Съдържание

[0. Списък с използваните съкращения 3](#_Toc519173994)

[1. Увод 4](#_Toc519173995)

[1.1. Общи сведения за биотехнологичните процеси 4](#_Toc519173996)

[1.2. Особености на БТП като обект на моделиране 6](#_Toc519173997)

[1.3. Особености на БТП като обект на управление 7](#_Toc519173998)

[2. Описание на обекта за управление 8](#_Toc519173999)

[2.1. Структурни и неструктурни модели на БТП 8](#_Toc519174000)

[2.2. Математичен модел на полупериодичен БТП 8](#_Toc519174001)

[2.3. Математически модел на полупериодичен ферментационен модел E. coli MC4110 11](#_Toc519174002)

[3. Изграждане на симулатор на обекта, работещ в реално време 13](#_Toc519174003)

[3.1. Симулатор на обекта изграден в Simulink 13](#_Toc519174004)

[3.2. Идентификация на нелинейния модел E. coli MC4110 и намиране на линеен модел 16](#_Toc519174005)

[4. Проектиране на регулатори и филтри на Калман 21](#_Toc519174006)

[4.1. Проектиране на ПИД регулатори 21](#_Toc519174007)

[4.1.1. Проектиране на аналогов ПИД регулатор 22](#_Toc519174008)

[4.1.2. Проектиране на цифров ПИД регулатор 27](#_Toc519174009)

[4.2. Feed Forward съставка на управлението 29](#_Toc519174010)

[4.3. Синтезиране на разширен филтър на Калман 30](#_Toc519174011)

[4.4. Оптимизационна процедура за настройка на ПИД регулатор базирана на генетичен алгоритъм 33](#_Toc519174012)

[4.5. Оптимизационна процедура за настройка на ПИД регулатор използваща линеаризиран модел 41](#_Toc519174013)

[4.6. Проектиране на линейно – квадратичен гаусов регулатор 44](#_Toc519174014)

[4.6.1. Проектиране на линейно – квадратичен регулатор с интегрална съставка 44](#_Toc519174015)

[4.6.2. Проектиране на стохастичен дискретен наблюдател – филтър на Калман 46](#_Toc519174016)

[4.7. Симулационни изследвания и сравнения между проектираните регулатори 50](#_Toc519174017)

[5. Разработка на софтуерно осигуряване на системата за управление 51](#_Toc519174018)

[5.1. Хардуерната платформа Arduino Due 52](#_Toc519174019)

[5.1.1. Комуникация 53](#_Toc519174020)

[5.1.2. Програмиране 53](#_Toc519174021)

[5.2. Интегрирана среда за разработка (ИДЕ) – Arduino IDE 54](#_Toc519174022)

[5.3. Разработка на системата за управление на биотехнологичен процес (БТП) 55](#_Toc519174023)

[5.3.1. Основна структура на Arduino програмите 56](#_Toc519174024)

[5.3.2. Серийна комуникация между микроконтролера и компютъра 56](#_Toc519174025)

[5.3.3. Разработени Функции и Алгоритми 57](#_Toc519174026)

[6. Експериментални резултати 63](#_Toc519174027)

[6.1. Сравнение между симулационните и реалните резултати при настройка на ПИД регулатора с генетичен алгоритъм 63](#_Toc519174028)

[6.2. Сравнение между симулационните и реалните резултати при настройка на ПИД регулатора с конвенционален метод на оптимизация 64](#_Toc519174029)

[6.3. Сравнение между симулационните и реалните резултати при линейно-квадратичния регулатор 65](#_Toc519174030)

[6.4. Сравнение между реалните резултати при настройка на регулатора с конвенционален метод на оптимизация и при настройка с ГА както и на линейно-квадратичния регулатор 66](#_Toc519174031)

[7. Заключение 67](#_Toc519174032)

[8. Използвана литература: 68](#_Toc519174033)

# 0. Списък с използваните съкращения

АЦП – Аналогово-Цифров преобразувател

БТП – Биотехнологичен процес

ГА – Генетичен алгоритъм

ППФП – Полупериодичен ферментационен процес

РФК – Разширен филтър на Калман

ФП – Ферментационен процес

ЦАП – Цифрово-Аналогов преобразувател

HIL – Hardware in the Loop

ШИМ – Широчинно импулсна модулация

IDE – Integrated Development Environment

# 1. Увод

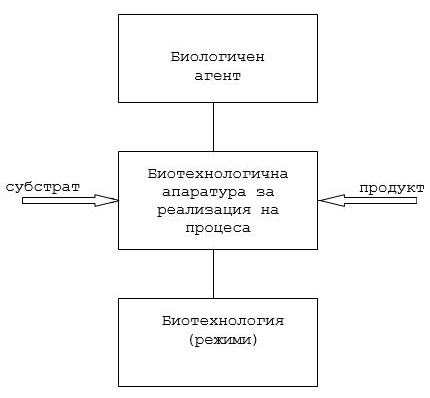
## 1.1. Общи сведения за биотехнологичните процеси

Биотехнологията е една от най-бързо развиващите се области на науката и практиката. Тя може да се разглежда като система от подходи, методи и технологични решения за индустриализиране на биологията, чрез нейната интеграция с химическите, физическите и техническите науки. Биотехнологичните продукти намират приложение в хранително-вкусовата, фармацевтичната, козметичната, фуражната, металургическата, нефтохимическата и много други промишлености. Чрез биотехнологията могат да се получат ценни продукти от метаболизма на живите клетки (антибиотици, витамини, аминокиселини, ензими и други биологично активни вещества), да се трансформират определени вещества в по-ценни крайни продукти (захарни сиропи във фруктоза, органически отпадъци в биогаз и т.н.), получаване на клетъчна биомаса (фуражен белтък, биоконсерванти, бактериални торове)[6].

Производствените процеси, които използват продукта от жизнената дейност на микроорганизмите, се наричат биотехнологични процеси (БТП). Те са основни процеси на биотехнологията. От тях се получават полезни продукти чрез използването на микроорганизми, вируси, клетки от животни и растения, а също и техните компоненти и екзопродукти.

Процесът на микробиологичният синтез, водещ до целенасочено образуване на необходимия биопродукт се извършва в биотехнологична система. На фиг. 1.1 е показана структурата на една такава типична система и нейните основни компоненти. Тя представлява съвкупност от взаимно свързани биотехнологични потоци и апарати, в които се извършва определена последователност от операции, които осигуряват преработването на изходните материали (субстрат) в краен продукт. По – важни елементи на биотехнологичната система са:

* Инокулатори за получаване на посевен материал;
* Сепаратори за отделяне и сгъстяване на биомасата;
* Биореактори (ферментори за провеждане на ферментационни процеси);
* Системи за наблюдение и обработка на информацията от БТП;
* Технически средства за контрол и автоматизация и други;



Фиг. 1.1.

Обект на научно изследване в тази разработка е самият микробиологичен процес – процесът на ферментация (култивиране, биосинтез, биотрансформация, биоконверсия и т.н.).

Той има следните особености:

* провежда се обикновено в условия на изключителна чистота на културата, която се постига чрез стерилизация на основната и на спомагателната апаратура, а също на всички компоненти, постъпващи в биореактора;
* осъществява се в хетерогенни многофазни системи, чиито биологични свойства се изменят в хода на процеса;
* многокомпонентност на хранителните среди;
* сложност на биохимичните механизми, регулиращи растежа на биомасата и синтеза на микробните метаболити;
* автокаталитичност, т.е. влияние на продуктите на реакцията (включително на образуваната биомаса и на синтезираните ензими) върху скоростта на протичане на процеса;
* по-голяма нестабилност в сравнение с химикотехнологичните процеси;
* относително ниски скорости на реакциите, ниски концентрации на субстратите и получаваните продукти, които обикновено не са крайни;
* умерена температура в обхвата 20-40 0C и близки до неутралните стойности на рН;
* разлика в оптималните условия на растеж на микроорганизмите и биосинтеза на целевите продукти на метаболизма;
* многофазна структура на сложната система на култивиране на микроорганизмите;
* сложност на технологичния процес на култивиране на микроорганизмите, който изисква комплексно техническо оборудване, метрологично и програмно осигуряване и др.

## 1.2. Особености на БТП като обект на моделиране

Както е известно, теорията на управлението предполага познаване на структурата и параметрите на модела на управлявания обект. Адекватното и пълно описание на тези процеси и по-конкретно на ферментационните процеси (ФП) се възпрепятства от сложността и спецификата на тези процеси, изразяваща се в:

* голям брой фактори, влияещи на хода на ФП. Отчитането им е практически невъзможно. Механизмът на влияние на някои фактори не е добре изучен, особено от гледна точка на математическото му описание. Сред факторите, които влияят по-съществено на ФП, са:
* температура;
* налягане;
* концентрация на водородни йони;
* концентрация на реагенти (клетъчна маса, лимитиращ субстрат, метаболитен продукт);
* концентрация на разтворените газове (кислород, въглероден диоксид);
* скорост на разбъркване;
* аериране;
* характеристики на средата (вискозност, плътност)
* някои от изброените по-горе фактори и въздействия е трудно да бъдат бързо и точно измерени или оценени.
* особеност на ферментационните процеси е силната нелинейност на характеристиките и нестационарността на параметрите на модела.
* друга особеност на ФП е неповторяемостта. Тя е следствие от множеството взаимодействия и невъзможността от пълното им отчитане. Друга причина е невъзможността за провеждане на множество експерименти при напълно еднакви условия (еднакви нива на контролируемите и неконтролируемите фактори).

Съществуват различни подходи за построяване на математически модели като най-резултатен се оказва комбинираният подход т.е. конкретния вид на зависимостите се намира аналитично, а числените значения на коефициентите по експериментални данни. Обикновено моделите се представят чрез система нелинейни диференциални уравнения. Структурата на системата се определя от вида на биотехнологичния процес (периодичен, полупериодичен, непрекъснат) и от характеристиките, използвани за неговата динамика.

При периодичните процеси всички компоненти на хранителната среда и посевния материал в биореактора се зареждат еднократно в началото на процеса. След известно време също така еднократно се източва културалната течност от апарата, след което се сепарират биомасата и другите биопродукти. Характерно е непрекъснатото изменение на физиологичното състояние на клетките и състава на културалната среда, което е свързано с жизнените процеси на микроорганизмите.

Полупериодичните процеси се отличават от периодичните по това, че в тях едно или няколко вещества, необходими за растежа и развитието на микроорганизмите, се добавят в биореактора в процеса на култивирането. Отделянето на крайния продукт става еднократно. Добавянето на хранителни вещества и други компоненти в хода на ферментацията дава възможност да се създадат по-благо­приятни жизнени условия в различните фази на култивирането.

Подаването на хранителна среда в биореактора при непрекъснатото култивиране и източването на културалната течност от него се извършва в непрекъснат процес. При постоянни скорости на потоците в тези системи се установява стационарно състояние, което се запазва продължително време.

## 1.3. Особености на БТП като обект на управление

Специфичните особености на моделирането на БТП са едновременно и особености на тяхното управление. По съществените от тях са:

* ниските скорости на изменение на основните параметри (концентрациите на субстанциите) са особености на ФП, която е благоприятна за управлението им. Те могат да бъдат дискретизирани през интервал от 1-2h. Това време е напълно достатъчно за решаване на по-сложни задачи за управление (оптимизационни задачи), за осъществяване на система на пряко цифрово управление, както и в режим "реално време". Това дава възможност за решаване на задачите за оптимизация в on-line режим;
* според вида на ФП (периодични, полупериодични, непрекъснати) съществуват и специфични особености при тяхната оптимизация.
* характеризират се с различие на оптималните условия за растежа на клетъчната маса и за продуциране на метаболитен продукт. Това затруднява формулирането на критерии на оптималност.

# 2. Описание на обекта за управление

## 2.1. Структурни и неструктурни модели на БТП

Един от подходите при създаване на модели на БТП се основава на механизма на биохимичните превръщания и влиянието на външните фактори върху структурата на клетката и процесите, които се извършват в нея. Този подход води получаване на т. нар. структурни модели и се прилага при моделирането на добре изучени биотехнологични процеси. Структурните модели описват вътрешноклетъчните механизми и отразяват биохимията на клетката. Това определя тяхната сложна структура, големия брой параметри, които трябва да бъдат идентифицирани, както и необходимостта от съответния обем експериментални данни за оценяване параметрите на модела[6].

Неструктурните модели описват на макро ниво схемата на биохимичните превръщания, като я заменят с приблизителна опростена схема от макро реакции. Структурата на модела е ограничена от използването на променливи, които могат да бъдат надеждно измервани и анализирани в хода на процеса. Това ги прави изключително широко приложими при управление на БТП. Основният подход при изграждането им се базира на материалния баланс на величините, характеризиращи растежа на микроорганизмите, потреблението на лимитиращите субстрати и натрупването на целеви продукти. Това са кинетични модели, които свързват основните променливи на процеса с управляващите въздействия.

## 2.2. Математичен модел на полупериодичен БТП

За този тип процеси са най-подходящи математични модели от неструктурен тип. Те най-лесно и най-просто се изграждат въз основа на материалните макро баланси за основните променливи: концентрация на биомаса *X*, концентрация на лимитиращ субстрат *S* и концентрация на продукт *P* и определен набор от зависимости за специфични скорости за растеж на биомаса *µ*, потребление на субстрат *η* и синтез на продукт *ε*.

Този подход, обаче обикновено води до изграждането на кинетични модели на полупериодичните ферментационни процеси, които не са пригодени за оптимизация и управление. Затова като втори етап е необходимо да се намери връзката между коефициентите на кинетичния модел и евентуално управляващо въздействие.

Ако се предположи, че БТП може да се разглежда като детерминиран процес и в биореактора се осъществява идеално смесване, скоростта на биохимичните превръщания може да се представи във вид на функционална зависимост:

rp = fp (pc, c, pH, T, …), (2.1)

rs = f(rp, …), (2.2)

където *rp* е вектор на скоростта на образуване на продукт в единица обем за единица време, *rs* – вектор на скоростта на потребление на хранителна среда в единица обем за единица време.

Функционалните зависимости (2.1), (2.2) не отразяват микроскопичните процеси вътре в отделните клетки. Материалните микробаланси за полупериодичния процес може да се представят по следния начин:

Баланс на масите във ферментационната среда

Балансът може да се представи с израза

= Fρ0, (2.3)

където *ρ1* е плътността на културалната среда, *ρ0* – плътност на хранителната среда. Тъй като на практика *ρ1* ≈ *ρ0*, горният израз може да се представи във вида:

= F. (2.4)

Баланс на продуктите на биосинтеза

Математически балансът на продукта се представя във вида

= rpV. (2.5)

Баланс на компонентите на хранителната среда

=

Математическото описание на горния баланс се дава с диференциалното уравнение

= FS - rsV. (2.6)

Съвкупността от диференциални уравнения (2.4) – (2.6) представлява математическия модел на полупериодичен биотехнологичен процес.

Всички фактори, оказващи влияние на хода на БТП могат да се разделят на две групи. Първата включва факторите *u*, които се контролират непосредствено в хода на процеса. Такива са температура *T*, *pH*, концентрация на разтворен кислород *pO2*, скорости на смесване и др. Може да се въведе обозначението

u’ = [T, pH, pO2, P, ….]T

Към първата група се отнасят и такива управляващи фактори като скорост на подаване на хранителен разтвор *Q*, скорост на разбъркване *D = Q / V*, концентрация на хранителен субстрат *S0* и други

u’’ = [D, Q, S0]T = [u1, u2, u3]T

Общо първата група фактори се дават с u = [u’, u’’].

Във втората група се включват такива фактори *ν*, които не се поддават на непосредствен контрол, като предисторията на процеса, възрастовата група на популацията, изменение на генетическите свойства на културата и други.

След отчитане на въведените обозначения моделът полупериодичния процес може да се представи по следния начин:

= fp (pc, S, u, V) - pc,

= - fs (pc, S, u, V) - (S0 - S),

= u2 = F

При синтеза на модела трябва да се имат предвид следните допълнителни допускания:

* математическият модел трябва да съдържа разумен брой уравнения;
* предполага се, че само един елемент от хранителната среда обуславя скоростта на растежа на микроорганизми и влияе на скоростта на синтез на целевия продукт и се нарича лимитиращ; останалите компоненти са в излишък;
* не се отчита биологичната инерционност при изменение на външните условия;
* неотчетените променливи и фактори влизат в параметрите на модела;

Изложените допускания намаляват адекватността на модела на реалния процес, но са необходими за синтеза, защото правят модела пригоден за ефективно използване на оптимизационните методи.

