácia a manažment v chémii a potravinárstve
Kombinácia študijných odborov:	5.2.14 automatizácia, 5.2.52 priemyselné inžinierstvo
Vedúca práce:	Ing. Katarína Vaneková
Miesto vypracovania:	Bratislava

Názov práce: **Tvorba GUI pre syntézu robustných PI regulátorov**

Špecifikácia zadania:

Tvorba grafického rozhrania pre syntézu robustných PI regulátorov v Matlabe. Doplnenie programu syntézy robustných regulátorov o výber aproximácie dopravného oneskorenia. Testovanie programu a vytvoreného grafického rozhrania pre syntézu robustných PI regulátorov.

Rozsah práce: 45

Riešenie zadania práce od: 15. 02. 2010
Dátum odovzdania práce: 22. 05. 2010

Slovenská technická univerzita v Bratislave Fakulta chemickej a potravinárskej technológie dekanát - 2 -

L. S.

Martin Klaučo 20
Martin Klaučo
Študent

prof. Ing. Miroslav Fikar, DrSc.
Vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
Garant študijného programu

Abstrakt

Bez grafického rozhrania – GUI *(Graphical User Interface)* – by väčšina užívateľov nebola schopná používať počítač. Dobre navrhnuté GUI je dôležitou súčasťou zavádzania programu do praxe. Cieľom tvorby akéhokoľvek GUI je odbremeniť užívateľa od používania CLI (*command-line interface*). Cieľom tohto bakalárskeho projektu je naučiť sa základy tvorby GUI v prostredí MATLAB a následne vytvoriť fungujúci program. GUI je možné v prostredí MATLAB tvoriť rôzne. V tejto práci sa budeme venovať profesionálnemu prístupu tvorby GUI pomocou príkazov uicontrol a figure. Vytvorené GUI bude nadstavba programu, ktorý sa zaoberá syntézou robustných PI regulátorov.

Abstract

Without graphical user interface – GUI, most users would not be able to use computer. Good design GUI is important part of introducing program to practice. Aim of creating any GUI is to unburden client of using CLI (*command-line interface*). Aim of this bachelor project is to learn basis of design GUI in MATLAB environment and then create a working program. There are several possibilities how to create GUI in MATLAB. In this thesis we are going to present professional way of creating GUI using command like uicontrol and figure. Designed GUI will be a system upgrade for program which synthesizes robust PI controllers.

Poďakovanie

Chcel by som poďakovať vedúcej bakalárskeho projektu Ing. Kataríne Matejičkovej za odborné vedenie pripomienky a cenné rady.

Obsah

Úvod

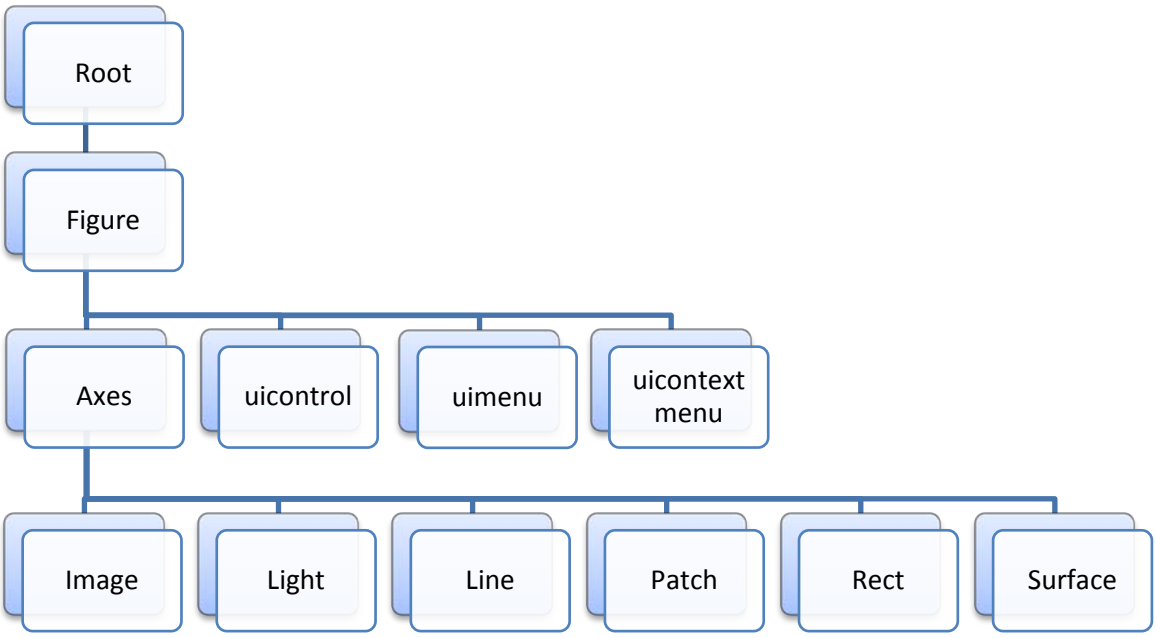
Grafické rozhranie je potrebné vytvoriť pre rôzne aplikácie z viacerých dôvodov – spríjemniť, zjednodušiť, urýchliť prácu s programom pre koncového užívateľa. Nebyť GUI – *graphical user interface* – museli by sme všetky operácie, ktoré chceme, aby nám počítač vykonal zadávať pomocou príkazového riadku. Grafické rozhranie by preto malo byť čo najviac *user-friendly* – intuitívne, jednoduché na používanie, prehľadné. Malo by obsahovať len tie najpotrebnejšie informácie, aby bol koncový užívateľ čo najviac odbremenený od používania príkazového riadku CLI – *command-line interface*. Na tvorbu GUI je vytvorených mnoho vývojárskych prostredí, avšak pre vedecké potreby sa naskytuje prostredie MATLAB. V tomto programe sa dajú elegantne spojiť matematické operácie (ako sú rôzne výpočty, vykresľovanie grafov, simulácie), vykonávané na úrovni príkazov v command-line a grafické rozhranie, v ktorom sa dajú použiť už hotové príkazy.

V tejto práci sa budeme venovať tvorbe GUI v prostredí MATLAB, ktoré bude plniť funkcie vytvárania grafov, testovania vybraných regulátorov v simuláciách, a ukladania výsledných dát [1].

Úlohou tejto práce je vytvoriť grafické rozhranie pre program návrhu robustných regulátorov. Robustné regulátory sú navrhované pre modely s dopravným oneskorením prvého a druhého rádu. Samotný program je spojením návrhu robustných regulátorov v (k_p, k_i) – rovine a metódy umiestnenia pólov, čím sa zvolenému regulátoru predpisujú zvolené kritériá kvality [2].

1 Základy GUI v prostredí MATLAB

Výpočtové prostredie MATLAB, okrem iným možností a toolbox-ov ponúka aj tvorbu grafických rozhraní (okien). Pomocou grafických okien sme schopní následne spúšťať *m-file* (scripty) a vykonávať rôzne operácie bez použitia CLI. MATLAB má v sebe zakomponované príkazy napríklad `plot()`, ktoré už spadajú do kategórie GUI, pretože ich výstup sa nevypisuje na príkazový riadok. Vykreslenie resp. generovanie okien sa vykonáva podľa istej hierarchie. Je dôležité si uvedomiť, aký sled krokov treba dodržiavať pri tvorení GUI, aby sme sa dopracovali k požadovaným výsledkom. Nasledujúci diagram znázorňuje hierarchiu grafických objektov v prostredí MATLAB [3].

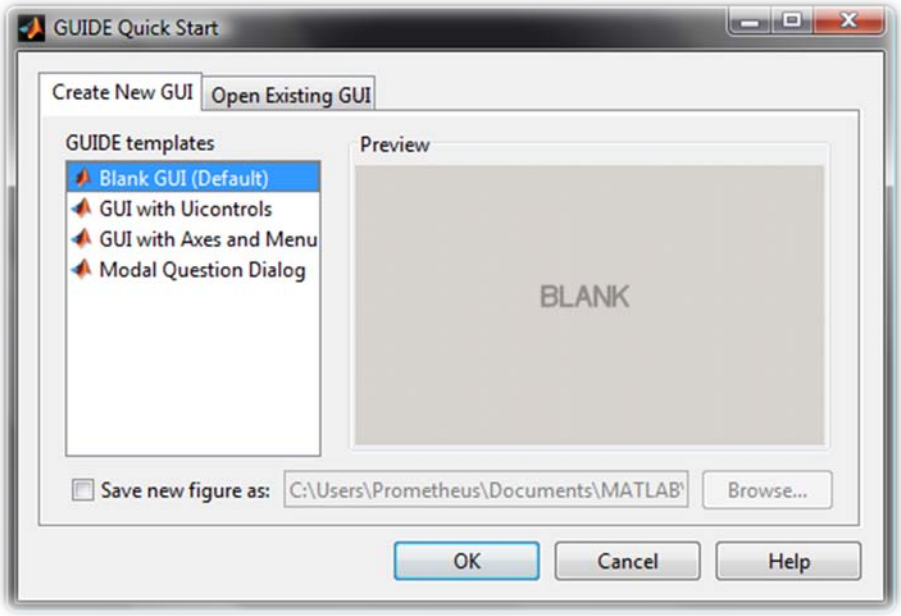


Obr. 1 - Hierarchia grafických objektov

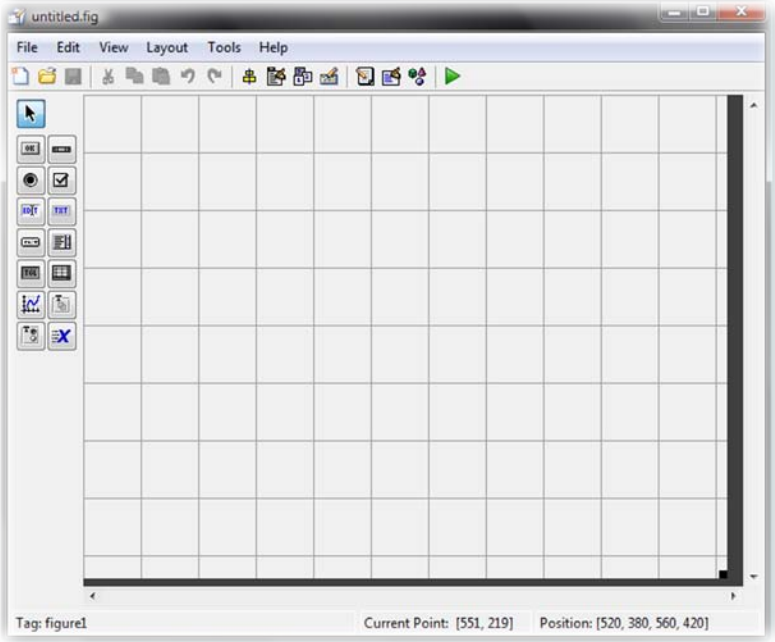
Prvá pozícia *Root* zodpovedá obrazovke. Položka *Figure* zodpovedá oknu, do ktorého budeme vkladať grafické prvky. Ďalej nás budú zaujímať položky *Axes* a *uicontrol*. Axes je potrebné na vkladanie grafov a obrázkov, ako napovedá rozvetvenie pod položkou Axes (Image, Surface...). Blok *uicontrol* sa používa na grafické prvky ako je textové pole, tlačidlo, radio button, atď. Správnym nastavením položiek *Figure*, *Axes*, *uicontrol*, dostaneme nástroj, ktorý môže kompletne nahradiť používanie príkazového riadku, čiže CLI.

1.1 Prostredie GUIDE

Generátor GUI sa volá GUIDE (obr. 2 a 3) – *Graphical User Interface Development Environment*. Už aj táto softwarová časť MATLAB-u je sama o sebe GUI, pretože príkazový riadok sa používa veľmi málo a grafické prvky sa vytvárajú jednoduchými kliknutiami myšou.



Obr. 2 - Spustenie GUIDE



Obr. 3 - Prostredie GUIDE

Na ľavom paneli si môžeme všimnúť grafické prvky, ktoré možno vkladať do okna. Výhoda tejto metódy je *Zr*ejmá. Metódou Drag-&-Drop si užívateľ vloží potrebné grafické prvky a následne ich môže nastaviť podľa potreby.

Toolbox GUIDE vytvorí súbor .fig, ktorý zodpovedá v hierarchii práve položke *figure*. V tomto súbore budú uložené všetky náležitosti grafických prvkov, ktoré sme do okna vložili. Zároveň sa vygeneruje rovnomenný m-file, v ktorom budú uložené príkazy, ktoré sa majú vykonávať napríklad po stlačení tlačidla.

