FELADATKIÍRÁS

A feladatkiírást a **tanszék saját előírása szerint** vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Konzulens

BUDAPEST, 2021

Tartalomjegyzék

[Összefoglaló 5](#_Toc87131174)

[Abstract 6](#_Toc87131175)

[1 Bevezetés 7](#_Toc87131176)

[2 Irodalmi háttér, módszerek, felhasznált adatok 8](#_Toc87131177)

[2.1 Élettani Háttér 8](#_Toc87131178)

[2.2 STAR protokoll 8](#_Toc87131179)

[2.3 In-silico validáció 9](#_Toc87131180)

[2.4 Julia nyelv 10](#_Toc87131181)

[3 Eredmények 11](#_Toc87131182)

[3.1 Tervezés és implementáció kihívásai 11](#_Toc87131183)

[3.1.1 Adatok beolvasása 11](#_Toc87131184)

[3.1.2 Protokollok beépítése 11](#_Toc87131185)

[3.1.3 JavaCall használata 12](#_Toc87131186)

[3.1.4 Differenciálegyenletek megoldása 13](#_Toc87131187)

[3.1.5 Statisztikai kiértékelés 15](#_Toc87131188)

[3.1.6 Mintavételezés 16](#_Toc87131189)

[3.1.7 Adatok ábrázolása 17](#_Toc87131190)

[3.2 Elkészült rendszer bemutatása 17](#_Toc87131191)

[3.2.1 Felsőszintű architektúra 17](#_Toc87131192)

[3.2.2 Részletes megvalósítás 18](#_Toc87131193)

[3.2.3 Elkészült rendszer értékelése mérnöki szempontból 18](#_Toc87131194)

[3.2.4 Elkészült rendszer értékelése orvosi szempontból 18](#_Toc87131195)

[3.3 Rendszer validálása 18](#_Toc87131196)

[4 Összefoglalás 19](#_Toc87131197)

[4.1 További fejlesztési lehetőségek 19](#_Toc87131198)

[Irodalomjegyzék 20](#_Toc87131199)

[Függelék 21](#_Toc87131200)

Hallgatói nyilatkozat

Alulírott **Rezeda Kázmér**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet (nem kívánt törlendő) meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2021. 11. 07.

...…………………………………………….

Összefoglaló

Ide jön a ½-1 oldalas magyar nyelvű összefoglaló, melynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

Abstract

Ide jön a ½-1 oldalas angol nyelvű összefoglaló, amelynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

# Bevezetés

Intenzív osztályon komoly kihívást jelent a páciens vércukorszintjének normoglikémiás tartományban tartása, amellyel az intenzív osztályon ápolt betegek mortalitása akár 25-40%-kal csökkenthető. A betegek tápanyag bevitelét, illetve inzulin adagolását mesterséges úton kell biztosítani. Erre úgynevezett szoros vércukor szabályozást (Tight Glycemic Control, TGC) alkalmaznak. A STAR (STochastic TARgeted) protokoll egy ilyen szoros vércukor szabályozó protokoll, melyet a világ több pontján használnak.

A STAR protokollnak létezik egy szimulációs környezete, melyet matlab környezetben implementáltak. Az önálló labor célja, hogy a szimulációs környezet Julia nyelven implementált változata létrejöjjön, amely jobban támogatja a szimulációs eredmények statisztikai összehasonlítását.

A feladat során több részfeladatot kellett megoldani. Először be kellett olvasni Java bináris fájlokból a betegek klinikai adatait Julia adatstruktúrákba. Ezután Julia kódból kellett futtatni a STAR protokollt megvalósító Java kódot, vagyis Java függvényeket kellett hívni Julia környezetből. Egy további kihívást jelentett az úgynevezett ICING modell differenciálegyenleteinek az implementálása és megoldása, amelyek leírják az emberi szervezet metabolikus működését, illetve folyamatait. Végül elkészült egy olyan modul is, amely a szimulálás során mért vércukor adatokból különböző statisztikákat készít.

