Optymalna strategia podawania leku

Tomasz Kanas

5 września 2020

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:

$$\begin{aligned} V_1'(t) &= \lambda_1 V_1 F\left(\frac{V_1+\alpha_{12}V_2}{K}\right) - \beta_1 V_1 g(t), \\ V_2'(t) &= \lambda_2 V_2 F\left(\frac{V_2+\alpha_{21}V_1}{K}\right) - \beta_2 V_2 g(t), \\ K'(t) &= -\mu K + (V_1+V_2) - d(V_1+V_2)^{2/3}K - \beta K g(t) \end{aligned} \tag{1}$$

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

$$V_1(0) = V_{10},\ V_2(0) = V_{20},\ K(0) = K_0 \tag{2}$$

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) minimalizują funkcjonał:

$$J(g, V_1, V_2, K) = \int_0^T V_1(t) + V_2(t)dt + \omega \int_0^T G\left(\frac{V_2(t)-V_1(t)}{\epsilon}\right)dt \tag{3}$$

gdzie

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

Problem ten w literaturze 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:

$$\begin{aligned} \dot{y}(t) &= f(t, y, g), \\ y(0) &= y_0 = (y_{10}, y_{20}, y_{30})^T, \end{aligned} \tag{4}$$

minimalizujące funkcjonał

$$J(g, y) = \int_0^T y_1(t) + y_2(t) dt + \omega \int_0^T G\left(\frac{y_2(t) - y_1(t)}{\epsilon}\right) dt, \tag{5}$$

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

$$\begin{aligned} f_1(t, y, g) &= \lambda_1 y_1 F\left(\frac{y_1 + \alpha_{12} y_2}{y_3}\right) - \beta_1 y_1 g(t), \\ f_2(t, y, g) &= \lambda_2 y_2 F\left(\frac{y_2 + \alpha_{21} y_1}{y_3}\right) - \beta_2 y_2 g(t), \\ f_3(t, y, g) &= -\mu y_3 + (y_1 + y_2) - d(y_1 + y_2)^{2/3} y_3 - \beta y_3 g(t). \end{aligned} \tag{6}$$

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 (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 (4) oznacza sekwencję równań różniczkowych postaci

$$\begin{aligned} \dot{y}|_{(\xi_i, \xi_{i+1})}(t) &= f|_{(\xi_i, \xi_{i+1})}(t, y, g) \\ y(\xi_i) &= \lim_{t \to \xi_i^-} y(t) \end{aligned} \qquad \text{gdzie } i \in \{0, \ldots, n\}, \ \xi_0 = 0, \ \xi_{n+1} = T \tag{7}$$

Zwróćmy jeszcze uwagę na fakt, że funkcja $F(x) = -\ln(x)$ posiada osobliwość w 0, więc prawa strona (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 (6) występuje dzielenie przez y_3, więc dla $y_3(t) = 0$ prawa strona (4) także nie jest dobrze określone.

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 $[0, T]$:

$$0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T \tag{8}$$

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:

$$\hat{g}(t) = g_i \text{ gdy } t \in [t_i, t_{i+1}) \quad \text{(sterowanie kawałkami stałe)} \tag{9}$$

albo

$$\hat{g}(t) = \frac{(t_{i+1} - t)g_i + (t - t_i)g_{i+1}}{t_{i+1} - t_i} \text{ gdy } t \in [t_i, t_{i+1}) \quad \text{(sterowanie kawałkami liniowe)} \tag{10}$$

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 (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:

$$\begin{aligned} k_1 &= f(t_n + mh, \hat{y}(t_n + mh), \hat{g}), \\ k_l &= f(t_n + c_l h, \hat{y}(t_n + mh) + h \sum_{i=1}^{l-1} a_{li} k_i, \hat{g}), \quad l = 2, \ldots, r \\ \hat{y}(t_n + (m+1)h) &= \hat{y}(t_n + mh) + h \sum_{i=1}^{r} b_i k_i, \end{aligned} \tag{11}$$