Обикновено като съществени променливи при моделирането на БТП се приемат концентрацията на биомаса *X* и концентрацията на лимитиращ субстрат *S*. Тогава за *pc* може да се запише

Pc = [X|P]T  (2.7)

При определяне на функционалните зависимости *fx*, *fp* и *fs* се приема, че концентрацията на целевия продукт не влияе на скоростта на микробиологическия синтез на биомаса, целевия продукт и потреблението на лимитиращ субстрат и за удобство при по-нататъшните разглеждания се приема, че функционалните зависимости са представени във вида

fx (X, S, u’) = Xµ (X, S, u’),

fp (X, S, u’) = Xη (X, S, u’), (2.8)

fs (X, S, u’) = Xε (X, S, u’),

където *µ*, *η*, *ε* са съответно специфичните скорости на натрупване на биомаса, целевия продукт и потреблението на лимитиращ субстрат.

След заместване на зависимостите (2.8) в (2.7) за модела на полупериодичния БТП се получава:

= X µ (X, S, u’) - X,

= - Xη (X, S, u’) - (S0 - S), (2.9)

= u2

Така получения модел на полупериодичен процес е обобщен, тъй като специфичните скорости *µ*, *η*, *ε* са дадени в общ вид. За всеки реален процес *µ*, *η*, *ε* се задават с конкретни изрази, които в голяма степен определят свойствата на модела.

## 2.3. Математически модел на полупериодичен ферментационен модел E. coli MC4110

В 2.2. беше представен и изведен обобщен модел на полупериодичен биотехнологичен процес. Тема на настоящата разработка е полупериодичен ферментационен процес (ППФП) описан с модел на E. coli MC4110 [1]. Динамиката на ППФП е представена като нелинеен модел в пространство на състоянията:

+ η(t) (2.10)

γs = HX + ξ(t),

където X(t) = [γx(t) γs(t) V(t)]T – вектор с променливи на състоянието;

f(X, Q) = , H = [0 1 0]

γx(t) – концентрация на биомасата [g.l-1];

γs(t) – концентрация на субстрата [g.l-1];

γin(t) – концентрация на субстрата в подхранващия разтвор [g.l-1];

V(t) – обем на биореактора [l];

Q(t) – дебит на подхранващия продукт [l.h-1];

Ys/x – бездименсионен икономичен коефициент [-];

Неточността в модела е моделирана посредством бял гаусов шум с нулева средна стойност и дисперсии съответно:

Dηγx = 0.0025 g2. l-2.h-1 – дисперсия на шума в концентрацията на биомасата;

Dηγs = 0.001 g2.l-2.h-1 – дисперсия на шума в концентрацията на субстрата;

Dξ = 0,0025g2.l-2.h-3 – дисперсия на измервателния шум;

В системата уравнения *f(X, Q) µ(t)* представлява специфична скорост на растеж на микроорганизмите, която се дава с кинетичния модел на Моно

µ(t) = µmax(t) (2.11)

Той описва растежа на микроорганизмите като функционална зависимост между специфичната скорост на растеж и концентрацията на субстрата. Моделът представлява монотонно растяща функция с горна граница *µmax(t)* при *γs* . Параметърът *µmax* следователно се нарича максимална специфична скорост на растеж на микроорганизмите [h-1] и *ks* – полу-наситена константа на насищане [g.l-1], защото когато *γs(t) = ks* специфичната скорост на растеж на микроорганизмите *µ* става равна на *,* което е половината от максималната специфична скорост на растеж. Ключовият момент в модела на Моно е, че *µ* нараства с нарастването на *γs* както се очаква, но при ниски нива на *γs* специфичната скорост на растеж *µ* рязко спада.

Максималната специфична скорост на растеж *µmax* е характеристика на всички организми и е свързана с тяхната способност да се възпроизвеждат в хранителна среда. Тя се характеризира като нарастване на биомасата за единица време при оптимални жизнени условия.

Константата *ks* е пряко свързана с транспортния механизъм на субстрата през клетъчната мембрана така, че зависи от свойствата на клетъчната мембрана и вътреклетъчните условия, от типа на транспортните протеини в клетката и от свойствата на субстрата.

Параметрите на модела (2.10) са оценени чрез идентификационна процедура, базирана на генетичен алгоритъм. Използвани са реални експериментални данни, измерени в Института по инженерна химия на университета в град Хановер като началните условия на експеримента са:

t0 = 6.68h, γs(t0) = 0.81 g.l-1, γx(t0) = 1.25 g.l-1, V(t0) = 1.35l.

След идентификацията за стойностите на параметрите се получава [1]:

ks = 0.012 g

l-1, Ys/x = 0.5, γin = 100gl-1, µmax =0.55h-1

# 3. Изграждане на симулатор на обекта, работещ в реално време

## 3.1. Симулатор на обекта изграден в Simulink

Симулаторът на обекта работещ в реално време е изграден в Simulink с помощта на пакета с инструменти Instrument Control Toolbox. Моделът на обекта за управление, описан с уравнения (2.10), е дискретизиран с първа права разлика (“forward difference”) при такт на дискретизация Т0 = 0.0056h:

= t = kT0 =>

=

или = + -

= t = kT0 =>

=

или = - -

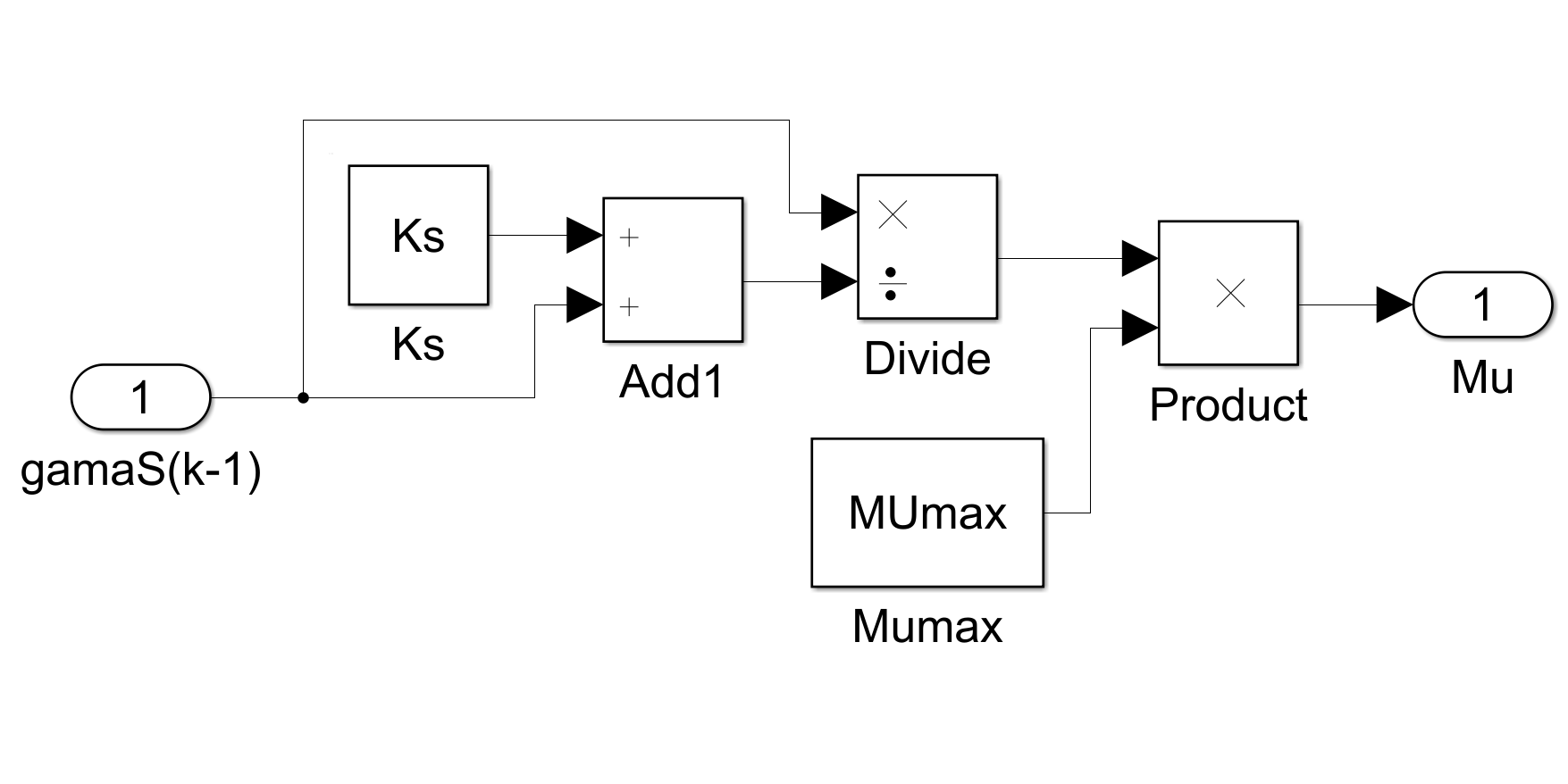
= Q(t)|t = kTo => или V(k+1) = V(k) +T0Q(k)

Така получените диференчни уравнения са моделирани в средата на Simulink и е получен дискретен нелинеен модел на обекта (фиг. 3.1)



Фиг. 3.1 Дискретен нелинеен модел на E. coli MC4110

Тук с *Random Number, Random Number2, Random Number3* са добавени шумовете на обекта, а блоковете Memory имат началните условия съответно [1.25 0.81 1.35]. Блокът *Mu* има следната структура(фиг. 3.2):



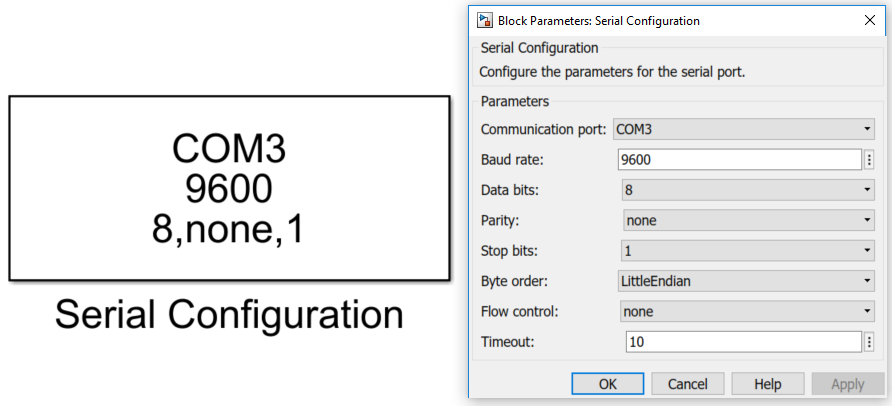
Фиг. 3.2 Модел на специфичната скорост на растеж на микроорганизмите

Симулаторът на дискретизирания нелинеен модел на обекта на управление, работещ в реално време, е показан на фиг. 3.3.



Фиг. 3.3 Simulink схема на модела на обекта на управление

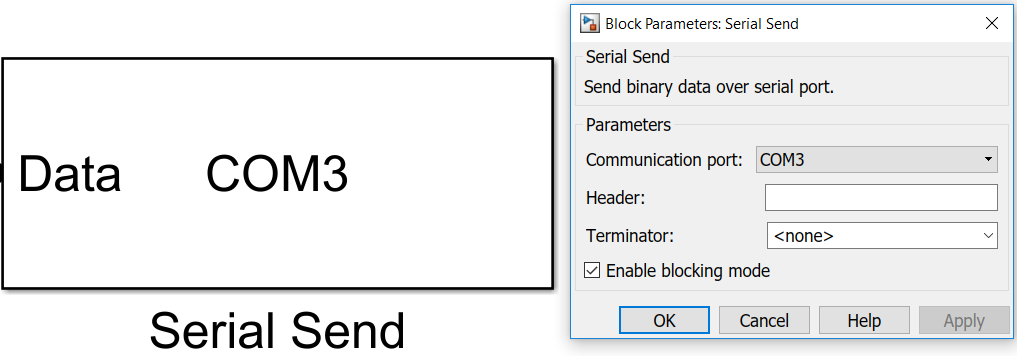
Посредством блока *Serial Configuration* се конфигурират параметрите за серийния порт така, че да могат да се изпращат и получават данни към и от микроконтролера. Този блок се добавя и конфигурира преди да са добавени блоковете Serial Receive и Serial Send като настройките му са показани на фиг. 3.4.



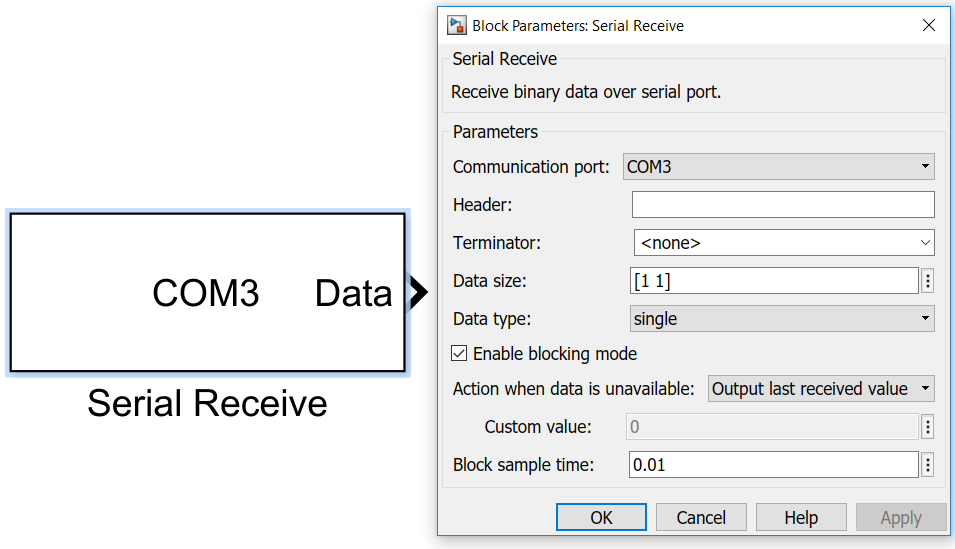
Фиг. 3.4 Настройки на блока Serial Configuration

При свързан микроконтролер операционната система разпознава го разпознава като виртуален *COM PORT*, който се задава в полето communication port. В следващото поле се указва т. нар. baud rate, който показва каква е максималната скорост, с която могат да се предават данните. В случая тя е 9600 бита за секунда. За конкретната реализация на системата следващите полета са оставени както са си по подразбиране.

След като сме конфигурирали серийния интерфейс можем да добавим и блоковете съответно за получаване и изпращане на данни, които заедно с техните конфигурации са показани на фиг. 3.5 и фиг. 3.6.



Фиг. 3.5 Настройки на блока Serial Send



Фиг. 3.6 Настройки на блока Serial Receive

След като данните биват получени през серийната комуникация те се преобразуват от числа с единична точност (single) към числа с двойна точност (double) и преди да се изпратят чрез блока Serial Send, те се преобразуват от double към single.

## 3.2. Идентификация на нелинейния модел E. coli MC4110 и намиране на линеен модел

За целите на изследването е необходимо да се проведе активен експеримент с изследвания обект, при който върху обекта се упражняват специално формирани въздействия, за да може да се наблюдава съответната реакция на обекта. Експериментът се прилага в режим на нормална експлоатация на обекта (функциониране в затворен контур). Към управляващия сигнал се добавя случайна двоична последователност за осигуряване на постоянно възбуждане на обекта за управление (фиг. 3.7), при което системата се разклаща около установената стойност 0.1. Допълнителният сигнал се формира от бял гаусов шум, с нулева средна стойност и единична дисперсия, който се преобразува през нелинеен елемент – реле със стойности ±1. Изпитателният сигнал преминава през коефициент 0.01, за да се постигне 10% зашумяване. Входно – изходните данни се запазват в масиви съответно *Q* и *gamaS*, където те подлежат на последваща обработка. Масивът *Q* се формира като от управлението се извади нелинейната компонента (feed forward control).