A munka eredménye egy úgynevezett in-silico validációs környezet megalkotása lett, amelynek lényege, hogy valós betegek adatai alapján, de biztonságos virtuális környezetben szimulált kezelés adatairól, a kezelés eredményességéről szolgáltat információt.

A továbbiakban részletesen szó lesz az élettani háttérről, a STAR protokollról, a megvalósításhoz használt programozási nyelv előnyeiről, a tervezés, illetve implementáció kihívásairól, az elért eredményekről és a továbblépési lehetőségekről.

# Irodalmi háttér, módszerek, felhasznált adatok

## Élettani Háttér

A szoros vércukor szabályozás megvalósítása nehéz feladat, hiszen az az emberi szervezetet, egy nagyon összetett fiziológiai rendszert kell modellezni, ráadásul a betegek egymástól lényegesen különböznek és gyorsan változhat az állapotuk is. Nagyon sok speciális paramétere van a rendszernek, amelyek közül csak néhányat lehet közvetlenül megmérni.

A normoglikémiás vagyis optimális vércukorszint tartomány 4,4 – 6 (8) mmol/l között van. Ha ez a tartomány alá esik a vércukorszint, akkor hipoglikémiáról, ha e fölé, akkor hiperglikémiáról beszélhetünk. Különböző mértékben és időtávon, de mindkettő negatív következményekkel jár [1].

## STAR protokoll

A STAR (Stochastic Targeted Control) protokoll egy szoros vércukor szabályozó protokoll. „Nurse-in-the-loop” típusú rendszer, ami azt jelenti, hogy nem avatkozik be közvetlenül, csak terápiás javaslatot ad, amit a nővér vagy orvos felülbírálhat. A nővér megmér bizonyos beteg paramétereket (jelen esetben a páciens vércukorszintjét), a döntéstámogató rendszer ezekből meghatározza a közvetlenül nem mérhető paramétereket, majd előállít terápiás javaslatokat. A nővér a kiválasztott terápiás javaslat alapján végül beállítja a javasolt értékeket az infúziós pumpákon [2] [3] [4].

Nagy előnye a STAR protokollnak, hogy személyre szabott terápiát lehet megvalósítani. A protokollt a nővérek és orvosok egy Android platformra, Java nyelven implementált, felhasználóbarát kezelői felülettel rendelkező környezetben, egy tablet-en használják mindennapi munkájuk során.

A protokoll az ICING2 matematikai modellt használja [7], amely egy klinikailag validált metabolikus modell (ábra 1).

A modell számos egyénspecifikus paramétert tartalmaz, amelyek közül az inzulin szenzitivitás paraméter (SI) maradt az egyetlen változó, ezzel reprezentálva a páciens állapotát. Az inzulin érzékenység (inzulin szenzitivitás) (SI) a szervezet inzulin függő glükóz felvételét jellemző mérőszám. Ha ez az érték nő, akkor a beteg érzékenyebb lesz az inzulinra, tehát már kevesebb inzulin hatására is jobban esik a vércukorszintje.

A protokoll a vércukorszint predikcióját az erre vonatkozó változást leíró differenciálegyenletek megoldásával határozza meg, amelyek az alábbi ábrán látszanak:

Diagram

Description automatically generated

Vércukorszintet befolyásoló fiziológiai modell

## In-silico validáció

Az in-silico olyan technikára vonatkozik, amikor a kísérlet számítógépes szimuláció során folyik.

Jelen esetben két fázisból áll:

* Az első fázisban a nyers klinikai adatokat felhasználva létrehozzuk a virtuális beteget. Majd, a mért vércukorszintek, inzulinadagolás, illetve táplálás alapján meghatározzuk az időbeli inzulinszenzitivitását.
* A második fázisban futtatjuk a szimulációt a virtuális betegeken, jegyezve a mért vércukorszinteket. A végén összehasonlítjuk az eredeti és a szimulált eredményeket, validálunk.

Ezáltal lehetséges különböző kezelési protokollok vagy különböző betegcsoportok kezelési adatainak összehasonlítása.