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

Zostało już tylko przybliżyć funkcjonał celu (5). Zapiszmy go w postaci

$$J(y) = \int_0^T j(y(t))dt, \tag{12}$$

gdzie

$$j(y) = y_1 + y_2 + \omega G\left(\frac{y_2 - y_1}{\epsilon}\right). \tag{13}$$

Wtedy ogólny wzór na kwadraturę ze stałym krokiem h przybliżającą (12) to

$$Q(\hat{y}) = h \sum_{i=0}^{N} \alpha_i j(\hat{y}(ih)), \tag{14}$$

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

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

$$\hat{J}(g_0, \ldots, g_n) = Q(\hat{y}), \tag{15}$$

ponieważ $\hat{y}$ jest jednoznacznie wyznaczony przez $\hat{g}$ za pomocą (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*

$$\hat{J}(g_0, \ldots, g_n), \tag{16}$$

i spełniające

$$\forall_{i \in \{0, \ldots, n\}} 0 \le g_i \le g_{\max}. \tag{17}$$

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 [4] i [8].

1.3 Alternatywne podejście

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

$$0 = t_0 < t_1 < \ldots < t_n = T. \tag{18}$$

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

$$\dot{y}(t) = f(t, y(t), g(t)). \tag{19}$$

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

$$\int_{t_i}^{t_{i+1}} \dot{y}(t)dt = \int_{t_i}^{t_{i+1}} f(t, y(t), g(t))dt \tag{20}$$

$$y_{i+1} - y_i \simeq \frac{1}{2}(t_{i+1} - t_i)(f(t_{i+1}, y_{i+1}, g(t_{i+1})) - f(t_i, y_i, g(t_i))). \tag{21}$$

Powyższy wzór dodaje się do ograniczeń optymalizatora jako ograniczenie równościowe. W ten sposób RRZ (19) zostaje w przybliżeniu wymuszone w rozwiązaniu.

Ograniczenia występujące w sformułowaniu oryginalnego problemu można zwykle bez problemu przekształcić na ograniczenia na 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 [5], występuje też w [4] i [8].

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.

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 (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 wymagało zapewne 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 tego powodu będziemy korzystać z tej techniki gdzie to tylko możliwe.

Aby sprowadzić problem optymalnego sterowania do problemu optymalizacji nieliniowej musimy obliczyć przybliżone sterowanie (9) lub (10), przybliżyć rozwiązanie równania różniczkowego (11) a następnie za pomocą uzyskanego rozwiązania obliczyć kwadraturę (14) przybliżającą funkcję celu (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 (9) lub (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 (9) lub (10):

$$\frac{\partial \hat{g}}{\partial g_i}(t) = \begin{cases} 1 \text{ gdy } t \in [t_{i-1}, t_i) \\ 0 \text{ w.p.p.} \end{cases} \tag{22}$$

albo

$$\frac{\partial \hat{g}}{\partial g_i}(t) = \begin{cases} \frac{t_i - t}{t_i - t_{i-1}} \text{ gdy } t \in [t_{i-1}, t_i) \\ \frac{t - t_i}{t_{i+1} - t_i} \text{ gdy } t \in [t_i, t_{i+1}) \\ 0 \text{ w.p.p.} \end{cases} \tag{23}$$

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 gradientu też jest zwektoryzowana.

2.2 Równanie różniczkowe

Aby przybliżyć rozwiązanie równania (1) należy zaimplementować funkcję $f(t, y, g)$, a następnie bezpośrednio zaimplementować odpowiednią metodę Rungego-Kutty (11). Wzory te implementuje się bezpośrednio.