Nevýhoda tvorby GUI týmto spôsobom je tá, že je to spätne nekompatibilné so staršími verziami MATLAB-u. Následne je dosť náročné sa orientovať v kóde, ktorý za nás GUIDE vytvára. Prehľadnejšia metóda tvorby GUI v MATLAB-e, avšak náročnejšia, je pomocou príkazu `uicontrol` – podkapitola 1.3.

Medzi ďalšie nevýhody patrí generovanie *m-file*. Tento skript je delený na funkcie, pričom ku každému grafickému prvku pripadá práve jedna funkcia. Ak sa vykonávajú nejaké príkazy, ktoré sú výsledkom povedzme stlačenia tlačidla, tak sa vykonávajú vo vnútornom *workspace* funkcie. Táto skutočnosť sťažuje prácu s premennými, ktoré sa počítajú v rámci týchto funkcií, keďže nie sú k dispozícii pre ostatné funkcie. Aj pre tento fakt je odporúčané vytvárať GUI pomocou príkazov `figure` a `uicontrol`, ktoré sú popísané v ďalších kapitolách.

1.2 Figure

Figure je okno v ktorom, umiestňujeme grafické prvky. Toto okno možno vytvoriť pomocou GUIDE ako s uvádza v predošlej podkapitole. Profesionálnejšia cesta je však vygenerovať si *figure* pomocou príkazu a zároveň aj s vlastnosťami, ktoré toto okno má mať. Odporúčaná syntax tohto príkazu je nasledovná:

```
figure_handle = figure('PropertyName','PropertyValue')
```

kde `figure_handle` predstavuje číselný identifikátor, ktorý je potrebný pre neskoršiu úpravu *figure*; *PropertyName* je názov vlastnosti a *PropertyValue* je hodnota vlastnosti. Táto syntax sa používa aj pri konfigurácií grafických prvkov. V nasledujúcich kapitolách sa tomu budeme bližšie venovať. Vymenujme najzákladnejšie vlastnosti, ktoré by každé okno malo mať definované.

Vlastnosť (PropertyName)	Hodnota (PropertyValue)	Vysvetlenie
Position	[x y w h]	x – súradnica x y – súradnica y w – šírka h - výška
name	reťazec	Názov figure, zobrazovaný v hlavičke okna
MenuBar	none / figure	Určuje, či sa zobrazí záhlavie okna
Numbertitle	On / Off	Poradie *figure*
resize	On / Off	Umožňuje užívateľovi meniť rozmery okna

Tab. 1 - Vlastnosti figure

1.3 Uicontrol

V prípade GUIDE, na tvorbu GUI použijeme GUI samotné, avšak teraz použijeme CLI. Napíšeme kód, ktorý nám GUI vygeneruje. Použijeme už niekoľkokrát spomenutý príkaz `uicontrol [4]`.

Príkazom `uicontrol` vytvárame grafické prvky. Už priamo v argumentoch príkazu, môžeme nastaviť presné atribúty grafických prvkov. Tento príkaz by mal obsahovať aspoň tri vstupné argumenty. Prvý z nich určuje, kde sa má grafický objekt vykresliť – v ktorom figure. Druhé dva určujú, aký grafický prvok sa má vykresliť. Ďalšie argumenty slúžia na to, aby sa vykreslil grafický prvok už s danými vlastnosťami. Odporúča sa, zadefinovať čo najviac vlastností už pomocou uicontrol, pretože potom nie sú potrebné ďalšie príkazy na zmenu vlastnosti grafických prvkov.

Syntax príkazu `uicontrol`:

```
handle = uicontrol(gcf,'Style','PushButton')
```

- `handle` – premenná do ktorej sa priradí číselný identifikátor daného grafického prvku
- `gcf` – get current figure – zistí aktuálny figure, určí do ktorého figure sa bude grafický prvok vykresľovať.
- `Style` – *property name* – názov vlastnosti
- `PushButton` – *property value* – hodnota vlastnosti

Vyššie uvedený príkaz by vykreslil tlačidlo – button.

1.4 Property Inspector

Property Inspector – ako už názov napovedá – je zoznam všetkých vlastností grafického prvku. Je možné ho vyvolať dvoma spôsobmi. Pokiaľ pracujeme v GUIDE, tak keď pravým tlačidlom myší klikneme na vytvorený prvok, tak sa nám v zobrazí menu, v ktorom je odkaz priamo na Property Inspector (obr. 4).

V prípade, že pracujeme v CLI a GUI generujeme pomocou príkazov, tak neexistuje príkaz, ktorý by Property Inspector takto vyobrazil. Je za neho však náhrada, ktorá je vhodná práve pre programátorov. Používa sa na to príkaz `set`, resp. `get`. Príkaz `set` sa dá použiť niekoľkými spôsobmi. Musí obsahovať aspoň jeden argument. A to je práve číselný identifikátor grafického objektu, ktorý sme vytvorili príkazom `uicontrol`. Ak napíšeme do príkazového riadku `set(handle)`, tak sa v príkazovom riadku vypíšu nielen všetky nastaviteľné vlastnosti objektu, ale aj možnosti hodnôt vlastností (obr. 5). Podobne pracuje aj príkaz `get(handle)`, avšak s tým rozdielom, že nevypíše možnosti nastavenia daného *handle*.

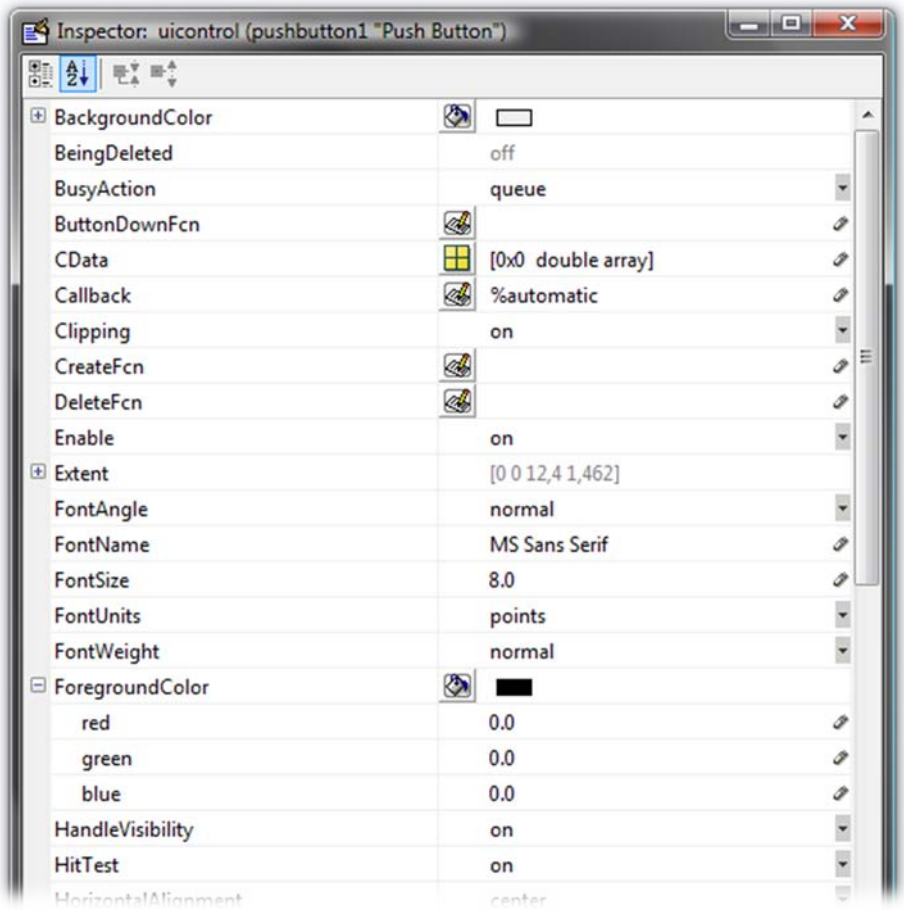
Príkaz set možno použiť aj v tvare `set(handle,'PropertyName')`, čo nám v príkazom riadku zobrazí len možnosti nastavenia hodnoty pre konkrétnu vlastnosť. V prípade, že potrebujeme zmeniť jednu alebo viacero vlastností, tak použijeme príkaz set, ktorý má nasledovnú syntax:

```
set(handle,'PropertyName','PropertyValue',…)
```

Príkaz get, slúži na zistenie konkrétnej hodnoty vlastnosti grafického prvku. Syntax je obdobná s príkazom set. V príkazovom riadku sa vypíše aktuálna hodnota, ktorú má vlastnosť daného objektu:

```
get(handle,'PropertyName')
```

V tabuľke 2 sú uvedené najčastejšie používané a zároveň najdôležitejšie vlastnosti grafických prvkov.



Obr. 4 - Ukážka Property Inspector

```
>> set(handle)
    BackgroundColor
    Callback: string -or- function handle -or- cell array
    CData
    Enable: [ {on} | off | inactive ]
    FontAngle: [ {normal} | italic | oblique ]
    FontName
    FontSize
    FontUnits: [ inches | centimeters | normalized | {points} | pixels ]
    FontWeight: [ light | {normal} | demi | bold ]
    ForegroundColor
    HorizontalAlignment: [ left | {center} | right ]
    KeyPressFcn: string -or- function handle -or- cell array
    ListboxTop
    Max
    Min
    Position
    String
    Style: [ {pushbutton} | togglebutton | radiobutton | checkbox | edit | text |
    SliderStep
    TooltipString
    Units: [ inches | centimeters | normalized | points | {pixels} | characters ]
    Value
```

Obr. 5 - Property Inspector CLI

Vlastnosť (PropertyName)	Hodnota (PropertyValue)	Vysvetlenie
Style	PushButton, Text, Edit, ...	Typ grafického prvku
String	reťazec	Zobrazovaný text
Position	[x y w h]	x – súradnica x y – súradnica y w – šírka h - výška
Cdata	matica typu `uint8`	Slúži na vykreslenie obrázku v danom grafickom prvku namiesto *String*
FontName	názov písma (Calibri, Arial)	Typ písma
FontSize	číslo	Veľkosť písma v bodoch (10,12,...).
FontWeight	bold, demi, light, normal	štýl písma
BackgroundColor	[R G B]	Farba pozadia zapísaná vo formáte RGB, pričom hodnoty R,G,B smú nadobúdať čísla v intervale $\langle 0; 1 \rangle$.
ForegroundColor	[R G B]	Farba písma zapísaná vo formáte RGB, pričom hodnoty R,G,B smú nadobúdať čísla v intervale $\langle 0; 1 \rangle$.
Visibility	On / Off	Určuje, či bude daný prvok zobrazený
Enable	On / Off / Inactive	Určuje aktivitu grafického prvku, napr.: možnosť stlačenia tlačidla.
CallBack	príkazy	Určuje, čo sa má vykonať po stlačení tlačidla.

Tab. 2 - Vlastnosti uicontrol

2 Požiadavky na grafické rozhranie

Ako sme v predchádzajúcich kapitolách viackrát spomenuli, GUI – grafické rozhranie – musí byť čo najjednoduchšie, user-friendly. tieto požiadavky sú znásobené tým, že ide o výpočtový toolbox, ktorého primárna úloha spočíva v tom, aby užívateľa odbremenila od používania príkazového riadku (CLI). Návrhu GUI predchádza dobre poznať, čo sa deje vo vnútri programu. V nasledujúcej kapitole stručne zhrnieme, aké matematické operácie vykonávané na úrovni CLI je potrebné „zabaliť“ do GUI.

2.1 Matematický základ

Primárnou úlohou toho programu bude navrhnúť PI regulátory. Tomuto úkonu predchádzajú niekoľké kroky, ktoré treba spraviť ako prvé. PI regulátor sa v tomto programe navrhuje ako prienik všetkých stabilných oblastí. V prípade, že chceme navrhnúť konkrétnejšie regulátory, tak použijeme metódu umiestnenia pólov [5], čím regulátorom predpisujeme isté kritériá – parameter ξ_{CL}. Regulátor sa navrhuje pre konkrétny systém, ktorý je opísaný matematickým modelom ako systém s dopravným oneskorím prvého alebo druhého rádu. Dopravné oneskorenie následne aproximujeme a zahrnieme do prenosu. V našom prípade budeme mať zadaných celkovo dvanásť prenosov. tieto prenosy sú odvodené pomocou rôznych aproximácií z prenosov prvého a druhého rádu s dopravným oneskorením [2]. Tento program bude navrhovať regulátory pre niektorý z týchto prenosov a následne odsimuluje stabilitu jednoduchého uzavretého regulačného obvodu.