## Julia nyelv

Az intenzív osztályokon ápolt betegek szoros vércukor szabályozó szoftverét szimuláló környezet megvalósításához a Julia programozási nyelvet [5] használtam. Ennek a nyelvnek számos olyan tulajdonsága van, amely ideálissá tette ennek a feladatnak a megoldására:

* Gyors, úgynevezett JIT (Just in time) fordítója van, amely a kód hatékony futásidejű fordítását teszi lehetővé.
* Széleskörű, gazdag könyvtárak érhetőek el benne, számos tudományterületről.
* Alapvetően kicsi, csak a szükséges könyvtárakat tartalmazza. Célzottan lehet importolni az adott feladathoz szükséges könyvtárakat, így magas a hatékonysága.
* Nyílt forráskódú.
* Főként a tudományos számítás (Scientific Computation) a fókusza, ami miatt az ilyen irányú tudományos munka felkapta és úgy néz ki, hogy ebbe az irányba mozdul el. Tipikusan ilyen például a Machine Learning, amivel elkezdünk foglalkozni az SI predikció kapcsán.

# Eredmények

A következőkben az elért eredmények kerülnek bemutatásra. Többek között ismertetve lesznek a tervezés és implementáció kihívásai, az elkészült rendszer felépítése, értékelése mérnöki, illetve orvosi szempontból, valamint a validálása.

## Tervezés és implementáció kihívásai

A fejlesztés során számos mérföldkő volt, számos technológiát kellett felhasználni. A továbbiakban ezek kerülnek bemutatásra.

### Adatok beolvasása

A szimuláció végrehajtásához először be kellett olvasni a páciens adatokat. Ezek lehetnek Java bináris vagy Matlab kiterjesztésű fájlokban tárolva. Matlab esetén rendelkezésre állt egy beolvasást segítő könyvtár, viszont Java fájlok esetébent először létre kellett hozni Juliában a Java objektumot a szerializált adatból, majd abból átemelni az adatot a Julia által is könnyen használható adatstruktúrákba.

### Protokollok beépítése

Mivel a cél egy in-silico szimulációs környezet megalkotása volt, ezért a validálni kívánt STAR protokoll mellé, még két másik is beépült a rendszerbe. Az egyik egy egyszerű (SIMPLE) terápiás eszköz, amely csak az aktuális vércukorszint alapján javasol kezelési módot. A másik, a historikus (HISTORIC) kezelési adatokat felhasználva számítja a ki a vércukorszinteket, ez azt jeleni, hogy a virtuális méréseket a valós mérési időpontokban végzi el (a STAR ugyanis pontosan 1-2 esetleg 3 óránként), valamint az inzulin és tápanyag adagolás is olyan időben és mennyiségben történik, ahogy az a valós kezelés során is. Az utóbbi kontroller rendkívül hasznos az ICING matematikai modell tesztelésére is, hiszen ezzel jól látszik, hogy a virtuális kezelés vércukor értékei mennyire közelítik meg a valós terápiáét, vagyis, ha változtatunk az ICING modellen, akkor rögtön látszik az okozott eltérés is.

A szimuláció indításakor lehet választani a felsorolt háromfajta közül, illetve más beállítási módokra is van lehetőség. A STAR és SIMPLE kontrollerek esetén ki lehet választani, hogy milyen fajta mérési időzítés legyen alkalmazva, a historikus, vagy 1-2, vagy 3 órás. Továbbá a SIMPLE kontrollert be lehet állítani úgy, hogy alacsony, közepes vagy magas szén-hidrát bevitelt alkalmazzon.

A felsorolt protokollokkal és azok beállításainak különböző variálásával szélesebb kép adható egy kezelés minőségéről és a páciens egészségügyi állapotáról.

### JavaCall használata

A legnagyobb nehézséget talán a Júliából történő Java kód hívás megvalósítása jelentette. Ennek a problémának a megoldása elkerülhetetlen volt, hiszen a validálni kívánt protokoll JAR fájlok formájában áll rendelkezésre.