Liczenie pochodnych $\frac{\partial \hat{y}}{\partial g_i}$ polega na zróżniczkowaniu wzorów (11):

$$\begin{aligned}
\frac{\partial k_1}{\partial g_i} &= D_f \cdot \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + \frac{\partial f}{\partial g_i}(t_n + mh, \hat{y}(t_n + mh), \hat{g}) \\
\frac{\partial k_l}{\partial g_i} &= D_f \cdot \left(\frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + h \sum_{j=1}^{l-1} a_{lj} \frac{\partial k_i}{\partial g_j} \right) + \frac{\partial f}{\partial g_i}(t_n + mh, \hat{y}(t_n + mh), \hat{g}) \\
\frac{\partial \hat{y}}{\partial g_i}(t_n + (m+1)h) &= \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + h \sum_{j=1}^{r} b_j \frac{\partial k_j}{\partial g_i}
\end{aligned} \tag{24}$$

gdzie

$$D_f = \left[\frac{\partial f_i}{\partial y_j} \right]_{i,j=1,\ldots,3} (t_n + ah, \hat{y}(t_n + mh), \hat{g}) \tag{25}$$

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 w trakcie iteracji będzie spamiętywać wyliczone wartości w tych punktach. Dla wygody założymy, że punkty podane będą w kolejności rosnącej, oraz odległości między sąsiednimi punktami będą podzielne przez długość kroku h.

2.3 Funkcja celu

Przybliżoną funkcję celu (14) można zaimplementować za pomocą iloczynu skalarnego:

$$Q(\hat{y}) = (h(\alpha_0), \ldots, h(\alpha_N))^T \cdot j(\hat{y}(0, h, 2h, \ldots, T)) \tag{26}$$

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:

$$\left[\frac{\partial J}{\partial g_i} \right]_{i=0,\ldots n} = (h(\alpha_0), \ldots, h(\alpha_N))^T \cdot \left[\frac{\partial j(\hat{y}(hi))}{\partial g_k} \right]_{i=0,\ldots N, k=1,\ldots n} \tag{27}$$

gdzie

$$\frac{\partial j(y(t))}{\partial g_i} = \frac{\partial y_1(t)}{\partial g_i} + \frac{\partial y_2(t)}{\partial g_i} + \omega G' \left(\frac{y_2(t) - y_1(t)}{\epsilon} \right) \frac{\frac{\partial y_2(t)}{\partial g_i} - \frac{\partial y_1(t)}{\partial g_i}}{\epsilon}, \quad G'(x) = \frac{1}{2\cosh^2(x)} \tag{28}$$

3 Eksperymenty

W eksperymentach, do przybliżania rozwiązania równania (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 (11) podane na tabeli Butchera:

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array} \tag{29}$$

Do przybliżania funkcjonału celu (3) użyjemy kwadratury trapezów. Kwadratura (14) wyraża się więc przez:

$$Q(\hat{y}) = h\left(\frac{1}{2}(j(\hat{y}(0)) + j(\hat{y}(T))) + \sum_{i=1}^{N-1} j(\hat{y}(ih))\right) \tag{30}$$

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 $\alpha_{12},\ \alpha_{21},\ \omega$ 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. Optymalizator FMINICON na tym środowisku umożliwia użycie jednego z dwóch algorytmów do optymalizacji nieliniowej:

- *lm_feasible* oznaczany w skrócie *lm*
- *sqp*

Jedną z różnic między tymi algorytmami jest to, że *lm_feasable* zachowuje ograniczenia w trakcie optymalizacji, natomiast *sqp* wymusza ograniczenia jedynie na koniec procesu optymalizacji. W eksperymentach przetestowane zostały oba z tych algorytmów.

3.1.3 Metody dyskretyzacji

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

- P_0: wykorzystująca splajny kawałkami stałe, zob. (22)
- P_1: wykorzystująca splajny kawałkami liniowe, zob. (23)

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 $t_n = \begin{cases} n \text{ if } n \leq 25 \\ 25 + 0.1 \cdot (n - 25) \text{ if } 25 < n \leq 525 \\ 75 + (n - 525) \text{ if } n > 525 \end{cases}$
- N_{kon}: Niejednorodna, gęstsza na końcach $t_n = \begin{cases} 0.5 \cdot n \text{ if } n \leq 100 \\ 50 + (n - 100) \text{ if } 100 < n \leq 200 \\ 150 + 0.5 \cdot (n - 200) \text{ if } n > 200 \end{cases}$