Ako prvé je potrebné zvoliť si vhodnú aproximáciu či už prvého alebo druhého rádu. Na tento účel slúži simulácia, ktorá bude porovnávať jednotlivé aproximácie. Pre prvý aj druhý rád máme odvodených šesť aproximácií dopravného oneskorenia. Lineárne a kvadratické aproximácie – Taylorov rozvoj čitateľa, Taylorov rozvoj menovateľa, Padého rozvoj. Po zvolení aproximácie, program vypočíta stabilné oblasti, ktoré sa zobrazia v grafe – k_p, k_i rovine. Následne sa dopočítajú regulátory metódou umiestnenia pólov, čo sa zobrazí do toho istého grafu. Súradnice bodov na novej dokreslenej čiare zodpovedajú parametrom PI regulátora. X-ová súradnica (k_p) je zhodná s parametrom P, a y-ová súradnica je zhodná s parametrom I v Simulinkovom bloku PID. Vstupné údaje do celého programu sú: Z – zosilnenie modelu, T – časová konštanta modelu, D – dopravné oneskorenie modelu, ξ – koeficient tlmenia pre systémy druhého rádu.

Na výpočet regulátorov je potrebné zadať ξ_{CL} - koeficient tlmenia uzavretého regulačného obvodu a ω_{CL} – frekvencia na základe, ktorej sa vypočítavajú stabilné oblasti a regulátory. Parameter ω_{CL} sa vo väčšine prípadov vypočíta.

Kvôli simuláciám je potrebné zadať čas simulácie - time, Set Point (SPT) – žiadaná hodnota. Čas skokovej zmeny je prednastavený na 10 sekúnd.

Tento program navrhuje robustné PI regulátory pre systémy s neurčitosťami vo veličinách *Z, T, D*. Z tohto dôvodu bude potrebné zadávať minimálnu a maximálnu hodnotu. Strednú hodnotu, pri ktorej sa tiež bude simulovať, dopočíta program. Po získaní súradníc sa odsimuluje jednoduchý uzavretý regulačný obvod. Výsledky zo simulácií sa zobrazia ako grafy. Simulácie sa budú robiť pre krajné hranice a stredné hodnoty parametrov *Z, T, D*.

2.2 Návrh grafického rozhrania

Ako prvé je potrebné si vybrať prenos prvého alebo druhého rádu s dopravným oneskorením. Na toto bude slúžiť úvodné okno s dvoma tlačidlami. Následne je potrebné si vybrať aproximáciu. Výber aproximácie je však podmienený simuláciou a porovnaním s originálnym prenosom. Na toto bude slúžiť okno, v ktorom zadáme konštanty *Z, T, D* poprípade ξ, ak pôjde o prenos druhého rádu, a necháme odsimulovať všetky aproximácie. Výsledky simulácie sa zobrazia v grafoch. Na základe prechodovej funkcie si môžeme zvoliť aproximáciu dopravného oneskorenia.

Ďalšie okno, v ktorom sa budú vykresľovať stabilné oblasti a regulátory, musí obsahovať editovacie polia, v ktorých možno zmeniť resp. zadať konštanty *Z, T, D*, ξ, ξ_{CL} a ω_{CL}. Regulátory možno navrhovať pre viaceré hodnoty ξ_{CL} a ω_{CL}, preto sa musí dať voliť farba, akou sa bude vykresľovať krivka, ktorá má reprezentovať možné regulátory. Výber farby bude ošetrený samostatným oknom, ktoré sa zobrazí pred počítaním robustného PI regulátora. V niektorých prípadoch program nevypočíta hodnotu ω_{CL}, alebo ju vypočíta ako nulovú, čo znamená, že sa musí dať zadať užívateľom. Na tento úkon slúži príkaz `inputdlg()`.

Pre lepšiu orientáciu v grafe je dôležitá možnosť upravovať osi a pridať mriežku. Graf bude možné uložiť ako *fig* súbor, s ktorým užívateľ môže naložiť podľa potreby. Na názov súboru sa program opýta pred uložením.

Ako sme spomenuli v predchádzajúcej podkapitole, súradnice bodov v grafe majú súvis s navrhovaným PI regulátorom. Aby bol tento program čo najvšeobecnejší, je tam vložený aj

prepočet medzi P a I konštantami, s ktorými pracuje program na *Zr*, *Ti* konštanty. Funkciou `ginput()` môžeme odčítať jednoduchým kliknutím na bod v grafe jeho súradnice. Táto funkcia sa bude spúšťať tlačidlom, ako možnosť uloženia grafu, výpočtu regulátorov a stabilných oblastí prenosu.

Po zvolení regulátora sa môžeme prepnúť do simulačného okna, kde programu necháme odsimulovať vybraný regulátor. Budú k dispozícií textové, polia v ktorých sú zobrazené hodnoty *Z, T, D*, ξ. Ďalej sú potrebné textové polia, v ktorých bude zadávať SPT – žiadaná hodnota, time (čas simulácie) a samozrejme budú načítané parametre regulátora odčítané pomocou funkcie `ginput()`. Všetky parametre v textových poliach bude možné zmeniť podľa potreby užívateľa. Výsledkom simulácie budú grafy. Grafy však budú štyri. Prvý a druhý bude určený pre krajné hodnoty parametrov *Z, T, D*. Tretí pre priemernú hodnotu a posledný štvrtý pre porovnanie všetkých simulácií. Grafy sa budú dať uložiť ako .fig súbory a zároveň sa budú dať uložiť aj údaje zo simulácie ako vektor resp. matice v .txt súbore. Program sa taktiež opýta na názov tohto .txt súboru pred uložením.

Medzi oknami sa bude dať voľne prepínať. V každom okne bude tlačidlo *Quit*, ktorým sa celý program bude dať vypnúť.

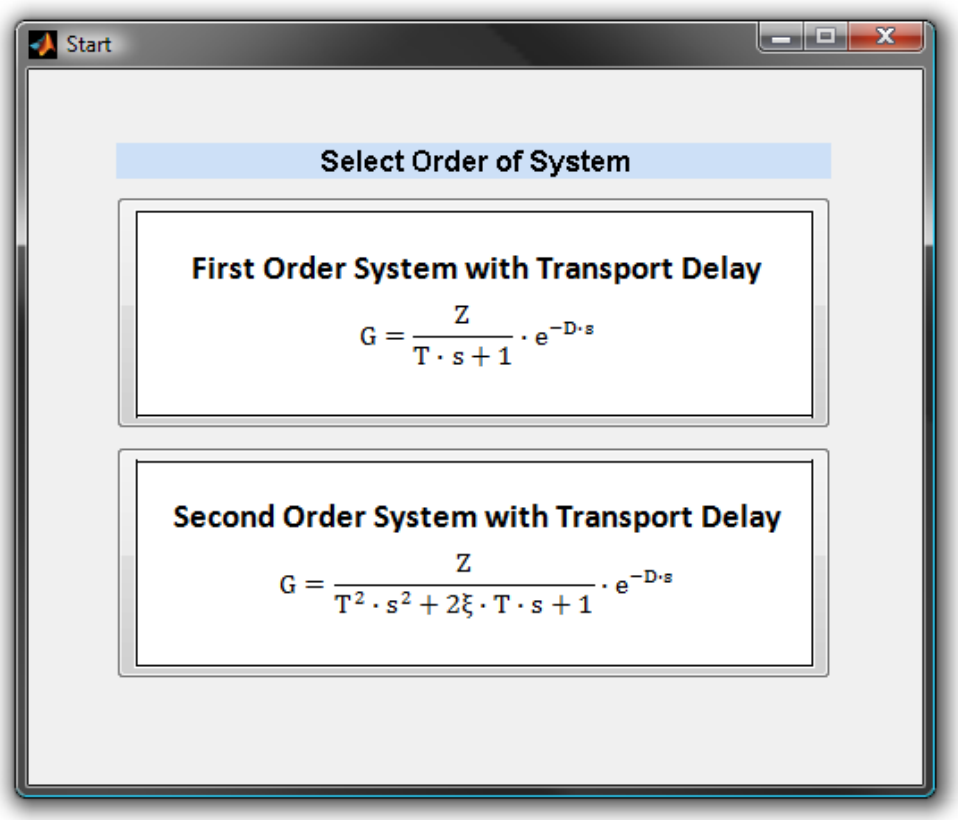
Všetky veličiny v tomto programu budú popisované a označované v angličtine.

3 Tvorba GUI

Po spoznaní matematických operácií, ktoré musí program vykonávať, po návrhu GUI sa môžeme pustiť to tvorby konkrétnych okien. Každé okno bude mať definovaný názov, nebude zobrazené poradie okna, nebude zobrazené menu okna a nebude sa dať meniť jeho veľkosť – iba pomocou príkazu `set`.

3.1 Úvodné okno

Úvodné okno (obr. 6) slúži len na výber rádu identifikovaného systému. Máme na výber dva systémy, systém prvého rádu s dopravným oneskorením alebo systém druhého rádu s dopravným oneskorením. Výber rádu spočíva len v tlačidle, ktorého stlačením sa zapíše hodnota do premennej na základe ktorej sa budú vykonávať ostatné príkazy. Väčšina skriptov je tvorená na základe podmienok (*if, elseif, else*) pretože jadro programu zostáva rovnaké, menia sa iba konštanty súvisiace s jednotlivými aproximáciami.



Obr. 6 - Úvodné okno

Okno sa otvorí v strede obrazovky. Na zabezpečenie tohto cieľa slúž jednoduchý matematický výpočet. Poznáme rozmery daného okna a funkciou

```
get(0,'ScreenSize')
```

zistíme aktuálne rozlíšenie obrazovky, ktoré si zapíšeme do premennej. Nula v príkaze reprezentuje v hierarchii (obr. 1) *root*, čo je obrazovka počítača. ScreenSize ako už názov napovedá je vlastnosť, ktorej hodnota je rozlíšenie obrazovky. Okno potom vytvoríme

```
intro = figure('Position',position_intro,'Color',[240 240 240]/255);
```

A druhým príkazom modifikujeme nastavenia okna.

```
set(gcf,'MenuBar','none','name','Start','numbertitle','off')
```

Parametre príkazu figure a set sú popísané v tabuľke 1. Tlačidlá s obrázkami sa dajú vytvoriť nasledovne:

```
o1 =  uicontrol(gcf,...
        'Style','PushButton',...
        'Position',[(w-400)/2 200 400 130],...
        'CData',fo,...
        'Callback','close, order = 1, run GUI_prep');
```

Premenná `o1` je handle (číselný identifikátor) tlačidla. Ostatné vlastnosti sú popísané v tabuľke 2. V premennej `fo`, je uložený obrázok. Obrázok sa konvertuje na maticu typu *uint8* pomocou príkazu `imread('picture.jpg')`. V parametri *Callback* si môžeme všimnúť tri príkazy, ktoré sa vykonajú po stlačení tlačidla. Prvý z nich *close* zatvorí aktuálne okno, druhý príkaz je obyčajné priradenie a tretí spustí ďalšie okno.

