Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Tomasz Kanas
Nr albumu: 385674

Optymalna strategia podawania leku

Praca licencjacka
na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
dr. Piotr Krzyżanowski
Instytut Matematyki Stosowanej i Mechaniki

Warszawa, Wrzesień 2020

Streszczenie

W pracy przedstawiono metodę numerycznego wyznaczania przybliżenia optymalnej strategii podawania leku przy leczeniu nowotworu. Sporządzono i opisano implementację metody w środowisku Octave oraz przeprowadzono i omówiono wyniki eksperymentów.

Słowa kluczowe

teoria sterowania, programowanie nieliniowe, nowotwór, Octave

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.1 Matematyka

Klasyfikacja tematyczna

49-XX Calculus of variations and optimal control; optimization
49M37 Numerical methods based on nonlinear programming

Tytuł pracy w języku angielskim

Optimal strategy of drug dosage

Spis treści

Wprowadzenie

Nowotwory są jednym z największych problemów współczesnej medycyny. Przyczyniają się one do bardzo wielu zgonów na całym świecie. Proces leczenia nowotworu nie zawsze jest skuteczny oraz potrafi być bardzo wyniszczający dla organizmu pacjenta, dlatego tak istotne jest, by terapia prowadzona była w jak najlepszy sposób.

W tej pracy przedstawimy metodę przybliżonego rozwiązania pewnego modelu przedstawiającego zależność między strategią podawania leku, a przebiegiem i skutecznością terapii. Celem naszym będzie znalezienie strategii podawania leku optymalizującą, podaną w tym modelu, miarę skuteczności terapii.

Do rozwiązania tego zadania sformułujemy je jako zadanie optymalizacji nieliniowej z ograniczeniami. Zaprojektujemy i zaimplementujemy algorytm poszukujący wspomnianej optymalnej strategii podawania leku. Przeprowadzimy też eksperymenty badające poprawność i skuteczność naszej implementacji, oraz wpływ jaki mają na nią rozmaite parametry wynikające z zastosowanej metody. Na końcu przeprowadzamy dyskusję uzyskanych wyników oraz użytych metod.

Rozdział 1

Sformułowanie problemu

Celem pracy jest znalezienie strategii podawania leku przy leczeniu nowotworu, pozwalającej osiągnąć możliwie największą skuteczność terapii. W tym celu skorzystamy z modelu przedstawionego w pracy [1] (zobacz też [2], [3] gdzie przedstawiono podobne zagadnienia). Model ten przedstawia rozwój nowotworu w czasie w zależności od dawkowania leku, $g(t)$, za pomocą równania różniczkowego:

$$\begin{aligned} V_1'(t) &= \lambda_1 V_1 F\left(\frac{V_1 + \alpha_{12} V_2}{K}\right) - \beta_1 V_1 g(t), \\ V_2'(t) &= \lambda_2 V_2 F\left(\frac{V_2 + \alpha_{21} V_1}{K}\right) - \beta_2 V_2 g(t), \\ K'(t) &= -\mu K + (V_1 + V_2) - d(V_1 + V_2)^{2/3} K - \beta K g(t) \end{aligned} \tag{1.1}$$

dla $t \in [0, T]$, z warunkami początkowymi

$$V_1(0) = V_{10},\ V_2(0) = V_{20},\ K(0) = K_0 \tag{1.2}$$

gdzie $F(x) = -\ln(x)$, $0 \le g(t) \le g_{\max}$, oraz

$$\lambda_1, \lambda_2, \alpha_{12}, \alpha_{21}, \beta_1, \beta_2, \beta, \mu, d, V_{10}, V_{20}, K_0$$

są zadanymi nieujemnymi parametrami.

Funkcja $V_1(t)$ modeluje liczbę komórek guza podatnych na lek w momencie t, $V_2(t)$ liczbę komórek guza odpornych na lek, a $K(t)$ jest parametrem nazwanym w pracy „unaczynieniem". Zauważmy, że rozwiązania V_1, V_2, K zależą od wyboru funkcji g którą nazywamy sterowaniem. W tym modelu wartość $g(t)$ ma interpretację jako wielkość dawki leku w czasie t.

Zgodnie z [1], zadanie polega na znalezieniu funkcji $g : [0, T] \to [0, g_{\max}]$ takiej, że funkcje $V_1, V_2, K : [0, T] \to (0, \infty)$ spełniające (1.1) minimalizują funkcjonał:

$$J(g, V_1, V_2, K) = \int_0^T V_1(t) + V_2(t) dt + \omega \int_0^T G\left(\frac{V_2(t) - V_1(t)}{\epsilon}\right) dt \tag{1.3}$$

gdzie

$$G(x) = \frac{1 + \tanh(x)}{2} \quad \omega, \epsilon > 0$$

Problem ten w literaturze (np. [5], [9]) nazywa się problemem optymalnego sterowania.

1.1. Problem wyjściowy

Zdefiniujmy teraz formalnie problem oraz uprośćmy notację.

Problem 1. *Znaleźć funkcję kawałkami ciągłą*

$$g : [0, T] \to [0, g_{\max}],$$

i funkcję

$$y = (y_1, y_2, y_3)^T : [0, T] \to (0, \infty)^3,$$

spełniającą równanie różniczkowe:

$$\begin{aligned} \dot{y}(t) &= f(t, y, g), \\ y(0) &= y_0 = (y_{10}, y_{20}, y_{30})^T, \end{aligned} \tag{1.4}$$

minimalizujące funkcjonał

$$J(g, y) = \int_0^T y_1(t) + y_2(t) dt + \omega \int_0^T G\left(\frac{y_2(t) - y_1(t)}{\epsilon}\right) dt, \tag{1.5}$$

gdzie $f = (f_1, f_2, f_3)^T$ *jest określone wzorem*

$$\begin{aligned} f_1(t, y, g) &= \lambda_1 y_1 F\left(\frac{y_1 + \alpha_{12} y_2}{y_3}\right) - \beta_1 y_1 g(t), \\ f_2(t, y, g) &= \lambda_2 y_2 F\left(\frac{y_2 + \alpha_{21} y_1}{y_3}\right) - \beta_2 y_2 g(t), \\ f_3(t, y, g) &= -\mu y_3 + (y_1 + y_2) - d(y_1 + y_2)^{2/3} y_3 - \beta y_3 g(t). \end{aligned} \tag{1.6}$$

Przez funkcję kawałkami ciągłą określoną na odcinku rozumiemy funkcję o skończonej liczbie punktów nieciągłości. Jako, że o funkcji f zakładamy tylko kawałkami ciągłość, należy doprecyzować co rozumiemy przez (1.4). Załóżmy, że punktami nieciągłości $f(t, y, g) : \mathbb{R}^5 \to \mathbb{R}$ są $t = \xi_1, \ \ldots, \ \xi_n \in \mathbb{R}$, wtedy (1.4) oznacza sekwencję równań różniczkowych postaci

$$\begin{aligned} \dot{y}|_{(\xi_i, \xi_{i+1})}(t) &= f|_{(\xi_i, \xi_{i+1})}(t, y, g) \\ y(\xi_i) &= \lim_{t \to \xi_i^-} y(t) \end{aligned} \qquad \text{gdzie } i \in \{0, \ldots, n\}, \ \xi_0 = 0, \ \xi_{n+1} = T \tag{1.7}$$

Zwróćmy jeszcze uwagę na fakt, że funkcja $F(x) = -\ln(x)$ posiada osobliwość w 0, więc prawa strona (1.6) nie jest dobrze zdefiniowana dla $y_1(t) = y_2(t) = 0$, a obliczenia w których argumenty F są bliskie 0 mogą być obarczone dużymi błędami numerycznymi. Podobnie we wzorze (1.6) występuje dzielenie przez y_3, więc dla $y_3(t) = 0$ prawa strona (1.4) także nie jest dobrze określona.

1.2. Problem przybliżony

Wyznaczenie rozwiązania problemu optymalnego sterowania w postaci jawnego wzoru rzadko kiedy jest możliwe. Z tego powodu zdecydujemy się na szukanie rozwiązania przybliżonego.

Rozwiązanie Problemu 1 przybliżymy rozwiązaniem pewnego problemu optymalizacji skończenie wymiarowej. W tym celu ustalimy siatkę dyskretyzacji przedziału $[0, T]$:

$$0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T \tag{1.8}$$

Możemy teraz przybliżać sterowanie g za pomocą splajnu opartego na punktach t_i. Dla prostoty ograniczymy się do splajnów stopnia 0 i 1. Ostatecznie przybliżone sterowanie $\hat{g}$ ma postać:

$$\hat{g}(t) = g_i \text{ gdy } t \in [t_i, t_{i+1}) \quad \text{(sterowanie kawałkami stałe)} \tag{1.9}$$

albo

$$\hat{g}(t) = \frac{(t_{i+1} - t)g_i + (t - t_i)g_{i+1}}{t_{i+1} - t_i} \text{ gdy } t \in [t_i, t_{i+1}) \quad \text{(sterowanie kawałkami liniowe)} \tag{1.10}$$

i jest jednoznacznie zdefiniowane przez wartości $g_i \simeq g(t_i)$ dla $i = 0, \ldots, n$. Te wartości będą optymalizowanymi zmiennymi.

