SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

TVORBA GUI PRE NÁVRH ROBUSTNÝCH REGULÁTOROV

DIPLOMOVÁ PRÁCA

FCHPT-5414-26067

2010 **Bc. Jozef Merčák**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

TVORBA GUI PRE NÁVRH ROBUSTNÝCH REGULÁTOROV

DIPLOMOVÁ PRÁCA

FCHPT-5414-26067

Študijný program: automatizácia a informatizácia v chémii a potravinárstve

Číslo a názov študijného odboru: 5.2.14 automatizácia

Školiace pracovisko: Oddelenie informatizácie a riadenia procesov

Vedúci záverečnej práce/školiteľ: Ing. Jana Závacká

Bratislava 2010 **Bc. Jozef Merčák**

Slovenská technická univerzita v Bratislave
Oddelenie informatizácie a riadenia procesov

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2009/2010
Evidenčné číslo: FCHPT-5414-26067

S T U
F C H P T

ZADANIE DIPLOMOVEJ PRÁCE

Študent:	**Bc. Jozef Merčák**
ID študenta:	26067
Študijný program:	automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor:	5.2.14 automatizácia
Vedúca práce:	Ing. Jana Závacká

Názov práce: **Tvorba GUI pre návrh robustných regulátorov**

Špecifikácia zadania:

- oboznámenie sa s problematikou intervalových neurčitostí a robustného riadenia (návrh robustných regulátorov)
- zvládnutie grafického užívateľského rozhrania (GUI) v prostredí Matlab
- vytvorenie GUI prostredia pre návrh robustných regulátorov
- porovnanie robustných regulátorov s regulátormi navrhnutými pomocou experimentálnych metód

Rozsah práce: 40

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

L. S.

Bc. Jozef Merčák
Študent

prof. Ing. Miroslav Fikar, DrSc.
Vedúci pracoviska

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

Poďakovanie

Za odbornú pomoc pri písaní diplomovej práce, cenné rady a usmernenia ďakujem Ing. Jane Závackej

Čestné prehlásenie

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne, podľa pokynov vedúceho práce a s použitím zdrojov uvedených v literatúre.

V Bratislave, 13. Mája .....

Podpis

Súhrn

Práca sa venuje oboznámeniu sa s GUIDE – grafickou užívateľskou aplikáciou, ktoré je integrované do prostredia Matlabu. Následne bola aplikácia využitá na tvorbu programov slúžiacich na návrh robustného P regulátora a robustného PI regulátora, ktorý je porovnávaný s inými typmi regulátorov. Návrh robustného PI regulátora je založený na vykreslení (k_p, k_i) – roviny a výberu vhodných parametrov regulátora. Na podobnom princípe bol zostavený program na výpočet parametra zosilnenia P regulátora, ktorý je vynesený do komplexnej roviny. Posledný program má za cieľ porovnať navrhnutý robustný PI regulátor s vybranými experimentálnymi metódami návrhu klasických PI regulátorov. Pomocou navrhnutých regulátorov sú testované systémy s intervalovými neurčitosťami a robustnosť navrhnutého PI regulátora sa testuje pomocou Charitonovovej teorémy. Všetky programy sú dostupné v prílohe k diplomovej práci.

Kľúčové slová: robustné riadenie, syntéza regulátorov, GUIDE

Abstract

The work is familiar with GUIDE - graphical user application that is integrated into Matlab. Application was subsequently used to develop programs for design Robust P controller and robust PI controller, which is compared with other types of controllers. Robust PI controller design is based on the plotting (kp, ki) - plane and the selection of appropriate controller parameters. On a similar principle was established a program to calculate the gain parameter of the P controller, which is delivered into the complex plane. The last program is to compare robust PI controller with the selected experimental design including conventional PI controllers. With controllers are tested systems with interval uncertainties and robustness of the proposed PI controller is tested by Kharitonov theorem. All programs are available in the Annex to the thesis.

Keywords: Robust Control, Synthesis of Controllers, GUIDE

Obsah

Zoznam príloh

Elektronická verzia diplomovej práce sa nachádza na priloženom CD.

Zoznam ilustrácií

Úvod

V súčasnosti je prítomnosť výpočtovej techniky pri rôznych úlohách, alebo riešeniach nejakej problematiky nevyhnutná. Podrobná znalosť ovládania aplikácií určených na riešenie konkrétnych problémov v niektorých prípadoch brzdí toto riešenie a to v prípadoch, kedy je znalosť potrebnej aplikácie len prostriedkom na dosiahnutie riešenia.

Za týmto účelom sú mnohé programy vybavené tzv. GUI, teda grafickým prostredím, ktoré do značnej miery zjednodušuje prácu. Softvérom, ktorého súčasťou je takéto GUI je aj Matlab. Keďže metódy, ktorých riešenie si vyžaduje prácu v tomto prostredí sú pomerne náročné na výpočet a znalosť problematiky, je v tomto prípade použitie GUI veľkou výhodou.

Jednou z metód, ktorých riešenie sa realizuje v prostredí Matlabu je aj návrh parametrov robustného PI regulátora. Táto metóda je založená na poznatkoch zachytených v Charitonovovej teoréme. Teoréma ako taká nie je zložitá, ale konkrétny výpočet je pomerne náročný na numerické operácie. Spracovanie metódy, pomocou ktorej možno navrhnúť parametre robustného PI regulátora, do podoby aplikácie GUI je preto značnou výhodou, ktorá šetrí čas.

Avšak ani vytváranie projektov v tomto prostredí nie je najjednoduchšie, keďže pri vytváraní treba mať na pamäti rôzne aspekty budúceho programu. Tvorca musí dbať na užívateľskú prívetivosť pri návrhu a teda je nútený spracovaný problém pretaviť do čo možno najjednoduchšej grafickej podoby. Taktiež musí do programu začleniť aj programovú stránku úlohy, čo si vyžaduje dobrú znalosť spracovávaného problému a programovú zručnosť.

1 TEÓRIA ROBUSTNÉHO RIADENIA

Vo všeobecnosti platí fakt, že pri modelovaní reálnych dynamických systémov sa nám nikdy nepodarí zostaviť úplne presný matematický model. Aj keď systém, ktorého model vykazuje po simulačnej stránke stabilitu, nemusí v skutočnosti dosahovať požadovanú úroveň stability, ba dokonca môže byť nestabilný. To vedie k poznatku, že pri metódach, ktoré sa používajú pri návrhoch riadenia sa musí brať dôraz na túto skutočnosť. V neposlednom rade to vedie k vzniku (zavedeniu) spätnej väzby. Táto by inak neexistovala, nebyť rozdielov medzi modelom riadeného systému a samotným systémom, ktoré majú zovšeobecňujúci názov neurčitosti (Matušů, 2005).

Vznik neurčitostí môže mať hneď niekoľko príčin:

- niektoré fyzikálne parametre systému sú buď zistené nepresne, alebo nie sú zistené vôbec
- niektoré parametre, ktoré majú v linearizovanom modeli charakter konštánt sa v skutočnosti časom menia
- obmedzené, alebo mylné chápanie fyzikálnej podstaty modelovaného systému
- časová a finančná náročnosť získania presného matematického modelu

Modely blížiace sa k presnej interpretácii systému sú veľmi zložité a tým pádom nepoužiteľné na riadenie.

V danom matematickom modeli sa dajú neurčitosti popísať dvoma spôsobmi:

a) Neparametricky – jeho podstatou je obmedzenie oblasti možného rozptylu frekvenčnej charakteristiky. Je vhodný pri zanedbaní rýchlych dynamických vlastností a nelinearít.

b) Parametricky – vyjadruje známu štruktúru, ale neznáme hodnoty konkrétnych fyzikálnych parametrov riadeného systému. Hodnoty týchto parametrov sú potom ohraničené intervalom.

Predmetom záujmu často býva neurčitý charakteristický polynóm uzavretého regulačného obvodu (Hurák, 2005):

$$p(s,q) = \sum_{i=0}^{n} \rho_i(q) s^i \qquad (1)$$

Ak pracujeme s neurčitými systémami, často sa uvažuje množina obmedzujúca parametre (Q), ktorá je ohraničená nejakou normou (H_∞, H_2).

Pre normu H_∞ platí:

$$\|q\|_\infty = \max_i |q_i| \tag{2}$$

Pre normu H_2 (Euklidova) platí:

$$\|q\|_2 = \left(\sum_{i=1}^{l} q_i^2\right)^{\frac{1}{2}} \tag{3}$$

Do polynómu $p(s,q) = \sum_{i=0}^{n} \rho_i(q) s^i$ sa neurčitosť vnáša cez funkcie koeficientov $\rho_i(q)$ a to hneď niekoľkými spôsobmi, na základe ktorých rozlišujeme niekoľko základných štruktúr neurčitosti:

- jednoparametrická štruktúra neurčitosti
- intervalová (nezávislá) štruktúra neurčitosti
- afinná lineárna štruktúra neurčitosti
- multilineárna štruktúra neurčitosti
- nelineárna štruktúra neurčitosti (polynomická)

1.1 Intervalové neurčitosti

Zatiaľ čo jednoparametrická neurčitosť nadobúda len jednu konkrétnu hodnotu, neurčitosť intervalového charakteru nadobúda rôzne hodnoty. Aj keď predstava o tom, že neurčitosť nadobúda rôzne hodnoty, sa môže zdať zovšeobecnená či dokonca zjednodušená, treba však podotknúť istý predpoklad, ktorý musí intervalová neurčitosť spĺňať.

Nutným predpokladom je, aby intervalová neurčitosť mala nezávislú štruktúru.

V prípade neurčitého polynómu

$$p(s,q) = \sum_{i=0}^{n} \rho_i(q) s^i \tag{4}$$

má tento nezávislú štruktúru neurčitosti v prípade, že každá zložka q_i z q vstupuje len do jedného koeficienta.

V prípade rodiny polynómov:

$$P = \{p(.;q) : q \in Q\} \tag{5}$$

môžeme tento nazvať intervalovým polynómom ak:

a) má nezávislú štruktúru neurčitosti

b) každý koeficient je spojitou funkciou q

c) Q je kváder

Vyzerá to napríklad takto:

$$\begin{aligned} p_1(s,q) &= (2+2q_2)s^2 + (1+q_1)s + (10+10q_0); \quad q_0 \in \langle -6;2 \rangle, q_1 \in \langle 3;5 \rangle, q_2 \in \langle -7;10 \rangle \\ p_2(s,q) &= 2s^2 + (1+q_1+q_2)s + (10+10q_0); \quad q_i \in \langle 0;5 \rangle \end{aligned} \tag{6}$$

pričom v druhom polynóme pri koeficiente s vidieť redundanciu, ktorá nie je ničím výnimočná.

Existuje aj zjednodušený zápis intervalového polynómu:

$$p(s,q) = \sum_{i=1}^{n} [q_i^-, q_i^+] s^i \tag{7}$$

napríklad:

$$p(s,q) = [-1;1]s^4 + [-2;2]s^3 + [-3;3]s^2 + [-4;4]s + [-5;5] \tag{8}$$

1.1.1 Charitonovova veta

Síce Charitonovova veta bola publikovaná už v roku 1978 v ruskej technickej literatúre, zvyšku sveta ju predstavili až neskôr Barmish a Bialas.

V odborných kruhoch považovaná za míľnik v analýze stability robustnosti systémov vyznačujúcich sa parametrickými neurčitosťami hovorí, že daný intervalový polynóm, ktorý má invariantný stupeň je vtedy a len vtedy stabilný, ak stabilitu vykazujú všetky štyri Charitonovove polynómy.

Upúšťa sa preto od nutnosti testovať stabilitu všetkých existujúcich hraničných možností. Konštrukcia Charitonovových polynómov je nasledovná (Matušů, 2005):

$$
\begin{aligned}
K_1 &= q_0^+ + q_1^+ s + q_2^- s^2 + q_3^- s^3 + q_4^+ s^4 + q_5^+ s^5 + \ldots \\
K_2 &= q_0^- + q_1^+ s + q_2^+ s^2 + q_3^- s^3 + q_4^- s^4 + q_5^+ s^5 + \ldots \\
K_3 &= q_0^- + q_1^- s + q_2^+ s^2 + q_3^+ s^3 + q_4^- s^4 + q_5^- s^5 + \ldots \\
K_4 &= q_0^+ + q_1^- s + q_2^- s^2 + q_3^+ s^3 + q_4^+ s^4 + q_5^- s^5 + \ldots
\end{aligned}
\tag{9}
$$

alebo:

$$
\begin{aligned}
&++--++\ldots \\
&-++--+\ldots \\
&--++--\ldots \\
&+--++-\ldots
\end{aligned}
\tag{10}
$$

1.2 Návrh robustného PI regulátora

Teória návrhu sa opiera o Charitonovovu teorému. Ide o jednoduchú metódu na výpočet parametrov PI regulátora, ktorý stabilizuje riadený systém s pevne danými parametrami. Navrhnutá metóda je založená na vykreslení oblastí do (k_p, k_i) – roviny a vypočítava stabilné PI regulátory (Tan, et al., 2003).