A JavaCall hívások előtt először be kell tölteni és inicializálni kell egy JVM-et (Java Virtual Machine), ezt a következő kódsorral tehetjük meg, ahol az átadott paraméterek JVM inicializációs argumentumok, jelen esetben ez csak a maximális Java Heap mérete:

JavaCall.init(["-Xmx128M"])

Júliából a konstruktor híváson kívül csak Java függvényeket lehet hívni [6], illetve osztály adattagokat lekérdezni Java Reflection szerűen. Azonban például adattagokat beállítani nem lehet közvetlenül, csak függvényhíváson keresztül.

A JavaCall hívásokat sikerült jól leválasztani a program többi részéről, létrehozva egy interfészt a Java és Julia kód között.

#### Konstruktor használata Java objektumok példányosításához

A konstruktor hívás két lépésben történik. Az első az, hogy létre kell hozni egy hivatkozást az osztályra:

Java\_class\_name = @jimport java.net.URL

Ez után jöhet a példányosítás:

Java\_object = Java\_class\_name((param1\_type, param2\_type, …), param1\_value, param2\_value, …)

A példányosítás után már rendelkezésünkre áll a Java objektum Julia kódban.

#### Jcall használata

A létrehozott Java objektumon, meg lehet hívni a metódusait a következő formában:

Return\_value = jcall(Java\_object, „method\_name”, return\_type, (param1\_type, param2\_type, …), param1\_value, param2\_value, …)

#### Jfield használata

A JavaCall könyvtár csak az adattagok lekérdezését teszi lehetővé, a változtatását kizárólag setter függvényhívásokkal lehet megtenni. A jfield szintaktikája hasonló a jcall-hoz:

Field\_value = jfield(Java\_object, „field\_name”, field\_type);

### Differenciálegyenletek megoldása

A szimuláció egy pontján, egy adott terápia alkalmazása után ki kell számítani a következő vércukormérési értéket (ezt valós esetben a nővér méri meg kb. 1, 2 vagy 3 órával később). Mivel rendelkezésünkre áll az ICING2, validált matematikai modell [7], amely helyesen írja le az emberi szervezet metabolikus folyamatait a vércukor háztartás szempontjából, valamint rendelkezésünkre állnak a páciens historikus SI adatai (állapotának változása), ezért az aktuális kiindulási állapotból az adott táplálási és inzulin bemeneti adatok ismeretében számítani tudjuk, hogy mi lesz a páciens következő mért vércukor értéke.

A szimuláció során szisztematikusan egy kezdeti értékből próbálom megjósolni a vércukorszint további alakulását a változó paraméterek függvényében a következő mérési időpontig. Ennek a változásnak az időbeli alakulása leírható az alább látható ICING2 modell differenciálegyenleteivel (lásd (1) - (7) egyenletek), amelyekben a változók a táblázatban (Táblázat 1) látott jelentéssel bírnak.

|  |  |  |
| --- | --- | --- |
| Model  var. | Description | Numerical value [typical range] |
|  | Endogenous glucose clearance | 0.006 |
|  | Insulin sensitivity |  |
|  | Saturation of insulin-dependent glucose clearance and receptor-bound insulin clearance from interstitium | 165 LmU |
|  | Rate of glucose transfer between the stomach and gut |  |
|  | Rate of glucose transfer from the gut to the bloodstream |  |
|  | Maximal disposal rate from the gut |  |
|  | Basal endogenous glucose production (unsuppressed by glucose and insulin concentration) |  |
|  | Non-insulin mediated glucose uptake by the central nervous system |  |
|  | Glucose distribution volume |  |
| , | Rate of transport between plasma and interstitial insulin compartments |  |
|  | Saturation of plasma insulin clearance by the liver |  |
|  | Insulin distribution volume |  |
|  | First-pass hepatic insulin clearance |  |
|  | Clearance of insulin from plasma via the renal route |  |
|  | Clearance of insulin from plasma via the hepatic route |  |

Az ICING2 modellben használt élettani paraméterek

()

()

()

()

()

()

()

ICING differenciálegyenletek

Ehhez implementálni kellett és meg kellett oldani az ICING modell differenciálegyenlet rendszerét Julia nyelven.