Korzystając z przybliżonej funkcji sterowania, na każdym odcinku $[t_n, t_{n+1}]$ przybliżymy rozwiązanie y równania różniczkowego (1.4). Użyjemy do tego M kroków metody Rungego-Kutty rzędu r ze stałym krokiem długości $h = \frac{t_{n+1} - t_n}{M}$. Wtedy przybliżone rozwiązanie $\hat{y}(t) \simeq y(t)$ w punkcie $t = t_n + (m+1)h$ dla $m = 0, \ldots, M$ wyraża się przez:

$$\begin{aligned} k_1 &= f(t_n + mh, \hat{y}(t_n + mh), \hat{g}), \\ k_l &= f(t_n + c_l h, \hat{y}(t_n + mh) + h\sum_{i=1}^{l-1} a_{li}k_i, \hat{g}), \quad l = 2, \ldots, r \\ \hat{y}(t_n + (m+1)h) &= \hat{y}(t_n + mh) + h\sum_{i=1}^{r} b_i k_i, \end{aligned} \tag{1.11}$$

gdzie c_l, a_{li}, b_i są stałymi zależnymi od wybranej metody.

Zostało już tylko przybliżyć funkcjonał celu (1.5). Zapiszmy go w postaci

$$J(y) = \int_0^T j(y(t))dt, \tag{1.12}$$

gdzie

$$j(y) = y_1 + y_2 + \omega G\left(\frac{y_2 - y_1}{\epsilon}\right). \tag{1.13}$$

Wtedy ogólny wzór na kwadraturę ze stałym krokiem h przybliżającą (1.12) to

$$Q(\hat{y}) = h\sum_{i=0}^{N} \alpha_i j(\hat{y}(ih)), \tag{1.14}$$

gdzie $N = \frac{T}{h}$, a α_i są stałymi zależnymi od wybranej kwadratury.

Zatem możemy zdefiniować przybliżoną funkcję celu jako funkcję $g_0, \ldots, g_n$:

$$\hat{J}(g_0, \ldots, g_n) = Q(\hat{y}), \tag{1.15}$$

ponieważ $\hat{y}$ jest jednoznacznie wyznaczony przez $\hat{g}$ za pomocą (1.11), a $\hat{g}$ jest wyznaczone jednoznacznie przez $g_0, \ldots, g_n$. Podsumowując, problem przybliżony to:

Problem 2. *Znaleźć $g_0, \ldots, g_n$ minimalizujące*

$$\hat{J}(g_0, \ldots, g_n), \tag{1.16}$$

i spełniające

$$\forall_{i \in \{0,\ldots,n\}} 0 \le g_i \le g_{\max}. \tag{1.17}$$

Jest to problem optymalizacji nieliniowej z ograniczeniami i istnieją implementacje metod pozwalających uzyskać przybliżone rozwiązanie tego problemu. To podejście do numerycznego problemu optymalnego nazywa się „direct single shooting" i występuje np. w [5] i [9].

1.3. Alternatywne podejście

Innym popularnym podejściem jest „direct collocation". W tym podejściu zaczyna się, tak samo jak powyżej, od dyskretyzacji czasu:

$$0 = t_0 < t_1 < \ldots < t_n = T. \tag{1.18}$$

Następnie zarówno sterowanie jak i stan układu przybliża się splajnem opartym na punktach t_i. Dzięki temu sterowanie jest jednoznacznie wyznaczone przez wartości $g_i = \hat{g}(t_i)$, natomiast stan układu przez wartości $y_i = \hat{y}(t_i)$. Celem jest, tak samo jak powyżej, sprowadzenie problemu do problemu optymalizacji nieliniowej z ograniczeniami. Aby móc to zrobić należy znaleźć warunki jakie wartości y_i muszą spełniać, aby układ spełniał w przybliżeniu równanie

$$\dot{y}(t) = f(t, y(t), g(t)). \tag{1.19}$$

Aby znaleźć te warunki przybliża się całkę z powyższego równania za pomocą pewnej kwadratury. Dla przykładu ustalmy kwadraturę trapezową. Wtedy dostajemy równanie przybliżone

$$\int_{t_i}^{t_{i+1}} \dot{y}(t)dt = \int_{t_i}^{t_{i+1}} f(t, y(t), g(t))dt \tag{1.20}$$

$$y_{i+1} - y_i \simeq \frac{1}{2}(t_{i+1} - t_i)(f(t_{i+1}, y_{i+1}, g(t_{i+1})) - f(t_i, y_i, g(t_i))). \tag{1.21}$$

Powyższy wzór dodaje się do ograniczeń optymalizatora jako ograniczenie równościowe. W ten sposób RRZ (1.19) zostaje w przybliżeniu wymuszone w rozwiązaniu.

Ograniczenia występujące w sformułowaniu oryginalnego problemu można zwykle bez problemu przekształcić na ograniczenia g_i oraz y_i. Przybliżony funkcjonał celu definiuje się, jako numeryczne przybliżenie oryginalnego funkcjonału celu zastosowane do przybliżonego stanu $\hat{y}$ i sterowania $\hat{g}$.

Podejście to jest dokładnie opisane w [7], występuje też w [5] i [9].

1.4. Plan rozwiązania

Aby obliczyć wynik problemu przybliżonego zdefiniowanego w rozdz. 1.2 zaimplementujemy przejście od problemu optymalnego sterowania do problemu optymalizacji nieliniowej z ograniczeniami. W tym celu skorzystamy ze środowiska MATLAB/Octave i dostarczonego w nim optymalizatora (FMINICON). Aby poprawić złożoność czasową optymalizacji i tym samym umożliwić stosowanie gęstszej siatki dyskretyzacji, obliczymy gradient przybliżonej funkcji celu. Będziemy też musieli znaleźć odpowiednią siatkę dyskretyzacji i punkt startowy dla optymalizatora. Należy też pamiętać, że dopuszczamy możliwość nieciągłości w sterowaniu, co może być przyczyną błędów przybliżenia i problemów ze zbieżnością optymalizacji. Zaimplementujemy rozwiązanie w taki sposób, aby ograniczyć problemy płynące z możliwej nieciągłości.

Rozdział 2

Implementacja

Optymalizator FMINICON wymaga dostarczenia funkcji do optymalizowania, czyli w naszym przypadku funkcji $\hat{J}$, oraz ograniczeń, czyli w przypadku metody „direct single shooting" jedynie (1.17). Domyślnie optymalizator wyznacza gradient za pomocą różnic skończonych, co jest kosztowne, gdyż wymaga uruchomienia funkcji celu liniowo wiele razy względem liczby parametrów. Aby poprawić wydajność optymalizacji zaimplementujemy liczenie gradientu przybliżonej funkcji celu ze względu na parametry $g_0, \ldots, g_n$.

Zauważmy, że znalezienie dobrego wyniku będzie zapewne wymagało wielokrotnego wywołania naszej funkcji celu przez optymalizator, więc zależy nam aby funkcja celu liczyła się możliwie szybko. Lepsza wydajnościowo implementacja pozwoli też na większe zagęszczenie siatki dyskretyzacji i tym samym wzrost dokładności aproksymacji.

Jedną z praktyk pozwalającą poprawić wydajność programów w środowiskach MATLAB i Octave jest tak zwana wektoryzacja. Polega ona na zastępowaniu pętli operacjami na wektorach. Korzysta to z faktu, że wiele funkcji w tych środowiskach można wywołać z wektorem parametrów, zamiast pojedynczego parametru i zwracają one wtedy wektor wyników, oraz wykonują się znacznie szybciej niż gdyby wywołać je wielokrotnie w pętli, z pojedyńczymi argumentami. Z tego powodu będziemy korzystać z tej techniki gdzie to tylko możliwe.

Aby sprowadzić problem optymalnego sterowania do problemu optymalizacji nieliniowej musimy obliczyć przybliżone sterowanie (1.9) lub (1.10), przybliżyć rozwiązanie równania różniczkowego (1.11) a następnie za pomocą uzyskanego rozwiązania obliczyć kwadraturę (1.14) przybliżającą funkcję celu (1.3). Ponadto chcemy obliczyć gradient funkcji celu, co będzie wymagało policzenia pochodnej każdego z wymienionych wzorów względem parametrów g_i.

2.1. Sterowanie

Jedynym problemem przy implementacji przybliżonego sterowania (1.9) lub (1.10) jest znalezienie indeksu i takiego, że $t \in [t_{i-1}, t_i)$. Użyjemy do tego funkcji *lookup*, która wydajnie znajduje żądany indeks korzystając z wyszukiwania binarnego.

