令和４年度　卒 業 研 究 論 文

主論文

１型糖尿病患者を対象とした糖代謝動態数理モデルの簡略化に関する研究

工学部 電気電子工学科

令和１年度入学

氏　　名　三輪　雄太

指導教員　平田　研二

令和５年３月１日提出

目 次

第 1 章　はじめに **1**
1.1 研究背景と目的 1
1.2 論文構成 2

第 2 章　UVA/Padova モデル **3**
2.1 概要 3
2.2 代謝パラメータ 3
2.3 UVA/Padova モデルの状態方程式 3
2.3.1 皮下インスリン動態 4
2.3.2 インスリン動態 4
2.3.3 グルコース消化吸収 5
2.3.4 グルコース動態 6
2.3.5 糖新生 6
2.3.6 グルコース利用 7
2.3.7 グルコース腎排出 8

第 3 章　線形モデルと UVA/Padova モデルの非線形性 **9**
3.1 概要 9
3.2 線形モデル 9
3.2.1 線形モデルの状態方程式 9
3.2.2 線形モデルの血糖値応答 10
3.3 UVA/Padvoa モデルの主な非線形性 11
3.3.1 グルコース代謝利用式 11
3.3.2 胃内グルコース排出速度係数 11

第 4 章　双線形モデル **13**
4.1 概要 13
4.2 グルコース代謝利用式の双線形近似 13
4.3 消化吸収式の簡略化 14
4.4 インスリン動態の簡略化 15
4.5 その他非線形項の省略 15
4.6 双線形モデルの状態方程式 16

第 5 章　線形切り換えモデル **17**
5.1 概要 17
5.2 消化吸収式の線形切り換えシステム近似 17
5.2.1 切り換え点の応答に対する影響 18
5.2.2 切り換え点 Q_b, Q_d の決定 18
5.2.3 Q_b, Q_d の回帰モデル 20
5.3 線形切り換えモデルの状態方程式 23

第 6 章　双線形切り換えモデル **25**
6.1 概要 25
6.2 双線形切り換えモデルの状態方程式 25

第 7 章　構築モデルのシミュレーション **27**
7.1 概要 27
7.2 標準的な入力に対する血糖値応答 27
7.2.1 標準入力量の設定 27
7.2.2 数値シミュレーション 28
7.3 事故的な入力に対する血糖値応答 28
7.3.1 追加インスリン投与を伴わない食事摂取 28
7.3.2 過剰ボーラスインスリン投与 28

第 8 章　構築モデルの離散化誤差 **35**
8.1 概要 35
8.2 検証方法 35
8.2.1 検証結果 37

第 9 章　おわりに **39**

第 10 章　参考文献 **41**

付録 A　コンパートメントモデル **43**
A.1 コンパートメントモデル 43
A.1.1 コンパートメントモデルの方程式 43
A.1.2 簡易吸収モデル 44

付録 B　UVA/Padova モデルの変数および代謝パラメータ **45**
B.1 変数 45
B.2 代謝パラメータ 46

付録 C　線形モデルのシステム行列 **49**
C.1 線形モデルのシステム行列 49

付録 D　双線形モデルのグルコース吸収速度定数 **51**

D.1 双線形モデルのグルコース吸収速度定数 51

付録 E 基礎代謝基準値 53

E.1 基礎代謝基準値 53

付録 F 糖代謝モデルシミュレータ 55

謝辞 81

図 目 次

2.1 Schematic diagram of UVA/Padova model. 4

3.1 Plasma glucose trajectories of UVA/Padova model and the linear model. . . . 10
3.2 Linearization of the glucose utilization equation U_{id} in UVA/Padova model. . . 11
3.3 Linearization of the gastric emptying coefficient k_{gut} in UVA/Padova model. . . 11

4.1 Bilinear approximation of the glucose utilization equation. 14
4.2 Schematic diagram of the approximation of the glucose absorption equation. . 15

5.1 Switch approximation of the gastric emptying coefficient k_{gut}. 17
5.2 Plasma glucose trajectories of switch apptoximation model. 18
5.3 The time responses of $Q_{sto}, \widehat{Q}_{sto}$ (left figure) and corresponding $k_{gut}, \widehat{k}_{gut}$ (right figure). 19
5.4 Positional relation between Q_b, $b \cdot D$ and D. 22
5.5 The distributions of Q_b and Q_d derived by numeric calculation. 23
5.6 The regressinon of switch points R_b, R_d by metabolic parameters. 23

7.1 Comparisons of plasma glucose trajectories using UVA/Padova model and the approximate models in the normal condition. 32
7.2 Comparisons of plasma glucose trajectories using UVA/Padova model and the approximate models when not administering bolus insulin. 33
7.3 Comparisons of plasma glucose trajectories using UVA/Padova model and the approximate models when administering over insulin. 34

8.1 Variation of error rate with sampling periods when using the bilinear switched model. 36

A.1 Two compartments of the very simplified human body representation. 44

F.1 Shematic diagram of diabetes simulator. 55

表 目 次

8.1 Allowable sampling periods. 37

B.1 State variables of UVA/Padova model. 45
B.2 Output, input and state-dependent variables of UVA/Padvoa model. 46
B.3 Metabolic parameters of UVA/Padova model. 47

D.1 Value of the glucose absorption rate of the bilinear model. 51

E.1 Basal metabolic rate for male. 53

第 1 章

はじめに

1.1 研究背景と目的

私たちの身体にはホメオスタシス（恒常性）が備わっており，生理機能は常に適正状態に維持される．しかし，病気や怪我などにより体内器官を損傷し，健康状態の維持が困難になる場合がある．現在では，生医学，制御工学をはじめとした科学技術の発展に伴い，外部から人間の状態を意図的に制御する治療法が実現されている．その一例として挙げられるのが，血糖値制御である．

糖尿病は，血漿中の糖濃度（血糖値）の調節機能に不具合が生じ，慢性的な高血糖となる病気である．体内血糖値は複数のホルモンによって調節されるが，その中でも重要な役割をもつホルモンがインスリンである．インスリンは膵臓の β 細胞から分泌され，唯一の血糖低下作用を持つホルモンである．糖尿病は，このインスリンによる血糖値低減効果の欠如により発症する [1].

糖尿病は，その成因により 1 型糖尿病と 2 型糖尿病の 2 つに大別される．2 型糖尿病は，遺伝的素因によるインスリン分泌能の低下に加え，運動不足や乱れた食生活などの環境的素因によるインスリン抵抗性の上昇により，インスリンの血糖値低減効果が減少することで発症する．体内インスリン分泌は存在するため，軽度の場合には運動や食生活の改善などによる非インスリン療法が一般的である．しかし，重度の場合はその効果が十分ではないため，インスリンを外部投与するインスリン療法が適用される．

一方，1 型糖尿病は，自己免疫学的機序によりインスリン分泌細胞である β 細胞が破壊され，体内インスリン分泌が停止することで発症する．１型糖尿病の場合，体内インスリンが存在しないためインスリン療法が不可欠であり，投与を怠ると昏睡や死に至る可能性もある [2].

従来からのインスリン投与方法は，患者ごとに医師が定めるグルコース-インスリン比に基づいて患者自身が食事量からインスリン投与量を決定し，注射器あるいはインスリンポンプ[1]を用いて自ら投与するものである．これは，患者にとって手間である上，誤った投与量の設定による事故リスクも懸念される．そのため現在では常に血糖値をモニタリングし，自動的に適量インスリンを投与する人工膵臓の開発が進められている．

人工膵臓の実装には，最適なインスリン投与量を決定する血糖値制御則が必要である．PID 制御を用いた人工膵臓は実用化されているが [3]，現段階では自動制御は基礎インスリン投与に限られ，完全自動化はなされていない．他方，モデルベース制御による人工膵臓も研究段階であり，モデル予測制御やロバスト制御をはじめとした研究が多くなされている [4, 5]．モデルベース制御の場合，多入力，多出力の制御や状態制約を組み込んだ制御の実装が容易である．また，モデル内の状態やパラ

[1]設定した量のインスリンを自動的に皮下投与する医療機器

メータは現実の生理的状態を表しており，より細かな制御設計が可能である．

モデルベース制御の実装には，正確な制御対象のモデルが必要である．糖代謝モデルは複数提案されているが，その 1 つが UVA/Padova モデル [6] である．1 型糖尿病患者の糖代謝動態を模擬した数理シミュレータである UVA/Padova T1DM Simulator [7] は 2008 年に米国の食品医薬品局 (FDA: Food and Drug Adoministration) に臨床試験前の動物実験の代替方法として承認されており，その内部に含まれる UVA/Padova モデルは正確性の高い糖代謝モデルとされている．また，2013 年には，グルカゴンホルモンの効果を追加したモデル [8] が提案されている他，文献 [9] においてはレプチン効果を含んだモデルも示されている．これらのモデルは十分な正確性と引き換えに，高次の状態空間と高い非線形性を含む複雑なモデルである．

血糖値制御では，制御対象の特性変動に対する適応が求められる．血糖値制御は制御対象が生体器官であり，その特性は時間差や個人差により大きく変わるためである．UVA/Padova モデルは，代謝パラメータ値を変更することで複数患者の模擬が可能であり，個人差による特性変動の考慮は可能である．しかし，対象患者にあったパラメータ値を限られた観測可能な出力値（たとえば血糖値）から決定しなければならないほか，時間差による特性変動を組み込むには，適応モデル予測制御のような実時間でモデル修正を行う制御則が必要になる．この場合，実時間で制御入力ならびにモデルパラメータの最適化計算を実行する必要があるが，ここに精緻なモデルをそのまま用いて実装することは計算量の観点から困難であると考えられる．そのため，非線形性が小さく状態空間の次元が低いモデルが必要となる．

本論文では，1 型糖尿病患者を対象としたモデルベース血糖値制御則の実装を念頭におき，既存の糖代謝動態数理モデルの簡略化による糖代謝動態近似モデルの構築と検討を行う．具体的には，UVA/Padova モデルを元に，精度を大きく損なわない範囲でその内部の非線形性の簡略化，状態数の削減を行い，3 つの糖代謝動態近似モデルを構築する．血糖値応答の精度と，離散化モデルの精度の観点から，構築したモデルの検討を行う．

1.2 論文構成

本章である 1 章では，本研究を行うに至った背景と，その背景を踏まえた研究の目的について述べた．2 章では，糖代謝動態近似モデルのベースとなる UVA/Padova モデルについての説明を行う．3 章では UVA/Padova モデルを線形化した線形モデルを示し，これをもとに UVA/Padova モデルの非線形性について説明する．4 章から 6 章では，UVA/Padova モデルを簡略化した糖代謝動態近似モデルを構築する．7 章では構築モデルの近似精度の比較を血糖値のシミュレーション応答を用いて行う．8 章では離散化により生じる誤差の観点から各モデルの比較を行い，9 章でまとめと今後の課題について述べる．

第 2 章

UVA/Padova モデル

2.1 概要

UVA/Padova T1DM Simulator は 1 型糖尿病患者の糖代謝動態を模擬した数理シミュレータであり，2008 年には米国の食品医薬品局に臨床試験前の動物実験の代替方法として承認されている．UVA/Padova モデルは，その内部に含まれる糖代謝動態の数理モデルである．

モデル全体の構成を Fig. 2.1 に示す．コンパートメントモデル[1] としてモデル化され，グルコース量やインスリン量などの物質量を記述するコンパートメント群とコンパートメント間の流量を定義する関数によって構成される．モデルの入力はインスリン投与流量 u_{ins} [pmol/kg/min] と食事摂取流量 d_{meal} [mg/min]，モデルの出力は血漿グルコース濃度（血糖値） G [mg/dl] である[2] ．

食事摂取により血液中に多量のグルコースが溶出し，血糖値の上昇を引き起こす．インスリンと血糖値上昇によって血糖利用が促進されるとともに糖新生が抑制され，定常血糖値へと回復する．本モデルは 1 型糖尿病を対象としたモデルのため，内因性インスリン分泌は含まれていない．

2.2 代謝パラメータ

モデルの状態方程式の説明に移る前に，UVA/Padova モデルの代謝パラメータについて説明する．UVA/Padova モデルには応答特性を決めるいくつかの代謝パラメータが存在している．UVA/Padova T1DM Simulator には，特性の異なる 300 人（大人 100 人，青年 100 人，子供 100 人）分の患者の代謝パラメータ値が組み込まれており，これらは 1 型糖尿病患者全体の特性を網羅している．これらの代謝パラメータ値を用いることで 1 型糖尿病患者全体に対するモデルの妥当性を検証することが可能である．このうち，30 人（大人 10 人，青年 10 人，子供 10 人）分の患者の代謝パラメータ値は文献 [10] において公開されており，本論文ではこのうち大人 10 人分の代謝パラメータ値を用いて検証を行う．付録 Table B.3 に代謝パラメータ値を示すが，ここでは代表患者の値を 1 組のみ示すこととし，他患者の値は先の文献を参照されたい．

2.3 UVA/Padova モデルの状態方程式

UVA/Padova モデルは，12 次元の状態次元を持つモデルである．大きく分けて皮下インスリン動態，インスリン動態，グルコース消化吸収，グルコース動態，糖新生，グルコース代謝利用，グルコース腎排出の 7 ブロックに分けられる．本節ではこれらの詳細を説明する．説明内で出てくる各変

[1] コンパートメントモデルの説明については，付録 A を参照のこと．

[2] 原著論文 [6] では制御時に実際に計測される皮下グルコース濃度 G_M を出力としているが，本論文では血糖値制御の実装は行わないため，血漿グルコース濃度 G を出力としている．

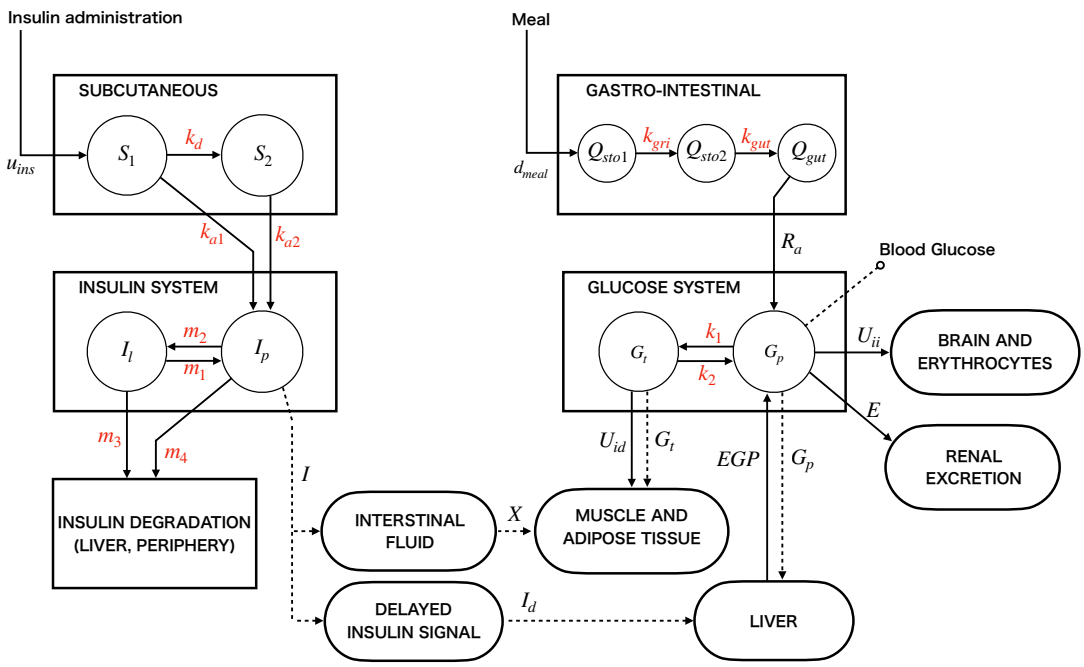


Fig. 2.1: Schematic diagram of UVA/Padova model. The circles represent compartments, the rounded rectangles represent functions that indicate the function of each organ and the red letters are the rate parameters of the compartmental models. The Solid arrows represent flows of substances and the dashed arrows represent control signals.

数，代謝パラメータ値は付録 Table B.1∼B.3 に示す．

2.3.1 皮下インスリン動態

インスリンは通常，皮下組織に複合体インスリンとして投与され，一定の速度定数で血中へと吸収される．複合体インスリンの一部は皮下中で単量体インスリンへと分解されるため，皮下インスリン動態は 2 つのコンパートメントを用いてモデル化され，

$$\dot{S}_1(t) = -(k_{a1} + k_d) \cdot S_1(t) + u(t) \tag{2.1}$$

$$\dot{S}_2(t) = k_d \cdot S_1(t) - k_{a2} \cdot S_2(t) \tag{2.2}$$

として表される．u_{ins} [pmol/kg/min] はインスリン投与流量，S_1, S_2 [pmol/kg] はそれぞれ皮下の複合体インスリン量と単量体インスリン量，k_{a1}, k_{a2} [1/min] はインスリンの血漿への吸収速度定数である．また，k_d [1/min] は複合体インスリンから単量体インスリンへの分解速度定数である．

2.3.2 インスリン動態

皮下に投与されたインスリンは，血液中に吸収され全身に分布する．インスリン動態は肝臓中インスリンと血漿中インスリンの 2 コンパートメントでモデル化され，

$$\dot{I}_l(t) = -(m_1 + m_3) \cdot I_l(t) + m_2 \cdot I_p(t) \tag{2.3}$$

$$\dot{I}_p(t) = -(m_2 + m_4) \cdot I_p(t) + m_1 \cdot I_l(t) + k_{a1} \cdot S_1(t) + k_{a2} \cdot S_2(t) \tag{2.4}$$

として表される．I_l, I_p [pmol/kg] はそれぞれ肝臓中インスリン量と血漿中インスリン量，m_1, m_2 [1/min] はコンパートメント間の移動速度定数，m_3 [1/min] は肝臓でのインスリン抽出速度

定数，m_4 [1/min] は血漿からのインスリン排出速度定数である．肝臓でのインスリン抽出 HE は，インスリン分泌 S を用いて

$$HE(t) = -m_5 \cdot S(t) + m_6, \quad HE_b = -m_5 \cdot S_b + m_6 \tag{2.5}$$

で表され，このとき肝臓でのインスリン抽出速度係数 m_3 は

$$m_3 = \frac{m_1 \cdot HE(t)}{1 - HE(t)} \tag{2.6}$$

として与えられる [11]．1 型糖尿病患者の場合，インスリン分泌は存在せず，$S(t) = 0$ である．しかし，定常状態では基礎インスリンが持続投与されることを仮定しており，ここでは $HE(t)$ は定常値 HE_b をとるとする．すなわち，m_3 は

$$m_3 = \frac{m_1 \cdot HE_b}{1 - HE_b} \tag{2.7}$$

として計算される．

また，血漿中インスリン濃度 I [pmol/l] は，I_p をインスリンの分布容積 V_I [l/kg] で割った

$$I(t) = \frac{I_p(t)}{V_I} \tag{2.8}$$

である．また，基礎インスリン流量 i_b [pmol/kg/min] は血漿中インスリン濃度の定常値 I_b が既知であれば

$$i_b = \left[(m_2 + m_4) - \frac{m_1 \cdot m_2}{m_1 + m_3}\right] \cdot I_b \cdot V_I \tag{2.9}$$

として求まる．

2.3.3 グルコース消化吸収

経口摂取した食事（グルコース）は．胃ですり潰されたのち腸へと運搬され，腸内で吸収される．胃における食事の状態を個体と液体の 2 つに分けて考えると，腸を合わせて 3 つのコンパートメントでモデル化され

$$\dot{Q}_{sto1}(t) = -k_{gri} \cdot Q_{sto1}(t) + d_{meal}(t) \tag{2.10}$$

$$\dot{Q}_{sto2}(t) = -k_{gut}(t) \cdot Q_{sto2}(t) + k_{gri} \cdot Q_{sto1}(t) \tag{2.11}$$

$$\dot{Q}_{gut}(t) = -k_{abs} \cdot Q_{gut}(t) + k_{gut}(t) \cdot Q_{sto2}(t) \tag{2.12}$$

$$R_a(t) = \frac{f \cdot k_{abs} \cdot Q_{gut}(t)}{BW} \tag{2.13}$$

として表される．d_{meal} [mg/min] が食事摂取流量，Q_{sto1}, Q_{sto2} [mg] はそれぞれ胃内個体グルコース量と胃内液状グルコース量，Q_{gut} [mg] が腸内グルコース量であり，R_a [mg/kg/min] は腸からのグルコース吸収を表す．また，k_{gri} [1/min] は胃内の食事すり潰し速度定数，k_{gut} [1/min] は胃内グルコース排出速度係数，k_{abs} [1/min] は血液中へのグルコース吸収速度定数である．実際に腸内から血液中に吸収されるグルコースの量は摂取量より少なく，f [−] はその割合を示す．BW [kg] は患者の体重である．

k_{gut} は tanh 関数によって表される時変パラメータで

$$k_{gut}(Q_{sto}(t)) = k_{min} + \frac{k_{max} - k_{min}}{2} \cdot \{\tanh[\alpha \cdot (Q_{sto}(t) - d \cdot D(t))] - \tanh[\beta \cdot (Q_{sto}(t) - b \cdot D(t))] + 2\} \tag{2.14}$$

として定義される．k_{gut} の非線形性は 3 章にて詳しく説明する．k_{max}, k_{min} [1/min] は胃内グルコース排出速度の上限と下限であり，b, d [−] は k_{gut} の変化特性を定めるパラメータである．ただし，

$$Q_{sto}(t) = Q_{sto1}(t) + Q_{sto2}(t) \tag{2.15}$$

$$\alpha = \frac{5}{2 \cdot D(t) \cdot (1-b)}, \quad \beta = \frac{5}{2 \cdot D(t) \cdot d} \tag{2.16}$$

$$D(t) = Q_{sto}(\bar{t}) + \int_{\bar{t}}^{\bar{t}_f} d_{meal}(\tau) d\tau \tag{2.17}$$

である．ここで，$\bar{t}$，$\bar{t}_f$ [min] はそれぞれ直近の食事摂取の開始時刻と終了時刻であり，D [mg] は直近の食事量である．食事時の D の時間変化について，原著論文 [6] において詳しく定義されていないが，ここでは $\bar{t}_f - \bar{t} \ll 1$ とし, 食事毎に切り換わる定数とする．

2.3.4 グルコース動態

腸から吸収されたグルコースは，主に血液中ならびに骨，筋肉，脂肪組織といった抹消組織中に分布する．グルコース動態は血漿と末梢組織の 2 つのコンパートメントによってモデル化され

$$\dot{G}_p(t) = EGP(t) + R_a(t) - U_{ii}(t) - E(t) - k_1 \cdot G_p(t) + k_2 \cdot G_t(t) \tag{2.18}$$

$$\dot{G}_t(t) = -U_{id}(t) + k_1 \cdot G_p(t) - k_2 \cdot G_t(t) \tag{2.19}$$

として表される．G_p, G_t [mg/kg] はそれぞれ血漿中グルコース量と組織中グルコース量，EGP [mg/kg/min] は糖新生によるグルコース生成流量 E [mg/kg/min] は腎臓におけるグルコースの排出流量を表す．U_{ii} [mg/kg/min] は脳や赤血球におけるグルコース利用流量，U_{id} [mg/kg/min] は抹消組織でのグルコース代謝利用流量を表し，k_1, k_2 [1/min] はコンパートメント間の移動速度定数である．

なお，血漿中グルコース濃度 G [mg/dl] は，G_p をグルコースの分布容積 V_G [dl/kg] で割った

$$G(t) = \frac{G_p(t)}{V_G} \tag{2.20}$$

として求まる．

2.3.5 糖新生

肝臓では，エネルギー源であるグルコースを常に生成しており，これを糖新生（Endogenous glucose production）という．糖新生によるグルコース生成流量 EGP は血漿中グルコース量 G_p とインスリン遅れ信号 I_d [pmol/l] によって決定され

$$EGP(t) = \max[0, k_{p1} - k_{p2} \cdot G_p(t) - k_{p3} \cdot I_d(t)] \tag{2.21}$$

