СЪЗДАВАНЕ НА СИСТЕМИ ЗА ОБУЧЕНИЕ В АВТОМАТИКАТА: *КОМПЮТЪРНО ПРОЕКТИРАНЕ НА ПИД УПРАВЛЕНИЕ*

CREATION OF LEARNING SYSTEMS IN AUTOMATICS: *PID CONTROL COMPUTER DESIGN*

Е.Гарипов, Л.Томов

ТУ – София, кат. Системи и управление, ФА, email: emgar@tu-sofia.bg

1. ВЪВЕДЕНИЕ

Програмният продукт MATLAB се е утвърдил като подходяща за проектиране на системи за управление среда, която предлага два известни подхода за работа. При първия се активират използваните конкретни функции според дефинирания им синтаксис и се получават резултати, които потребителят преобразува в подходяща по формат информация за използване на следващите функции и т.н. докато се реши определен проблем. Очевидна е възможната свобода на изследователската дейност, но сложността на тези действия изискват детайлно познаване на възможностите на библиотечните функции на програмната среда. Дори когато трябва да се прилагат добре познати алгоритми, голяма част от времето на потребителя се губи за обработване и прехвърляне на информация между използваните функции. Другият подход представлява използване на завършен програмен продукт, който автоматизира решаването на типови проектантски задачи и спестява времето на потребителя за сметка на вложения труд от системния програмист при изготвянето на програмната система. В този случай ползвателят получава сравнително ограничени възможности в рамките на заложените опции на системата, но може да се концентрира върху същинската работа – синтез на управляващи устройства и анализ на САУ.

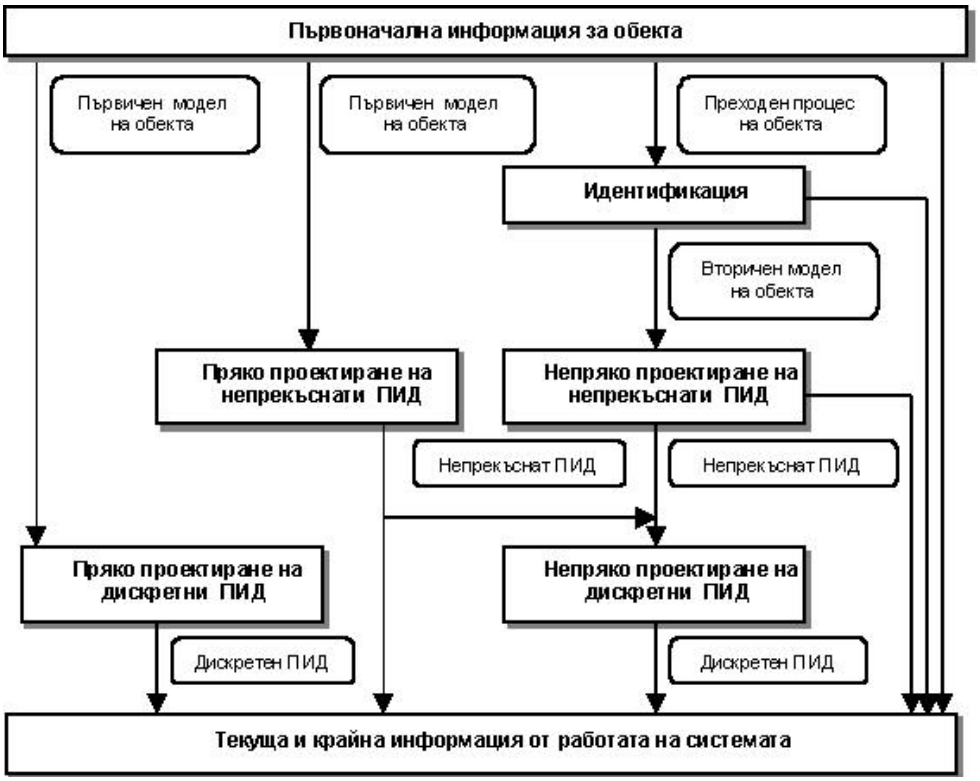
Автоматизираното проектиране е доказало своите предимства в практиката и навлиза все повече във всички области на инженерните науки, затова в представения материал авторите представят един обобщен алгоритъм за създаване на такава система. Той е заложен в основата на написаната от тях програмна система за проектиране на ПИД регулатори “PIDBul”, претърпяла две модификации през 2003 и 2004 г. (Том:Диплом.работи). Тази постоянно развивана обучаваща система обхваща основни елементи от теоретичния материал и свързаните с него програмни продукти, които се създават в част от лабораторни упражнения, курсови работи и самостоятелна дейност в процеса на обучение по дисциплината "Цифрови системи за управление" от бакалавърския курс в катедра “Системи и управление” на ТУ-София [Гар:ЦСУ,2004].

2. ИЗГРАЖДАНЕ НА АВТОМАТИЗИРАНА СИСТЕМА В MATLAB “PIDBul”

Създаването на програмна система трябва да следва определена последователност от действия, съобразени със сценария на решаваните задачи и предлаганите възможности от MATLAB/SIMULINK да го реализират.

Стъпка 1. *Дефиниране на целите и задачите на системата*.

Началният етап обхваща определяне на всички задачи, които потребителят би искал да решава чрез проектираната автоматизираната система. Така се създава един възможен набор от функции, които искаме тя да осъществява, и организация на изпълнението им в рамките на една предварителна функционална схема. Програмната система PIDBul е замислена като удобно за потребителя средство за настройване на непрекъснати и дискретни ПИД регулатори посредством преки (ръчни, с информация от автоколебателен режим, оптимизационни методи), и непреки (с информация от идентификация) методи. Реализираната функционална схема е показана на фиг. 1.



Фигура 1. Функционална схема на “PIDBul”

В табл. 1 са описани основните задачи, които решава системата.

Таблица 1

1. Първично описание на обекта на управление
1.1. Непрекъсната предавателна функция от произволен ред
1.2. Преходен процес
2. Апроксимиран модел на обекта чрез идентификация по преходен процес
2.1. Двупараметрична предавателна функция
2.2. Трипараметрична предавателна функция
3. Настройване на непрекъснати ПИД регулатори.