Aby policzyć gradient przybliżonego sterowania $\hat{g}$ ze względu na parametry $g_1, \ldots, g_n$ należy zróżniczkować równanie (1.9) lub (1.10):

$$\frac{\partial \hat{g}}{\partial g_i}(t) = \begin{cases} 1 \text{ gdy } t \in [t_{i-1}, t_i) \\ 0 \text{ w.p.p.} \end{cases} \tag{2.1}$$

albo

$$\frac{\partial \hat{g}}{\partial g_i}(t) = \begin{cases} \frac{t_i - t}{t_i - t_{i-1}} \text{ gdy } t \in [t_{i-1}, t_i) \\ \frac{t - t_i}{t_{i+1} - t_i} \text{ gdy } t \in [t_i, t_{i+1}) \\ 0 \text{ w.p.p.} \end{cases} \tag{2.2}$$

Implementacja powyższych wzorów jest standardowa, znajdowanie odpowiedniego indeksu i odbywa się tak samo, jak przy liczeniu wartości przybliżonego sterowania.

Zarówno funkcja *lookup* jak i pozostałe operacje wykorzystywane do policzenia przybliżonego sterowania i jego gradientu są zwektoryzowane. W szczególności, środowiska MATLAB i Octave umożliwiają wektorowe indeksowanie, więc nasza implementacja sterowania i jego gradientu też jest zwektoryzowana.

2.2. Równanie różniczkowe

Aby przybliżyć rozwiązanie równania (1.1) należy zaimplementować funkcję $f(t, y, g)$, a następnie bezpośrednio zaimplementować odpowiednią metodę Rungego-Kutty (1.11). Wzory te implementuje się bezpośrednio.

Liczenie pochodnych $\frac{\partial \hat{y}}{\partial g_i}$ polega na zróżniczkowaniu wzorów (1.11):

$$\begin{gathered} \frac{\partial k_1}{\partial g_i} = D_f \cdot \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + \frac{\partial f}{\partial g_i}(t_n + mh, \hat{y}(t_n + mh), \hat{g}) \\ \frac{\partial k_l}{\partial g_i} = D_f \cdot \left(\frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + h \sum_{j=1}^{l-1} a_{lj} \frac{\partial k_i}{\partial g_j} \right) + \frac{\partial f}{\partial g_i}(t_n + mh, \hat{y}(t_n + mh), \hat{g}) \\ \frac{\partial \hat{y}}{\partial g_i}(t_n + (m+1)h) = \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + h \sum_{j=1}^{r} b_j \frac{\partial k_j}{\partial g_i} \end{gathered} \tag{2.3}$$

gdzie

$$D_f = \left[\frac{\partial f_i}{\partial y_j} \right]_{i,j=1,\ldots,3} (t_n + ah, \hat{y}(t_n + mh), \hat{g}) \tag{2.4}$$

Wzory na elementy macierzy D_f pominiemy, liczy się je i implementuje standardowo.

Nasza implementacja umożliwia też przekazanie wektora punktów w czasie jako argumentu. W takim przypadku wyliczane wartości w podanych punktach będą spamiętywane w trakcie optymalizacji i zwrócone w wyniku. Dla wygody założymy, że punkty podane będą w kolejności rosnącej, oraz odległości między sąsiednimi punktami będą podzielne przez długość kroku h. Dzięki temu założeniu potencjalne punkty nieciągłości naszego przybliżonego sterowania będą zawsze wypadały między krokami metody R-K, dzięki czemu pojedynczy krok algorytmu będzie przybliżał rozwiązanie równania różniczkowego o ciągłej prawej stronie, co jest jednym z założeń metody R-K.

2.3. Funkcja celu

Przybliżoną funkcję celu (1.14) można zaimplementować za pomocą iloczynu skalarnego:

$$Q(\hat{y}) = (h(\alpha_0), \ldots, h(\alpha_N))^T \cdot j(\hat{y}(0, h, 2h, \ldots, T)) \tag{2.5}$$

Zauważmy, że korzystamy tu, dla prostoty, z tej samej długości kroku, co w (1.11).

W implementacji w Octave korzystamy ze zwektoryzowanej implementacji funkcji $\hat{y}$. Implementacja funkcji j też jest zwektoryzowana, ponieważ środowisko dostarcza nam zwektoryzowaną funkcję tanh, a pozostałe operacje wektoryzują się naturalnie.

Gradient funkcji celu możemy zaimplementować jako mnożenie wektora przez macierz:

$$\left[\frac{\partial J}{\partial g_i}\right]_{i=0,\dots n} = (h(\alpha_0), \dots, h(\alpha_N))^T \cdot \left[\frac{\partial j(\hat{y}(hi))}{\partial g_k}\right]_{i=0,\dots N, k=1,\dots n} \tag{2.6}$$

gdzie

$$\frac{\partial j(y(t))}{\partial g_i} = \frac{\partial y_1(t)}{\partial g_i} + \frac{\partial y_2(t)}{\partial g_i} + \omega G'\left(\frac{y_2(t) - y_1(t)}{\epsilon}\right)\frac{\frac{\partial y_2(t)}{\partial g_i} - \frac{\partial y_1(t)}{\partial g_i}}{\epsilon}, \quad G'(x) = \frac{1}{2\cosh^2(x)} \tag{2.7}$$

Rozdział 3

Eksperymenty

W eksperymentach, do przybliżania rozwiązania równania (1.4), będziemy korzystać z metody Rungego-Kutty 4-tego rzędu, ponieważ daje ona dobrą dokładność, a liczenie wartości funkcji f nie jest zbyt kosztowne. Wartości stałych c_l, a_{li}, b_i z wzoru (1.11) podane na tabeli Butchera:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{3.1}$$

Do przybliżania funkcjonału celu (1.3) użyjemy kwadratury trapezów. Kwadratura (1.14) wyraża się więc przez:

$$Q(\hat{y}) = h\left(\frac{1}{2}(j(\hat{y}(0)) + j(\hat{y}(T))) + \sum_{i=1}^{N-1} j(\hat{y}(ih))\right) \tag{3.2}$$

3.1. Zbiór testowy

Opiszemy teraz szeroki zbiór testowy, dobrany w taki sposób, aby móc zbadać wpływ różnych czynników na ostateczny wynik optymalizacji.

3.1.1. Zestawy parametrów

Użyjemy wartości parametrów podanych w wyjściowej pracy [1]:

$t_0 = 0$	$\lambda_1 = 0.192$	$g_{\max} = 3$
$T = 200$	$\lambda_2 = 0.192$	$\epsilon = 0.01$
$V_{10} = 20$	$\beta_1 = 0.15$	$d = 0.00873$
$V_{20} = 280$	$\beta_2 = 0.1$	$\mu = 0$
$K_0 = 650$	$\beta = 0.05$	

Wartości parametrów α_{12}, α_{21}, ω nie zostały ustalone w pracy, jedyne co jest ustalone to

$$\alpha_{12} < \alpha_{21} \quad 0 < \omega \leq 2000$$

więc zbadamy problem dla 2 arbitralnie wybranych zestawów wartości tych parametrów (oznaczamy jako „Param.” w tabelach z wynikami):

- (CC) $\alpha_{12} = 0.1 \quad \alpha_{21} = 0.15 \quad \omega = 1000$
- (DC) $\alpha_{12} = 0.5 \quad \alpha_{21} = 0.75 \quad \omega = 2000$

3.1.2. Algorytmy

Do eksperymentów użyte zostało środowisko Octave. Funkcja FMINICON dostępna w pakiecie Optim [10] umożliwia użycie jednego z dwóch algorytmów do optymalizacji nieliniowej:

- *lm_feasible* oznaczany w tabelach *lm*

  Algorytm ten opiera się na algorytmie Levenberga–Marquardta, zob. [6]. Implementacja tego algorytmu w Octave gwarantuje spełnianie ograniczeń w każdym kroku optymalizacji.

- *sqp*

  Algorytm ten jest implementacją algorytmu sekwencyjnego programowania kwadratowego (ang. Sequential Quadratic Programming), zob. [4]. Implementacja tego algorytmu w Octave gwarantuje spełnienie ograniczeń jedynie na końcu procesu optymalizacji.

3.1.3. Metody dyskretyzacji

Do dyskretyzacji sterowania użyjemy dwóch metod (oznaczane „aproks." tabelach z wynikami):

- P_0: wykorzystująca splajny kawałkami stałe, zob. (2.1)
- P_1: wykorzystująca splajny kawałkami liniowe, zob. (2.2)

Splajny te opierają się na siatce która, na użytek eksperymentu, może być jednego z trzech rodzajów (oznaczane „siatka" w tabelach z wynikami):

- S_τ: jednorodna, z krokiem τ, więc $t_n = \tau n$. Ograniczymy się do $\tau \in \{0.1, 0.5, 1, 4\}$.
- N_{sr}: Niejednorodna, gęstsza w środku, $t_n = \begin{cases} n & \text{gdy } n \le 25 \\ 25 + 0.1 \cdot (n - 25) & \text{gdy } 25 < n \le 525 \\ 75 + (n - 525) & \text{gdy } n > 525 \end{cases}$
- N_{kon}: Niejednorodna, gęstsza na końcach, $t_n = \begin{cases} 0.5 \cdot n & \text{gdy } n \le 100 \\ 50 + (n - 100) & \text{gdy } 100 < n \le 200 \\ 150 + 0.5 \cdot (n - 200) & \text{gdy } n > 200 \end{cases}$

3.1.4. Krok dyskretyzacji h metody R-K rozwiązania (1.4)

Ograniczymy się do $h \in \{0.02, 0.1, 0.5\}$.