として表される．インスリンの遅れ信号は

$$\dot{I}_1(t) = -k_i \cdot [I_1(t) - I(t)] \tag{2.22}$$

$$\dot{I}_d(t) = -k_i \cdot [I_d(t) - I_1(t)] \tag{2.23}$$

である．ここで，k_{p1} [mg/kg/min] は外挿法によって推定されるグルコースとインスリンが 0 の場合の EGP の値である．また，k_{p2} [1/min], k_{p3} [mg/kg/min per pmol/l] はそれぞれ肝臓へのグルコース効果と肝臓へのインスリン作用の大きさを表すパラメータであり，k_i [1/min] はインスリン信号の遅れを決める速度定数である．

EGP の定常値 EGP_b と血漿中グルコース濃度の定常値 G_b ならびに血漿中インスリン濃度の定常値 I_b が既知であれば，k_{p1} は

$$k_{p1} = EGP_b + k_{p2} \cdot G_b \cdot V_G + k_{p3} \cdot I_b \tag{2.24}$$

として与えられる．

2.3.6 グルコース利用

体内では，脳や赤血球，抹消組織において常にグルコースを利用，吸収している．抹消組織中のグルコース代謝利用 U_{id} は間質液中インスリン濃度と組織中グルコース量に依存しており，次のミカエリス・メンテン式と呼ばれる非線形式で表される．

$$U_{id}(t) = \frac{V_m(t) \cdot G_t(t)}{K_{m0} + G_t(t)} \tag{2.25}$$

U_{id} の非線形性は，3 章にて詳しく説明する．V_m [mg/kg/min] が間質液中のインスリン濃度 X [pmol/l] の 1 次式

$$V_m(t) = V_{m0} + V_{mx} \cdot X(t) \tag{2.26}$$

として定義されており，インスリン感受性の大きさはこの傾き V_{mx} によって決まる．ここで，X は

$$\dot{X}(t) = -p_{2U} \cdot X(t) + p_{2U} \cdot [I(t) - I_b] \tag{2.27}$$

で表される．p_{2U} はグルコース代謝利用へのインスリン作用速度を表す定数である．

また，脳や赤血球によるグルコース利用 U_{ii} [mg/kg/min] は，UVA/Padova モデルでは定数としており，

$$U_{ii}(t) = F_{cns} \tag{2.28}$$

で与えられる．

ここで，EGP の定常値 EGP_b，血漿中グルコース濃度の定常値 G_b が既知であれば，組織中グルコース量の定常値 G_{tb} は

$$G_{tb} = \frac{F_{cns} - EGP_b + k_1 \cdot G_b \cdot V_G}{k_2} \tag{2.29}$$

であり，V_{m0} は

$$V_{m0} = \frac{(EGP_b - F_{cns}) \cdot (K_{m0} + G_{tb})}{G_{tb}} \tag{2.30}$$

として与えられる．

2.3.7 グルコース腎排出

非常に高血糖となった場合には腎臓から尿糖が排出される．グルコース腎排出の式は,

$$E(t) = \begin{cases} k_{e1} \cdot [G_p(t) - k_{e2}] & \text{if } G_p(t) > k_{e2} \\ 0 & \text{if } G_p(t) \leq k_{e2} \end{cases} \tag{2.31}$$

で表される．血糖値が閾値 k_{e2} [mg/kg] を超えることで，腎臓からのグルコース排出が行われる．k_{e1} [1/min] はグルコース排出速度定数である．

第 3 章

線形モデルと UVA/Padova モデルの非線形性

3.1 概要

2 章で示した UVA/Padova モデルは，複数の非線形項を含むモデルである．中でも応答に対する影響が大きい非線形項が，グルコース代謝利用式 U_{id} ((2.25) 式) と胃排出速度係数 k_{gut} ((2.14) 式) である．本章では，UVA/Padova モデルを線形化したモデルを示し，その応答と比較する形で UVA/Padova モデルの非線形性について説明を行う．3.2 節で UVA/Padova モデルを線形化した線形近似モデルについて示し，UVA/Padova モデルとの血糖値応答の比較を行う．3.3 節では 3.2 節の結果をもとに，問題となる UVA/Padova モデルの非線形性について説明する．

3.2 線形モデル

UVA/Padova モデルを平衡点周りで線形化した線形近似モデルを考える．以降は簡単のため，線形モデルと呼ぶ．空腹状態かつ基礎インスリン i_b 投与時の平衡点 x_b で線形化し，平衡状態での出力（血糖値）を G_b とする．また，この平衡点をこれ以降簡単に平衡点と呼ぶ．ただし，胃内グルコース排出速度係数の非線形項 k_{gut} ((2.14) 式) は文献 [6] と同様に $k_{gut} = (k_{max} - k_{min})/2$ と仮定して線形化を行う (Fig. 3.3 参照)．k_{gut} は k_{max} から k_{min} の間で変動するが，単純な線形近似では $k_{gut} = k_{max}$ となる．これにより k_{min} の情報は欠落し，誤差が非常に大きくなってしまうためである．

3.2.1 線形モデルの状態方程式

線形モデルの状態変数 x，状態方程式は

$$\dot{\delta x}(t) = A\delta x(t) + B\delta u_{ins}(t) + M\delta d_{meal}(t) \tag{3.1}$$

$$\delta y(t) = C\delta x(t) \tag{3.2}$$

で与えられる．ただし，

$$\delta x = x - x_b \tag{3.3}$$

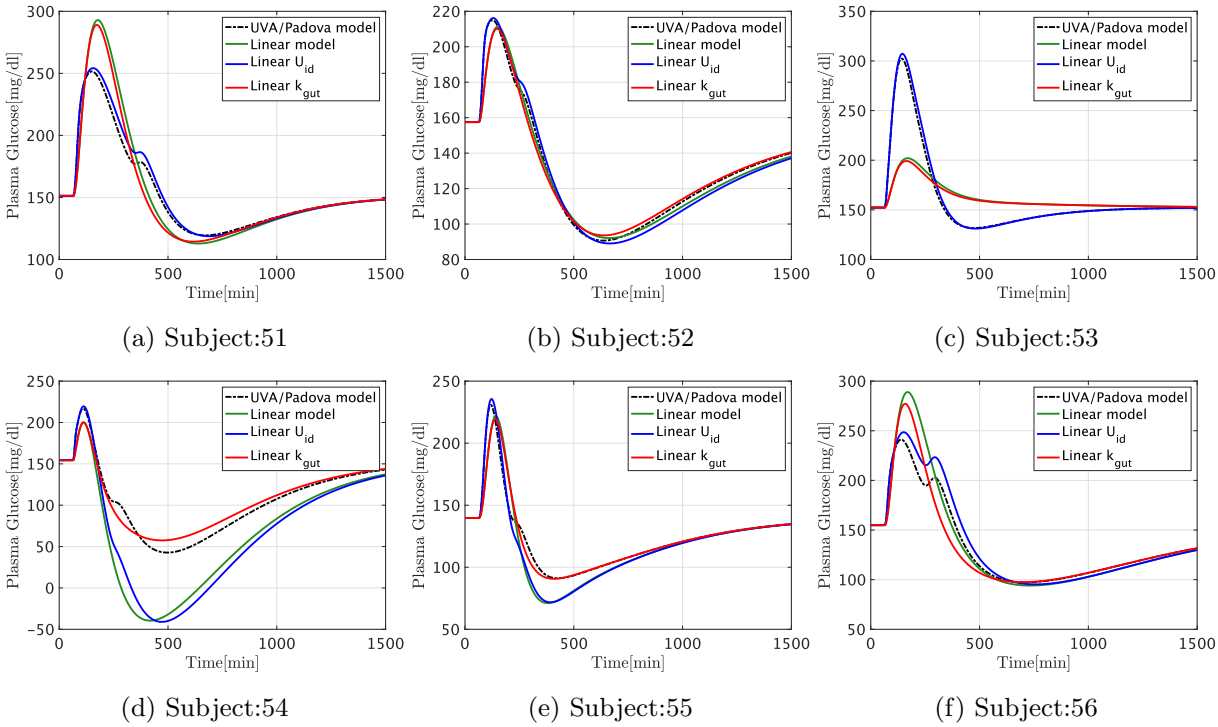
$$\delta y = G - G_b \tag{3.4}$$

$$\delta u = u_{ins} - i_b \tag{3.5}$$

$$\delta d = d_{meal} \tag{3.6}$$

$$x = [\, G_p \; G_t \; Q_{sto1} \; Q_{sto2} \; Q_{gut} \; X \; I_1 \; I_d \; I_l \; I_p \; S_1 \; S_2 \,]^\top \tag{3.7}$$

である．行列 A，B，M，C の詳細は付録 C.1 を参照のこと．



(a) Subject:51 (b) Subject:52 (c) Subject:53

(d) Subject:54 (e) Subject:55 (f) Subject:56

Fig. 3.1: Plasma glucose trajectories of UVA/Padova model (Black chain line) and the linear model (Green line). Also, The blue line represents the trajectory of the model which only U_{id} is linearized in and the red one is the trajectory of the model which only k_{gut} is linearized in. The plasma glucose trajectories are simulated for six different subjects: subject 51 to 56.

3.2.2 線形モデルの血糖値応答

線形モデルの血糖値応答を Fig. 3.1 に示す．緑線が線形モデル，黒の一点鎖線が UVA/Padova モデルの応答である．また，6 枚のパネルはそれぞれ異なる患者の代謝パラメータ値を用いてシミュレーションした結果である．線形モデルの血糖値応答は，患者によっては UVA/Padova モデルと大きな誤差があることがわかる．Subject: 51 や Subject: 53，Subject:56 の応答においては，食事後の血糖値ピークの値が大きく異なるほか，UVA/Padova モデルに見られる双峰性の血糖値動態が線形モデルでは省略されている．また，Subject: 54 や Subject: 55 の応答においては，低血糖時におおきな下振れ誤差が見られる．

これらの誤差の主な原因は，UVA/Padova モデルのグルコース代謝利用式 U_{id} ((2.25) 式) と胃内グルコース排出速度係数 k_{gut} ((2.14) 式) である．Fig. 3.1 の青線に，UVA/Padova モデルの U_{id} のみを線形化したときの血糖値応答を併せて示す．Subject: 54 や Subject: 55 のグラフでは，緑線（線形モデル）と青線はほとんど一致しており，線形モデルの下振れ誤差が U_{id} の線形化によって生じていることが分かる．

同様に，Fig. 3.1 の赤線に，UVA/Padvoa モデルの胃内グルコース排出係数 k_{gut} のみを線形化したときの血糖値応答を併せて示す．Subject: 51 や Subject: 53，Subject: 56 のグラフでは緑線（線形モデル）と，赤線はほとんど一致しており，これらに見られる線形モデルの誤差は k_{gut} の線形化に起因していることが分かる．

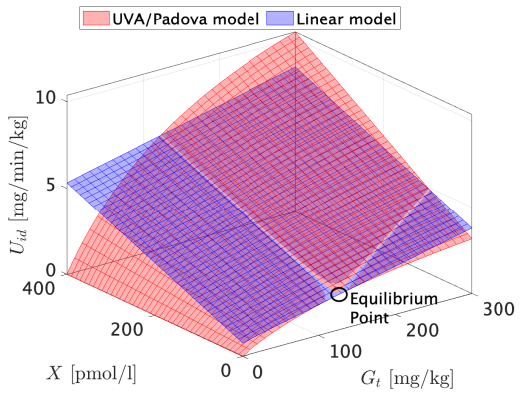


Fig. 3.2: Linearization of the glucose utilization equation U_{id} in UVA/Padova model.

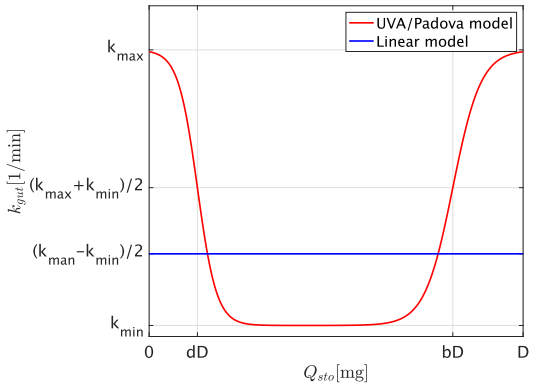


Fig. 3.3: Linearization of the gastric emptying coefficient k_{gut} in UVA/Padova model.

3.3 UVA/Padvoa モデルの主な非線形性

3.3.1 グルコース代謝利用式

グルコース代謝利用 U_{id} ((2.25) 式) は，非線形のミカエリス・メンテン式で表される．X, G_t の変化に対する U_{id} の推移を Fig. 3.2 の赤曲面に示す．一方，グルコース代謝利用式を平衡点で線形化したものが青平面である．そのまま線形化した場合，平衡点以外では誤差が大きい．特に，組織中グルコース量 G_t が低い場合，U_{id} は大きく見積もられることがわかる．U_{id} は，体内でのグルコース代謝利用を表す非線形項であり，血糖値を下げる働きをもつ．平衡点より組織中グルコース量が減る，つまり血糖値が下がると，U_{id} によるグルコース代謝利用が多く見積もられることになり，さらなる血糖値の低下を引き起こす．これにより，Fig. 3.1(d)，3.1(e) に見られるような下振れ誤差が生じる．

3.3.2 胃内グルコース排出速度係数

胃内グルコース排出速度係数 k_{gut} ((2.14) 式) は，胃内液状グルコースの腸への排出速度を定める係数で，2 つの tanh 関数によって表される．状態 Q_{sto} と k_{gut} の関係を図示したものが，Fig. 3.3 である．赤線が UVA/Padova モデルの k_{gut} であり，$k_{gut} = (k_{max} - k_{min})/2$ として線形化したものが青線である．k_{gut} は食事後の血糖値上昇速度を左右するパラメータであり，血糖値のピーク値は k_{gut} の値によって大きく変化する．また，この k_{gut} の非線形性により，Fig. 3.1(a)，3.1(f) に見られるような双峰性の血糖値応答が表現されているが，線形化によって省略される．

線形化時に現れる誤差の原因は主にこの 2 つの非線形性である．糖代謝動態近似モデルの構築には，これら非線形項の精度よい簡略化が必要となる．

第 4 章

双線形モデル

4.1 概要

3 章では，UVA/Padova モデルの非線形性について説明し，グルコース代謝利用式と胃内グルコース排出速度係数の非線形性を精度良く近似することが必要であることを確認した．

4 章 ∼ 6 章では，糖代謝動態近似モデルの構築を行う．本章では，グルコース代謝利用式を双線形の形に近似し，さらにコンパートメントごとに次数の削減を行ったモデルについて説明する．以降では，本モデルを簡単に双線形モデルと呼ぶ．

4.2 節では，グルコース代謝利用式の双線形近似について説明する．4.3 節では，胃内グルコース排出速度係数を含む消化吸収式全体を 1 次遅れの式に簡略化することを考える．4.4 節では，インスリンコンパートメントの省略について説明し，4.5 節ではその他の非線形項の省略について説明した後，4.6 節 で統合した状態方程式を示す．

4.2 グルコース代謝利用式の双線形近似

グルコース代謝利用式 U_{id} ((2.25) 式) は，3.3 節で示した通り，状態量 X, G_t に依存する非線形項である．X, G_t は $0 < X < 400$, $50 < G_t < 300$ 程度の範囲で動作する．線形近似を考えるには動作範囲が大きく，よい近似を得られない．そこで，ここではグルコース代謝利用式を双線形に近似することで，簡単な非線形項に置き換えることを考える．

状態量 G_t が平衡値 G_{tb} のとき，U_{id} の G_t 方向の傾き W は，

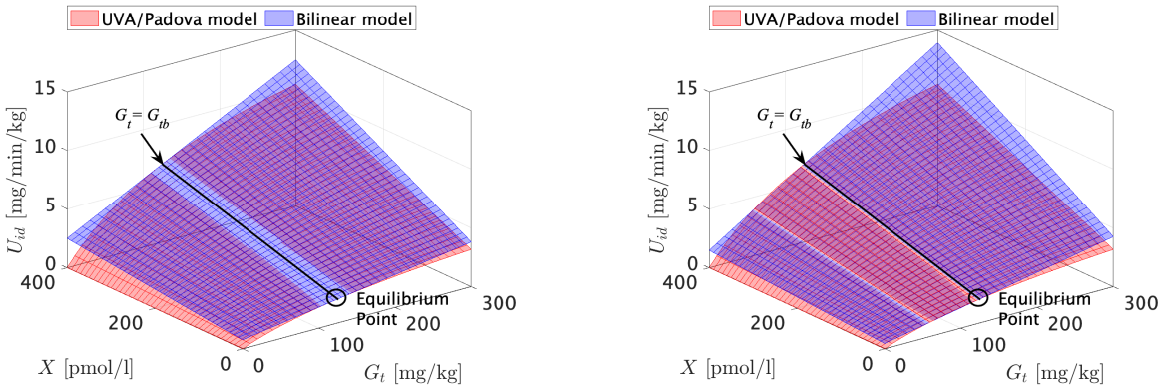
$$\begin{aligned} W &= \frac{\partial U_{id}}{\partial G_t} \\ &= \frac{\partial}{\partial G_t}\left[\frac{(V_{m0} + V_{mx}\cdot X(t))\cdot G_t}{K_{m0} + G_t}\right]\Bigg|_{G_t = G_{tb}} \\ &= \frac{V_{m0}\cdot K_{m0} + V_{mx}\cdot K_{m0}\cdot X(t)}{(K_{m0} + G_{tb})^2} \\ &= W_x \cdot X(t) + W_0 \end{aligned} \tag{4.1}$$

となり，状態量 X の 1 次式として表される．ただし，

$$W_x = \frac{V_{mx}\cdot K_{m0}}{(K_{m0} + G_{tb})^2}, \quad W_0 = \frac{V_{m0}\cdot K_{m0}}{(K_{m0} + G_{tb})^2} \tag{4.2}$$

である．これより，各 X において $G_t = G_{tb}$ で線形近似した場合，全体では双線形近似となり，その近似式は

$$\widehat{U}_{id}(t) = W(X(t))\cdot(G_t(t) - G_{tb}) + U_{idb}(X(t)) \tag{4.3}$$



(a) Bilinear approximation of U_{id} linearized around $G_t = G_{tb}$, given $X \in [0, 400]$, Eq. 4.3.

(b) Improvement of the bilinear approximation of U_{id}, Eq. 4.5.

Fig. 4.1: Bilinear approximation of the glucose utilization equation. The UVA/Padova model is represented by the red surface and its bilinear approximation by the blue surface.

として与えられる．ここで，$U_{idb}(X(t))$ は，直線 $G_t = G_{tb}$ 上の U_{id} の値で，

$$U_{idb}(X(t)) = \frac{V_{m0} \cdot G_{tb} + V_{mx} \cdot G_{tb} \cdot X(t)}{K_{m0} + G_{tb}} \tag{4.4}$$

である．この双線形近似の様子を Fig. 4.1(a) に示す．図から分かるとおり，この近似では $\widehat{U}_{id}$ を U_{id} よりも多く見積もることになる．U_{id} は血糖値を減少させる項である．グルコース濃度が平衡値よりも高い範囲では，近似により $\widehat{U}_{id}$ を多く見積もったとしても平衡点に向かうにつれて誤差は収束していく．一方で，グルコース濃度が平衡値よりも低い範囲では，近似により $\widehat{U}_{id}$ を多く見積もると誤差が広がっていくことになる．そのため，低血糖時の誤差を少なくするように近似することで，全体としての誤差を小さくすることができる．

ここでは，低血糖時の応答を重視するために W を 5/4 倍した値を G_t 方向の傾きとし，

$$\widehat{U}_{id}(t) = \frac{5}{4} \cdot W(X(t)) \cdot (G_t(t) - G_{tb}) + U_{idb}(X(t)) \tag{4.5}$$

を双線形モデルのグルコース代謝利用式とする（Fig. 4.1(b) 参照）．なお，5/4 は大人 10 人分の代謝パラメータを用いたシミュレーションから血糖値応答を比較し，経験的に定めた値である．

4.3 消化吸収式の簡略化

UVA/Padova モデルの胃内グルコース排出速度係数 k_{gut} ((2.14) 式) は 2 つの tanh 関数によって記述される非線形項である．線形システムに置き換える場合，k_{gut} が表現する双峰性の血糖値応答が省略されることはやむを得ない．線形モデルの場合，患者パラメータ全体で同じ線形化を行っていたため，患者によって大きく血糖値応答に誤差が生じていた．ここでは食事吸収式全体で低次元化をしつつ，患者ごとに線形化パラメータを選ぶことで精度を保った簡略化を行う．

胃内固体グルコース，胃内液体グルコース，腸内グルコースの 3 つのコンパートメントから構成される UVA/Padova モデルの消化吸収式 ((2.10)∼(2.17) 式) を，消化管全体を 1 つのコンパートメン

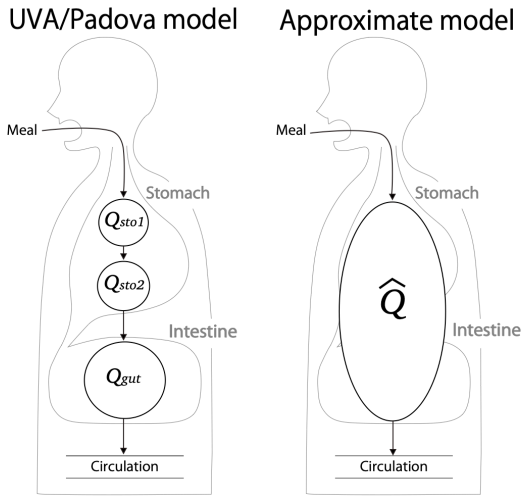


Fig. 4.2: Schematic diagram of the approximation of the glucose absorption equation.

