Mat á styrkgildum umbrotsefna í efnaskiptalíkönum með Monte Carlo hermun

Freyr Sverrisson

Raunvísindadeild
Háskóli Íslands
2015

Mat á styrkgildum umbrotsefna í efnaskiptalíkönum með Monte Carlo hermun

Freyr Sverrisson

15 eininga ritgerð sem er hluti af
Baccalaureus Scientiarum gráðu í Verkfræðilegri eðlisfræði

Leiðbeinandi
Steinn Guðmundsson

Raunvísindadeild
Verkfræði- og náttúruvísindasvið
Háskóli Íslands
Reykjavík, maí 2015

Mat á styrkgildum umbrotsefna í efnaskiptalíkönum með Monte Carlo hermun
15 eininga ritgerð sem er hluti af *Baccalaureus Scientiarum* gráðu í Verkfræðilegri eðlisfræði

Raunvísindadeild
Verkfræði- og náttúruvísindasvið
Háskóli Íslands
Dunhagi 5
107 Reykjavík

Sími: 525 4000

Skráningarupplýsingar:
Freyr Sverrisson, 2015, *Mat á styrkgildum umbrotsefna í efnaskiptalíkönum með Monte Carlo hermun*, BS ritgerð, Raunvísindadeild, Háskóli Íslands, 31 bls.

ISBN

Prentun: Háskólaprent ehf.
Reykjavík, maí 2015

Útdráttur

Skortur á gögnum hefur lengi hamlað þróun á kvikum líkönum í kerfislíffræði. Erfitt er að mæla gildi hraðafasta og styrkgildi allra umbrotsefna í lífverum. Tilraun var gerð til að meta styrkgildi umbrotsefna í grunnefnaskiptalíkani *E.coli* með slembisýnatöku á varmafræðilega mögulegum ástöndum. Með þeim má reikna beint út sérstaka hraðafasta sem eiga að nálga kvika eiginleika frumunnar. Einnig er fjallað um C++ útfærslu af ACHR-sampling reikniritinu sem nýtt var við slembisýnatökuna. Styrkgildin sem fengust úr slembisýnatökunni pössuðu illa við mæld gildi úr *E.coli*. C++ útfærsla ACHR-sampling reikniritsins var ekki mikið hraðari en MATLAB útfærsla þess.

Abstract

Lack of data has long hampered the development of dynamic models in Systems Biology. It is difficult to measure the values of kinetic constants and all metabolite concentrations in organisms. An attempt was made to estimate metabolite concentrations in a model of the central metabolism of *E. coli* by random sampling of thermodynamically feasible states. Then, specific rate constants that are supposed to approximate the dynamic properties of the cell can be calculated directly. Also discussed is an C++ implementation of the ACHR-sampling algorithm that was used during the sampling process. The concentration values obtained from random sampling fit poorly to measured values from *E.coli*. The C++ implementation of the ACHR-sampling algorithm was not much faster than MATLAB implementation.

Efnisyfirlit

Kafli 1

Inngangur

Upplýsingaöflun á líffræðilegum gögnum með háafkastaaðferðum svo sem háafkastaraðgreiningu (e. high-throughput sequencing) veitir okkur miklar upplýsingar um einstaka hluta lífvera. Þessir einstöku hlutar veita okkur þó takmarkaða þekkingu á virkni frumna. Það er ekki einungis mikilvægt að vita úr hvaða hlutum fruman er byggð heldur þurfum við líka að geta sagt hvernig þeir verka hver á annan.

Kerfislíffræðin (e. systems biology) leitast við að safna þessum gögnum saman, tengja hlutana saman og mynda kerfisfræðileg líkön af virkni lífverunnar. Með slík líkön við höndina má herma hegðun frumunnar í tölvu við ákveðin skilyrði sem hefur augljósa hagnýtingu í t.d. líftækni, læknisfræði og við lyfjaþróun.

Gróflega má skipta hlutum frumu og tengingum þeirra á milli upp í þrenns konar net. Þau eru efnaskiptanet (e. metabolic networks) sem lýsa niðurbroti og uppbyggingu á lífefnum, reglunarnet (e. transcriptional regulatory networks) sem lýsa stjórnun á tjáningu gena og merkjanet (e. signaling networks) sem lýsa upplýsingaöflun frumna um ytra umhverfi sitt og samskiptum milli frumna. Þessi net tengjast síðan saman og mynda eitt heildarnet sem lýsir allri starfsemi frumunnar.

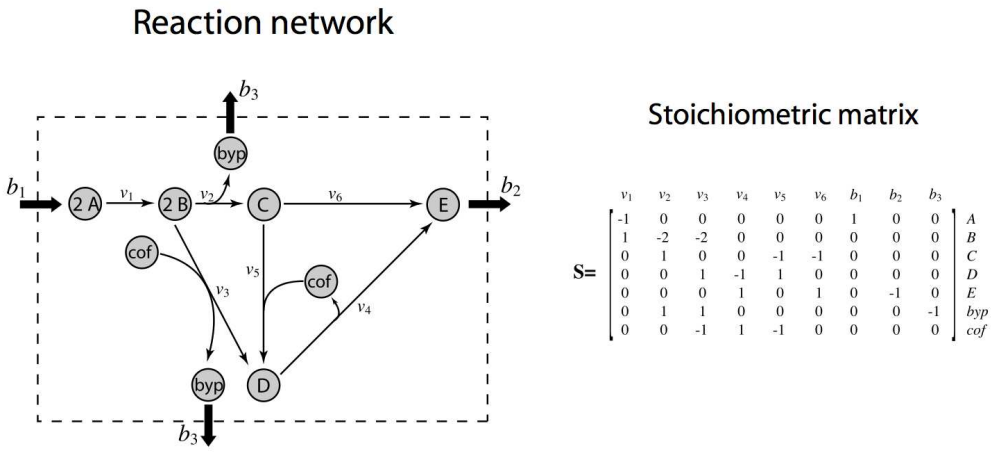
Það er að mörgu leiti erfitt að byggja upp þessi net við líkanagerð. Eins og áður hefur komið fram veitir háafkastaupplýsingaöflun upplýsingar um einstaka hluta netsins en tengingarnar þeirra á milli eru langt í frá augljósar. Ekki bætir úr að hver hluti getur haft margvíslegum hlutverkum að gegna í mismunandi netum. Þannig getur svo dæmi sé tekið umbrotsefni (e. metabolite) í efnaskiptaferli átt þátt í stýringu á tjáningu gens (t.d. lac genagengið).

Vegna rannsókna lífefnafræðinga í gegnum árin hefur safnast upp mikil þekking á efnaskiptaferlum og hvarfgangi ensíma. Þekking á reglun og merkjum er ófullkomnari. Því hefur gengið langbest að útbúa líkön af efnaskiptum

og rannsaka þau án þess að taka tillit til reglunar og merkja. Efnaskipti eru þau efnahvörf sem eiga sér stað í lifandi frumum. Þeim er skipt í sundrunarferli, þar sem flóknari efni brotna niður í smærri sameindir og nýmyndunarferli, þar sem flókin efni eru byggð upp úr einfaldari einingum. Í lífverum skipta þessi hvörf þúsundum og saman mynda þau flókið net þar sem efni lífverunnar eru tengd saman í gegnum efnahvörfin.