3.1.5. Sterowania startowe

Optymalizator FMINICON wymaga podania początkowego przybliżenia sterowania optymalnego. Rozpoczęcie optymalizacji od odpowiednio dobrego sterowania może być niezbędne do uzyskania dobrych wyników, choć istnieją metody optymalizacji pozwalające osiągać dobre rezultaty nawet przy niezbyt dobrym przybliżeniu początkowym. Jest to bardzo przydatne w zadaniach w których nie znamy dobrego punktu startowego. Czasem jednak wiedza ekspercka pozwala ustalić dobre przybliżenie początkowe które wystarczy lekko poprawić aby uzyskać

optimum. W naszym przypadku nie znamy dobrych kandydatów na sterowanie startowe. Przetestujemy więc sterowania ekstremalne (stale równe 0 lub $g_{\max}$) oraz sterowania kawałkami stałe postaci

$$g_{\alpha,\beta,\gamma} = \alpha \cdot \mathbb{1}_{[t_0,\gamma)} + \beta \cdot \mathbb{1}_{[\gamma,T)}. \tag{3.3}$$

Ograniczymy się do: $g_{0,0.55,42.5}$, $g_{0.07,0.59,48.2}$. Zastosowane sterowanie startowe będzie podane w kolumnie „start" w tabelach z wynikami.

3.1.6. Heurystyczna metoda znajdowania sterowania startowego

Aby znaleźć sterowanie startowe które pozwoli uzyskać dobre wyniki, posłużyliśmy się pewną heurystyczną metodą. Będziemy szukać punktu startowego kawałkami stałego z maksymalnie jednym punktem nieciągłości. Zauważmy, że sterowanie takie można sparametryzować trzema wartościami, co zrobiono w (3.3). Aby uzyskać jak najlepszy punkt startowy tej postaci, dostosowaliśmy naszą implementację rozwiązania problemu do podanej parametryzacji, czyli optymalizowaliśmy problem

$$\min_{\alpha,\beta,\gamma} \check{J}(\alpha,\beta,\gamma) := \hat{J}(\alpha,\beta) \quad \text{z dyskretyzacją } t = (t_0,\gamma,T) \tag{3.4}$$

i z ograniczeniami

$$0 \le \alpha,\beta \le g_{\max}, \quad t_0 < \gamma < T. \tag{3.5}$$

Zauważmy, że ta metoda jest heurystyczna, ponieważ nie wiemy czy $\check{J}(\alpha,\beta,\gamma)$ zależy od γ w sposób różniczkowalny, ani nawet ciągły. Z tego też powodu nie mogliśmy policzyć analitycznie gradientu dla tego zadania i skorzystaliśmy z metody różnic skończonych. Aby otrzymać dobre rezultaty trzeba było uważnie ustawić parametr zatrzymania optymalizacji („TolFun"), oraz długość kroku różnic skończonych. Warto jeszcze nadmienić, że przez takie sformułowanie problemu nie mamy gwarancji, że punkt nieciągłości wypada między krokami algorytmów R-K (1.11) i kwadratury (1.14), co może skutkować znacznym błędem w wynikach tych metod na jednym przedziale długości h. Z tego powodu zastosowano wartość $h = 0.05$, niższą niż w większości eksperymentów.

Pomimo heurystyczności tej metody, udało się za jej pomocą uzyskać sterowanie $g_{0.07,0.59,48.2}$, które osiągnęło najlepsze wyniki w eksperymentach.

3.1.7. Warunek stopu

Algorytm *lm* kończy optymalizację gdy spełniony jest jeden z kilku warunków. Jednym z nich jest zbyt niewielka względna poprawa funkcji celu. Minimalna wartość tej poprawy jest ustawiana w FMINCON parametrem „TolFun". Domyślną wartością tego parametru jest 10^{-4}.

Podobnie dla algorytmu *sqp* istnieje w Octave parametr „octave_sqp_tolerance", który odpowiada za różne kryteria stopu (niestety dokumentacja nie precyzuje za które). Domyślną wartością tego parametru jest pierwiastek z precyzji arytmetyki, czyli ok. 10^{-8}.

W eksperymentach przetestujemy wpływ obu tych parametrów na proces optymalizacji. Dla prostoty oznaczmy oba te parametry przez „Tol", co oznacza, że gdy testujemy algorytm *lm* to przez „Tol" mamy na myśli „TolFun", a gdy testujemy *sqp*, to mamy na myśli „octave_sqp_tolerance". Przetestujemy wartości Tol $\in \{10^{-6}, 10^{-9}, 10^{-11}, 10^{-13}\}$.

3.2. Wyniki eksperymentów

Jako, że powyższy zbiór jest duży, wykorzystamy jedynie niektóre testy. Przeprowadzimy kilka eksperymentów, każdy eksperyment będzie przeprowadzony na podzbiorze zbioru testowego, wybranym tak, aby zaprezentować wpływ konkretnego parametru na wyniki.

Wyniki eksperymentów będziemy oceniać na podstawie kilku wartości: wartości przybliżonego funkcjonału celu $\hat{J}$, liczby iteracji optymalizatora (oznaczaną przez „iter"), oraz liczbę wywołań funkcjonału celu w trakcie optymalizacji (oznaczaną przez $\#\hat{J}$).

Podamy też wartości pomagające ocenić jak blisko jest rozwiązaniu do faktycznego minimum lokalnego. W zadaniach optymalizacji z ograniczeniami częstym warunkiem stopu optymalizatora jest zerowanie się gradientu Lagrangianu wynikającego z warunków KKT. Jest to opisane np. w [5]. W naszym przypadku ograniczenia (1.17) mają prostą postać, więc gradient wspomnianego Lagrangianu upraszcza się do

$$\nabla L = (l_i)_{i=0,\dots,n}^{T} \quad \text{gdzie} \quad l_i = \begin{cases} 0 & \text{gdy } g_i = 0 \wedge \frac{\partial \hat{J}}{\partial g_i} > 0 \\ 0 & \text{gdy } g_i = g_{\max} \wedge \frac{\partial \hat{J}}{\partial g_i} < 0 \\ \frac{\partial \hat{J}}{\partial g_i} & \text{w.p.p.} \end{cases} . \tag{3.6}$$

Będziemy podawać normę gradientu Lagrangianu dla punktu zakończenia optymalizacji $\|L\|_1$, oraz normę względną, czyli stosunek normy gradientów w punkcie zakończenia optymalizacji i w punkcie startowym: $\frac{\|L\|_1}{\|L_0\|_1}$.

Wartości funkcji celu są duże, więc dla zwiększenia czytelności będziemy podawać wartości dla $10^{-5}\hat{J}$ zaokrąglone do 2 miejsc po przecinku. Podobnie, dla normy lagrangianu będziemy podawać wartość $10^{-5}\|L\|_1$ zaokrągloną do 2 miejsc po przecinku. Wartość normy względnej $\frac{\|L\|_1}{\|L_0\|_1}$ będziemy podawać zaokrągloną do 3 miejsc po przecinku, lub (gdy jest mniejsza niż 10^{-2}) w postaci wykładniczej z dwoma cyframi znaczącymi.

3.2.1. Test poprawności liczenia gradientu

Pierwszy eksperyment ma na celu zaprezentowanie poprawności implementacji obliczania gradientu funkcji celu. Aby to sprawdzić porównamy naszą implementację, z domyślną implementacją przybliżającą gradient za pomocą różnic skończonych (FD) dostępną w FMINICON. Obliczanie wyników metodą domyślną wymaga wiele czasu, więc eksperyment obejmie tylko jeden przypadek, opisany na Tabeli 3.1. Celem tego eksperymentu jest sprawdzenie poprawności implementacji, więc ograniczymy liczbę iteracji optymalizatora do 20, oraz pominiemy podawanie norm gradientów.

Gradient	Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
FD	(CC)	*lm*	P_0	S_4	0.1	g_0	3.26	20	38
wg. rozdz. 2.1, 2.2	(CC)	*lm*	P_0	S_4	0.1	g_0	3.25	20	37

Tablica 3.1: Test implementacji gradientu

Rozwiązania uzyskane obiema metodami są bardzo podobne, o czym świadczą wyniki zaprezentowane na Tabeli 3.1 i Rysunku 3.1. Ponadto nasza implementacja osiągnęła minimalnie lepszy wynik.