3.1.4 Krok dyskretyzacji h metody R-K rozwiązania (4)

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

3.1.5 Sterowania startowe

Ważnym elementem zadania optymalizacji nierzadko jest znalezienie odpowiedniego punktu startowego. Niektóre metody optymalizacji działają tak, aby móc uciec z optimum lokalnego w poszukiwaniu optimum globalnego. Jest to bardzo przydatne w zadaniach w których nie znamy dobrego punktu startowego. Czasem jednak wiedza ekspercka pozwala ustalić dobry punkt startowy który wystarczy lekko poprawić aby uzyskać optimum. W naszym przypadku nie znamy dobrych kandydatów na sterowanie startowe, poza sugestią, że sterowanie optymalne może być nieciągłe. Na użytek eksperymentów przetestujemy więc zarówno nieskąplikowane sterowania startowe, jak i skożystamy z sugestii i przetestujemy sterowania które mają szansę być bliskie optymalnemu. W rozdziale 3.3 napisaliśmy jak szukaliśmy tych punktów startowych.

- $g_0 \equiv 0$
- $g_3 \equiv g_{\max} = 3$
- $g_{\alpha,\beta,\gamma} = \alpha \cdot \mathbb{1}_{[t_0,\gamma)} + \beta \cdot \mathbb{1}_{[\gamma,T)}$. Ograniczymy się do: $g_{0,0.55,42.5}$, $g_{0.07,0.59,48.2}$.

Oznaczane „start" w tabelach z wynikami.

3.1.6 Warunek stopu

Dla algorytmu *lm* jednym z warunków zakończenia optymalizacji jest zbyt niewielka względna poprawa funkcji celu. Minimalna wartość tej poprawy jest ustawiana w FMINCON parametrem „TolFun". Podobnie dla algorytmu *sqp* istnieje w Octave parametr „octave_sqp_tolerance", który odpowiada za różne kryteria stopu (niestety dokumentacja nie precyzuje za które dokładnie). Oznaczmy oba te parametry przez „Tol". Przetestujemy wartości Tol $\in \{10^{-6}, 10^{-9}, 10^{-11}, 10^{-13}\}$. Dla porównania domyślną wartością dla TolFun jest 10^{-4}, natomiast dla octave_sqp_tolerance jest pierwiastek z precyzji arytmetyki, czyli ok 10^{-8}.

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 $\#\hat{J}$). Ponadto, aby móc ocenić jak blisko jest rozwiązaniu do faktycznego minimum lokalnego, będziemy podawać normę gradientu w punkcie zakończenia optymalizacji $\|G\|_1$, oraz nnormę względną, czyli stosunek normy gradientów w punkcie zakończenia optymalizacji i w punkcie startowym: $\frac{\|G\|_1}{\|G_0\|_1}$.

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 gradientu będziemy podawać wartość $10^{-5}\|G\|_1$ zaokrągloną do 2 miejsc po przecinku. Wartość stosunku norm gradientów $\frac{\|G_0\|_1}{\|G\|_1}$ będziemy zaokrąglać do 1 miejsca po przecinku.

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 1. Celem tego eksperymentu jest sprawdzenie poprawności implementacji, więc ograniczymy liczbę iteracji optymalizatora do 20, oraz pominiemy podawanie norm gradientów.