トとして以下の 1 次遅れ系で近似することを考える（Fig. 4.2 参照）.

$$\dot{\widehat{Q}}(t) = -k \cdot \widehat{Q}(t) + d_{meal}(t) \tag{4.6}$$

$$\widehat{R}_a(t) = \frac{f \cdot k \cdot \widehat{Q}(t)}{BW} \tag{4.7}$$

ここで，$\widehat{Q}$ は消化管内全体のグルコース量であり，k [1/min] はグルコースの血漿吸収速度定数である．また，d_{meal}, f, BW は UVA/Padova モデルと同義である．設定パラメータ k は，患者ごとに食事投与シミュレーションを行い，UVA/Padova モデルとの血糖値応答の二乗誤差が最も小さくなる値を採用する．採用したパラメータ値は付録 D.1 節を参照のこと．

4.4 インスリン動態の簡略化

UVA/Padova モデルのインスリン動態は, 肝臓中インスリンと血漿中インスリンの 2 コンパートメントで記述されている ((2.3), (2.4) 式)．システム全体の応答は数時間かけて進行するのに比べ，インスリンコンパートメント間のインスリン分布に要する時間は数分であり，短い．そこでここでは，インスリンコンパートメントを血漿中インスリンコンパートメントのみに簡略化し，状態数の削減を行う．肝臓中インスリンコンパートメントを取り除いたインスリン動態は,

$$\dot{I}_p(t) = -m \cdot I_p(t) + k_{a1} \cdot S_1(t) + k_{a2} \cdot S_2(t) \tag{4.8}$$

として与えられる．速度定数である m は，基礎インスリン i_b を持続投与したとき，平衡状態での血漿中インスリン濃度が UVA/Padova モデルの平衡値 I_b と等しくなるように次式によって定める．

$$m = \frac{i_b}{I_b \cdot V_I \cdot BW} \tag{4.9}$$

なお，各定数は UVA/Padova モデルと同義である．

4.5 その他非線形項の省略

UVA/Padova モデルには，グルコース代謝利用式と胃内グルコース排出速度係数のほかに糖新生 EGP ((2.21) 式)，グルコース腎排出 ((2.31) 式) の非線形項が存在する．しかし，この非線形性の応答に対する影響は小さい．ここでは，EGP の最大値演算子は取り除き，$E = 0$ とする．

4.6 双線形モデルの状態方程式

以下に，双線形モデルの状態方程式を示す．12 次元の状態空間をもつ UVA/Padova モデルと比べ，インスリンコンパートメントと消化吸収式の状態数削減により，双線形近似モデルの状態空間は 9 次元である．また，非線形項はグルコース利用式の双線形項 ((4.5) 式) の 1 つのみである．

皮下インスリン動態

(2.1), (2.2) 式

インスリンシステム

(4.8), (2.8) 式

消化吸収式

(4.6), (4.7) 式

グルコースシステム

$$\dot{G}_p(t) = EGP(t) + \widehat{R}_a(t) - U_{ii}(t) - E(t) - k_1 \cdot G_p(t) + k_2 \cdot G_t(t) \tag{4.10}$$

$$\dot{G}_t(t) = -\widehat{U}_{id}(t) + k_1 \cdot G_p(t) - k_2 \cdot G_t(t) \tag{4.11}$$

(2.20) 式

糖新生

$$EGP(t) = k_{p1} - k_{p2} \cdot G_p(t) - k_{p3} \cdot I_d(t) \tag{4.12}$$

(2.22), (2.23) 式

グルコース利用

(2.27), (2.28), (4.1), (4.2), (4.4), (4.5) 式

グルコース腎排出

$$E(t) = 0 \tag{4.13}$$

第 5 章

線形切り換えモデル

5.1 概要

双線形モデルでは，胃内グルコース排出速度係数 k_{gut} を線形に近似するため，それによって記述される食事後の双峰性の血糖値動態は取り除かれる（あとの Fig. 7.1 参照）．本節では，胃内グルコース排出速度係数 k_{gut} の非線形性を 2 つの線形切り換えシステムとして捉え，非線形特有の動態を保って近似したモデルを考える．以降，この近似モデルを簡単に線形切り換えモデルと呼ぶ．

5.2 消化吸収式の線形切り換えシステム近似

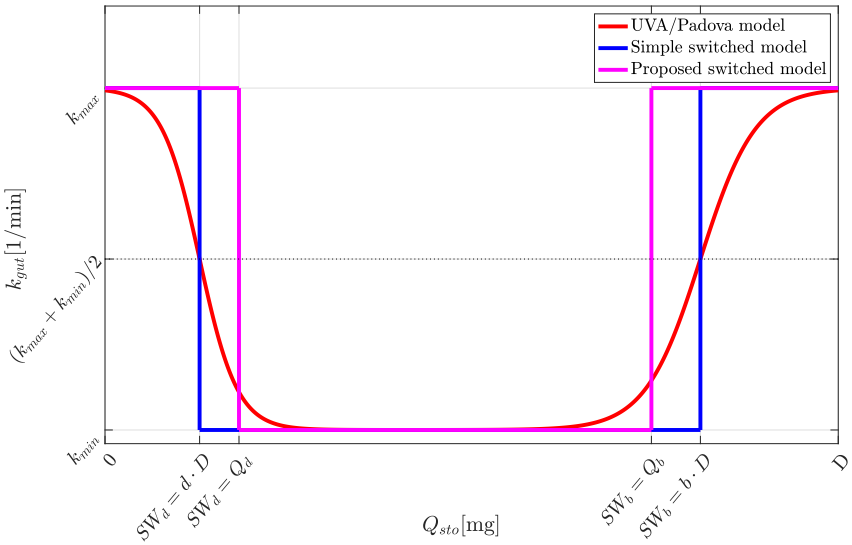


Fig. 5.1: Switch approximation of the gastric emptying coefficient. The red line is k_{gut} in UVA/Padova model , the magenta line is $\widehat{k}_{gut}$ in the model switching at $b \cdot D$ and $d \cdot D$ and the blue line is $\widehat{k}_{gut}$ in the model switching at proposed switch points Q_b and Q_d.

UVA/Padova モデルの胃内グルコース排出速度係数 k_{gut} ((2.14) 式) は，Fig. 5.1 の赤線であらわされる．k_{gut} は胃内グルコース量 Q_{sto} の値によって，最大値 k_{max} と最小値 k_{min} の間を連続的に推移するが，その変化は急峻である．

そこで，ここでは Fig. 5.1 の青線のように，k_{gut} が k_{max}, k_{min} の値のみを取ると考え，k_{gut} を 2

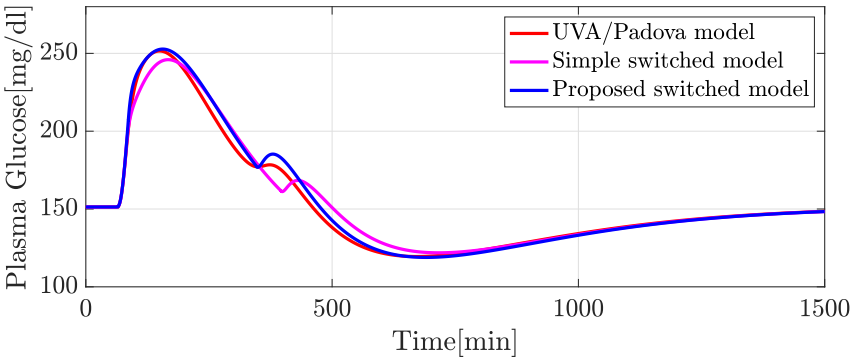


Fig. 5.2: Plasma glucose trajectories of switch apptoximation model. The red line represents UVA/Padova models simulation, the magenta line represents the switch model switching at $d \cdot D$ and $b \cdot D$ and the blue line represents the switch model switching at the proposed switch points.

モードの線形切り換えシステムとして近似する．このとき，消化吸収式は

(2.10) 式

$$\dot{\widehat{Q}}_{sto2}(t) = -\widehat{k}_{gut}(t) \cdot \widehat{Q}_{sto2}(t) + k_{gri} \cdot Q_{sto1}(t) \tag{5.1}$$

$$\dot{\widehat{Q}}_{gut}(t) = -k_{abs} \cdot \widehat{Q}_{gut}(t) + \widehat{k}_{gut}(t) \cdot \widehat{Q}_{sto2}(t) \tag{5.2}$$

$$\widehat{R}_a(t) = \frac{f \cdot k_{abs} \cdot \widehat{Q}_{gut}(t)}{BW} \tag{5.3}$$

$$\widehat{Q}_{sto}(t) = Q_{sto1}(t) + \widehat{Q}_{sto2}(t) \tag{5.4}$$

$$\widehat{k}_{gut} = \begin{cases} k_{max} & \text{if } \widehat{Q}_{sto} < SW_d \text{ or } SW_b \leq \widehat{Q}_{sto} \\ k_{min} & \text{if } SW_d \leq \widehat{Q}_{sto} < SW_b \end{cases} \tag{5.5}$$

として表される．ここで，SW_b, SW_d はシステムの切り換え点である．

(5.2.1), (5.2.2) 節にて，この SW_b, SW_d を求めるための条件を検討する．また，(5.2.3) 節では切り換え点の値決定のための回帰モデルを提案する．

5.2.1 切り換え点の応答に対する影響

k_{gut} 関数 ((2.14) 式) は，2 つの tanh によって記述され最大値 k_{max}，最小値 k_{min} をとる．そこで，その中点 $(k_{max} + k_{min})/2$ を通り，かつ tanh 関数の対象点である $b \cdot D$ および $d \cdot D$ を切り換え点として採用することが考えられる．この切り換え点を採用した線形切り換えモデルの $\widehat{k}_{gut}$ 関数を Fig. 5.1 の桃線に示す．また，このときの血糖値応答を Fig. 5.2 の桃線に示す．UVA/Padova モデルの血糖値応答（赤線）と比べると，血糖値のピーク高や，300[min] 程度に見られる血糖値の揺れ戻しのタイミングに誤差が生じている．近似モデル構築において必要なことは，モデルの内部状態が精度よく近似されることである．つまり，胃内グルコース Q_{sto2} の排出速度を決める k_{gut} ではなく，Q_{sto2} の近似精度を良くする必要がある．ゆえに，切り換え点 SW_b および SW_d は Q_{sto2} と $\widehat{Q}_{sto2}$ の誤差を小さくするように決定するべきである．これを達成する切り換え点を Q_b, Q_d と置く．

5.2.2 切り換え点 Q_b, Q_d の決定

Q_b, Q_d を求めるために，食事後の Q_{sto} の時間応答と k_{gut} の推移をを併せて考える．Fig. 5.3 に，時刻 t_0 で D [mg] の食事を摂取したときの Q_{sto} ならびに $\widehat{Q}_{sto}$ の時間変化（左図）と，それに対応

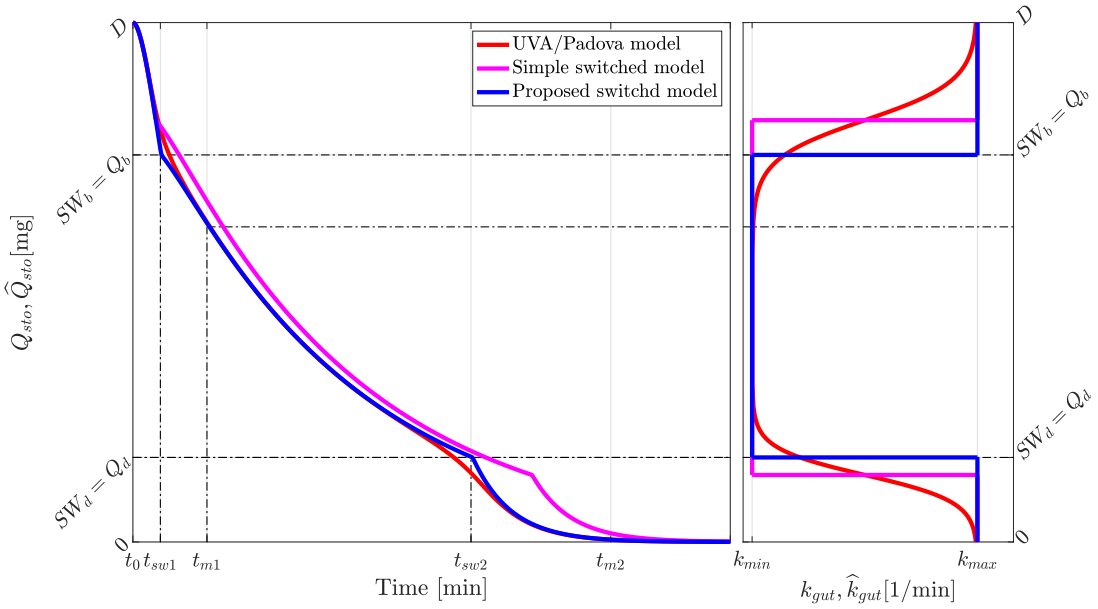


Fig. 5.3: The time responses of $Q_{sto}, \widehat{Q}_{sto}$ and corresponding $k_{gut}, \widehat{k}_{gut}$. t_{sw1} and t_{sw2} is the switching time in the switch model switching at the proposed switch point.

する k_{gut}，$\widehat{k}_{gut}$ （右図）を示す．赤線が UVA/Padova モデルを，桃線が $b \cdot D$ および $d \cdot D$ を切り替え点として採用した場合の線形切り換えモデルを，青線が以降で求める Q_b, Q_d を切り換え点として採用した場合の線形切り換えモデルを示す．

切り換え点を $b \cdot D, d \cdot D$ としたモデルの場合，UVA/Padova モデルと比べて Q_{sto} の大きさが常に異なっている．一方，切り換え点を Q_b, Q_d としたモデルの場合，$t \approx t_{sw1}$ および $t \approx t_{sw2}$ では誤差が存在するが，その他の時刻においては UVA/Padova モデルとの誤差が極めて少ない．その結果，Fig. 5.2 に示すように，血糖値応答をよりよく近似できる．

ここで，Q_{sto1} を決定する式 ((2.10) 式) は UVA/Padova モデルと線形切り換えモデルにおいて同じであり，Q_{sto1} は両モデルで等しい．ゆえに，誤差 $Q_{sto2} - \widehat{Q}_{sto2}$ と誤差 $Q_{sto} - \widehat{Q}_{sto}$ は同じであることに注意されたい．

まず，UVA/Padova モデルの Q_{sto}, k_{gut} の推移を考える．食事直後は，$Q_{sto} = D$ であり[1]，k_{gut} は k_{max} の値をとる．時間経過により Q_{sto} が減少すると，これに伴い k_{gut} の値も減少し，最小値 k_{min} に近づく．その後，はじめて $k_{gut}(t_{m1}) \approx k_{min}$ となる時刻を $t_{m1} > t_0$ とする．それ以降はしばらく $k_{gut} \approx k_{min}$ のまま推移し，さらに Q_{sto} が減少すると k_{gut} が再び増加をはじめる．この時刻を t_l とする．その後，Q_{sto} は 0 へと漸近し，このとき $k_{gut} \approx k_{max}$ となる．この時刻を t_{m2} とする．

次に，線形切り換えシステムの $\widehat{Q}_{sto}, \widehat{k}_{gut}$ の推移を考える．食事直後は $\widehat{Q}_{sto} = D$ であり，食事後しばらく $\widehat{k}_{gut}$ は k_{max} の値をとる．その後，時間経過により $\widehat{Q}_{sto}$ が減少し，$\widehat{Q}_{sto} = SW_b$ となった時点で $\widehat{k}_{gut}$ は k_{min} の値に切り換わる．この時刻を t_{sw1} とする．$\widehat{Q}_{sto}$ がさらに減少し，$\widehat{Q}_{sto} = SW_d$ となった時点で再び $\widehat{k}_{gut}$ は k_{max} の値に切り換わる．この時刻を t_{sw2} とする．その後，Q_{sto} は 0 へと漸近する．

[1] 食事は十分短時間のうちに入力されると仮定する．

時刻 $t_{m1} < t < t_l$ では $k_{gut} \approx k_{min}$ である．このとき Q_{sto2} の値は，(2.14) 式より

$$\dot{Q}_{sto2}(t) = -k_{min} \cdot Q_{sto2}(t) + k_{gri} \cdot Q_{sto1}(t) \tag{5.6}$$

によって決定される．また，時刻 $t_{sw1} < t < t_{sw2}$ では $\widehat{k}_{gut} = k_{min}$ である．このとき $\widehat{Q}_{sto2}$ の値は，(5.5) 式より

$$\dot{\widehat{Q}}_{sto2}(t) = -k_{min} \cdot \widehat{Q}_{sto2}(t) + k_{gri} \cdot Q_{sto1}(t) \tag{5.7}$$

によって決定される．ここで，(5.6) 式と (5.7) 式は同一の式である．つまり，$t_{sw1} < t_{m1}, t_l < t_{sw2}$ と仮定すれば，時刻 $t_{m1} < t < t_l$ では Q_{sto2} と $\widehat{Q}_{sto2}$ は同一の式により決定される．これより，$Q_{sto2}(t_{m1}) \approx \widehat{Q}_{sto2}(t_{m1})$ であれば，時刻 $t_{m_1} < t < t_l$ において $Q_{sto2}(t) \approx \widehat{Q}_{sto2}(t)$ が成立し，UVA/Padova モデルと線形切り換えモデルの誤差を十分に小さくすることができる．これをみたす切り換え点を $SW_b = Q_b$ として採用する．

$Q_{sto2}(t_{m1}) \approx \widehat{Q}_{sto2}(t_{m1})$ となるとき，t_{sw1} がみたすべき条件は

$$Q_{sto}(t_{m1}) = D - \left[\int_{t_0}^{t_{sw1}} k_{max} \cdot \widehat{Q}_{sto2}(t)dt + \int_{t_{sw1}}^{t_{m1}} k_{min} \cdot \widehat{Q}_{sto2}(t)dt\right] \tag{5.8}$$

である．(5.8) 式 をみたす t_{sw1} が決まれば

$$Q_b = D - \int_{t_0}^{t_{sw1}} k_{max} \cdot \widehat{Q}_{sto2}(t)dt \tag{5.9}$$

より，切り換え点の値 $SW_b = Q_b$ を定めることができる．

もう一方の切り換え点 $SW_d = Q_d$ についても同様に，k_{gut} の値が最大値 k_{max} に近づく時刻 t_{m2} を考え，t_{m2} 以降の時刻 $t > t_{m2}$ で $Q_{sto2}(t) \approx \widehat{Q}_{sto2}(t)$ が成立するように Q_d の値を定める．このとき t_{sw2} がみたすべき条件は

$$Q_{sto}(t_{m2}) = Q_{sto}(t_{m1}) - \left[\int_{t_{m1}}^{t_{sw2}} k_{max} \cdot \widehat{Q}_{sto2}(t)dt + \int_{t_{sw2}}^{t_{m2}} k_{min} \cdot \widehat{Q}_{sto2}(t)dt\right] \tag{5.10}$$

であり，t_{sw2} が決まれば

$$Q_d = Q_{sto}(t_{m1}) - \int_{t_{m1}}^{t_{sw2}} k_{min} \cdot \widehat{Q}_{sto2}(t)dt \tag{5.11}$$

により切り換え点の値 $SW_d = Q_d$ を定めることができる．

5.2.3 Q_b, Q_d の回帰モデル

切り換え点 Q_b, Q_d は，各患者の代謝パラメータから計算できると好ましい．しかし，(5.8) 式，(5.10) 式 をみたす切り換え時刻 t_{sw1}, t_{sw2} を解析的に求めることは難しいと考えられる．ここでは，数値的に計算した切り換え時刻 t_{sw1}, t_{sw2} および Q_b, Q_d のデータを用いて，関連する代謝パラメータおよび食事量を説明変数とする切り換え点 Q_b, Q_d の非線形回帰を行う．Q_b, Q_d の決定に関係する代謝パラメータは，Q_{sto2} の値に影響を与える代謝パラメータであり，$k_{gri}, k_{max}, k_{min}, b, d$ の 4 つである．ここに食事量 D を加えた 5 つの定数[2]が，回帰モデルの説明変数となる．k_{gut} は 2 つの

[2] D は食事毎に切り換わる定数である．

tanh が組み合わさっている関数であり，以下簡単のために，

$$k_{gut}(Q_{sto}) = k_{min} + \frac{k_{max} - k_{min}}{2}\{\tanh[\alpha \cdot (Q_{sto} - b \cdot D(t))] + 1\} \tag{5.12}$$

$$k_{gut}(Q_{sto}) = k_{min} + \frac{k_{max} - k_{min}}{2}\{-\tanh[\beta \cdot (Q_{sto} - d \cdot D(t))] + 1\} \tag{5.13}$$

の 2 つに分けて考える．

1. 数値シミュレーションによる Q_b,Q_d の分布の取得

((5.12) 式) を適用した UVA/Padova モデルを用いて時刻 0 に D [mg] の食事を与えたシミュレーションを行い，$k_{gut} \approx k_{min}$ となる時刻 t_{m1} を数値計算により導出する．次に，切り換え時刻 t_{sw1} を変更しつつ，線形切り換えモデルを用いて同じシミュレーションを繰り返し行い，$Q_{sto2}(t_{m1}) \approx \widehat{Q}_{sto2}(t_{m1})$ となるような切り替え時刻 t_{sw1} を探索する．得られた t_{sw1} から，(5.9) 式により切り換え点 Q_b を導出する．説明変数 $k_{gri}, k_{max}, k_{min}, b, d, D$ の値は考えうる範囲を網羅的に設定し，各パラメータ値に対する Q_b の分布を求めた．

同様に，((5.13) 式) を適用した UVA/Padova モデルを用いて t_{m2} を求め，$Q_{sto2}(t_{m2}) \approx \widehat{Q}_{sto2}(t_{m2})$ となるような切り替え時刻 t_{sw2} を探索する．得られた t_{sw2} から，(5.11) 式により切り換え点 Q_d を導出する．説明変数の値は考えうる範囲を網羅的に設定し，各パラメータ値に対する Q_d の分布を求めた．

2. 目的変数 R_b, R_d の導入

代謝パラメータ $k_{gri}, k_{max}, k_{min}, b, d$ および食事量 D を説明変数とし，Q_b, Q_d の値を与える回帰モデルの導出を単純化するため，回帰関数の目的変数として新しい変数 R_d, R_b を導入する．まず，tanh 関数の広がりをあらわす数を定義する．(5.12) 式の場合，tanh 関数の Q_{sto} 方向の縮尺は変数 Q_{sto} の係数，すなわち α の大きさによって決まる．α は $D \cdot (1-b)$ の関数であるから ((2.16) 式)，Q_{sto} 方向の縮尺は，$D \cdot (1-b)$ によって決まることになる．ここで，(5.12) 式に $Q_{sto} = D \cdot (1-b) + b \cdot D$, $Q_{sto} = -D \cdot (1-b) + b \cdot D$ を代入すると，

$$\begin{aligned} k_{gut}(D \cdot (1-b) + b \cdot D) &= k_{min} + \frac{\tanh(5/2) + 1}{2} \cdot (k_{max} - k_{min}) \\ &\approx k_{min} + \frac{1.98}{2} \cdot (k_{max} - k_{min}) \\ &\approx k_{max} \end{aligned} \tag{5.14}$$

$$\begin{aligned} k_{gut}(-D \cdot (1-b) + b \cdot D) &= k_{min} + \frac{-\tanh(5/2) + 1}{2} \cdot (k_{max} - k_{min}) \\ &\approx k_{min} + \frac{-0.98}{2} \cdot (k_{max} - k_{min}) \\ &\approx k_{min} \end{aligned} \tag{5.15}$$

となり，$D \cdot (1-b)$ は tanh 関数の対象点 $b \cdot D$ からおおよそ $k_{gut} \approx k_{max}$ または $k_{gut} \approx k_{min}$ となるまでの距離として考えることができる．以上の理由から，(5.12) 式の tanh 関数の広がりを表す数として，$w_r = D \cdot (1-b)$ を定義する．(5.13) 式の場合も同様に，tanh 関数の広がりを表す数として $w_l = d \cdot D$ を定義する．

次に，tanh 関数の対象点から切り換え点 Q_b, Q_d の距離 l_b, l_d を考える．切り換え点 Q_b の場合は，$b \cdot D$ からの距離とし，

$$l_b = |Q_b - b \cdot D| \tag{5.16}$$

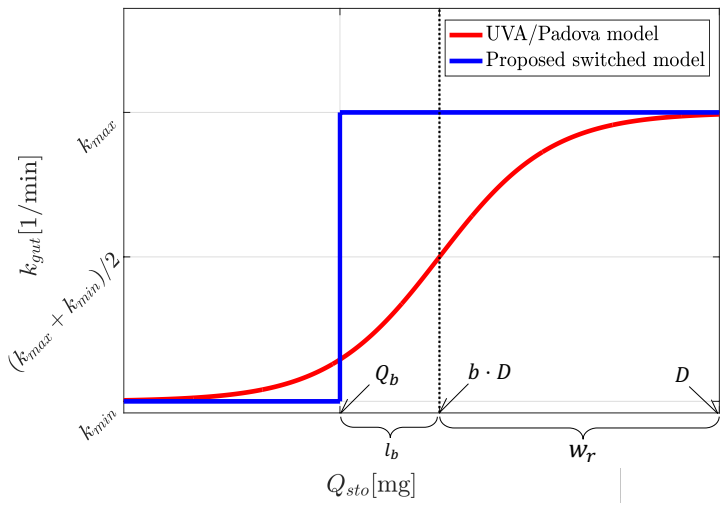


Fig. 5.4: Positional relation between Q_b, $b \cdot D$ and D.