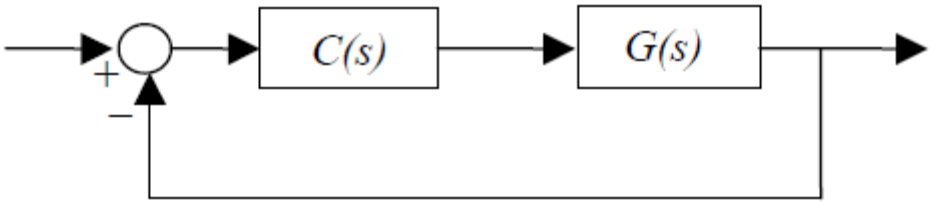
Stabilizácia použitého PI regulátora predpokladá použitie SISO riadeného systému na Obr. 1 opísaného prenosom

$$
G(s) = \frac{N(s)}{D(s)} \tag{11}
$$

ktorý je riadený daným PI regulátorom v tvare

$$
C(s) = k_p + \frac{k_i}{s} = \frac{k_p s + k_i}{s} \tag{12}
$$

Problémom je vypočítať parametre PI regulátora v rovnici (12), ktorý stabilizuje tento systém.



Obr. 1 Schéma uzavretého regulačného obvodu

Rozložením polynómov čitateľa a menovateľa z rovnice (11) na párne a nepárne časti a nahradením premennej s na $j\omega$ dostaneme

$$G(s) = \frac{N_e\left(-\varpi^2\right) + j\varpi N_o\left(-\varpi^2\right)}{D_e\left(-\varpi^2\right) + j\varpi D_o\left(-\varpi^2\right)} \tag{13}$$

Uzavretú slučku charakteristického polynómu systému môžeme napísať ako

$$\begin{aligned}\Delta(s) = &\left[k_i N_e\left(-\varpi^2\right) - k_p\varpi^2 N_o\left(-\varpi^2\right) - \varpi^2 D_o\left(-\varpi^2\right)\right] \\ &+ j\left[k_p\varpi N_e\left(-\varpi^2\right) + k_i\varpi N_o\left(-\varpi^2\right) + \varpi D_e\left(-\varpi^2\right)\right] = 0\end{aligned} \tag{14}$$

Potom ak porovnáme reálnu a imaginárnu časť Δ(s), získame

$$k_p\left(-\varpi^2 N_o\left(-\varpi^2\right)\right) + k_i\left(N_e\left(-\varpi^2\right)\right) = \varpi^2 D_o\left(-\varpi^2\right) \tag{15}$$

a

$$k_p\left(N_e\left(-\varpi^2\right)\right) + k_i\left(N_o\left(-\varpi^2\right)\right) = D_e\left(-\varpi^2\right) \tag{16}$$

Pre zjednodušenie práce s jednotlivými časťami môžeme zaviesť

$$\begin{aligned}Q(\varpi) &= -\varpi^2 N_o\left(-\varpi^2\right) \\ R(\varpi) &= N_e\left(-\varpi^2\right) \\ S(\varpi) &= N_e\left(-\varpi^2\right) \\ U(\varpi) &= N_o\left(-\varpi^2\right) \\ X(\varpi) &= \varpi^2 D_o\left(-\varpi^2\right) \\ Y(\varpi) &= -D_e\left(-\varpi^2\right)\end{aligned} \tag{17}$$

Potom rovnice (15) a (16) môžeme napísať ako

$$\begin{aligned}k_p Q(\varpi) + k_i R(\varpi) &= X(\varpi) \\ k_p S(\varpi) + k_i U(\varpi) &= Y(\varpi)\end{aligned} \tag{18}$$

Z tejto rovnice potom odvodíme vzťahy na výpočet jednotlivých parametrov PI regulátora

$$k_p = \frac{X(\varpi)U(\varpi) - Y(\varpi)R(\varpi)}{Q(\varpi)U(\varpi) - R(\varpi)S(\varpi)} \tag{19}$$

a

$$k_i = \frac{X(\varpi)U(\varpi) - X(\varpi)S(\varpi)}{Q(\varpi)U(\varpi) - R(\varpi)S(\varpi)} \tag{20}$$

Riešením týchto dvoch rovníc súčasne sa získa ohraničená množina *l(kp,ki,ω)* v rovine (k_p, k_i), ktorá túto rozdelí na stabilnú a nestabilnú oblasť. Ak zvolíme bod vnútri stabilnej oblasti (približne v jej strede), získame parametre PI regulátora, ktorý bude stabilne riadiť prenos v systéme.

Príklad 1

Uvažujme riadený systém (Obr. 1) s prenosovou funkciou

$$G(s) = \frac{10}{s^3 + 5s^2 + 2s + 1} \tag{21}$$

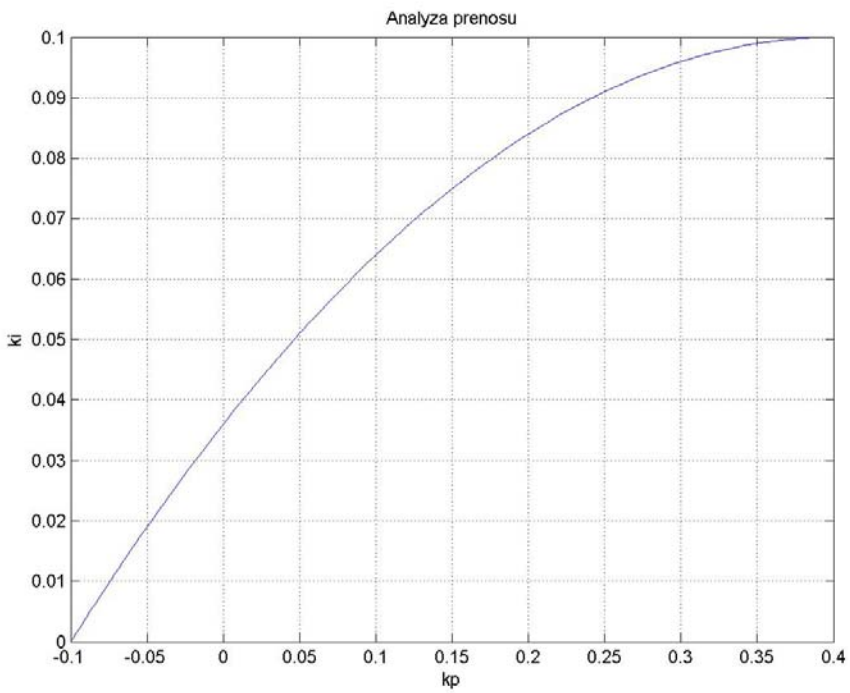
potom podľa rovníc (19) a (20) vypočítame

$$k_p = \frac{50\varpi^2 - 10}{100} = 0.5\varpi^2 - 0.1 \tag{22}$$

a

$$k_i = \frac{-10\varpi^4 + 20\varpi^2}{100} = -0.1\varpi^2 + 0.2 \tag{23}$$

Pre zvolené ω = 1 a vykreslení do roviny dostaneme neuzavretú množinu (Obr. 2), čo znemožní určiť hodnoty parametrov PI regulátora.

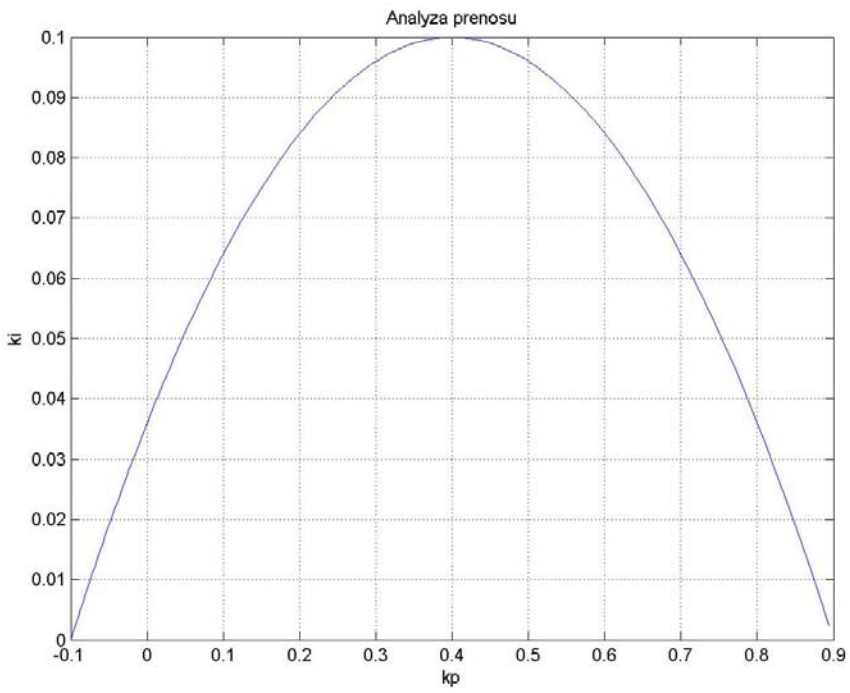


Obr. 2 Vykreslenie stabilnej množiny pri ω=1

Výpočet premennej omega je založený na predpoklade, že

$$G\left(j\varpi\right) = 0 \tag{24}$$

Ak použijeme predpoklad (24), výpočtom získame hodnotu 1.4142. Potom po vykreslení, na základe rovníc (22) a (23), dostaneme pre systém (21) množinu stabilizujúcich regulátorov (Obr. 3).



Obr. 3 Vykreslenie stabilnej množiny pri ω=1.4142

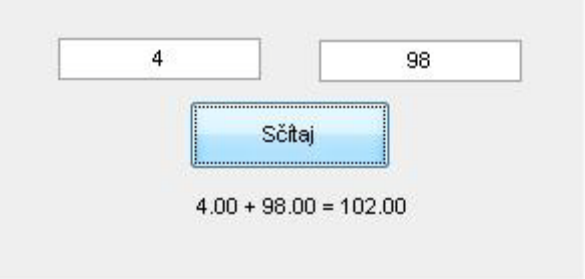
1.2.1 Obmedzenia pre metódu návrhu robustného PI regulátora

Testovaním aplikácie využívajúcej túto metódu sa prišlo na skutočnosť, že nie je vhodná pre systémy prvého rádu a druhého rádu s čitateľom prvého rádu. Nízky stupeň čitateľa nedovoľuje výpočet parametra omega, pretože nastáva prípad, kde rovnica (24) nemá riešenie.

2 TEÓRIA GUIDE

Grafické užívateľské rozhranie (GUI) má grafickú podobu jedného alebo viacerých okien, ktoré obsahujú ovládacie prvky, nazývané komponenty, tie umožňujú užívateľovi vykonávať prácu s projektom. Užívateľ GUI nevytvára skript alebo nepíše príkazy do príkazového riadku aby sa vykonali nejaké úlohy. Tiež nemusí rozumieť detailom ako sa ním zadané úlohy vykonávajú.

Každý projekt GUI pozostáva z komponentov rôznych druhov ako sú napr.: tlačidlá, polia textu, editovacie polia, menu, atď. Keďže každý z komponentov je iný, tiež inak reaguje ak ho užívateľ použije. Napríklad v prípade jednoduchej kalkulačky (Obr. 4), užívateľ zadá do editovacích polí číselné hodnoty a stlačí tlačidlo. Výsledkom je výpis výsledku.



Obr. 4 Jednoduchý príklad projektu GUIDE

Princíp fungovania takýchto projektov spočíva v tom, že jednotlivé komponenty „čakajú“ na impulz od užívateľa a potom vykonávajú príslušnú akciu. Každý komponent má jeden, alebo viacej preddefinovaných situácii, ktoré môžu pri práci s ním nastať. Tieto situácie môžu byť zapísané do matlabovského skriptu a nazývajú sa Callbacky. To aby sa určitý Callback uskutočnil musí byť zapríčinené nejakou udalosťou podmienenou zo strany užívateľa. Môže ísť o stlačenie tlačidla, vybratie položky z menu, zapísanie textového reťazca alebo čísla do editovacieho poľa. Takéto situácie sa nazývajú udalosti a je na tvorcovi GUI projektu, aké operácie priradí jednotlivých udalostiam. Callbacky sú v skripte písané jednoducho ako funkcie, je to z dôvodu väčšej flexibility a rýchlosti pri výpočtoch, čo funkcie rozhodne ponúkajú.

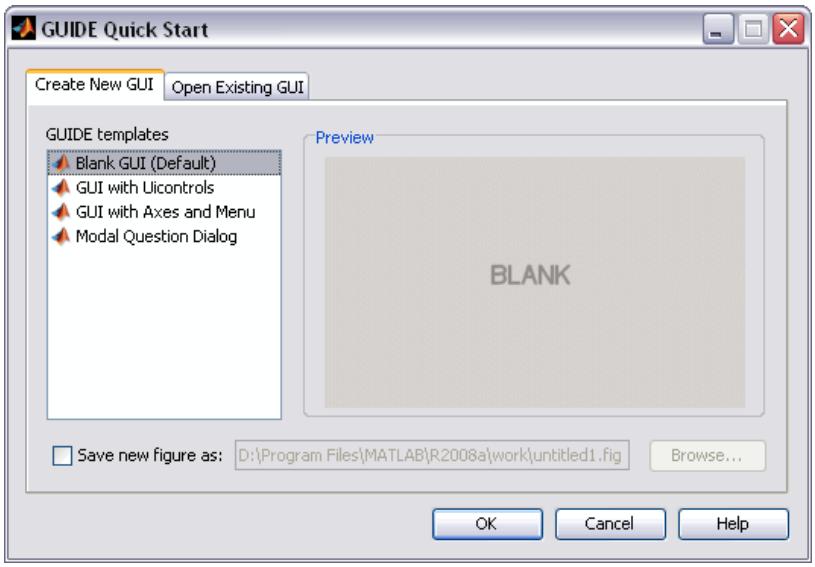
2.1 Navrhovanie projektu v prostredí GUIDE

Grafické užívateľské rozhranie v prostredí Matlab môže byť vytvorené dvoma spôsobmi. Programátor môže aplikáciu vytvoriť pomocou tzv. Layout Editora, ktorý slúži na grafický návrh samotného programu, teda na návrh vzhľadu okna aplikácie. Celý tento návrh je

zachytený v .fig súbore. Samotná funkčná časť sa programuje v .m súbore vytvorenom spoločne so súborom .fig.