3.2 Porovnanie aproximácií

Po výbere rádu, ktorý realizujeme kliknutím na tlačidlo s obrázkom (obsahuje vzorec a rád) sa nám nastaví do premennej `order`, požadované číslo (1 alebo 2) a otvorí sa nám ďalšie okno (obr. 7). Toto nové okno musí obsahovať polia, v ktorých zadáme vstupné údaje pre prenosy *Z, T, D*, ξ – vľavo hore. Okno obsahuje tlačidlo *Load Test Values*, ktoré načíta testovacie hodnoty. Hodnotu zadaných konštánt načítame pomocou nasledovného príkazu:

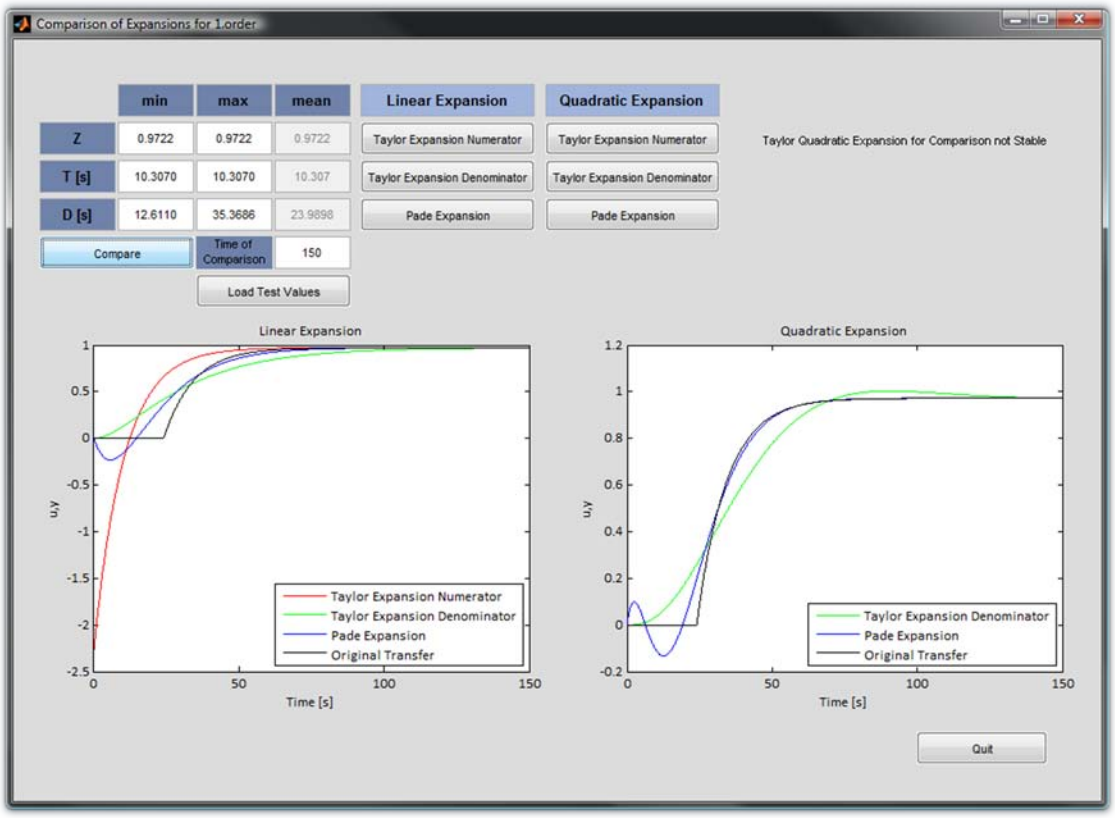
```
Tmin = str2double(char(get(edit(4),'String')));
```

V tomto prípade nám funkcia get vráti premennú typu *string*. Keďže potrebujeme číslo, tak požijeme funkcie `str2double()`. Táto funkcia však potrebuje na vstup premennú typu *char*, takže výstup z funkcie `get()` zmeníme na typ *char* a následne to zmeníme na *double*, čo je

číslo. Po zadaní potrebných údajov môžeme porovnávať aproximácie. Stlačením tlačidla *Compare* (Porovnaj) sa spustí m-file, v ktorom sú zadané jednotlivé prenosy.

Výsledkom týchto simulácií budú dva grafy – pre lineárne a kvadratické aproximácie. Program simuluje postupne najprv pôvodný prenos s dopravným oneskorením a následne odsimuluje fyzikálne realizovateľné aproximácie. Z tohto dôvodu sa však zvyšuje výpočtová náročnosť týchto simulácií, čo sa prejaví na čase kedy sa vykreslí graf. Pri aproximáciách druhého rádu je potrebný čas na ustálenie až 1000 sekúnd.

Po odsimulovaní aproximácií pre konkrétny užívateľom zvolený prenos (*Z, T, D*, ξ) si užívateľ zvolí aproximáciu a stlačí príslušné tlačidlo. Po zvolení aproximácie tlačidlom sa otvorí nové okno, v ktorom už navrhujeme regulátory a zároveň sa do premennej `approx` zapíše, číslo aby sme vedeli identifikovať, akú aproximáciu sme zvolili.



Obr. 7 - Porovnávanie Aproximácií

3.3 Výpočet regulátora

Nové okno, ktoré sa otvorí po výbere aproximácie, bude zobrazené taktiež do stredu obrazovky. Toto okno obsahuje:

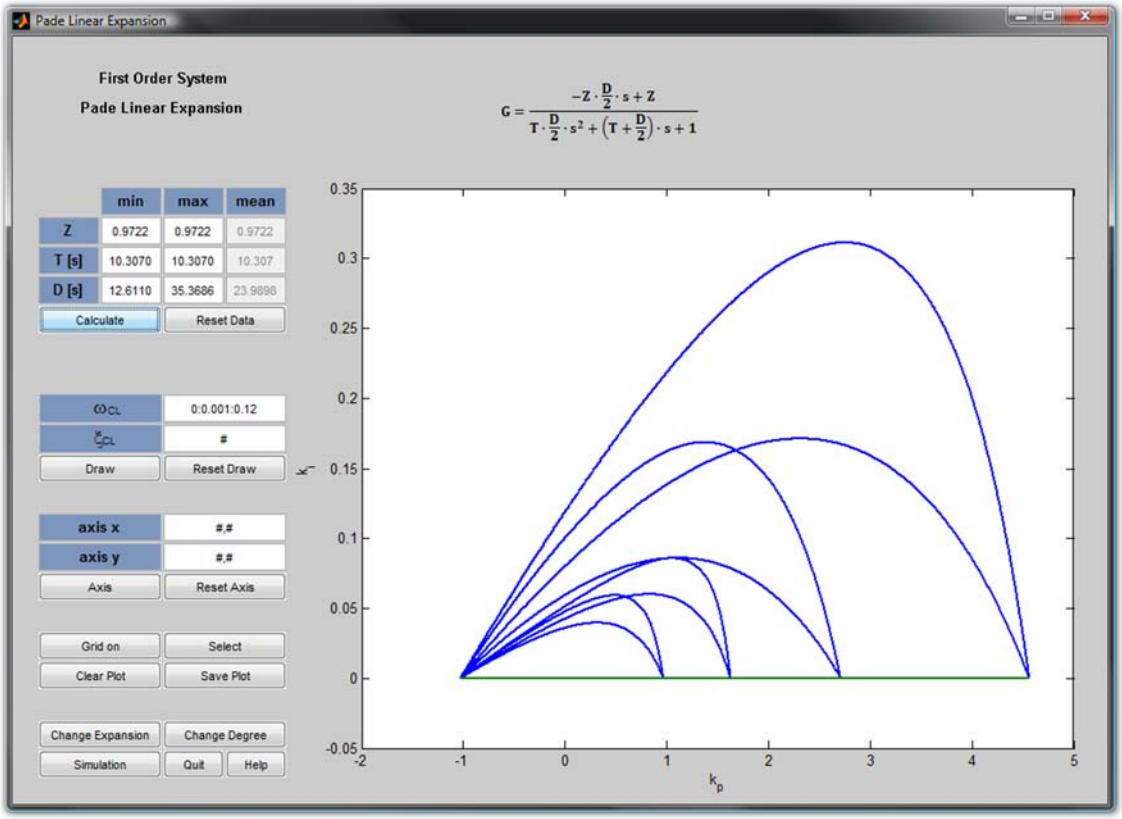
- Textové polia
  - Názov aproximácie
  - Textové polia na zadanie vstupných konštánt
  - Textové polia na zadanie konštánt ξ_{CL} *a* ω_{CL} (určenie regulátora)
  - Textové polia na zmenu osí - editovacie textové polia
- tlačidlá spúšťajúce výpočtové algoritmy
  - Tlačidlo *Calculate* – spustí sa algoritmus výpočtu stabilných oblastí
  - Tlačidlo *Draw* – vykreslenie krivky, ktorá reprezentuje regulátory
- Riadiace tlačidlá pre prácu s grafom
  - Tlačidlo *Select* – určenie konštánt regulátora
  - Tlačidlo *Grid* – zobrazenie mriežky v grafe
  - Tlačidlo *Simulate* – spustí simuláciu zvoleného regulátora
- Ostatné riadiace tlačidlá
  - Tlačidlo, ktoré vykoná zmenu osí
  - *Reset*, *Quit*, *Save Plot*, *Clear Plot*, *Change Expansion*, *Change Degree*
- Ostatné objekty
  - Graf – objekt *axes*
  - Vzorec prenosu – obrázok

3.3.1 Grafické znázornenie stabilných oblastí

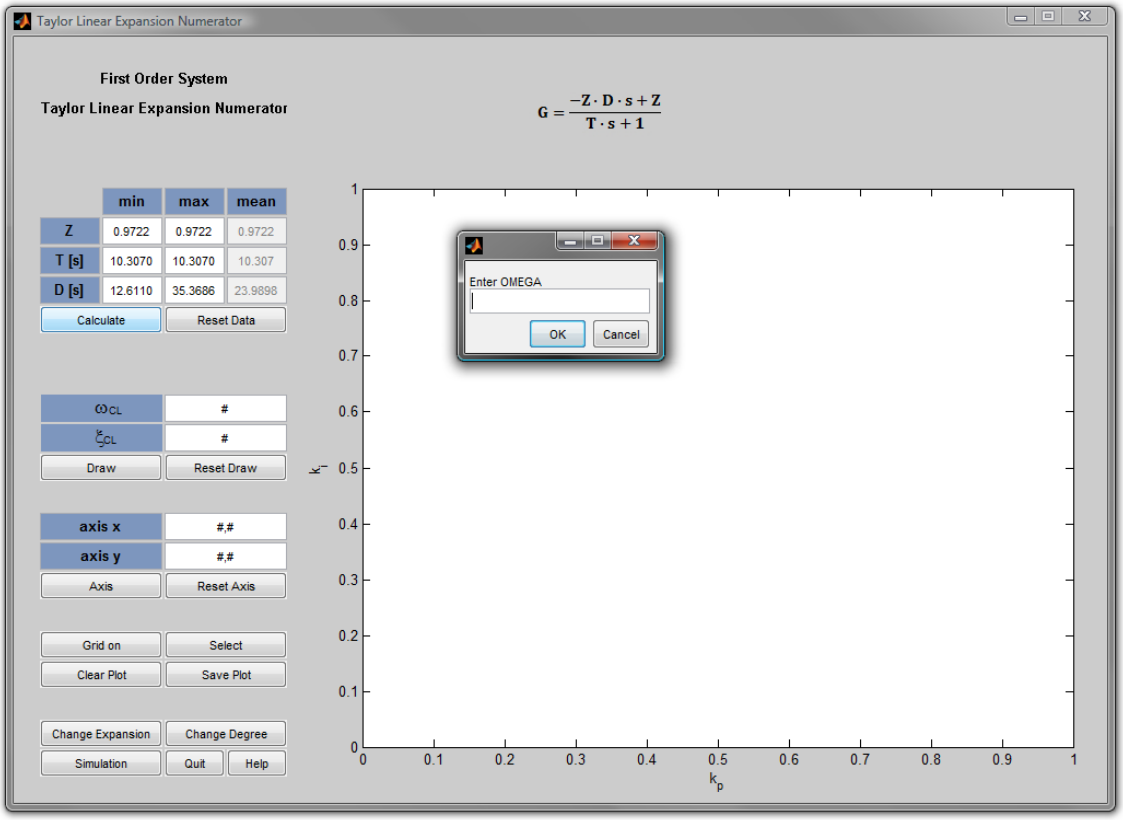
Okno, ktoré vidíme na obrázku číslo 8 je naprogramované v jednom m-file. Zobrazovanie názvov a vzorcov aproximácií sa zabezpečuje pomocou podmienok (*if, elseif, else*) a premennej `approx`.

V ľavo hore vidíme informáciu o ráde prenosu, ktorý sme aproximovali. Nasleduje názov aproximácie. Vstupné údaje (*Z, T, D*, ξ) sme už zadávali pri porovnávaní aproximácií, takže ich programovo skopírujeme aby ich nebolo potrebné zadať znova. Je ich možné samozrejme meniť. Po stlačení tlačidla *Calculate* sa spustí výpočtový algoritmus a v grafe - objekt axes – sa vykreslia stabilné oblasti.