Фиг. 3.7 Схема събиране на входно-изходни данни в затворен контур

Вземат се установените стойности на сигналите, след което входно-изходните наблюдения се центрират:

InputData = Q(91: end)-mean (Q (91: end));

OutputData = gamaS(91: end)-mean(gamaS(91:end));

Данните се разделят на две части, като едната част от тях се ползва за същинската идентификация, докато другата част се ползва за валидиране на полученият модел:

InputDataЕ = InputData(1:end/2);

OutputDataЕ = OutputData(1:end/2);

InputDataV = InputData(end/2:end);

OutputDataV = OutputData(end/2:end);

От така разделените данни се създават програмни обекти *iddata object* съответно за оценяване *datae* и за валидация *datav*.

datav=iddata(OutputDataV, InputDataV,0.0056)

dataе=iddata(OutputDataЕ, InputDataЕ,0.0056)

Задачата за идентификация на изследвания обект включва процедура за търсене на модели с най-подходяща структура и определяне на техните параметри в съответствие с тази структура, за целта трябва да се сравнят два или повече модела от различен тип и/или с различни структури, за да се избере най-добрият от тях въз основа на някаква мярка изчислена след оценяване на параметрите им.

Идентификацията е започната с избор на достатъчно общ линеен дискретен параметричен модел от тип ARX *A(q)y(k) = B(q)u(k) + e(k).* Дефинираме множество от модели, от които се избира този с най-добра структура:

NN =struc(1:15,1:15,1:15);

V = arxstruc(datae, datav, NN);

modelOrder = selstruc(V,'aic');

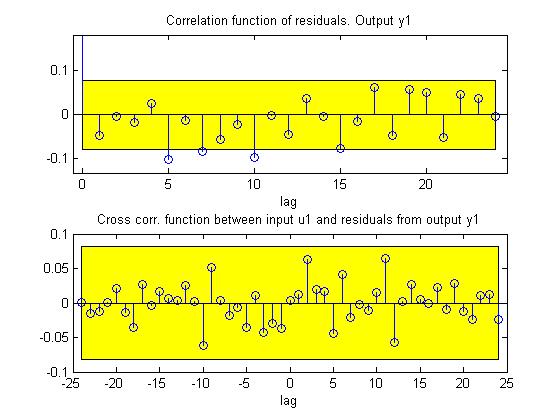
С функцията *struc* се създава матрица всеки ред, от която съответства на конкретни стойности на структурните параметри *na*, *nb*, *nk*. Функцията *arxstruc* изчислява показателите на качеството за всички структури на модела от тип ARX и накрая *selstruc* връща най-добрата структура на модела като използва модифицираният показател критерии на Акаике ‘aic’ и избраният модел е с параметри *na = 10*, *nb = 10* и *nk = 1*. Оценен е модел и е представен с доверителните си интервали:

m=arx (datae, [ 10 10 1])

present(m) Таблица 3.1

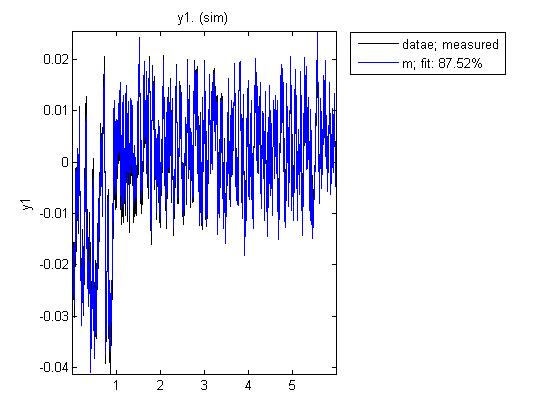
|  |
| --- |
| Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t) |
| A(q) = 1 - 0.7841 (+-0.03101) q^-1 - 0.02641 (+-0.03952) q^-2 |
| - 0.006618 (+-0.03946) q^-3 + 0.001188 (+-0.03964) q^-4 |
| + 0.005868 (+-0.03957) q^-5 + 0.006265 (+-0.03956) q^-6 |
| - 0.03197 (+-0.03956) q^-7 - 0.05061 (+-0.03959) q^-8 |
| -0.02036(+-0.03961)q^-9 - 0.08411(+-0.03145)q^-10 |
| B(q) = 0.294(+-0.001115)q^-1 + 0.06205 (+-0.009183)q^-2 + 0.05351 (+-0.009299) q^-3 + 0.05238 (+-0.009364)q^-4 + 0.05363(+-0.009378) q^-5 + 0.05553 (+-0.009399) q^-6 + 0.05671 (+-0.009432) q^-7 + 0.04652 (+-0.009488) q^-8 + 0.0338 (+-0.009495) q^-9 + 0.02592 (+-0.009382) q^-10 |
| Estimated using ARX from data set datae |
| Loss function 5.65746e-007 and FPE 5.86876e-007 |
| Sampling interval: 0.0056 |

От таблица 3.1 се вижда, че стандартните отклонения на някои оценки са съизмерими със самите оценки, което ни дава основание да редуцираме реда на модела. Правим опит с 5-ти ред. Тук тестът за остатъчната грешка след оценяването (фиг. 3.8) показва, че автокорелационната функция има стойности извън ограничителните зони, откъдето съдим за това, че остатъчната грешка не е бял гаусов шум и моделът не е достоверен. Преминаваме към по-сложния модел от тип ARMAX.

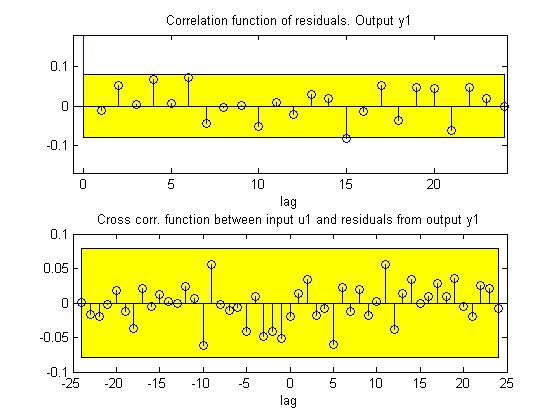


Фиг. 3.8 Графика на остатъчната грешка за ARX(5, 5, 1)

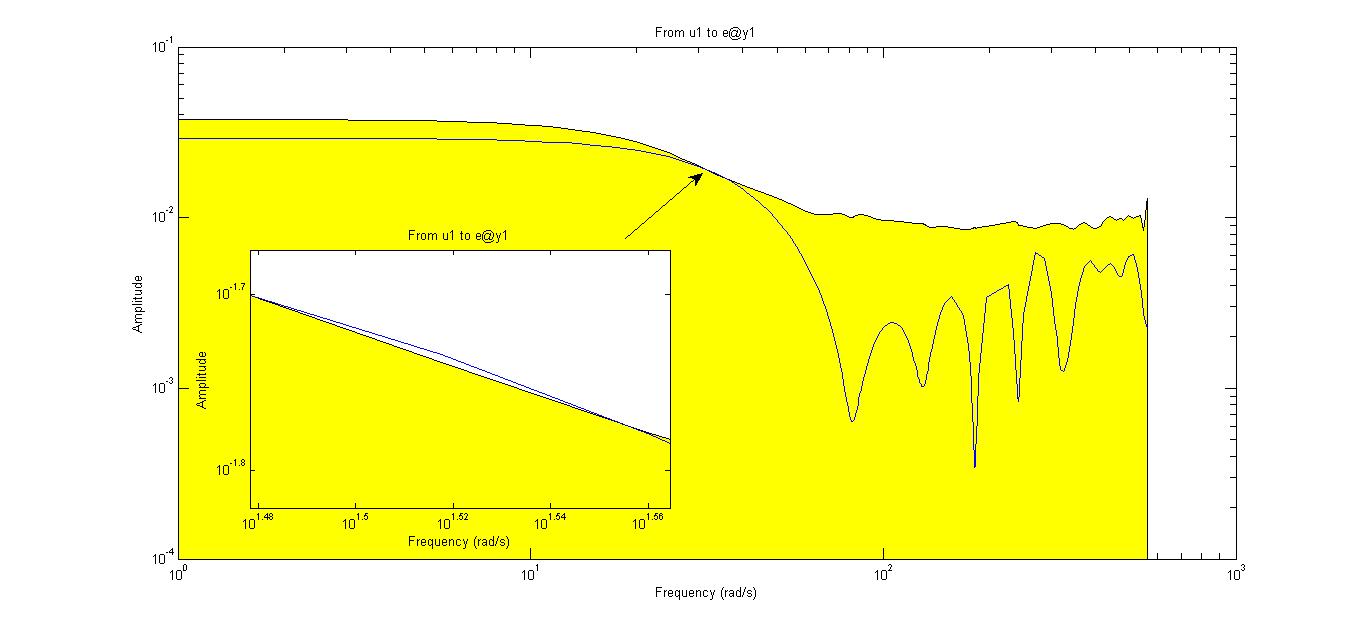
Тук съвпадението между модел и обект 87% (фиг. 3.9) и теста за бял шум на остатъчната грешка във времевата област се издържа (фиг.3.10). Всички тестове се издържат от модел ARMAX от 3-ти ред поради , което той се използва по-нататък в дипломната работа като линеаризиран модел на ППФП.



Фиг. 3.9 Тест за близост между модел и обект



Фиг. 3.10 Графика на остатъчната грешка за ARMAX(5, 5, 1)



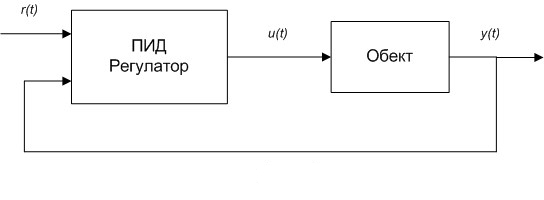
Фиг.3.11 Графика на остатъчната грешка в честотната област за ARMAX(5,5,1)

# 4. Проектиране на регулатори и филтри на Калман

## 4.1. Проектиране на ПИД регулатори

ПИД регулаторът е най-често срещания регулатор в практиката. ПИД регулаторите са важен компонент от инструментариума на всеки инженер по автоматично регулиране. Те са надживели много промени на технологията – от механиката и пневматиката до микропроцесорите през електронните лампи, транзисторите, интегралните схеми.

ПИД регулаторите са елементи в една САУ *(фиг. 4.1)* за реализиране на управляващо въздейст­вие *u(t)* върху обекта на управление с цел регулируемата величина *y(t)* да отработва по желан на­чин заданието *r(t)*. Имат постоянна структура, избрана сред известни схемни решения, независимо от вида на обекта на управление.

****

Фиг. 4.1

Операторен вид на класически аналогов ПИД регулатор с паралелна структура се представя със следното уравнение:

|  |  |
| --- | --- |
|  | (4.1) |

където:

– грешка (разлика) между заданието и регулируемата величина;

– коефициент на пропорционалност;

– времеконстанта на интегриране;

– времеконстанта на диференцираце.

Управляващото въздействие върху обекта на управление е сума от три съставки П (пропорционална на грешката), И (пропорционална на интеграла от грешката) и Д (пропорционална на производната от грешката).

### 4.1.1. Проектиране на аналогов ПИД регулатор

Първоначално за целите на управлението ще бъде описан реален аналогов ПИД регулатор [2].

#### 4.1.1.1. Пропорционална (П) съставка

Пропорционалната съставка произвежда управляващ сигнал *up(t)*, който е пропорционален на коефициента *Kp* на грешката на системата *e(t)*, формирана на неговия вход. Във времевата област управлението се описва с (4.2)

up(t) = Kp e(t) (4.2)

докато в операторен вид то се представя по следния начин (4.3)

up(p) = Kp e(p). (4.3)

Следователно П съставката се описва с предавателната функция (4.4)

Wp(p) = = Kp (4.4)

Работата на пропорционалната съставка влияе на регулируемата величина, като със увеличаване на коефициента на пропорционалност *Kp* статичната грешка в системата се намалява, но никога не се нулира затова е предложено добавяне на И съставка, която е описана в следващата точка. При това увеличаване на *Kp* се влошава устойчивостта на регулируемата променлива както и се увеличава пререгулирането.

Всеки физически сигнал се изменя в рамките на своите минимална и максимална стойност в конкретни физически единици. За да се избегнат големите систематични грешки в изчисляването на ПИД закона на управление е необходимо входните сигнали, да се преобразуват според своя мащаб. Някои сигнали, като заданието например, могат да имат наложени допълнителни ограничения на стойностите. Затова често се предлага пропорционалната съставка да се модифицира по следния начин (4.3)

up(p) = Kp (br(p) – y(p)). (4.5)

Тук заданието се претегля с коефициента b, чрез който се отразява степента на влияние на заданието върху П съставката. Коефициента регулира резките промени в заданието или напълно го изключва, като се препоръчват стойности между нула и единица.

#### 4.1.1.2. Интегрална (И) съставка

Възниква идеята за последователна (циклична) процедура за постепенно компенсиране на статичната грешка в съответствие със следния алгоритъм:

Стъпка 1: Измерва е y в момента *tk*

Стъпка 2: Изчислява се текущата за момента грешка

ek стат = е(tk) = r - y(tk)

Стъпка 3: Формира се пропорционална на тази грешка поправка в управлението

u (tk) = Gе (tk)

Стъпка 4: Отчита се следващия текущ момент *k + 1* и процедурата се повтаря.

Прието е представената стратегия да се нарича интегрално управление, в което коефициента на усилване на грешката се определя с израза:

G = T0,

при условие, че *T0* представлява такт на измерване на величините в схемата на управлението, *Kp* – коефициент на усилване на регулатора, *Ti* – времеконстанта на интегриране.

Нека управлението в целия интервал на действие на регулатора се описва от момента *t0* на включването му до момента *tk* на отработване на грешката в системата.

u(tk) = u(tk-1) + Gе(tk),

u(tk-1) = u(tk-2) + Gе(tk-1),

...

u(t1) = u(t0) + Gе(t1).

Тогава след последователно заместване се достига до

u(tk) = u(t0) + G[е(t1) + е(t2)+ … + е(tk)] = u(t0) + T0.

Ако *T0 → 0*, сумата се замества с интеграл, за да се представи като

u(tk) = u(t0) + + T0 = ui(t). (4.6)

а в операторен вид

u(p) = e(p) = ui(p). (4.7)

Беше установено :

* С П съставката се реализира усилване на постоянен сигнал на грешката в системата.
* С И съставката се осъществява постоянен управляващ сигнал дори в отсъствие на грешка води до изменение на *И* управлението с постоянна скорост докато се елиминира грешката в установен режим т.е. *eстат(И) = 0*.

С нарастване на коефициента *(Kp↑, Ti↓)* се ускорява отработването на грешката, но системата става по-малко устойчива. Този дестабилизиращ ефект може да бъде намален чувствително, ако в управляващото устройство се комбинират едновременно пропорционално и интегриращо действие(4.8).

u(t) = Kp e(t) + = up(t) + ui(t), (4.8)

а в операторен вид

u(p) = Kp e(p) + e(p) = up(p) + ui(p). (4.9)

ПИ регулаторът се представя с предавателната функция (4.10)

Wpi(p) = = Wp(p) + Wi(p) = Kp + = Kp (4.10)

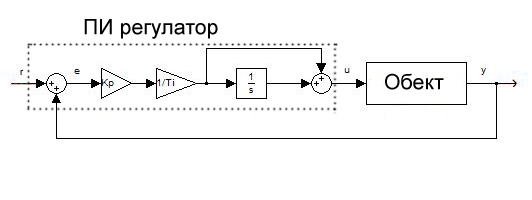
ПИ алгоритъмът реализира следното относително изменение (4.11) на управлението в момента *tk*

u(tk) = е(tk) +T0 е(tk) (4.11)

или

u = е + Kie, (K i= T0)

Структурната схема на САУ с ПИ регулатор е показана на фиг. 4.2.



Фиг. 4.2 Непрекъсната САУ с ПИ регулатор

При условие, че коефициентът на усилване на обекта е положителен, двете съставки на *u* (пропорционалната и интегралната) взаимодействат помежду си. В резултат от взаимодействието се наблюдава следният ефект:

* Когато *y* е много под *r*, грешката в САУ ще бъде толкова голяма, че интегралната съставка ще надхвърля пропорционалната. Управляващият алгоритъм ще увеличава стойността на *u*.
* Когато *у* се доближи до *r*, грешката в САУ става малка и пропорционалната съставка започва да превъзхожда интегралната. Настъпва промяна в *u*, така че управлението спира да расте и започва да намалява, докато *y* се изравни с *r*. Именно този механизъм предпазва от пререгулиране.
* В случай, че *y* надхвърли *r*, интегралната съставка в (4.11) ще бъде отрицателна. Тъй като *y* расте, промените в него са също положителни, така че и пропорционалната съставка в (4.11) е също отрицателна. В резултат от взаимодействието на двете отрицателни съставки управляващият алгоритъм ще предизвика намаляване на *u*. В този случай интегралната и пропорционалната съставка взаимно се подпомагат, именно тази комбинация от противопоставяне и съвместно действие на двете компоненти прави *ПИ* управлението много мощно.