Ezeknek az egyenleteknek az implementálására tökéletesen alkalmas volt a Julia OrdinaryDiffEq könyvtára, amely 4-ed rendű Runge-Kutta módszerrel dolgozik (Tsit5). A könyvtár a kezdeti értékek megadása után numerikusan, tetszőleges pontossággal oldja meg az egyenletrendszert.

prob=ODEProblem(ICING\_model\_ODE!,ODEinit,(insulinTime[i],insulinTime[i+1]))

Ints = solve(prob, Tsit5(), reltol=1e-8, abstol=1e-8)

A felhasznált könyvtár segítségével, a fenti két sorral meg lehet oldani egy definiált differenciálegyenletrendszert két időpont között, bizonyos kezdeti értékek mellett. A megoldási adatsor az Ints változóba kerül.

### Statisztikai kiértékelés

A mért értékek kiértékelése céljából elkészült egy statisztikát előkészítő, számoló és ábrázoló modul.

A szimulátor által előállított nyers adatokat, páciensenként egy-egy JLD2 kiterjesztésű fájlba menti el a program. A statisztikát számoló modul ezeket használja fel bemeneti adatként, illetve az adott szimuláció statisztikáit egy CSV kiterjesztésű fájlba menti el.

A statisztika rendkívül széles spektrumú, több megközelítést alkalmaz:

* Whole Cohort Statistics

Ezek az adatok egy általános képet adnak a szimulációról olyan adatokkal, mint például a kezelési órák száma, vércukormérések száma.

* Raw BG stats

Ezek a statisztikák a nyers szimulált adatokon számolnak olyan értékeket ki, mint például a mért vércukor értékek mediánja, átlaga, szórása, valamint a mért értékek hány százaléka esik egy bizonyos intervallumba.

* Hourly Resampled BG Stats

Ezek a statisztikák hasonló értékeket számolnak ki, mint az előző pontban említettek, azzal a különbséggel, hogy itt végbemegy egy előfeldolgozási folyamat is, ami annyit jelent, hogy óránként újramintavételezi a nyers adatokat és ezeken végzi el a statisztikai kiértékelést.

* Per-episode statistics

Ezek a statisztikák a külön kezelésekről fogalmaznak meg információkat, nem az összesített adathalmazról.

* Intervention Cohort Stats

Ezek az adatok arról árulnak el információt, hogy összesítve milyen volt a betegek táplálása és inzulin adagolása.

* Intervention Per-episode Stats

Ezek a statisztikák hasonló értékeket számolnak, mint az előző pontban említettek, azzal a különbséggel, hogy itt nem összesítve, hanem kezelésenként hajtódik végre a számítás.

### Mintavételezés

Az előző fejezetben említésre került az egyik fajta statisztikához szükséges előfeldolgozási folyamat, a mintavételezés. Mivel több fajta protokoll van a rendszerben, amelyek különböző időzítéseket használnak, ez nem feltétlenül egy triviális feladat. Ez alapvetően kétféle lehet, az egyik amikor a vércukorértékeket kell óránként mintavételezni a differenciálegyenletrendszer által előállított sűrű lépésközű adatfolyamból, a másik, amikor az óránként bevitt inzulin és tápanyag mennyiséget kell kiszámolni.

**function resampleHourlyBG(timeSolnT::Vector{Float64}, timeSolnGIQ::Matrix{Float64})**

    hourlyBG = [timeSolnGIQ[1,1]]

    hours = 1

    for i in 2:length(timeSolnT)

        if timeSolnT[i] >= hours \* 60

            push!(hourlyBG, timeSolnGIQ[i,1])

            hours = hours + 1

        end

    end

    return hourlyBG

end

**function resample\_hourly(mtx::Matrix{Float64}, func)**

    mtx\_hourly = [ func( mtx[1,2]) ]

    cnt = 0.0

    for i in 2:length(mtx[:,1])

        if (mtx[i,1] - mtx[i-1,1] + cnt) < 60

            mtx\_hourly[end] = mtx\_hourly[end] + func( mtx[i,2] )

            cnt = cnt + (mtx[i,1] - mtx[i-1,1])

        end

        if (mtx[i,1] -mtx[i-1,1] + cnt ) >= 60

&& (mtx[i,1] - mtx[i-1,1] + cnt) < 120

            push!(mtx\_hourly, func( mtx[i,2] ) )