V našom prípade (Padého Lineárna Aproximácia prenosu prvého rádu s dopravným oneskorením) sa parameter ω_{CL} vypočítal sám. Pri niektorých aproximáciách, napr. Taylorova aproximácia do čitateľa je potrebné omega zadať. Ako vstupný parameter sa zadá horná hranica tohto vektora a algoritmus si rozdelí interval 0 až zložené číslo na definovaný počet prvkov (obr. 9).



Obr. 8 – Vykreslenie Stabilných Oblastí



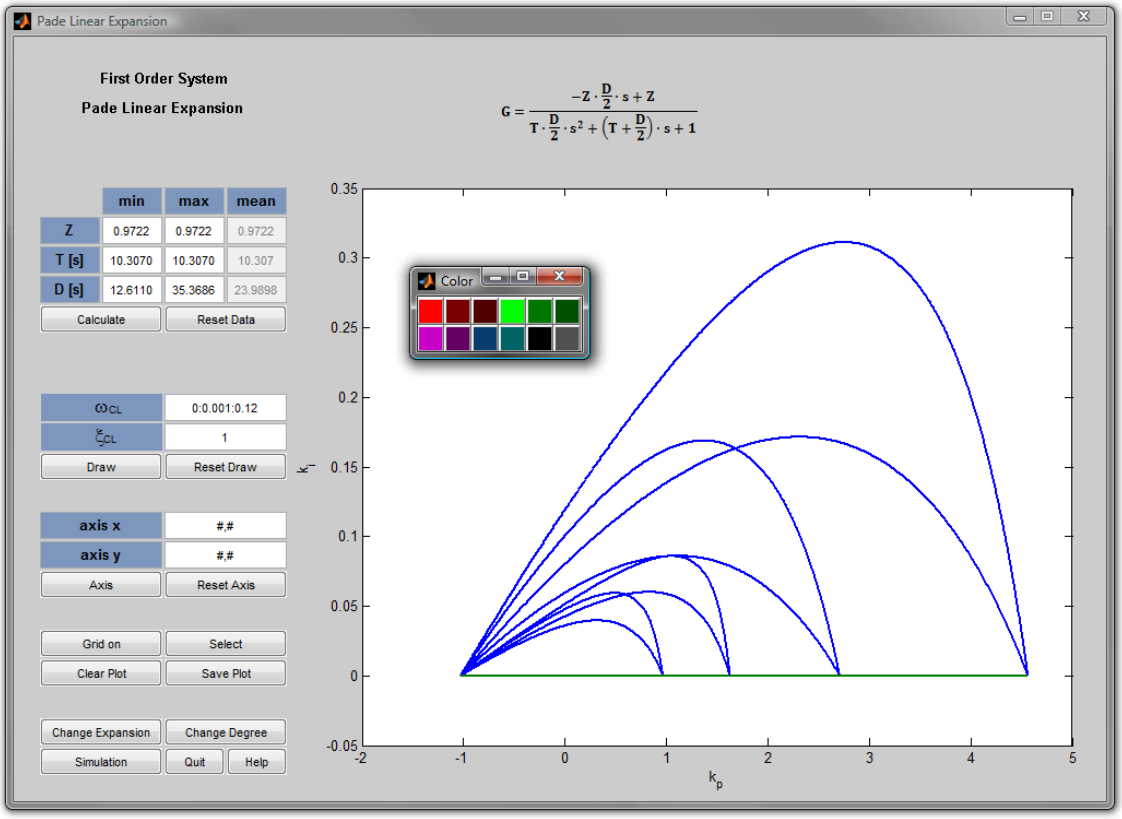
Obr. 9 - vloženie parametra ω

V skrátenom vektorovom tvare ho zobrazujeme v danom editovacom textovom poli. V tomto prípade je treba povedať, že zobraziť vektor v tvare `[start:step:end]` nepatrí medzi elementárne operácie v MATLAB-e. Parametru String síce môžeme priradiť na zobrazenie vektor, ale v našom prípade to nie je vhodné. Vektor ω_{CL} obsahuje zväčša tisíc prvkov, čo je veľmi veľa na zobrazenie v editovacom poli. Tento vektor však vieme zapísať vo vyššie uvedenom tvare. Prvý a posledný prvok vektora vieme získať jednoducho – `start = variable(1)` resp. `end = variable(end)`. Určiť krok je jednoduché – rozdiel akýchkoľvek dvoch prvkov – `step = variable(2)-variable(1)`. Teraz potrebujeme jednotlivé čísla zmeniť z typu `double` na `string`. Použijeme príkaz `mat2str` [6]. Potom pospájame 3 premenné typu `string` a môžeme ich vypísať do textového poľa. Prv než sa použije príkaz `mat2str` je vhodné zaokrúhliť čísla na požadovaný počet desatinných miest. Na túto operáciu použijeme príkaz `ceil()`. Tento príkaz však zaokrúhľuje na celé čísla, takže ak potrebujeme zaokrúhliť na tisíciny, tak najprv číslo vynásobíme 1000. Potom použijeme príkaz `ceil()` a výsledné číslo vydelíme 1000.

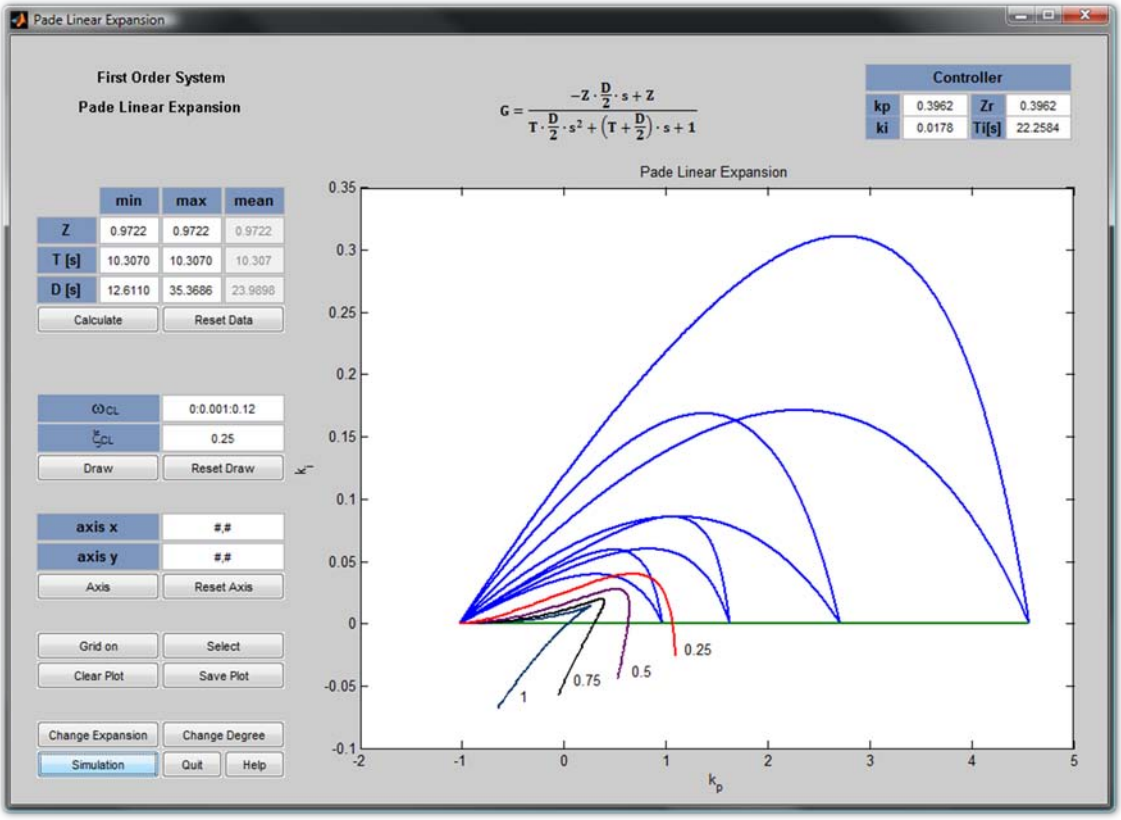
3.3.2 Grafické znázornenie PI regulátorov

PI regulátorom prislúcha krivka, ktorú vykreslíme na základe zvolených konštánt ξ_{CL} a ω_{CL}. Ako sme už spomenuli, vektor ω_{CL} sa väčšinou vypočíta programom. Konštantu ξ_{CL} volí užívateľ. Po stlačení tlačidla *Draw*, sa otvorí nové okno s voľbou farby. Na výber je dvanásť kontrastných farieb (obr. 10).

Keďže je možné do grafu dokresľovať regulátory pre rôzne ξ_{CL} a ω_{CL}, tak po vykreslení každej čiary sa spustí príkaz `gtext(sprintf('%g', xi))`, ktorým ku každej čiare dopíšeme pre aké ξ_{CL} sme PI regulátor navrhli (obr. 11).



Obr. 10 - Zvolenie farby



Obr. 11 - Návrh PI regulátorov

3.3.3 Nástroje a funkčnosť

Pre lepšiu orientáciu v grafe je potrebná mriežka – tlačidlo *Grid* – a zmena osí. Osi sa menia nasledovným príkazom, po stlačení tlačidla *Axis*:

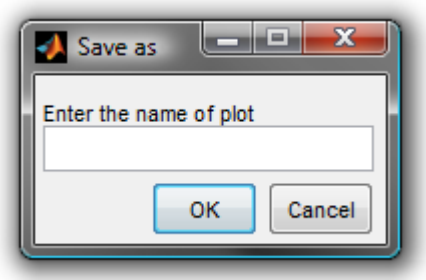
```
set(axes_input,'XLim',xx,'Ylim',yy)
```

kde `axes_input` predstavuje handle grafu, premenné `xx` mierka na x-ovej osi a premenná `yy` mierka na y-ovej osi. Premenné `xx` a `yy` sú v tvare vektora `[xmin xmax]` resp. `[ymin ymax]`. Vedľa tlačidla Axis, je tlačidlo Reset Axis, ktorým vrátime osi do pôvodného stavu. Na to slúži nasledovný príkaz:

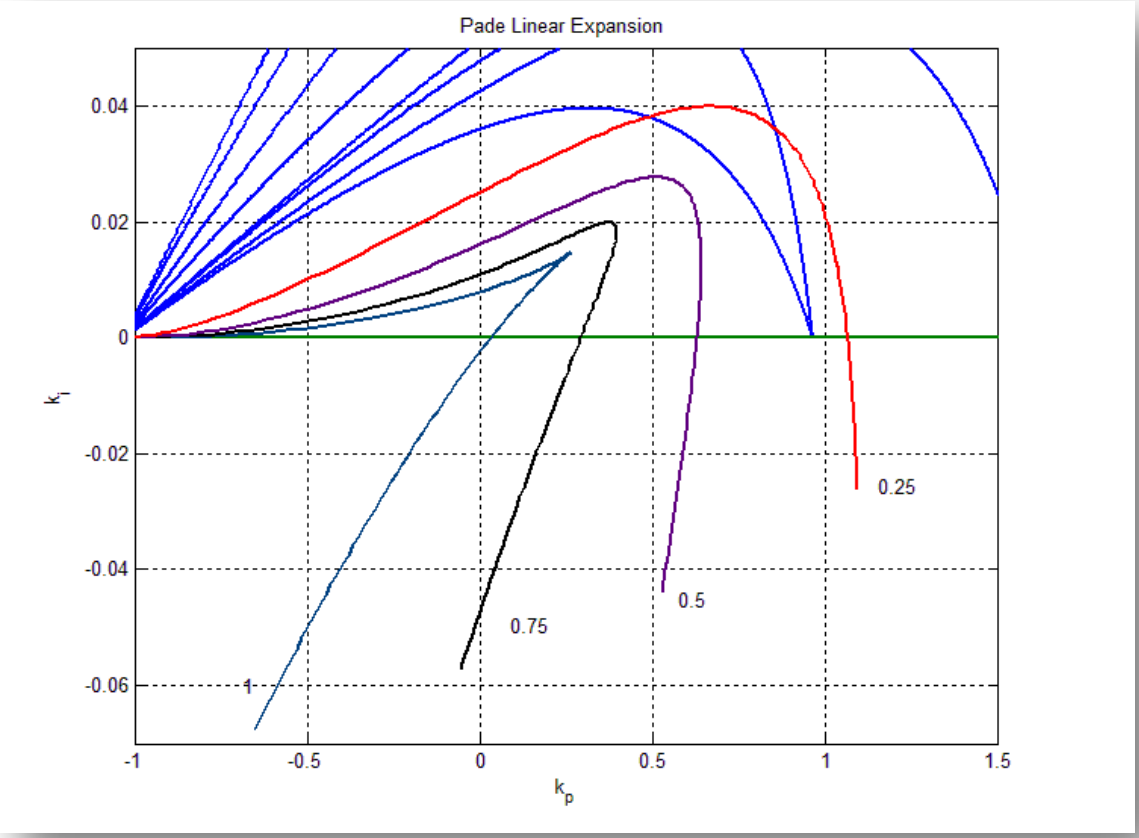
```
set(axes_input,'XLimMode','auto','YlimMode','auto')
```

Graf je možné uložiť ako .fig súbor (obr. 11), s ktorým môže užívateľ ďalej pracovať, napríklad uložiť graf ako .jpeg (obr. 12). Na názov sa program opýta, pričom príponu nie je potrebné písať. Program ju sám doplní. Pokiaľ sa užívateľ rozhodne, že potrebuje nanovo vykresliť grafy, tak mu na to slúžia tlačidlá ako sú *Reset Data*, *Reset Draw* a *Clear Plot*. Prvé tlačidlo vymaže vložené dáta a vymaže aktuálny graf. Tlačidlo s názvom *Reset Draw* vymaže všetky vygene-rované regulátory. *Clear Plot* vymaže len graf.