3.1. Пряко настройване
3.1.1. Изчисляване на критичните параметри на автоколебателна САУ
3.1.2. Преки методи
3.1.2.1. Ziegler-Nichols 2
3.1.2.2. Модифициран Ziegler-Nichols 2
3.1.2.3. Astrom-Hagglund: kappa (M=1.4 / 2.0)
3.1.2.4. Hang-Astrom-Ho
3.1.2.5. Авторски методи
3.1.3. Настройване чрез оптимизация
3.1.3.1. Начални условия
3.1.3.2. Схема за формиране на показателя на качеството
3.1.3.2.1. Настройваема САУ
3.1.3.2.2. Настройваема и еталонна САУ
3.1.3.3. Критерии
3.1.3.3.1. IAE
3.1.3.3.2. ITAE
3.1.3.3.3. ISE
3.1.3.3.4. ITSE
3.1.3.3.5. Комплексни критерии
3.1.3.3.6. Авторски критерии
3.1.3.4. Оптимизация без ограничения
3.1.3.4.1. Симплекс-метод
3.1.3.4.2. Нелинейни най-малки квадрати
3.1.3.5. Оптимизация с ограничения
3.1.3.5.1. Ограничителни условия
3.2. Непряко настройване
3.2.1. Ziegler-Nichols 1
3.2.2. Cohen-Coon
3.2.3. Chien-Hrones-Reswick (задание, 0 / 20%)
3.2.4. Chien-Hrones-Reswick (товар, 0 / 20%)
3.2.5. Astrom-Hagglund: tau) (M=1.4 / 2.0)
3.2.6. Авторски методи
3.3. Ръчно донастройване
3.4. Реакция на товарно смущение
4. Настройване на дискретни ПИД регулатори
4.1. Непряко настройване
4.2. Пряко настройване
4.3. Ръчно донастройване
4.4. Реакция на товарно смущение

Стъпка 2. *Дефиниране на информационните потоци*.

(а) Входно-изходни потоци.

Основната задача е да се дефинират входно-изходните потоци от количествени данни, както за цялата система, така и за всяка нейна функция поотделно. За “PIDBul” главният входен поток е съпътстващата информация за обекта (математически модел под формата, напр., на линейна непрекъсната предавателна функция, или съответна динамична характеристика като експериментално измерен преходен процес, и т.н. Главният изходен поток е информация за типа и стойностите на параметрите на настроения регулатор и неговия ефект върху обекта при симулиране на затворената система на управление посредством визуализиране на управляващия и управлявания сигнал. За всяка самостоятелна функция, която системата използва за решаване на конкретна подзадача, се извършва аналогично дефиниране, докато се изгради в пълен обем информационната структура на системата.

(б) Статична базова информация.

Това е необходимата информация, с която се доуточнява конкретното действие на всяка функция. Чрез определени параметри се регламентира начинът за обработване на входните потоци и възможните начини за трансформирането им в изходни. Напр., указания за избор на конкретен метод, форма и параметри на използван модел, дори стойностите на вектора на цветовете при избор на външното оформяне на програмата.

Стъпка 3. *Дефиниране на принципи за изграждане на системата*.

Аналогични системи се изграждат върху следните четири основни принципа: принцип на статичността, принцип на наследствеността, принцип на преформатирането на информацията и принцип на йерархичната проверка за коректност.

(а) Принцип на статичността

Основните задачи (тялото) на една програмна система (програмата) и вариантите (разклоненията) за тяхното решаване са статични (неизменни) в процеса на функционирането й, т.е. не са разрешени произволни комбинации от подзадачи извън предварително заложените. Предлага се едно основно меню, като изборът на подзадача (етап на задача, опция) от него води до отваряне на зададени списъци с различни опции в допълнително меню. При тази структура на програмата отделните етапи на задачите се оформят чрез съвкупност от самостоятелни модули, а тяхното изпълнение е възможно след препращане на информацията от по-високо към по-ниско йерархично ниво на опциите.

(б) Принцип на наследствеността

Състои се в предаване на информацията от един клон на тялото на програмата, както към неговите преки разклонения, така и към всички други клонове и техни разклонения, които се нуждаят от нея. Това означава, че данните, получени от модула “Първоначална информация за обекта” (фиг. 1), се запазват и са достъпни за всички онаследяващи го модули, докато информацията от модула “Идентификация” се наследява автоматично само от модула “Непряко проектиране на непрекъснат ПИД”. Предаването на информацията върви едновременно по вертикален и по хоризонтален път, а механизмът на запазване и използване на наследяваната информация е напълно автоматичен.

(в) Принцип на преформатиране на информацията

Състои се в това, че при въвеждане на нов текущ обект (модел на изследване), новата информация се презаписва върху старата, всички стари данни стават невалидни, а всички флагове за управление на програмната система се нулират, т.е. цялата програмна система се рестартира.

(г) Принцип на йерархична проверка за коректност

Той означава, че не се налага да бъдат проверявани всички необходими условия за функциониране на модули от даден клон от програмата, защото тези качества или свойства се наследяват йерархично от по-горни клонове. Така например, на определено ниво в програмната система се извършва проверка за устойчивост на обекта на управление, преди да бъде допуснат потребителят да продължи работата си по-надолу по тялото и разклоненията на програмата. Проверките се извършват не само по вертикален, но и по хоризонтален път. Така например, модулът "Идентификация" проверява дали обектът е устойчив, преди да допусне потребителя да слезе по-надолу по тялото и разклоненията на програмата. Затова по-долните клонове не проверяват това условие, а го наследяват йерархично. Модулът 3.1.3 "Настройване чрез оптимизация" съдържа субмодулите "Оптимизация без ограничения" и "Оптимизация с ограничения" на едно хоризонтално ниво, които наследяват избора на начални условия и т.н.