として定義する (Fig. 5.4 参照)．同様に，切り換え点 Q_d の場合は $d \cdot D$ からの距離とし，

$$l_d = |Q_d - d \cdot D| \tag{5.17}$$

として定義する．これら w_l, w_r, l_b, l_d を用いて

$$R_b = \frac{l_b}{w_r} = \frac{|Q_b - b \cdot D|}{D \cdot (1-b)} \tag{5.18}$$

$$R_d = \frac{l_d}{w_l} = \frac{|Q_d - d \cdot D|}{d \cdot D} \tag{5.19}$$

として R_b, R_d を定める．

3．R_b, R_d の非線形回帰

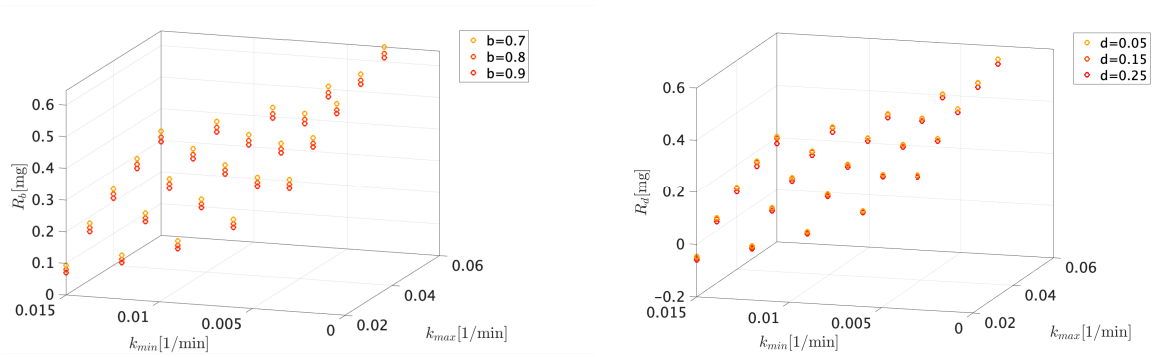
説明変数 $k_{gri}, k_{max}, k_{min}, b, d, D$ による R_b, R_d の非線形回帰を行う．代謝パラメータ k_{gri}, D の変化に対し，R_b, R_d の値はほとんど変化しない結果となった．そのため，ここでは D と k_{gri} に対する R_b, R_d の依存性は無視する．また，Q_b を決定する際には (5.12) 式を用いており，d は現れない．同様に Q_d は b に依らない．ここでは，R_b の回帰についてはパラメータ k_{min}, k_{max}, b のみを，R_d の回帰についてはパラメータ k_{min}, k_{max}, d のみを説明変数とする．

各説明変数の変化に対する R_b, R_d の分布を Fig. 5.5 に示す．R_b, R_d は，k_{max}, k_{min} の変化に対して緩やかな曲線を描いており，指数関数での近似を考える．b, d による R_b, R_d の変化率は k_{min}, k_{max} に依存しており，b, d の変化に対する R_b, R_d の変化率も一定ではない．しかし，R_b, R_d の変化率は b, d の変化に対して大きくない．ここでは R_b は b に対して，R_d は d に対して線形に変化すると考える．以上より，以下の回帰モデルにより R_b, R_d の値を与えることとする．

$$R_b = \beta_{b1} \cdot \exp(\beta_{b2} \cdot k_{max}) + \beta_{b3} \cdot \exp(\beta_{b4} \cdot k_{min}) + \beta_{b5} \cdot b + \beta_{b6} \tag{5.20}$$

$$R_d = \beta_{d1} \cdot \exp(\beta_{d2} \cdot k_{max}) + \beta_{b3} \cdot \exp(\beta_{d4} \cdot k_{min}) + \beta_{d5} \cdot d + \beta_{d6} \tag{5.21}$$

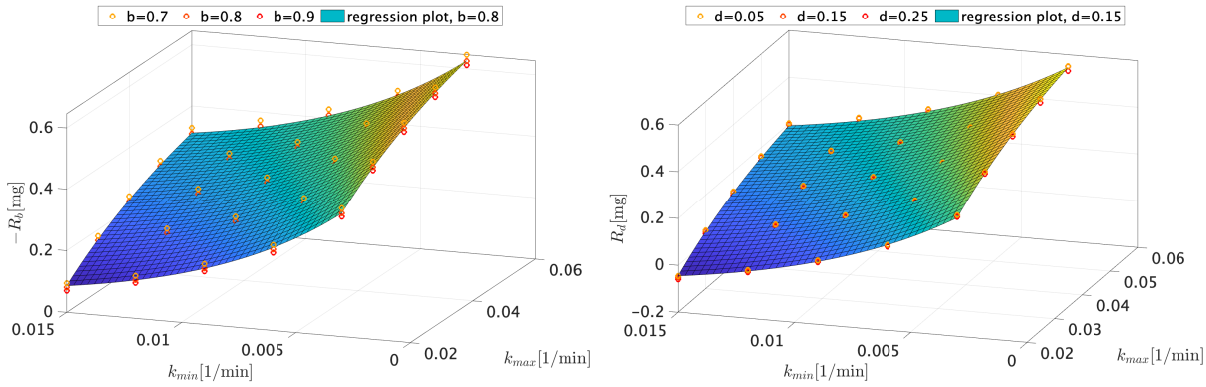
決定パラメータは $\beta_{bi}, \beta_{di}, i = 1, \cdots, 6$ であり，これらパラメータを CasADi/IPOPT による非線形



(a) The distribution of R_b that relative to the changes of k_{max}, k_{min}, b.

(b) The distribution of R_d that relative to the changes of k_{max}, k_{min}, d.

Fig. 5.5: The distributions of Q_b and Q_d derived by numeric calculation.



(a) The regression of R_b.

(b) The regression of R_d.

Fig. 5.6: The regressinon of switch points R_b, R_d by metabolic parameters. Plot the surface representing the regression model over the numeric distribution.

最適化で求めた．求まったパラメータは以下である．

$$\beta_{b1} = -0.588, \quad \beta_{b2} = -27.4, \quad \beta_{b3} = 0.587, \quad \beta_{b4} = -138, \quad \beta_{b5} = -0.147 \quad \beta_{b6} = 0.496$$
$$\beta_{d1} = -0.622, \quad \beta_{b2} = -30.1, \quad \beta_{b3} = 0.657, \quad \beta_{b4} = -125, \quad \beta_{b5} = -0.0656 \quad \beta_{b6} = 0.205$$

Fig. 5.5 に示した R_b, R_d の値と回帰モデルの値を併せて Fig. 5.6 に示す．

4. 切り換え点 Q_b, Q_d の回帰モデル

以上により，患者の代謝パラメータが与えられれば，(5.20),(5.21) 式により R_b, R_d を決定し，

$$Q_b = b \cdot D - R_b \cdot (1 - b) \tag{5.22}$$
$$Q_d = d \cdot D + R_d \cdot d \cdot D \tag{5.23}$$

により切り換え点の値 $SW_b = Q_b$，$SW_d = Q_d$ を求めることができる．

5.3 線形切り換えモデルの状態方程式

以下に，線形切り換えモデルの状態方程式を示す．UVA/Padova モデルの k_{gut} を 2 モードの線形切り換えシステムとして置き換える．また，グルコース代謝利用式，糖新生，グルコース腎排出の非

線形項は，線形モデルと同様の平衡点で線形化する．

皮下インスリン動態

(2.1), (2.2) 式

インスリンシステム

(2.3), (2.4), (2.8) 式

消化吸収式

(2.10), (5.1), (5.2), (5.3), (5.4), (5.5) 式

グルコースシステム

(4.10), (4.11) 式

糖新生

(4.12), (2.22), (2.23) 式

グルコース利用

$$\widehat{U}_{id}(t) = h_1 \cdot (G_t(t) - G_{tb}) + h_2 \cdot (X(t) - X_b) + \bar{U}_{idb} \tag{5.24}$$

$$h_1 = \left.\frac{\partial U_{id}}{\partial G_t}\right|_{G_t=G_{tb}, X=X_b}$$

$$h_2 = \left.\frac{\partial U_{id}}{\partial X}\right|_{G_t=G_{tb}, X=X_b}$$

$$\bar{U}_{idb} = U_{id}(G_{tb}, X_b)$$

(2.27) 式

(2.28) 式

グルコース腎排出

(4.13) 式

第 6 章

双線形切り換えモデル

6.1 概要

5 章で説明した線形切り換えモデルでは，胃内グルコース排出速度係数を線形切り換えシステムとして近似することで，非線形特有の複雑な動態を再現することができる．しかし，グルコース代謝利用式は線形近似しており，それにより低血糖時の応答は精度良く近似できない（Fig. 7.1 参照）．一方，双線形モデルでは，胃内グルコース排出速度係数は線形に近似していたため非線形特有の複雑な動態を模擬することはできないが，グルコース代謝利用式を双線形に近似しており，低血糖時の応答の精度良く近似することができる．

本章では，双線形切り換えモデルと線形切り換えモデルの近似を組み合わせた双線形切り換えモデルを説明する．

6.2 双線形切り換えモデルの状態方程式

グルコース代謝利用式においては，4.2 節の双線形近似を採用する．消化吸収式においては，5.2 節の線形切り換え近似を採用する．さらに，インスリン動態式においては，4.4 節のインスリンコンパートメントの省略を組み込む．また，その他の非線形項は 4.5 節と同様に扱うこととする．以下に，双線形切り換えモデルの状態方程式を示す．

皮下インスリン動態

(2.1), (2.2) 式

インスリンシステム

(4.8), (2.8) 式

消化吸収式

(2.10), (5.1), (5.2), (5.3), (5.4), (5.5) 式

グルコースシステム

(4.10), (4.11), (2.20) 式

糖新生

(4.12), (2.22), (2.23) 式

グルコース利用

(2.27), (2.28), (4.1), (4.2), (4.4), (4.5) 式

グルコース腎排出

(4.13) 式

第 7 章

構築モデルのシミュレーション

7.1 概要

本章では，4 ～ 6 章で構築した簡略化糖代謝動態モデルを用いて血糖値シミュレーションを行い，各モデルの近似精度の比較を行う．7.2 節では，標準的な食事とインスリン投与を行ったときの数値シミュレーションを行う．7.3 節では，事故的な状況におけるシミュレーションとして，食事のみを摂取した場合と，過剰インスリンを投与した場合のシミュレーションを行う．

7.2 標準的な入力に対する血糖値応答

7.2.1 標準入力量の設定

標準的な入力値を定めるために，まず標準的な 1 日のグルコース摂取量を定める．ここでは，厚生労働省の食品摂取基準 [12] をもとに 1 日のグルコース摂取量を定めることとする．

1 日の推定エネルギー必要量 E_{day} [kcal/day] は

$$E_{day} = BM \times BW \times L_{active} \tag{7.1}$$

で与えられる．ここで，BM [kcal/kg/day] は基礎代謝基準値であり，BW [kg] は体重，L_{active} [−] は身体活動レベルを表す．基礎代謝基準値は，食品摂取基準に各性別，年齢別の分布が記載されている．UVA/Padova モデルの代謝パラメータには，患者の性別属性は含まれていないため，ここでは基礎代謝基準値は男性のものを用いる．値は付録 Table E.1 を参照のこと．身体活動レベルは “ふつう” とし，値として $L_{active} = 1.75$ を用いる．以上により，推定エネルギー必要量が求まる．

炭水化物の摂取基準は，エネルギー必要量の 50∼65% と定められている．ここでは，糖尿病患者という背景を鑑み，エネルギー必要量の 50% を炭水化物から摂取することとする．炭水化物内のエネルギー源はほとんど糖質が占めるため，エネルギー必要量の 50% は糖質から摂取されるとしてよい．糖質の持つ熱量は，4 [kcal/g] であるから，1 日の標準糖質摂取量 G_{day} は

$$G_{day} = \frac{1}{8} E_{day} \tag{7.2}$$

となる．ここで，ほとんどの糖質は消化管内でグルコースへと分解されるため，1 日の標準糖質摂取量をそのまま 1 日の標準グルコース摂取量として用いることとする．そして，1 日の標準グルコース摂取量を

$$\textbf{朝食時:昼食時:夕食時 (グルコース摂取量)} = 3 : 3 : 4 \tag{7.3}$$

として分け，1 食あたりの標準グルコース摂取量を定める．

インスリン投与は基礎インスリンと，追加インスリンに分けられる．基礎インスリンは定常的に投与されるインスリンである．一方，追加インスリンは食事毎に追加投与されるインスリンである．追加インスリン量は，患者ごとに設定されるグルコース-インスリン比（$CHOr$ [g/U]）に基づいて決定する．これは，1 単位のインスリンで食事後の血糖値を抑制することのできるグルコース摂取量を表す値である．$CHOr$ を用いると，追加インスリン投与量 i_{meal} [pmol] は食事量 D を用いて

$$i_{meal} = \frac{k_c \cdot D}{CHOr} \tag{7.4}$$

として表される．ただし，k_c [pmol/U] はインスリン単位 [U] からモル単位 [pmol] への変換係数である[1]．

7.2.2 数値シミュレーション

Fig. 7.1 に, 標準的な入力を行った場合の血糖値応答を示す．各パネルでは, 異なる患者の代謝パラメータ (Subject: 51∼60) を用いてシミュレーションを行なっている．黒の一点鎖線が UVA/Padvoa モデル，緑線が UVA/Padova モデルを単純に線形化したモデル，赤線が双線形モデル，桃線が線形切り換えモデル，青線が双線形切り換えモデルを表す．次節の Fig. 7.2, 7.3 においても同様である．

ここで，Subject: 52, 54, 56, 58 においては，(7.4) 式で求める追加インスリンをそのまま適用すると明らかな低血糖となるため，これらの患者に関してはインスリン量を手動で修正した．

すべての患者において，線形モデルに比べて近似モデルの方が優れた近似精度を示している．線形切り換えモデルは，血糖値の双峰性の応答を精度よく近似することができている．一方で，グルコース代謝利用式を線形の形で近似したため，Subject:55 や 59 の応答では低血糖の範囲で大きく誤差が生じている．双線形モデルは，血糖値の双峰性の応答を模擬することはできないが，大きく精度を損なう範囲は少ない．双線形切り換えモデルは，胃内グルコース排出速度係数の切り換え近似とグルコース代謝利用式の双線形近似により，最も優れた近似が可能である．

7.3 事故的な入力に対する血糖値応答

7.3.1 追加インスリン投与を伴わない食事摂取

Fig. 7.2 に追加インスリンを伴わず食事摂取を行った場合の血糖値応答を示す．線形モデルは，患者によって大きく誤差が存在するが，近似モデルはいずれも概ね UVA/Padova モデルと一致している．グルコース代謝利用式の双線形近似は，(4.3) 式によって低血糖時の応答を重視した形にしていた．そのため，高血糖時の振る舞いにおいてはグルコース代謝利用式の双線形近似を含む双線形モデル，双線形切り換えモデルに比べて線形切り換えモデルの方が精度よく近似している．

また，双線形モデルは血糖値の双峰性の応答を模擬できないため，他 2 つの近似モデルに比べて血糖ピーク値付近の応答に誤差が生じている．

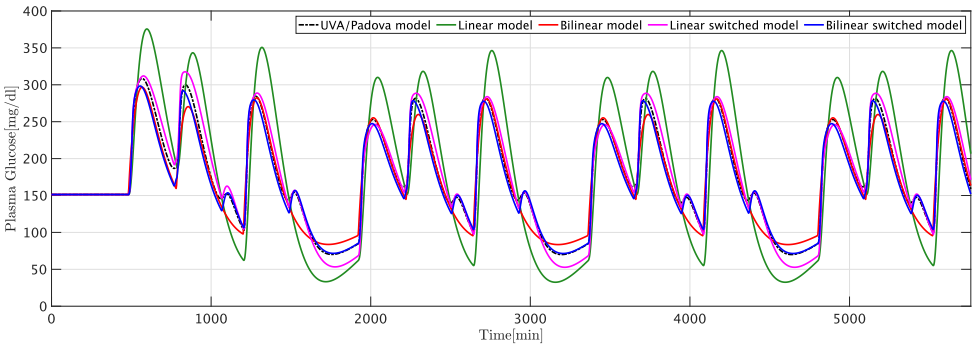
7.3.2 過剰ボーラスインスリン投与

Fig. 7.3 に食事時に過剰インスリンを投与した場合の血糖値応答を示す．線形モデル，線形切り換えモデルではグルコース代謝利用式を線形に近似したため，低血糖時の誤差が大きい．Subject: 53 を除く全ての患者において大きな下振れ誤差がみられる．

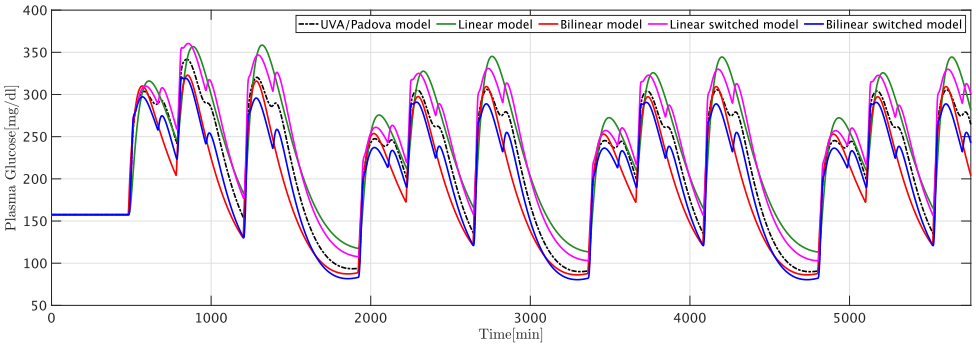
一方，双線形モデル，双線形切り換えモデルでは，グルコース代謝利用式を双線形に近似したことで低血糖時の誤差が大きく改善されている．Subject: 52，54, 58 など血糖値が 0 を下回るような過

[1] インスリン単位 [U] から モル単位 [mol] への変換係数は，$k_c = 6000$ [pmol/U] とする [13].

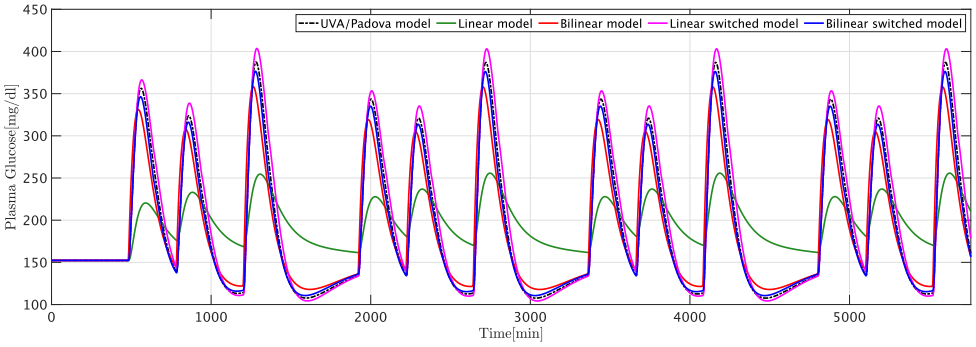
度の低血糖状態においては誤差が見られるが，実用上このような血糖値をとることは考えられない．血糖値が 50mg 以下となると，意識障害や昏睡症状が現れ，回復に第三者の介助が必要になる．この血糖値は重症低血糖と定義され，速やかなグルコース投与が必要である [14]．そのため，50 [mg/dl] 以下の血糖値状態の処置は手動または一定の対処用プロトコルに沿って行われるべきであり，自動制御は血糖値 50 [mg/dl] 以上の範囲で使用されると考えられる．双線形モデル，双線形切り換えモデルは，血糖値が 50 [mg/dl] 以上の範囲においては優れた精度で近似できており，十分有用であると言える．



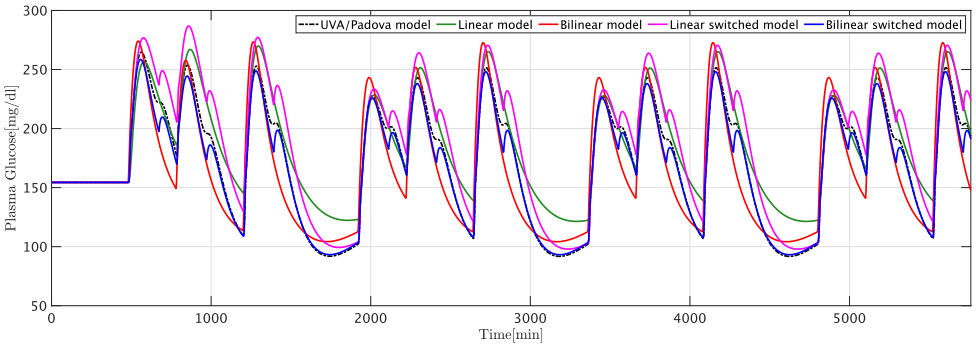
(a) Subject: 51



(b) Subject: 52

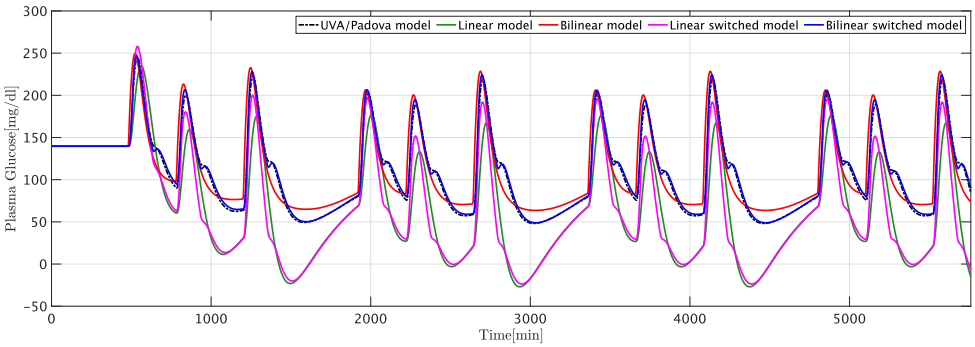


(c) Subject: 53

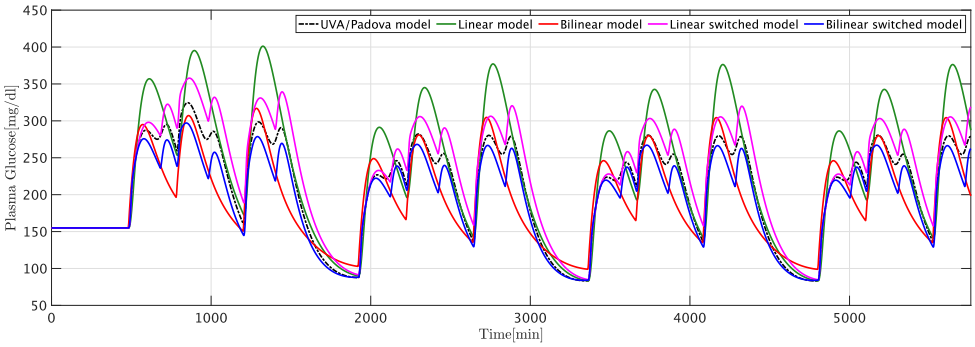


(d) Subject: 54

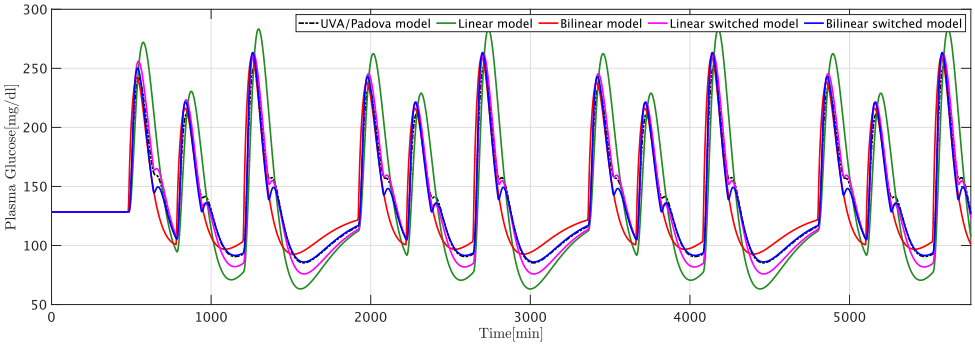
Fig. 7.1: Comparisons of UVA/Padova model and the approximate models.