#### 4.1.1.3. Диференциална (Д) съставка (ПИД управление)

При изчисляване на управляващия сигнал *u(t)* се добавя съставка, която да зависи от скоростта на изменение на грешката в САУ, но без да влияе на поведението на системата в установен режим.

Във времевата област ПИД управлението се представя с (4.12)

u(t) = Kp e(t) + + KpTd = up(t) + ui(t) + ud(t), (4.12)

където *Td* се нарича времеконстанта на диференциране.

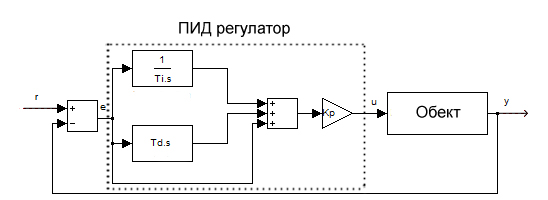
В операторен вид закона за управление се представя с (4.13)

u(p) = Kp e(p) + e(p)+ KpTde(p) = up(p) + ui(p)+ ud(p). (4.13)

Тук структурата на *Д* частта се модифицира, както при *П* частта с добавяне на тегловен коефициент *c* на заданието, с помощта който се регулират резки промени в заданието. Влиянието на високочестотните смущения в регулирания сигнал се ограничава с нискочестотен филтър от първи ред с времеконстанта *Tf* *= Td / N*. Коефициента *N* се избира в интервала [1, 30]. Така се получава реален *ПИД* регулатор представен по следния начин (4.14)

(4.14)

Структурната схема на САУ с ПИД регулатор е показана на фиг. 4.1.3.1

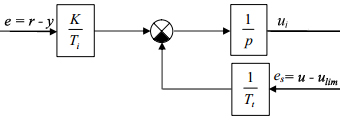


Фиг. 4.3 САУ с ПИД регулатор

#### 4.1.1.4. Механизъм против интегрално насищане (anti-windup)

В много практически приложения стойността на управляващият сигнал е ограничена от физически съображения: един клапан не може да бъде отворен повече от конструктивните му възможности, един тиристорно управляван нагревател не може да даде отрицателна по знак топлинна енергия. Когато управляващият сигнал надхвърли максималните възможности на изпълнителния механизъм в рамките на *[ulow, uhigh]*, регулаторите с интегриращо действие започват да работят неефективно. Ако заданието изисква промяна на управлението, *САУ* започва да функционира с голяма грешка и интегралната съставка на регулатора започва да нараства. Съответно управлението и регулируемата величина също непрекъснато нарастват, докато максимално възможното (100%) въздействие върху обекта на управление, ненадхвърлящо съществуващите физически ограничения, се реализира, без заданието все още да е достигнато. От този момент регулаторът чрез интегриращата съставка започва да акумулира в управлението енергия, която реално не въздейства върху обекта с достатъчна мощност поради ограниченията, но с времето способства регулируемата величина да продължи да нараства, достигне и дори подмине заданието. От този втори момент интегралната съставка поради голямата акумулирана стойност, която не успява да се редуцира за кратко време, продължава да държи управлението в максимална стойност, независимо, че текущата регулируема величина вече е надхвърлила заданието. Този ефект се нарича интегрално насищане (integral saturation), пренавиване (wind-up) и води до лошо поведение на регулатора извън рамките на ограниченията на управляващия сигнал, защото практически за известно време системата функционира в отворен контур т.е. благоприятното влияние на обратната връзка е елиминирано.

Използваното решение против това явление е схемно [1, 2]. Реализира се допълнителна обратна връзка (ОВ) с тегло *1 / Tt*, която се включва към интегриращия грешката контур в основната схема на регулатора. Това води до преизчисляване на И съставката, така че управлението да остава винаги в рамките на ограниченията (фиг. 4.4).



Фиг. 4.4 И – съставка със схемно антиинтегрално насищане

Отрицателната ОВ е изградена по отношение на отклонението *es* между управляващия сигнал *u* и ограничението на изпълнителния механизъм *ulim*. На входа на интегратора от схемата действа сигнал *es = e - es*. в установен режим той се нулира и може да се изведе зависимостта *es = e*. Следователно управлението приема стойности *u = ulim + (r - y)*, които не надхвърлят *ulim* при поява на пререгулиране в *САУ* (*y > r*). скоростта на възобновяване на *И* съставката зависи от тегловния коефициент *Tt*, който може да се приеме като времеконстанта на възобновяването.

### 4.1.2. Проектиране на цифров ПИД регулатор

Основното предназначение на разработваната система за поддържане на концентрацията на субстрата е нейната реализация върху цифров микроконтролер, а именно Arduino Due. Оттук произлиза необходимостта от проектирането на цифров вариант на непрекъснатия ПИД регулатор. Цифровите (дискретните) регулатори формират управление само в моменти от времето, кратни на такта на дискретизация *T0* на базата на получен сигнал от обратната връзка също в дискретни моменти [2]. Получаването на цялостен дискретен еквивалент е предхождано от поетапно дискретно апроксимиране на отделните съставки на аналоговия *ПИД*.

#### 4.1.2.1. Апроксимиране на П – съставката

Тази апроксимация се извършва по зависимостта:

,

или , (4.15)

където *e(k) = br(k) – y(k)* е разсъгласуването в системата, формирано в дискретни моменти от време, *b* – тегловен коефициент в заданието, *r(k)* – задаващ сигнал, *y(k)* – регулируема величина. И така за *П* – частта от управлението получаваме

up(k) = Kp(br(k) - γs(k)), (4.16)

където *γs(k)* е регулируемата величина (концентрация на субстрата [g.l-1]).

#### 4.1.2.2. Апроксимиране на И – съставката

Апроксимацията се извършва не върху самата И – съставка, а върху нейната производна при нулеви начални условия в съответствие с израза (4.17)

ui(t) = T0 => = e(t), ui(0) = 0 (4.17)

Тук диференциалът *dui(t)* в текущия момент *t = kT­0* се изчислява въз основа на стойностите на управлението в *k* – тия и (к-1) – ия момент (обратна първа разлика). Тогава (4.18)

= t = kT0 => = (4.18)

или ui(k) = ui(k-1) + bi1e(k),

където коефициентът *bi1 = T0* , а *e(k) = r(k) - γs(k)* грешката от разсъгласуване, *γs(k)* – изход на системата (концентрация на субстрата [g.l-1]).

При преход от непрекъснат в цифров вариант е нужно модифициране на anti-windup механизма. За целта И-съставката се модифицира по следния начин:

I(k) = I(k-1) + Kie(k) + – u(k-1), ν = ,

където *ulow* и *uhigh* са съответно долната и горната граница на ограничителния блок saturation. По този начин при ниво на управление в работния диапазон, И съставката запазва стандартната си структура, а при насищане променливата *ν* подпомага бързото връщане на *ui* в допустимите граници.

#### 4.1.2.3. Апроксимиране на Д – съставката

Аналогично на И – съставката  в текущия момент *t = kT­0* се изчислява въз основа на стойностите на грешката в k-тия и (k-1)-ия момент. Тогава

,

или  (4.19)

Операторът *q* притежава свойството да премества във времето стойностите на произволен сигнал *f(k)* в съответствие с правилото *q{f(k±i)} = q±if(k)*.

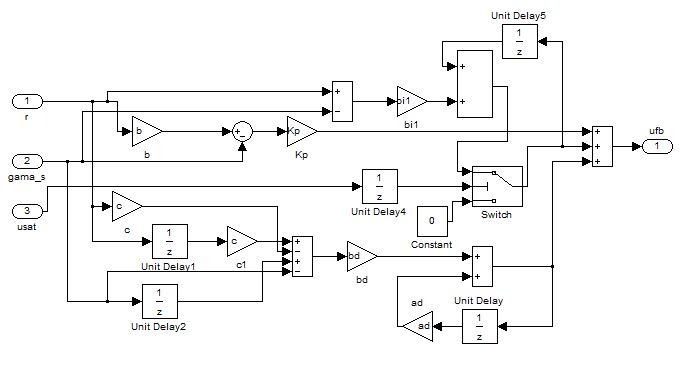
След добавяне на тегловния коефициент *c* и филтърът в Д – съставката получаваме следното диференчно уравнение (4.20)

ud(k) = adud(k -1) + bd(cr(k) – cr(k-1)γs(k) - γs(k-1)) (4.20)

Коефициентите *ad* и *bd* се определят от параметрите на непрекъснатия регулатор посредством изразите:

ad = , bd = (4.21)

Структурата на цифровия ПИД регулатор с алгоритъм за антиинтегрално насищане е дадена на фиг. 4.5



Фиг. 4.5 Цифров ПИД с две степени на свобода и anti-windup

И така алгоритъмът за управление от цифровия ПИД има следния вид

ufb(k) = up(k) + ui(k) + ud(k) (4.22)

## 4.2. Feed Forward съставка на управлението

Частта от управляващия сигнал *uff*, която поддържа процеса в текущата работна точка се определя от условието за равновесие на нелинейното уравнение (4.23)

= - - = 0 =>

= (4.23)

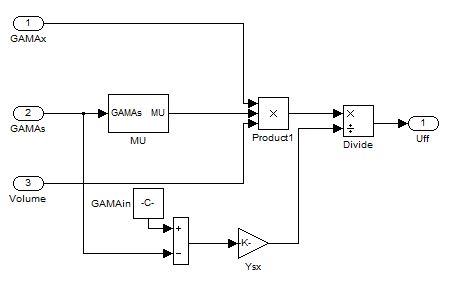
И така получаваме следния вид на управляващия сигнал uff (4.24)

uff (k) = , (4.24)

където се дава с модела на Моно

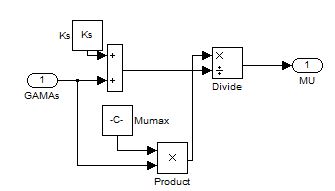
µ(к) = µmax(к) (4.25)

Схемната реализация на тази съставка е показана на фиг. 4.6



фиг. 4.6 FeedForward съставка на управлението

Входните сигнали постъпват от изхода на разширения филтър на Калман (РФК), а MU (специфична скорост на растеж на микроорганизмите) има следната структура (фиг. 4.7)



фиг. 4.7 Simulink схема на модела на Моно

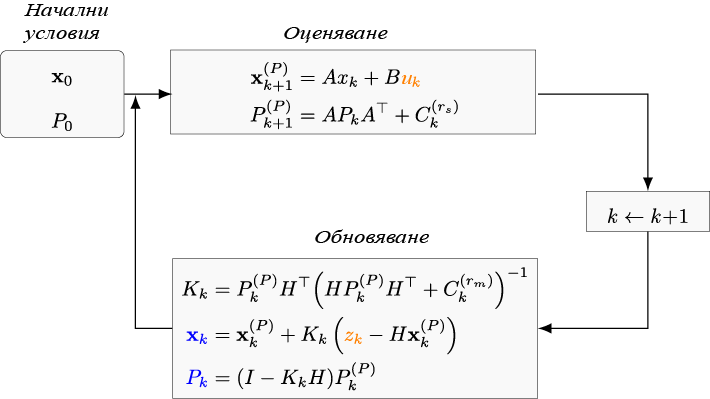
От двете съставки на управлението се вижда, че за реализиране на управлението са необходими измервания на всички променливи на състоянието. Единствено концентрацията на субстрата се измерва в реално време. За попълване на липсващата информация е синтезиран нелинеен филтър на Калман (Разширен Филтър на Калман РФК).

## 4.3. Синтезиране на разширен филтър на Калман

Ако векторът на състоянието е недостъпен за измерване за всяко време , обратната връзка се реализира по оценката на състоянието . Оценката се изчислява по измеримия изход и управлението с помощта на динамична система (наблюдател на състоянието). Тя попълва липсващата информация за вектора на състоянието.

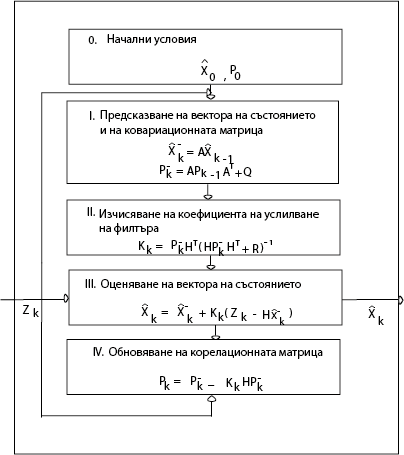
В настоящата разработка наблюдателят на състоянието е нелинейният разширен филтър на Калман. Той представлява рекурсивен предсказващ филтър (алгоритъм), който прави оценка на състоянието на системата. Алгоритъмът се състои основно от две стъпки (фиг. 4.8):

1. Предсказване
2. Корекция (Обновяване)



Фиг. 4.8 Алгоритъм на филтъра на Калман

В първата стъпка се предсказва състоянието на динамичната система посредством уравнението на състоянието, докато във втората стъпка се прави корекция на оценката получена от стъпка 1 посредством уравнението на наблюдението. В резултат, на което ковариационната матрица на оценителя се минимизира. В този смисъл филтърът на Калман е оптимален оценител. Тази процедура се повтаря на всяка стъпка, като за начални стойности използват изчислените от предишната стъпка. Следователно филтърът на Калман може да се нарече рекурсивен филтър. За по-добра нагледност алгоритъмът на филтъра е представен на следната блок схема (фиг. 4.9)



Фиг. 4.9. блок схема на филтъра на Калман

Алгоритъмът получава вектор с входните променливи *zk* – управлението и изхода на системата и връща оценките на състоянието. Процедурата се реализира в 4 стъпки.

Стъпка 1: Изчисляват се предсказаната стойност на оценката на вектора на състоянието *к-* и ковариационната матрица *P-k* които се използват в следващите стъпки. Индексът „-“ означава предсказана стойност.

Стъпка 2: Изчислява се коефициента на РФК като се използва изчислената в стъпка 1 ковариационна матрица *P-k*.

Стъпка 3: Изчислява се оценката на вектора на състоянието на базата на управлението и измеримия изход от системата и на базата на коефициента на РКФ получен в стъпка 2.

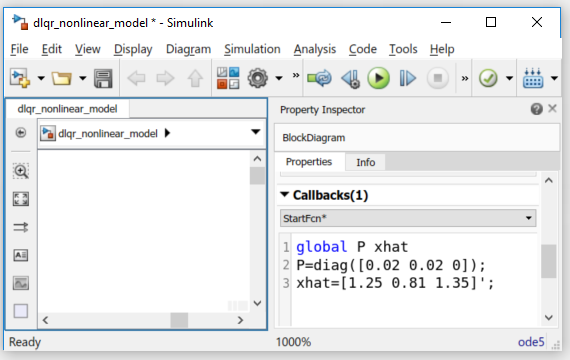
Стъпка 4: Изчислява се ковариационната матрица Pk, която е мярка за това колко точна е оценката.