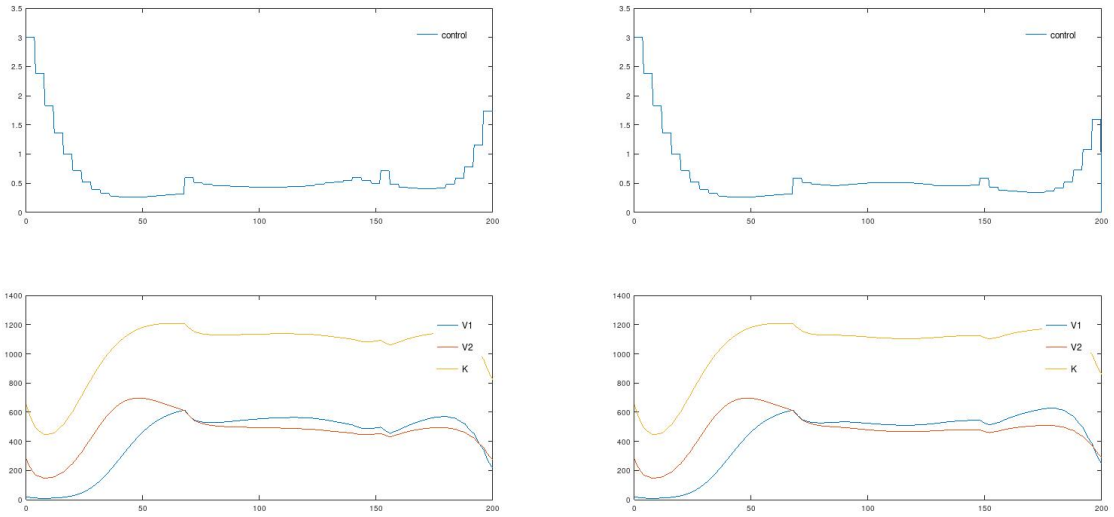
Dodatkowo policzymy gradient G w punkcie będącym sterowaniem zaprezentowanym na wykresie (3.1b), oraz obliczymy przybliżenie G_τ tego gradientu za pomocą metody FD z krokiem τ, dla różnych wartości kroku:

$$\tau \in \{10^{-3}, 10^{-4}, \dots, 10^{-10}\}.$$

Następnie dla każdego τ obliczymy normę różnicy: $\|G - G_\tau\|_1$. Wyniki tych obliczeń przedstawiamy na Tabeli 3.2.

(a) Rozwiązanie przy użyciu różnic skończonych

(b) Rozwiązanie z liczeniem gradientu



Rysunek 3.1: Rozwiązania dla przypadku testowego

τ	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
$\|G - G_\tau\|_1$	444599.7	34471.1	357.2	3.6	0.5	4.7	54.0	384.5

Tablica 3.2: Różnice wyników między naszą implementacją gradientu, a różnicami skończonymi.

Zgodnie z oczekiwaniami, gdy krok różnic skończonych τ maleje, to do pewnego momentu maleje też $\|G - G_\tau\|_1$. W pewnym momencie norma ta zaczyna rosnąć wraz z dalszym zmniejszaniem τ, gdyż dla tak niskich wartości τ obliczanie różnic dzielonych obarczone jest znacznym błędem numerycznym.

Warto tu jeszcze dodać, że liczba wywołań $\hat{J}$ podana w Tabeli 3.1 nie uwzględnia wywołań potrzebnych do aproksymacji gradientu metodą różnic skończonych. Aby policzyć pochodną dla jednego parametru potrzeba raz dodatkowo wywołać $\hat{J}$, więc jednorazowe policzenie gradientu wymaga tylu wywołań ile jest parametrów (w tym przypadku 51). Gradient jest liczony w każdej iteracji, co daje łącznie $20 \cdot 51 = 1020$ dodatkowych wywołań $\hat{J}$. Inną wadą przybliżania gradientu metodą różnic skończonych są błędy. Zauważmy, że w okolicy punktu przecięcia się krzywych y_1 i y_2, wartość i pochodna wyrażenia $G((y_2(t) - y_1(t))/\epsilon)$, gdzie $G(x) = \frac{1+\tanh(x)}{2}$, zmienia się bardzo szybko w czasie, co powoduje znaczne błędy przy zbyt dużym kroku metody różnic skończonych. Możemy je zaobserwować na Tabeli 3.2, w postaci dużej normy różnicy gradientów, gdy krok różnic skończonych to 10^{-3}. Wartość 10^{-3} jest wartością domyślna w FMINICON i została użyta do obliczenia wyników tego eksperymentu. Oznacza to w szczególności, że powyższe wyniki uzyskane przy użyciu FD są zapewne bardzo odległe od optymalnych.

3.2.2. Parametry (CC)

Wyniki dla parametrów (CC) są dość jednoznaczne. Najmniejszą wartość $\hat{J}$ osiągamy dla sterowania $g \equiv g_{\max} = 3$, które zaprezentowano na Rysunku 3.2. Wszystkie testy zaczynające z tego sterowania zatrzymały się po jednej iteracji, patrz Tabela 3.3. Na podstawie tego, że norma lagrangianu w g_3 jest zerowa, możemy wnioskować, że rozwiązanie to jest minimum lokalnym

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|L\|_1$	$\frac{\|L\|_1}{\|L_0\|_1}$
(CC)	*lm*	P_0	S_1	0.1	g_0	10^{-6}	2.2	3	8	0.32	0.059
(CC)	*lm*	P_0	S_1	0.1	g_0	10^{-9}	2.2	64	69	0.32	0.059
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	10^{-6}	2.19	3	8	0.31	0.058
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	10^{-9}	2.19	58	63	0.31	0.058
(CC)	*lm*	P_0	N_{kon}	0.1	g_0	10^{-6}	2.64	9	14	0.79	0.146
(CC)	*lm*	P_0	N_{kon}	0.1	g_0	10^{-9}	2.63	87	92	0.7	0.13
(CC)	*lm*	P_1	S_1	0.1	g_0	10^{-6}	2.21	2	7	0.33	0.061
(CC)	*lm*	P_1	S_1	0.1	g_0	10^{-9}	2.21	89	94	0.33	0.061
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_0	10^{-6}	2.21	2	7	0.33	0.061
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_0	10^{-9}	2.21	79	84	0.33	0.061
(CC)	*lm*	P_1	N_{kon}	0.1	g_0	10^{-6}	2.66	2	7	0.82	0.152
(CC)	*lm*	P_1	N_{kon}	0.1	g_0	10^{-9}	2.66	24	29	0.82	0.152
(CC)	*lm*	P_0	S_1	0.1	g_3	10^{-6}	2.14	1	2	0.0	—
(CC)	*lm*	P_0	S_1	0.1	g_3	10^{-9}	2.14	1	2	0.0	—
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-6}	2.14	1	2	0.0	—
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-9}	2.14	1	2	0.0	—
(CC)	*lm*	P_0	N_{kon}	0.1	g_3	10^{-6}	2.14	1	2	0.0	—
(CC)	*lm*	P_0	N_{kon}	0.1	g_3	10^{-9}	2.14	1	2	0.0	—
(CC)	*lm*	P_1	S_1	0.1	g_3	10^{-6}	2.14	1	2	0.0	—
(CC)	*lm*	P_1	S_1	0.1	g_3	10^{-9}	2.14	1	2	0.0	—
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_3	10^{-6}	2.14	1	2	0.0	—
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_3	10^{-9}	2.14	1	2	0.0	—
(CC)	*lm*	P_1	N_{kon}	0.1	g_3	10^{-6}	2.14	1	2	0.0	—
(CC)	*lm*	P_1	N_{kon}	0.1	g_3	10^{-9}	2.14	1	2	0.0	—
(CC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	10^{-6}	2.14	9	10	0.0	0

Tablica 3.3: Eksperymenty z parametrami (CC) dla różnych wyborów aproksymacji, siatki i startu

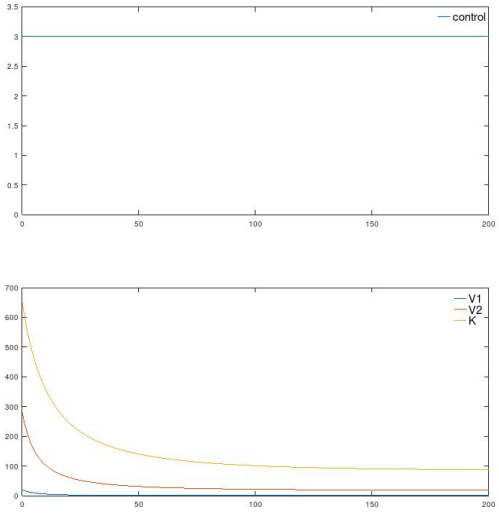
naszego problemu.