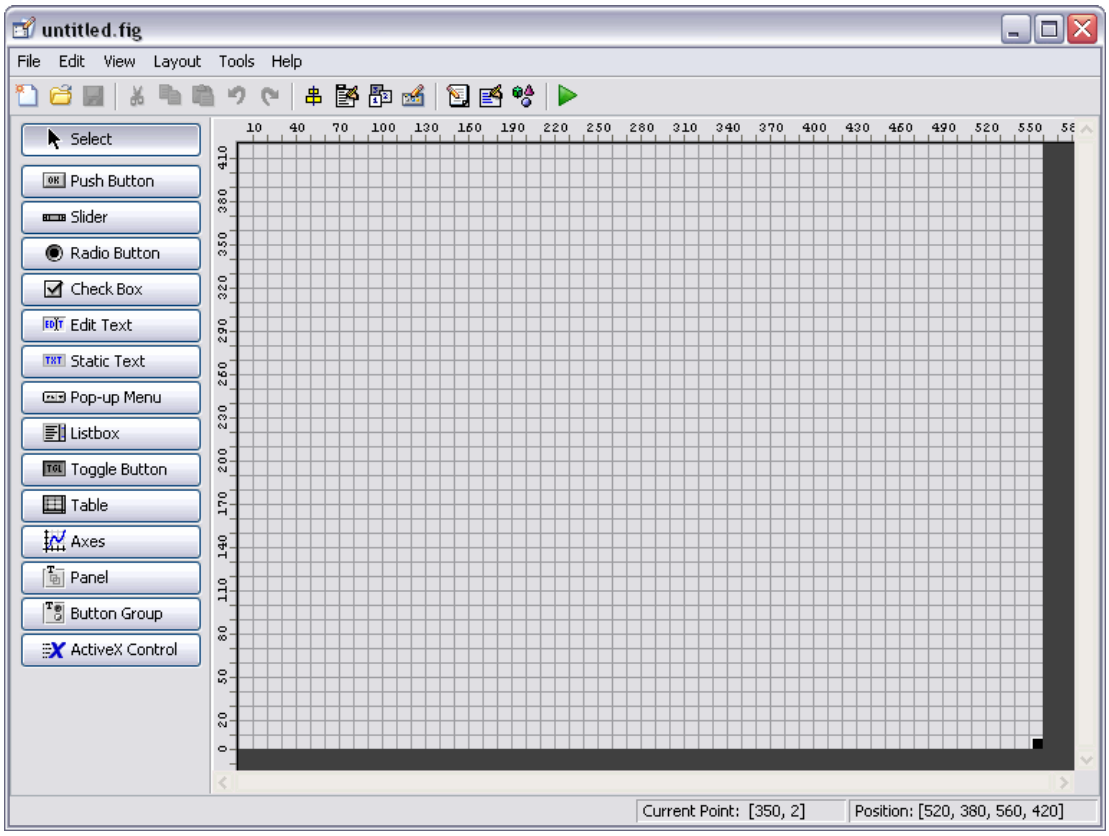
Druhým spôsobom je celú štruktúru navrhovanej aplikácie zachytiť len v jednom .m súbore. Tu je okrem funkčnej časti definovaný aj vzhľad aplikácie a to pomocou príkazov, ktoré definujú rozmiestnenie jednotlivých častí okna programu, ich farebné prevedenie, názvy jednotlivých komponentov a všetky vlastnosti, ktoré sa nastavujú v prostredí Layout Editora. V tomto sa zračí istá nevýhoda, keďže pozíciu a farby komponentov je nutné definovať vo vektorovom tvare a teda tvorca projektu nemá bezprostrednú predstavu o vzhľade. Toto je elegantne riešené v GUIDE prostredí prostredníctvom „Layout Editora“.

Prostredie GUIDE, interpretované v Layout Editore, sa spúšťa viacerými spôsobmi, buď priamo z príkazového riadka príkazom *guide*, alebo kliknutím na ikonu  v hlavnom okne Matlabu. Počas spúšťania sa zobrazí ponuka (Obr. 5), s viacerými možnosťami ako začať vytvárať vlastný projekt. Štandardne je prednastavená možnosť začať vytvárať nový projekt s prázdnou šablónou, no GUIDE ponúka aj tri možnosti preddefinovaných šablón, z ktorých každá obsahuje rôzne typy komponentov. Existuje tiež možnosť otvoriť už existujúci projekt.



Obr. 5 Spustenie GUIDE

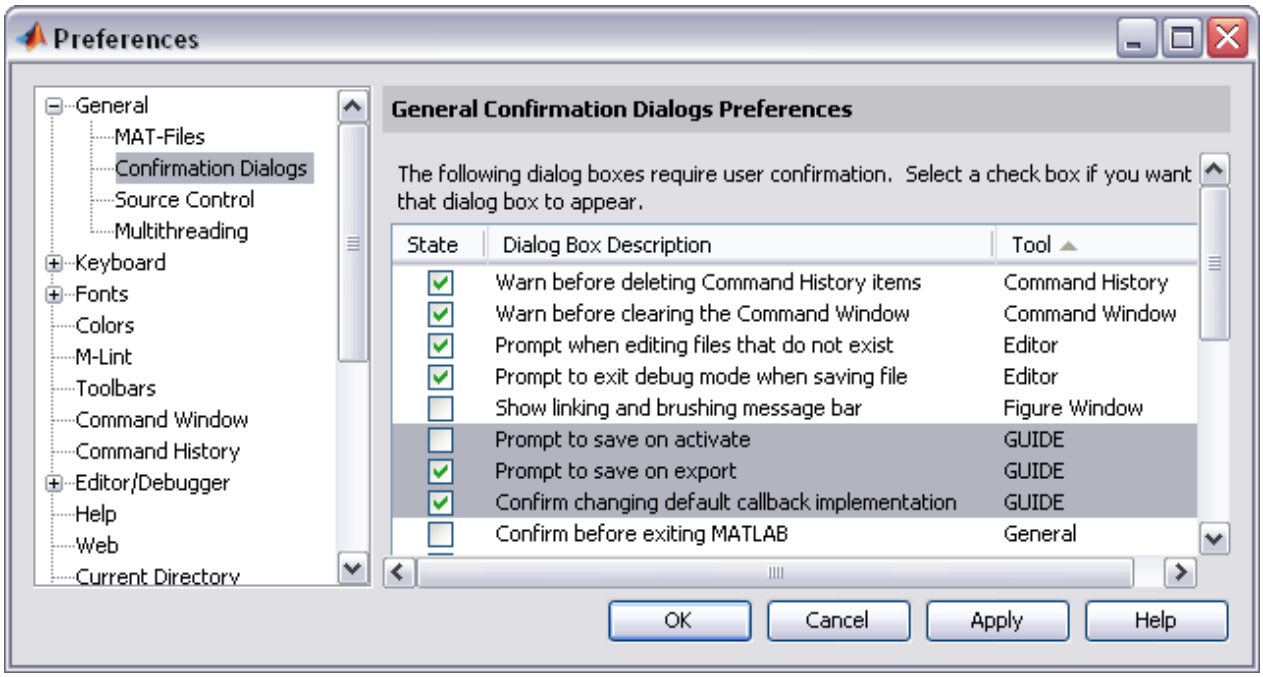
Prostredie Layout, teda návrhového editora (Obr. 6), ponúka intuitívnu prácu pri návrhu vlastnej užívateľskej obrazovky jednoduchým pridávaním komponentov do hlavného okna, čo dáva celkom jasnú predstavu o podobe výsledného projektu.



Obr. 6 Prostredie Layout Editora

Tvorca musí mať samozrejme na začiatku práce určitú predstavu o tom, ako bude jeho projekt vyzerať. Zhmotniť túto predstavu do finálneho výsledku je pre začínajúcich programátorov jednoduchšie práve v tomto editore keďže pozície, farby a vôbec všetky vlastnosti jednotlivých komponentov je možné v tomto prostredí nastaviť podľa vlastných predstáv.

Samotné prostredie GUIDE si tvorca projektu môže prispôsobiť vlastným potrebám v menu preferencií. Toto menu sa zobrazí kliknutím v okne na položku File > Preferences. V otvorenej stromovej štruktúre nastavení, kliknutím na vetvu General > Confirmation Dialog, sa zobrazí prehľad situácií, na ktoré Matlab dokáže reagovať upozornením. Tieto dialógové okná sa zobrazujú vtedy, ak prostredie vyžaduje povolenie užívateľa. Ak chceme upraviť možnosti dialógových okien upozornení pre GUIDE, všimneme si riadky venované tomuto prostrediu (Obr. 7).

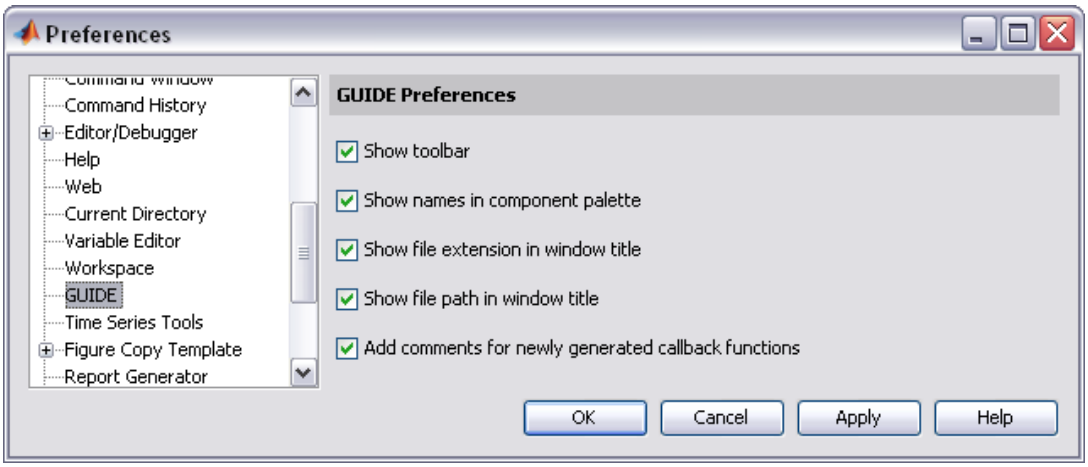


Obr. 7 Nastavenia dialógových okien v prostredí GUIDE

Prvou možnosťou je *„Promt to save on activate“*. Ak aktivujete GUI stlačením tlačidla Spustiť , budete upozornení o tom či chcete svoj projekt predtým uložiť s možnosťou, aby sa toto hlásenie už nezobrazovalo. Toto hlásenie sa nemusí zobrazovať, avšak v tom prípade zakaždým ako sa spustí GUI cez tlačidlo Spustiť, vopred sa uloží celý projekt (MathWorks, 2009).

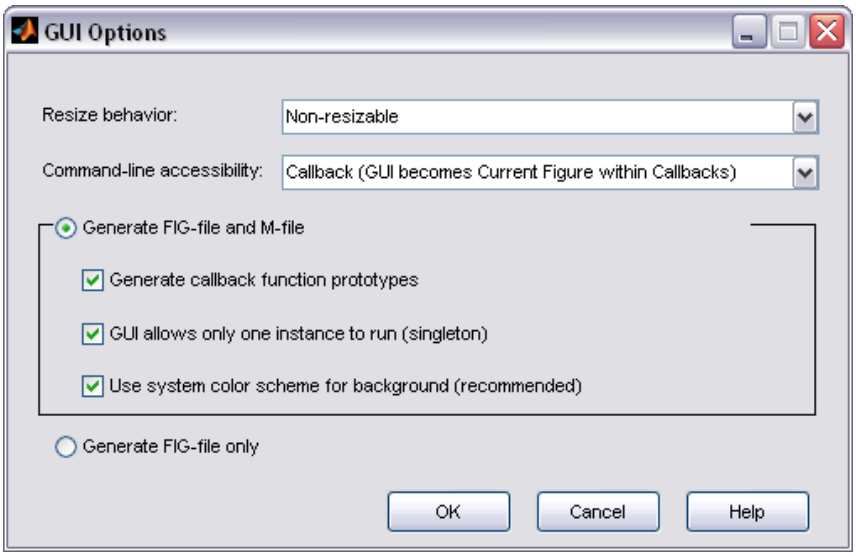
Ak zvolíte možnosť Export z menu File *„Promt to save on export“*, tiež sa zobrazí hlásenie o tom, či chcete najskôr projekt uložiť. Okno ponúka možnosť viackrát toto hlásenie nezobrazovať. Export projektu slúži na vygenerovanie projektu do samostatného súboru .m. Zároveň sa však vytvorí aj súbor .mat ktorý uchováva premennú „mat“. Táto premenná je typu cell a sú v nej informácie o všetkých Callbackoch komponentov, ktoré v projekte vystupujú.