С цел симулация на изследваната система РФК е разработен в средата на Matlab/Simulink. За целта се дефинира собствена функция EKF, с която се реализира алгоритъма от фиг. 4.9. Примерен вид на функция е даден в таблица 4.1

Таблица 4.1

|  |
| --- |
| function xhatOut = EKF (pinput) |
| %Инициализация на началните условия като глобални, които се извикват |
| %при симулацията с натискане на бутона start чрез callback функция |
| %initialize.m |
| %global P xhat |
| %xhat = [1.25; 0.81; 1.35]; |
| % P = diag([0.02 0.02 0]); |
| global P xhat |
| %Инициализация на променливите като глобални |
| global Ks Ysx GAMAin MUmax T0 H |
| %Входни параметри за функцията са измеримия изход и управлението |
| gamaSmeasured = pinput(1); |
| Q = pinput(2); |
| % |
| gamaX = xhat(1); |
| gamaS = xhat(2); |
| Volume = xhat(3); |
| %Изчисляване на Якобианите |
| a11 = ( MUmax \* gamaS ) / (Ks + gamaS) - Q / Volume; |
| a12 = ( MUmax \* gamaX \* ( Ks + gamaS) ) / ((Ks + gamaS)^2); |
| a13 = ( Q \* gamaX ) / Volume; |
| a21 = - (MUmax \* gamaS) / ( Ysx \* ( Ks + gamaS ) ); |
| a22 = - (MUmax \* gamaX \* Ks) /((Ks + gamaS)^2) - Q / Volume; |
| a23 = (Q \* (GAMAin - gamaS)) / (Volume^2); |
| a31 = 0; |
| a32 = 0; |
| a33 = 0; |
| F=[a11 a12 a13; |
| a21 a22 a23; |
| a31 a32 a33; |
| ]; |
|  |
| % Определяне на матрицата F |
| F = eye(3) + T0\*F; |
| %Диверсионна матрица на шумовете |
| Deta = T0^2\*diag([0.001 0.001 0]); |
| %дисперсия на измервателния шум |
| Dceta = 0.0025; |
|  |
| % 1. Предсказване на вектора на състоянието и ковариационната матрица |
|  |
| Ppredicted = F\*P\*F' + Deta; |
|  |
| Mu = (MUmax\*gamaS)/(Ks+gamaS); |
| X = Mu\*gamaX-Q/Volume\*gamaX; |
| S = -1/Ysx\*Mu\*gamaX+Q/Volume\*(GAMAin-gamaS); |
| Fpredicted =xhat+0.0056\*[X S Q]'; |
|  |
| % 2. Изчисляване на коефициента на Калман |
| Kekf = Ppredicted\*H'\*inv(H\*Ppredicted\*H'+Dceta); |
|  |
| % 3. Изчисляване на вектора на състоянието |
| xhat = Fpredicted+Kekf\*(gamaSmeasured-H\*Fpredicted); |
|  |
| % 4. Обновяване на ковариационната матрица |
| P = (eye(3) - Kekf\*H)\*Ppredicted; |
| %Извеждане на оценките |
| xhatOut = xhat; |

На всяка следваща итерация се използват изчислените стойности от стъпка 3 и стъпка 4 като на първа итерация се ползват началните условия. Инициализацията на началните условия като глобални, които се извикват при симулацията с натискане на бутона start чрез callback функция (фиг. 4.10). Тя се реализира чрез т. нар. property inspector, който се включва от менюто View > Property Inspector или чрез бързия клавиш Ctrl + Shift + I, след което от секцията Callbacks се избира от падащото меню StartFcn и в полето по-долу се задават началните условия на ковариацинната матрица и на вектора на състоянията.



Фиг. 4.10 Задаване на началните условия като Callback функция

## 4.4. Оптимизационна процедура за настройка на ПИД регулатор базирана на генетичен алгоритъм

Генетичните алгоритми (ГА) са стохастичен метод за глобално търсене и оптимизация, който имитира еволюцията на живите индивиди, описана от Чарлз Дарвин в “За произхода на видовете и значението на естественият подбор”.

В еволюционните алгоритми се използват трите основни принципа на естествената еволюция, описани от Дарвин: репродукция, естествен подбор и разнообразие на индивидите, поддържано чрез разликите на всяко поколение с предишното. През 60-те години на 20 век тези 3 характеристики на естествената еволюция вдъхновяват европейски и американски изследователи да създадат независими един от друг методи за стохастично търсене:

* Еволюционно програмиране
* Еволюционни стратегии
* Генетични алгоритми

При трите метода се работи с набор от индивиди. Принципа на подбора се прилага, като се използва критерий, даващ оценка за близостта на индивида с желаното решение. В следващото поколение продължават най-добре приспособените индивиди. Ще бъде представена кратка аналогия между живите организми и генетичните алгоритми.

***Хромозоми***

Всички живи организми се състоят от клетки. Едни от органичните съединения изграждащи клетките са нуклеиновите киселини. Те биват два вида: ДНК и РНК. Хроматинът, съдържащ се в ядрото е изграден от ДНК, белтъци и РНК. При делене на клетката той се уплътнява образува спирални нишки – хромозоми.

Хромозомите са разположени гени, които носят наследствените белези на клетката. Всеки ген кодира конкретен протеин и представлява самостоятелен фактор на генетичната информация, който обуславя изявата на определени белези. Условно може да се каже, че всеки ген кодира белег (примерно цвят на очите). Всеки ген има своя позиция в хромозомата. Тази позиция се нарича място.

Възможните комбинации за белега (т.е. синьо, кафяво) се наричат алели, а сборът от всички гени в организъма – геном.

Съвкупността от всички наследствени фактори на организма се нарича генотип. Това представлява съвкупността от всички локализирани в хромозомите гени на организма.

Съвкупността от проявените белези и свойства на организма се нарича фенотип. Той зависи от наследствените качества (генотипа) и влиянието на средата.

При генетичните алгоритми хромозомите представляват набор от гени, които кодират неизвестните променливи. Всяка хромозома представлява допустимо решение на поставената задача. Индивид и вектор на променливите ще бъдат използвани като понятия еквивалентни на хромозома.

Множеството от различни хромозоми (индивиди) съставят текущото поколение. Чрез еволюционни операции, като селекция, рекомбинация и мутация, се достига до следващото поколение (потомство).

***Селекция***

B природата селекцията на индивиди се извършва чрез естественият подбор. Колкото по-приспособен е даден индивид към заобикалящата среда, толкова по-голям е шансът му да оцелее и да създаде потомство, като по този начин предаде генетичната си информация на следващото поколение.

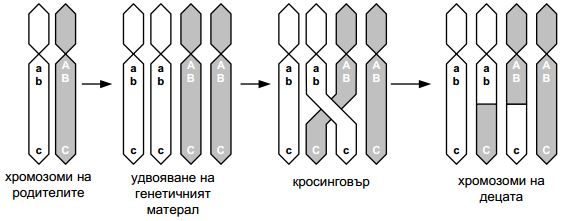
При еволюционните алгоритми селекцията на най-добрите индивиди става въз основа на функционал или функционали даващи оценка на конкретният индивид. Например такъв функционал може да е квадратичен критерий на грешката между изходите на желана от нас система и реалната, близост на полюсите на затворена система до желаните и т. н. Ако задачата е за минимиза­ция, то индивидите с по-малка стойност на функционала ще имат по-голям шанс да бъдат избрани за рекомбинация и съответно за продължаване на поколението.

***Рекомбинация***

B природата деленето на клетките бива два вида:

* Амитоза (просто) – при него наследственият материал не се разпределя равномерно между дъщерните клетки;
* Митоза (сложно) – при него генетичния материал се удвоява преди деленето, при което двете дъщерни клетки получават еднаква наследствена информация.

При репродукцията, първо се появява рекомбинацията (кръстосване или кросинговър) фиг. 4.11. При нея гените от родителите формират по някакъв начин изцяло нова хромозома.



Фиг. 4.11 Кръстосване при митоза

Типичната рекомбинация при генетичните алгоритми е операция, изискваща два родителя (възможни са и схеми с повече родители). Два от най-често използваните алгоритми са стандартно кръстосване и кръстосване със смесване.

Стандартно кръстосване

При този тип рекомбинация родителите си разменят съответни гени. Кръстосването може да бъде едноточково или многоточково. За рекомбинацията се използва битова маска *Mask*.

Кръстосване със смесване

Математическото описание на този тип кръстосване е:

C1 = γ.P1 + (1-γ).P2

C2 = (1-γ).P1 + γ.P2

γ = (1+2.α ).r – α

P1, P2 – хромозоми на родителите

C1, C2 – хромозоми на децата (резултантни индивиди)

α - коефициент на изследване – задава се от потребителя

r – случайно число в границите (0, 1)

С помощта на *α* се променя областта, в която може да попадне стойността на резултантният ген. При *α = 0* се гарантира, че стойността на резултантният ген ще бъде между тази на родителите, а при по-големи стойности може да се изследват съседни области.

***Мутация***

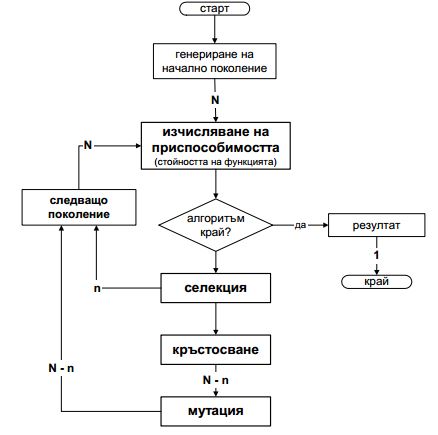
Новосъздаденото чрез селекция и кръстосване потомство след това може да бъде подложено на мутация. Мутация означава, че елементи от ДНК се променят. Тези промени са породени главно от грешки при копирането на гените от родителите.

В термините на генетичните алгоритми мутация означава произволна промяна на стойността на ген в поколението. Хромозомата, чийто ген ще бъде променен и мястото на гена също се избират по случаен принцип.

**Обща схема на еволюционните алгоритми**

Еволюционните алгоритми поддържат популация от индивиди (хромозоми), които еволюират чрез използване на селекция и други операции като кръстосване и мутация. Всеки индивид в популацията получава оценка за приспособимостта си (fitness function) към средата. В термините на оптимизацията това означава, че за всеки набор от променливи се изчислява стойността на функцията, която се минимизира или максимизира. Селекцията служи за избор на най -добрите комбинации, които чрез кръстосване и мутация трябва да доведат до по-добри решения в следващото поколение.

На фиг. 4.12 е показана една от най-често използваните схеми на генетичните алгоритми.



Фиг. 4.12 Обща схема на генетичните алгоритми

1. Създаване на начално поколение – при повечето алгоритми първото поколение се генерира по случаен принцип – гените на отделните хромозоми се избират случайно измежду азбуката на допустимите гени. Заради по-лесната изчислителна процедура се приема, че всички поколения се състоят от еднакъв брой индивиди (*N* на брой).

2. Следващата стъпка е изчисляване на стойността на функцията, която минимизираме или максимизираме.

3. Проверка за край на алгоритъма – както при всички алгоритми за оптимизация и тук е възможно алгоритъмът да бъде прекратен по:

* Стойност на функцията – стойността на функцията на най-добрия индивид става достатъчно близка до зададена стойност. Обикновено не се препоръчва използването само на този критерий, защото поради стохастичният характер на търсенето не може да се гарантира достигане до желания екстремум в разумно време.
* Максимален брой итерации – това е най-често използвания критерий за спиране. Той гарантира, че независимо дали алгоритъмът е достигнал до екстремум или не, ще спре след определено време.
* Достигане на установена стойност – ако в продължение на предварително зададен брой итерации (поколения) не е настъпило подобрение на стойността на функцията алгоритъмът спира.

4. Селекция – измежду всички индивиди в текущото поколение се избират тези, които да продължат развитието си в кръстосването и мутацията. На този етап може да се използва елитарен подход. Това означава част от най-добрите индивиди (*n* на брой) да се прехвърлят без промяна в следващото поколение. По този начин се гарантира, че достигнатата стойност на функцията не може да се влоши (веднъж достигнат екстремумът няма да бъде изпуснат).

5. Кръстосване – избраните чрез селекция индивиди се кръстосват. По този начин се получават нови индивиди, като стремежът е тези индивиди да наследят възможно най-добрата комбинация от характеристики на родителите си.

6. Мутация – чрез случайна промяна на някои от гените се гарантира, че дори нито един от индивидите в текущото поколението да не съдържа необходимият ген, пак е възможно да се достигне до екстремум.

7. Ново поколение – избраните от селекцията индивиди се обединяват с получените чрез селекция и мутация и образуват следващото поколение.

В дипломната работа е използвана стандартната функция *ga* от *Global Optimization Toolbox* на *Matlab*, която изисква от потребителя да се напише стандартна функция, в която на всяка итерация от оптимизацията се формира целевата функция.Като целева функция отразяваща качеството на системата за управление могат да се използват различни показатели с интегрален характер. Тези показатели обхващат различни математически описания, но всички те зависят поне от един фактор, наричан основна грешка *e(t)* в критерия, като съществуват различни начини за нейното формулиране.

Оптималното настройване на регулатора е задача за минимизиране на скаларна нелинейна функция на *n* променливи в съответствие с израза

minxf(x) => xоптимални.

Най-често използваните основни показатели се дават с изразите:

**А.** Показател на абсолютната грешка  .

**Б.** Показател на квадратичната грешка  .

**В.** Показател на претеглената с *t* абсолютната грешка 

**Г.** Показател на претеглената с *t* или ** квадратичната грешка  или .

Разгледаните показатели могат да бъдат разширявани (допълвани) с още един фактор – или *Δu(t)*, чрез който да се отчита влиянието на отклонението на управляващата променлива *u(t)* от някаква установена (ограничена) стойност, или *u(t)*, чрез който се отчита текущото поведение на управлението. Така напр. *ISE* може да приеме следния разширен вид

, (*α <* 1),

като съотношението между двата фактора в подинтегралната функция се определя от стойността на тегловния коефициент *α* пред втория фактор: колкото по-голям е *α*, толкова по-малко се изменя *u(t)*. По този начин в проектираната САУ не се допуска управление с твърде големи амплитуди, но грешката *e(t)* се отработва относително бавно.

На практика при компютърна реализация на оптимизационната процедура, независимо дали регулаторът е аналогов или цифров, описаните показатели се преобразуват така, че интегралът се заменя със сума при фиксирана горна граница на изменение на величините. Така *ISE* с допълнителния фактор (маркиран в името с индекса \*) се модифицира посредством израза

.

като  представлява такта на дискретизация на сигналите за компютърното им обработване.

В използваната оптимизационна схема (фиг. 4.18) се сравнява оптимизируемата посредством ПИД регулатора регулируема величина *y(t)* (изходът на обекта) с желан процес на поведение на затворената САУ, напр. заданието *r(t)*. Грешката *e*(*t*) в описаните по-горе показатели на качеството на оптимизацията съвпада с грешката в управлението на *САУ *:

*e*(*t*) = *r*(*t*) *- y*(*t*) = .

При оптимизацията с генетичен алгоритъм се използва целевата функция

Iise =

и нелинейния модел на биотехнологичния процес в съответствие със схемата, показана на фиг. 4.13. В резултат на оптимизацията за параметрите на регулатора се получава:

Kp = 0.6502, Ti=0.0556, Td=0.5184, b=0.2352, c=0.7181, N = 0.4832



Фиг.4.13 Схема на оптимална настройка на ПИД регулатор по ГА

След настройката на регулатора са показани получени следните резултати съответно за концентрацията на субстрата (фиг. 4.14), дебита на подхранващия разтвор (фиг.4.15), концентрацията на биомаса (фиг.4.16) и обема на биореактора (фиг.4.17)



Фиг.4.14 Концентрация на субстрата Фиг.4.15 Дебит на подхранващия разтвор

Вижда се успешното поддържане на концентрацията на субстрата на зададеното ниво 0.1.



Фиг. 4.16 Концентрация на биомаса Фиг. 4.17 Обем на биореактора

## 4.5. Оптимизационна процедура за настройка на ПИД регулатор използваща линеаризиран модел

При използване на линеаризирания модел на обекта, получен със ср­ед­стват­а на идентификацията, ПИД регулаторът може да бъде настроен с конвенционален метод за оптимизация. Оптимизацията се осъществява с помощта на функцията *fminsearch* на *Optimization Тoolbox* на *Matlab,* съгласно схемата показана на фиг. 4.18.



Фиг. 4.18 Оптимално настройване по грешката в управлението

Изпълняват се следните стъпки за извършване на оптимизационната процедура:

Стъпка 1: Използва се проектираният регулатор в точка 4.1.2. настройваеми коефициенти .