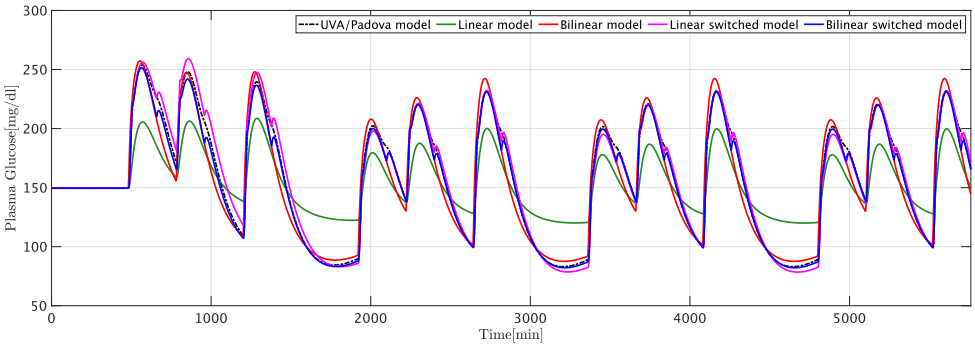
(e) Subject: 55



(f) Subject: 56

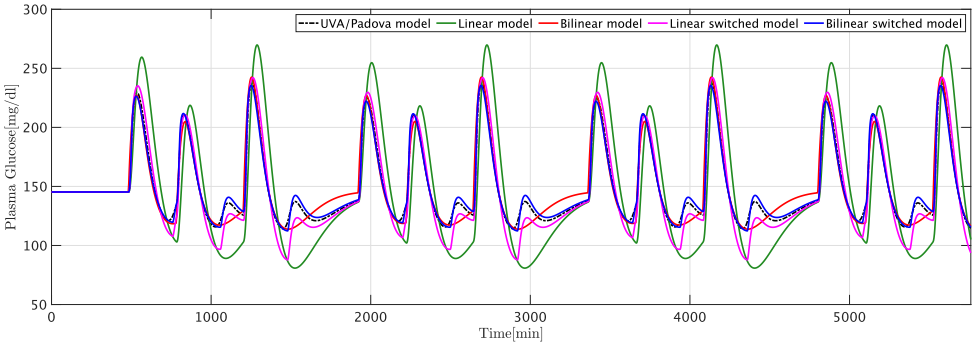


(g) Subject: 57

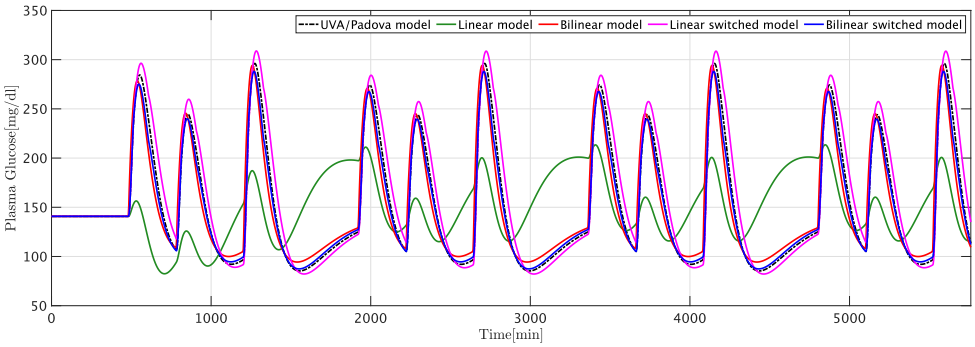


(h) Subject: 58

Fig. 7.1: Comparisons of UVA/Padova model and the approximate models.

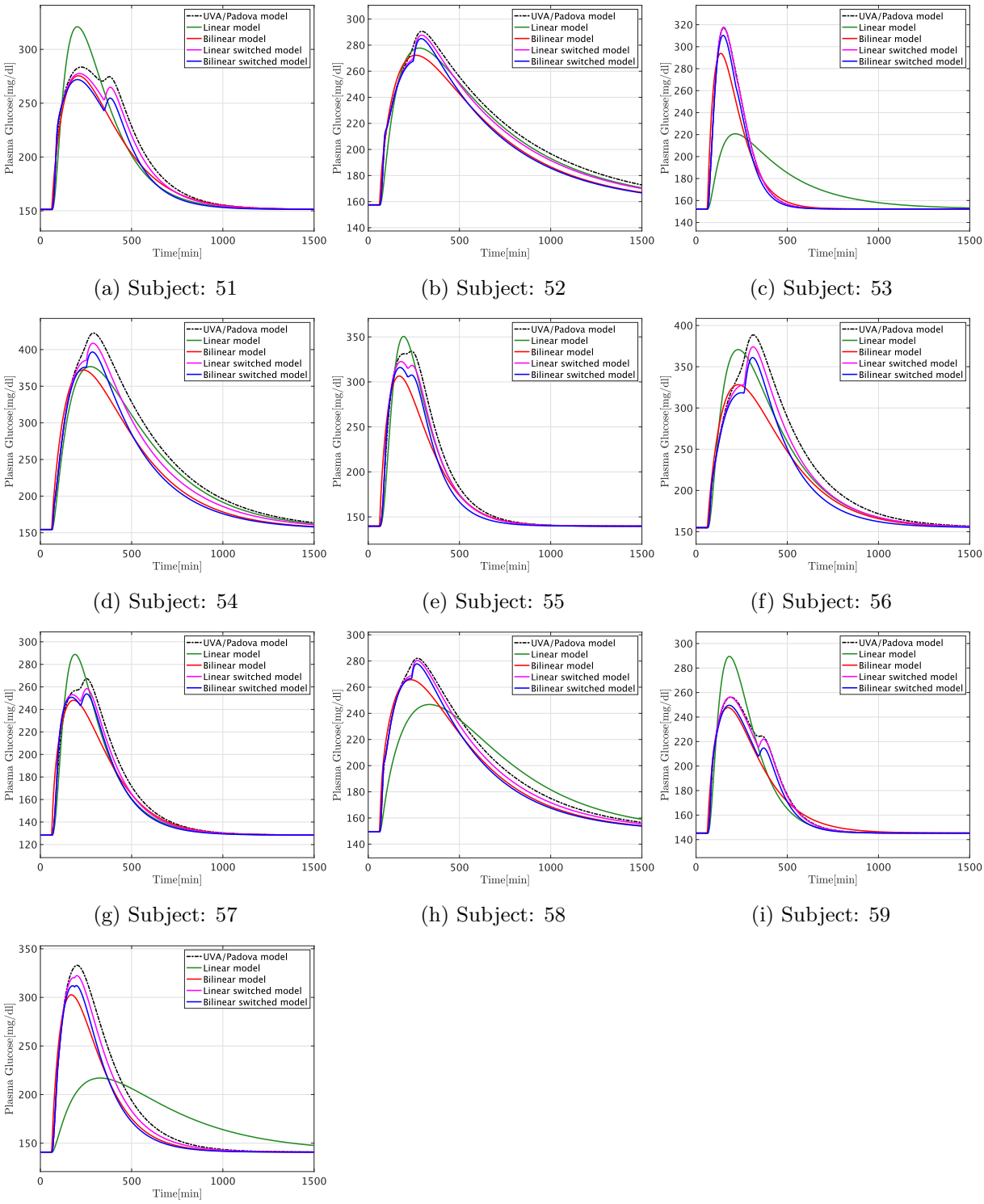


(i) Subject: 59



(j) Subject: 60

Fig. 7.1: Comparisons of plasma glucose trajectories using UVA/Padova model and the approximate models in normal condition. The glucose traces are simulated from subject 51 to subject 60. Each subject received the three standard meals for 4 days and corresponding bolus insulin, which is computed by the $CHOr$.



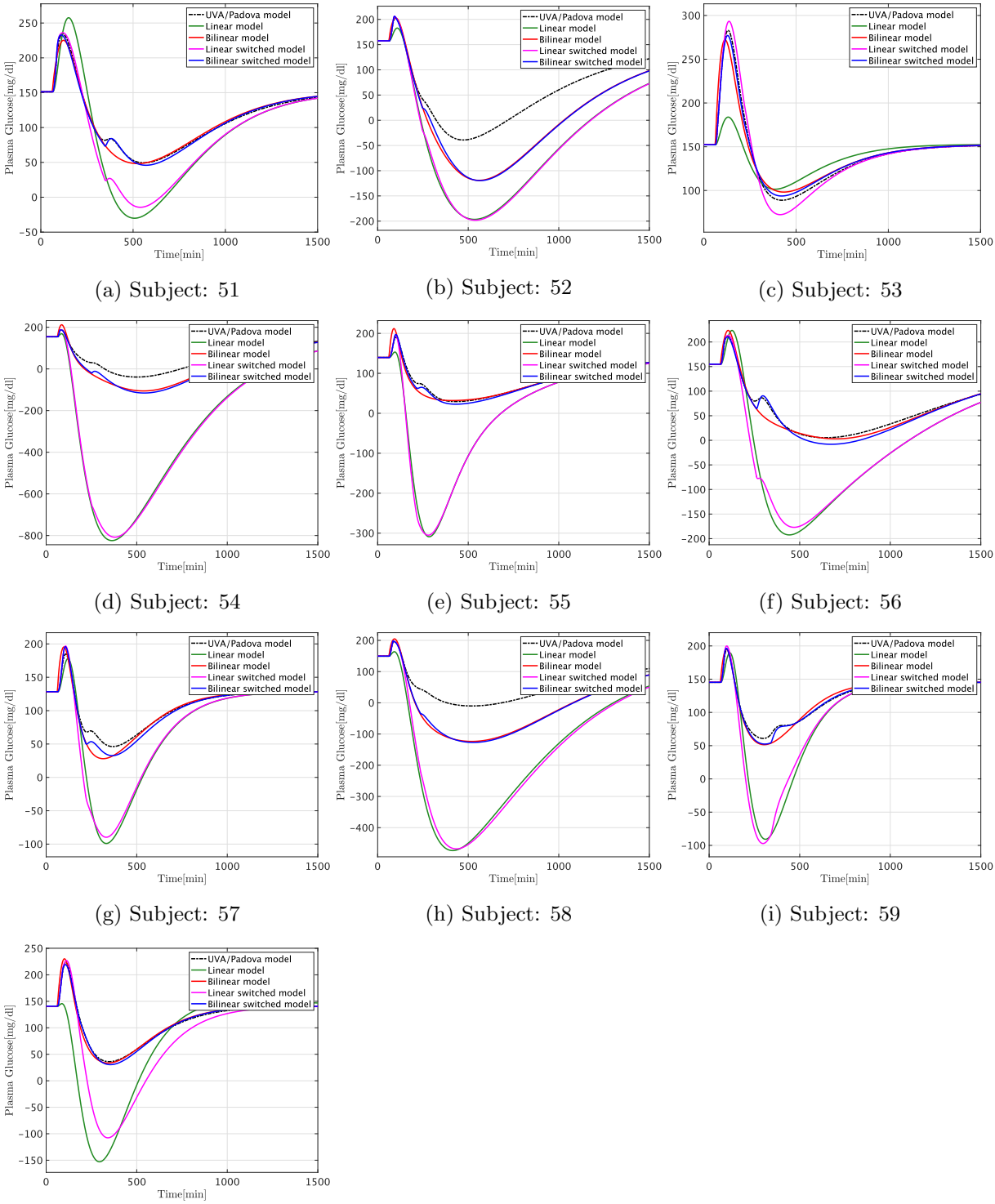
(a) Subject: 51 (b) Subject: 52 (c) Subject: 53

(d) Subject: 54 (e) Subject: 55 (f) Subject: 56

(g) Subject: 57 (h) Subject: 58 (i) Subject: 59

(j) Subject: 60

Fig. 7.2: Comparisons of plasma glucose trajectories using UVA/Padova model and the approximate models when not administering bolus insulin. The glucose traces are simulated from subject 51 to subject 60. Each subject received the same amounts of glucose (75g) but no bolus insulin.



(a) Subject: 51 (b) Subject: 52 (c) Subject: 53

(d) Subject: 54 (e) Subject: 55 (f) Subject: 56

(g) Subject: 57 (h) Subject: 58 (i) Subject: 59

(j) Subject: 60

Fig. 7.3: Comparisons of plasma glucose trajectories using UVA/Padova model and the approximate models when administering over insulin. The glucose traces are simulated from subject 51 to subject 60. Each subject received the same amounts of glucose (75g) and over bolus insulin, which is three times the standard insulin computed by the $CHOr$.

第 8 章

構築モデルの離散化誤差

8.1 概要

モデル予測制御では，内部モデルによって現状態から計算される未来の予測状態の最適化に基づいて制御入力を決定する[1]．モデル予測制御は，どれだけ未来までの予測をするかが制御性能を左右する．この予測長のことを予測ホライズンと呼ぶ．一般に，予測ホライズンは長いほうが制御性能は向上する．

コンピュータ内で予測状態の最適化を行うには，連続時間システムを離散時間システムに変換（離散化）する必要がある．そして，離散時間システムにより計算される離散時間上の予測状態と入力系列を最適化変数として，最適化計算を行う．この最適化変数の数 N_{opt} は，離散化サンプリング周期 T_s と予測ホライズン H に依存し，

$$N_{opt} \propto \frac{H}{T_s} \tag{8.1}$$

となる．つまり，予測ホライズンを長くするほど，また離散化サンプリング周期を短くするほど最適化変数の数が多くなり，計算量が増大する．

連続時間システムと離散時間システムの間には，離散化によって生じる誤差が存在する．以下では簡単のため，離散化誤差と呼ぶ．離散化誤差は，サンプリング周期を短くすることで十分に小さくできる．しかし，上記の理由から無制限にサンプリング周期を短くすることはできない．また，モデル予測制御の最適化計算は，設備的，また時間的に限られた状況で行う必要があり，計算量の低減が求められる．離散化モデルの離散化誤差が少なく，サンプリング周期を長くすることができれば，計算量を減少させ，長い予測ホライズンをとることが可能になる．そのため，離散化モデルの誤差の程度はモデルの評価指標の一つとして重要である．

本章では，連続時間モデルと離散時間モデルのシミュレーションを行い，十分小さな誤差で離散化が可能なサンプリング周期について検証する．

なお，今回の検証は時間経過に伴い累積する離散化ダイナミクスの計算誤差に焦点を当てる．そのため，シミュレーション時の入力や外乱は離散時間モデルのサンプリング点ごとにホールドし，連続時間モデルと離散時間モデルに同一のものを用いる．

8.2 検証方法

まず，各糖尿病の近似モデルを離散化する．非線形システムのモデルは 4 次のルンゲクッタ法を用いて離散化を行い，線形システムのモデルは 0 次ホールド法を用いて離散化を行う．

[1]詳しいモデル予測制御の説明に関しては，[15] を参照のこと．

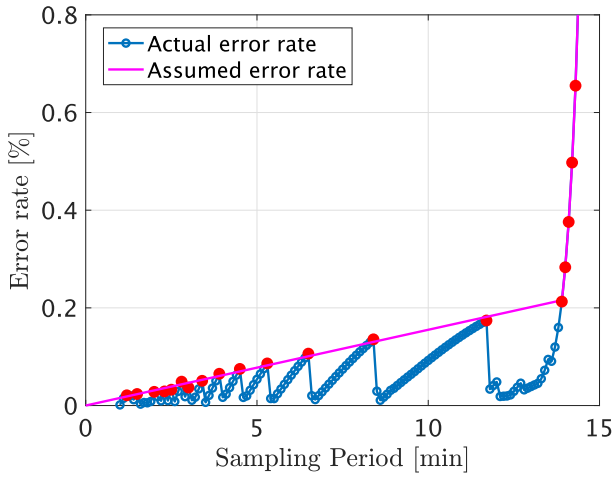


Fig. 8.1: Variation of error rate with sampling periods when using the bilinear switched model. Using the metabolic parameters of subject 55.

その後，各近似モデルの血糖値応答のシミュレーションを連続時間システムと離散時間システムを用いて行い，毎サンプリング点における誤差率を調べる．ただし，サンプリング点 k における誤差率 $\varepsilon(k)$ は，

$$\varepsilon(k) = \frac{|G_c(k) - G_d(k)|}{G_b} \tag{8.2}$$

として定義する．$G_c(k)$ はサンプリング点 k における連続時間システムの血糖値，$G_d(k)$ はサンプリング点 k における離散時間システムの血糖値，G_b は平衡状態の血糖値である．シミュレーション時間にわたる全てのサンプリング点において離散化誤差が 5% を下回る場合のみ，そのときのサンプリング周期で離散化が可能とする．

今回の検証では，時刻 60 分で 75g のグルコースと $CHOr$ により計算されるインスリンを同時に一度投与する条件のもと 1500 分間のシミュレーションを行い，離散化誤差を計測する．サンプリング周期を 1.0 分から 15.0 分まで 0.1 分刻みで変化させ，また代謝パラメータは Subject:51∼ 60 の 10 人分を用いて，1 モデルにつき 1510 回のシミュレーションを行う．各近似モデル，代謝パラメータの組に対して，離散化が可能な最大のサンプリング周期 (以下，許容サンプリング周期と記述) を求めた．

検証においては，前出の 3 つの近似モデルに加えて，双線形モデルに用いた 3 つの簡略化[2] を適用する/しないによって構成される 6 つのモデルも評価対象に加える．

切り換えモデルの離散化誤差は Fig. 8.1 に示すように，のこぎり波状に変化したのち，発散へ向かう．のこぎり波状の変化は，サンプリング点に対して食事等の時刻がちょうど噛み合うか否かで生じる．誤差率のピーク点の位置は食事のタイミングや量によって変化するので，ここでは，ピーク点を結んだ線 (Fig. 8.1, 桃線) をその切り換えモデルの誤差率とする．

[2]インスリン式，グルコース式，消化吸収式の簡略化の 3 つ．ただし，グルコース式の簡略化には，U_{id}, EGP, E の簡略化を含む

Table 8.1: Allowable sampling periods.

被験者番号	51	52	53	54	55	56	57	58	59	60	状態次元
UVA/Padova モデル	5	2.9	5.4	3.6	6.7	4.2	1.8	3	4.7	2.8	12
線形化モデル	-	-	-	-	-	-	-	-	-	-	12
双線形モデル	14.5	14.5	6.2	13.5	14.4	9.8	14.4	14.5	14	13.6	9
線形切り換えモデル	2.4	4.2	6.8	2.4	2.9	2.2	2.1	3.8	2.4	7.4	12
双線形切り換えモデル	2.5	4.5	4.5	2.9	3.2	2.3	2.4	2.8	3.0	2.6	11
インスリン式のみ簡略化	**14.5**	**14.3**	5.4	3.6	**14.4**	**9.7**	**14.4**	3	**13.6**	2.8	11
グルコース式のみ簡略化	5	2.9	5.4	3.6	6.7	4.2	1.8	3	4.7	2.8	12
消化吸収式のみ簡略化	5	2.9	**6.2**	**4.4**	6.7	4.2	1.8	**4.4**	4.7	**5.4**	10
インスリン, グルコース式を簡略化	14.5	14.3	5.4	3.6	14.4	9.6	14.4	3	13.5	2.8	11
消化吸収, インスリン式を簡略化	14.5	14.5	6.3	**13.1**	14.4	9.8	14.4	**14.5**	14	**13.8**	9
グルコース, 消化吸収式を簡略化	5	2.9	6.2	4.4	6.7	4.2	1.8	4.4	4.7	5.4 [min]	10

8.2.1 検証結果

各近似モデル，代謝パラメータの組に対して得られた許容サンプリング周期を Table 8.1 に示す．線形モデルは，今回の条件では離散化誤差は生じないため空欄としている．また，5 ～ 10 行目は双線形モデルの 3 つの簡略化を適用する/しないによって構成されるモデルである．

双線形モデルの許容サンプリング周期は，すべての患者において，UVA/Padova モデルよりも長くなっている．表の 6 行目を見ると，Subject: 51, 52, 55, 56, 57, 59 ではインスリン式の簡略化によって許容サンプリング周期が伸びている（緑色部）．一方， 表の 8 行目を見ると，Subject: 53, 54, 58, 60 では消化吸収式の簡略化によって許容サンプリング周期が伸びている（赤色部）．また，10 行目，Subject: 54, 58, 60 を見ると，インスリン式と消化吸収式をどちらも簡略化することにより許容サンプリング周期が大幅に伸びている（青色部）．許容サンプリング周期を伸ばすためにはインスリン式と消化吸収式の簡略化が効果的である．一方で，線形切り換えモデル，双線形切り換えモデルの離散化誤差は大きく，他のモデルに比べて許容サンプリング周期は短い．

第 9 章

おわりに

本論文では，1 型糖尿病患者を対象としたモデルベース血糖値制御即の実装を念頭におき，3 つの糖代謝動態近似モデルの構築を行った．また，これらのモデルの近似精度と離散化時に生じる誤差について検証した．各モデルは線形モデルに比べて良い近似精度を示しており，特に双線形切り換えモデルは最も良い近似が可能である．しかし，内部に切り換えシステムを含んでおり，複雑性は大きい．一方，双線形切り換えモデルは食事時の双峰性の応答は模擬できないものの，大きく精度を損なう範囲はなく，また 9 次元の双線形システムであるため複雑性が低い．さらに，許容サンプリング時間は他のモデルに比べて長くなっており，制御やモデル推定の計算に関して優位であると考えられる．

モデルベース制御体系の構築には，モデル構築と並行して制御則の選定，開発が必要である．各モデルの特徴に応じた制御アルゴリズムの構築が今後の課題である．

第 10 章

参考文献

[1] 医療情報科学研究所, 糖尿病・代謝・内分泌, 第 4 版, Medic Media, 2014.

[2] 日本内分泌学会, “糖尿病｜一般の皆様へ,” 2019. [Online]. Available: http://www.j-endo.jp/modules/patient/index.php?content_id=11

[3] L. H. Messer, G. P. Forlenza, J. L. Sherr, R. P. Wadwa, B. A. Buckingham, S. A. Weinzimer, D. M. Maahs, and R. H. Slover, “Optimizing hybrid closed-loop therapy in adolescents and emerging adults using the minimed 670g system,” *Diabetes Care*, vol. 41, no. 4, pp. 789–796, 2018.

[4] M. Messori, G. P. IncreMona, C. Cobelli, and L. Magni, “Individualized model predictive control for the artificial pancreas: In silico evaluation of closed-loop glucose control,” *IEEE Control Systems Magazine*, vol. 38, no. 1, pp. 86–104, 2018.

[5] L. Kovacs, P. Szalay, Z. Almassy, and L. Barkai, “Applicability results of a nonlinear model-based robust blood glucose control algorithm,” *Journal of Diabetes Science and Technology*, vol. 7, no. 3, pp. 708–716, 2013.

[6] L. Magni, D. M. Raimondo, L. Bossi, C. D. Man, G. D. Nicolao, B. Kovatchev, and C. Cobelli, “Model predictive control of type 1 diabetes: An in silico trial,” *Journal of Diabetes Science and Technology*, vol. 1, no. 6, pp. 804–812, 2007.

[7] B. P. Kovatchev, M. Breton, C. D. Man, and C. Cobelli, “In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes,” *Journal of Diabetes Science and Technology*, vol. 3, no. 1, pp. 44–55, 2009.

[8] C. Toffanin, M. Messori, F. D. Palma, G. D. Nicolao, C. Cobelli, , and L. Magni, “Artificial pancreas: Model predictive control design from clinical experience,” *Journal of Diabetes Science and Technology*, vol. 7, no. 6, pp. 1470–1483, 2013.

[9] R. Kadota, K. Sugita, K. Uchida, H. Yamada, M. Yamashita, and H. Kimura, “A mathematical model of type 1 diabetes involving leptin effects on glucose metabolism,” *Journal of Theoretical Biology*, vol. 456, pp. 213–223, 2018.

[10] B. P. Kovatchev, M. D. Breton, C. Cobeli, and C. D. Man, “Method,systemandcomputer simulation environment for testing of monitoring and control strategiesindabetes,” *UnitedStates Patent Application Publication*, vol. US2010/0179768A1, 2010.

[11] C. D. Man, R. A. Rizza, , and C. Cobelli, “Meal simulation model of the glucose-insulin system,” *IEEE transactions on bio-medical engineering*, vol. 54, no. 10, pp. 1740–1749,

2007.

[12] 厚生労働省, 日本人の食事摂取基準, 2020. [Online]. Available: https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/kenkou_iryou/kenkou/eiyou/syokuji_kijyun.html

[13] J. L. Knopp, L. Holder-Pearson, , and J. G. Chase, "Insulin units and conversion factors: A story of truth, boots, and faster half-truths," *Journal of Diabetes Science and Technology*, vol. 13, no. 3, pp. 597–600, 2019.

[14] 島津 章, Vii, 低血糖性昏睡, 日本内科学会雑誌, vol. 105, no. 4, pp. 683–689, 2016.

[15] 大塚 敏之, 非線形最適制御入門, システム制御工学シリーズ, コロナ社, 2011.