Nižšie v stromovej štruktúre preferencií, sa nachádza samotná vetva nastavení venovaná prostrediu GUIDE (Obr. 8). Táto časť sa venuje prevažne zjednodušeniu práce v prostredí a to vo forme rozšíreného zobrazenia ovládacích prvkov.



Obr. 8 Nastavenie rozšíreného zobrazenia prvkov v prostredí GUIDE

Ďalšie možnosti nastavení sú implementované priamo v prostredí GUIDE. V položke Tools > GUI Options (Obr. 9) má tvorca projektu možnosť rozhodnúť o podobe vygenerovaného projektu, či o správaní sa pri zmene veľkosti okna projektu.



Obr. 9 Možnosti nastavení generovania a správania sa projektov v GUIDE

Po upravení prostredia vlastným požiadavkám môžeme pristúpiť v návrhu vlastného projektu. Pridávanie komponentov do projektu je možné vykonať dvoma spôsobmi. Pre rýchle umiestenie stačí kliknúť vľavo na lištu komponentov a ťahom umiestniť komponent na požadované miesto v okne. Ak je nutné meniť preddefinovanú veľkosť komponentu (veľkosť osí na lepšie zobrazenie grafov), je lepšie ak klikneme na lištu komponentov, prejdeme nad plochu okna projektu pričom sa kurzor zmení na krížik a ťahom definujeme vlastné rozmery komponentu.

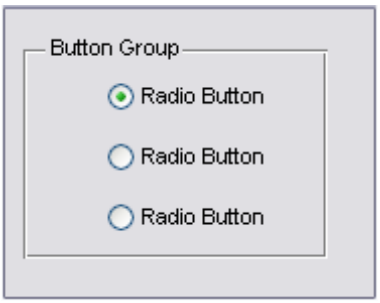
Komponenty GUI plnia v projekte rôzne funkcie a preto majú aj rôzne vlastnosti, a preto sa používajú na rôzne udalosti. Hlavnú skupinu tvoria komponenty slúžiace pre vkladanie údajov samotným užívateľom. Údaje sa môžu vkladať buď z klávesnice (Edit Text), ale aj prostredníctvom myši a to z predvolenej ponuky (Listbox , Popup Menu).

Pre nastavenie bližších parametrov v projekte sú k dispozícií iné komponenty(Checkbox , Slider , Radio Button , Toggle Button).

Hlavné udalosti spojené s operáciami v jadre celého projektu sa definujú prevažne pre jeden typ komponentu (Push Button).

Zobrazovanie výsledkov sa premieta do komponentov Axes , Table prípadne Static Text .

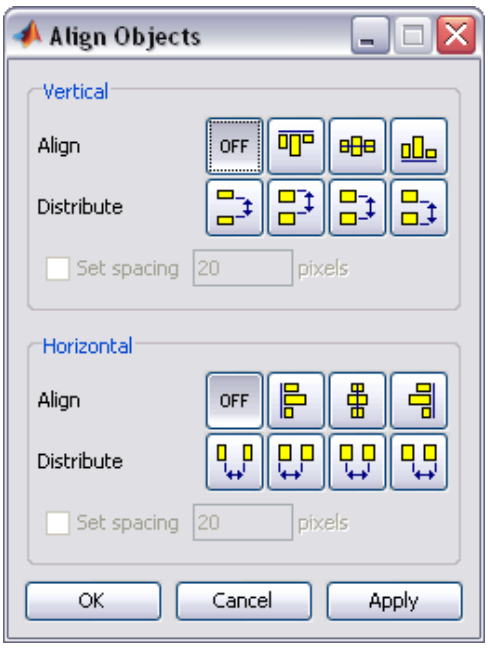
Nemusíme pridávať len komponenty samostatne. Existujú aj prvky Panel a Button Group , určené na integráciu komponentov s príbuzným významom. Prvok Button Group je špecifický svojim charakterom, pretože sa doň vkladajú komponenty, ktorých akcie sa navzájom ovplyvňujú a súvisia spolu(skupina checkboxov, alebo toggle buttonov). Ak vložíme takéto komponenty do tohto prvku, tak vo výslednom programe sa budú spoločne ovplyvňovať a to tak, že aktívny bude vždy len jeden konkrétny prvok (Obr. 10). Takýto prvok značne urýchľuje prácu.



Obr. 10 Ukážka činnosti prvku Button Group

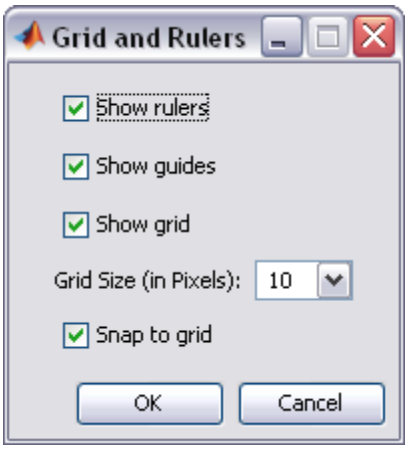
Ďalšími nástrojmi na sprehľadnenie a uľahčenie návrhu grafickej stránky vytváraného projektu sú Align Objects (Obr. 11). Tento nástroj usporiada vyznačené komponenty, pričom upravuje ich polohu navzájom medzi sebou. Nástroj umožňuje ako horizontálnu tak aj

vertikálnu orientáciu. Dovoľuje nastaviť zarovnanie a odsadenie medzi komponentmi navzájom. Tiež je možné veľkosť odsadenia nastaviť ručne.



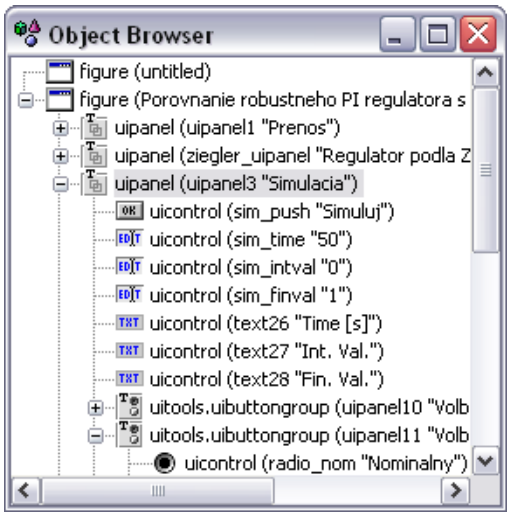
Obr. 11 Nástroj Align Objects

V ploche navrhovaného projektu je tiež možné zobraziť mriežku a vodiče pre uľahčenie usporiadania komponentov (Obr. 12).



Obr. 12 Nástroj na uľahčenie usporiadania komponentov

Ak sa v objekte nachádza väčšie množstvo komponentov a tie sú integrované do rôznych panelov, začne byť práca s nimi komplikovaná v dôsledku ťažšej orientácie. Jednoduchý prehľad ich usporiadania poskytuje Object Browser  (Obr. 13).



Obr. 13 Object Browser

Údaje o všetkých komponentoch vrátane komponentu samotného okna projektu sú zahrnuté v tzv. „Property Inspectore“, ktorý sa spúšťa buď dvojklikom na komponent, ktorého vlastnosti chceme zobraziť, alebo kliknutím na ikonu . V okne, ktoré sa následne zobrazí, sú všetky vlastnosti komponentu, ktorý je práve vyznačený.

Po navrhnutí vlastného projektu, teda keď projekt má finálnu podobu v Layout Editore, sa projekt uloží pod ľubovoľným názvom a automaticky po uložení sa vytvorí súbor .m s rovnakým názvom akým sme nazvali súbor .fig. Súbor .fig obsahuje kompletný popis návrhu GUI a jeho komponentov, ako napr. tlačidlá, osi, panely, menu, atď. V tomto momente končí fáza grafického návrhu projektu a začína programátorská fáza. V .m súbore sú zachytené hlavné Callbacky všetkých komponentov projektu. Súbor .m obsahuje inicializačný skript a predlohu niekoľkých Callbackov, ktoré sú potrebné na riadenie činnosti GUI. Do tohto súboru je nutné pridať Callbacky napísané pre vami definované GUI komponenty. Tieto Callbacky majú podobu jednoduchých funkcií s vopred definovanými vstupnými parametrami. Súbor typu .m, ktorý je vygenerovaný GUIDE-sprievodcom je súborom funkcií. Meno hlavnej funkcie je totožné s menom samotného súboru. Každý Callback v tomto súbore je podfunkciou hlavnej funkcie (MathWorks, 2009).

Keď GUIDE vygeneruje .m súbor, vloží doň automaticky Callbacky pre inicializáciu samotného projektu, ktorý sa chápe ako hlavný komponent *Figure* a tiež vloží užívateľom definované Callbacky pre každý komponent. Hlavnou funkciou vygenerovaného .m súboru je funkcia nesúca meno projektu, táto sa nesmie editovať. Ďalej nasledujú samotné Callbacky hlavného komponentu a to *OpeningFcn* (Obr. 14) a *OutputFcn* (Obr. 15).

```
47      function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
48      % This function has no output args, see OutputFcn.
49      % hObject    handle to figure
50      % eventdata  reserved - to be defined in a future version of MATLAB
51      % handles    structure with handles and user data (see GUIDATA)
52      % varargin   command line arguments to untitled (see VARARGIN)
53
54      % Choose default command line output for untitled
55 -    handles.output = hObject;
56
57      % Update handles structure
58 -    guidata(hObject, handles);
59
60      % This sets up the initial plot - only do when we are invisible
61      % so window can get raised using untitled.
62 -    if strcmp(get(hObject,'Visible'),'off')
63 -        plot(rand(5));
64 -    end
```

Obr. 14 Funkcia *OpeningFcn*

V týchto sa definujú udalosti, ktoré sa majú vykonať pri otvorení projektu alebo jeho zatvorení. Využívajú sa vtedy, ak je potrebné vykonať nejaké príkazy bez toho aby si to užívateľ všimol, alebo aby tieto príkazy nemusel prácne sám vykonávať. Po týchto troch funkciách už nasledujú Callbacky jednotlivých komponentov obsiahnutých v projekte. Typy Callbackov môže užívateľ sám definovať a v prostredí GUIDE v Property Inspectore je dobre vidieť, aké Callbacky sú pre ten ktorý komponent dostupné.

```
70      % --- Outputs from this function are returned to the command line.
71      function varargout = untitled_OutputFcn(hObject, eventdata, handles)
72      % varargout  cell array for returning output args (see VARARGOUT);
73      % hObject    handle to figure
74      % eventdata  reserved - to be defined in a future version of MATLAB
75      % handles    structure with handles and user data (see GUIDATA)
76
77      % Get default command line output from handles structure
78 -    varargout{1} = handles.output;
```

Obr. 15 Funkcia *OutputFcn*

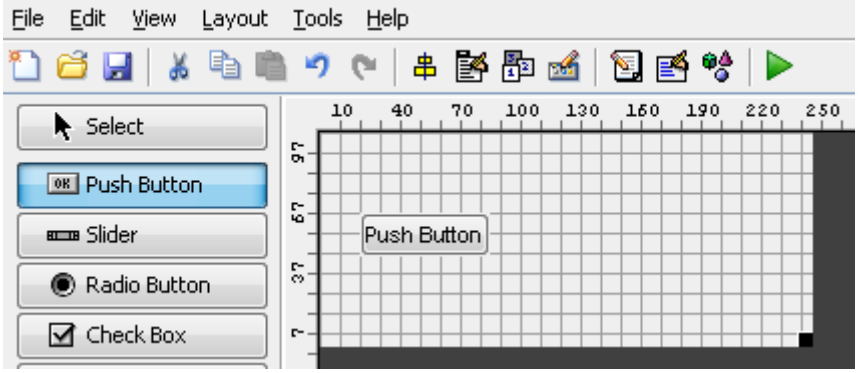
Oba súbory musia mať rovnaké meno a zvyčajne musia byť v tom istom adresári. Ak upravujete GUI v návrhovom editore, vaša práca sa uloží do .fig súboru, ak programujete GUI, vaša práca sa uloží v spolupracujúcom .m súbore (MathWorks, 2009).

Vkladanie vlastného skriptu spočíva v naprogramovaní jednotlivých komponentov v projekte.

2.2 Programovanie GUIDE

Push Button:

Programovanie tohto komponentu sa začína jeho vložením do projektu (MathWorks, 2009) (Obr. 16).