Избраните за начални параметри са

Стъпка 2: Изгражда се блок-диаграма на САУ от *фиг. 4.18*

Стъпка 3: Определя се сигналът на грешката с САУ. За целта се дефинира собствена функция от *Matlab* под името *cls\_er\_1*, чрез която се описва симулирането на САУ и се изчислява грешката . Функцията е представена в *табл. 4.2*.

|  |
| --- |
| Табл. 4.2. Функция на грешката в показателя (*cls\_er\_1*) |
| %Формиране на грешката в показателя за оптимизиране  function f = cls\_er\_1(pid)  %оптимизируемите променливи се дефинират като глобални  global Kp Ti Td b c bi1 bi2 ad bd e1 u  %pid - вектор на оптимизируемите променливи  %f - сигнал на грешката в целия интервал на наблюдение T  %S - такт на измерване  k = pid(1); ti = pid(2); td = pid(3);  %Симулиране на затворената система с име s\_cls\_1 чрез  %функцията sim в интервала от 0 до 7 сек.  sim(‘modelOptim’, 7);  e = е1;  %оформяне на грешката  f=e\*(e')+ u'\*u;  %Край на функцията |

Стъпка 4: Използва се функцията *fminsearch*, така че квадратичния критерий от грешката на управлението да бъде минимален. Съставя се главна програма за реализирана на оптимизацията показана в *табл. 4.3*.

|  |
| --- |
| Табл. 4.3. Главна програма за реализация на оптимизационната процедура |
| %Настойваемите параметри се дефинират като глобални  global Kp Ti Td b c bi1 bi2 ad bd e1 u  %Начални стойности на настройваемите параметри  k0 = 1; ti0 = 1.45; td0 = ti0/4;  pid = [k0 ti0 td0]  %настройки за функцията fminsearch  options = optimset('Diagnostics','on', 'display','iter');  %Изчисление на оптималните параметри на регулатора чрез fminsearch  pid = fminsearch('cls\_er\_1', pid ,options); |

Стъпка 5: Провежда се оптимизационната процедура.

Началните стойности на параметрите на регулатора са *K0 = 1*, *Ti0 = 1.45*, *Td0 = Ti0 / 4*, а стойностите на тегловните коефициенти и филтъра в Д-съставката са взети от точка 4.4 *b = 0.2352*, *c = 0.7181*, *N = 0.4832*. Процедурата стартира при начална стойност на функцията *f(x) = 0.786056* и приключва след 78 итерации със стойност на *f(x) = 0.0449583*. Получените оптимални параметри на регулатора са съответно *Kp = 1.0703*, *Ti = 0.0242*, *Td = 0.4397*.

С така настроения ПИД регулатор и синтезирания разширен филтър на Калман е изградена симулационна схема на системата за управление на биотехнологичния процес, показана на фиг.4.18.

Получените резултати за дебита на подхранването и концентрацията на субстрата са представени на фиг. 4.19 и фиг. 4.20



Фиг. 4.19 Дебит на подхранващия разтвор и концентрация на субстрата



Фиг. 4.20 Обем на биореактора и концентрация на биомасата

## 4.6. Проектиране на линейно – квадратичен гаусов регулатор

### 4.6.1. Проектиране на линейно – квадратичен регулатор с интегрална съставка

При формиране на показателя на качеството в практиката често се налага да се вземе предвид и възможност за ограничаване на амплитудата на управляващото въздействие в хода на управлението. Затова един от най-често използваните показатели на качеството косвено отчитащи бързодействието на системата и амплитудните ограничения на управляемите величини и управляващите въздействия, е квадратичния функционал от вида (4.26) [9]:

*J =* , (4.26)

където *Q* и *R* са квадратни положително определени тегловни матрици. Първият член в подинтегралния израз на функционала отчита текущата стойност на вектора *x(t)*, респективно управляемата величина *y(t)*, докато втория член отчита „мощността“ на управлението. Ако синтезът се извършва от условието за минимум на функционала (4.26), съответните компоненти в него водят до намаляване на отклонението на управляемата величина от нулевата стойност, което косвено води до ускоряване на процесите в системата и намаляване на енергията, която се изразходва за управление. При дискретни системи функционала има аналогичен вид (4.27):

*J =* (4.27)

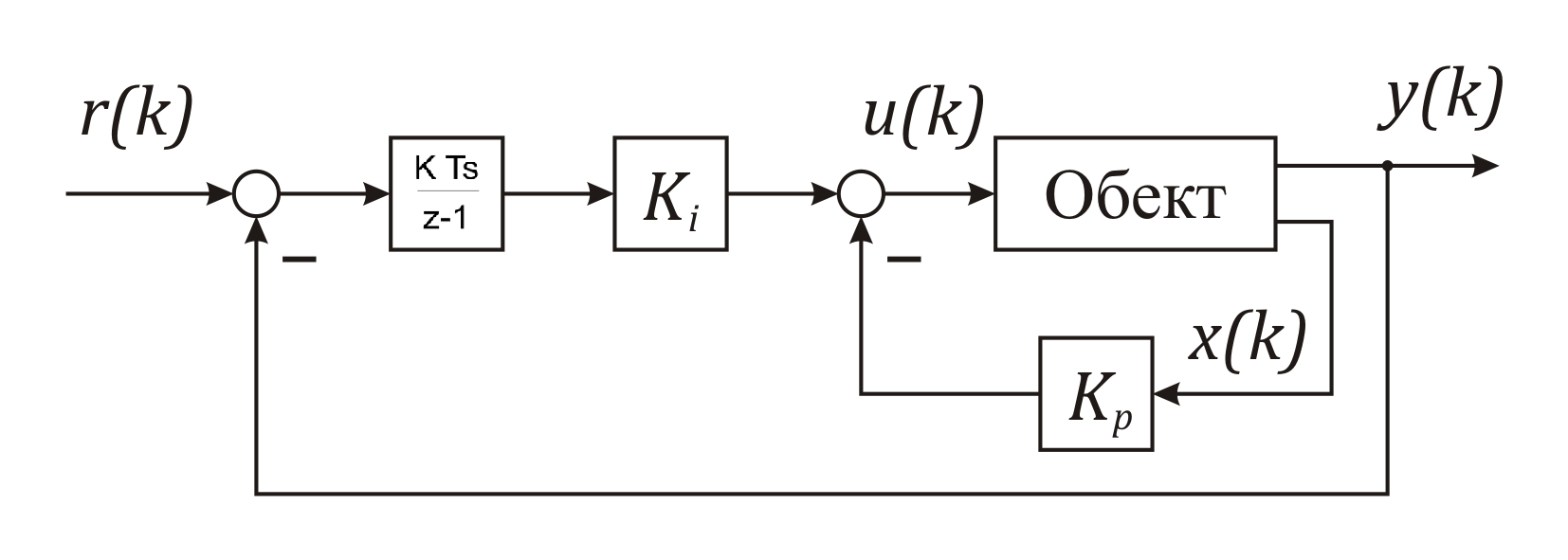
Матриците се избират от съображения за равностойно / неравностойно участие на управляемите величини, променливите на състоянието и управляващите въздействия. Най-често те се избират диагонални.

Структурната схема на модела на обекта описан със уравненията (4.28):

x(k+1) = Fx(k) + Gu(k) (4.28)

y(k) = Cx(k)

е показана на фиг. 4.21.



Фиг. 4.21 Структурна схема на система с LQR регулатор

Регулаторът на нулевото състояние на системата (4.28) е реализиран като обратна връзка по състояние (4.29):

u(k) = -Kpx(k) (4.29)

Матрицата *Kp* на обратната връзка, която минимизира квадратичният показател (4.27) се определя от съотношението (4.30) [9]:

Kp = (R + GT P G)-1 GT P F, (4.30)

където *P* е положително определено решение на уравнението на Рикати (4.31):

FT P F – F - FT P G (R + GT P G)-1 GT P F + Q = 0 (4.31)

За намаляване на статичната грешка и подобряване на точността в установен режим се включва интегрална съставка. Сигналът *xi* на изхода на интегратора, който е включен в правия тракт на системата е умножен с коефициент *ki*и е добавен към управляващия сигнал, формиран от обратната връзка по състояние.

Системата с добавен интегратор се описва със следните уравнения (4.32)

x(k+1) = Fx(k) + Gu(k)

y(k) = Cx(k) (4.32)

u(k) = -Kpx(k) + kixi(k),

където

xi(k + 1) = xi(k) + Т0е(к) = xi(k) + Т0(r(k) – Cx(k)) (4.33)

И така за разширената с интегратор система в матричен вид можем да запишем (4.34):

= + u(k) + r(k)

u(k) = = -Kp (4.34)

С цел симулация на системата регулаторът е синтезиран в средата на Matlab. По-горе в глава 3.2 беше намерен линеен дискретен линеен параметричен модел ARMAX от 3-ти ред, който е преобразуван към модел в пространство на състоянията с помощта на функцията *ss* от пакета с инструменти Control System Toolbox като към този модел е включен и шума с подаването на параметъра ‘augmented’.

m=armax(datae,[3 3 3 1]);

sys\_ss = ss(m,'augmented');

Оттук разширените матрици имат следния вид:

A1=[1 -T0\*C; zeros(3,1) A]; B1=[0;B];

A1 = B1 =

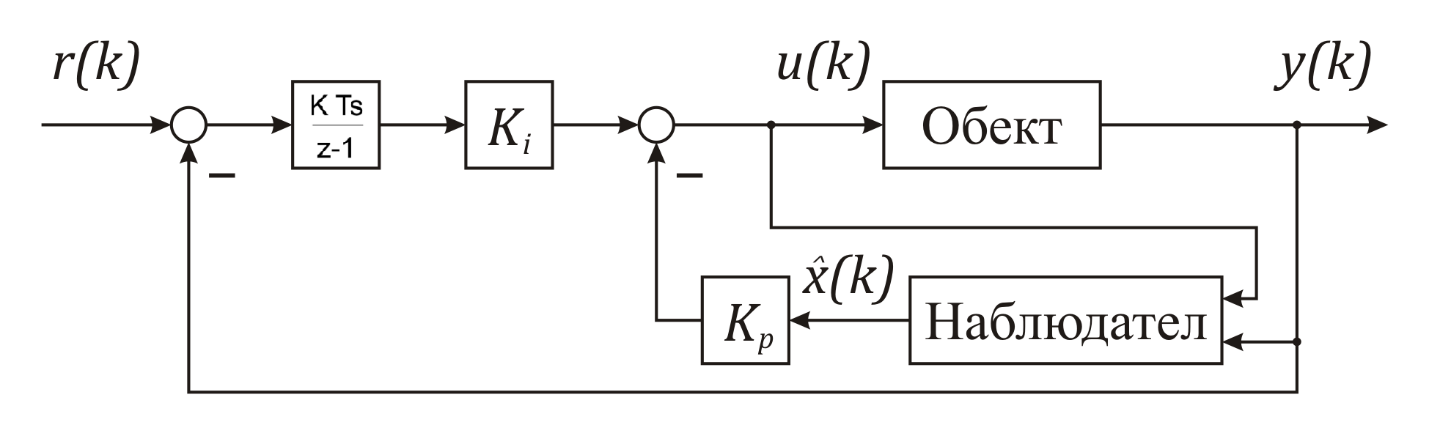
Матриците *R* и *Q* са избрани за R = 1, за Q = CTC = diag([0, 0, 0.01\*0.25]) като за теглото на интегралната съставка е избрано 1000, при което с помощта на функцията *dlqr* за коефициентите на регулатора се получава:

[K1, S, e] = dlqr(A1, B1, Q, R);

K1 = [-26.9109 0.1330 0.1550 0.1744]

### 4.6.2. Проектиране на стохастичен дискретен наблюдател – филтър на Калман

Когато векторът на състоянието или негови компоненти не са достъпни за измерване (в конкретния случай обемът на биореактора и концентрацията на биомасата не могат да бъдат измервани) се проектира управление при непълна информация. На фиг. 4.22 е показана структурна схема на системата с обратна връзка по състояние и наблюдател.



Фиг. 4.22 Система с наблюдател и обратна връзка по състояние

На фигурата по-горе, понеже изходите не могат да бъдат измерени, обратната връзка е синтезирана по оценката *(k)*на състоянието *x*. Тази оценка се изчислява по измеримия изход *y* и управлението *u* с помощта на допълнителна динамична система (наблюдател на състоянието). Този наблюдател наред с избора на обратна връзка е предмет на проектиране от инженера по автоматика.

Заради наличието на шум в изходните сигнали на системата от фиг. 3.1, адитивно добавен към изходите на обекта (блоковете *Random Number, Random Number2, Random Number3*), се налага проектиране на стохастичен наблюдател – филтър на Калман.

Към моделът на обекта (4.35):

x(k+1) = Fx(k) + Gu(k) + Гη(к)

y(k) = Cx(k) + n(k) (4.32)

е добавен адитивно шум, като се предполага, че *η(к)* и *n(k)* са дискретни, взаимно некорелирани бели шумове с дисперсия *Vη* и *Vn* и нулеви средни стойности. При такава постановка на задачата регулатора на нулевото състояние има вида (4.33):

u(k) = -Kp(k), (4.33)

където *(k)* е оценка на векторът на състоянието *x(k),* получена с помощта на дискретен филтър на Калман (4.34):

(k+1) = F(k) + Gu(k) + KН[y(k+1) – CGu(k) – CF(k)] (4.34)

Матрицата KН в наблюдателя се определя така, че грешката *e = x -* , която е случаен процес, да удовлетворява следните условия:

* Средната стойност *me* на *e* асимптотично да клони към нула () = 0, което осигурява „асимптотично неизместена оценка“ *(k)*.
* Дисперсията *De* на *e* да е минимална, което осигурява т. нар. „ефективна“ оценка *(k).*

И така процесите, които описват системата от фиг. 4.22 имат следния вид (4.35):

u(k) = -Kp(k) + kii(k),

xi(k + 1) = i(k) + Т0е(к) = i(k) + Т0(r(k) – C(k)) (4.35)

С цел симулация на системата филтърът на Калман е синтезиран в средата на Matlab. Както линейно -квадратичния регулатор от глава 4.6.1 така и филтъра на Калман се синтезира по линеаризирания модел получен в глава 3.2. С помощта на функция *kalman* е синтезиран следният филтър:

Kal=kalman(sys\_ss,m.NoiseVariance,0.1);

Kal =

A =

x1\_e x2\_e x3\_e

x1\_e 0 0 0.84

x2\_e 1 0 0.7937

x3\_e 0 1 -0.6443

B = u1 y1

x1\_e 0.4977 6.658e-10

x2\_e 0.9701 1.297e-09

x3\_e 0.5903 7.904e-10

C = x1\_e x2\_e x3\_e

y1\_e 0 0 0.5

x1\_e 1 0 -3.313e-10

x2\_e 0 1 -6.456e-10

x3\_e 0 0 1

D = u1 y1

y1\_e 0 3.972e-10

x1\_e 0 6.627e-10

x2\_e 0 1.291e-09

x3\_e 0 7.868e-10

Input groups:

Name Channels

KnownInput 1

Measurement 2

Output groups:

Name Channels

OutputEstimate 1

StateEstimate 2,3,4

Sample time: 0.0056 seconds

Discrete-time state-space model.

Схемата, по която е настроен линейно-квадратичният регулатор и е синтезиран филтър на Калман е показана на фиг. 4.23



Фиг. 4.23 Simulink схема за настройка на регулатора по линеаризирания модел

След като е настроен регулаторът се провежда симулация с нелинейния модел и РФК синтезиран в глава 4.3 по схемата показана на фиг. 4.24



Фиг. 4.24 Simulink схема с нелинейния модел и LQG регулатор

Поради наличието на интегрална съставка е добавен и anti-windup механизъм. Получените резултати са показани съответно за концентрацията на субстрата (фиг. 4.25), дебита на подхранващия разтвор (фиг.4.26), концентрацията на биомаса (фиг.4.27) и обема на биореактора (фиг.4.28)



Фиг.4.25 Концентрация на субстрата Фиг.4.26 Дебит на подхранващия разтвор

Вижда се успешното поддържане на концентрацията на субстрата на зададеното ниво 0.1.



Фиг. 4.27 Концентрация на биомаса Фиг. 4.28 Обем на биореактора

На фигурите (4.29 – 4.31) са показани резултати между оценените и изходните величини.



Фиг. 4.29 Изход и оценка на обема на биореактора



Фиг. 4.30 Изход и оценка на концентрацията на субстрата



Фиг. 4.31 Изход и оценка на концентрацията на биомасата

Резултатите показват работоспособността на РФК. Той дава неизместени оценки на променливите на състоянието.

## 4.7. Симулационни изследвания и сравнения между проектираните регулатори

Тук са направени сравнения между резултатите от различните настройки на ПИД регулатора и линейно-квадратичния регулатор. За по-добро сравнение изходните шумове на обекта са изключени от симулацията.

Получените резултати за дебита на подхранването и концентрацията на субстрата са представени на фиг. 4.32, а на фиг. 4.33 са показани съответно Обем на биореактора и концентрация на биомасата.