Líkön af efnaskiptaferlum eru af tvennum toga: kvik (e. dynamic) og kyrrstæð (e. static). Kvik líkön lýsa tímaháðum eiginleikum frumunnar en kyrrstæð gera það ekki.

Náðst hefur árangur með því að kortleggja sístöðuástönd efnaskiptaferla [1, 3]. Sístöðuástönd efnaskiptanna eru þau ástönd þar sem styrkur hvers efnis í frumunni breytist ekki (þetta er því kyrrstætt líkan). Til að kortleggja þessi ástönd er hlutfallsstuðlum (e. stoichiometric coefficients) efnahvarfanna safnað saman í stuðlafylkið (e. stoichiometry matrix) S (sjá dæmi á mynd 1.1). Táknum styrk m umbrotsefna með vigrinum $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$. Breyt-



Mynd 1.1: Dæmi um efnaskiptanet og tilsvarandi stuðlafylki. Myndin er fengin úr *Systems Biology: Properties of Reconstructed Networks* [1].

ingu í styrk umbrotsefna má þá tákna með $\frac{d\boldsymbol{x}}{dt} = S \cdot \boldsymbol{v}$, þar sem stökin í $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ eru flæði gegnum efnahvörfin. Sístöðuástöndin eru þá þeir flæðisvigrar sem uppfylla $S \cdot \boldsymbol{v} = \mathbf{0}$, þ.e. þeir vigrar sem eru í núllrúmi stuðlafylkisins.

Til eru nokkrar aðferðir við við að ákvarða eiginleika núllrúmsins. Þær sem reynst hafa best þegar efnaskiptanetin verða gríðarstór (e. genome-scale metabolic networks) eru slemisýnataka (e. random sampling) [5] og línuleg bestun á völdu markfalli. Slembisýnataka leitast við að finna allar mögulegar

lausnir sem uppfylla skorðurnar en línuleg bestun reynir að finna eina lausn sem uppfyllir skorðurnar og hámarkar/lágmarkar markfallið. Hægt er að skilgreina svokallað lífmassamarkfall (e. biomass objective function) sem er mælikvarði á vöxt lífverunnar. Með því að hámarka lífmassamarkfallið má fá ástand sem líkst getur raunverulegu ástandi frumunnar að því gefnu að lífveran reyni að vaxa sem mest.

Ástæðan fyrir áhuga á þessum sístöðuástöndum er samvægi (e. homeostasis) frumna. Undir nær stöðugum aðstæðum breytist styrkur metabólíta ekki mikið og sagt er að ástand frumu sé breytilegt sístöðuástand (e. dynamic steady state). Ókostir þessarar nálgunar eru þeir að ómögulegt er að segja til um hegðun frumu þegar umhverfisþættir breytast að miklu leyti. Einnig er stýribúnaður frumunnar ekki innbyggður í líkanið sem dregur úr nákvæmni og takmarkar notagildi þess talsvert.

Kvik líkön af starfsemi frumna eru hið heilaga gral kerfislíffræðinnar. Kvik líkön í kerfislíffræði eru ýmist fullvissulíkön (e. deterministic models) eða slembilíkön (e. stochastic models). Fullvissulíkön skila alltaf sömu hegðun ef upphafsskilyrðin eru óbreytt en slembilíkön eru að einhverju leiti handahófskennd og hegðunin er gefin með líkindafalli. Það sem hamlað hefur þróun á kvikum líkönum er skortur á mælingum á kvikum eiginleikum frumna svo sem hraðaföstum og styrkgildum ensíma og umbrotsefna. Gögn af þessu tagi eru af skornum skammti og mikilvægt er fyrir þróun greinarinnar að þeirra sé aflað.

Í þessari skýrslu verður fjallað um slembisýnatöku á varmafræðilega mögulegum ástöndum til að meta styrkgildi umbrotsefna í efnaskiptalíkönum. Með þeim má t.d. reikna beint út hraðafasta sem eiga að nálga kvika hegðun frumunnar, svokallaða PERCs (Pseudo-first-order rate constants) [2, 6]. Einnig verður fjallað um hagkvæma útfærslu af ACHR-sampling reikniritinu sem nýtt var við sýnatökuna og hraðamælingu á henni.

Kafli 2

Aðferðir

2.1 Varmafræðilega möguleg styrkgildi og útreikningur á PERCs

Stefna efnahvarfs ræðst af fríorkubreytingu þess, ΔG. Fríorkubreytingin er háð breytingu í óreiðu, ΔS og breytingu í vermi, ΔH [9]. Við fast hitastig, T, gildir:

$$\Delta G = \Delta H - T\Delta S. \tag{2.1}$$

Sýna má fram á að samband milli fríorkubreytingu og styrks hvarfefna og myndefna er

$$\Delta G = \Delta G^\circ + RT \ln Q \tag{2.2}$$

þar sem ΔG° er fríorkubreyting við staðalaðstæður, R er gasfastinn og Q er hlutfallið milli margfeldis á styrk myndefna og margfeldis á styrk hvarfefna.

Sagt er að efnahvarfið hafi náð jafnvægi þegar heildarflæðið gegnum það er ekkert. Þá er fríorkubreytingin engin og $Q = K_{eq} = \exp(-\Delta G^\circ/RT)$ þar sem K_{eq} er kallaður jafnvægisfasti efnahvarfsins. Í jafnvægi gildir því um efnahvarfið $\sum_{i=1}^{n} c_i x_i = 0$ að

$$\prod_{i=1}^{n} [x_i]^{c_i} = K_{eq} \tag{2.3}$$

þar sem $[x_i]$ er styrkgildi efnisins x_i og c_i er hlutfallsstuðull efnisins. Undir þessum kringumstæðum er heildarflæðið gegnum efnahvarfið $v = 0$. Ef $v > 0$ þá er

$$\prod_{i=1}^{n} [x_i]^{c_i} < K_{eq} \tag{2.4}$$

og eins

$$\prod_{i=1}^{n}[x_i]^{c_i} > K_{eq} \tag{2.5}$$

ef $v < 0$. Tökum sem sem dæmi efnahvarfið $2A + B \rightleftharpoons 2C$. Margfeldið í jöfnum (2.3)-(2.5) verður þá $\prod_{i=1}^{n}[x_i]^{c_i} = [A]^{-2}[B]^{-1}[C]^2$.

Með því að taka logrann af jöfnum (2.3)-(2.5) fyrir efnahvörf í efnaskiptalíkani má fá kúpt lausnarúm fyrir logrann af styrkgildum umbrotsefna. Um kúpt lausnarúm gildir að allir punktar sem liggja á línustrikinu milli tveggja punkta í rúminu eru einnig í lausnarúminu. Auðveldara er að ákvarða eiginleika þeirra en lausnarúma sem ekki eru kúpt. Þ.e. fyrir efnahvarf númer i í efnaskiptalíkaninu fæst

$$(S^T \cdot \log(\boldsymbol{x}))_i = (\boldsymbol{K_{eq}})_i, \text{ ef } (\boldsymbol{v})_i = 0 \tag{2.6}$$

$$(S^T \cdot \log(\boldsymbol{x}))_i < (\boldsymbol{K_{eq}})_i, \text{ ef } (\boldsymbol{v})_i > 0 \tag{2.7}$$

$$(S^T \cdot \log(\boldsymbol{x}))_i > (\boldsymbol{K_{eq}})_i, \text{ ef } (\boldsymbol{v})_i < 0 \tag{2.8}$$

$$\log(\boldsymbol{l}) \leq \log(\boldsymbol{x}) \leq \log(\boldsymbol{u}) \tag{2.9}$$

þar sem S er stuðlafylkið, $\boldsymbol{x}$ er vigur sem inniheldur styrkgildi umbrotsefnanna, $\boldsymbol{K_{eq}}$ inniheldur jafnvægisfastana, $\boldsymbol{v}$ inniheldur flæðisgildi efnahvarfanna og $\boldsymbol{l}$ og $\boldsymbol{u}$ eru neðri- og efrimörk á styrkgildunum.