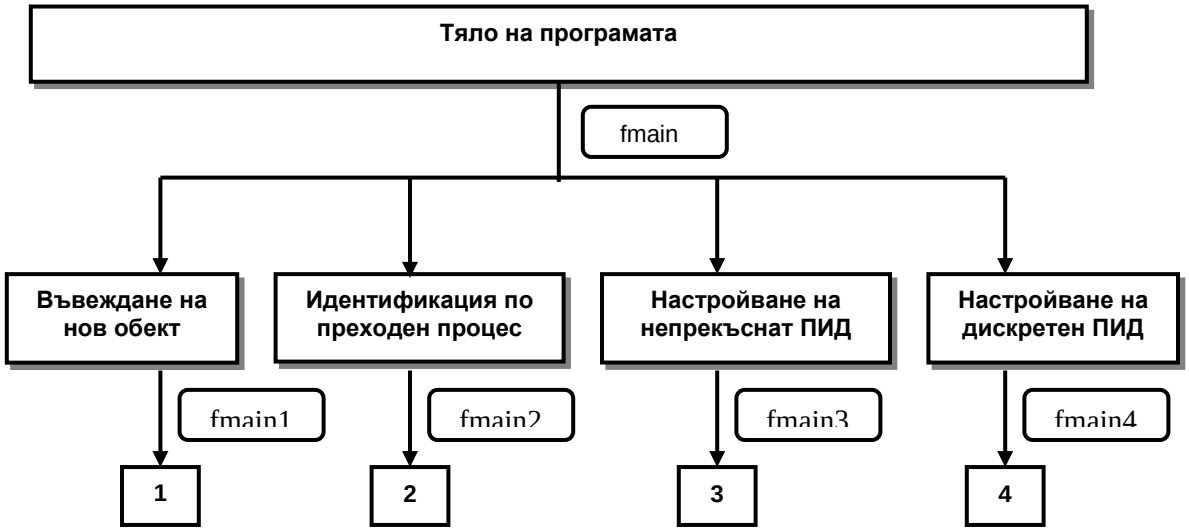
Стъпка 4. Дефиниране на структура на програмната система.

(а) Функционална структура

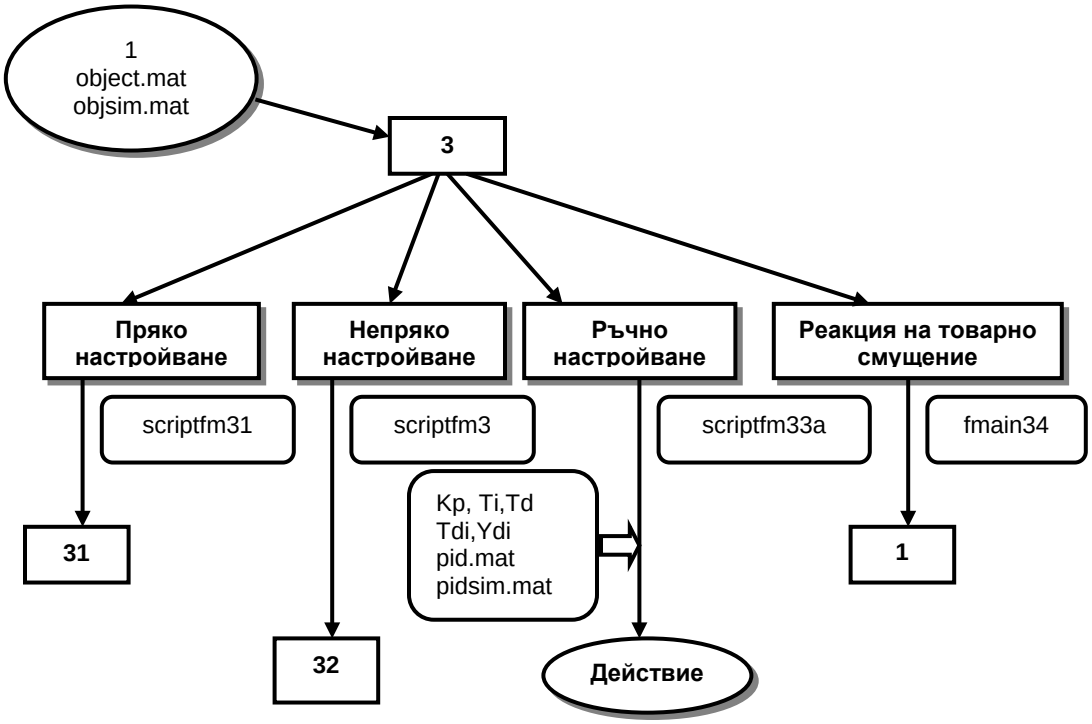
Дефинирането на структура е процес, който стъпва върху функционалната схема на системата, върху принципите на изграждане на системата и върху избраното съотношение на интегро-диференциален баланс на функционалната структура. Какво означава интегро-диференциален баланс? Една програмна система се състои от компютърно оформление (дизайн, маска) на връзката с потребителя и мрежа от функции. Съществуват два подхода за проектиране на тази мрежа: при интегралния подход се отива към окрупняване на функциите и файловете, докато при диференциалния посоката е към строгото им индивидуализиране. Именно балансът между двата подхода определя степента на окрупненост на функционалната структура на програмната система. При "PIDBul" е заложен повече диферециалният принцип за сметка на интегралния. Например, множеството от методи за непряка настройка на регулатора, които използват с информация за модела, получен от идентификацията на обект, са реализирани чрез отделни функции със специфични входни параметри. Докато при един интегралния подход всички тези методи можеха да се реализират с една функция, а чрез допълнителен параметър за избор на конкретен метод се утчняват използваните входни аргументи. Въпросът за окрупняването или раздробяване на функциите следва да се реши от проектантите в зависимост от задачите, които се поставят на системата, и удобството за работа с нея.

След като се реши какъв трябва да е интегро-диференциалния баланс на функционалната структура, трябва да се изгради дърво на задачите. То представлява структурна схема на взаимните връзките между отделните нейни задачи и подзадачи. На фиг.2 и 3 са

показани части от дървото на “PIDBul” и *m*–файловете на MATLAB, чрез които се осъществяват връзките с възможните преходи към низходящите клони.