Wyniki testów z Tabeli 3.3 korzystające z siatki niejednorodnej wypadają zauważalnie gorzej od pozostałych. Widzimy też, że zmniejszenie Tol poprawiło wartość $\hat{J}$ w bardzo małym stopniu, a znacząco zwiększyło liczbę iteracji. Algorytm *sqp* jest jedynym który zbiegł z zerowego sterowania do sterowania optymalnego, co pokazuje że algorytm ten lepiej sprawuje się w powyższych przypadkach testowych.

Jako, że najlepsze znalezione, dla parametrów (CC), rozwiązanie trywializuje się, w dalszych eksperymentach skupimy się na parametrach (DC), które dodatkowo zdają się prowadzić do nieciągłego optymalnego sterowania.

3.2.3. Wpływ sterowania startowego

Wyniki badające różne sterowania startowe dla różnych algorytmów i wartości tolerancji przedstawione zostały na Tabeli 3.4. Pozwalają one na wyciągnięcie kilku ciekawych wniosków. Po pierwsze widzimy, że dla Tol = 10^{-6} względne normy lagrangianów są bardzo duże, w szczególności prawie zawsze większe od 1. Oznacza to, że rozwiązania te są zapewne dalekie od optimów lokalnych, więc ta wartość Tol jest zapewne zbyt duża i dalej będziemy skupiać się na badaniu



Rysunek 3.2: Najlepsze znalezione rozwiązanie dla parametrów (CC)

niższych wartości tolerancji. Ponadto wygląda na to, że zmniejszenie Tol nie poprawiło wyników algorytmu *sqp*.

Na Tabeli 3.4 widzimy, że norma lagrangianu dla rozwiązania g_3 wynosi 0, więc możemy wnioskować, że przy parametrach (DC), podobnie jak przy parametrach (CC), sterowanie g_3 (czyli stale równe 3 = $g_{\max}$) jest minimum lokalnym. Jednak w tym przypadku wartości $\hat{J}$ są znacznie gorsze od pozostałych rezultatów.

Najlepsze wartości $\hat{J}$ osiągane są przy sterowaniach startowych $g_{0,0.55,42.5}$ i $g_{0.07,0.59,48.2}$. Widzimy, że nawet dla Tol = 10^{-6} wychodzą dobre wartości funkcjonału celu, ale dopiero ustawienie Tol na 10^{-9} pozwala uzyskać niskie wartości $\frac{\|L\|_1}{\|L_0\|_1}$ i tutaj wyraźnie lepiej wypada sterowanie startowe $g_{0.07,0.59,48.2}$. Warto też zauważyć, że pomimo podobnych wartości funkcji $\hat{J}$ dla tych czterech przypadków testowych, wynikowe sterowania zauważalnie się dla nich różnią, co pokazano na Rysunkach 3.3 i 3.4.

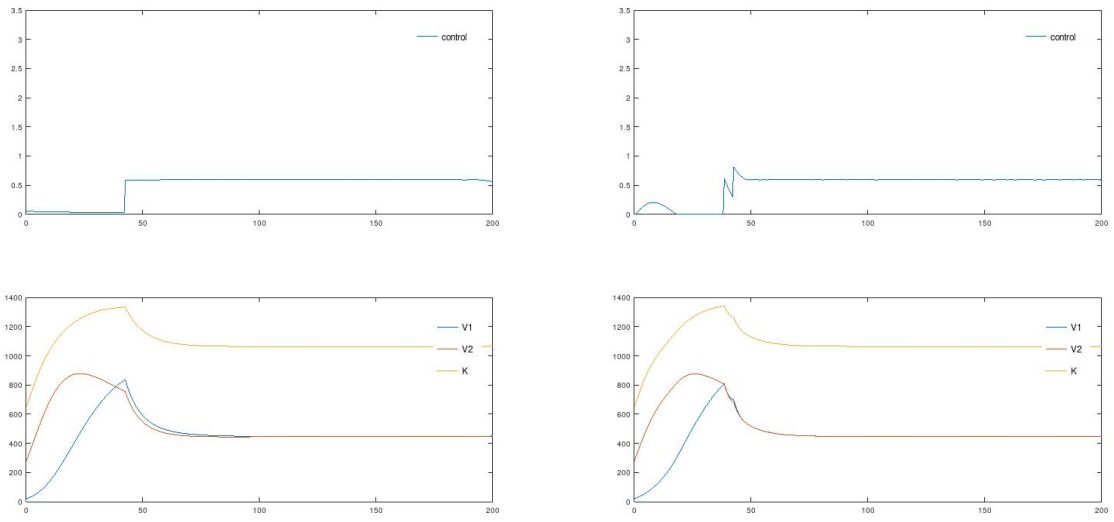
Widzimy też, że dla startu g_0 przy Tol = 10^{-9} i algorytmie *lm* udało się osiągnąć niską normę lagrangianu, jednak wartość $\hat{J}$ dla tego przypadku jest znacznie gorsza niż dla wcześniej wspomnianych przypadków. Wnioskujemy więc z tego, że dla tego punktu startowego optymalizator zbiega do nieoptymalnego minimum lokalnego. Potwierdzają to wyraźne różnice między kształtem sterowania uzyskanego dla tego przypadku, a wyżej omawianymi wynikami, które możemy zaobserwować porównując wykresy przedstawione na Rysunkach 3.5, 3.3b i 3.4b. Z tego powodu nie będziemy dalej zajmować się tym sterowaniem startowym.

3.2.4. Metoda dyskretyzacji

Jak widzimy na Tabeli 3.5 dyskretyzacja kawałkami stała wydaje się osiągać znacznie lepsze rezultaty od optymalizacji kawałkami liniowej. Co prawda dla algorytmu *sqp* wynik wydaje się taki sam, a norma względna gradientu nawet lepsza, jednakże norma bezwzględna jest w tym przypadku zauważalnie gorsza. Problemy z dyskretyzacją P_1 są zapewne powodowane możliwą nieciągłością sterowania optymalnego i przybliżaniem nieciągłego sterowania $g_{0.07,0.59,48.2}$ sterowaniem kawałkami liniowym. Zakończenie optymalizacji po pierwszej iteracji przez algorytm *lm* jest spowodowane brakiem znalezienia lepszego sterowania w otoczeniu sterowania startowego,

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|L\|_1$	$\frac{\|L\|_1}{\|L_0\|_1}$
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	10^{-6}	3.04	81	158	141.46	47.286
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	10^{-9}	2.92	1003	1791	0.01	2.2e-03
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	10^{-6}	3.3	18	205	21.37	7.145
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	10^{-9}	3.3	18	205	21.37	7.145
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-6}	4.07	1	2	0.0	—
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-9}	4.07	1	2	0.0	—
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_3	10^{-6}	4.07	1	2	0.0	—
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_3	10^{-9}	4.07	1	2	0.0	—
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0,0.55,42.5}$	10^{-6}	2.72	31	64	50.43	25.094
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0,0.55,42.5}$	10^{-9}	2.69	412	722	0.49	0.242
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0,0.55,42.5}$	10^{-6}	2.71	10	130	3.62	1.799
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0,0.55,42.5}$	10^{-9}	2.71	10	130	3.62	1.799
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-6}	2.72	45	91	2.18	0.725
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.0	6.7e-04
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-6}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	93	6.16	2.045

Tablica 3.4: Wpływ doboru sterowania startowego na przebieg optymalizacji



(a) Tol = 10^{-6}

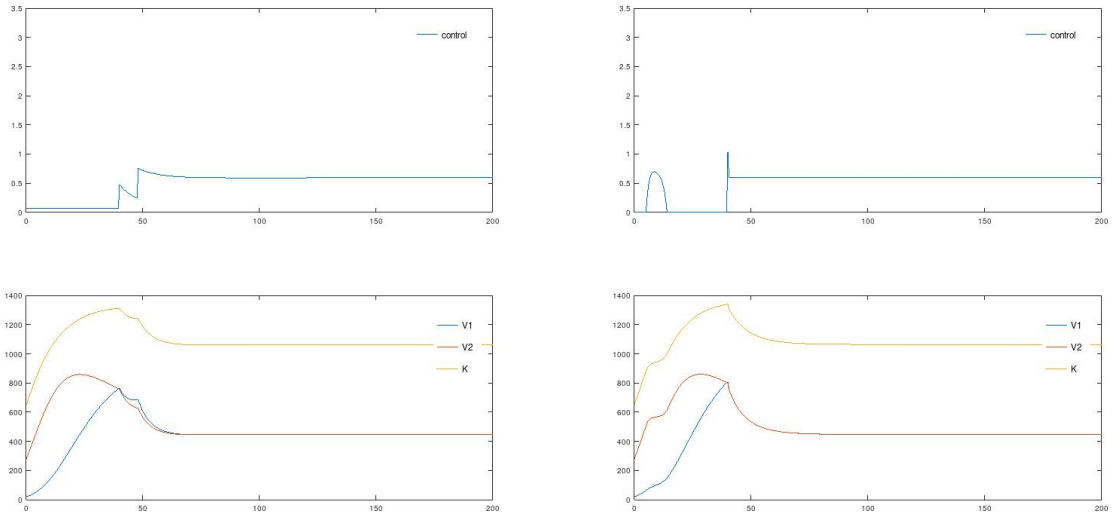
(b) Tol = 10^{-9}

Rysunek 3.3: Rozwiązania uzyskane przy użyciu sterowania startowego $g_{0,0.55,42.5}$

co z kolei może być powodowane błędami wynikającymi z przybliżania sterowania nieciągłego sterowaniem kawałkami liniowym.