付録 A

コンパートメントモデル

A.1 コンパートメントモデル

コンパートメントモデルは，複数のコンパートメント（区画）と．コンパートメント間で受け渡される物質，ならびに制御信号によって表現される．

生理システムの場合，特定の物質を含む体内区画としてコンパートメントを配置し，その区画の中の物質濃度は一定とみなす．このコンパートメントの集合とコンパートメント間の物質流量[1]，制御信号を定義することで，生体システム内の物質動態を簡便に模擬することができる．

A.1.1 コンパートメントモデルの方程式

N_c 個のコンパートメントによってモデル化される場合，コンパートメントモデルは一般に，

$$\dot{Q}_m^i(t) = \sum_{\substack{j=1 \\ j\neq i}}^{N_c} R_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{N_c} R_{ji}, \quad i = 1, \cdots, N_c \tag{A.1}$$

として表される．ここで $Q_m^i(t)$ は i 個目のコンパートメントに含まれる物質の量，R_{ij} は j 番目のコンパートメントから i 番目のコンパートメントに移動する物質の流量である．(A.1) 式によって各コンパートメント間の物質の流出入が記述され，全体のシステムが定義される．

コンパートメント間の物質流量は線形または非線形式によって記述される．非線形式の場合，例えば

$$R_{ij}\left(Q_m^j(t)\right) = \frac{V_{\max} \cdot Q_m^j(t)}{C_m + Q_m^j(t)} \tag{A.2}$$

$$R_{ij}\left(Q_m^j(t)\right) = \begin{cases} k_r\left(Q_m^j(t) - k_t\right) & Q_m^j(t) < k_t \\ 0 & Q_m^j(t) \geq k_t, \end{cases} \tag{A.3}$$

が挙げられる．(A.2) 式はミカエリス・メンテン式であり，有限触媒中の反応速度を表す式である．本文中では，グルコース代謝利用式 (2.25) 式 がこれに当てはまる．Q_m^j の濃度が小さい範囲では，Q_m^j に対して R_{ij} は線形に変化する．Q_m^j が大きくなると，R_{ij} は飽和し，最大値 V_{max} に漸近する．C_m は，R_{ij} が $V_{max}/2$ をとるときの Q_m^j の値である．(A.3) 式は閾値 k_t を境に流量変化するような特性を表す．本文中では，グルコース腎排出式 (2.31) 式 がこれに当てはまる．k_r は速度定数である．

[1] ここでは，流量を，単位時間あたりに移動する物質の質量として定義する．

線形の場合，コンパートメント間の流量は，

$$R_{ij}(Q_m^j(t)) = k_{ij}Q_m^j(t) \tag{A.4}$$

として与えられる．ここで，k_{ij} は速度定数で，j 個目のコンパートメントから i 個目コンパートメントへの流量を決定するパラメータである．

A.1.2 簡易吸収モデル

体内における経口物質の動態を最も簡単に表すと，胃コンパートメントと血液コンパートメントによる 2-コンパートメントモデルが定義される（Fig. A.1 参照）．最初のコンパートメント Q_m^1 は胃内の物質量を表し，2 番目のコンパートメント Q_m^2 は血液中の物質量を表す．$u(t)$ の流量で経口摂取された物質は，胃内に到達し，流量 R_{21} で血液中へと吸収される．そして，流量 R_{02} で外部へと排泄される．

コンパートメントモデルのブロック図表現では，丸つきの破線によって外部から観測可能なコンパートメントであることを示す．例えば，血液中物質濃度は観測可能であるため，Fig. A.1 では，丸つきの波線を血液コンパートメントに付している．

さらに，コンパートメント間で質量を伴わない信号のみのやりとりが行われる場合，その作用は点線矢印で表現される．

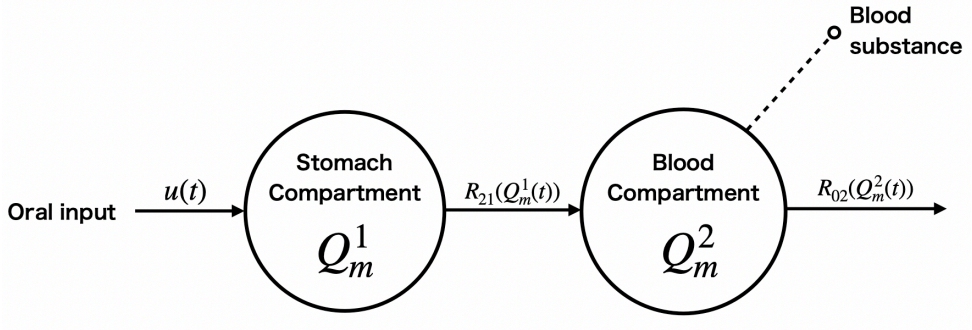


Fig. A.1: Two compartments of the very simplified human body representation.

付録 B

UVA/Padova モデルの変数および代謝パラメータ

B.1 変数

UVA/Padova モデルの状態変数をまとめて Table B.1 に示す．なお，空腹状態かつ基礎インスリン i_b 投与時の各状態変数の平衡値 $G_{pb}, G_{tb}, Q_{sto1b}, Q_{sto2b}, Q_{gutb}, X_b, I_{1b}, I_{db}, I_{lb}, I_{pb}, S_{1b}, S_{2b}$ は，(B.1)∼(B.12) 式によって求まる．ただし，各代謝パラメータ値は既知であり，血漿中グルコース濃度，血漿中インスリン濃度，およびグルコース生成流量の平衡値 G_b, I_b, EGP_b はあらかじめ患者ごとに与えられるとする．また，出力，入力ならびに状態依存の変数を Table B.2 に示す．G_b, I_b, EGP_b として記載した値は，Subject: 51 の平衡値である．

Table B.1: State variables of UVA/Padova model.

変数	単位	説明
G_p	[mg/kg]	血漿中グルコース量
G_t	[mg/kg]	組織中グルコース量
Q_{sto1}	[mg]	胃内個体グルコース量
Q_{sto2}	[mg]	胃内液体グルコース量
Q_{gut}	[mg]	腸内グルコース量
X	[pmol/l]	間質液中インスリン濃度
I_1	[pmol/l]	インスリン信号の遅延コンパートメント
I_d	[pmol/l]	インスリン遅れ信号
I_l	[pmol/kg]	肝臓中インスリン量
I_p	[pmol/kg]	血漿中インスリン量
S_1	[pmol/kg]	皮下複合体インスリン量
S_2	[pmol/kg]	皮下単量体インスリン量

$$G_{pb} = \frac{G_b}{V_G} \tag{B.1}$$

$$G_{tb} = \frac{F_{cns} - EGP_b + k_1 \cdot G_b \cdot V_G}{k_2} \tag{B.2}$$

$$Q_{sto1b} = 0 \tag{B.3}$$

$$Q_{sto2b} = 0 \tag{B.4}$$

$$Q_{gutb} = 0 \tag{B.5}$$

$$X_b = 0 \tag{B.6}$$

$$I_{1b} = I_b \tag{B.7}$$

$$I_{db} = I_b \tag{B.8}$$

$$I_{lb} = \frac{m_2 \cdot I_b \cdot V_I}{m_1 + m_3} \tag{B.9}$$

$$I_{pb} = \frac{I_b}{V_I} \tag{B.10}$$

$$S_{1b} = \frac{i_b}{k_{a1} + k_d} \tag{B.11}$$

$$S_{2b} = \frac{k_d \cdot S_{1b}}{k_{a2}} \tag{B.12}$$

Table B.2: Output, input and state-dependent variables of UVA/Padvoa model.

変数	値	単位	説明
G	151.3 ($= G_b$)	[mg/dl]	血漿中グルコース濃度（出力）
u_{ins}		[pmol/kg/min]	インスリン投与流量（入力）
d_{meal}		[mg/min]	食事摂取流量（入力）
I	90.56 ($= I_b$)	[pmol/l]	血漿中インスリン濃度
EGP	3.001 ($= EGP_b$)	[mg/kg/min]	グルコース生成流量
R_a		[mg/kg/min]	血漿中へのグルコース吸収流量
U_{ii}		[mg/kg/min]	インスリン非依存のグルコース利用流量
V_m		[mg/kg/min]	グルコース代謝利用の飽和流量
U_{id}		[mg/kg/min]	グルコース代謝利用流量
E		[mg/kg/min]	腎臓からのグルコース排出流量
Q_{sto}		[mg]	胃内グルコース量
k_{gut}		[1/min]	胃内グルコース排出速度係数
D		[mg]	食事量

B.2 代謝パラメータ

UVA/Padvoa モデルの代謝パラメータをまとめて Table B.3 に示す．ただし，記載した値は Subject: 51 のものである．* 印を付した値については，文献 [10] には記載がない代謝パラメータであり，文献 [11] に記載のパラメータ値を参照した．また，式番号を示したパラメータ値は，他パラ

メータの値から参照式の計算によって求まる.

Table B.3: Metabolic parameters of UVA/Padova model.

	パラメータ	値	単位	説明
皮下インスリン動態	k_{a1}	0.001	[1/min]	複合体インスリン吸収速度定数
	k_{a2}	0.007	[1/min]	単量体インスリン吸収速度定数
	k_d	0.017	[1/min]	皮下コンパートメント間速度定数
インスリン動態	m_1	0.156	[1/min]	インスリンコンパートメント間速度定数
	m_2	0.484*	[1/min]	インスリンコンパートメント間速度定数
	m_3	(2.6) 式	[1/min]	インスリン抽出速度定数
	m_4	0.194*	[1/min]	インスリン分解速度定数
	HE_b	0.6*	[-]	肝臓における定常インスリン抽出
	V_I	0.059	[l/kg]	インスリンの分布容積
	i_b	(2.9) 式	[pmol/kg/min]	基礎インスリン流量
食事吸収	k_{gri}	0.0558*	[1/min]	食事すりつぶし速度定数
	k_{abs}	0.111	[1/min]	腸からのグルコース吸収速度定数
	k_{max}	0.031	[1/min]	胃内グルコース排出速度係数の最大値
	k_{min}	0.006	[1/min]	胃内グルコース排出速度係数の最小値
	b	0.812	[-]	胃内グルコース排出速度係数のパラメータ
	d	0.129	[-]	胃内グルコース排出速度係数のパラメータ
	f	0.9	[-]	腸内グルコース吸収割合
	$\bar{t}$	-	[min]	食事開始時刻
	$\bar{t}_f$	-	[min]	食事終了時刻
グルコース動態	k_1	0.053	[1/min]	グルコースコンパートメント間の速度定数
	k_2	0.098	[1/min]	グルコースコンパートメント間の速度定数
	V_G	1.748	[dl/kg]	グルコースの分布容積

Table B.3: Metabolic parameters of UVA/Padova model.

	パラメータ	値	単位	説明
糖新生	k_{p1}	(2.24) 式	[mg/kg/min]	グルコース量とインスリン量が 0 のときの EGP の値
	k_{p2}	0.004	[1/min]	肝臓へのグルコース効果を表す定数
	k_{p3}	0.008	[mg/kg/min per pmol/l]	肝臓へのインスリン作用を表す定数
	k_i	0.006	[1/min]	インスリン遅れ信号の速度定数
グルコース利用	K_{m0}	199.5	[mg/kg]	ミカエリス・メンテン定数
	V_{m0}	(2.30) 式	[mg/kg/min]	$X = 0$ におけるグルコース代謝利用式の飽和速度
	V_{mx}	0.030	[mg/kg/min per pmol/l]	インスリン感受性の大きさを表す定数
	p_{2U}	0.015	[1/min]	組織へのインスリン作用の速度定数
	F_{cns}	1*	[mg/kg/min]	脳, 赤血球におけるグルコース利用
グルコース腎排出	k_{e1}	0.0005*	[1/min]	グルコース腎排出の速度定数
	k_{e2}	339*	[mg/kg]	グルコース腎排出の閾値
その他	age	24		年齢
	BW	76.37	[kg]	体重
	CHO_r	11.6	[g/U]	グルコース-インスリン比

付録 C

線形モデルのシステム行列

C.1 線形モデルのシステム行列

線形モデルのシステム行列 A, B, M, C を以下に示す.

$$A = \begin{bmatrix} a_{11} & k_2 & 0 & 0 & a_{15} & 0 & 0 & -k_{p3} & 0 & 0 & 0 & 0 \\ k_1 & a_{22} & 0 & 0 & 0 & a_{26} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -k_{gri} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{gri} & a_{44} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{54} & -k_{abs} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -p_{2U} & 0 & 0 & 0 & a_{710} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -k_i & 0 & 0 & a_{810} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k_i & k_i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{99} & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 & a_{1010} & k_{a1} & k_{a2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{1111} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_d & -k_{a2} \end{bmatrix} \tag{C.1}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{C.2}$$

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^\top \tag{C.3}$$

$$C = \begin{bmatrix} 1/V_G & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.4}$$

$$\begin{aligned}
a_{11} &= -k_{p2} - k_1 \\
a_{15} &= f k_{abs} / BW \\
a_{22} &= -\frac{V_{m0} \cdot K_m + V_{mx} \cdot K_{m0} \cdot X_b}{(K_{m0} + G_{tb})^2} - k_2 \\
a_{26} &= -\frac{V_{mx} \cdot G_{tb}}{K_{m0} + G_t b} \\
a_{44} &= -\frac{k_{max} - k_{min}}{2} \\
a_{54} &= \frac{k_{max} - k_{min}}{2} \\
a_{710} &= p_{2U} / V_I \\
a_{810} &= k_i / V_I \\
a_{99} &= -m_1 - m_3 \\
a_{1010} &= -m_2 - m_4 \\
a_{1111} &= -(k_{a1} - k_d)
\end{aligned}$$

付録 D

双線形モデルのグルコース吸収速度定数

D.1 双線形モデルのグルコース吸収速度定数

双線形モデルのグルコース吸収速度定数を，まとめて Table D.1 に示す.

Table D.1: Value of the glucose absorption rate of the bilinear model.

患者 ID	グルコース吸収速度定数 k
51	0.0065
52	0.0089
53	0.0117
54	0.0084
55	0.0097
56	0.0070
57	0.0087
58	0.0086
59	0.0066
60	0.0116

付録 E

基礎代謝基準値

E.1 基礎代謝基準値

厚生労働省食品摂取基準 [12] に記載の，男性における基礎代謝量の年齢分布を Table E.1 に示す.

Table E.1: Basal metaboli rate for male[12].

年齢 [age]	基礎代謝基準値 [kcal/kg/day]
1 ∼ 2	61.0
3 ∼ 5	54.8
6 ∼ 7	44.3
8 ∼ 9	40.8
10 ∼ 11	37.4
12 ∼ 14	31.0
15 ∼ 17	27.0
18 ∼ 29	23.7
30 ∼ 49	22.5
50 ∼ 64	21.8
65 ∼ 74	21.6
75 以上	21.5

付録 F

糖代謝モデルシミュレータ

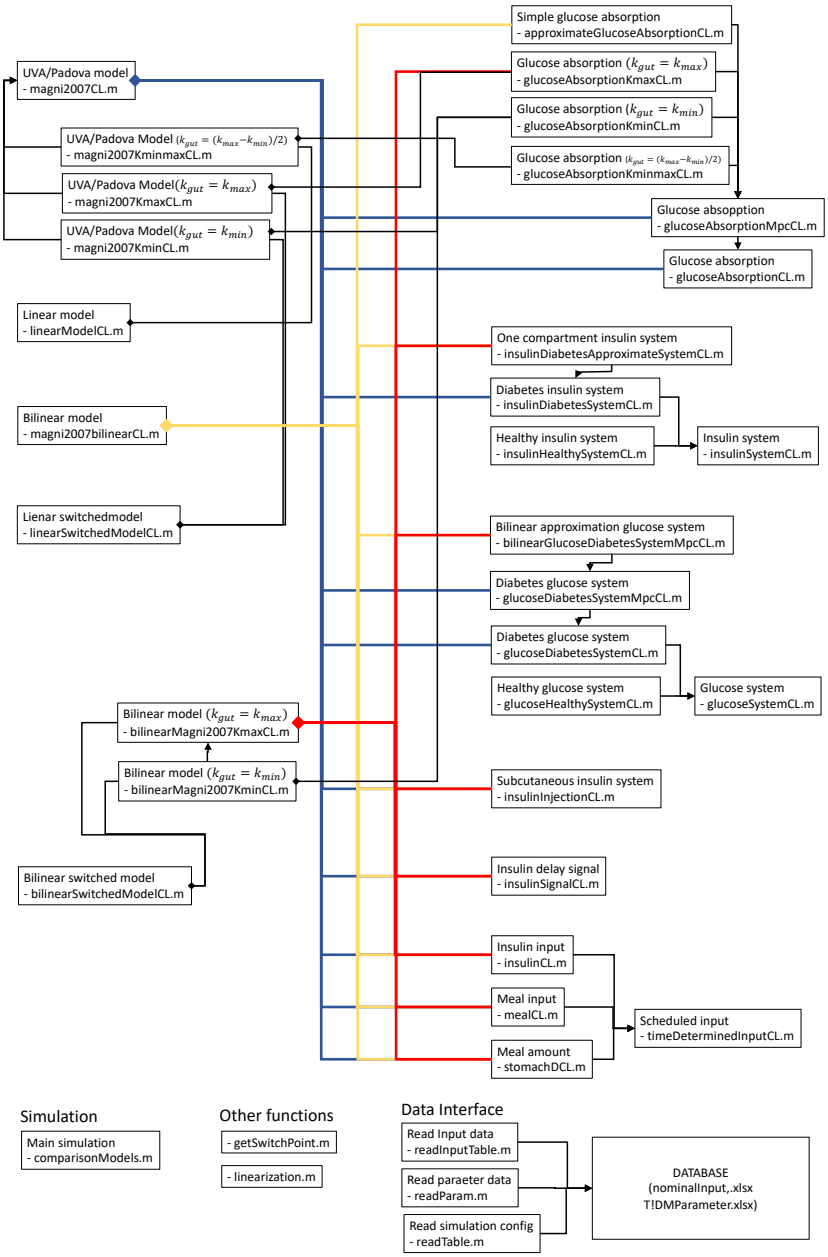


Fig. F.1: Shematic diagram of diabetes simulator.

本研究において構築した糖代謝モデルのシミュレータの構造，およびプログラムコードを示す．本シミュレータは，MATLAB 言語を用いて制作した．オブジェクト指向型の設計となっており，各臓器のシステムごとにクラスを作成し，それらを組み合わせる形でシミュレーションが構築される．近似モデルの場合には，基本的にオリジナルのシステムクラスを継承する形で近似システムのクラスを作成を行い，それを用いて近似モデルを構築する．各クラス間の関係は Fig. F.1 に示す通りである．各ブロックの 1 段目にはクラス，および関数の役割，2 段目にはクラス，および関数のファイル名が示されている．

また，患者パラメータや条件の設定などのシミュレータ全体で共有されるべき情報は，Excel 上のデータベースとして保管され，各ブロックからデータインターフェイスを通して呼び出される．

以下にプログラムコードを示す．掲載順はプログラムのファイル名に基づく辞書順である．

List F.1: approximateGlucoseAbsorptionCL

```
classdef approximateGlucoseAbsorptionCL < glucoseAbsorptionMpcCL
    methods
        function self = approximateGlucoseAbsorptionCL(subjectNum)
            self@glucoseAbsorptionMpcCL(subjectNum)

            self.k_abs = readParam(subjectNum, 'k');
            self.x_0 = 0;
        end

        function x_dot = dynamics(self, ~, x, ~, d)
            k_abs = self.k_abs;

            Q_gut = x(1);
            d_in = d(1);

            x_dot = -k_abs*Q_gut+d_in;
        end
    end
end
```

List F.2: bilinearGlucoseDiabetesSystemMpcCL

```
classdef bilinearGlucoseDiabetesSystemMpcCL < glucoseDiabetesSystemMpcCL
    methods
        function self = bilinearGlucoseDiabetesSystemMpcCL(subjectNum)
            self@glucoseDiabetesSystemMpcCL(subjectNum);
        end

        function U_id = getU_id(self, G_t, X)
            refine_W = 1.25;

            V_mx = self.V_mx;
            V_m0 = self.V_m0;
            K_m0 = self.K_m0;
            G_tb = self.G_tb;

            V_m = V_m0 + V_mx*X;
            U_idb =(V_m * G_tb) ./ (K_m0 + G_tb);
            U_id= U_idb + (V_m*K_m0*refine_W/((K_m0 + G_tb)^2))*(G_t-G_tb);
        end

        function E = getE(~,~)
            E = 0;
        end
    end
end
```

List F.3: bilinearMagni2007KmaxCL

```
classdef bilinearMagni2007KmaxCL
    properties
        attribute, subjectNum, glucoseSystem, insulinSystem, insulinInjection, insulinSignal, ...
            stomachD, glucoseAbsorption, meal, insulin, y_0, x_0, d_0, u_0, nx, nu, nd, ny, V_G, CR,
    end
    methods
        function self = bilinearMagni2007KmaxCL(use)
            self.attribute = 'diabetes';
            self.subjectNum = readTable('simulationConfig','subjectNum');
            self.V_G = readParam(self.subjectNum, 'V_G');
            self.CR = readParam(self.subjectNum, 'CHOr');

            self.insulinSystem = insulinDiabetesApproximateSystemCL(self.subjectNum);
            self.insulinSignal = insulinSignalCL(self.subjectNum);
            self.insulinInjection = insulinInjectionCL(self.subjectNum, self.insulinSystem.ins_b_perMinKg);

            if strcmp(use,'mpc')
                self.glucoseSystem = bilinearGlucoseDiabetesSystemMpcCL(self.subjectNum);
                self.glucoseAbsorption = glucoseAbsorptionKmaxCL(self.subjectNum);
            else
                error('normal is not exist with this model')
            end

            self.meal = mealCL;
            self.insulin = insulinCL(self.insulinInjection.ins_b, self.CR);
            self.stomachD = stomachDCL(self.glucoseAbsorption, self.meal);

            self.nx = 11;
            self.nu = 1;
            self.nd = 2;
            self.ny = 1;

            self.x_0 = [self.glucoseSystem.x_0
                        self.insulinSystem.x_0
                        self.insulinSignal.x_0
                        self.insulinInjection.x_0
                        self.glucoseAbsorption.x_0
                       ];
            self.y_0 = self.x_0(1)/self.V_G;
            self.u_0 = self.insulinInjection.ins_b;
            self.d_0 = [self.meal.d_0
                        self.stomachD.D_0
                       ];
        end

        function x_dot = dynamics(self,t,x,u,d)
            G_p = x(1);
            G_t = x(2);
            I_p = x(3);
            I_1 = x(4);
            I_d = x(5);
            X = x(6);
            S_1 = x(7);
            S_2 = x(8);
            Q_sto1 = x(9);
            Q_sto2 = x(10);
            Q_gut = x(11);

            insInput = u(1);
            d_in = d(1);
            D = d(2);

            Ra = self.glucoseAbsorption.getR_a(Q_gut);
            I = self.insulinSystem.getI(I_p);
```

```
            S = self.insulinInjection.getS([S_1;S_2]);

            G_dot = self.glucoseSystem.dynamics(t,[G_p;G_t],0,[Ra; X; I_d]);
            I_dot = self.insulinSystem.dynamics(t, I_p, 0, S);
            Is_dot = self.insulinSignal.dynamics(t, [I_1; I_d; X], 0, I);
            S_dot = self.insulinInjection.dynamics(t, [S_1;S_2],insInput,0);
            Q_dot = self.glucoseAbsorption.dynamics(t,[Q_sto1; Q_sto2; Q_gut],0,[d_in;D
                ]);

            x_dot = [G_dot; I_dot; Is_dot; S_dot; Q_dot];
        end

        % 出力変数，次元を変えたらny，y_0 もかえること
        function y = output(self,~,x,~,~)
% y = [x(1); x(2); x(3); x(4); x(5); x(6); x(7); x(8); x(9); x(10); x(11); x(12)];
             y = x(1)/self.V_G;
        end
    end
end
```

List F.4: bilinearMagni2007KminCL

```
% k_gut を，k_min に変更

classdef bilinearMagni2007KminCL < bilinearMagni2007KmaxCL
    methods
        function self = bilinearMagni2007KminCL(~)
            self@bilinearMagni2007KmaxCL('mpc')
            self.glucoseAbsorption = glucoseAbsorptionKminCL(self.subjectNum);
        end
    end
end
```