Það er ýmsum vandkvæðum háð að mynda gott lausnarúm og kortleggja það. Nákvæmar mælingar á jafnvægisföstum eru ekki fyrir hendi. Í þessu verkefni var notast við jafnvægisfasta frá Goldberg et al [8]. Jafnvægisfastarnir eru mjög háðir aðstæðum svo sem hitastigi, sýrustigi og styrk hjálparþátta. Í mörgum (og raunar flestum) tilfellum fengust ekki gildi á jafnvægisföstum við þær aðstæður sem óskað var eftir. Einnig voru mælingar það strjálar að ekki þýddi að brúa þær til að meta gildin. Þegar þannig stóð á var efnahvarfinu annað hvort sleppt úr lausnarúminu eða jafnvægisfasti notaður sem mældur var aðstæður líkar þeim sem óskað var eftir. Ekki er ljóst hversu mikil áhrif það hefur að notast við ónákvæma jafnvægisfasta en ákveðið var að nota gildi sem takmörkuðu lausnarúmið sem minnst. Þ.e. há gildi fyrir jafnvægisfasta efnahvarfa sem eru í jákvæða stefnu og lág gildi fyrir þau sem eru í neikvæða stefnu.

Flæðisvigurinn sem ræður stefnu ójöfnumerkja í jöfnum (2.6)-(2.8) var ákvarðaður með slembisýnatöku. Meðalflæðisvigurinn var ákvarðaður með því að taka meðaltal af þeim sístöðuástöndum sem fengust með slembisýnatökunni og það ástand valið sem lágmarkaði evklíðslu firðina að meðalflæðisvigrinum. Með þessari aðferð fékkst vigur sem var örugglega í núllrúmi stuðlafylkisins og mögulegt ástand fyrir kerfið. Þetta þarf ekki endilega að vera rétt nálgun. Jafnvel þó það sé líklegast að fá eitthvert tiltekið flæði við

slembisýnatöku þá þarf fruman ekki endilega að kjósa það ástand. Slembisýnataka á mögulegum ástöndum veitir ekki líkindadreifingu fyrir ástöndin eins og fæst fyrir möguleg ástönd í safneðlisfræði. Fruman velur ástand sitt með stjórnbúnaði sínum. Mögulega er betra að nota þann flæðisvigur sem hámarkar lífmassamarkfallið (gekk ekki í þessu tilfelli) eða þann sem lýsir markmiðum frumunnar best. Einnig var prófað að nýta aðferðina að ofan með þeim skorðum að lágmarkið á lífmassamarkfallinu var 20% af hæsta mögulega gildi en það skilaði sömu niðurstöðum.

Skorða verður rúmið með efra og neðra marki ($\boldsymbol{u}$ og $\boldsymbol{l}$). Lausnarúmið verður að vera takmarkað og því er í flestum tilfellum ekki nóg að setja neðri mörk á styrk efnis sem 0 M. Þó má ekki takamarka rúmið um of og valda því að það verði tómt. Ágætt er að gefa stórt bil í upphafi og takmarka það smám saman.

Þegar stuðlafylkið er undirbúið fyrir sýnatöku á styrkgildum verður að gæta þess að efni og efnahvörf sem ekki eiga heima þar séu fjarlægð, svo sem vatn og lífmassahvarfið. Ytri styrkur efna er stilltur með því að breyta gildum í $\boldsymbol{K_{eq}}$. Þ.e. fyrir efnahvarfið $\rightarrow A$ verður fastinn $(\boldsymbol{K_{eq}})_i = [A_{ex}]$ þar sem $[A_{ex}]$ er ytri styrkur A.

Eftir að styrkgildin hafa verið ákörðuð má nota þau og flæðisgildi efnahvarfanna til að reikna PERCs. Almennt má skrifa hraða ensímhvataðs efnahvarfs sem $v = f(\boldsymbol{x}, [e])$ þar sem $[e]$ er styrkur ensímsins. Röðum styrkgildum efnanna í $\boldsymbol{x}$ þannig að styrkgildi hvarfefna efnahvarfsins eru númer 1 til j og styrkgildi myndefnanna eru númer j+1 til k. Gerum nú ráð fyrir að styrkur ensímsins breytist ekki og gerum Taylor nálgun um núllpunkt styrkgildanna. Þar sem hraðinn er enginn í jákvæða stefnu ef hvarfefni vantar og enginn í neikvæða stefnu ef myndefni vantar fæst að:

$$v \simeq k_+ \prod_{i=1}^{j} \boldsymbol{x}_i - k_- \prod_{i=j+1}^{k} = k_+ \left(\prod_{i=1}^{j} \boldsymbol{x}_i - \prod_{i=j+1}^{k} \boldsymbol{x}_i / K_{eq} \right) \tag{2.10}$$

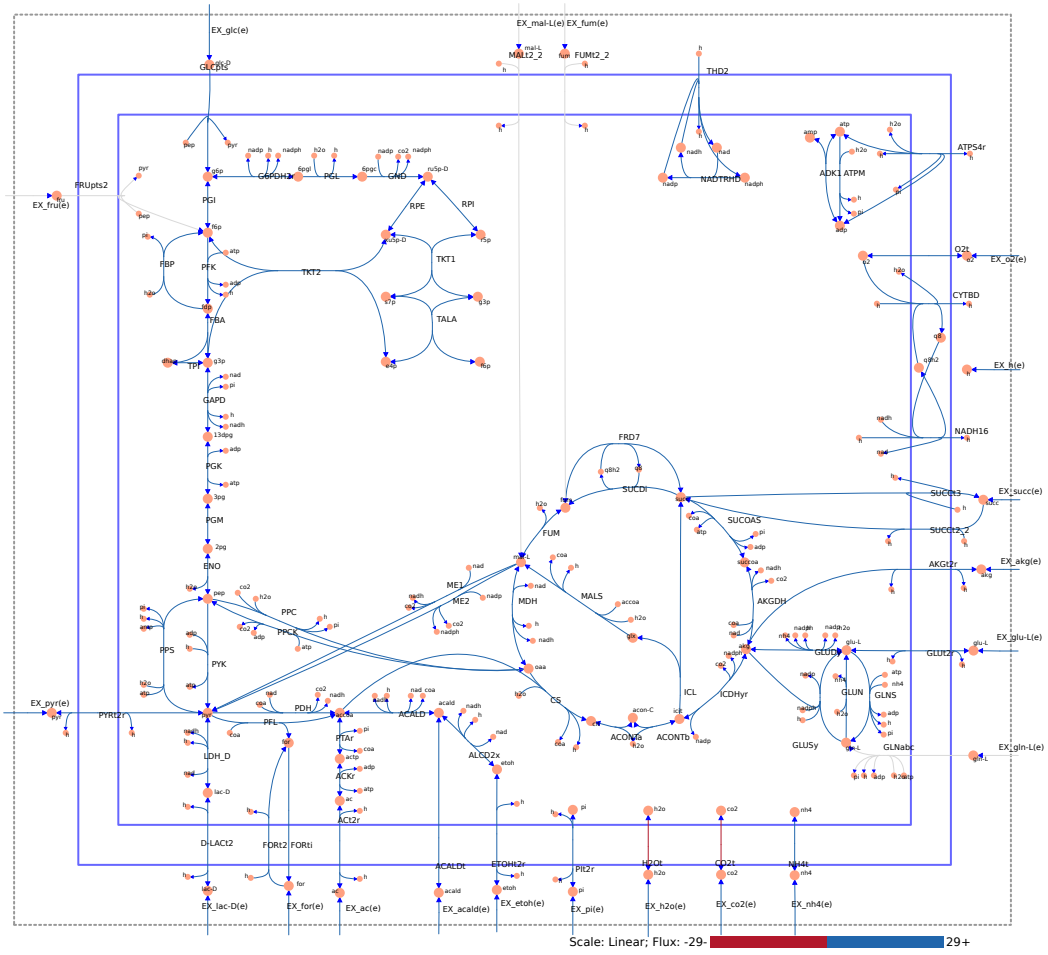
ef við hunsum liði af hærra stigi. PERC-inn fyrir þetta hvarf verður þá