Фигура 2. “PIDBul” и главните задачи (опции) в нея



Фигура 3. Настройване на непрекъснат ПИД

(б) Информационна структура

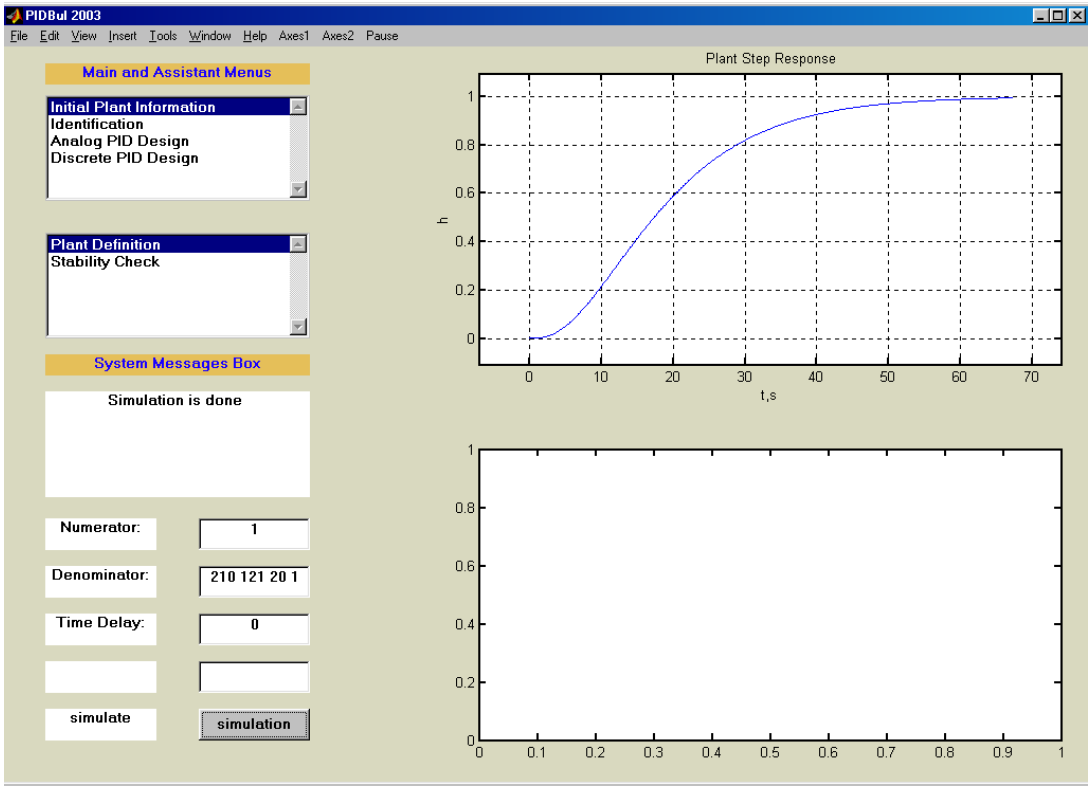
Функционалната структура на програмната система се съпровожда от носители на прехвърляната между отделните елементи информация. Нейната организация може да се осъществи по различни начини. Най-често се осигурява достъпност на всички функции до цялата информация в работното пространство на MATLAB. Те могат да ползват и обновяват нейното съдържание, при условие, че предварително се дефинират съответни глобални променливи на програмната система. Възможен е подход, при който цялото текущо съдържание на работното пространство се съхранява в един общ *mat*-файл, който постоянно се изменя след действието на свързания с функционалния блок *script*-файл. Остава и реализацията на обмен на данни като входно-изходни аргументи на поредни функции.

Стъпка 5. *Проектиране и изграждане на външно оформление на системата*.

(а) Структура на интерфейса на програмата с потребителя

Връзката с потребителя се оформя така, че тя да осигурява последователен и логичен достъп до задачите, заложени в структурата на програмната система. За предпочитане е структурата да не е прекалено усложнена, за да не обърква потребителя с множеството подусловия и конкретни опции. Той трябва да е в състояние да я разучи в максимално кратък срок и търсенето на варианти на решение на задача да заема пренебрежимо малко време спрямо времето на същинската изследователска работа. Необходимо е, едновременно с въвеждането на данни от потребителя чрез клавиатурата, да се извършва задължителна проверка за тяхната достоверност и приложимост в конкретната стъпка на решаване на задачата.

Проектантът трябва да реши формата на диалога на програмната система с потребителя – брой на прозорците, съдържание на менютата, обратна връзка по отношение на данните, качествено изобразяване на графичната информация и т.н. “PIDBul” е реализирана с един единствен прозорец и падащи менюта, както може да се види на фиг.2.5. Разработва се нова версия с допълнителни прозорци, които придават по-голяма самостоятелност на отделни елементи от програмната система.



Фигура 2.5. Общ вид на програмата PIDBul

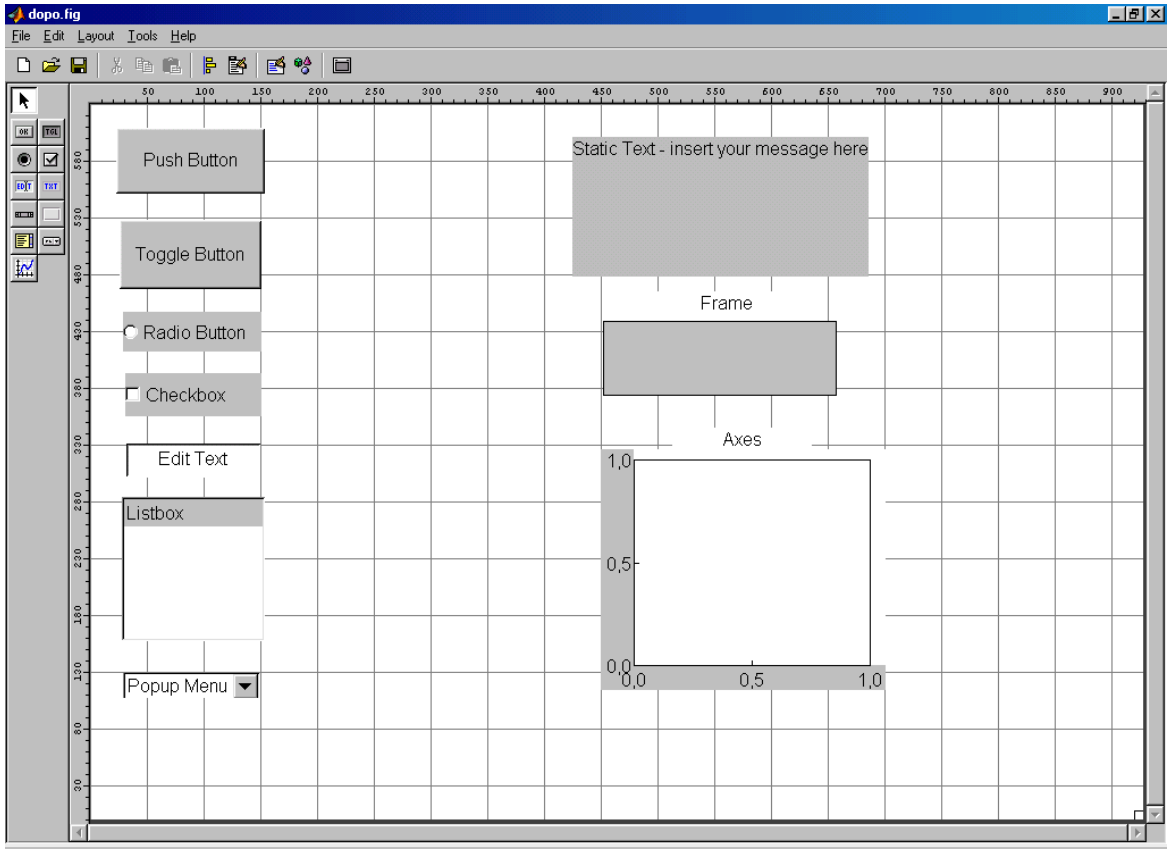
(б) Избор на външен вид на програмната система

Тук въпросът е естетически, поради което е в област, в която други специалисти могат да направят своите препоръки. Авторите могат да допълнят, че подборът на цветовете, шрифтовете, размери на рамки, удебеляване на линии и т.н. трябва да се извърши и от практическа гледна точка, за да не се уморяват очите на потребителите и/или да не се предизвикват неприятни емоции у тях.