3.2.5. Siatka dyskretyzacji

Zgodnie z oczekiwaniami gęstsza siatka jednorodna osiąga lepsze wyniki dla algorytmu *lm*, co widzimy na Tabeli 3.6, natomiast siatka niejednorodna okazuje się osiągać gorsze wyniki od



(a) Tol = 10^{-6} (b) Tol = 10^{-9}

Rysunek 3.4: Rozwiązania uzyskane przy użyciu sterowania startowego $g_{0.07,0.59,48.2}$

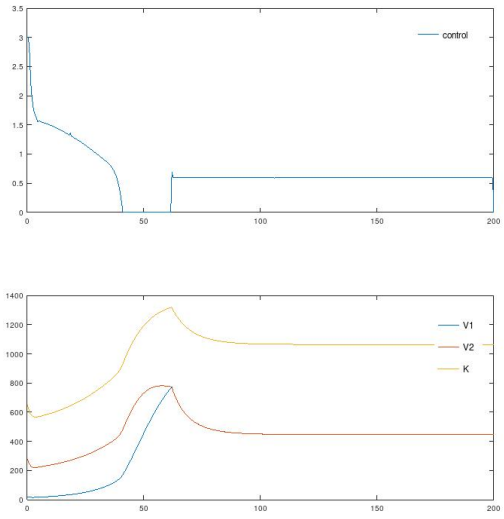
Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|L\|_1$	$\frac{\|L\|_1}{\|L_0\|_1}$
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.0	6.7e-04
(DC)	*lm*	P_1	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	1	6	4.54	1.0
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_1	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	101	7.06	1.556

Tablica 3.5: Wpływ metody dyskretyzacji sterowania na przebieg optymalizacji

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|L\|_1$	$\frac{\|L\|_1}{\|L_0\|_1}$
(DC)	*lm*	P_0	S_1	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.7	620	1110	0.18	0.058
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.0	6.7e-04
(DC)	*lm*	P_0	N_{sr}	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.7	280	505	2.86	1.237
(DC)	*sqp*	P_0	S_1	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	11	187	5.45	1.812
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_0	N_{sr}	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	8	147	14.51	6.276

Tablica 3.6: Wpływ doboru siatki dyskretyzacji na przebieg optymalizacji

siatek jednorodnych. Co prawda wartości $\hat{J}$ dla siatki niejednorodnej są niewiele gorsze od tych dla siatki jednorodnej, a liczba iteracji znacznie mniejsza, ale po normach gradientu widzimy, że punkt do którego zbiegła optymalizacja jest znacznie odległy od minimum lokalnego. Dla algorytmu *sqp* wartości $\hat{J}$ są takie same dla każdej siatki, ale widzimy, że gradient wyniku jest najmniejszy dla siatki S_1, czyli w tym wypadku lepiej wypada rzadsza siatka jednorodna.



Rysunek 3.5: Rozwiązanie uzyskane przy użyciu sterowania startowego g_0 i Tol = 10^{-9}

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|L\|_1$	$\frac{\|L\|_1}{\|L_0\|_1}$
(DC)	*lm*	P_0	$S_{0.5}$	0.5	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	974	1741	0.0	6.7e-04
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.0	6.7e-04
(DC)	*lm*	P_0	$S_{0.5}$	0.02	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	1	6	2.95	1.0
(DC)	*sqp*	P_0	$S_{0.5}$	0.5	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	3	66	9.55	2.824
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_0	$S_{0.5}$	0.02	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	100	17.14	5.814

Tablica 3.7: Wpływ doboru kroku dyskretyzacji h na przebieg optymalizacji

3.2.6. Krok dyskretyzacji h

Wyniki eksperymentów badających wpływ dobory kroku dyskretyzacji h przestawiono na Tabeli 3.7. Widzimy, że przy obniżeniu długości kroku z 0.5 do 0.1 wyniki się poprawiły. Dla algorytmu *lm* spadła liczba potrzebnych iteracji, a dla algorytmu *sqp* spadła norma lagrangianu. Przy dalszym obniżaniu kroku h wyniki wydają się pogarszać, w szczególności dla algorytmu *lm*, któremu nie udało się znaleźć żadnego sterowania lepszego od sterowania startowego i zakończył optymalizację po pierwszej iteracji. Zauważmy tutaj, że dla różnych długości kroków h rozpatrujemy faktycznie inne zadanie, więc takie zachowanie jest możliwe.

3.2.7. Warunek stopu

Najlepsze wyniki na Tabeli 3.8 osiągane są dla Tol = 10^{-11}. Przy Tol = 10^{-6} norma $\|G\|_1$ jest duża, więc rozwiązanie uzyskane w ten sposób jest raczej odległe od minimum lokalnego. Widzimy też poprawę normy względnej lagrangianu o rząd wielkości przy obniżeniu wartości Tol z 10^{-9} do 10^{-11}, choć należy zauważyć, że poprawa ta wymagała przeprowadzenia aż 500 dodatkowych iteracji. Przy dalszym obniżaniu Tol do 10^{-13} wyniki wydają się pogarszać, więc zapewne optymalna wartość Tol dla tego przypadku jest bliska 10^{-11}.

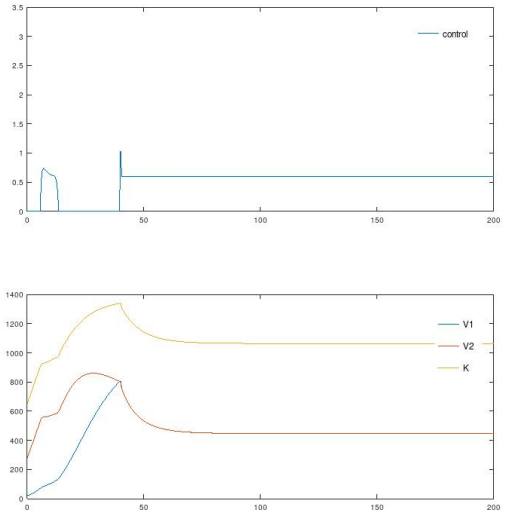
Ten eksperyment potwierdza nasze podejrzenia, że dla algorytmu *sqp* parametr tolerancji nie

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|L\|_1$	$\frac{\|L\|_1}{\|L_0\|_1}$
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-6}	2.72	45	91	2.18	0.725
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.0	6.7e-04
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-11}	2.69	1314	2313	0.0	3.7e-05
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-13}	2.69	1401	2466	0.0	5.7e-04
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-6}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-11}	2.76	5	93	6.16	2.045
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-13}	2.76	5	93	6.16	2.045

Tablica 3.8: Wpływ warunku stopu na przebieg optymalizacji

ma wpływu na wynik w naszych eksperymentach, a wyniki osiągane przez ten algorytm wydają się być dalekie od minimum zarówno lokalnego jak i globalnego.

Podsumowując najlepsze wyniki udało się osiągnąć przy użyciu algorytmu *lm*, aproksymacji kawałkami stałej, siatki $S_{0.5}$ i kroku $h = 0.1$, zaczynając ze sterowania $g_{0.07,0.59,48.2}$, przy wartości parametru tolerancji 10^{-11}. Są one zaprezentowane na Rysunku 3.6.



Rysunek 3.6: Najlepsze znalezione rozwiązanie dla parametrów (DC)

3.3. Dyskusja wyników

Jak widzimy, wyniki dla parametrów (CC) i (DC) znacznie się różnią. Dokładniejsze przyjrzenie się funkcjonałowi celu (1.5) pozwala uzasadnić takie zjawisko na poziomie intuicji. Jak widzimy parametr ω jest mnożony przez wartość drugiej całki. Zauważmy, że funkcja pod drugą całką jest gładkim przybliżeniem funkcji znaku wyrażenia $y_2(t) - y_1(t)$. W przypadku (DC) parametr ω jest znacznie większy niż w (CC), więc w tym przypadku druga całka ma większy wpływ na wynik funkcjonału. Jak przyjrzymy się rozwiązaniu z np. Rysunku 3.4b, widzimy, że punkt nieciągłości znajduje się od razu po przecięciu się krzywych y_1 i y_2, a po nim sterowanie ma taką

wartość aby krzywe te były bardzo blisko siebie, ale przy zachowaniu $y_2(t) - y_1(t) < 0$. Wygląda więc na to, przy parametrach (DC) bardziej opłaca się utrzymywać stan $y_2(t) - y_1(t) < 0$, natomiast przy parametrach (CC) lepsze wyniki daje minimalizacja pierwszej całki, czyli pola pod $y_1 + y_2$. Możemy się też spodziewać, że wysokie wartości sterowania powodują zmniejszanie się zarówno y_1 jak i y_2, ale od pewnej wartości sterowania y_1 maleje szybciej niż y_2. Taka hipoteza zgadzałaby się z interpretacją y_1 i y_2 jako liczba komórek nowotworowych odpowiednio podatnych na działanie leku i odpornych na niego, a wartości sterowania jako dawki leku.