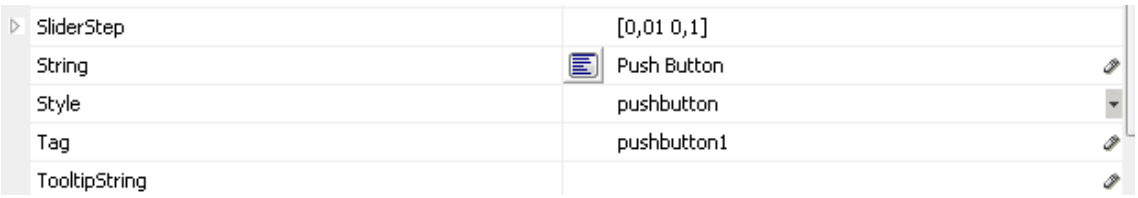
Obr. 16 Pridanie komponentu Push Button

Uložením projektu sa zároveň otvorí príslušný .m súbor v Editore. Skript bude okrem štandardného kódu reprezentujúceho samotný projekt obsahovať aj funkciu reprezentujúcu práve pridaný komponent (Obr. 17).

```
76      % --- Executes on button press in pushbutton1.
77      function pushbutton1_Callback(hObject, eventdata, handles)
78      % hObject    handle to pushbutton1 (see GCBO)
79      % eventdata  reserved - to be defined in a future version of MATLAB
80      % handles    structure with handles and user data (see GUIDATA)
81
```

Obr. 17 Funkcia komponentu Push Button

Názov tejto funkcie je zakotvený v parametri „Tag“ v Propery Inspectore ktorý sa zobrazí dvojklikom na príslušný komponent (Obr. 18). Tento parameter uľahčuje pomenovanie funkcie, aby bolo zrejmejšie, ktorému komponentu prislúcha, a teda aby bolo uľahčené programovanie (ak sa v projekte nachádza veľké množstvo funkcií, je vhodné zaviesť systém do pomenovania komponentov). „Tag“ sa nezobrazuje a neoznačuje názov komponentu pri behu programu. Táto vlastnosť je zakotvená v parametri „String“.



Obr. 18 Parameter „Tag“ v Property Inspectore

Ak chceme tomuto tlačidlu priradiť nejakú akciu, ktorá sa vykoná pri jeho stlačení, tak jednoducho napíšeme príslušné príkazy do jeho funkcie resp. Callbacku (Obr. 19).

```
80      % --- Executes on button press in pushbutton1.
81      function pushbutton1_Callback(hObject, eventdata, handles)
82      % hObject    handle to pushbutton1 (see GCBO)
83      % eventdata  reserved - to be defined in a future version of MATLAB
84      % handles    structure with handles and user data (see GUIDATA)
85 -    display Dovidenia
86 -    close(handles.figure1);
```

Obr. 19 Callback komponentu Push Button s vlastným kódom

Pridaním tohto kódu sa po stlačení tlačidla zobrazí v príkazovom riadku MatLabu text „Dovidenia“ a zároveň sa zavrie okno projektu.

Toggle Button:

Pre Callback tohto komponentu je vždy potrebné definovať kedy je aktivovaný a kedy deaktivovaný. Pre tento účel je v Property Inspectore parameter „Value“. Podobným spôsobom sa programuje aj komponent Radio Button alebo komponent Check Box keďže jeho hlavný Callback spočíva v tom či je tento komponent vyznačený alebo nie (Obr. 20).

```
153     % --- Executes on button press in togglebutton1.
154     function togglebutton1_Callback(hObject, eventdata, handles)
155     % hObject    handle to togglebutton1 (see GCBO)
156     % eventdata  reserved - to be defined in a future version of MATLAB
157     % handles    structure with handles and user data (see GUIDATA)
158
159     % Hint: get(hObject,'Value') returns toggle state of togglebutton1
160 -   isDown = get(hObject,'Value');
161 -   if isDown
162 -       disp('Toggle je aktivovane')
163 -   else
164 -       disp('Toggle je deaktivovane')
165 -   end
```

Obr. 20 Callback komponentu Toggle Button s vlastným kódom

Edit Text:

Toto je asi najdôležitejší komponent zo všetkých pretože slúži na priame zadávanie parametrov užívateľom. Údaj vložený to tohto textového poľa sa ukladá do parametra „String“. Z toho je zrejmé že údaje sú vkladané ako textové reťazce a to aj keď sú vložené čísla alebo špeciálne znaky špecifické pre prostredie MatLab (napr.: [ ] ; , atď.). Ak chceme napríklad vykonať nejakú matematickú operáciu s nami zadanými údajmi, je nutné ich najprv previesť na číselné znaky a až potom s nimi pracovať.

Na začiatok si môžeme do tohto komponentu poslať určitú hodnotu (Obr. 21).

```
182     % --- Executes during object creation, after setting all properties.
183     function cislo_CreateFcn(hObject, eventdata, handles)
184     % hObject    handle to cislo (see GCBO)
185     % eventdata  reserved - to be defined in a future version of MATLAB
186     % handles    empty - handles not created until after all CreateFcns called
187
188     % Hint: edit controls usually have a white background on Windows.
189     %       See ISPC and COMPUTER.
190
191     % % % % TENTO SKRIPT JE STANDARDNY
192 -   if ispc && isequal(get(hObject,'BackgroundColor'), ...
193             get(0,'defaultUicontrolBackgroundColor'))
194 -       set(hObject,'BackgroundColor','white');
195 -   end
196
197     % % % % TENTO SKRIPT JE PRIDANY UZIVATELOM
198 -   set(hObject,'String',4)
```

Obr. 21 Definovanie hodnoty komponentu pri štarte projektu

Tu si môžeme všimnúť iný názov funkcie, resp. Callbacku. Ako vždy sa názov funkcie skladá z názvu parametra „Tag“(číslo), podtržníka(_) a z typu Callbacku(CreateFcn). Skript v Callbacku „CreateFcn“ sa vykonáva spolu s počiatočným spustením projektu a predstavuje akúsi inicializáciu.

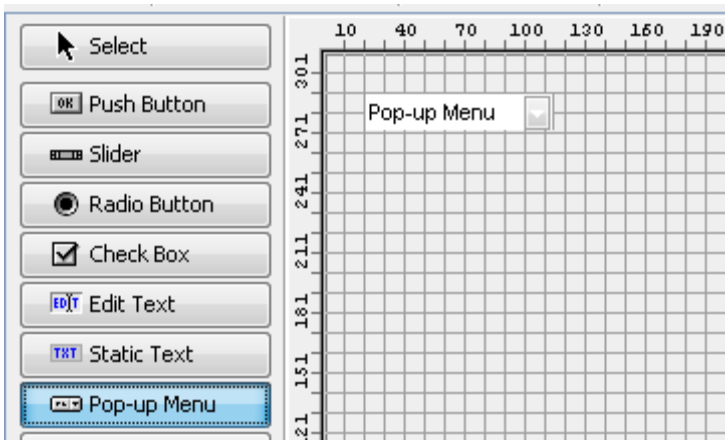
Teraz keď sme zvolili prednastavenú hodnotu v poli Edit Text, môžeme pristúpiť k samotnému spracovaniu. Môžeme nejakému tlačidlu definovať operáciu, aby po jeho stlačení sa zistila hodnota parametra „String“ komponentu „Edit Text“. Potom túto hodnotu prevedieme na číslo a vykonáme nejakú matematickú operáciu (Obr. 22).

```
80      % --- Executes on button press in pushbutton1.
81      function pushbutton1_Callback(hObject, eventdata, handles)
82      % hObject    handle to pushbutton1 (see GCBO)
83      % eventdata  reserved - to be defined in a future version of MATLAB
84      % handles    structure with handles and user data (see GUIDATA)
85
86 -    number = str2num(get(handles.cislo,'String'));
87 -    cislo = number^2
```

Obr. 22 Definovanie jednoduchého skriptu na výpočet druhej mocniny čísla

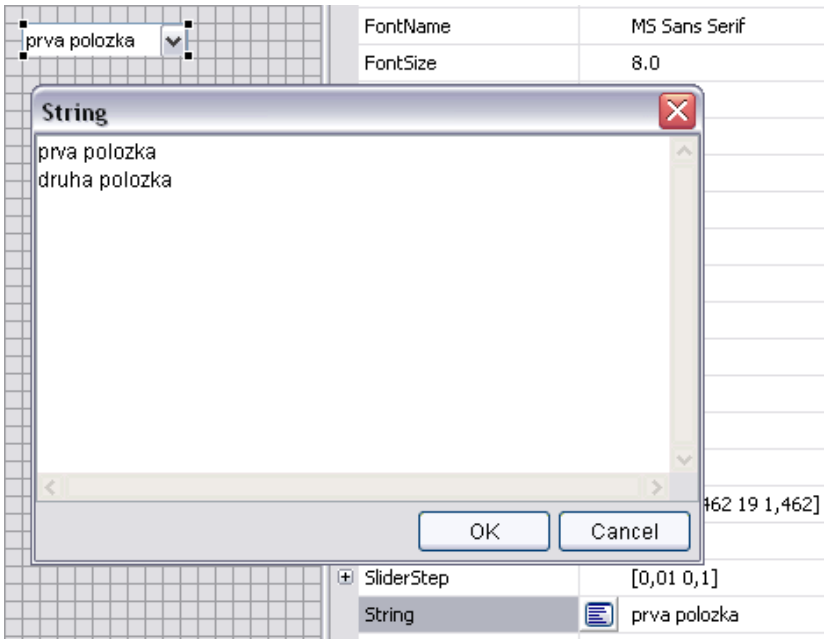
Pop-up Menu (Obr. 23):

Tento komponent sa definuje v prípade, ak chceme vybrať jednu z možností. Takýto prípad sa síce dá realizovať aj pomocou komponentov Check Button, avšak takéto riešenie by mohlo v projekte zaberať veľa miesta ak by bolo treba spraviť výber z väčšieho množstva položiek.



Obr. 23 Vloženie komponentu Pop-up Menu

Jednotlivé položky vkladáme v Property Editore a to konkrétne do parametra „String“ (Obr. 24).



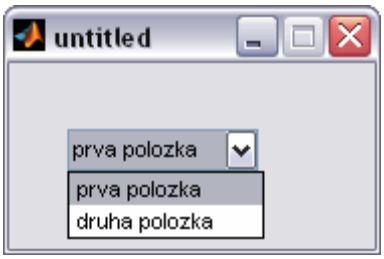
Obr. 24 Vytvorenie ponuky v Pop-up Menu

Pre zistenie ktorá položka zoznamu bola vybraná slúži parameter Value. Potom pre jednotlivé prípady výberu môžeme definovať konkrétne operácie (Obr. 25):

```
129     % --- Executes on selection change in popupmenu1.
130     function popupmenu1_Callback(hObject, eventdata, handles)
131     % hObject    handle to popupmenu1 (see GCBO)
132     % eventdata  reserved - to be defined in a future version of MATLAB
133     % handles    structure with handles and user data (see GUIDATA)
134
135     % Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
136     %        contents{get(hObject,'Value')} returns selected item from popupmenu1
137
138     hodnota = get(hObject,'Value');
139     switch hodnota
140         case 1
141             disp('bola vybrana prva polozka')
142         case 2
143             disp('bola vzbrana druha polozka')
144     end
```

Obr. 25 Definovanie operácií v Pop-up Menu

Ak spustíme projekt, môžeme potom ľubovoľne vyberať buď jednu alebo druhú možnosť a výsledok sa bude zobrazovať v príkazovom okne (Obr. 26)



Obr. 26 Vzhľad komponentu Pop-up Menu po spustení vlastného projektu

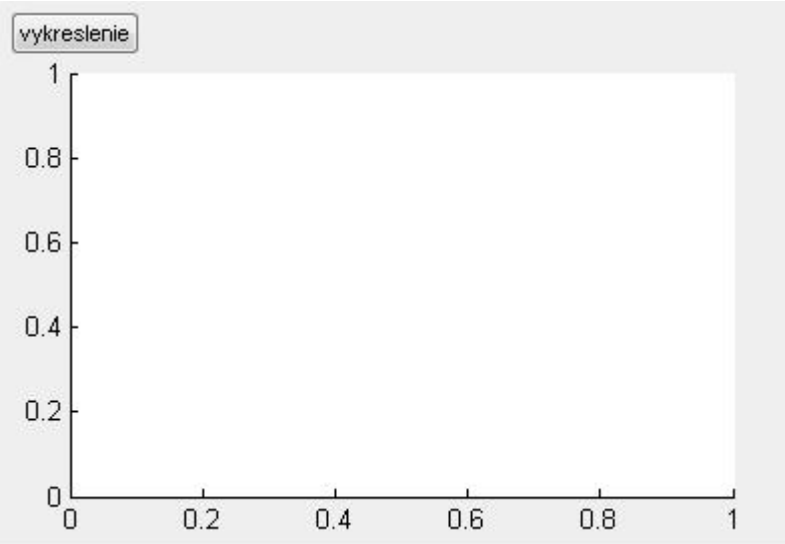
Axes:

Komponent Axes slúži na grafickú interpretáciu získaných výsledkov. Aby sa do tohto komponentu preniesli nejaké dáta, stačí v Callbacku (Obr. 27) príslušného komponentu (Push Button, Toggle Button, atď.) pridať príkaz na vykreslenie:

```
80      % --- Executes on button press in vykresli_graf.
81      function vykresli_graf_Callback(hObject, eventdata, handles)
82      % hObject    handle to vykresli_graf (see GCBO)
83      % eventdata  reserved - to be defined in a future version of MATLAB
84      % handles    structure with handles and user data (see GUIDATA)
85
86 -    surf(handles.axes1,(peaks(35)));
```

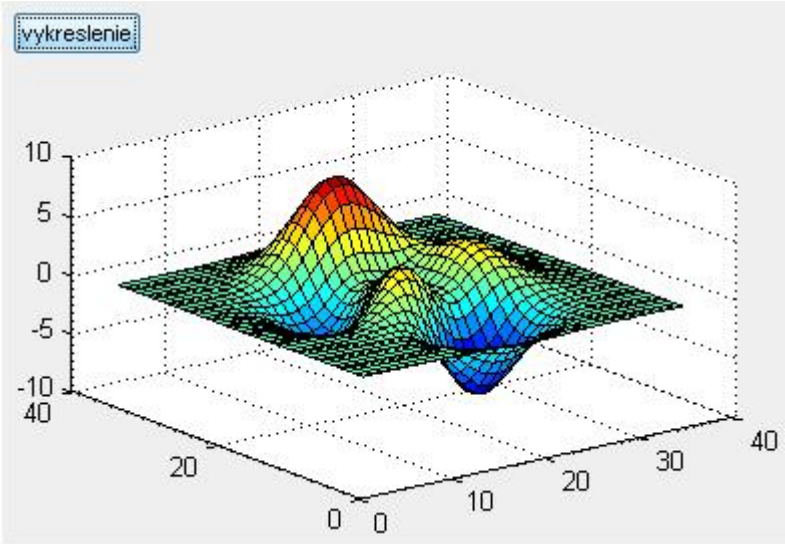
Obr. 27 Príkaz na vykreslenie 3d grafu

Po spustení projektu sa štandardne zobrazuje prázdny graf (Obr. 28). Dá sa však definovať veľkosť mierky osí, zapnutie či vypnutie osí atď.