Стъпка 6. Програмиране на програмната система.

Качественото оформяне на системата и реализирането на всички идеи на авторите изискват да се познава на добро ниво синтаксиса на езика, на който се пише програмното осигуряване. Когато програмата се реализира в MATLAB, наред с естественото изискване разработчиците да ползват стандартните функции на тази програмна среда, стои необходимостта те да владеят командите от ***uitools*** (user interface tools) и/или да използват програмния модул на MATLAB за автоматизирано проектиране на интефейс – ***Guide***, чиито възможности за показани на фиг.2.6. Кодът на така оформения екран за интерфейс с потребителя, генериран от MATLAB, е показан в табл.1.

.



Фигура 2.6. Възможности на програмния модул ***Guide***

Същността на модула се представя от ограничен набор от **елементи за взаимодействие** на потребителя с програмата – менюта, бутони и др. средства за интерфейс, от една страна, и **механизъм за обратна връзка *CallBack*** за съответна реакция на програмата при реализиран интерфейс. Всеки елемент за взаимодействие съдържа указател (***handle***), който може програмно да се променя чрез функцията ***set*** и по този начин да се задават нови свойства на елементите, включително и да се задават различни обратни връзки (реакции). По този начин едно средство за интерфейс може да се използва многократно в различни части на програмната система, без да възникват конфликти.***CallBack*** представлява символен низ, който се изпълнява (изчислява от ***feval***), когато даденото меню или бутон се задейства. При натискането на бутон низът предава на програмната среда указание за пренасочване на действието на програмата. Например, ако това е име на текстови файл (скрипт), следва извършване на пряко програмно действие, ако това е име на функция, следва активирането й с аргумент, съответстващ на стойността на елемента (бутона). Когато елементът за взаимодействие е меню, неговата стойност показва коя поредна опция е избрана. В зависимост от това, извиканата чрез ***CallBack*** функция реагира съответно.

При програмната реализация на системата се изгражда един описателен текстови (скриптов) файл, напр., ***main.m***, който представя външното оформяне на програмата и вгражда първоначалните обратни връзки на заложените средства на интерфейса –

бутони, менюта и пр. В този файл трябва да се декларират и всички глобални променливи (особено тези, които участват във входно-изходните потоци). Не е необходимо вътрешни за отделните функции променливи да се декларират по същия начин.

При изпълняване на този файл се активира програмната система с нейните първоначални варианти за решаване на конкретни подзадачи. При избор на някоя опция се задейства обратна връзка, която или предизвика действие на програмата (с или без допълнително въвеждане на данни от потребителя), или осъществява пренасочване на потока на решението към друга опция, като променя обратната връзка на някой графичен обект (средство на интерфейса).

3. ХАРАКТЕРИСТИКИ НА ПРОГРАМНАТА СИСТЕМА

Прилаганите методики за настройване на П, ПИ, ПД и ПИД регулатори са описани в [3], като някои авторски идеи са разгледани в [5]. При оптималното настройване на регулаторите се използват качествените функции за безусловна и условна оптимизация от Optimization Toolbox на MATLAB [4].

Управлението на програмната система се извършва от три вида функции:
- ♦ Организиращи, т.е. функции, които подменят менютата с нови опции и осигуряват достъп до всички елементи на системата;
- ♦ Инициализиращи, т.е. функции, които реализират диалогов режим с потребителя и подготвят изпълнението на заявена процедура от програми;
- ♦ Обслужващи, т.е. функции, които изчисляват параметри, преобразуват информация, симулират системи и визуализират процеси.

Общият брой MATLAB функции и скриптори (описания на оператори, които не са оформени като функции) надхвърля 120. От съществено значение за решаваните задачи в областта представляват обслужващите функции на PIDBul2003 за реализиране на различни методи за идентификация и настройване на непрекъснати и дискретни ПИД регулатори. Тук се включат и програмните средства за изчисляване на крайното време за симулиране на САУ, формиране на ограниченията в САУ, изчисляване на различни показатели на качеството за оптимизационни процедури, контрол върху въвеждане на данни от потребителя, коригиране на изображения на процеси, водене на дневник на изследванията и проиграване на запомнени резултати, копиране на графики и т.н.

Системата разполага с богат инструментариум за **реализиране на учебно-изследователска дейност** по отделните задачи и **удобен (дружелюбен) интерфейс** за представяне на графичната част на резултатите. Тъй като всички процедури са напълно или почти напълно **автоматизирани**, потребителят може само с мишката да извърши множество процедури, като тези, които изискват допълнително въвеждане на информация от негова страна, не му създават излишни проблеми. В рамките на минути потребителят може да разучи всички възможности на програмата. Но часове

и дни обаче може да не му стигнат, за да изследва всички възможни комбинации от методи, критерии, ограничения и начални условия. Програмата пази в специално организирани **информационни файлове** с регламентирани имена и разширение .mat всяка важна предварителна, текуща и крайна информация от работата си, която препредава автоматично от модул на модул.

Освен изследователски характер системата за автоматизирано обучение има и полезна **педагогическа и методологична насоченост** – тя подсказва как да се използват работоспособни алгоритми при решаване на задачи от този клас и помага на студентите да концентрират усилията си в процеса на самостоятелна работа. По този начин се **интензифицира** учебно-изследователската дейност, като акцентът на работа със студентите се пренася от програмиране на близки по съдържание MATLAB функции в тяхното активно усъвършенстване и използване чрез удобни стандартни форми, каквито предоставя на разработчика графичният интерфейс с потребителя GUI в MATLAB.