$$k_+ = \frac{v}{\left(\prod_{i=1}^{j} \boldsymbol{x}_i - \prod_{i=j+1}^{k} \boldsymbol{x}_i / K_{eq} \right)} \tag{2.11}$$

Við erum sem sagt í rauninni að gera ráð fyrir að hvarfið sé ekki ensímhvatað og sé af fyrsta stigi. Fyrir efnahvarfið sem tekið var sem dæmi að ofan, $2A + B \rightleftharpoons 2C$, fengist $k_+ = v/\left([A][B] - [C]/K_{eq}\right)$

2.2 Lýsing á kerfinu

Efnaskiptanetið sem nýtt var við tilraunina var líkan af grunnefnaskiptum í *E.coli* sem kallað er E.coli Core [10]. Líkanið samanstendur af 94 efnahvörfum og 72 umbrotsefnum (sjá mynd 2.1). Áður en slembisýnatakan var framkvæmd var líkanið smækkað til að taka út efnahvörf sem aldrei var flæði um, svo sem efnahvörf sem tengdust inntöku á frúktósa og glútamíni. Slembisýnataka á styrkgildum umbrotsefna í E.coli Core líkaninu var framkvæmd með það að markmiði að bera gildin saman við mæld gildi fengin úr Bennett et al [7]. Ytri styrkur glúkósa var valinn sem 22.2 mM og ytra sýrustig sem pH 7. Önnur ytri gildi voru ýmist sett sem 10^{-10} M eða 1 M eftir því hvort flæði efnanna var út úr eða inn í frumuna. Þannig er líkanið tiltölulega lítið skorðað. Neðra mark á styrk umbrotsefna inni í frumunni var sett sem 10^{-7} M og efra mark sem 10^{-1} M. Gildi á jafnvægisföstum sem notaðir voru við slembisýnatökuna má finna í töflu B.1. Þeir voru fengnir frá Goldberg et al [8]. Jafnvægisfastar fyrir flæði efna yfir frumuhimnur (e. transport reactions) voru settir sem 1. Þetta gera alls 63 hraðafasta. Upphitunarpunktar voru í upphafi fundnir með bestun á handahófskenndu markfalli (sbr. kafli 2.3). Þannig fást punktar sem liggja á jaðri lausnarúmsins.



Mynd 2.1: E.coli Core líkanið. Flæðisvigurinn sem hefur verið teiknaður með litakóða er sá sem nýttur var við ákvörðun á stefnu ójöfnumerkja í jöfnum (2.6)-(2.8). Hvörf í jákvæða stefnu eru teiknuð með bláum lit, hvörf í neikvæða stefnu með rauðum lit og hvörf með ekkert flæði með gráum

2.3 ACHR-sampling reikniritið

ACHR-sampling (Artificial Centering Hit-and-Run sampling) kóðinn var skrifaður í C++ með sambærilegan kóða úr COBRA Toolbox [3] að fyrirmynd. ACHR-sampling reikniritið er tilbrigði við hefðbundið Hit-and-Run reiknirit og nýtir gervimiðju rúmsins (sjá skref 1 að neðan) til að ákvarða handahófskennda stefnu [4]. Þetta veldur því að ACHR-sampling reikniritið þekur betur lausnarúmið þegar rúmið er mjög mislangt eftir því í hvaða stefnu er farið [5]. Þetta er einmitt eiginleiki lausnarúma efnaskiptalíkana. Mersenne Twister [11] reikniritið var nýtt til að búa til slembitölur og CBLAS úr vecLib umgjörðinni til að framkvæma vigra- og fylkjaaðgerðir. CBLAS er C útgáfa af BLAS (Basic Linear Algebra Subprograms) sem er safn af hröðum reikniritum fyrir einfalda fylkjaútreikninga. Kóðinn var þýddur í MATLAB til að búa til MEX-skrá (MATLAB Executable) sem gerði kleift að kalla á forritið úr MATLAB.

ACHR-sampling reikniritið er Monte Carlo reiknirit sem kortleggur opið og takmarkað mengi $V \subset \mathbf{R}^n$. Það virkar í megindráttum þannig að:

1. Valinn er punktur $\boldsymbol{x} \in V$ og fundin er gervimiðja (e. artifical center), $\boldsymbol{c}$ út frá fyrirframfengnum upphitunarpunktum (e. warmup points), $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots \boldsymbol{w}_n\}$. Þ.e. $\boldsymbol{c} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{w}_i$.

2. Einn af upphitunarpunktunum er valinn af handahófi, $\boldsymbol{w}_r$, og stefnan $\boldsymbol{d}$ valin út frá $\boldsymbol{d} = \frac{\boldsymbol{w}_r - \boldsymbol{c}}{\|\boldsymbol{w}_r - \boldsymbol{c}\|}$.

3. Finnum gildi á α_{min} og α_{max} þ.a. fyrir öll $\alpha \in (\alpha_{min}, \alpha_{max})$ þá er $\boldsymbol{x} + \alpha \cdot \boldsymbol{d} \in V$

4. Veljum $\alpha \in (\alpha_{min}, \alpha_{max})$ af handahófi og fáum nýjan punkt $\boldsymbol{y} = \boldsymbol{x} + \alpha \cdot \boldsymbol{d} \in V$

Ekki gafst tími til að útfæra ójöfnur í C++ kóðanum. Einungis gafst tími til að útfæra söfnun með jöfnuskorðum. Því MATLAB var nýtt við sýnatöku á styrkgildum (sbr. jöfnur (2.6)-(2.8)).