List F.5: bilinearSwitchedModelCL

```
classdef bilinearSwitchedModelCL
    properties
        attribute, subjectNum, model1, model2, linearModel1, linearModel2, ...
            insulin, meal, stomachD, ...
            y_0, x_0, d_0, u_0, nx, nu, nd, ny, V_G, CR
    end

    methods
        function self = bilinearSwitchedModelCL(use)
            self.attribute = 'diabetes';
            self.subjectNum = readTable('simulationConfig','subjectNum');
            self.V_G = readParam(self.subjectNum, 'V_G');
            self.CR = readParam(self.subjectNum, 'CHOr');

            if strcmp(use,'mpc')
                self.model1 = bilinearMagni2007KmaxCL('mpc');
                self.model2 = bilinearMagni2007KminCL('mpc');
            elseif strcmp(use, 'normal')
                self.model1 = bilinearMagni2007KmaxCL('normal');
                self.model2 = bilinearMagni2007KminCL('normal');
            else
                error('mpc or normal')
            end

            self.meal = self.model1.meal;
            self.stomachD = self.model1.stomachD;
            self.insulin = self.model1.insulin;

            self.x_0 = self.model1.x_0;
```

```
            self.y_0 = self.model1.y_0;
            self.u_0 = self.model1.u_0;
            self.d_0 = self.model1.d_0;
        end

        function x_dot = dynamics(self,t,x,u,d,sw1,sw2)
            if x(9)+x(10) > sw2 || x(9)+x(10) < sw1
                x_dot = self.model1.dynamics(t,x,u,d);
            elseif x(9)+x(10) >= sw1 && x(9)+x(10) <= sw2
                x_dot = self.model2.dynamics(t,x,u,d);
            end
        end
    end
end
```

List F.6: comparisonModels

```
%% 各モデルのオブジェクトを生成
magniModel = magni2007CL('mpc'); %  UVA/Padova モデル
magniKminmaxModel = magni2007KminmaxCL('mpc'); % k_gut を (k_max - k_min)/2 にしたモデル
linearModel = linearModelCL(magniKminmaxModel); % 線形モデル
bilinearModel = magni2007bilinearCL('mpc'); % 双線形モデル
%%
linearSwitchedModel = linearSwitchedModelCL('mpc'); % 線形切り換えモデル
%%
bilinearSwitchedModel = bilinearSwitchedModelCL('mpc'); % 双線形切り換えモデル

%% 切り換え点を計算
mealEvent = readInputTable(1);
D = mealEvent(3);

b = readParam(subjectNum,'b');
d = readParam(subjectNum,'d');
subjectNum = readTable('simulationConfig','subjectNum');
[sw_b, sw_d] = getSwitchPoint(D, subjectNum);

sw1 = sw_d;
sw2 = sw_b;

%% 各モデルの軌道を計算
options = odeset('MaxStep', 0.5, 'RelTol',1e-12);
[t_m,x_m] = ode45(@(t,x)magniModel.dynamics(t,x,magniModel.insulin.getInsulinInput(t),[
    magniModel.meal.getMealInput(t); magniModel.stomachD.getDInput(t)]),[0 1500],
    magniModel.x_0, options);
[t_l,x_l] = ode45(@(t,x)linearModel.dynamics(t,x,magniModel.insulin.getInsulinInput(t),[
    magniModel.meal.getMealInput(t); magniModel.stomachD.getDInput(t)]),[0 1500],
    magniModel.x_0, options);
[t_b,x_b] = ode45(@(t,x)bilinearModel.dynamics(t,x,magniModel.insulin.getInsulinInput(t
    ),magniModel.meal.getMealInput(t)),[0 1500],bilinearModel.x_0, options);
[t_lh,x_lh] = ode45(@(t,x)linearSwitchedModel.dynamics(t,x,magniModel.insulin.
    getInsulinInput(t),[magniModel.meal.getMealInput(t); magniModel.stomachD.getDInput(
    t)], sw1, sw2),[0,1500],magniModel.x_0, options);
[t_bh,x_bh] = ode45(@(t,x)bilinearSwitchedModel.dynamics(t,x,magniModel.insulin.
    getInsulinInput(t),[magniModel.meal.getMealInput(t); magniModel.stomachD.getDInput(
    t)], sw1, sw2),[0,1500],bilinearSwitchedModel.x_0, options);

%% プロット
V_G = magniModel.V_G;
fig = figure;
plot(t_m,x_m(:,1)/V_G,'k-.', 'Linewidth',2.5)
hold on
plot(t_l,x_l(:,1)/V_G,'color','#228b22','Linewidth',2.5)
plot(t_b,x_b(:,1)/V_G,'color','#ff0000','Linewidth',2.5)
plot(t_lh,x_lh(:,1)/V_G,'color','#ff00ff','Linewidth',2.5)
plot(t_bh,x_bh(:,1)/V_G,'color','#0000ff','Linewidth',2.5)
xlabel('Time[min]');
ylabel('Plasma Glucose[mg/dl]')
```

```
legend('UVA/Padova model','Linear model','Bilinear model','Linear switched model','
    Bilinear switched model');
```

List F.7: getSwitchPoint

```
function [sw_b, sw_d] = getSwitchPoint(D,subjectNum)

    k_min = readParam(subjectNum,'k_min');
    k_max = readParam(subjectNum,'k_max');
    b = readParam(subjectNum,'b');
    d = readParam(subjectNum,'d');

    load('parameta_b.mat');
    beta = betaSol;

    b1 = beta(1);
    b2 = beta(2);
    b3 = beta(3);
    b4 = beta(4);
    b5 = beta(5);
    b6 = beta(6);

    r_b = b1*exp(b2*k_max) + b3*exp(b4*k_min) + b5*b + b6;


    load('parameta_d.mat');
    beta = betaSol;

    b1 = beta(1);
    b2 = beta(2);
    b3 = beta(3);
    b4 = beta(4);
    b5 = beta(5);
    b6 = beta(6);

    r_d = b1*exp(b2*k_max) + b3*exp(b4*k_min) + b5*d + b6;

    sw_b = b*D - r_b*D*(1-b);
    sw_d = d*D + r_d*D*d;
end
```

List F.8: glucoseAbsorptionCL

```
classdef glucoseAbsorptionCL
    properties
        k_min, k_max, k_gri, k_abs, d, b, x_0, f, BW
    end
    methods
        function self = glucoseAbsorptionCL(subjectNum)
            self.k_min = readParam(subjectNum,'k_min');
            self.k_max = readParam(subjectNum,'k_max');
            self.k_gri = readParam(subjectNum,'k_gri');
            self.k_abs = readParam(subjectNum,'k_abs');
            self.BW = readParam(subjectNum,'BW');
            self.d = readParam(subjectNum,'d');
            self.b = readParam(subjectNum,'b');
            self.f = readParam(subjectNum,'f');
            self.x_0 = [0;0;0];
        end

        function x_dot = dynamics(self,~,x,~,dist)

            k_gri = self.k_gri;
            k_abs = self.k_abs;
```

```
            Q_sto1 = x(1);
            Q_sto2 = x(2);
            Q_gut = x(3);
            Q_sto = Q_sto1 + Q_sto2;

            d_in = dist(1);
            D = dist(2);

            k_gut = self.getK_gut(Q_sto, D);

            x_dot = [-k_gri*Q_sto1+d_in
                -k_gut*Q_sto2+k_gri*Q_sto1
                -k_abs*Q_gut+k_gut*Q_sto2
                ];
        end

        function k_gut = getK_gut(self,Q_sto, D)

            k_min = self.k_min;
            k_max = self.k_max;
            b = self.b;
            d = self.d;

            alpha_gut = 5 / (2 * D * (1 - b));
            beta_gut = 5 / (2 * D * d);

            if D ~= 0
                k_gut = k_min + (k_max - k_min) * (tanh(alpha_gut * (Q_sto - b * D))-tanh
                    (beta_gut * (Q_sto - d * D))+2)/2;
            elseif D == 0
                k_gut = k_max;
            end
        end

        function R_a = getR_a(self, Q_gut)
            BW = self.BW;
            f = self.f;
            k_abs = self.k_abs;
            R_a = f .* k_abs .* Q_gut ./ BW;
        end
    end
end
```

List F.9: glucoseAbsorptionKmaxCL

```
classdef glucoseAbsorptionKmaxCL < glucoseAbsorptionMpcCL
    methods
        function self = glucoseAbsorptionKmaxCL(subjectNum)
            self@glucoseAbsorptionMpcCL(subjectNum)
        end
        function k_gut = getK_gut(self,~, ~)
            k_max = self.k_max;
            k_gut = k_max;
        end
    end
end
```

List F.10: glucoseAbsorptionKminCL

```
classdef glucoseAbsorptionKminCL < glucoseAbsorptionMpcCL
    methods
        function self = glucoseAbsorptionKminCL(subjectNum)
            self@glucoseAbsorptionMpcCL(subjectNum)
        end
```

```
        function k_gut = getK_gut(self,~, ~)
            k_min = self.k_min;
            k_gut = k_min;
        end
    end
end
```

List F.11: glucoseAbsorptionKminmaxCL

```
classdef glucoseAbsorptionKminmaxCL < glucoseAbsorptionMpcCL
    methods
        function self = glucoseAbsorptionKminmaxCL(subjectNum)
            self@glucoseAbsorptionMpcCL(subjectNum)
        end
        function k_gut = getK_gut(self,~, ~)
            k_min = self.k_min;
            k_max = self.k_max;

            k_gut = (k_max-k_min)/2;
        end
    end
end
```

List F.12: glucoseAbsorptionMpcCL

```
classdef glucoseAbsorptionMpcCL < glucoseAbsorptionCL
    methods
        function self = glucoseAbsorptionMpcCL(subjectNum)
            self@glucoseAbsorptionCL(subjectNum);
        end
        function x_dot = mpcDynamics(self,~,x,~,dist)

            k_gri = self.k_gri;
            k_abs = self.k_abs;

            Q_sto1 = x(1);
            Q_sto2 = x(2);
            Q_gut = x(3);
            Q_sto = Q_sto1 + Q_sto2;

            d_in = dist(1);
            D = dist(2);

            k_gut = self.getK_gut(Q_sto, D);

            x_dot = [-k_gri*Q_sto1+d_in
                -k_gut*Q_sto2+k_gri*Q_sto1
                -k_abs*Q_gut+k_gut*Q_sto2
                ];
        end

        function k_gut = getK_gut(self,Q_sto, D)
            k_min = self.k_min;
            k_max = self.k_max;
            b = self.b;
            d = self.d;

            alpha_gut = 5 / (2 * D * (1 - b));
            beta_gut = 5 / (2 * D * d);
            k_gut = k_min + (k_max - k_min) * (tanh(alpha_gut * (Q_sto - b * D))-tanh(
                beta_gut * (Q_sto - d * D))+2)/2;
        end
    end
end
```

List F.13: glucoseDiabetesSystemCL

```
classdef glucoseDiabetesSystemCL < glucoseSystemCL
    methods
        function self = glucoseDiabetesSystemCL(subjectNum)
            self@glucoseSystemCL(subjectNum)
        end
        % 抽象メソッドを実体化
        function EGP = getEGP(self, G_p, I_d, ~)
            k_p1 = self.k_p1;
            k_p2 = self.k_p2;
            k_p3 = self.k_p3;
            EGP = max(k_p1 - k_p2 * G_p - k_p3 * I_d, 0);
        end
    end
end
```

List F.14: glucoseDiabetesSystemMpcCL

```
classdef glucoseDiabetesSystemMpcCL < glucoseDiabetesSystemCL
    methods
        function self = glucoseDiabetesSystemMpcCL(subjectNum)
            self@glucoseDiabetesSystemCL(subjectNum)
        end

        % 不連続な関数を近似してオーバーライド，getE は grantParent の method
        function E = getE(self, G_p)
            k_e2 = self.k_e2;
            k_e1 = self.k_e1;

            E = k_e1*(G_p - k_e2)/(1+exp(-(G_p-k_e2)));
        end

        function EGP = getEGP(self, G_p, I_d, ~)
            k_p1 = self.k_p1;
            k_p2 = self.k_p2;
            k_p3 = self.k_p3;
            EGP = k_p1 - k_p2 * G_p - k_p3 * I_d;
        end
    end
end
```

List F.15: glucoseHealthySystemCL

```
classdef glucoseHealthySystemCL < glucoseSystemCL
    properties
        k_p4, I_pob
    end
    methods
        function self = glucoseHealthySystemCL(subjectNum)
            self@glucoseSystemCL(subjectNum)
            self.k_p4 = readParam(subjectNum,'k_p4');
            self.I_pob = readParam(subjectNum, 'I_pob');
            self.k_p1 = self.EGP_b+self.k_p2*self.G_b*self.V_G+self.k_p3*self.I_b+self.
                k_p4*self.I_pob;

            self.G_pb = self.G_b*self.V_G;
            self.x_0 = [self.G_pb; self.G_tb];
        end

        function EGP = getEGP(self, G_p, I_d, I_po)
            k_p1 = self.k_p1;
            k_p2 = self.k_p2;
            k_p3 = self.k_p3;
```

```
            k_p4 = self.k_p4;
            EGP = max(k_p1 - k_p2 * G_p - k_p3 * I_d - k_p4*I_po, 0);
        end
    end
end
```

List F.16: glucoseSystemCL

```
classdef glucoseSystemCL
    properties
        F_cns, k_1, k_2, k_p1, k_p2, k_p3, k_e1, k_e2, ...
            K_m0, V_m0, K_mx, V_mx, EGP_b, G_b, V_G, x_0, I_b, ...
            G_tb, G_pb,
    end

    methods
        function self = glucoseSystemCL(subjectNum)
            self.F_cns = readParam(subjectNum,'F_cns');
            self.k_1 = readParam(subjectNum,'k_1');
            self.k_2 = readParam(subjectNum,'k_2');
            self.k_p2 = readParam(subjectNum,'k_p2');
            self.k_p3 = readParam(subjectNum,'k_p3');
            self.k_e1 = readParam(subjectNum,'k_e1');
            self.k_e2 = readParam(subjectNum,'k_e2');
            self.K_m0 = readParam(subjectNum,'K_m0');
            self.K_mx = readParam(subjectNum,'K_mx');
            self.V_mx = readParam(subjectNum,'V_mx');
            self.EGP_b = readParam(subjectNum,'EGP_b');
            self.G_b = readParam(subjectNum,'G_b');
            self.I_b = readParam(subjectNum,'I_b');
            self.V_G = readParam(subjectNum,'V_G');
            self.k_p1 = self.EGP_b+self.k_p2*self.G_b*self.V_G+self.k_p3*self.I_b;
            self.G_tb = (self.F_cns-self.EGP_b+self.k_1*self.G_b*self.V_G)/self.k_2;
            self.V_m0 = (self.EGP_b-self.F_cns)*(self.K_m0+self.G_tb)/self.G_tb;
            self.G_pb = self.G_b*self.V_G;
            self.x_0 = [self.G_pb; self.G_tb];
        end

        function x_dot = dynamics(self,~, x, ~, d)
            k_1 = self.k_1;
            k_2 = self.k_2;
            U_ii = self.F_cns;

            G_p = x(1);
            G_t = x(2);
            R_a = d(1);
            X = d(2);
            I_d = d(3);

            I_po = [];

            if length(d) > 3
                I_po = d(4);
            end

            EGP = self.getEGP(G_p, I_d, I_po);
            U_id = self.getU_id(G_t, X);
            E = self.getE(G_p);

            x_dot = [EGP+R_a-U_ii-k_1*G_p+k_2*G_t-E
                    -U_id+k_1*G_p-k_2*G_t
                    ];
        end

        function E = getE(self, G_p)
            k_e2 = self.k_e2;
            k_e1 = self.k_e1;
            if G_p > k_e2
```

```
                E = k_e1*(G_p - k_e2);
            else
                E = 0;
            end
        end

        function U_id = getU_id(self, G_t, X)
            K_m0 = self.K_m0;
            K_mx = self.K_mx;
            V_m0 = self.V_m0;
            V_mx = self.V_mx;

            K_m = K_m0 + K_mx*X;
            V_m = V_m0 + V_mx*X;
            U_id =(V_m * G_t) / (K_m + G_t);
        end

        function G = getG(self, G_p)
            G = G_p/self.V_G;
        end
    end

    methods (Abstract = true)
        EGP = getEGP(self, G_p, I_d, I_po);
    end
end
```

List F.17: insulinCL

```
% 時間依存の入力の設定，
% insulinArray = [[入力開始時刻，入力時間，入力量]; ... [];];
% 重複する場合は，前の入力が優先される
% インスリン量の単位は [pmol]
% input の単位は[pmol/min]

classdef insulinCL < timeDeterminedInputCL
    properties
        ins_0
    end
    methods
        function self = insulinCL(basalInsulin, CR)
            self@timeDeterminedInputCL()
            self.ins_0 = basalInsulin;
            CRInsulinTF = readTable('inputConfig','CRInsulinTF');

            % CRInsulinTF が true の場合は，インスリン量は全てCR 比によって決定される
            if CRInsulinTF == true
                mealInputArray = self.makeInputArray('meal');
                % CR (g/U) で割り，1U = 6000 をかけ、pmol 単位で inputArray を作成
                % mealinputArray が空なら，inputArray も空にする
                if ~isempty(mealInputArray)
                    inputArray = ...
                        [mealInputArray(:,[1 2]), mealInputArray(:,3).*6000./(1000*CR)];
                elseif isempty(mealInputArray)
                    inputArray = [];
                end
                self.inputArray = inputArray;
            else
                self.inputArray = self.makeInputArray('insulin');
            end
        end

        function insulinInput = getInsulinInput(self, t)
            determinedInsulinInput= self.getDeterminedInput(t);
            insulinInput = determinedInsulinInput + self.ins_0;
        end
    end
```

```
end
```

List F.18: insulinDiabetesApproximateSystemCL

```
classdef insulinDiabetesApproximateSystemCL < insulinDiabetesSystemCL
    properties
        m_7
    end
    methods
        function self = insulinDiabetesApproximateSystemCL(subjectNum)
            self@insulinDiabetesSystemCL(subjectNum);
            self.x_0 = self.I_pb;
            self.m_7 = self.ins_b_perMinKg/(self.I_b*self.V_I);
        end

        function x_dot = dynamics(self,~,x,~,d)
            m_7 = self.m_7;

            I_p = x(1);
            S = d(1);

            x_dot = -(m_7)*I_p+S;
        end
    end
end
```

List F.19: insulinDiabetesSystemCL

```
classdef insulinDiabetesSystemCL < insulinSystemCL
    properties
        ins_b_perMinKg
    end
    methods
        function self = insulinDiabetesSystemCL(subjectNum)
            self@insulinSystemCL(subjectNum)

            m_3b = self.getM_3(0);
            self.I_lb = self.m_2*self.I_b*self.V_I/(self.m_1+m_3b);
            self.I_pb = self.I_b*self.V_I;
            self.x_0 = [self.I_lb; self.I_pb];

            self.ins_b_perMinKg = ...
                ((self.m_2+self.m_4)*self.I_b*self.V_I-self.m_1*self.m_2*self.I_b*self.
                    V_I/(self.m_1+m_3b)); %pmol/min/kg
        end

        function x_dot = dynamics(self, t, x, u, d)
            m_1 = self.m_1;
            m_2 = self.m_2;
            m_4 = self.m_4;

            I_l = x(1);
            I_p = x(2);
            S = d(1);

            m_3 = self.getM_3(S);

            x_dot = [-(m_1+m_3)*I_l+m_2*I_p
                    -(m_2+m_4)*I_p+m_1*I_l+S
                    ];
        end

        function m_3 = getM_3(self, ~)
            m_1 = self.m_1;
            HE_b = self.HE_b;
```

```
            m_3 = m_1*(HE_b/(1-HE_b));
        end
    end
end
```

List F.20: insulinHealthySystem

```
classdef insulinHealthySystem < insulinSystemCL
    properties
        m_5, m_6,
    end
    methods
        function self = insulinHealthySystem(subjectNum)
            self@insulinSystemCL(subjectNum)
            self.m_5 = readParam(subjectNum, 'm_5');
            self.m_6 = readParam(subjectNum, 'm_6');

            self.I_pb = self.I_b*self.V_I;
            self.I_lb = (self.m_2+self.m_4)*self.I_pb/self.m_1;

            self.x_0 = [self.I_lb; self.I_pb];
        end

        function x_dot = dynamics(self, ~, x, ~, d)

            I_l = x(1);
            I_p = x(2);
            S = d(1);

            m_1 = self.m_1;
            m_2 = self.m_2;
            m_4 = self.m_4;
            m_5 = self.m_5;
            m_6 = self.m_6;

            HE = -m_5 * S + m_6;
            m_3 = HE * m_1/(1-HE);


            x_dot = [-(m_1+m_3)*I_l+m_2*I_p+S
                    -(m_2+m_4)*I_p+m_1*I_l
                    ];
        end
    end
end
```

List F.21: insulinInjectionCL

```
classdef insulinInjectionCL
    properties
        k_a1, k_a2, k_d, BW, x_0, S_1b, S_2b, ins_b;
    end
    methods
        function self = insulinInjectionCL(subjectNum, ins_b_perMinKg)
            self.k_a1 = readParam(subjectNum, 'k_a1');
            self.k_a2 = readParam(subjectNum, 'k_a2');
            self.k_d = readParam(subjectNum, 'k_d');
            self.BW = readParam(subjectNum, 'BW');
            self.ins_b = ins_b_perMinKg*self.BW; %pmol/min
            self.S_1b = self.ins_b/(self.BW*(self.k_a1+self.k_d)); %pmol/kg
            self.S_2b = self.k_d*self.S_1b/self.k_a2;
            self.x_0 = [self.S_1b; self.S_2b];
        end

        function x_dot = dynamics(self, ~, x, u, ~)
```

```

            S_1 = x(1);
            S_2 = x(2);
            ins = u(1);

            k_a1 = self.k_a1;
            k_a2 = self.k_a2;
            BW = self.BW;
            k_d = self.k_d;

            x_dot = [-(k_a1+k_d)*S_1+ins/BW
                    k_d*S_1-k_a2*S_2
                    ];
        end

        function S = getS(self, x)
            k_a1 = self.k_a1;
            k_a2 = self.k_a2;
            S_1 = x(1);
            S_2 = x(2);
            S = k_a1*S_1+k_a2*S_2;
        end
    end
end
```

List F.22: insulinSecretionCL

```
classdef insulinSecretionCL
    properties
        S_b, alpha, beta, gamma, K, h, x_0, I_pob, Y_b

    end
    methods
        function self = insulinSecretionCL(subjectNum)
            self.S_b = readParam(subjectNum, 'S_b');
            self.alpha = readParam(subjectNum, 'alpha');
            self.beta = readParam(subjectNum, 'beta');
            self.gamma = readParam(subjectNum, 'gamma');
            self.h = readParam(subjectNum, 'h');
            self.K = readParam(subjectNum, 'K');
            self.I_pob = self.S_b/self.gamma;
            self.Y_b = 0;

            self.x_0 = [self.I_pob; self.Y_b];
        end


        function x_dot = dynamics(self,~,x,~,d)
            S_b = self.S_b;
            alpha = self.alpha;
            beta = self.beta;
            gamma = self.gamma;
            h = self.h;
            K = self.K;

            I_po = x(1);
            Y = x(2);
            G_dot = d(1);
            G = d(2);

            if G_dot > 0
                S_po = Y + K*G_dot + S_b;
            else
                S_po = Y + S_b;
            end

            if beta*(G-h) >= -S_b
                YIF = -alpha*(Y-beta*(G-h));
```

```
            else
                YIF = -alpha*Y-alpha*S_b;
            end
            x_dot = [-gamma*I_po+S_po
                     YIF
                     ];
        end

        function S = getS(self,I_po)
            gamma = self.gamma;
            S = gamma*I_po;
        end
    end
end
```

List F.23: insulinSignalCL

```
classdef insulinSignalCL
    properties
        k_i, p_2U, x_0, I_b, I_1b, I_db, X_b
    end
    methods
        function self = insulinSignalCL(subjectNum)
            self.k_i = readParam(subjectNum, 'k_i');
            self.p_2U = readParam(subjectNum, 'p_2U');
            self.I_b = readParam(subjectNum, 'I_b');

            self.I_1b = self.I_b;
            self.I_db = self.I_b;
            self.X_b = 0;

            self.x_0 = [self.I_1b; self.I_db; self.X_b];
        end

        function x_dot = dynamics(self, ~, x, ~, d)

            k_i = self.k_i;
            p_2U = self.p_2U;
            I_b = self.I_b;

            I_1 = x(1);
            I_d = x(2);
            X = x(3);
            I = d(1);

            x_dot = [ -k_i*(I_1-I)
                      -k_i*(I_d-I_1)
                      -p_2U*X+p_2U*(I-I_b)
                      ];
        end
    end
end
```