Na výber regulátora slúži už spomenuté tlačidlo *Select*. Za tlačidlom *Select* sa skrýva funkcia `ginput()`, ktorej výstup sú súradnice bodu v grafe, na ktorý klikneme. Táto funkcia má jeden vstup, bez ktorého by nepracovala správne. Ide o počet kliknutí v grafe. Pre tlačidlo *Select* je to nastavené na 1, čo znamená, že iba jeden regulátor môžeme určiť. Ak *Select* použijeme znova, tak aktuálny zvolený regulátor sa prepíše za novozvolený. tieto súradne sú priamo parametrami regulátora, ktoré sa zobrazia v pravom hornom rohu pod nápisom „Controller“. Je tu uvedený aj prepočet z k_i, k_p, čo zodpovedá konštantám P, I v Simulinkovom modeli PID bloku na konštanty *Zr* a *Ti*. Prepočet uvádzame pretože konštanty *Zr* a *Ti* vieme rýchlejšie porovnať s prenosom.

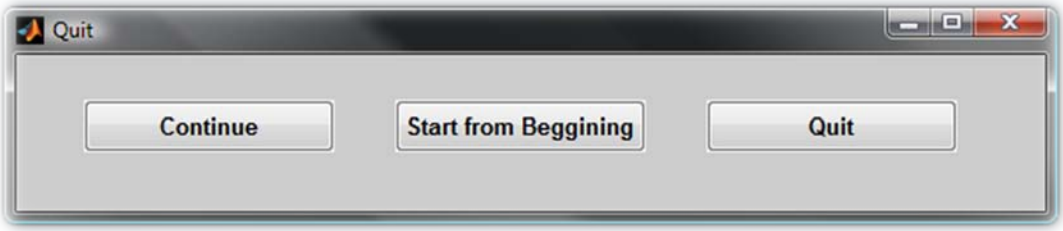


Obr. 11 - Uloženie grafu



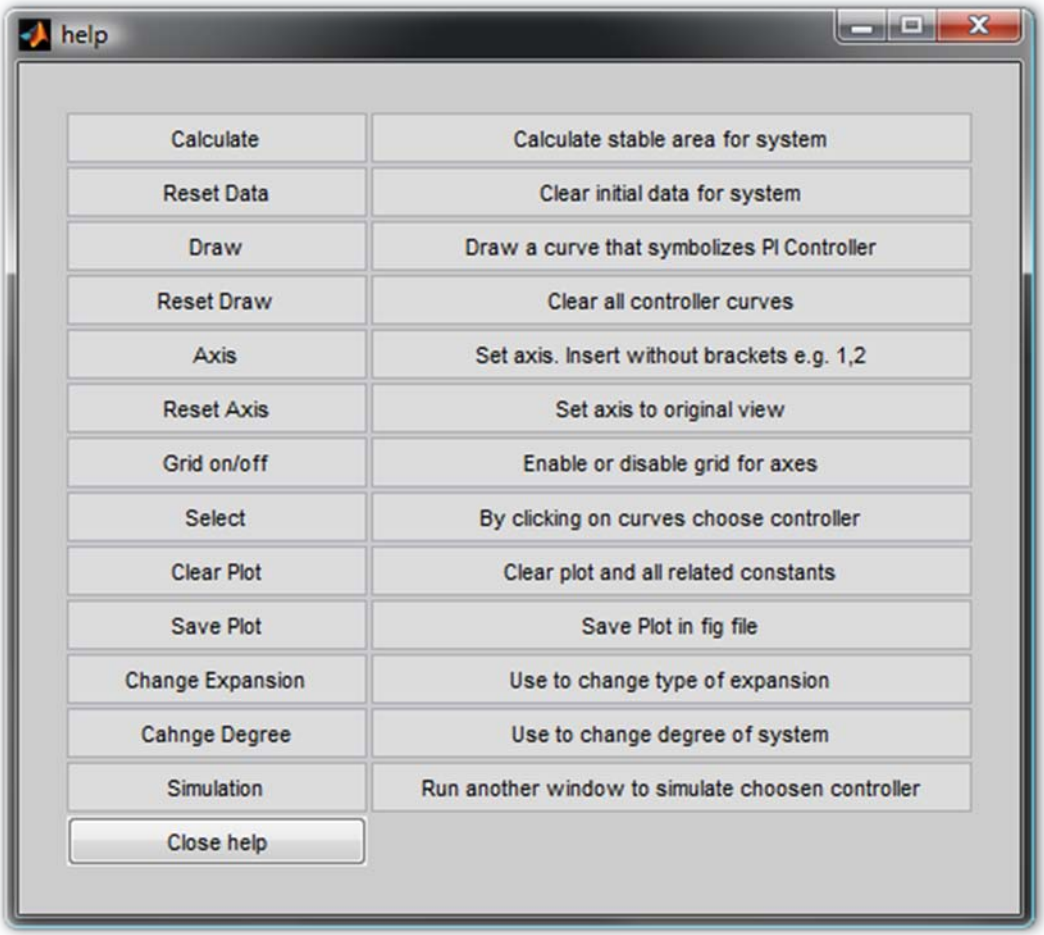
Obr. 12 - Graf uložený ako JPEG

Okno ďalej obsahuje tlačidlá *Change Degree* a *Change Expansion*. Prvé z nich nás vráti na úplne prvé okno v ktorom sme si volili rád systému s dopravným oneskorením. Druhé z nich nás vráti na obrazovku, kde sme porovnávali aproximácie. Ako posledné tlačidlo, ktoré sa v tomto okne nachádza je *Quit*. Stlačením tohto tlačidla sa spustí ďalšie okno s troma tlačidlami (obr. 13). Prvé – *Continue* – je tam z toho dôvodu, že ak by sa užívateľ pomýlil, aby náhodou nevypol celý program. Druhé – *Start form Beggining* – vráti program na začiatok, čiže na výber systému s prvým alebo druhým rádom. A posledné – *Quit* – vypne program a vymaže workspace.



Obr. 13 - Ukážka Quit

Keďže popis tlačidiel niekedy nemusí byť *Zro*zumiteľný pre užívateľa, tak tlačidlom *Help* sa nám otvorí okno, v ktorom sú funkcie jednotlivých tlačidiel vysvetlené (obr. 14).

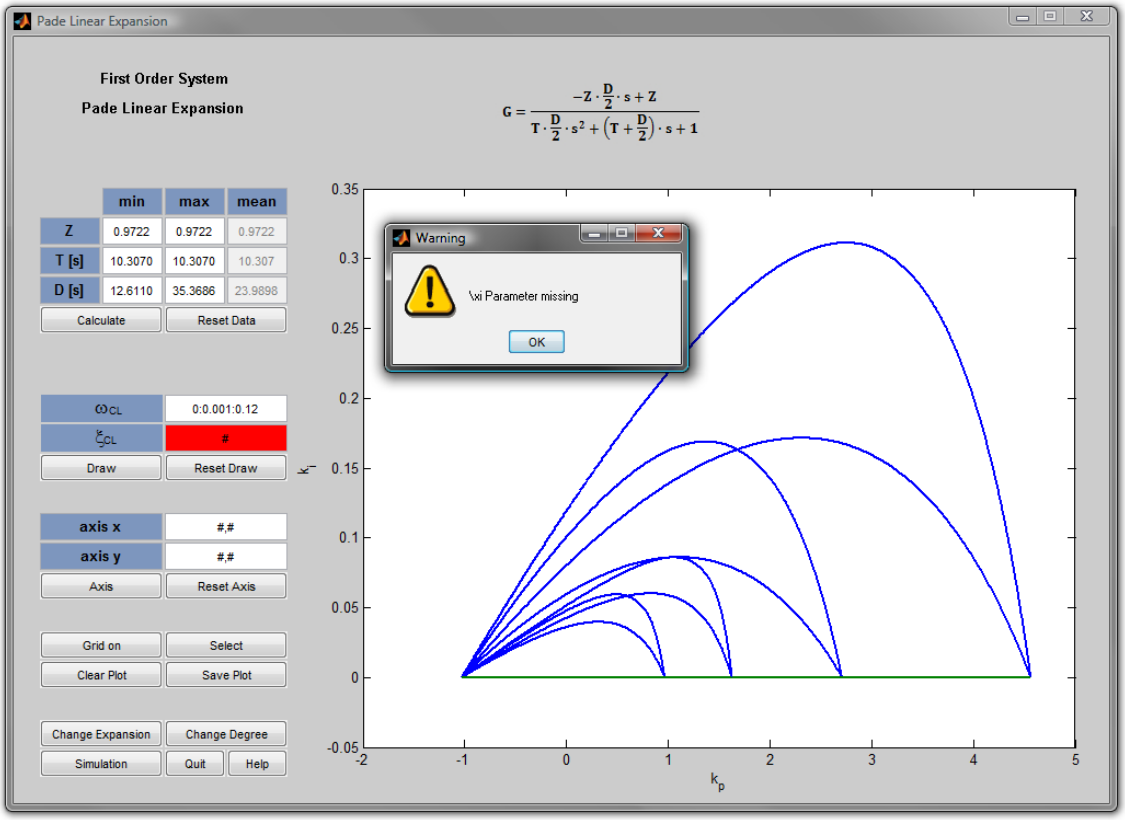


Obr. 14 - Pomocník

Tento program (GUI) obsahuje niekoľko prvkov, ktoré upozornia užívateľa, že by mohlo dôjsť k chybe pri výpočte. Špeciálne ide o vykresľovanie kriviek, ktoré reprezentujú regulátory. Je *Zr*ejmé, že regulátor nevieme navrhnúť pokiaľ nie sú zadané hodnoty ξ_{CL} a ω_{CL}. Parameter ω_{CL} sa buď vypočíta sám, alebo ho treba špecifikovať už pri výpočte stabilných oblastí. Avšak parameter ξ_{CL} musí užívateľ zadať. Ak ho nezadá, a napriek tomu stlačí *Draw*, tak sa ukáže editovacie pole vysvieti na červeno (obr. 15). Zobrazí sa okno, v ktorom program upozorní užívateľa, že chýba daný parameter. Toto je ošetrené nasledovným príkazom:

```
warndlg('\xi Parameter missing','Warning')
```

Príkaz `warndlg` patrí medzi vstavané funkcie GUI.