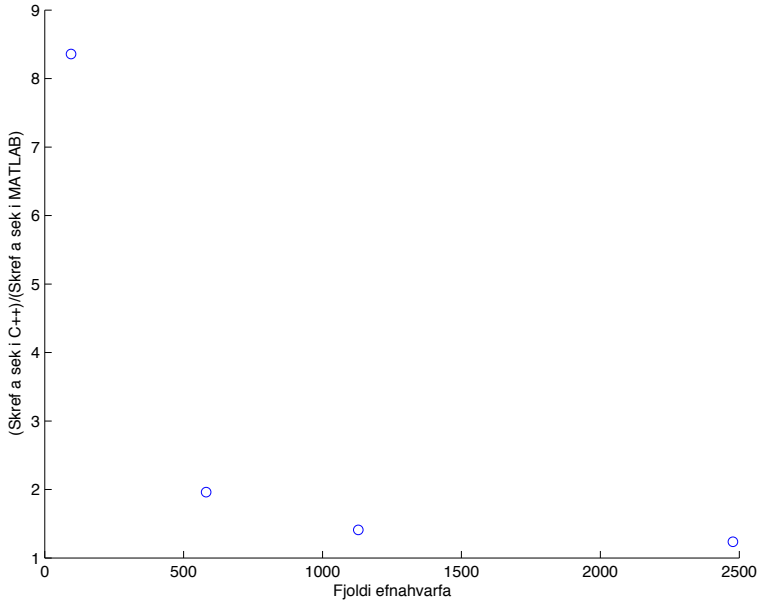
Kafli 3

Niðurstöður

3.1 Hraðaaukning í ACHR-sampling forriti

Fjöldi skrefa á sekúndu var notaður sem mælikvarði á hraða forritsins þar sem eitt skref er ein umferð af skrefum 1-4 í kafla 2.2. Borinn var saman skrefafjöldi á sekúndu í C++ útfærslu reikniritsins við skrefafjölda á sekúndu í MATLAB útfærslu þess. Niðurstöður á slembisýnatöku á sístöðuástöndum fyrir nokkur efnaskiptanet má sjá á mynd 3.1. Þar er hlutfallið milli skrefa á sekúndu í báðum forritunum sýnt sem fall af stærð efnaskiptanetsins. Efnaskiptanetin sem notast var við voru: Ecoli core (95 efnahvörf), iSR432 (581 efnahvörf) [12], IAN840m (1129 efnahvörf) og iCA1273 (2477 efnahvörf) [13].

Hraðaaukningin stóð ekki undir væntingum. Tímataka leiddi í ljós að eftir því sem netin urðu stærri fór sífellt stærri hluti tímans í að tryggja að nýir punktar færðust ekki út fyrir núllrúmið. Þá er fyrst margfeldið $S \cdot \boldsymbol{v}$ reiknað og ef punkturinn er ákveðið langt frá núllrúminu er honum varpað inn í það með $N(N^T \cdot \boldsymbol{v})$ þar sem N er fylki sem inniheldur þverstaðlaða grunnvigra stuðlafylkisins. Þetta eru stór fylkjamargfeldi sem MATLAB getur reiknað hratt og því stefnir hlutfallið á mynd 3.1 á einn.



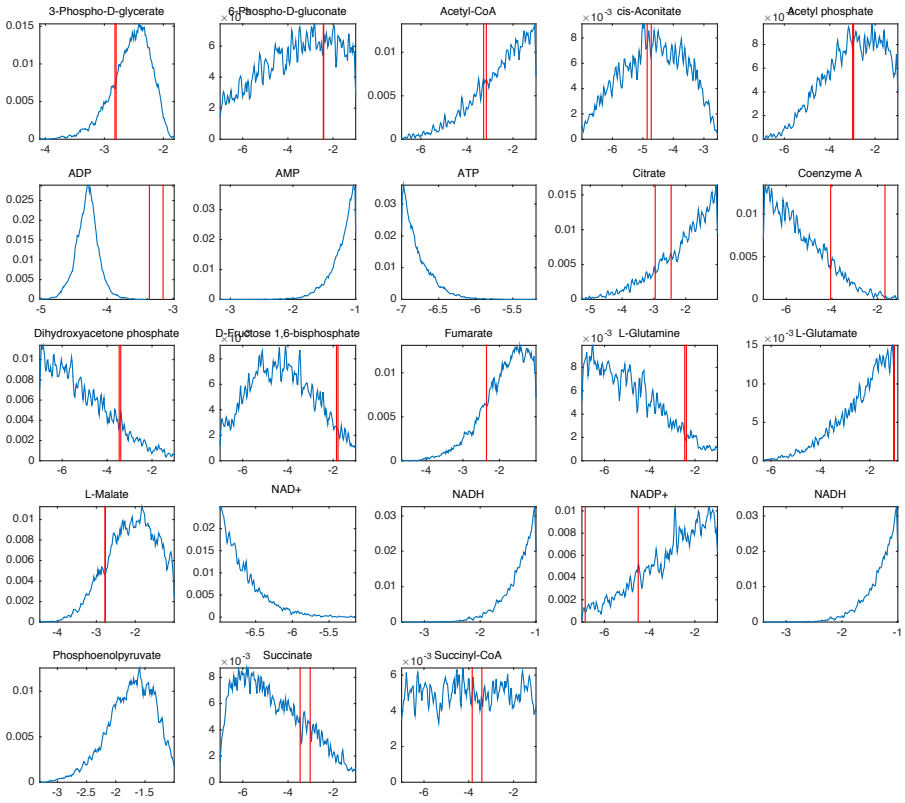
Mynd 3.1: Hraðasamanburður á C++ útfærslu og MATLAB útfærslu á ACHR-sampling reikniritinu.

3.2 Styrkgildi umbrotsefna í grunnefnaskiptalíkani *E.coli*

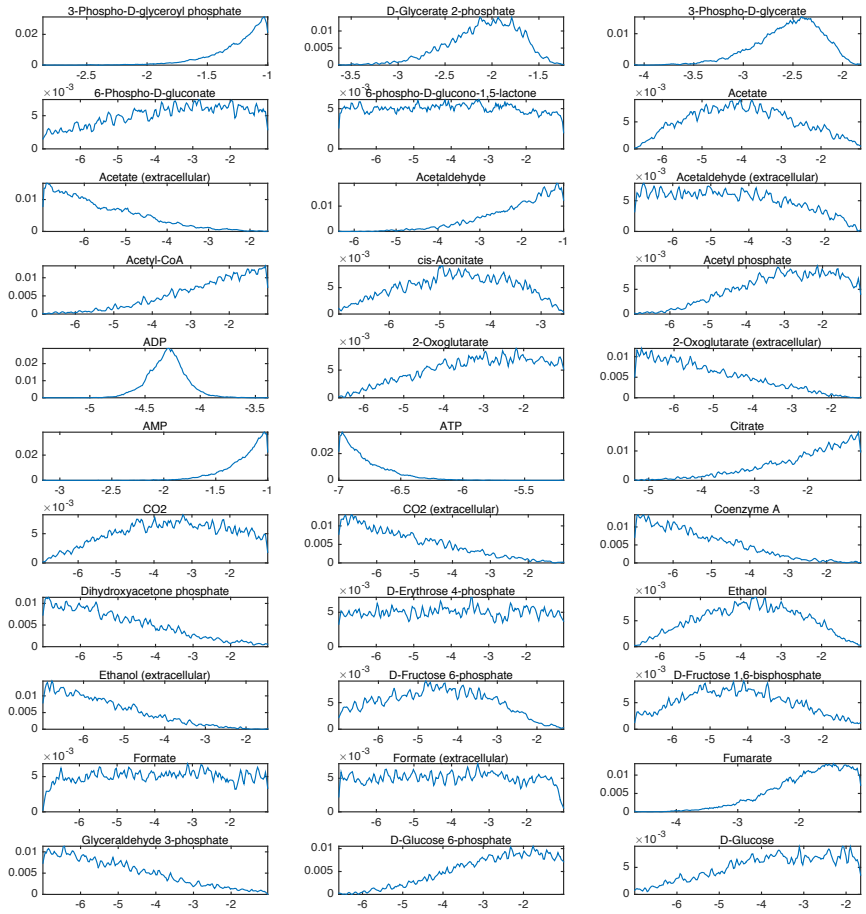
Á mynd 3.2 má sjá valdar niðurstöður úr slembisýnatökunni ásamt mörkum sem sýna 95% öryggisbil á styrkgildum fengið úr [7]. Styrkgildin fengin úr sýnatökunni teygja sig yfir stórt bil og eru í sumum tilvikum langt frá mældum gildum. Erfitt er að segja til um hvort það sé tilviljun að sum gildanna virðast passa ágætlega við mæld gildi. Þessum samanburði verður þó að taka með þeim fyrirvara að E.coli Core líkanið er ekki líkan af öllum efnaskiptum sem fara fram í *E.coli* bakteríunni. Þá má vel vera að stærra líkan skili betri niðurstöðum.