List F.24: insulinSystemCL

```
classdef (Abstract = true) insulinSystemCL
    properties
        m_1, m_2, m_4, HE_b, x_0, V_I, I_b, I_lb, I_pb
    end

    methods
        function self = insulinSystemCL(subjectNum)
            self.m_1 = readParam(subjectNum, 'm_1');
            self.m_2 = readParam(subjectNum, 'm_2');
            self.m_4 = readParam(subjectNum, 'm_4');
```

```
            self.HE_b = readParam(subjectNum, 'HE_b');
            self.V_I = readParam(subjectNum, 'V_I');
            self.I_b = readParam(subjectNum, 'I_b');
        end

        function I = getI(self, I_p)
            I = I_p/self.V_I;
        end
    end
end
```

List F.25: linearization

```
function [A_eq, Bu_eq, Bd_eq, C_eq, Du_eq, Dd_eq] = linearization(nonlinearModel)
    nx = nonlinearModel.nx;
    nu = nonlinearModel.nu;
    nd = nonlinearModel.nd;

    x_0 = nonlinearModel.x_0;
    u_0 = nonlinearModel.u_0;
    d_0 = nonlinearModel.d_0;

    syms x [nx, 1];
    syms u [nu, 1];
    syms d [nd, 1];

    % D は結局 tanh の中で Qsto/D という形ででてくるので，並行点ではQsto=0より，D の偏微分は 0
    x_dot = nonlinearModel.dynamics(0, x, u, d);
    y = nonlinearModel.output(0,x,u,d);

    % jacobian
    A = jacobian(x_dot, x);
    Bu = jacobian(x_dot, u);
    Bd = jacobian(x_dot, d);
    C = jacobian(y, x);
    Du = jacobian(y, u);
    Dd = jacobian(y, d);

    %%
    % equilibrium point
    A_eq = subs(A, [x;u;d], [x_0;u_0;d_0]);
    Bu_eq = subs(Bu, [x;u;d], [x_0;u_0;d_0]);
    Bd_eq = subs(Bd, [x;u;d], [x_0;u_0;d_0]);
    C_eq = subs(C, [x;u;d], [x_0;u_0;d_0]);
    Du_eq = subs(Du, [x;u;d], [x_0;u_0;d_0]);
    Dd_eq = subs(Dd, [x;u;d], [x_0;u_0;d_0]);
end
```

List F.26: linearModelCL

```
classdef linearModelCL
    properties
        attribute, subjectNum, nonlinearModel, meal, insulin, stomachD, ...
        Ac, Bc, Cc, Dc, Buc, Bdc, Duc, Ddc, nx, nu, nd, ny, y_0, x_0, u_0, d_0, V_G,
            R_aArray
    end
    methods
        function self = linearModelCL(model)
            self.nonlinearModel = model;
            self = self.setABCD;
            self.meal = model.meal;
            self.insulin = model.insulin;
            self.stomachD = model.stomachD;
            self.subjectNum = model.subjectNum;
```

```
            self.attribute = model.attribute;
            self.y_0 = model.y_0;
            self.x_0 = model.x_0;
            self.u_0 = model.u_0;
            self.d_0 = model.d_0;
            self.nx = model.nx; %状態の次元
            self.nu = model.nu; %入力の次元
            self.nd = model.nd; %外乱の次元
            self.ny = model.ny;
            self.V_G = model.V_G;
        end

        function self = setABCD(self)
            [Ac, Buc, Bdc, Cc, Duc, Ddc] = linearization(self.nonlinearModel);

            self.Ac = double(Ac);
            self.Buc = double(Buc);
            self.Bdc = double(Bdc);
            self.Bc = [self.Buc self.Bdc];
            self.Cc = double(Cc);
            self.Duc = double(Duc);
            self.Ddc = double(Ddc);
            self.Dc = [self.Duc self.Ddc];
        end

        function x_dot = dynamics(self, ~, x, u, d)
            Ac = self.Ac;
            Buc = self.Buc;
            Bdc = self.Bdc;

            x_tilda = (x - self.x_0);
            u_tilda = (u - self.u_0);
            d_tilda = (d - self.d_0);

            x_dot = Ac*x_tilda + Buc*u_tilda + Bdc*d_tilda;
        end

        function y = output(self, t, x, u, d)
            nonlinearModel = self.nonlinearModel;
            y = nonlinearModel.output(t, x, u, d);
        end
    end
end
```

List F.27: linearSwitchedModelCL

```
classdef linearSwitchedModelCL
    properties
        attribute, subjectNum, model1, model2, linearModel1, linearModel2, ...
            insulin, meal, stomachD, ...
            y_0, x_0, d_0, u_0, nx, nu, nd, ny, V_G, CR
    end

    methods
        function self = linearSwitchedModelCL(use)
            self.attribute = 'diabetes';
            self.subjectNum = readTable('simulationConfig','subjectNum');
            self.V_G = readParam(self.subjectNum, 'V_G');
            self.CR = readParam(self.subjectNum, 'CHOr');

            if strcmp(use,'mpc')
                self.model1 = magni2007KmaxCL('mpc');
                self.model2 = magni2007KminCL('mpc');
            elseif strcmp(use, 'normal')
                self.model1 = magni2007KmaxCL('normal');
                self.model2 = magni2007KminCL('normal');
            else
                error('mpc or normal')
```

```
            end

            self.linearModel1 = linearModelCL(self.model1);
            self.linearModel2 = linearModelCL(self.model2);

            self.meal = self.model1.meal;
            self.stomachD = self.model1.stomachD;
            self.insulin = self.model1.insulin;

            self.x_0 = self.model1.x_0;
            self.y_0 = self.model1.y_0;
            self.u_0 = self.model1.u_0;
            self.d_0 = self.model1.d_0;
        end

        function x_dot = dynamics(self,t,x,u,d,sw1,sw2)
            if x(10)+x(11) > sw2 || x(10)+x(11) < sw1
                x_dot = self.linearModel1.dynamics(t,x,u,d);
            elseif x(10)+x(11) >= sw1 && x(10)+x(11) <= sw2
                x_dot = self.linearModel2.dynamics(t,x,u,d);
            end
        end
    end
end
```

List F.28: magni2007bilinearCL

```
classdef magni2007bilinearCL
    properties
        attribute, subjectNum, glucoseSystem, insulinSystem, insulinInjection,
            insulinSignal, ...
            stomachD, glucoseAbsorption, meal, insulin, y_0, x_0, d_0, u_0, nx, nu, nd,
                ny, V_G, CR,
    end
    methods
        function self = magni2007bilinearCL(use)
            self.attribute = 'diabetes';
            self.subjectNum = readTable('simulationConfig','subjectNum');
            self.V_G = readParam(self.subjectNum, 'V_G');
            self.CR = readParam(self.subjectNum, 'CHOr');

            self.insulinSystem = insulinDiabetesApproximateSystemCL(self.subjectNum);
            self.insulinSignal = insulinSignalCL(self.subjectNum);
            self.insulinInjection = insulinInjectionCL(self.subjectNum, self.
                insulinSystem.ins_b_perMinKg);

            if strcmp(use,'mpc')
                self.glucoseSystem = bilinearGlucoseDiabetesSystemMpcCL(self.subjectNum);
                self.glucoseAbsorption = approximateGlucoseAbsorptionCL(self.subjectNum);
            else
                error('normal is not exist with this model')
            end

            self.meal = mealCL;
            self.insulin = insulinCL(self.insulinInjection.ins_b, self.CR);

            self.nx = 9;
            self.nu = 1;
            self.nd = 1;
            self.ny = 1;

            self.x_0 = [self.glucoseSystem.x_0
                        self.insulinSystem.x_0
                        self.insulinSignal.x_0
                        self.insulinInjection.x_0
                        self.glucoseAbsorption.x_0
                       ];
            self.y_0 = self.x_0(1)/self.V_G;
```

```
            self.u_0 = self.insulinInjection.ins_b;
            self.d_0 = self.meal.d_0;
        end

        function x_dot = dynamics(self,t,x,u,d)
            G_p = x(1);
            G_t = x(2);
            I_p = x(3);
            I_1 = x(4);
            I_d = x(5);
            X = x(6);
            S_1 = x(7);
            S_2 = x(8);
            Q_gut = x(9);

            insInput = u(1);
            d_in = d(1);

            Ra = self.glucoseAbsorption.getR_a(Q_gut);
            I = self.insulinSystem.getI(I_p);
            S = self.insulinInjection.getS([S_1;S_2]);

            G_dot = self.glucoseSystem.dynamics(t,[G_p;G_t],0,[Ra; X; I_d]);
            I_dot = self.insulinSystem.dynamics(t, I_p, 0, S);
            Is_dot = self.insulinSignal.dynamics(t, [I_1; I_d; X], 0, I);
            S_dot = self.insulinInjection.dynamics(t, [S_1;S_2],insInput,0);
            Q_dot = self.glucoseAbsorption.dynamics(t,Q_gut,0,d_in);

            x_dot = [G_dot; I_dot; Is_dot; S_dot; Q_dot];
        end

        % 出力変数
        function y = output(self,~,x,~,~)
             y = x(1)/self.V_G;
        end
    end
end
```

List F.29: magni2007CL

```
classdef magni2007CL
    properties
        attribute, subjectNum, glucoseSystem, insulinSystem, insulinInjection,
             insulinSignal, ...
            stomachD, glucoseAbsorption, meal, insulin, y_0, x_0, d_0, u_0, nx, nu, nd,
                ny, V_G, CR,
    end
    methods
        function self = magni2007CL(use)
            self.attribute = 'diabetes';
            self.subjectNum = readTable('simulationConfig','subjectNum');
            self.V_G = readParam(self.subjectNum, 'V_G');
            self.CR = readParam(self.subjectNum, 'CHOr');

            self.insulinSystem = insulinDiabetesSystemCL(self.subjectNum);
            self.insulinSignal = insulinSignalCL(self.subjectNum);
            self.insulinInjection = insulinInjectionCL(self.subjectNum, self.
                insulinSystem.ins_b_perMinKg);

            % mpc の場合は,
            if strcmp(use,'mpc')
                self.glucoseSystem = glucoseDiabetesSystemMpcCL(self.subjectNum);
                self.glucoseAbsorption = glucoseAbsorptionMpcCL(self.subjectNum);
            elseif strcmp(use, 'normal')
                self.glucoseSystem = glucoseDiabetesSystemCL(self.subjectNum);
                self.glucoseAbsorption = glucoseAbsorptionCL(self.subjectNum);
            else
```

```
                error('mpc or normal')
            end

            self.meal = mealCL;
            self.stomachD = stomachDCL(self.glucoseAbsorption, self.meal);
            self.insulin = insulinCL(self.insulinInjection.ins_b, self.CR);

            self.nx = 12;
            self.nu = 1;
            self.nd = 2;
            self.ny = 1;

            self.x_0 = [self.glucoseSystem.x_0
                        self.insulinSystem.x_0
                        self.insulinSignal.x_0
                        self.insulinInjection.x_0
                        self.glucoseAbsorption.x_0
                       ];
            self.y_0 = self.x_0(1)/self.V_G;
            self.u_0 = self.insulinInjection.ins_b;
            self.d_0 = [self.meal.d_0
                        self.stomachD.D_0
                       ];
        end

        % dynamics equation
        function x_dot = dynamics(self,t,x,u,d)
            G_p = x(1);
            G_t = x(2);
            I_l = x(3);
            I_p = x(4);
            I_1 = x(5);
            I_d = x(6);
            X = x(7);
            S_1 = x(8);
            S_2 = x(9);
            Q_sto1 = x(10);
            Q_sto2 = x(11);
            Q_gut = x(12);

            insInput = u(1);
            d_in = d(1);
            D = d(2);

            Ra = self.glucoseAbsorption.getR_a(Q_gut);
            I = self.insulinSystem.getI(I_p);
            S = self.insulinInjection.getS([S_1;S_2]);

            G_dot = self.glucoseSystem.dynamics(t,[G_p;G_t],0,[Ra; X; I_d]);
            I_dot = self.insulinSystem.dynamics(t, [I_l;I_p], 0, S);
            Is_dot = self.insulinSignal.dynamics(t, [I_1; I_d; X], 0, I);
            S_dot = self.insulinInjection.dynamics(t, [S_1;S_2],insInput,0);
            Q_dot = self.glucoseAbsorption.dynamics(t,[Q_sto1; Q_sto2; Q_gut],0,[d_in; D
                ]);

            x_dot = [G_dot; I_dot; Is_dot; S_dot; Q_dot];
        end

        % output equation
        function y = output(self,~,x,~,~)
             y = x(1)/self.V_G;
        end
    end
end
```

List F.30: magni2007KmaxCL

```
% k_gut を，k_max に変更
classdef magni2007KmaxCL < magni2007CL
    methods
        function self = magni2007KmaxCL(~)
            self@magni2007CL('mpc')
            self.glucoseAbsorption = glucoseAbsorptionKmaxCL(self.subjectNum);
        end
    end
end
```

List F.31: magni2007KminCL

```
% k_gut を，k_min に変更
classdef magni2007KminCL < magni2007CL
    methods
        function self = magni2007KminCL(~)
            self@magni2007CL('mpc')
            self.glucoseAbsorption = glucoseAbsorptionKminCL(self.subjectNum);
        end
    end
end
```

List F.32: magni2007KminmaxCL

```
% k_gut を，k_min に変更

classdef magni2007KminmaxCL < magni2007CL
    methods
        function self = magni2007KminmaxCL(~)
            self@magni2007CL('mpc')
            self.glucoseAbsorption = glucoseAbsorptionKminmaxCL(self.subjectNum);
        end
    end
end
```

List F.33: man2007CL

```
classdef man2007CL
    properties
        attribute, subjectNum, glucoseSystem, insulinSystem, insulinSecretion,
             insulinSignal, ...
            stomachD, glucoseAbsorption, meal, insulin, y_0, x_0, d_0, u_0, nx, nu, nd,
                ny, V_G,
    end
    methods
        function self = man2007CL(~)
            self.attribute = 'normal';
            self.subjectNum = 0; % normal パラメータはsubjectNum = 0
            self.V_G = readParam(self.subjectNum, 'V_G');

            self.insulinSystem = insulinHealthySystem(self.subjectNum);
            self.insulinSignal = insulinSignalCL(self.subjectNum);
            self.insulinSecretion = insulinSecretionCL(self.subjectNum);
            self.glucoseSystem = glucoseHealthySystemCL(self.subjectNum);
            self.glucoseAbsorption = glucoseAbsorptionCL(self.subjectNum);

            self.meal = mealCL;
            self.stomachD = stomachDCL(self.glucoseAbsorption, self.meal);

            self.nx = 12;
            self.nu = 0;
```

```
            self.nd = 2;
            self.ny = 1;

            self.x_0 = [self.glucoseSystem.x_0
                        self.insulinSystem.x_0
                        self.insulinSignal.x_0
                        self.insulinSecretion.x_0
                        self.glucoseAbsorption.x_0
                        ];

            self.y_0 = self.x_0(1)/self.V_G;
            self.d_0 = [self.meal.d_0
                        self.stomachD.D_0
                       ];
        end

        function x_dot = dynamics(self, t, x, ~, d)
            G_p = x(1);
            G_t = x(2);
            I_l = x(3);
            I_p = x(4);
            I_1 = x(5);
            I_d = x(6);
            X = x(7);
            I_po = x(8);
            Y = x(9);
            Q_sto1 = x(10);
            Q_sto2 = x(11);
            Q_gut = x(12);

            d_in = d(1);
            D = d(2);

            Ra = self.glucoseAbsorption.getR_a(Q_gut);
            I = self.insulinSystem.getI(I_p);
            G = self.glucoseSystem.getG(G_p);
            S = self.insulinSecretion.getS(I_po);

            G_dot = self.glucoseSystem.dynamics(t,[G_p; G_t], 0, [Ra; X; I_d; I_po]);
            I_dot = self.insulinSystem.dynamics(t,[I_l; I_p], 0, S);
            Is_dot = self.insulinSignal.dynamics(t, [I_1; I_d; X], 0, I);
            S_dot = self.insulinSecretion.dynamics(t, [I_po; Y], 0, [G_dot(1)/self.V_G; G
                ]);
            Q_dot = self.glucoseAbsorption.dynamics(t,[Q_sto1; Q_sto2; Q_gut],0,[d_in; D
                ]);

            x_dot = [G_dot; I_dot; Is_dot; S_dot; Q_dot];
        end

        function y = output(self,~,x,~,~)
            y = x(1)/self.V_G;
        end
    end
end
```

List F.34: mealCL

```
% 時間依存の入力の設定,
% insulinArray = [[入力開始時刻, 入力時間, 入力量]; ... [];];
% 重複する場合は, 前の入力が優先される
% 食事量の単位は [mg]
% input の単位は[mg/min]

classdef mealCL < timeDeterminedInputCL
    properties
        d_0
    end
```

```
    methods
        function self = mealCL()
            self@timeDeterminedInputCL()
            self.d_0 = 0;
            self.inputArray = self.makeInputArray('meal');
        end

        function mealInput = getMealInput(self, t)
            determinedMealInput = self.getDeterminedInput(t);
            mealInput = determinedMealInput;
        end
    end
end
```

List F.35: readInputTable

```
function array = readInputTable(sheet)
    value = readtable('nominalInput.xlsx','sheet',sheet);
    tableValue = table2array(value);
    array = tableValue(:,2:4);
end
```

List F.36: readParam

```
function parameter = readParam(subjectNum, parameterName)
        value = readtable('T1DMparameter.xlsx');
        eval(['parameterArray = value.s',num2str(subjectNum),';']);
        parameterNameArray = value.Parameter;
        paramIndex = find(strcmp(parameterName, parameterNameArray));
        parameter = parameterArray(paramIndex);
        if isempty(parameter)
            error([parameterName,'という名前のパラメータは存在しません'])
        end
end
```

List F.37: readTable

```
function value = readTable(sheet, label)
    sheetValue = readtable('nominalInput.xlsx','sheet',sheet);
    eval(['value = sheetValue.',label,';']);
    if iscell(value)
        value = cell2mat(value);
    else
        findValue = find(~isnan(value));
        value = value(findValue);
    end
end
```

List F.38: stomachDCL

```
classdef stomachDCL < timeDeterminedInputCL
    properties
        D, D_0, beforePhaseNum, glucoseAbsorption, q_0, stomachDArray, meal,
    end
    methods
        function self = stomachDCL(glucoseAbsorption, meal)
            self@timeDeterminedInputCL();
            self.inputArray = self.makeInputArray('meal');
```

```
            self.glucoseAbsorption = glucoseAbsorption;
            self.q_0 = self.glucoseAbsorption.x_0;
            self.D_0 = 45000;
            self.meal = meal;
            self.stomachDArray = self.makeStomachDArray;
        end

        %
        function stomachDArray = makeStomachDArray(self, ~)
            inputArray = self.inputArray;
            q_0 = self.q_0;

            stomachDArray = [];
            for phaseNum = 1:size(inputArray,1)
                D = q_0(1)+q_0(2) + inputArray(phaseNum,3);

                if phaseNum == size(inputArray,1)
                   stomachDArray = [stomachDArray;
                                inputArray(phaseNum,1), self.t_end-inputArray(phaseNum,1),
                                    D];
                    break
                else
                   stomachDArray = [stomachDArray;
                                inputArray(phaseNum,1), inputArray(phaseNum+1,1)-
                                    inputArray(phaseNum,1),D];
                end

                t1 = inputArray(phaseNum,1);
                t2 = inputArray(phaseNum+1,1);
                options = odeset('MaxStep', 0.5, 'RelTol',1e-12);
                [t, q] = ode45(@(t,q)self.glucoseAbsorption.dynamics(t,q,0,[self.meal.
                    getMealInput(t),D]),[t1,t2],q_0, options);
                q_0 = q(end,:);
            end
        end

        % timeDetermindInputCL の getDeterminedInput をoveride
        % stomachD 用の getDeterminedInput
        % inputArray(i, 2)で割らない
        function input = getDeterminedInput(self, t)
            inputArray = self.stomachDArray;
            input = self.D_0;
            for i = 1:size(inputArray,1)
                if t >= inputArray(i,1) && t < inputArray(i,1) + inputArray(i,2)
                    input = inputArray(i,3);
                    break
                end
            end
        end

        function D = getDInput(self, t)
            D = self.getDeterminedInput(t);
        end

    end
end
```

List F.39: timeDeterminedInputCL

```
% 時間依存の入力の設定,
% inputArray = [[入力開始時刻, 入力時間, 入力量]; ... [];];
% 重複する場合は, 前の入力が優先される
% 入力時間等はサンプリング時刻に一致させること

classdef timeDeterminedInputCL
    properties
        inputArray, t_end
```

```
    end
    methods
        function self = timeDeterminedInputCL()
            self.t_end = readTable('simulationConfig','t_end');
        end
        function input = getDeterminedInput(self, t)
            inputArray = self.inputArray;
            input = 0;
            for i = 1:size(inputArray,1)
                if t >= inputArray(i,1) && t < inputArray(i,1) + inputArray(i,2)
                    input = inputArray(i,3) / inputArray(i,2);
                    break
                end
            end
        end

        % makeInputArray
        % kind : meal, insulin, (leptin, glucagon ...)
        % regularTF, irregularTF : irregular や regular のどちらを使うか

        function inputArray = makeInputArray(self, kind)

            regularTF = readTable('inputConfig','regularTF');
            irregularTF = readTable('inputConfig','irregularTF');

            regularInputArray = readInputTable(['regular',kind]);
            regularInputArray(:,1) = regularInputArray(:,1)*60;
            irregularInputArray = readInputTable(['irregular',kind]);
            irregularInputArray(:,1) = irregularInputArray(:,1)*60;


            % インプットが 0, の場合ははじく.
            if ~isempty(regularInputArray)
            zeroInputIndex = find(regularInputArray(:,3)==0);
            regularInputArray(zeroInputIndex,:) = [];
            end

            if ~isempty(irregularInputArray)
            zeroInputIndex = find(irregularInputArray(:,3)==0);
            irregularInputArray(zeroInputIndex,:) = [];
            end

            inputArray = [];
            for j = 1:size(regularInputArray,1)
                simulationDays = floor(self.t_end/(60*24));
                last_t_end = self.t_end-simulationDays*(60*24);

                for i = 1:simulationDays
                    inputArray = [inputArray;
                        regularInputArray(j,1)+24*60*(i-1), regularInputArray(j,2:3)];
                end

                if regularInputArray(j,1) < last_t_end
                    inputArray = [inputArray;
                        regularInputArray(j,1)+24*60*simulationDays, regularInputArray(j
                            ,2:3)];

                end
            end

            index = find(irregularInputArray(:,1) > self.t_end);
            irregularInputArray(index,:) = [];

            if irregularTF == true && regularTF == true
                inputArray = [inputArray; irregularInputArray];
            elseif irregularTF == false && regularTF == true
                inputArray = [inputArray];
            elseif irregularTF == true && regularTF == false
                inputArray = [irregularInputArray];
            else
```

```
                error('regularTF か irregularTF のどちらかは true にしてください')
            end
            inputArray = sortrows(inputArray);

            for i = 1:size(inputArray,1)-1
                if inputArray(i,1)+inputArray(i,2) > inputArray(i+1,1)
                    error('入力の範囲が重なっているようです．訂正してください')
                else
                    if inputArray(end,1)+inputArray(end,2) > self.t_end
                        inputArray(end,2) = self.t_end - inputArray(end,1);
                    end
                end
            end
        end
    end
end
```

謝辞

大学での研究生活において, 終始暖かいご指導を賜り，対外的な研究活動に際しても多くの機会を下さりました 富山大学 平田 研二 先生に心より感謝の意を表します.

シミュレーション実装や計算環境構築をはじめ，多方面で研究指導をして頂きました富山大学 Tam W. Nguyen 先生に厚く御礼申し上げます.

本研究にあたり，複数回にわたるミーティングにご参加くださり，数多くの助言や情報提供をして頂きました早稲田大学 内田 健康 先生，和佐 泰明 先生に深く感謝申し上げます.

また, 日々の研究生活で多くのご協力を頂きました研究室の先輩方，後輩方，学友の皆様に心から感謝いたします.

最後に, これまでの学生生活を支え, いつも暖かく見守って頂いた父母や祖父母をはじめとする多くの皆様に心より感謝いたします．本当にありがとうございました.

2023 年 3 月