Важно качество на разработената програмна система е **относително лесното разширяване на нейните функционални възможности** и подмяната на отделни модули от програмната библиотека с по-усъвършенствани. В тази първа версия системата не е реализирана чрез изпълним код, а съдържа програмни процедури от функционален тип (функции) и от описателен тип (скриптове). Затова системата може да се комплектова и с документация за нейното модифициране. Отвореният й стил на представяне позволява разпространяваната в основния си вид програма да бъде разширявана и подобрявана от потребителя. Това не е труден процес, но при добавяне на нови опции потребителят трябва да се съобразява със заложените основни задачи на програмата и техните разклонения, които образуват дървото на системата, и с наследяваната информация от по – горно в по – долно ниво. За модификациите той трябва да знае флаговата система, системата за запазване и обмен на информация между клоновете на програмата, системата от изпълними и обслужващи файлове и програми.

Подробна спомагателна информация съпътства изследователската дейност и при необходимост съветва на всяка стъпка потребителите с различно ниво на знания в областта при избор на конкретен вариант на работа.

4. ПРИМЕРИ ЗА ИЗПОЛЗВАНЕ НА PIDBUL

Множество примери са изследвани в [3] и [4], по – специално, изследване на авторски методи за настройване на непрекъснати ПИД регулатори, авторски показатели на качеството за оптимално настройване. Задачата за разработване на универсален цифорв ПИД регулатор с обобщена дискретизация може да се види в [6]. Критериите за избор на такт на дискретизация на така разработения регулатор са изследвани в [7]. Подходите за оптимално настройване на цифровите ПИД регулатори (настройване включително на такта и методите на дискретизация) са разгледани в [8].

5. ЗАКЛЮЧЕНИЕ

В съвременното инженерно обучение автоматизираните програми по проектиране ще се развиват и прилагат поради очевидните предимства, които осигуряват – възможност за облекчаване на труда на потребителя, разширяване на неговите познания в съответната област, създаване на интуитивни познания и практически умения, както и възможност за анализ и сравнение на приложимостта на различните методики при различни условия на работа.

ЛИТЕРАТУРА

1. Astrom, K. and T. Hagglund. *PID Controllers (2^{nd} ed.)*. Instrument Society of America, 1995.
2. Гарипов, Е. Цифрови системи за управление: Част I – Цифрови ПИД регулатори, ТУ-София, 2004
3. Томов, Л. Програмна система за проектиране на аналогови/цифрови ПИД регулатори. Бакалавърска дипломна работа. ТУ-София, 2003.
4. Томов, Л. Изследване на универсален цифров ПИД регулатор. Магистърска дипломна работа. ТУ-София, 2005.
5. MathWorks. MATLAB, Realise 12 (Лицензирано копие, ТУ-София).
6. Гарипов, Е.,Л.Томов, Проектиране и реализиране на универсален цифров ПИД регулатор с обобщена дискретизация, Годишник на ТУ-София, *Автоматика и информатика*, кн. 1, 2004, 25-32..
7. Томов, Л,. Е. Гарипов .Избор на такт на дискретизация в цифрови ПИД регулатори, Созопол, 2005.
8. Томов, Л,. Е. Гарипов. Оптимално настройване на цифрови ПИД регулатори, международна конференция Автоматика и информатика, София, България, стр.251-254.

Забележки:

1.Твърдение на стр.9 ln_9, col_49: “Всеки елемент за взаимодействие съдържа указател (***handle***), който може програмно да се променя чрез функцията ***set”.*** По – правилно в да се каже “Всеки елемент за взаимодействие съдържа указател (***handle***), чрез който може програмно да се променят свойствата му, използвайки функцията ***set”***, тъй като самият указател е статичен и е може да се променя, за да не се изгуби връзка с елемента. Той е константа.

2.Твърдение на стр.9 ln_17, col_16: “ако това е име на функция, следва активирането й с аргумент, съответстващ на стойността на елемента (бутона)”. Аргументът се въвежда от потребителя, след като е взет от елемента като свойство посредством функцията get. Тази функция се използва в същия символен низ CALLBACK, в който е и функцията, въведена от потребителя.

3. Твърдение на стр.10 ln_7, col_59: “или осъществява пренасочване на потока на решението към друга опция, като променя обратната връзка на някой графичен обект (средство на интерфейса).”Това твърдение може да допълни и доуточни както следва: или осъществява пренасочване на потока на решението към друга опция, като променя обратната връзка и/или друго свойство на някой графичен обект (средство на

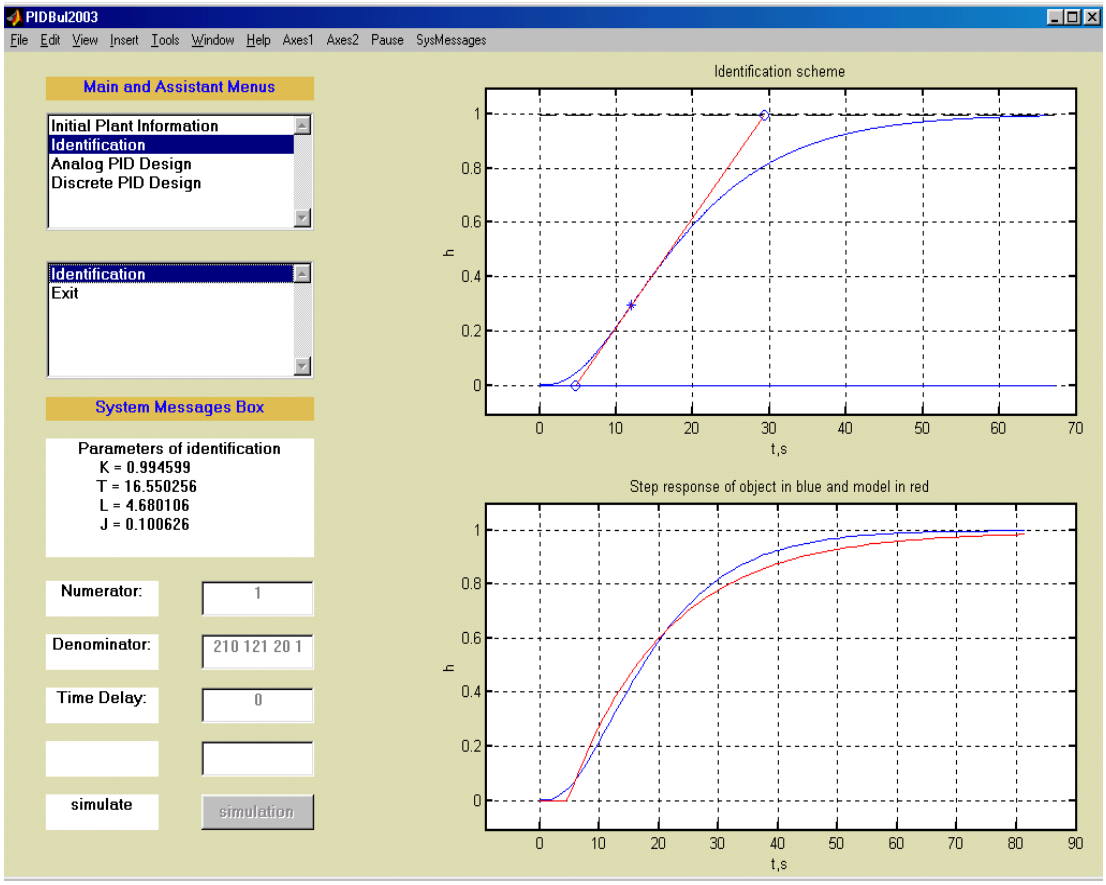
интерфейса). Напр: при подмяна на опциите в главния Listbox на PIDBUL се сменя не само обратната му връзка, но и самите опции като списък от символни низове.