Niðurstöður úr slembisýnatökunni fyrir öll umbrotsefnin má finna á myndum 3.3 og 3.4.

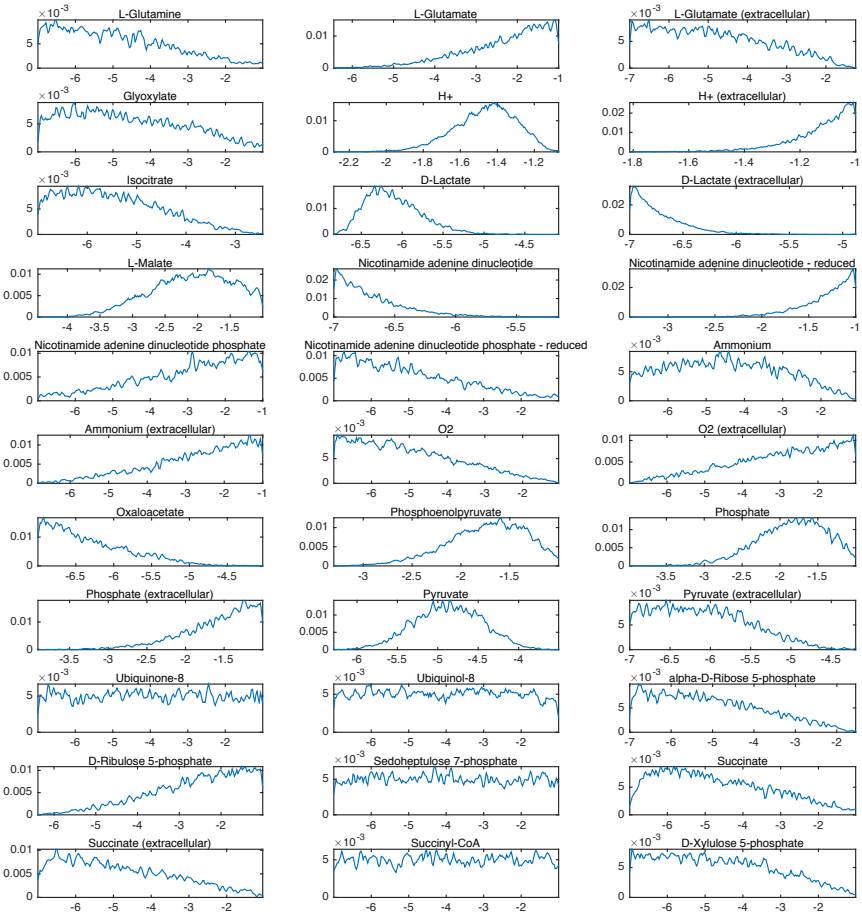
Einnig var athugað hvaða áhrif það hefði á slembisýnatökuna að skorða eitt umbrotsefnið, súkkínat, við 95% öryggisbilið. Tíðniritin (myndir 3.3 og 3.3), fyrir utan tíðnirit súkkinats, virtust ekki breytast við þetta (niðurstöður ekki sýndar).



Mynd 3.2: Valin styrkgildi fengin með slembisýnatöku borin saman við mæld styrkgildi fengin frá Bennett et al [7] ($\log_{10}$ skali). Rauðu línurnar sýna 95% öryggisbil á mælingum. Á sumum gröfum falla línurnar fyrir utan rammann.



Mynd 3.3: Mat á styrkgildum fyrir stök efnahvörf ($\log_{10}$ skali). Fyrri mynd



Mynd 3.4: Mat á styrkgildum fyrir stök efnahvörf ($\log_{10}$ skali). Seinni mynd

Kafli 4

Lokaorð

Niðurstöður úr bæði mati á styrkgildum og hraðamælingum á ACHR-sampling forritinu voru ekki eins og helst hefði verið á kosið. Matið teygði sig í mörgum tilfellum yfir stórt bil og var ekki í samræmi við niðurstöður mælinga. Samanburðinum verður að taka með þeim fyrirvara að E.coli core lýsir aðeins hluta af efnaskiptum *E.coli* bakteríunnar.

Markmiðið hafði verið að nýta niðurstöðurnar úr slembisýnatökunni til að reikna út PERCs hraðafasta og mynda þannig kvikt líkan. Ljóst varð seinna að niðurstöðurnar voru ekki nægilega góðar til að reikna hraðafastana. Einnig kom í ljós að skv. Bennett et al. eru flest ensím í *E.coli* nærri mettuð af umbrotsefnum sínum [7]. Þetta gefur til kynna að það sé rangt að gera nálgun um núllpunkt styrkgildanna og það kemur til álita hvort þessir hraðafastar dugi nokkuð við að lýsa kvikri hegðun frumunnar.

Gera þarf frekari tilraunir á takmörkun á lausnarúminu og notkun á öðrum flæðisgildum við gerð þess. Einnig væri áhugavert að sjá hvort slembisýnataka með stærri efnaskiptalíkönum skilaði betri niðurstöðum. Skortur á áreiðanlegum jafnvægisföstum gerir þó erfitt að nota aðferðina á mikið stærri líkön.

Útreikningur á stórum fylkjamargfeldum hægði verulega á ACHR-sampling forritinu svo ekki er mikill ávinningur af því að nýta það frekar en MATLAB útfærsluna við sýnatöku á stórum efnaskiptalíkönum. Mögulega mætti skoða hvort draga mætti úr fjölda þessara margfelda til að flýta fyrir ferlinu.

Heimildir

[1] B.O. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, New York (2006).

[2] B.O. Palsson. *Systems biology: Simulation of Dynamic Network States*. Cambridge University Press, New York (2011).

[3] J. Schellenberger et al. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature Protocols*, 6:1290-1307, 2011.

[4] D.E Kaufman og R.L Smith. Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research*, 46(1):84-96, 1988.

[5] J. Schellenberger (2010). Monte Carlo Simulation in Systems Biology. Ph.D Thesis. University of California, San Diego: Bandaríkin

[6] A. Bordbar & B.O Palsson. (2012). Moving Toward Genome-Scale Kinetic Models: The Mass Action Stoichiometric Simulation Approach. *Functional Coherence of Molecular Networks in Bioinformatics* (bls. 201-220). Springer New York.