Obr. 28 Vzhľad komponentu axes po spustení projektu

Stlačením tlačidla (Obr. 29), v Callbacku sme preddefinovali príkaz ktorý pracuje s komponentom axes, sa výsledok vykreslí do grafického poľa tohto komponentu.



Obr. 29 Ukážka činnosti komponentu Axes

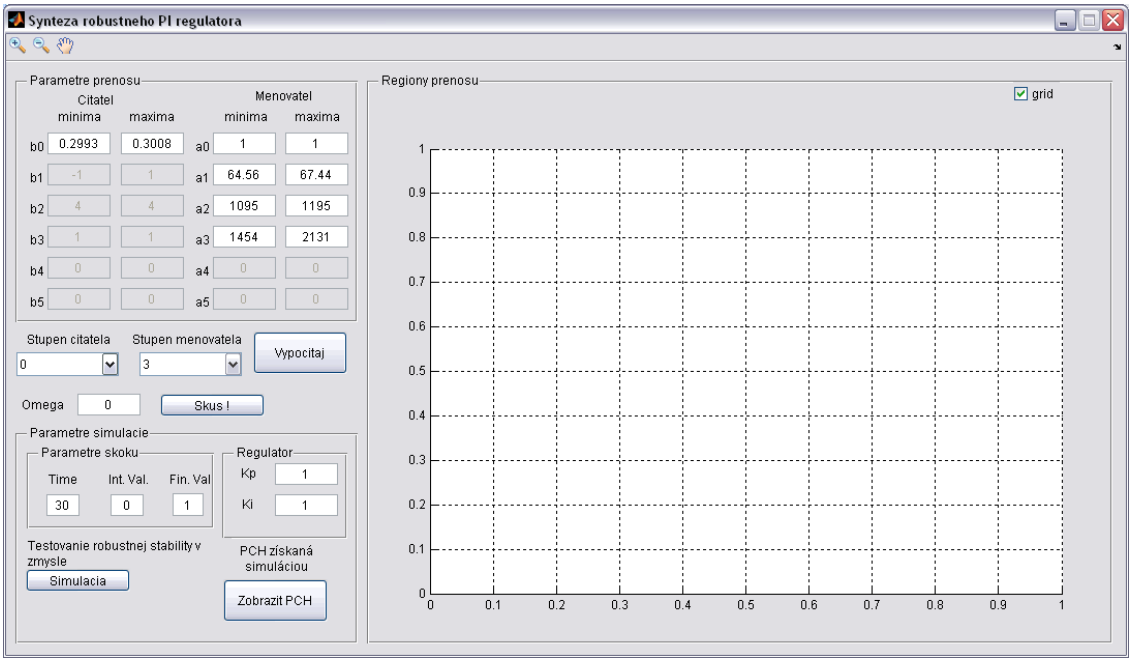
3 PROJEKTY SYNTÉZY P A PI REGULÁTORA V GUIDE

Projekty slúžia na návrh robustných regulátorov, ktoré budú stabilizovať prenos s neurčitosťami. Grafická podoba projektov je do značnej miery totožná, keďže metóda syntézy P regulátora sa opiera o metódu syntézy PI regulátora. Kým v druhom prípade sa do roviny vykresľuje závislosť integračnej zložky od proporcionálnej *kp=f(ki)*, v prvom prípade sa proporcionálna zložka vykresľuje do komplexnej roviny. Z grafu sa odčítava iba reálna časť zosilnenia regulátora.

Oba programy poskytujú tiež možnosť simulačne overiť zvolený regulátor. Zobrazený priebeh simulácie znázorňuje priebeh riadenia uzavretého regulačného obvodu. Tento obvod pozostáva zo zvoleného regulátora, ktorý je zapojený pred nominálny a hraničné prenosy. Spolu s priebehmi riadenia program poskytuje aj údaj o kvalite regulácie a to konkrétne hodnoty absolútnej hodnoty regulačnej odchýlky. Keďže ide o robustné regulátory, testovanie ich stability sa deje v zmysle Charitonovovej vety.

3.1 Návrh robustného PI regulátora

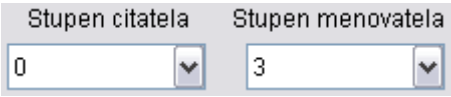
Spustenie projektu sa zaháji príkazom *„navrh“* v príkazovom riadku. Následne sa spustí samotné okno projektu, ktoré je kvôli prehľadnosti rozdelené na viacero častí (Obr. 30).



Obr. 30 Program syntézy robustného PI regulátora

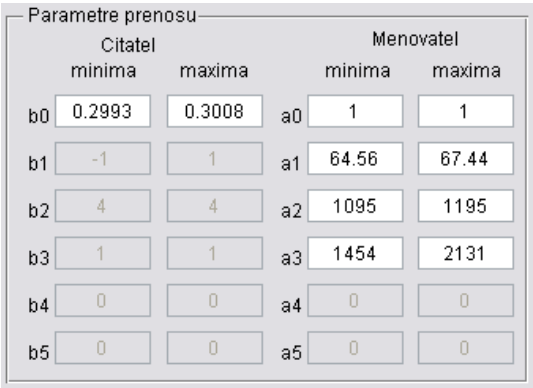
V prvom rade užívateľ zvolí stupeň čitateľa a menovateľa prenosu. Tieto dva komponenty sú ošetrené na prípad, ak by sa omylom zadal stupeň čitateľa väčší ako stupeň menovateľa (Obr. 31).

Metóda, ktorú tento program využíva na syntézu regulátora, nedovoľuje použiť ľubovoľné prenosy a jej použitie je komplikované pri nízkych rádoch polynómov prenosu, viď kapitola 1.2.1. Tento nedostatok je programovo ošetrený formou automatickej korekcie stupňov prenosu v prípade zadania nízkeho rádu. Program po zadaní nesprávneho resp. príliš nízkeho stupňa či už čitateľa, alebo menovateľa automaticky zvýši tento na takú úroveň, aby pri ďalšom výpočte nedošlo k chybám. Zadávanie stupňov prenosu je taktiež ošetrené pre prípad zadania vyššieho stupňa čitateľa ako je stupeň menovateľa.



Obr. 31 Voľba parametrov prenosu

Hlavná časť slúži na zadanie parametrov prenosu a má podobu panelu, v ktorom sú integrované komponenty „Text Edit“. Tieto slúžia na zadanie konkrétnych parametrov (Obr. 32).

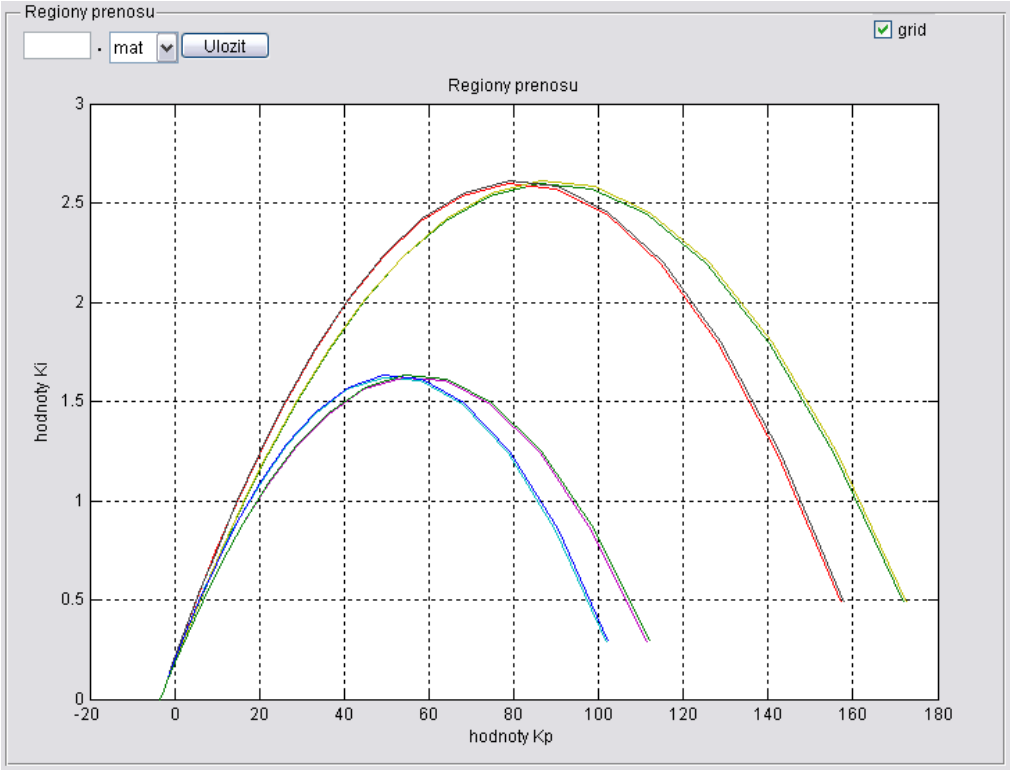


Obr. 32 Panel na zadávanie parametrov prenosu

Všetky textové polia sú ošetrené pre prípad zadania textu namiesto číselného údaju. Tiež sa skúma zadanie nekorektných parametrov ktoré sú v prostredí Matlabu chápané ako určité operátory (napr. , ;). Okrem toho program užívateľa upozorní ak omylom zadá minimálnu hranicu neurčitosti väčšiu ako maximálnu. V prípade ak sa zadá nekorektný parameter, program sám zmení nekorektný parameter na desatinnú bodku. Celý proces kontroly prebieha zvlášť pri každom vkladaní jednotlivých parametrov keďže každý komponent predstavuje

funkciu v skripte projektu (tzv. Callback). To zabezpečí že sa kontrola zadávania údajov vykonáva hneď na pozadí a nespomaľuje prácu.

Po správnom vložení všetkých údajov sa stlačí tlačidlo „*Vypočítaj*“. Výsledkom je zobrazenie grafickej závislosti P (resp. Kp) a I (resp. Ki) od parametra omega v zmysle spracovávanej metódy (Obr. 33).



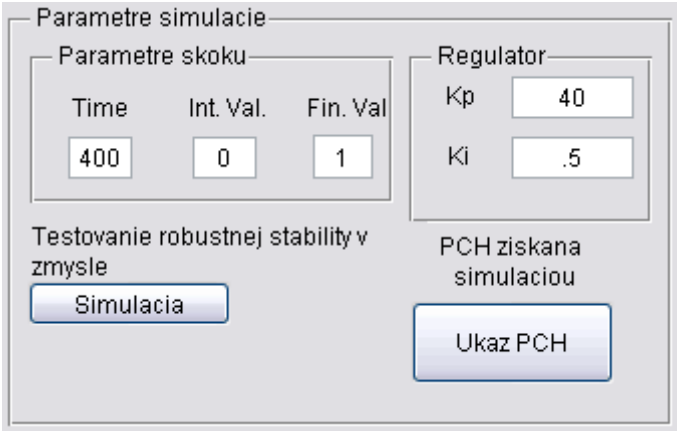
Obr. 33 Grafický výstup výpočtu

V paneli sa zároveň po prebehnutí výpočtového algoritmu zobrazí možnosť uložiť vykreslené oblasti. Užívateľ môže v rámci väčšej flexibility uložiť oblasti ako súbor .mat, do ktorého sa uložia v podobe vektorov všetky hodnoty vykreslených oblastí. Po importovaní premennej z tohto súboru do workspace prostredia Matlab je ďalej možné s týmito údajmi pracovať. Ďalšími možnosťami je uloženie do súboru .fig, ak užívateľ ďalej nechce s údajmi pracovať, ale má záujem upraviť graf podľa vlastných predstáv. Pre ľahšiu interpretáciu výsledkov je možné vytvoriť hotový obrázok vo formáte .jpg, alebo údaje zoradiť do prehľadnej štruktúry v súbore .xml.