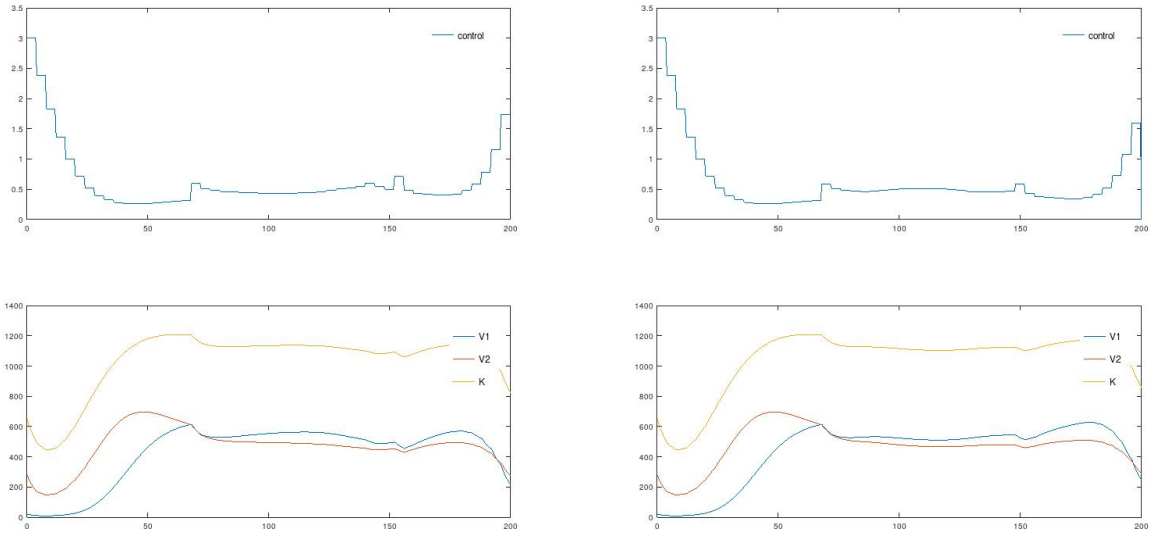
Gradient	Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
FD	(CC)	*lm*	P_0	S_4	0.1	g_0	3.26	20	38
wg. rozdz. 2.1, 2.2	(CC)	*lm*	P_0	S_4	0.1	g_0	3.25	20	37

Tabela 1: Test implementacji gradientu

Rozwiązania uzyskane obiema metodami są bardzo podobne, o czym świadczą wyniki zaprezentowane na Tabeli 1 i Rysunku 1. Ponadto nasza implementacja osiągnęła minimalnie lepszy wynik.

(a) Rozwiązanie przy użyciu różnic skończonych

(b) Rozwiązanie z liczeniem gradientu



Rysunek 1: Rozwiązania dla przypadku testowego

Dodatkowo policzymy gradient G, w punkcie będącym sterowaniem zaprezentowanym na wykresie (1b), obliczony przy użyciu naszej implementacji, oraz gradient G_τ uzyskany za pomocą metody FD dla różnych wartości kroku różnic skończonych

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

Następnie dla wszystkich τ obliczymy normę różnicy: $\|G - G_\tau\|_1$. Wyniki tych obliczneń przedstawiamy na Tabeli 2.

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

Tabela 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, wartości funkcji w punktach odległych od τ są bardzo bliskie sobie, więc obliczanie ich różnicy obarczone jest znacznym błędem numerycznym.

Warto tu jeszcze dodać, że liczba wywołań $\hat{J}$ podana na Tabeli 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

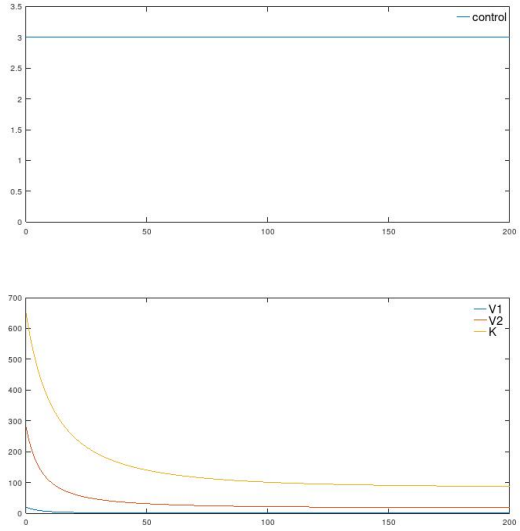
y_2 wartość i pochodna wyrażenia $G((y_2(t) - y_1(t))/\epsilon)$, gdzie $G(x) = \frac{1+\tanh(x)}{2}$, zmienia się bardzo szybko w czasie, co powoduje znaczne błędy przy zbyt dużym kroku metody różnic skończonych, co możemy też zaobserwować na Tabeli 2, w postaci dużej normy różnicy gradientów, gdy krok różnic skończonych to 10^{-3}, czyli wartość domyślna w FMINICON użyta też 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)