Obr. 15 - Upozornenie

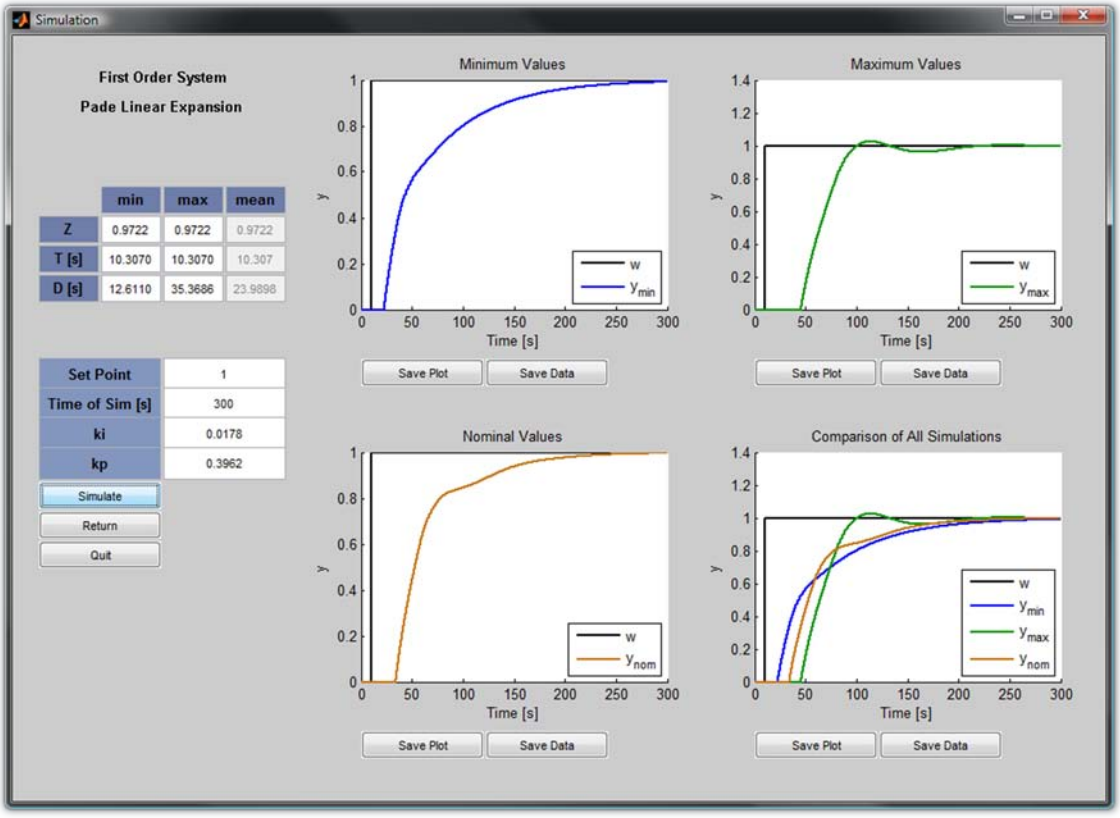
3.4 Simulácie

Číselné parametre regulátora, či už sú v tvare *Zr* a ti, alebo v tvare P a I nám sami o sebe nič nehovoria o kvalite riadenia. Okno, do ktorého sa dostaneme stlačením tlačidla *Simulation*, v sebe obsahuje objekty axes, v ktorých sa vykreslia simulácie so zvoleným regulátorom. Vstupné údaje pre simuláciu jednoduchého uzavretého regulačného obvodu sú: *Z, T, D*, ξ ak ide o prenos druhého rádu, čas simulácie a žiadaná hodnota. Parametre regulátora sa skopírujú z predošlého okna. Všetky hodnoty možno meniť podľa potreby. Čas simulácie, ktorý je prednastavený v mnohých prípadoch, môže byť príliš veľký, alebo príliš malý. Preto je niekedy potrebné niekoľkokrát zopakovať simuláciu, aby sme sa našli vhodný regulátor. Taktiež je možné, že bude potrebné aj zopakovať návrh regulátora.

3.4.1 Porovnanie jednotlivých simulácií

Okno, ktoré sa nám otvorí po stlačení tlačidla *Simulate* bude obsahovať štyri objekty *Axes*, v ktorý sa postupne budú zobrazovať výsledky zo simulácie pre minimálne, maximálne a nominálne hodnoty intervalov pre *Z, T, D.* Posledný grafu bude spojením predchádzajúcich troch grafov, aby sme mohli porovnať jednotlivé výsledky. (obr. 16).

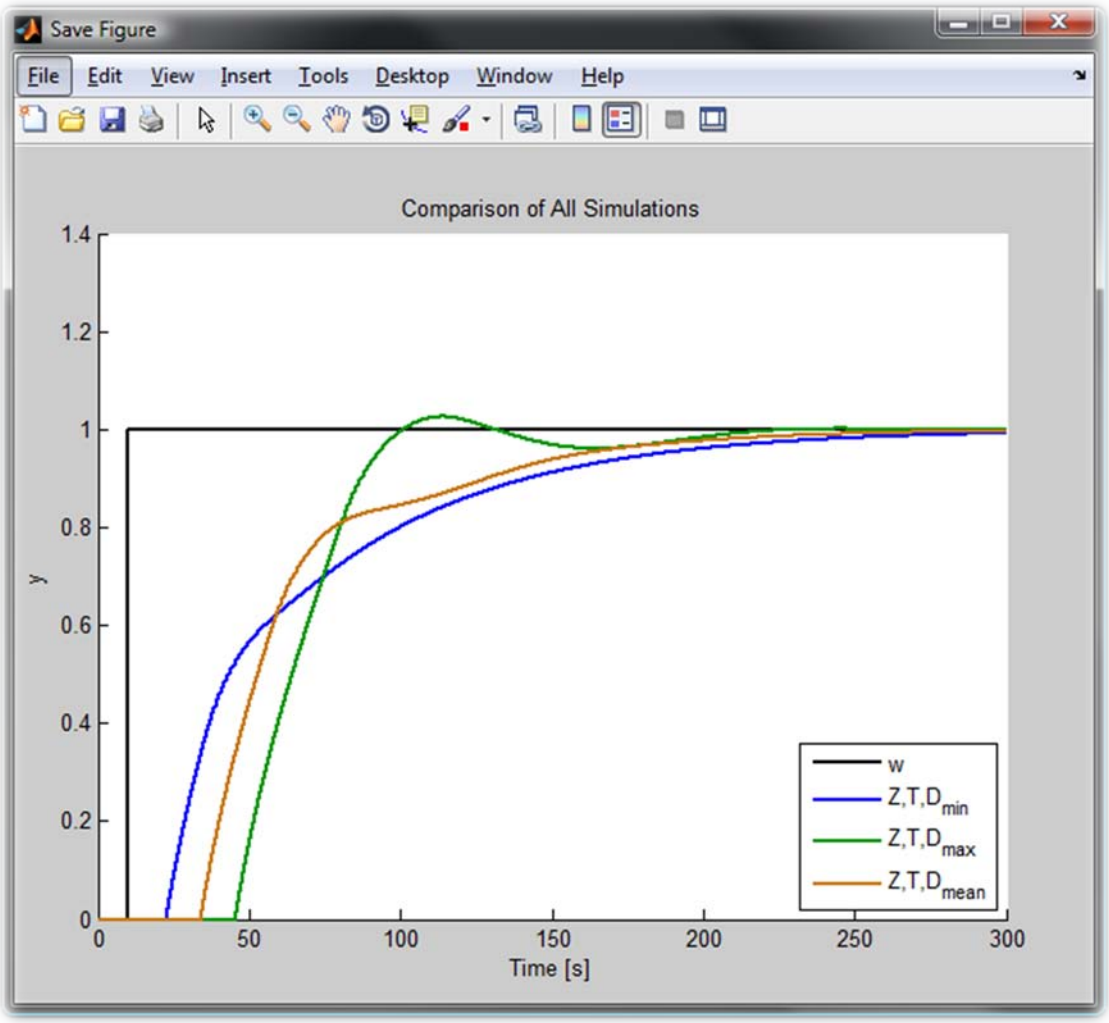
Vľavo hore máme textové pole, ktoré obsahuje informáciu, ktorý rád sme aproximovali. Pod týmto textovým poľom je ďalšie textové pole, v ktorom je napísané akú aproximáciu dopravného oneskorenia sme použili. Potom sa tam nachádzajú editovacie polia, v ktorých zadávame základné konštanty prenosu a editovacie polia ktoré sa týkajú simulácií. Tlačidlo *Simulate* nám spustí skript, cez ktorý sa spustí Simulinkový model a vykreslia sa grafy. Pod tlačidlom *Slimulate* vidíme ešte dve riadiace tlačidlá: *Return* (vráti program na návrh regulátorov) a *Quit*, ktorý vypne celý program.



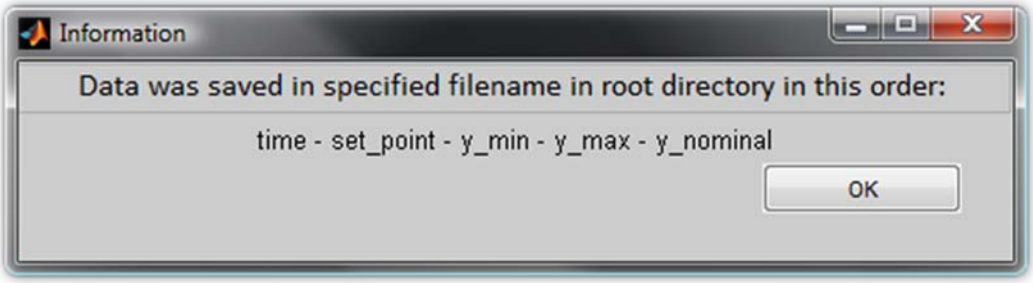
Obr. 16 - Okno so simuláciami

3.4.2 Nástroje a funkčnosť

Vkladanie mriežky, poprípade zmena osí v tomto prípade nemá zmysel, keďže potrebujeme len približne porovnať navrhnutý regulátor. Program však ponúka možnosť uložiť jednotlivé grafy a zároveň aj matice, v ktorých sú zapísané údaje zo Simulinku. Grafy si ukladajú v súbore .fig, na ktorého názov sa program opýta (obr. 17). Údaje so Simulinku sa ukladajú ako .txt súbor, na ktorého názov sa tiež program opýta. Doplnok k ukladaniu údajov, je ďaľšie okno, v ktorom sa zobrazia názvy stĺpcov (obr. 18).



Obr. 17 - Uloženie grafu



Obr. 18 - Uloženie údajov

4 Práca s grafom

4.1 Vytváranie grafu

Základným príkazom na vykreslenie plošného grafu je `plot(x,y)`, kde premenná *x* reprezentuje x-ove súradnice bodov a premenná *y* reprezentuje y-ové súradnice bodov. Ak napíšeme tento príkaz do príkazového riadku v MATLAB-e, tak sa udejú dve veci. Ako prvé sa vytvorí okno figure a potom sa vytvorí podriadený objekt axes. Pri vytváraní interaktívneho GUI však okno figure vytvoríme najprv a doplníme grafické prvky typu uicontrol. Ak chceme vykresliť graf, musíme vytvoriť aj objekt axes. Môžeme síce použiť priamo príkaz plot, ale MATLAB by objekt axes vytvoril v celom figure a prekryl by prvky typu uicontrol. Aby sme sa vyhli tomuto problému, tak objekt axes vytvoríme príkazom prv než pôjdeme vykresľovať graf a nastavíme mu parametre, aké požadujeme. Osvedčené vytváranie jednoznačného objektu axes tvorí nasledovné poradie príkazov:

```
axes;
axes_input = gca;
set(axes_input,'Units','pixels')
set(axes_input,'position',position)
hold(gca)
```

Príkazom axes vytvoríme objekt v aktuálnom figure. Príkazom `gca` – get current axes, zapíšeme handle objektu do premennej, aby sme jednoznačne zadefinovali do ktorého objektu axes budeme neskôr vykresľovať grafy pomocou príkazu `plot()`. Príkazmi `set` nastavíme požadované vlastnosti objektu, ako je napríklad pozícia, veľkosť a jednotky, v ktorých sa budú počítať súradnice objektu. Ak potrebujeme presne zistiť, aké všetky možnosti môžeme modifikovať pre objekt axes, tak stačí napísať do príkazového riadku `set(gca)` a vypíšu sa nám podobne ako pre grafické prvky typu uicontrol všetky nastaviteľné položky.

Veľmi dôležitým parametrom ktorý musíme nastaviť je *Nextplot*. Táto vlastnosť určuje, čo sa stane, ak chceme doplniť ďalšiu krivku do grafu, alebo graf prepísať novým grafom. Je niekoľko možností, čo nastaviť do tohto parametra. Prednastavená hodnota je *replace. Replace* znamená prepíš. My však potrebujeme krivky do grafu dokresliť a nie graf prepísať. Ako hodnotu tejto vlastnosti nastavíme *Add*, čo plní funkciu *hold on*. To znamená, že pokiaľ graf nevymažeme príkazom `cla` – clear axes, každým ďalším príkazom `plot()` sa doplnia krivky bez zmeny pôvodných.