            cnt = cnt - 60 + (mtx[i,1] - mtx[i-1,1])

        end

        if (mtx[i,1] - mtx[i-1,1] + cnt) >= 120

&& (mtx[i,1] - mtx[i-1,1] + cnt) < 180

            push!(mtx\_hourly, func( mtx[i,2] ) )

            push!(mtx\_hourly, func( mtx[i,2] ) )

            cnt = cnt - 120 + (mtx[i,1] - mtx[i-1,1])

        end

        if (mtx[i,1] - mtx[i-1,1] + cnt) >= 180

            push!(mtx\_hourly, func( mtx[i,2] ) )

            push!(mtx\_hourly, func( mtx[i,2] ) )

            push!(mtx\_hourly, func( mtx[i,2] ) )

            cnt = cnt - 180 + (mtx[i,1] - mtx[i-1,1])

        end

    end

    return mtx\_hourly

end

### Adatok ábrázolása

A Julia nyelv Plots könyvtára nagyon rugalmas és változatos diagram készítési lehetőségeket támogat. A szervezet metabolikus állapotát leíró paraméterek, köztük a vércukor értékek, valamint a kezelési paraméterek grafikus ábrázolása nagyon tanulságos tud lenni. Az előzőekben említett statisztikák mellett tehát a vizuális megjelenítés is nagyban hozzájárul a helyes következtetés levonásához az alkalmazott kezelést illetően. Ezen okokból kifolyólag különös elkészült egy ábrázoló modul, amely többek közt remekül szemlélteti a páciens aktuális metabolikus állapotát mérési pillanattól mérési pillanatig. Továbbá lehetőség van összehasonlítani vele két kezelési módot, egyszerű időfüggő grafikon, előjeles különbséget ábrázoló hisztogram, valamint CDF segítségével.

## Elkészült rendszer bemutatása

Az elkészült rendszer 3 nagyobb koherens egységből, valamint 21 Julia forrásfájlból áll.

### Felsőszintű architektúra

A JavaCall komponens intézi az összes JVM-mel való kommunikációt, egy interfészt kínál a többi modul számára.

A Simulator komponens felhasználja a patients\_data/original mappában lévő betegek adatait és elvégzi a kívánt szimulációt. Ez után az eredményt elmenti a patients\_data/simulated mappába.

A Statistics komponens felelős minden statisztikai vonatkozású feladatkörért. A konkrét statisztikai számítások mellett az adatok szerializációja, előfeldolgozása, illetve megjelenítése is a felelősségi körébe tartozik. Az eredményeit a graphs, illetve sim\_stats mappákba menti.

High Level architektúra ábra

### Részletes megvalósítás

A projekt strukturális felépítése a következő:

* + src
    - JavaCall
      * Star1libs
      * JavaCallHelper.jl
      * JavaClasses.jl
      * setup\_java\_libraries.jl
      * SPRINT\_whole\_cohort.StochasticModel
    - Simulator
      * runSimulations.jl
      * runSimulationOnPatients.jl
      * simulateOnePatientJul.jl
      * HISTORIC\_controller\_simulator.jl
      * SIMPLE\_controller\_simulator.jl
      * STAR\_controller\_simulator.jl
      * ICING2\_model\_sim\_init.jl
      * ICING2\_model\_solver.jl
      * BG\_sensor.jl
      * Simulation\_Structs.jl
    - Statistics
      * runStatistics.jl
      * JuliaStatistics.jl
      * StatisticsCalculator.jl
      * StatisticsExporter.jl
      * Visualiser.jl
      * VisualiserExporter.jl
      * Serializer.jl
      * Resampler.jl
  + patiens\_data
    - original
    - simulated
  + graphs
  + sim\_stats

#### A szimuláció folyamata

A felhasznált betegek historikus adatai a patients\_data/original/HU\_java és patients\_data/original/HU\_mat mappákban vannak. A Java fájlok esetében, betegenként külön mappákba szervezve egy-egy .GUIData, .TimeSoln meg .PatientStruct java bináris fájlban vannak eltárolva az adatok. Sikerült olyan deszerializáló interfészt létrehozni, amely ugyanúgy képes működni Java bináris, MAT, illetve JLD2 (Julia szerializációs adattípus) bemeneti fájlok esetén.