Описание на функциите:

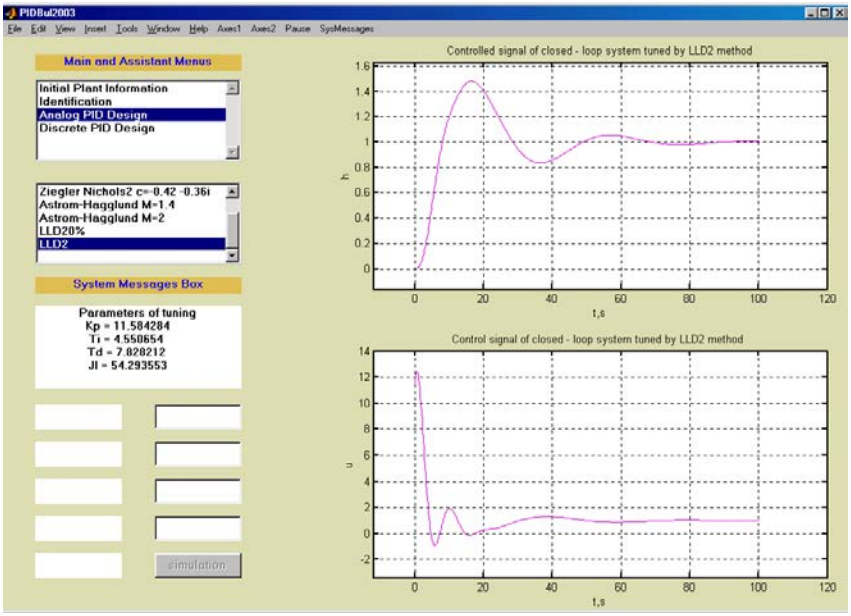
Външен вид на системата:

.

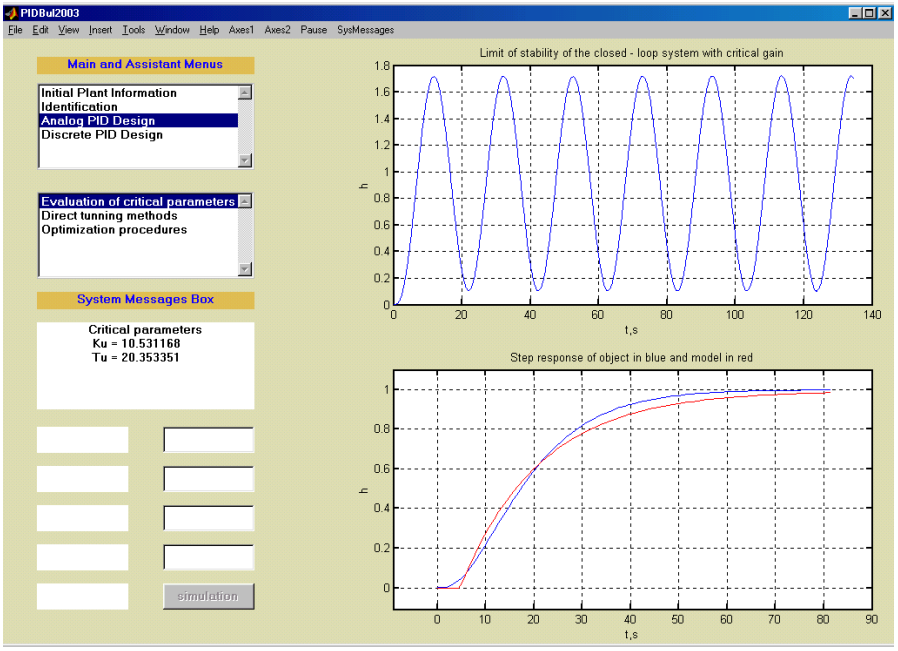
1.



2.



3.



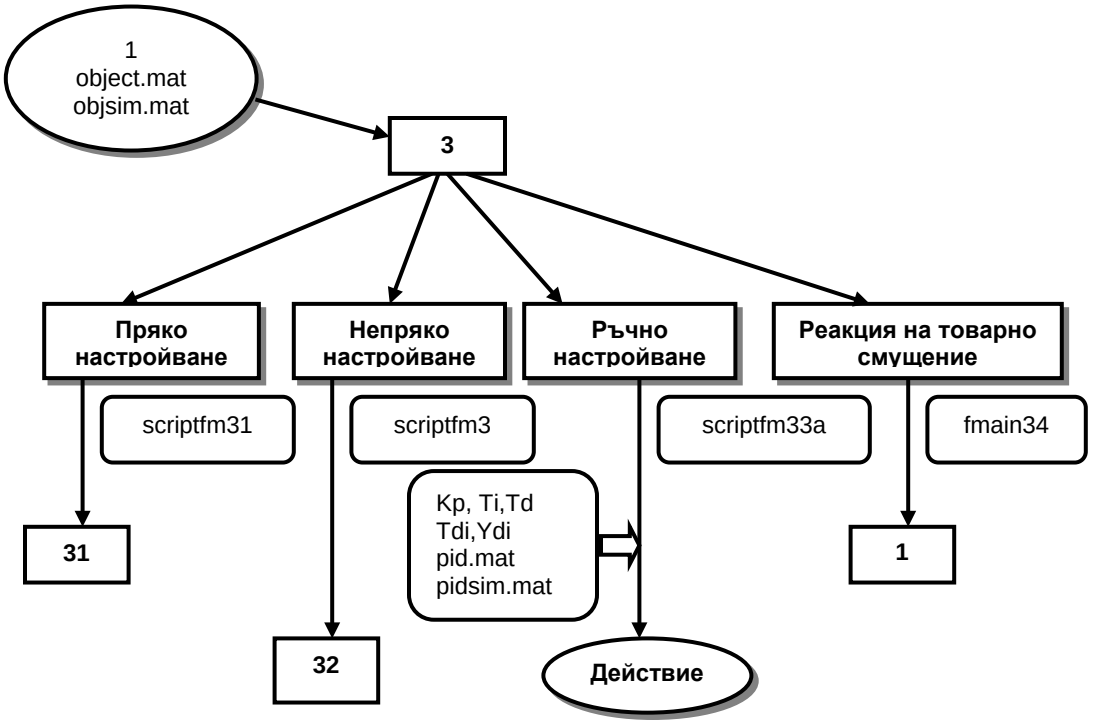
Типове базови функции в програмната система PIDBUL