Otrzymywane normy względne lagrangianów były dość niewielkie dla najlepszych otrzymywanych wyników. Sugeruje to, że otrzymywane wyniki są bliskie optimom lokalnym. Należy jednak pamiętać, że nie jest to jeszcze dowód i wyniki te mogą jednak okazać się odległe od optimów lokalnych. Ponadto otrzymywane wyniki były mocno zależne od punktu startowego, co pozwala podejrzewać, że istnieją punkty startowe dla których moglibyśmy osiągnąć znacznie lepsze wyniki. W szczególności podejrzewamy, że sterowania startowe z różnymi punktami nieciągłości prowadzą do innych optimów lokalnych. Podejrzenia te są oparte na obserwacji, że optymalizacja, zaczynając z nieciągłego sterowania, kończyła się zwykle wynikiem mającym nieciągłość w tym samym punkcie. Być może zastosowanie innych algorytmów optymalizacji nieliniowej umożliwiłoby szukanie rozwiązania w sposób mniej zależący od znalezienia dobrego punktu startowego. Może też da się sformalizować heurystyczną metodę poszukiwania punktu startowego opisaną w rozdziale 3.1.6, oraz uzyskać za jej pomocą znacznie lepsze wyniki. Ostatecznie, być może przyjęta przez nas metoda „direct single shooting" nie jest dobrym wyborem dla tego zadania. Być może użycie np. metody „direct collocation" pozwoliłoby osiągnąć lepsze, bliższe lokalnemu i globalnemu minimum wyniki.

Niejednorodna siatka dyskretyzacji jest metodą nierzadko pozwalającą na poprawę tempa zbieżności, ale nie udało się jej tu z sukcesem zastosować. Podobnie przybliżanie sterowania za pomocą splajnu wyższego rzędu też mogłoby poprawić tempo zbieżności. W literaturze rozważa się też metody automatyzacji znajdowania siatki dyskretyzacji i odpowiedniego rzędu dyskretyzacji. Przykładem jest tu praca [8]. Być może zastosowanie ich umożliwiłoby uzyskanie lepszych wyników.

Dla parametrów (DC) nie udało nam się skutecznie zastosować algorytmu *sqp*. Algorytm ten potrafi osiągać dobre wyniki, co mogliśmy zaobserwować w przypadku (CC), dla którego osiągnął on znacznie lepsze rezultaty niż *lm_feasible*. Podejrzewamy, że problemem w naszym przypadku były zbyt luźne warunki zakończenia optymalizacji tego algorytmu. Niestety dokumentacja algorytmu *sqp* nie specyfikuje jakie są warunki zakończenia optymalizacji i na co wpływa parametr „octave_sqp_tolerance". Ponadto, zgodnie z dokumentacją, jest to jedyny parametr służący do kontrolowania warunków zakończenia optymalizacji używany przez ten algorytm.

Przypomnijmy jeszcze uwagę z początku pracy, że gdy $y_1 = y_2 = 0$, lub $y_3 = 0$ to zadanie nie jest dobrze określone, a w przypadku gdy wartości te są bliskie zeru, mogą występować znaczne błędy numeryczne. W żadnym eksperymencie wartości y_1 ani y_2 nie były bliskie zeru. Minimalna wartość y_3 z Rysunku 3.2 to 14.9, więc i ta nie jest zbyt bliska 0. Okazuje się jednak, że przy parametrach (DC), w rozwiązaniu (1.4) dla sterowania stale równego $g_{\max}$ minimalna wartość y_3 wynosi ok. 0.02. Nie jest to wartość na tyle mała, by móc powodować istotne błędy, ale jej występowanie sygnalizuje, że w innych eksperymentach być może pojawiają się mniejsze wartości.

Zauważmy, że najlepsze otrzymywane wyniki dla parametrów (DC) sugerują, że optymalne sterowanie jest nieciągłe. Jeśli jest tak w rzeczywistości, to zapewne wszystkie nasze przybliżenia obarczone są pierwotnym błędem aproksymacji, powodowanym faktem, że prawie na pewno optymalny punkt nieciągłości nie należy do naszej siatki dyskretyzacji. Zauważmy też, że użyte przez nas metody, w szczególności aproksymacja sterowania przez funkcję kawałkami stałą, oraz dostosowanie długości kroku h algorytmu R-K do długości przedziału dyskretyzacji, mogą sku-

tecznie ograniczać skutki możliwej nieciągłości optymalnego sterowania. Wskazywać na to mogą na przykład zauważalnie lepsze wyniki dyskretyzacji kawałkami stałej w porównaniu do dyskretyzacji kawałkami liniowej, które możemy zaobserwować na Tabeli 3.5.

Na koniec należy podkreślić znaczenie zaimplementowania funkcji obliczającej gradient $\hat{J}$. Pozwoliło to znacząco zmniejszyć liczbę obliczeń funkcjonału celu $\hat{J}$. Asymptotyczna liczba obliczeń $\hat{J}$ wykonywanych w jednej iteracji optymalizacji spadła z liniowej wzgl. liczby parametrów do stałej. Przykładowo w eksperymentach zawierających 400 parametrów i wymagających 1000 iteracji, pozwoliło no przyspieszyć obliczenia ok. 100 razy.

Rozdział 4

Podsumowanie

W tej pracy sprowadziliśmy problem optymalnej strategii podawania leku do problemu optymalizacji nieliniowej z ograniczeniami, korzystając z metody „direct single shooting". Jedną z istotnych cech podejścia było opracowanie procedury obliczającej (dokładny) gradient funkcji celu, z pominięciem aproksymacji różnicami skończonymi, która jest kosztowna obliczeniowo i może skutkować istotnymi błędami przybliżenia. Przeprowadziliśmy też testy potwierdzające poprawność naszej implementacji, przez porównanie z różnicową aproksymacją gradientu.

Przeprowadzone eksperymenty pozwoliły ustalić, że w przypadku tego zadania potencjalnie lepsze wyniki udaje się osiągać przy użyciu algorytmu *lm_feasible* niż *sqp* oraz przybliżaniu sterowania funkcją kawałkami stałą niż kawałkami liniową. Oszacowaliśmy też optymalną, dla tego problemu, długość kroku dyskretyzacji algorytmu R-K na 0.1, oraz wartość parametru tolerancji optymalizatora na 10^{-11}. Omówiliśmy ponadto możliwe zagrożenia dla naszego rozwiązania, takie jak potencjalna nieciągłość optymalnego sterowania, brak zbieżności globalnej, czy możliwa znaczna odległość wyniku optymalizacji od minimum lokalnego.

Po przeanalizowaniu otrzymanych wyników, dochodzimy do wniosku, że wyglądają one obiecująco, ale nie nadają się jeszcze do wykorzystania w praktyce. Zastosowana metoda wydaje się mieć potencjał, ale dalsza praca jest potrzebna aby zweryfikować jakość i optymalność uzyskanych wyników. Niewątpliwie warto też sprawdzić alternatywne podejścia, jak np. metodę „direct collocation". Mogą one umożliwić uzyskanie lepszych wyników, oraz dać pewien punkt odniesienia co do jakości uzyskanych przez nas wyników.

Bibliografia

[1] P. Bajger, M. Bodzioch, and U. Foryś. Overcoming acquired chemotherapy resistance: insights from mathematical modelling. *niepublikowany manuskrypt*.

[2] P. Bajger, M. Bodzioch, and U. Foryś. *Role of Cell Competition in Acquired Chemotherapy Resistance*, pages 132–141. 01 2016.

[3] P. Bajger, M. Bodzioch, and U. Foryś. Overcoming acquired chemotherapy resistance: insights from mathematical modelling. *DISCRETE AND CONTINUOUS DYNAMICAL SYSTEMS SERIES B*, 24, 05 2019.

[4] P. Boggs and J. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 01 1995.

[5] M. Diehl and S. Gros. Numerical optimal control. `https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf`, 2017. Accessed: 01-08-2020.

[6] C. T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999.

[7] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59:849–904, 01 2017.

[8] M. Patterson, W. Hager, and A. Rao. A ph mesh refinement method for optimal control. *Optimal Control Applications and Methods*, 36, 02 2014.

[9] A. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135, 01 2010.

[10] O. Till et al. Optim: Non-linear optimization toolkit for GNU Octave. `https://octave.sourceforge.io/optim/package_doc/`, 2019.