A szimuláció futási eredménye-képpen létrejön egy Simresults-actual-date nevű mappa a patients\_data/simulated/julia\_results elérési útvonalon, amelybe páciensenként egy JLD2 fájlba szerializálódnak ki az aktuális szimuláció eredményei.

A runSimulations.jl a program szimulációs részének a belépési pontja, itt lehet beállítani, hogy milyen paraméterekkel fusson a rendszer, ebből kerül meghívásra a runSimulationOnPatients() függvény. Ez az utóbbi modul végzi az eredménymappa létrehozását, ha szükséges meghívja a setup\_java\_libraries() függvényt, ami inicializálja a JVM-et, majd végighaladva a bemeneti mappa tartalmán, minden betegre meghívja a simulateOnePatientJul() függvényt.

A SimulateOnePatientJul() függvény futtatja egy betegre a kiválasztott típusú szimulációt. Először beolvassa a beteg adatait a bemeneti mappából, amely az egységes interfésznek köszönhetően, amely elrejti, hogy valójában milyen típusú fájlból is történik a beolvasás az alábbi egyszerű módon végezhető el:

patient = Simulation\_Structs.Patient()

serPatient = Serializer.deserialize(srcPath, name)

patient.Treal\_orig = serPatient.Treal

patient.Greal\_orig = serPatient.Greal

patient.u\_orig = serPatient.u

patient.P\_orig = serPatient.P

patient.PN\_orig = serPatient.PN

patient.Uo = serPatient.Uo

patient.Po = serPatient.Po

patient.rawSI = serPatient.rawSI

patient.GoalFeed = serPatient.GoalFeed

A fenti módon létrejön a Patient() adatstrukúra, ezen kívül még egy TimeSoln() változó is szükséges a szimulációhoz. Ezek után meghívódik az ICING2\_model\_sim\_init() függvény, ezzel inicializálódnak a használt adatstruktúrák. Majd ciklusban futtatódik az ICING2\_model\_solver(), BG\_sensor(), illetve a kiválasztott kontrollert megvalósító függvények. Az ICING2\_model\_solver() modul felelős a differenciálegyenlet-rendszer vagyis az ICING2 modell megoldásáért. A BG\_sensor() valósítja meg a virtuális „vércukorszint mérést”, vagyis veszi az előzőleg megoldott differenciálegyenlet utolsó értékét és eltárolja. A kontrollerek a tápanyag, illetve inzulin bevitelt szabályozzák különböző módokon. A STAR\_controller\_simulator() a JAR fájlokba hív bele, vagyis a STAR protokollt futtatja. A SIMPLE\_controller\_simulator() egy egyszerű kontroller, amely csak az aktuális vércukorszint alapján javasol kezelési módot. A HISTORIC\_controller\_simulator() az eredeti kezelési adatokat felhasználva számítja a ki a vércukorszinteket, ez azt jeleni, hogy a virtuális méréseket a valós mérési időpontokban végzi el (a STAR ugyanis pontosan 1-2 esetleg 3 óránként), valamint az inzulin és tápanyag adagolás is olyan időben és mennyiségben történik, ahogy az a valós kezelés során is.

Simulation\_Structs.jl fájlban vannak definiálva a szimulációhoz használt Julia adatstruktúrák.

A Java hívásokban két modul vállal szerepet. Az egyik a JavaClasses.jl, amelyben a használt Java osztályok vannak felsorolva, a másik a JavaCallHelper.jl, amelyben Julia függvényekbe vannak becsomagolva Java függvényhívások. A SPRINT\_whole\_cohort.StochasticModel…

Szekvencia diagramok

#### A statisztikai kiértékelés folyamata

### Elkészült rendszer értékelése mérnöki szempontból

A Julia nyelv tudományos számításokra lett optimalizálva és a rendszernek a két fő aspektusa a differenciálegyenletek megoldása és a számított nagy adathalmazon számolt statisztika, ezért ez nagy előnyt jelent a hatékonyságban és gyorsaságban.