1. Динамично работещи в MATLAB и SIMULINK.
   Това са функции, които в процеса на работа циклично извикват SIMULINK – схеми, които използват в процеса на изчисления на изходните си променливи. Такава функция например е функцията за определяне на критичните показатели на системата. Тя използва оптимизационна процедура, която извиква на всяка итерация SIMULINK –схема, за да изчисли до колко близо на дадения етап, е системата до желания (автоколебателен) режим. След привеждането на системата в него, функцията снема критичните и параметри, (период на колебанията и коефициент на усилване) и ги предава като изходни. Други динамични функции в PIDBUL са функциите за оптимално настройване на регулаторите. Те използват също оптимизационна процедура, но не за да привеждат системата в автоколебателен режим, а за да минимизират богат набор от показатели на качеството, гарантиращи желаното поведение на затворената система.
   Функцията за оптимална трипараматрична идентификация по квадратичен критерий е също динамична.
2. Статично работещи в MATLAB.
   Това са класически функции, които използват вграден статичен алгоритъм за изчисляване на изходните променливи от входните. При тях няма циклични изчисления, извикващи допълнителни функции и схеми. Такива са всички инициализиращи и организиращи функции на PIDBUL и част от обслужващите.
   Статичните функции могат да се класифицират по следния начин:
   - Според целите и задачите на функцията:

Клас на функцията	**Действие на функцията**	**Функции от този клас**	**Основни входни аргументи**	**Изходни аргументи**
Непряко настройване на непрекъснат ПИД	Изчислява параметрите на регулатора по данните от идентификацията	ZN1.m CC.m CHRL.m	Параметри на типов модел на обекта и тип на регулатора	Параметри на регулатора
Пряко настройване на непрекъснат ПИД	Изчислява параметрите на регулатора по критичните параметри на обекта	ZN2.m HAH.m ZN2_mod.m	Критични параметри на САУ и тип на регулатора	Параметри на регулатора
Модифицирано пряко настройване на непрекъснат ПИД	Изчислява параметрите на регулатора по критичните параметри на обекта и по други негови характеристики	LLD2.m	Предавателна функция	Параметри на регулатора
Определяне на инфлексната точка	Определя координатите на инфлексната точка в преходния процес	INFL_IND	Преходен процес на обекта	Координати на три точки около инфлексната точка
Автоматизирано привеждане на САУ в автоколебателен режим	Изчислява критичните параметри	fcriticalpar	Предавателна функция	Критични параметри на САУ
Проверка за устойчивост	Прилага алгебричен критерий за устойчивост на обекта	isstablemodel	Характеристичен полином на обекта	1, ако обектът е устойчив, и 0 в противен случай
Изчисляване на крайното време за симулиране на САУ	Изчислява крайното време за симулиране на обекта или на настроената САУ	tstop.m tstop2.m tstopa.m	Данни за обекта и/или за настройки на регулатора	Крайно време на симулация
Формиране на ограниченията в САУ	Изчислява ограниченията в регулатора при оптимизационно настройване	Constr	Текущи настройки на регулатора	Вектор на ограниченията
Изчисляване на показателите на качеството за	Симулира САУ с текущо настроен ПИД регулатор и	Jer.m JER1.m Jer2.m	Текущи настройки на регулатора	Показател на качеството

оптимизационни процедури	изчислява съответния показател на качеството			
Въвеждене на данни от потребителя	Проверява коректността на въведените от потребителя параметри	freadden.m freadhb.m freadK.m	Проверяваните параметри	Флагове със стойност 1, ако параметрите са коректно въведени, и стойност 0, в противен случай
Коригиране на изображения на процеси	Уточнява подходящите за потребителя параметри на графичния прозорец за визуализиране на резултати	axL	Вектори с информация за изобразявания процес	Нови параметри на графичния прозорец

- Според йерархия на функциите – при зиползване на диференциален метод на програмиране на системата, имаме множество функции с различна йерархия. Главните функции извикват допълнителни (помощни функции) за изчисляване на част от задачите. Допълнителните функции от своя страна също могат да извикват други и т.н. Главни функции са следните: fcriticalpar (статичен еквивалент на динамичната функция за определяне на критичните параметри на системата), функции за идентификация и всички инициализиращи и обслужващи. Помощните функции се делят на следните:
  - Функции за пряко и непряко настройване на ПИД – регулатори.
  - Функции, които използват модел на обекта – функция за определяне на инфлексна точка (INFL_IND), функция за определяне на крайно време на симулацията (tstop.m), функция за проверка на устойчивостта на въведения модел на обекта (isstablemodel).
  - Помощни функции към интерфейса: функции за проверка на коректно въведени параметри, функции за преформатиране на въведените параметри, функция за разчертаване границите на графиките (axL).
  - Помощни функции в оптимизацията: функция за изчисляване на ограниченията (constr), и функции за изчисляване на показателя на качеството.


1
object.mat
objsim.mat
3
Пряко настройване
Непряко настройване
Ръчно настройване
Реакция на товарно смущение
scriptfm31
scriptfm3
scriptfm33a
fmain34
Kp, Ti,Td
Tdi,Ydi
pid.mat
pidsim.mat
31
32
1
Действие