[7] Bennett, B. D., Kimball, E. H., Gao, M., Osterhout, R., Van Dien, S. J., & Rabinowitz, J. D. (2009). Absolute metabolite concentrations and implied enzyme active site occupancy in Escherichia coli. Nature chemical biology, 5(8), 593-599. Chicago

[8] Goldberg, R. N., Tewari, Y. B., & Bhat, T. N. (2004). Thermodynamics of enzyme-catalyzed reactions—a database for quantitative biochemistry. Bioinformatics, 20(16), 2874-2877. Chicago

[9] Nelson, D. L., Lehninger, A. L., & Cox, M. M. (2008). *Lehninger principles of biochemistry*. Macmillan. Chicago

[10] Orth, J. D., Fleming, R. M. T., & Palsson, B. (2009). Reconstruction and Use of Microbial Metabolic Networks: the Core Escherichia coli Metabolic Model as an Educational Guide. Chicago

[11] Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1), 3-30.

[12] Roberts, S. B., Gowen, C. M., Brooks, J. P., & Fong, S. S. (2010). Genome-scale metabolic analysis of Clostridium thermocellum for bioethanol production. emphBMC systems biology, 4(1), 31. Chicago

[13] Kim, J., Lee, S. Y., Vickers, C. E., Archer, C. T., Nielsen, L. K., Jeong, H., & Park, J. H. (2011). The genome sequence of E. coli W (ATCC 9637): comparative genome analysis and an improved genome-scale reconstruction of E. coli.

Viðauki A

C++ kóði fyrir ACHR-sampling reikniritið

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <Accelerate/Accelerate.h>
#include "mex.h"
#include "matrix.h"

void mvPoint(double *newPoint, double *prevPoint, int pointsRow,
    int pointsCol, double *points, int stepsPerPoint, double *ub,
     double *lb, int ARow,int ACol, double *A, int NRow,int NCol,
     double *N);

void achr_sampler(int pointsRow,int pointsCol, double *points,int
     ARow,int ACol, double *A,double *ub,double *lb,int NRow,int
    NCol, double *N, int npoints, int warmupRow,int warmupCol,
    double *warmupPoints,double maxTime,int stepsPerPoint);

void maxminStep(double *maxmin, double *u, double *oldPoint,
    double *ub, double *lb, int nRxns, double dTol, double uTol);


void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const
    mxArray *prhs[])
{
     /*Tenging vid MATLAB: points = achr(A,ub,lb,null(A),
    warmupPoints,npoints,maxTime,stepsPerPoint). Points er fylki
    af punktum sem uppfylla A*x=0 og lb<x<ub. npoints er fjoldi
    punkta i points, maxtime hamarkstimi og stepsPerPoint er
    fjoldi umferda af reikniritinu fyrir hvern punkt.
    warmupPoints eru upphitunar punktar og npoints tharf ad vera
    staerri eda jafnt og fjoldi upphitunarpunkta*/
```

```
18
19      double *A, *ub, *lb, *N, *warmupPoints, *points;
20      int npoints, maxTime, stepsPerPoint;
21
22      A = mxGetPr(prhs[0]);
23      int ARow = (int) mxGetM(prhs[0]);
24      int ACol = (int) mxGetN(prhs[0]);
25
26      ub = mxGetPr(prhs[1]);
27      lb = mxGetPr(prhs[2]);
28
29      N = mxGetPr(prhs[3]);
30      int NRow = (int) mxGetM(prhs[3]);
31      int NCol = (int) mxGetN(prhs[3]);
32
33      warmupPoints = mxGetPr(prhs[4]);
34      int warmupRow = (int) mxGetM(prhs[4]);
35      int warmupCol = (int) mxGetN(prhs[4]);
36
37      npoints = (int) *(mxGetPr(prhs[5]));
38      maxTime = (double) *(mxGetPr(prhs[6]));
39      stepsPerPoint = (int) *(mxGetPr(prhs[7]));
40
41
42      /*oupphafsstillt fylki*/
43      plhs[0] = mxCreateDoubleMatrix(0,0,mxREAL);
44      mxSetM(plhs[0],ACol);
45      mxSetN(plhs[0],npoints);
46      mxSetData(plhs[0], mxMalloc(sizeof(double)*ACol*npoints));
47
48      points = mxGetPr(plhs[0]);
49
50      achr_sampler(ACol,npoints,points,ARow,ACol,A,ub,lb,NRow,NCol,
        N,npoints,warmupRow,warmupCol,warmupPoints,maxTime,
        stepsPerPoint);
51  }
52
53
54  void achr_sampler(int pointsRow,int pointsCol, double *points,int
        ARow,int ACol, double *A,double *ub,double *lb,int NRow,int
        NCol, double *N, int npoints, int warmupRow,int warmupCol,
        double *warmupPoints,double maxTime,int stepsPerPoint){
55
56      int nRxns = warmupRow;
57      int nWrmup = warmupCol;
58
59      long int totalStepCount = 0;
60
61      clock_t begin = clock(), end;
```

```
62
63      /*afritun upphitunarpunktana yfir i points fylkid*/
64      for(int i = 0; i<nWrmup;i++){
65
66          double *vect1 = (warmupPoints+i*nRxns);
67          double *vect2 = (points+i*nRxns);
68
69          cblas_dcopy(nRxns, vect1, 1, vect2, 1);
70      }
71
72      /*ef npoints er staerra en fjoldi upphitunarpunkta er auka
        punktum baett vid points*/
73      srand(123);
74      for(int i = nWrmup; i<npoints;i++){
75          int randPointID = rand()%nWrmup;
76          double *randPoint = (warmupPoints+randPointID*nRxns);
77
78          mvPoint((points+i*nRxns), randPoint, warmupRow, warmupCol
        , warmupPoints, stepsPerPoint, ub, lb, ARow, ACol, A, NRow,
        NCol, N);
79
80
81          totalStepCount += stepsPerPoint;
82      }
83
84
85      end = clock();
86      double timeElapsed = (double)(end - begin) / CLOCKS_PER_SEC;
87
88      /*punktarnir i points eru af handahofi faerdir til ad spanna
        rumid betur*/
89      while(timeElapsed<=maxTime){
90
91          int randPointID = rand()%npoints;
92          double *randPoint = (points+randPointID*nRxns);
93
94          mvPoint(randPoint, randPoint, pointsRow, pointsCol,
        points, stepsPerPoint, ub, lb, ARow, ACol, A, NRow, NCol, N);
95
96          totalStepCount+=stepsPerPoint;
97          end = clock();
98          timeElapsed = (double)(end - begin) / CLOCKS_PER_SEC;
99      }
100 }
101
102
103
104
105 void mvPoint(double *newPoint, double *prevPoint, int pointsRow,
```