От графиките се вижда, че ПИД-а с различните настройки дава почти идентични резултати, докато линейно-квадратичния при тези си настройки като, че ли е леко по-бавен и с по-голямо пререгулиране.

Фиг. 4.32 Дебит на подхранващия разтвор и концентрация на субстрата

Фиг. 4.33 Обем на биореактора и концентрация на биомасата

# 5. Разработка на софтуерно осигуряване на системата за управление

След като управляващите алгоритми са вече известни, както и настройката на параметрите на регулатора предстои представяне на практическата реализация на разработената система за управление на ППФП E. coli MC4110.

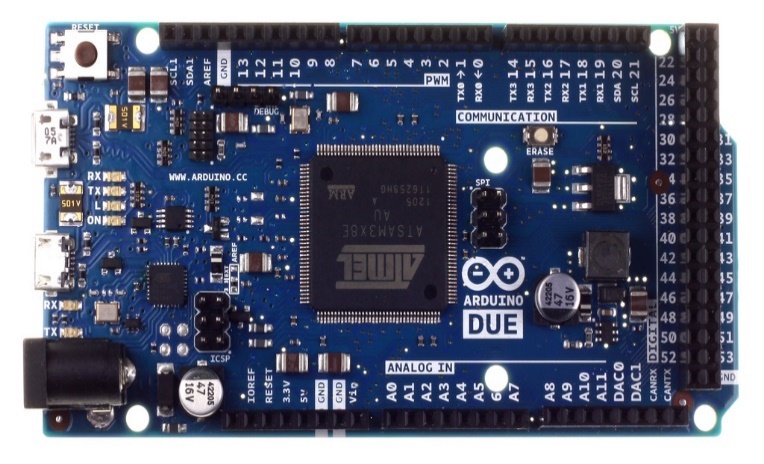
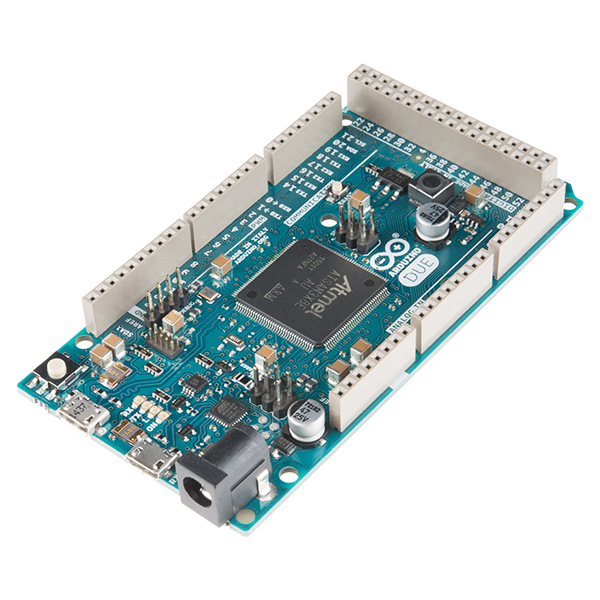
За прилагането на проектираните алгоритми за управление (глава 4) и реализирането на хардуерна симулация в реално време (т. нар. *HIL – Hardware in the Loop*), е използван едноплатковия микроконтролер *Arduino Due*.

*Arduino* е софтуерен и хардуерен проект, който стартира през 2005 година в италианския град Иврея. Днес *Arduino* е компания, която произвежда различни едноплаткови микроконтролерни развойни среди, използващи няколко различни производителя на микропроцесори, сред които и *Atmel Corporation*.

*Atmel Corporation* е американска компания, основана през далечната 1984г. като производител на полупроводникови прибори. В момента тя се е фокусирала върху вградените системи, като продукцията й включва микроконтролери, памети, сензори и други.

## 5.1. Хардуерната платформа Arduino Due

За целите на *HIL* симулацията е използван едноплатковия микроконтролер *Arduino Due* показан на фиг. 5.1.



Фиг. 5.1 Външен вид на Arduino Due

*Arduino Due* е микроконтролерна развойна платка за изграждане на прототипи с процесор от фамилията *Atmel AT91SAM3X8E ARM Cortex-M3*. Това е първият *Arduino* – модул базиран на 32-битов микроконтролер с *ARM* ядро. Има 54 цифрови входно-изходни (*I/O*) порта, 12 аналогови входа, 4 *UART* порта *(hardware serial ports*), 84 *MHz* кварцов резонатор, *USB OTG* съвместим порт, 2бр. DAC (цифрово-аналогов преобразувател), 2 броя *I2C* – интерфейси, захранващ куплунг, *SPI* и *JTAG* конектори, *reset* и *erase* бутони. 12 от цифровите портове могат да се използват като *PWM* (ШИМ).

*Arduino Due* има два микро *USB* конектора - единият е свързан с *USB*-сериен порт конвертор (Atmega16U2), а другият – директно към процесора AT91SAM3X8E. Първият порт е с функции като на останалите *Arduino* микроконтролерни платки, които ползват *USB*-сериен порт конвертор *Atmega8U2* или *Atmega16U2*. Вторият порт микро *USB* може да работи като *USB OTG*, *USB*-сериен порт (*CDC*), *USB-Host* интерфейс и да комуникира с клавиатури, мишки, смартфони, таблети, фотоапарати и др.

За захранване на *Arduino Due* може да се използва *USB* порт на компютър или външен захранващ източник, като превключването между различните захранвания става автоматично. Външният източник може да е батерия или AC-DC адаптер, осигуряващ напрежение в диапазон 7-12V.

Техническите характеристики за по-добра нагледност са представени в таблица 5.1

Таблица 5.1

|  |  |
| --- | --- |
| Microcontroller | AT91SAM3X8E |
| Operating Voltage | 3.3V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-16V |
| Digital I/O Pins | 54 (of which 12 provide PWM output) |
| Analog Input Pins | 12 |
| Analog Output Pins | 2 (DAC) |
| Total DC Output Current on all I/O lines | 138mA |
| DC Current for 3.3V Pin | 800 mA |
| DC Current for 5V Pin | 800mA |
| Flash Memory | 512 KB all available for the user applications |
| SRAM | 96 KB (two banks: 64KB and 32KB) |
| Clock Speed | 84 MHz |
| Length | 101.52 mm |
| Width | 53.3 mm |
| Weight | 36 g |

### 5.1.1. Комуникация

*Arduino Due* има редица възможности за комуникация с компютър, друго *Arduino* или други микроконтролери и различни устройства като телефони, таблети, камери и други. *SAM3X* предоставя един хардуерен UART и три хардуерни *USARTs* за серийна комуникация.

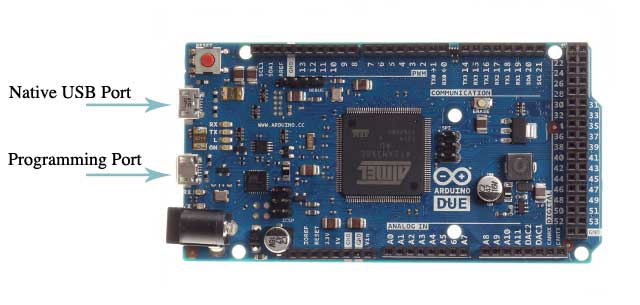
Портът за програмиране е свързан с *ATmega16U2*, който осигурява виртуален *COM* порт на софтуера на свързания към *Arduino*-то компютър. *16U2* също е свързан към *UART* хардуера *SAM3X*. Светодиодите *RX* и *TX* на платката премигват, когато данните се предават през ATmega16U2 чипа и USB връзката към компютъра.

*Native USB* портът е свързан към *SAM3X*. Той позволява серийна комуникация през USB. Това осигурява серийна връзка към серийния монитор или други приложения на компютъра. С помощта на този порт платката може да емулира *USB* мишка или клавиатура.

### 5.1.2. Програмиране

Качването (upload) на скици (sketches) в *SAM3X* е различно от *AVR* микроконтролерите, намиращи се в други *Arduino* платки, понеже флаш паметта трябва да бъде изтрита, преди да бъде препрограмирана. Качването в чипа се управлява от *ROM* на *SAM3X*.

Всеки от *USB* портовете (фиг. 5.2) може да се използва за програмиране на платката въпреки, че се препоръчва да се използва *Programming Port*-а.

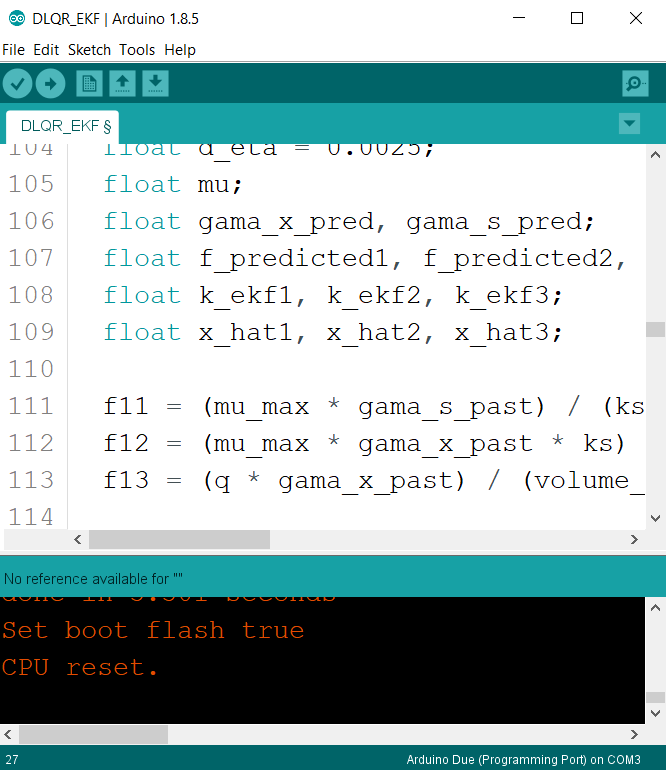


Фиг. 5.2 Портове за комуникация с компютър

Наличието на буутлоудър (*bootloader*) дава възможност да се зарежда код в микроконтролера без нужда от междинен допълнителен хардуерен програматор.

## 5.2. Интегрирана среда за разработка (ИДЕ) – Arduino IDE

*Arduino* e платформа с отворен код, с голяма обществена подкрепа, обширен набор от библиотеки и примерни проекти. За създаване на приложения и програмиране на платките *Arduino* се използва свободен софтуер *Arduino IDE*, който е изключително удобен и лесен за употреба. Той може да бъде изтеглен безплатно от официалния сайт [10]. Инсталирането е тривиално и след като се отвори средата за разработка има следния общ вид (фиг. 5.3).



Фиг. 5.3 Общ изглед на *Arduino IDE*

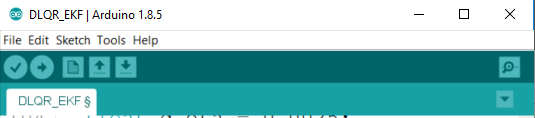
Преди да може да бъде качен какъвто и да било код на платката трябва да се да се избере модела на *Arduino* платката, за която ще се програмира от менюто *Tools > Board > Arduino Due (Programming Port)*. След като свържем платката със компютъра операционната система назначава номер на виртуален порт, който можем да видим от диспечера на устройствата в случай че сме на *Windows* (*Device Manager*). Вече разбрали номера на COM порта трябва да го зададен в ИДЕ-то от менюто *Tools > Ports*.

Това е специална среда за програмиране, която позволява да се пишат програми за *Arduino* на лесен език, базиран на езика *Processing*. Когато се натисне бутона, който ъплоудва програмата на платката, кода се превежда на С и се предава на avr-gcc компилатора, софтуер с отворен код, който прави превода на разбираем за микроконтролера език.

Програмите написани на *Arduino IDE* се наричат скици (*sketches*). Файловете имат разширение .ino, но са си обикновени текстови файлове и могат да бъдат отворени с всеки текстов редактор.

Средата за разработка се състои от текстов редактор, където се пишат програмите (бялото поле в по-голямата част от фиг. 5.3), поле за съобщения (системни, за грешка и т.н. – черното поле под текстовия редактор), лента с инструменти (фиг. 5.4) с бутони за най-общите функции като отляво надясно бутоните са както следва:

* бутон за проверка на програмата за синтактични грешки (*verify*);
* бутон за качване на програмата на платката (*upload*), който бутон всъщност преди да се качи проверява и за синтактични грешки;
* бутон за отваряне на нова празна скица (*New*);
* бутон за отваряне на вече съществуваща скица (*Open*);
* бутон за запазване на скицата (*Save*)
* бутон за отваряне на сериен монитор, на който може да се пращат различни стойности на променливи (*Serial Monitor*)



Фиг. 5.4. Лента с инструменти и лента с менюта

## 5.3. Разработка на системата за управление на биотехнологичен процес (БТП)

Следващата стъпка за реализиране на хардуерната симулация (*HIL*) е проектираните регулатори и филтри на Калман от глава 4 да бъдат вградени в микроконтролера. За целта са разработени функции и алгоритми в *Arduino IDE* средата. Допълнително с цел процесите да се наблюдават в реално време е изградена комуникация между компютъра (*Simulink*) и микроконтролера.

### 5.3.1. Основна структура на Arduino програмите

Основната структура на езика за програмиране на *Arduino* е относително проста и се състои от поне две части. Тези две задължителни части (или функции) обгръщат блокове от код.

void setup () { код; }

void loop () { код; }

И двете функции са задължителни за да работи програмата. Функцията *setup* () се извиква веднага щом програмата тръгне. Изпълнява се само веднъж и се използва при задаване на режима на пиновете (pinMode – дали да са входове или изходи) или да отвори серийната комуникация. След това се изпълнява функцията loop(). Тя е ядрото на всяка *Arduino* програма и прави точно това, което името и подсказва, че ще прави – цикли постоянно.

### 5.3.2. Серийна комуникация между микроконтролера и компютъра

За целта се използва вградения в езика т. нар. „Сериен обект” (Serial). Този обект съдържа всичкия код необходим за изпращане и приемане на данни както и някои полезни методи (функции). Един такъв полезен метод е *Serial.begin(speed),* който се поставя във функцията *setup*(), за да отвори серийния порт за изпращане на данни обратно към компютъра със с максималната скорост, зададена като параметър в битове за секунда(*baud rate*). Обикновено тази скорост е 9600bps както е и в настоящата разработка.

Понеже във всеки един момент от времето наведнъж могат да се пратят максимум 8 бита = 1 байт, а разработените алгоритми изискват да се обменят данни (числа) с плаваща запетая с единична точност (float, single), които са по 4 байта, е разработен специален алгоритъм за комуникация, чийто код е показан в таблица 5.1.

Таблица 5.1

|  |
| --- |
| float res;  union SerialFloat {  char arr[4];  float number;  };  union SerialFloat data;  void setup() {  // open a serial connection  Serial.begin(9600);  }  void loop() {  //on every byte read the byte and put it in arr  for (int i = 0; i <= 3; i++) {  if (Serial.available() > 0) {  data.arr[i] = Serial.read();  }  }  // take data from union as float  res = data.number;    if (isnan(res)) {  res = 0.0;  }    Serial.write((unsigned char\*)(&res), 4);  } |

Тук във структурата заделяме 4 байта, които използваме веднъж като *float* число, веднъж като масив от 4 еднобайтови елемента. След това декларираме променлива от тип тази структура, която променлива по-надолу използваме, за да достъпим тази заделена памет чрез полето й от тип масив, който пълним със данните, получени по серийната комуникация. Винаги проверяваме дали има данни за четене чрез *Serial.available()* и ако има ги записваме байт по байт в масив,а после това парче памет го използваме като *float* число, което сме проверили дали е прочетено правилно. И едва сега можем да използваме получените данни за различни изчисления. И накрая, за да изпратим отново данните към компютъра трябва да го „разглобим“ на байтове чрез промяна на към тип данни указател към *char* и да го изпратим байт по байт. Както подсказват имената на методи на обекта *Serial, Serial.read()* получава (чете) данни, а Serial.write() пише (изпраща) данни.

### 5.3.3. Разработени Функции и Алгоритми

**extended\_kalman\_filter**

В тази функция е реализиран разширеният филтър на Калман. Като входни променливи постъпват управлението и управляемия изход на обекта (концентрация на субстрата), която се получава по серийната комуникация.

След това се изчисляват Якобияните на матрицата *F* като променливите gama\_x\_*past*, gama\_s\_*past*, *volume*\_past са декларирани като глобални и на всяко извикване използват стойностите пресметнати по-надолу в алгоритъма.