V niektorých prípadoch môže nastať situácia že vykreslenie regiónov nie je úplné, teda sa nedá určiť stabilná oblasť. V takomto prípade sa môže zvoliť iná hodnota parametra omega. Program ešte predtým vypočíta aktuálnu hodnotu tohto parametra, ktorá však môže byť ľubovoľne menená. Výsledok sa interpretuje pre všetky regióny (Obr. 34).

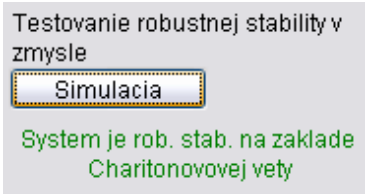
Obr. 34 Časť pre samostatnú voľbu parametra omega

Parametre regulátora sa vyberajú z oblasti, ktorá je prienikom všetkých ostatných, toto samozrejme platí len v prípade ak prenos na ktorý je potrebné navrhnúť regulátor obsahuje neurčitosti. Tieto parametre sa vložia do panela slúžiaceho na testovanie robustnej stability. Robustná stabilita sa testuje na užívateľom zadaný prenos, ktorý je vložený do spätnoväzbovej slučky s PI regulátorom (Obr. 35).



Obr. 35 Časť pre nastavenie simulácie a testovanie robustnej stability

Štandardné údaje potrebné pre simuláciu sú prednastavené na určité hodnoty, avšak tiež je možné ich ľubovoľne meniť. Stlačením tlačidla „*Simulácia*“ sa spustí skript testovania robustnej stability, ktorý (ak prenos obsahuje neurčitosti) vytvorí všetky možné kombinácie vyplývajúce z pôvodného tvaru prenosu. Tieto čiastkové prenosy postupne vkladá do URO a čiastkové charakteristické rovnice slúžia ako základ pre určenie maximálneho a minimálneho polynómu. Tieto sú vstupom do funkcie *kharit* ktorá testuje robustnú stabilitu a jej výstupom sú štyri polynómy, z ktorých každý musí byť stabilný. Hlásenie o robustnej stabilite sa zobrazí v priestore pod tlačidlom (Obr. 36).



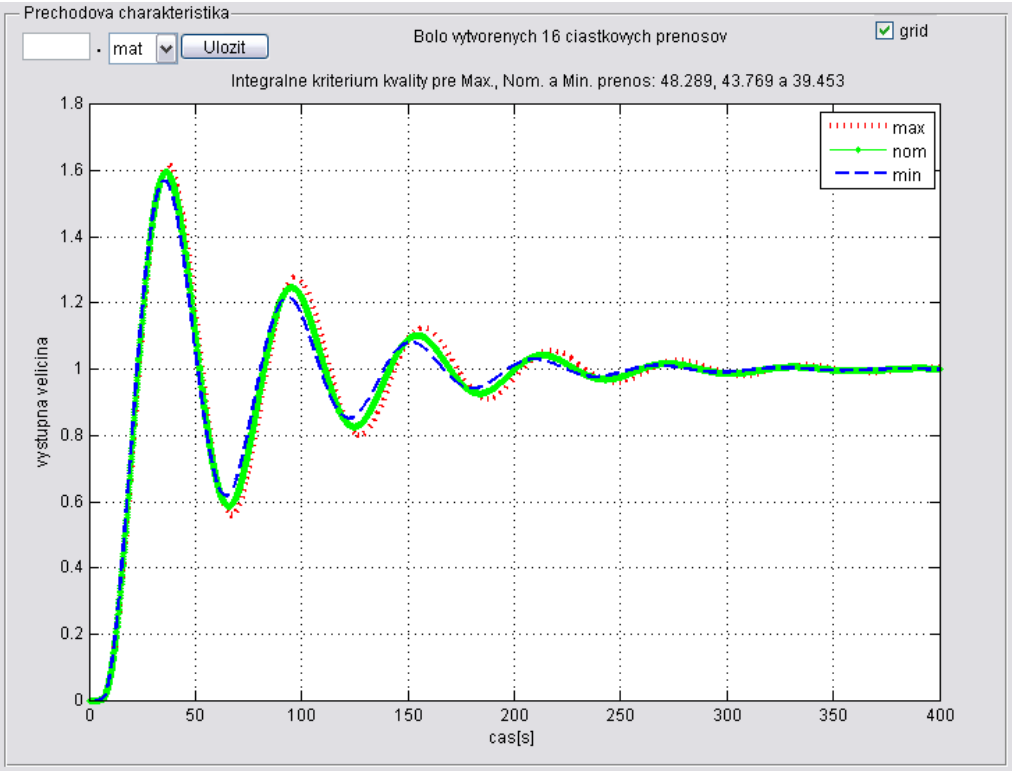
Obr. 36 Hlásenie o robustnej stabilite

Zároveň po otestovaní stability sa vykoná simulácia troch uzavretých regulačných obvodov reprezentujúcich systém v maximálnom, nominálnom a minimálnom stave. Keďže

celé grafické riešenie zobrazujúce regióny prenosu zaberá podstatnú časť projektu, na zobrazenie priebehu simulácie už nie je miesto.

Pre zjednodušenie práce bol skript programu doplnený o funkciu, ktorá automaticky prepína medzi zobrazovanými výsledkami. Ak teda program aktuálne zobrazuje panel s regiónmi prenosu a užívateľ po nastavení parametrov tlačidlom spustí simuláciu, program automaticky zobrazí panel s priebehom simulácie. Opačná akcia sa vykoná ak je zobrazený panel simulácie a užívateľ stlačí tlačidlo na výpočet alebo vykreslenie regiónov prenosu.

Tento problém rieši komponent Toggle Button, ktorého aktivovaním zmizne panel s regiónmi a nahradí ho panel integrujúci výsledok simulácie (Obr. 37). Tento komponent je do projektu osadený kvôli tomu, aby zobrazené grafy mali dostatočnú veľkosť a v nich zobrazované výsledky bolo ľahko interpretovať.



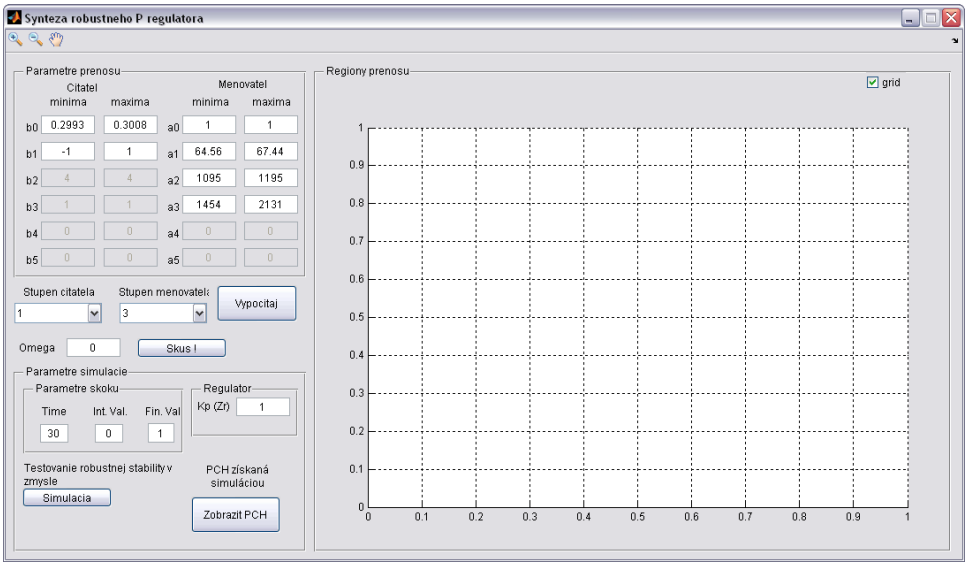
Obr. 37 Panel zobrazujúci priebeh simulácie s výsledkami

V tomto paneli (Obr. 37) sú okrem zobrazenia priebehu simulácie zobrazené aj doplňujúce informácie o počte vytvorených čiastkových prenosov. Toto číslo v podstate hovorí o maximálnom možnom počte vytvorených čiastkových prenosov, teda o všetkých situáciách, do ktorých sa daný systém môže dostať. O kvalite riadenia vypovedá integrálne kritérium absolútnej hodnoty regulačnej odchýlky IAE. Tento údaj je zobrazovaný pre nominálny a pre hraničné prenosy.

Tak ako aj v paneli zobrazujúcom regióny prenosu, tak aj v tomto paneli sa po odsimulovaní zobrazí ponuka na uloženie priebehov simulácie s rovnakými možnosťami ako v paneli pre regióny prenosu.

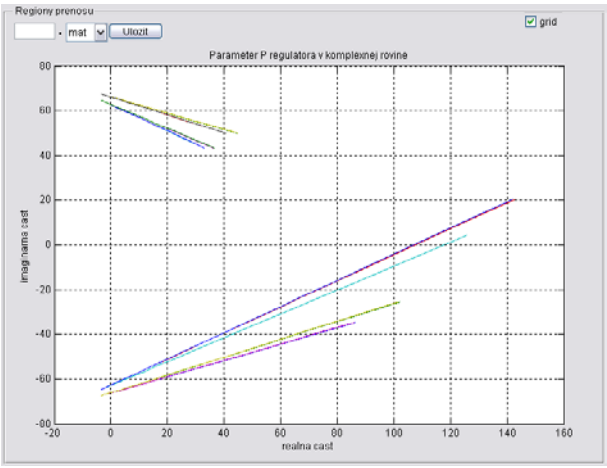
3.2 Návrh robustného P regulátora

Projekt syntézy robustného P regulátora vychádza z predošlého projektu a teda vo väčšej časti je rovnaký (Obr. 38). Iný je len princíp vykreslenia regiónov prenosu.



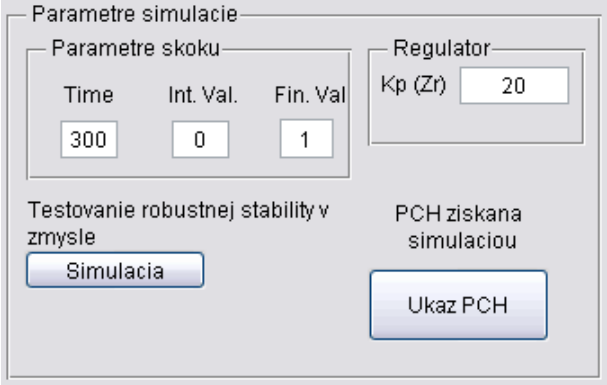
Obr. 38 Projekt syntézy robustného P regulátora

V tomto prípade je syntéza založená na určení zosilnenia regulátora a teda nie je možné vyniesť do roviny závislosť I zložky regulátora od P zložky. Môžeme však do roviny vyniesť závislosť komplexnej zložky zosilnenia P regulátora od jeho reálnej časti (Obr. 39).



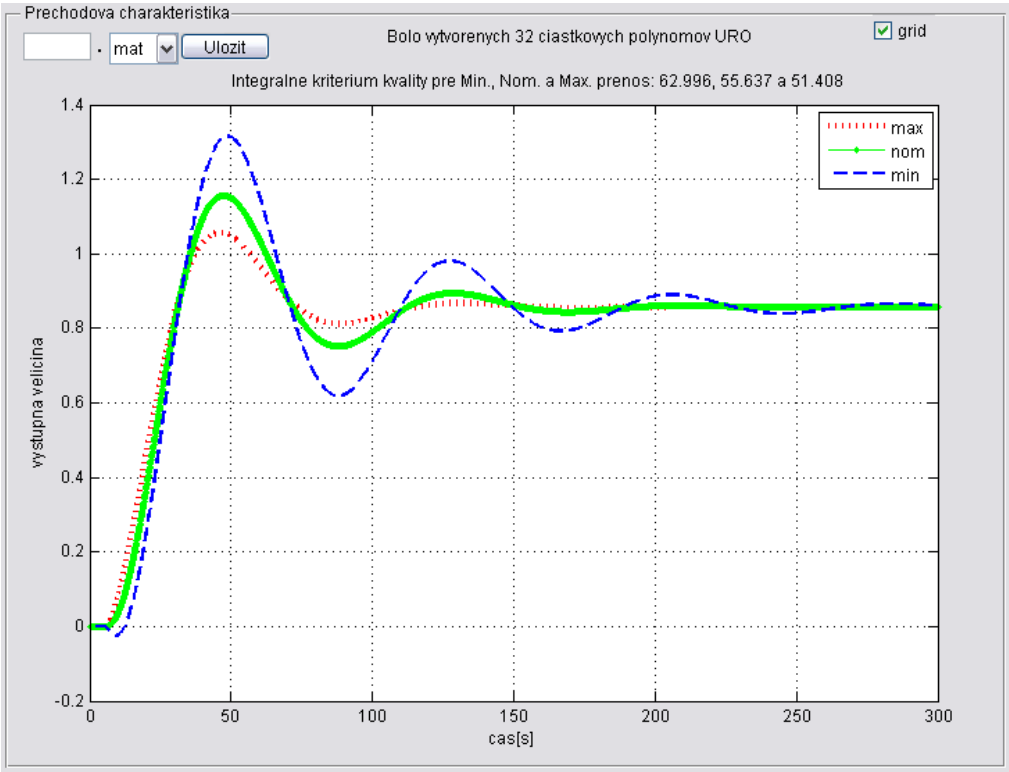
Obr. 39 Grafický výstup výpočtu pri návrhu robustného P regulátora

Tu užívateľ volí zosilnenie z oblasti, ktorá je podmnožinou všetkých ostatných. Za zosilnenie sa pokladá hodnota na reálnej osi. Od tohto momentu je ďalšia časť programu totožná s predošlou. Rovnako sa do simulácie zadá hodnota zosilnenia regulátora (Obr. 40).