Annak ellenére, hogy a Julia nyelv nem támogatja az objektum orientált programozást, sikerült létrehozni egy interfészt a Julia és Java között, tehát a Java hívások jól le vannak választva a program többi részéről, így téve könnyen bővíthetővé, illetve átláthatóbbá a kódot.

A Julia alapvetően nem típusos nyelv, ezért nagy a kockázata a futás idejű hibáknak. Ez azzal lett kiküszöbölve, hogy ahol ésszerű ott a változóknak, paramétereknek feltüntettem az elvárt típusát is.

### Elkészült rendszer értékelése orvosi szempontból

## Rendszer validálása

A rendszer validálásához 112 páciens historikus kezelési adatai állnak rendelkezésre. A rendszert több megközelítéssel is lehet ellenőrizni. Az egyik, hogy a régi Matlab szimulációs környezet eredményeit vannak referenciaként véve, a másik lehetőség, hogy az egyes protokoll eredményei, statisztikája önmagában, orvosi szempontok alapján értékelődnek ki.

SignDiffGraph-matlab

# Összefoglalás

A szakdolgozatom keretében sikerült kifejlesztenem egy in-silico szimulációt megvalósító rendszert.

A főbb elért eredmények a munka során:

* A klinikai adatok beolvasása a virtuális betegek létrehozása céljából;
* A STAR protokoll Java függvényhívásai Julia kódból;
* A szimulált vércukormérést megvalósító differenciálegyenlet rendszer implementálása;
* Statisztikai számítások implementációja;
* Az elkészült funkciók tesztelése valós körülmények között.

Az elvégzett munka előremutató ezen a területen, hiszen az elkészült rendszer könnyen használható kutatási célokra, illetve orvosi területen és mérnöki vonatkozásban is értékes eredményeket sikerült elérni.

## További fejlesztési lehetőségek

Irodalomjegyzék

1. J. L. Dickson, K. W. Stewart, C. G. Pretty, M. Flechet, T. Desaive, S. Penning és B. C. Lambermont, „Generalisability of a Virtual Trials Method Glycaemic Control in Intensice Care,” IEEE Transaction on Biomedical Engineering, pp. 65: 7 pp. 1534-1553, 11 p., 2016.
2. F. M. Liam, A. J. Le Compte, G. M. Shaw, S. Penning, T. Desaive és G. J. Chase, „STAR Development and Protocol Comparison,” IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, pp. vol. 59, no. 12, pp. 3357-3364, 2012.
3. A. Evans, A. Le Compte, C.-S. Tan, L. Ward, J. Steel, C. G. Pretty, S. Penning, F. Suhaimi, G. M. Shaw, T. Desaive és G. J. Chase, „Stochastic Targeted (STAR) GlycemicControl: Design, Safety, and Performance,” Journal of Diabetes Science and Technology, pp. 6(1):102-15, 2012.
4. L. M. Fisk, A. J. Le Compte, G. M. Shaw, S. Penning, T. Desaive and J. G. Chase, "STAR Development and Protocol Comparison," in IEEE Transactions on Biomedical Engineering, vol. 59, no. 12, pp. 3357-3364, Dec. 2012, doi: 10.1109/TBME.2012.2214384.
5. Julia: A Fast Dynamic Language for Technical Computing. Jeff Bezanson, Stefan Karpinski, Viral B. Shah, Alan Edelman. (2012).
6. Call Java programs from Julia, https://juliainterop.github.io/JavaCall.jl/ (utolsó látogatás ideje 2021. 05. 16.)
7. Jessica Lin, Normy N. Razak, Christopher G. Pretty, Aaron Le Compte, Paul Docherty, Jacquelyn D. Parente, Geoffrey M. Shaw, Christopher E. Hann, J. Geoffrey Chase, „A physiological Intensive Control Insulin-Nutrition-Glucose (ICING) model validated in critically ill patients”, Computer Methods and Programs in Biomedicine, Volume 102, Issue 2, 2011.

Függelék