//Jacobians

f11 = (mu\_max \* gama\_s\_past) / (ks + gama\_s\_past) - q / volume\_past;

f12 = (mu\_max \* gama\_x\_past \* ks) / ((ks + gama\_s\_past) \* (ks + gama\_s\_past));

f13 = (q \* gama\_x\_past) / (volume\_past \* volume\_past);

f21 = -(mu\_max \* gama\_s\_past) / (y\_sx \* (ks + gama\_s\_past));

f22 = -(mu\_max \* gama\_x\_past \* ks) / (y\_sx \* (ks + gama\_s\_past) \* (ks + gama\_s\_past)) - q / volume\_past;

f23 = -(q \* (gama\_in - gama\_s\_past)) / (volume\_past \* volume\_past);

f31 = 0; f32 = 0; f33 = 0;

Всеки елемент от матрицата *F* се умножава по такта на дискретизация и се събира с единичната матрица. За целта е използван двумерен масив *F*, чийто елементи се попълват по следния начин

F[0][0] = 1 + ts \* f11;

F[0][1] = ts \* f12;

F[0][2] = ts \* f13;

F[1][0] = ts \* f21;

F[1][1] = 1 + ts \* f22;

F[1][2] = ts \* f23;

F[2][0] = ts \* f31;

F[2][1] = ts \* f32;

F[2][2] = 1 + ts \* f33;

Масива *Ft* представлява транспонираната матрица *F*

//F'

Ft[0][0] = 1 + ts \* f11;

Ft[0][1] = ts \* f21;

Ft[0][2] = ts \* f31;

Ft[1][0] = ts \* f12;

Ft[1][1] = 1 + ts \* f22;

Ft[1][2] = ts \* f32;

Ft[2][0] = ts \* f13;

Ft[2][1] = ts \* f23;

Ft[2][2] = 1 + ts \* f33;

За умножаване на матриците *F*, *P* и *Ft* е разработена функция *matrix\_multiply*

for (int row = 0; row < 3; row++) {

for (int col = 0; col < 3; col++) {

result\_matrix[col][row] = 0;

for (int i = 0; i < 3; i++) {

result\_matrix[col][row] += left[i][row] \* right[col][i];

}

}

}

Интересното тука е, че тази функция използва глобален масив за съхранение на резултата от умножението на матриците, като преди всеки един запис на резултат от умножението на текущия елемент той се занулява.

Първата стъпка от алгоритъма се реализира като се вика 2 пъти функцията *matrix\_multiply* за умножаване на матриците *F*, *P* и *Ft* като втория път използваме резултата от умножението на *F* и *P* записан в глобалния масив, след което резултатът от това умножение е записан във ковариационната матрица:

matrix\_multiply(F, P);

matrix\_multiply(result\_matrix, Ft);

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

p\_predicted[i][j] = result\_matrix[i][j];

}

}

към диагоналните елементи се добавя диверсионната матрица на шумовете

p\_predicted[0][0] = p\_predicted[0][0] + d\_ceta;

p\_predicted[1][1] = p\_predicted[1][1] + d\_ceta;

p\_predicted[2][2] = p\_predicted[2][2] + d\_ceta;

Следва изчисляване на предсказаната оценка на векторът на състоянието и ковариационна матрица

//-predicted state

mu = (mu\_max \* gama\_s\_past) / (ks + gama\_s\_past);

gama\_x\_pred = mu \* gama\_x\_past - (q / volume\_past) \* gama\_x\_past;

gama\_s\_pred = -(1 / y\_sx) \* mu \* gama\_x\_past + (q / volume\_past) \* (gama\_in - gama\_s\_past);

f\_predicted1 = gama\_x\_past + ts \* gama\_x\_pred;

f\_predicted2 = gama\_s\_past + ts \* gama\_s\_pred;

f\_predicted3 = volume\_past + ts \* q;

Преминава се към стъпка две изчисляване на коефициента на филтъра на Калман

//2. Compute Kalman Gain

k\_ekf1 = p\_predicted[0][1] \* 1 / (p\_predicted[1][1] + d\_eta);

k\_ekf2 = p\_predicted[1][1] \* 1 / (p\_predicted[1][1] + d\_eta);

k\_ekf3 = p\_predicted[2][1] \* 1 / (p\_predicted[1][1] + d\_eta);

Следва стъпка три изчисляване на оценката на вектора на състоянието на базата на управлението и измеримия изход от системата и на базата коефициента на РКФ получен в стъпка 2

//3.Compute Estimate

x\_hat1 = f\_predicted1 + k\_ekf1 \* (gama\_s\_measured - f\_predicted2);

x\_hat2 = f\_predicted2 + k\_ekf2 \* (gama\_s\_measured - f\_predicted2);

x\_hat3 = f\_predicted3 + k\_ekf3 \* (gama\_s\_measured - f\_predicted2);

В стъпка 4 се изчислява ковариационната матрица *Pk*

//4.Compute error covariance

temp\_k\_ekf[0][0] = 1;

temp\_k\_ekf[0][1] = -k\_ekf1;

temp\_k\_ekf[1][1] = 1 - k\_ekf2;

temp\_k\_ekf[2][1] = -k\_ekf3;

temp\_k\_ekf[2][2] = 1;

matrix\_multiply(temp\_k\_ekf, p\_predicted);

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

P[i][j] = result\_matrix[i][j];

}

}

Като изходни променливи на филтъра са оценките, изчислени в стъпка 3, които се присвояват на глобалните променливи gama\_x\_*past*, gama\_s\_*past*, *volume*\_past, които ще се ползват като начални стойности на следващата итерация. Тези оценки се присвояват и на изходните променливи на филтъра

//output

gama\_x\_past = x\_hat1;

gama\_s\_past = x\_hat2;

volume\_past = x\_hat3;

**feed\_forward\_component**

В отделна функция е реализирана частта от управляващия сигнал, която поддържа процеса на ферментация в текущата работна точка. Като вход този блок получава оценките на изходните променливи изчислени в *extended\_kalman\_filter*. Първо се пресмята *MU* (специфична скорост на растеж на микроорганизмите) след което и самата съставка на управлението

float result = 0, mu;

mu = (mu\_max \* gama\_s) / (ks + gama\_s);

result = (volume \* mu \* gama\_x) / (y\_sx \* (gama\_in - gama\_s));

return result;

На изхода на блока се формира управлението *Uff*.

**pid\_component**

В тази функция се формира цифровия ПИД алгоритъм получен в точка 4.1. Като входни променливи функционалният блок получава заданието (SP) и получената оценка на управляемата променлива получена на изхода на РФК. В блока са заложени коефициентите на регулатора получени чрез оптимизационните процедури получени в глава 4. Следва формиране на грешката и поетапно пресмятане на отделните съставки на регулатора като накрая тези съставки се събират. Подобно на функцията реализираща филтъра на Калман и тук старите стойности от предишния момент се пазят в глобални променливи. Примерната реализация се дава със следния сорс код

//pid term

float ufb = 0;

float error = set\_point - pv;

//p term

up = kp \* (b \* set\_point - pv);

//I term

if (v >= 0) {

ui = ui\_old + bi1 \* error;

} else {

ui = 0.0;

}

//d term

ud = ad \* ud\_old + bd \* (c \* (set\_point - set\_point) - (pv - pv\_old));

ui\_old = ui; ud\_old = ud; pv\_old = pv; ufb = up + ui + ud;

return ufb;

**saturation**

Тук се реализира ограничението на управляващия сигнал, получен чрез сумиране на блоковете *feed\_forward\_component* и *pid\_component* във главната функция loop(). Естествено вход е управляващия сигнал, а изход е ограничения такъв. Реализацията е представена по следния начин:

float result = 0;

if (control < 0) {

result = 0;

} else if (control > 1) {

result = 1;

} else {

result = control;

}

return result;

Понеже върнатата стойност от функцията се подава към серийната комуникация, тя трябва да се изпрати байт по байт. Това се реализира чрез така нареченото кастване *Serial.write((unsigned char\*)(&q), 4);*.

**linear\_quadratic\_gaussian**

Този тип регулатор използва обратна връзка по състояние по квадратичен критерий и използва вместо състоянията, техните оценки получени чрез линеен филтър на Калман, синтезиран в глава 4.6.2. Настоящия регулатор е реализиран в главната функция *loop()* на програмата.

От уравнение 4.32 и резултатът от изпълнението на функцията *kalman* следва, че за оценките може да се запише

= + u(k)

Това ни позволява да изчислима оценките на състоянията:

//linear kalman filter equations

gama\_x\_est = volume\_est\_past \* 0.84 + ufb\_past \* 0.4977;

gama\_s\_est = x\_est\_past + 0.7937 \* volume\_est\_past + ufb\_past \* 0.9701;

volume\_est = s\_est\_past - 0.6443 \* volume\_est\_past + ufb\_past \* 0.5903;

Следващата стъпка е да изчислим интегралната съставка като към нея е добавен и механизъм против интегрално насищане *xi(k + 1) = xi(k) + Т0е(к)*

float error = set\_point - gama\_s\_past;

// -26.9109 0.1330 0.1550 0.1744

xi = xi\_past + ts \* error + ts \* anti\_windup\_past;

Съставката против интегрално насищане е реализирана като от наситения сигнал на управлението е изваден ненаситения и за да увеличим действието й е умножен по коефициент и след това този резултат е добавен към интегратора.

float anti\_windup = 10 \* (q - v);

И накрая ни остава да сметнем и самото управление по формула (4.34)

u(k) = = ,

където коефициентите на матрица на регулатора са получени в глава 4.6

float ufb = 26.9109 \* xi - (0.1330 \* gama\_x\_est + 0.1550 \* gama\_s\_est + 0.1744 \* volume\_est);

И тук, подобно на функциите реализиращи филтъра на Калман и ПИД регулатора, старите стойности от предишния момент се пазят в глобални променливи.

ufb\_past = ufb;

xi\_past = xi;

x\_est\_past = gama\_x\_est;

s\_est\_past = gama\_s\_est;

volume\_est\_past = volume\_est;

anti\_windup\_past = anti\_windup;

q\_past = q;

# 6. Експериментални резултати

Работоспособността на разработената вградена система за HIL симулация на модел на ППФП е потвърдена с експериментални изследвания. В тази глава се представят сравнения между симулационните, получени в средата на *Matlab*, резултати и експериментално снетите резултати за двата вида на настройка на ПИД регулатора и на линейно-квадратичния регулатор както и сравнение между експериментално снетите характеристики между ПИД регулатора с двата вида настройки и линейно-квадратичния регулатор.

## 6.1. Сравнение между симулационните и реалните резултати при настройка на ПИД регулатора с генетичен алгоритъм

Първоначално обект на разглеждане ще бъде системата с ПИД регулатор настроен по генетичен алгоритъм.

От фиг. 6.1 до фиг. 6.4 са показани сравненията на данните от симулацията в *Matlab* и системата за управление в реално време на ферментационен процес.

Фиг. 6.1 Концентрация на субстрата Фиг. 6.2 Дебит на подхранващия разтвор

Вижда се успешното поддържане на концентрацията на субстрата на зададеното ниво 0.1.

Фиг. 6.3 Концентрация на биомаса Фиг. 6.4 Обем на биореактора

От фиг. 6.2 се вижда, че регулаторът бързо установява и успешно поддържа γs на желаното ниво 0.1g.l-1. В резултат на това в края на ферментацията се постига концентрация на биомасата от приблизително 23g.l-1 (фиг.6.3)

## 6.2. Сравнение между симулационните и реалните резултати при настройка на ПИД регулатора с конвенционален метод на оптимизация

Тук обект на разглеждане ще бъде системата с ПИД регулатор настроен по линеаризирания модел от глава 4.5.

От фиг. 6.5 до фиг. 6.8 са показани сравненията на данните от симулацията в *Matlab* и системата за управление в реално време на ферментационен процес.

Фиг. 6.5 Концентрация на субстрата Фиг. 6.6 Дебит на подхранващия разтвор

Вижда се успешното поддържане на концентрацията на субстрата на зададеното ниво 0.1.

Фиг. 6.7 Концентрация на биомаса Фиг. 6.8 Обем на биореактора

От фиг. 6.6 се вижда, че регулаторът бързо установява и успешно поддържа γs на желаното ниво 0.1g.l-1. В резултат на това в края на ферментацията се постига концентрация на биомасата от приблизително 23g.l-1 (фиг.6.7)

От фигурите се вижда доброто съвпадение между резултатите от симулациите в SIMULINK и тези от експеримента със системата за хардуерна симулация, което потвърждава правилно избрания ARMAX модел от 3-ти ред благодарение, на който успешно е оптимално настроен ПИД регулатора.

## 6.3. Сравнение между симулационните и реалните резултати при линейно-квадратичния регулатор

Тук обект на разглеждане ще бъде системата с линейно-квадратичен регулатор проектиран в глава 4.6

От фиг. 6.9 до фиг. 6.12 са показани сравненията на данните от симулацията в *Matlab* и системата за управление в реално време на ферментационен процес.

Фиг. 6.9 Концентрация на субстрата Фиг. 6.10 Дебит на подхранващия разтвор

Вижда се успешното поддържане на концентрацията на субстрата на зададеното ниво 0.1.

Фиг. 6.11 Концентрация на биомаса Фиг. 6.12 Обем на биореактора

От фиг. 6.10 се вижда, че регулаторът бързо установява и успешно поддържа γs на желаното ниво 0.1g.l-1. В резултат на това в края на ферментацията се постига концентрация на биомасата от приблизително 23g.l-1 (фиг.6.11)

## 6.4. Сравнение между реалните резултати при настройка на регулатора с конвенционален метод на оптимизация и при настройка с ГА както и на линейно-квадратичния регулатор

Тук са направени сравнения между резултатите от различните настройки на ПИД регулатора и линейно-квадратичния регулатор за системата за HIL симулация.

От фиг. 6.13 до фиг. 6.16 са показани сравненията на данните от HIL симулацията при различните регулатори.

2.4

Фиг. 6.13 Концентрация на субстрата Фиг. 6.14 Дебит на подхранващия разтвор

Фиг. 6.15 Концентрация на биомаса Фиг. 6.16 Обем на биореактора

# 7. Заключение

В дипломната работа е представена разработената система за оптимално ПИД управление в реално време на нелинеен симулационен модел на полупериодичен ферментационен процес. Системата се състои от два стандартни PC, програмируем логически контролер на Schneider Electric M238, FPGA Board и авторски софтуер, осигуряващ потребителски интерфейс и работа на системата в реално време. За попълване на липсващата информация за променливите на състоянието е синтезиран и е реализиран разширен филтър на Калман. Получените оценки на състоянието се използват за формиране на управляващия сигнал. Дадени са резултати от работата на системата за управление и от симулации в средата на MATLAB/SIMULINK. Близостта на резултатите потвърждава работоспособността на системата. Доброто съвпадение на експерименталните резултати показват успешно настроените ПИД регулатори по двата метода както и успешно избрания линеаризиран модел от идентификацията в затворен контур. Регулаторът успешно установява и под-държа реалната концентрация на субстрата на желаното ниво. В резултат на това са постигнати оптимални условия за растеж на микроорганизмите задоволително увеличение на концентрацията на биомасата в реактора.

# 8. Използвана литература:

1. Славов Ц., Роева О., Система за оптимално ПИД управление в реално време на симулационен модел на ППФП

2. Гарипов Е., Цифрови системи за управление: част 1. Проектиране на ПИД регулатори. Второ допълнително и преработено издание. ТУ, София, 2007.

3. Гарипов Е., Цифрови системи за управление: част 1. Проектиране на ПИД регулатори. Второ допълнено и преработено издание. ТУ, София, 2007.

4. Гарипов Е., Славов Ц., Ръководство за лабораторни упражнения по идентификация на системи. Второ допълнително и преработено издание. ТУ, София, 2009.

5. T. Slavov, O. Roeva. INT. J. BIOAUTOMATION, 2011, Genetic Algorithm Tuning of PID Controller in Smith Predictor for Glucose Concentration Control

6. Цонков Ст., Основи на автоматичното управление на БТП. София, 1994г.

7. Schneider Electric, SoMachine Programming Guide (10/2010)

8. Schneider Electric, M238 Logic Controller Hardware Guide (10/2010)

9. Хараланова Е., Пулева Т., Теория на управлението – II част. ТУ – София 2016

10. https://www.arduino.cc/en/Main/Software/