Obr. 40 Nastavenie parametrov simulácie

Tlačidlom sa spustí simulácia na rovnakom princípe a na rovnakom princípe sa zobrazí jej priebeh spolu s výsledkami (Obr. 41). Taktiež aj tu je možnosť uložiť všetky získané výsledky do rôznych formátov súborov.

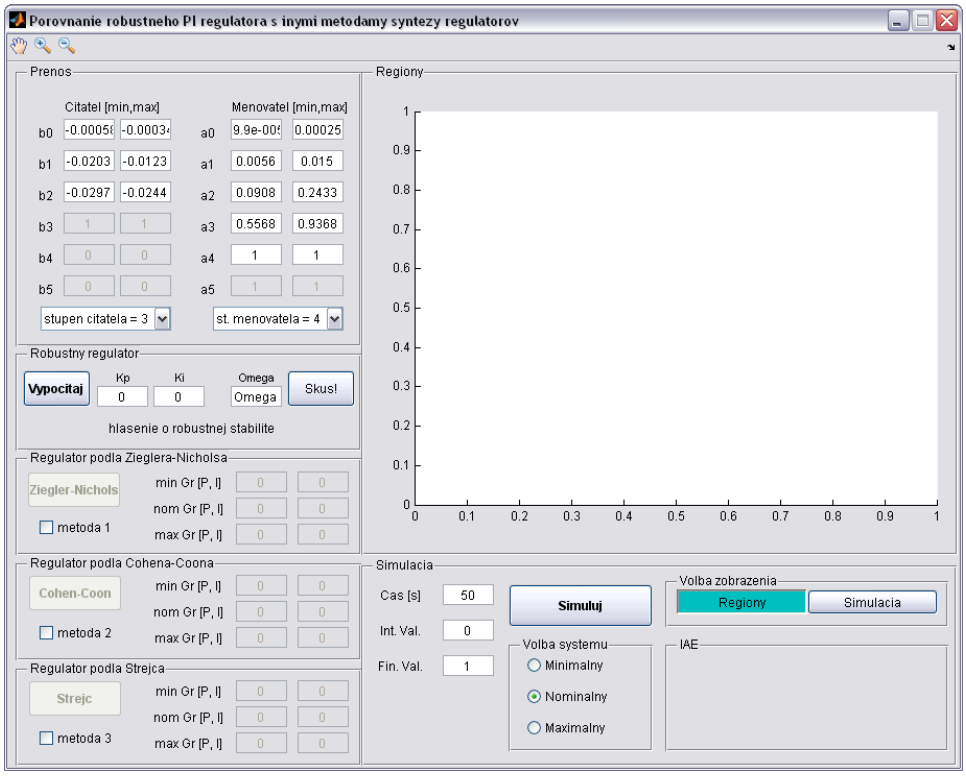


Obr. 41 Riadenie robustným P regulátorom

4 POROVNANIE ROBUSTNÝCH REGULÁTOROV S INÝMI REGULÁTORMI

Potreba porovnať navrhnutý robustný PI regulátor s inými regulátormi mala za následok vytvorenie projektu, v ktorom by bolo možné navrhnúť ako robustný, tak aj iné regulátory a navzájom ich porovnať.

Samotný projekt sa spustí príkazom „projekt“ v príkazovom riadku Matlabu (Obr. 42). Ako vo všetkých projektoch, aj tu sú vopred nastavené hodnoty konkrétneho prenosu. Takto bez väčších ťažkostí sa môže užívateľ intuitívne oboznámiť s činnosťou a ovládaním programu.



Obr. 42 Projekt syntézy a porovnania regulátorov

Návrh robustného regulátora prebieha identicky ako v predchádzajúcich projektoch, selektívnou voľbou parametrov regulátora zo stabilnej oblasti. Následne sa parametre vložia do políčok označujúcich príslušné zložky regulátora a môže sa spustiť simulácia.

Program je obohatený o experimentálne metódy syntézy klasického PI regulátora a to konkrétne o metódy syntézy regulátora podľa Zieglera-Nicholsa, Cohena-Coona a podľa Strejca (Bakošová, a další, 2003) (Obr. 43). Konkrétna metóda návrhu sa aktivuje príslušným checkboxom v bloku prislúchajúcom tejto metóde.

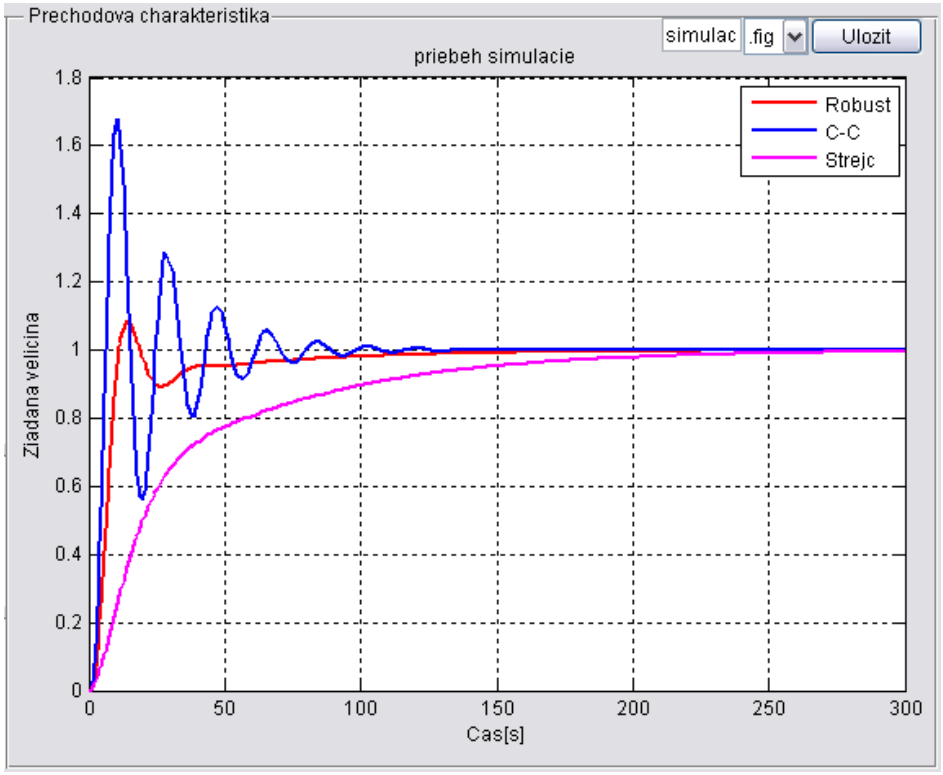
Obr. 43 Bloky rôznych metód syntézy PI regulátora

Všetky spôsoby výpočtu regulátora podľa konkrétnych metód sú založené na rovnakom spôsobe výpočtu, mení sa len vzorec, podľa ktorého vypočítavajú proporcionálnu a integračnú konštantu. Celý návrh je postavený na princípe identifikácie z prechodovej charakteristiky. Cieľom je získať k nominálnemu a hraničným prenosom ich adekvátne prenosy v tvare podľa Strejca. Na začiatku teda konkrétna funkcia zisťuje, či je daný systém stabilný, alebo na hranici stability, aby sa dala spraviť prechodová charakteristika potrebná na identifikáciu. Ak je všetko v poriadku, program pristúpi k funkcii na určenie periodicity systému, aby mohol použiť buď identifikáciu podľa Strejca, alebo identifikáciu kmitavého systému.

Identifikované parametre sa požijú na výpočet parametrov regulátorov a zobrazia sa v príslušných komponentoch v konkrétnom bloku syntézy regulátora(Obr. 44).

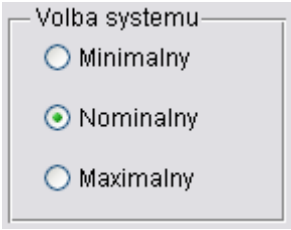
Obr. 44 Vypočítané parametre regulátorov podľa Zieglera-Nicholsa

Po výpočte parametrov môže užívateľ pristúpiť k simulačnému porovnaniu navrhnutých regulátorov (Obr. 45). Spustením simulácie sa tak ako v predchádzajúcich projektoch program sám prepne do bloku zobrazujúceho priebeh simulácie.



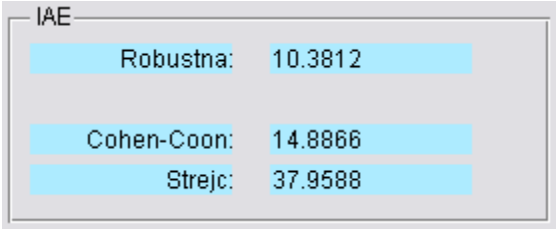
Obr. 45 Priebeh riadenia rôznymi typmi regulátorov

Rozdiel v tomto programe oproti iným je v tom, že zatiaľ čo v programoch slúžiacich len na návrh robustného regulátora sa zobrazovali simulačné priebehy nominálneho a hraničných systémov, tu sa zobrazuje iba jeden. Zobrazenie sa zameriava na vykreslenie priebehov riadenia vybraných metód syntézy regulátora pre konkrétny systém. Užívateľ si však môže voliť medzi zobrazením priebehov riadenia nominálneho, maximálneho, alebo minimálneho systému (Obr. 46).



Obr. 46 Možnosť zobrazenia nominálneho a hraničných systémov

Samotné porovnanie kvality regulácie konkrétnym regulátorom sa nedeje len vizuálnym zhodnotením prechodových charakteristík, ale aj zobrazením číselnej hodnoty integrálneho kritéria absolútnej hodnoty regulačnej odchýlky (Obr. 47). Tá je zisťovaná pre každú metódu návrhu.



Obr. 47 Zobrazenie hodnôt integrálneho kritéria kvality riadenia

Tiež aj v tomto programe nechýba možnosť uloženia všetkých získaných výsledkov do výstupného súboru.

Záver

V tejto práci sú zhrnuté poznatky z teórie robustného riadenia, konkrétne z oblasti systémov s intervalovými neurčitosťami. Jednako je priblížená metóda návrhu robustných PI regulátorov, ktorá je spracovaná do aplikácie slúžiacej na ich syntézu.

V práci som sa snažil priblížiť a čo možno najzrozumiteľnejšie popísať spôsob návrhu aplikácií v prostredí GUIDE. Práca popisuje prostredie a možnosti jeho prispôsobenia, ako aj ovládacie prvky a ich funkcie a spôsoby použitia. Je ozrejmený spôsob vytvárania vlastného projektu, vkladanie a programovanie komponentov a tiež ich vzájomná závislosť. Tým pádom práca môže v budúcnosti poslúžiť ako užitočný návod na vytváranie projektov s rozmanitým charakterom použitia a činnosti.

Aplikácia je ďalej rozšírená o možnosť porovnania užívateľom navrhnutého robustného regulátora s niektorými klasickými typmi regulátorov. Metódy syntézy týchto regulátorov sa opierajú o experimentálne poznatky a teda nie všetky sú vhodné na uspokojivý návrh regulátora.

Používaním aplikácie na rôzne prípady štruktúry prenosovej funkcie reprezentujúcej rôzne technologické procesy sa zistilo, že síce v mnohých prípadoch je robustný regulátor v porovnaní s klasickými lepší, avšak hodnota jeho integrálneho kritéria stúpa s približovaním sa k maximálnym hodnotám parametrov systému. Táto metóda sa osvedčila aj pri syntéze robustného PI regulátora pre nestabilné systémy, alebo systémy na hranici stability, pri ktorých sa klasické metódy syntézy regulátora nedajú použiť.

V prípadoch kedy sa dali navrhnúť klasické regulátory sa tiež vykonala ich robustná stabilita, no s negatívnym výsledkom.

Literatúra

[1] **Bakošová Monika, Fikar Miroslav a Čirka Ľuboš** Základy automatizácie [Kniha]. - Bratislava : STU, 2003. - ISBN 80-27-1831-9.

[2] **Hurák Zdeňek** Robustní řízení [Kniha]. - Praha : Fakulta elektrotechnická ČVUT, 2005.

[3] **MathWorks The** Creating Graphical User Inrterfaces [Kniha]. - 2009. - http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/buildgui.pdf.

[4] **Matušů Radek** Algoritmy robustního řízení pro systémy s parametrickými neurčitostmi [Kniha]. - Zlín : IŘPI FI UTB, 2005.

[5] **Tan Nusret a Kaya Ibrahim** Computation of stabilizing PI controllers for interval systems [Kniha]. - Rhodes : Inonu University, Engineering Faculty, Dept. of Electrical and Electronics Engineering, 2003.