```
int pointsCol, double *points, int stepsPerPoint, double *ub,
 double *lb, int ARow,int ACol, double *A, int NRow,int NCol,
 double *N){
106
107
108  const double maxMinTol = 0.000000009;
109  const double constraintViolation = 0.000000009;
110  const double uTol = 0.000000009;
111  const double dTol = 0.00000000000001;
112
113  double randVector[stepsPerPoint];
114
115  /*vigur med handahofskenndum skreflengdum*/
116  for(int i = 0; i<stepsPerPoint;i++){
117      randVector[i] = ((double)rand()/(double)RAND_MAX);
118  }
119
120
121
122  int nRxns = pointsRow;
123  int nPoints = pointsCol;
124
125  double oldPoint[nRxns];
126  cblas_dcopy(nRxns, prevPoint, 1, oldPoint, 1);
127
128  /*gervimidja rumsins*/
129  double centerPoint[nRxns];
130  memset(centerPoint, 0, sizeof(centerPoint));
131  double alpha = 1.0/((double)nPoints);
132
133  for(int i = 0; i<nPoints; i++)
134      cblas_daxpy(nRxns, alpha, (points+i*nRxns), 1,
centerPoint, 1);
135
136  int stepCount = 1;
137
138  /*Tokum skref i ruminu*/
139  while(stepCount <= stepsPerPoint){
140
141      /*handahofskennd stefna u fundin*/
142      int randPointID = (int)rand()%nPoints;
143      double *randPoint = (points+randPointID*nRxns);
144
145      double u[nRxns];
146      cblas_dcopy(nRxns, randPoint, 1, u, 1);
147
148      cblas_daxpy(nRxns, -1, centerPoint, 1, u, 1);
149      double unorm = cblas_dnrm2(nRxns, u, 1);
150      cblas_dscal(nRxns, 1.0/unorm, u, 1);
```

```
151
152         /*staersta og minnsta skreflengd fundin*/
153         double maxmin[2];
154         maxminStep(maxmin,u, oldPoint, ub, lb, nRxns, dTol, uTol)
    ;
155         double maxStep = maxmin[0];
156         double minStep = maxmin[1];
157
158
159
160         if((fabs(minStep) < maxMinTol && fabs(maxStep)<maxMinTol)
     || (minStep > maxStep)){
161             printf("Warning %f %f\n",minStep,maxStep);
162             continue;
163         }
164
165         double stepDist = randVector[stepCount-1]*(maxStep-
    minStep)+minStep;
166
167         /*Nyr punktur fenginn*/
168         double curPoint[nRxns];
169         cblas_dcopy(nRxns, oldPoint, 1, curPoint, 1);
170         cblas_daxpy(nRxns, stepDist, u, 1, curPoint, 1);
171
172         /*Ath hvort punkturinn se utan nullrumsins og ef svo er
    vorpum vid honum thangad*/
173         double constr1[ARow], constr2[NCol];
174         cblas_dgemv(CblasColMajor, CblasNoTrans, ARow, ACol, 1.0,
     A, ARow, curPoint, 1, 0.0, constr1, 1);
175         if(fabs(constr1[cblas_idamax(ARow, constr1, 1)])>
    constraintViolation){
176             cblas_dgemv(CblasColMajor, CblasTrans, NRow, NCol,
    1.0, N, NRow, curPoint, 1, 0.0, constr2, 1);
177             cblas_dgemv(CblasColMajor, CblasNoTrans, NRow, NCol,
    1.0, N, NRow, constr2, 1, 0.0, curPoint, 1);
178         }
179
180         /*Ath hvort punkturinn fer ut fyrir ub eda lb og ef svo
    er setjum vid hann a morkin*/
181         double constr3[nRxns], constr4[nRxns];
182         cblas_dcopy(nRxns, curPoint, 1, constr3, 1);
183         cblas_dcopy(nRxns, curPoint, 1, constr4, 1);
184
185
186         cblas_daxpy(nRxns, -1, ub, 1, constr3, 1);
187         cblas_daxpy(nRxns, -1, lb, 1, constr4, 1);
188
189
190         for(int i = 0; i<nRxns; i++){
```

```
191             if((*(constr3+i))>0){
192                 *(curPoint+i) = *(ub+i);
193             }
194             if((*(constr4+i))<0){
195                 *(curPoint+i) = *(lb+i);
196             }
197 
198         }
199 
200 
201         cblas_dcopy(nRxns, curPoint, 1, oldPoint, 1);
202         stepCount = stepCount+1;
203     }
204 
205     cblas_dcopy(nRxns, oldPoint, 1, newPoint, 1);
206 
207 }
208 
209 
210 void maxminStep(double *maxmin, double *u, double *oldPoint,
    double *ub, double *lb, int nRxns, double dTol, double uTol){
211 
212     double distUb[nRxns], distLb[nRxns], maxStepVec[nRxns],
    minStepVec[nRxns];
213 
214     /*Finnum fjarlaegdina ad efri og nedri morkum*/
215     cblas_dcopy(nRxns, ub, 1, distUb, 1);
216     cblas_dcopy(nRxns, oldPoint, 1, distLb, 1);
217 
218     cblas_daxpy(nRxns, -1, oldPoint, 1, distUb, 1);
219     cblas_daxpy(nRxns, -1, lb, 1, distLb, 1);
220 
221 
222 
223     /*Reiknum skreflengd ad efri og nedri morkum i stefnu u*/
224     for(int i = 0; i<nRxns; i++){
225 
226         if((*(u+i)>uTol) && (*(distUb+i)>dTol) && (*(distLb+i) >
    dTol)){
227 
228             maxStepVec[i] = distUb[i]/u[i];
229             minStepVec[i] = -distLb[i]/u[i];
230 
231         }
232         else if (*(u+i)<-uTol && (*(distUb+i)>dTol) && (*(distLb+
    i) > dTol)){
233             maxStepVec[i] = -distLb[i]/u[i];
234             minStepVec[i] = distUb[i]/u[i];
235         }
```

```
236             else{
237                 maxStepVec[i] = -1;
238                 minStepVec[i] = 1;
239             }
240         }
241 
242 
243 
244         double max = DBL_MAX;
245         double min = -DBL_MAX;
246 
247 
248         /*Finnum mestu og minnstu skreflengd i stefnu u sem heldur
        punktinum innan marka ub og lb*/
249         for(int i = 0; i <nRxns; i++){
250             if(maxStepVec[i]>0 && maxStepVec[i]<max){
251                 max = maxStepVec[i];
252             }
253 
254             if(minStepVec[i]<0 && minStepVec[i]>min){
255                 min = minStepVec[i];
256             }
257         }
258 
259         *maxmin = max;
260         *(maxmin+1) = min;
261 
262 }
```

Viðauki B

Jafnvægisfastar notaðir við slembisýnatöku

Tafla B.1: Jafnvægisfastar fengnir frá Goldberg et al [8].

	Jafnvægisfasti
acetaldehyde dehydrogenase (acetylating)	2000
acetate kinase	0.0087
aconitase (half-reaction A, Citrate hydro-lyase)	0.03
aconitase (half-reaction B, Isocitrate hydro-lyase)	2
adenylate kinase	0.4
alcohol dehydrogenase (ethanol)	7.7e-05
citrate synthase	2000000
enolase	4
fructose-bisphosphate aldolase	0.0001
fructose-bisphosphatase	100
fumarase	4
glucose 6-phosphate dehydrogenase	15
glutamine synthetase	1800
glutaminase	900
phosphogluconate dehydrogenase	0.079
isocitrate dehydrogenase (NADP)	0.9
Isocitrate lyase	0.03
D-lactate dehydrogenase	1.4e-05
malate synthase	30
malate dehydrogenase	2.4
malic enzyme (NADP)	0.044
phosphofructokinase	800
glucose-6-phosphate isomerase	0.5
phosphoglycerate kinase	0.0003
6-phosphogluconolactonase	80000
phosphoglycerate mutase	5
phosphotransacetylase	0.0068
pyruvate kinase	0.00013
ribulose 5-phosphate 3-epimerase	2
ribose-5-phosphate isomerase	0.3
NAD(P) transhydrogenase	1.4