4.2 Kopírovanie a uloženie grafu

Niekoľkokrát spomenutá operácia ukladania grafu je náročnejšou operáciou z pohľadu počtu príkazov. Medzi základné požiadavky patrilo uložiť graf ako súbor .fig, aby ho užívateľ mohol neskôr spracovať podľa svojich vlastných potrieb – napr. uložiť ako JPEG súbor. Ako súbor .fig sa však dá uložiť iba okno figure. Takže ako prvé musíme vytvoriť nové okno, teda nový figure. Potom do neho treba skopírovať požadovaný objekt axes a potom to môžeme uložiť [7]. Nasledovný súbor príkazov sa nachádza v m-file, ktorý sa spúšťa tlačidlom *Save Plot*.

```
position_new = position_g;

if screenxy(3)<=1750
    position_new(1) = 10;
else position_new(1) = (screenxy(3)-1000-750)/2;
end

set(gcf,'Position',position_new)

position_copy(1) = (screenxy(3)-1000-750)/2+1030;
position_copy(2) = (screenxy(4)-550)/2;
position_copy(3) = 750;
position_copy(4) = 550;

if (position_copy(1)<=(screenxy(3)-760)) &&
(position_copy(1)>=274)
    position_copy(1) = (screenxy(3)-1000-750)/2+1030;
else position_copy(1) = screenxy(3)-760;
end

fig = figure;
set(fig,'Position',position_copy,'resize','off');
```

Vyššie uvedený súbor príkazov nám vytvorí nové okno a nastaví pozíciu na obrazovke. Výpočty sú spravené tak, aby sa tieto dve okná (pôvodné figure, kde sa navrhujú regulátory a nové okno) zobrazili vedľa seba pokiaľ to veľkosť obrazovky dovoľuje. Ak nie, tak sa pozícia okien prispôsobí šírke obrazovky tak aby bolo čo najviac vidieť z obidvoch okien.

```
ff = copyobj(axes_input,fig);
axes_units = get(gca,'Units');
axes_pos = get(gca,'Position');
set(ff,'Units',axes_units);
position_axes(1) = (750-axes_pos(3)+80)/2;
position_axes(2) = (550-axes_pos(4)+80)/2;
position_axes(3) = axes_pos(3)-80;
position_axes(4) = axes_pos(4)-50;
```

Príkaz `copyobj()` skopíruje všetky nastavenia objektu axes do objektu fig, pričom nový objekt axes bude mať handle uložený v premennej `ff`. Keďže nové okno je už prispôsobené veľkosti pôvodného objektu axes a neobsahuje žiadne grafické prvky, tak ako prvé je potrebné zmeniť jeho pozíciu voči novému oknu. Na to slúži príkaz

```
set(ff,'Position',position_axes);
```

kde pozícia v premennej `position_axes` sa vypočítala predchádzajúcimi príkazmi. Teraz sme zabezpečili, aby v novom okne bol zobrazený objekt axes tak ako bol v pôvodnom okne, z ktorého sme ho kopírovali. Jediné čo ostáva, je potrebné ho uložiť. Na to slúži nasledovný súbor príkazov.

```
name = inputdlg('Enter the name of plot','Save as',1);
name = char(name)';
namefig = '.fig';
namefig = char(namefig)';
namesave = char(name,namefig)';
saveas(ff,namesave);
close(fig)
```

Nesmieme však zabudnúť na to, že sme posunuli pôvodné interaktívne okno s návrhom regulátorov, ktoré musíme vrátiť naspäť na stred obrazovky. To ošetríme jednoduchým príkazom:

```
set(gcf,'Position',position_g);
```

kde v premennej `position_g` je uložená pôvodná pozícia okna.

Záver

Ako prvú otázku, ktorú som si položil pri tvorbe grafiky bola či ju budem tvoriť cez GUIDE, alebo pomocou uicontrol. Keďže som mal skúsenosť s GUIDE a naskytli sa tam mnohé problémy, napríklad s workspace a funkciami, tak som zvolil náročnejší, avšak prehľadnejší a profesionálnejší prístup. Uicontrol je príkaz, ktorým priamo upravujem vlastnosti a parametre grafických prvkov. V kombinácií s príkazmi get a set som upravoval a nastavoval grafické prvky tak, aby sa dosiahol želaný výsledok - user-friendly.

Celý program pozostáva zo štyroch hlavných okien, ktoré obsahujú množstvo textových polí, riadiacich tlačidiel a objektov *axes*. Textové polia slúžia ako vstupy do programu a tlačidlá plnia funkciu spúšťania rôznych výpočtových algoritmov a pomocných *m-file*. Dbal som na to, aby okná GUI boli prehľadné, aby farby textu a pozadí boli jasne čitateľné. Pre lepšiu orientáciu v programe som napísal menší *help*, ktorý sa spúšťa z hlavného okna, v ktorom sa navrhujú regulátory.

Medzi náročnejšie úlohy tejto práce patrilo jednoznačne ukladanie grafu, pre ktoré bolo treba vytvoriť špeciálny skript na kopírovanie objektu axes, následné vykreslenie a až potom to bolo možné uložiť. O veľké zjednodušenie a minimalizovanie kódu sa postaral for-cyklus, ktorým som vytvoril takmer všetky textové a editovacie polia.

Úlohou tejto práce bolo nielen zvládnuť prácu s uicontrol, ale taktiež vytvoriť grafickú nadstavbu programu, ktorý sa zaoberá návrhom robustných PI regulátorov pre systémy prvého a druhého rádu s dopravným oneskorením. Vhodnými doplnkami tohto programu sú upozornenia. Zobrazia sa ak užívateľ nezadá niektoré konštanty, a chcel by pokračovať v práci s programom, čo by viedlo k „červeným oznamom“ v príkazovom riadku.

Literatúra

[1] **Klaučo, Martin.** *Tvorba grafického rozhrania (GUI) pre návrh robustných PI regulátorov.* Bratislava, 2009.

[2] **Vaneková, Katarína.** *Návrh robustných regulátorov pre riadenie chemicko-technologických procesov.* Bratislava, 2008.

[3] **Zaplatílek, Karel a Doňar, Bohuslav.** *MATLAB - tvorba uživatelských aplikací.* Praha: Technická Literatura BEN, 2004.

[4] **Mathworks, Inc.** Documentation MATLAB. *Handle Graphic.* [Online] 27. 10 2009. http://www.mathworks.com/access/helpdesk/help/techdoc/.

[5] **Bakošová, Monika a Fikar, Miroslav.** *Riadenie Procesov.* Bratislava: Slovenská Technická Univerzita v Bratislave, 2008. ISBN: 978-80-227-2841-6.

[6] **Mathworks, Inc.** *Product Help MATLAB.* 2009.

[7] **Quach, Quan a Slazas, Rob.** Blinkdagger. *Matlab GUI Tutorial.* [Online] 06. 11 2009. http://blinkdagger.com/matlab/matlab-gui-tutorial-saving-plots-within-gui/.

Prílohy

Príloha A: CD médium s prácou a programom

Príloha B: Ukážky zdrojových kódov

Úvodné okno

```
%---start-script-----------------------------------------------------------%

%---add-path-in-order-to-use-othe-folders----------------------------------%

addpath('graphic/','graphic/pictures/','scripts/')

%---constants-&-parameters-------------------------------------------------%

screenxy = get(0,'ScreenSize');
w = 500;
h = 400;
position_intro(3) = w;
position_intro(4) = h;
position_intro(1) = (screenxy(3)-w)/2;
position_intro(2) = (screenxy(4)-h)/2;

fo = imread('first_order_2.png');
so = imread('second_order_2.png');

%---creating-figure---------------------------------------------------------%

intro = figure('Position',position_intro,'Color',[240 240 240]/255);

set(gcf,'MenuBar','none','name','Start','numbertitle','off','resize','off')

%---creating-objects--------------------------------------------------------%

o1 =  uicontrol(gcf,...
        'Style','Pushbutton',...
        'Position',[(w-400)/2 200 400 130],...
        'CData',fo,...
        'Callback','close, order = 1, run GUI_prep');

o2 =  uicontrol(gcf,...
        'Style','Pushbutton',...
        'Position',[(w-400)/2 60 400 130],...
        'CData',so,...
        'Callback','close, order = 2, run GUI_prep');

start_text = uicontrol(gcf,...
        'Style','Text',...
        'BackgroundColor',[205 224 247]/255,...
        'Position',[(w-400)/2 340 400 20],...
        'FontSize',12,...
        'FontWeight','bold',...
        'String','Select Order of System');

%---end-of-script-----------------------------------------------------------%
```

Simulácie uzavretých regulačných obvodov

```
%---simsim-script----------------------------------------------------------%
time_s = str2double(char(get(edit_ss(2),'String')));
set_p = str2double(char(get(edit_ss(1),'String')));
```

```
xgsim = str2double(char(get(edit_ss(3),'String')));
ygsim = str2double(char(get(edit_ss(4),'String')));

cla(axes1)
cla(axes2)
cla(axes3)
cla(axes4)

if approx<7
    sim('schema1',[0 time_s*1])
else
    sim('schema2',[0 time_s*1])
end


set(axes1,'nextplot','add')
title(axes1,'Minimum Values')
plot(axes1,data_simul(:,1),data_simul(:,2),'color',[0 0 0],'LineWidth',2)
plot(axes1,data_simul(:,1),data_simul(:,3),'color',[0 0
255]/255,'LineWidth',2)
xlabel(axes1,'time [s]')
ylabel(axes1,'y')
legend(axes1,'w','Z,T,D_{min}',4)

set(axes2,'nextplot','add')
title(axes2,'Maximum Values')
plot(axes2,data_simul(:,1),data_simul(:,2),'color',[0 0 0],'LineWidth',2)
plot(axes2,data_simul(:,1),data_simul(:,4),'color',[0 150
0]/255,'LineWidth',2)
xlabel(axes2,'time [s]')
ylabel(axes2,'y')
legend(axes2,'w','Z,T,D_{max}',4)


set(axes3,'nextplot','add')
title(axes3,'Mean Values')
plot(axes3,data_simul(:,1),data_simul(:,2),'color',[0 0 0],'LineWidth',2)
plot(axes3,data_simul(:,1),data_simul(:,5),'color',[200 120
20]/255,'LineWidth',2)
xlabel(axes3,'time [s]')
ylabel(axes3,'y')
legend(axes3,'w','Z,T,D_{mean}',4)


set(axes4,'nextplot','add')
title(axes4,'Comparison of All Simulations')
plot(axes4,data_simul(:,1),data_simul(:,2),'color',[0 0 0],'LineWidth',2)
plot(axes4,data_simul(:,1),data_simul(:,3),'color',[0 0
255]/255,'LineWidth',2)
plot(axes4,data_simul(:,1),data_simul(:,4),'color',[0 150
0]/255,'LineWidth',2)
plot(axes4,data_simul(:,1),data_simul(:,5),'color',[200 120
20]/255,'LineWidth',2)
xlabel(axes4,'time [s]')
ylabel(axes4,'y')
legend(axes4,'w','Z,T,D_{min}','Z,T,D_{max}','Z,T,D_{mean}',4)

%---end-of-script----------------------------------------------------------%
```

Okno Quit

```
%---clear-all-data-script-------------------------------------------------%

screenxy = get(0,'ScreenSize');
w = 595;
h = 90;
position_clear(3) = w;
position_clear(4) = h;
```

```
position_clear(1) = (screenxy(3)-w)/2;
position_clear(2) = (screenxy(4)-h)/2;

%---creating-figure-------------------------------------------------------%

intro = figure('Position',position_clear,'Color',[240 100 20]/255);

set(gcf,'MenuBar','none','name','Quit','numbertitle','off')

%---creating-objects------------------------------------------------------%

clear_continue =  uicontrol(gcf,...
        'Style','Pushbutton',...
        'Position',[40 35 145 30],...
        'String','Continue',...
        'FontSize',10,...
        'FontWeight','bold',...
        'Callback','close');

clear_stop =  uicontrol(gcf,...
        'Style','Pushbutton',...
        'Position',[40+140+40 35 145 30],...
        'String','Start from Beggining',...
        'FontSize',10,...
        'FontWeight','bold',...
        'Callback','close all, clear, run start');

quit_c =  uicontrol(gcf,...
        'Style','Pushbutton',...
        'Position',[40+140+40+140+40 35 145 30],...
        'String','Quit',...
        'FontSize',10,...
        'FontWeight','bold',...
        'Callback','close all, clear');


%---end-of-script---------------------------------------------------------%
```