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|G\|_1$	$\frac{\|G\|_1}{\|G_0\|_1}$
(CC)	*lm*	P_0	S_1	0.1	g_0	10^{-6}	2.2	3	8	0.32	0.06
(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	0.32	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	0.8	0.148
(CC)	*lm*	P_0	N_{kon}	0.1	g_0	10^{-9}	2.63	87	92	0.8	0.147
(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	0.83	0.153
(CC)	*lm*	P_1	N_{kon}	0.1	g_0	10^{-9}	2.66	24	29	0.83	0.153
(CC)	*lm*	P_0	S_1	0.1	g_3	10^{-6}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_0	S_1	0.1	g_3	10^{-9}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-6}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-9}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_0	N_{kon}	0.1	g_3	10^{-6}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_0	N_{kon}	0.1	g_3	10^{-9}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_1	S_1	0.1	g_3	10^{-6}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_1	S_1	0.1	g_3	10^{-9}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_3	10^{-6}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_3	10^{-9}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_1	N_{kon}	0.1	g_3	10^{-6}	2.14	1	2	0.26	1.0
(CC)	*lm*	P_1	N_{kon}	0.1	g_3	10^{-9}	2.14	1	2	0.26	1.0
(CC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	10^{-6}	2.14	9	10	0.26	0.47

Tabela 3: Eksperymenty z parametrami (CC) dla różnych wyborów aproksymacji, siatki i startu

Wyniki dla parametrów (CC) są dość jednoznaczne. Najlepszym znalezionym rozwiązaniem jest $g \equiv g_{\max} = 3$, które zaprezentowano na Rysunku 2. Wszystkie testy zaczynające z tego sterowania zbiegły do niego po jednej iteracji, patrz Tabela 3. Widzimy też z tej tabeli, że testy korzystające z siatki niejednorodne wypadają zauważalnie gorzej od pozostałych. Widzimy też, że zmniejszenie tolerancji (Tol) poprawiło wartość $\hat{J}$ w bardzo małym stopniu, a znacząco zwiększyło liczbę iteracji.



Rysunek 2: Rozwiązanie dla parametrów (CC)

Algorytm *sqp* jest jedynym który zbiegł z zerowego sterowania do sterowania optymalnego.

Warto jeszcze wyjaśnić dlaczego dla sterowania startowego g_3 optymalizacja kończy się po jednej iteracji pomimio niezerowego gradientu G. Jest to powodowane ograniczeniem na wartość sterowania (17): $g_i \leq g_{\max} = 3$ które jest spełnione równościowo dla każdego parametru sterowania g_3. Rzeczywistym warunkiem stopu optymalizatora jest zerowanie się gradientu Lagrangianu wynikającego z warunków KKT, które opisane są np. w [6]. W naszym przypadku ograniczenia (17) mają prostą postać, więc gradient wspomnianego Lagrangianu w naszym szczególnym przypadku to

$$\nabla L = (l_i)_{i=0,\dots,n}^{T} \quad \text{gdzie} \quad l_i = \begin{cases} 0 \text{ if } g_i = 0 \wedge \frac{\partial \hat{J}}{\partial g_i} < 0 \\ 0 \text{ if } g_i = g_{\max} \wedge \frac{\partial \hat{J}}{\partial g_i} > 0 \\ \frac{\partial \hat{J}}{\partial g_i} \text{ otherwise} \end{cases} \tag{31}$$

i ta wartość, jest dla sterowania g_3 zerowa.

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

3.2.3 Sterowanie startowe

Wyniki badające różne sterowania startowe dla różnych algorytmów i wartości tolerancji przedstawione zostały na Tabeli 4. Pozwalają one na wyciągnięcie kilku ciekawych wniosków. Po pierwsze widzimy, że dla Tol = 10^{-6} wynikowe gradienty są bardzo duże, w szczególności prawie zawsze większe od gradientów startowych. Oznacza to, że rozwiązania te są zapewne dalekie od optimów lokalnych, więc ta wartość tolerancji jest zapewne zbyt duża i dalej będziemy skupiać się na badaniu niższych wartości tolerancji. Ponadto wygląda na to, że zmniejszenie tolerancji nie poprawiło

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|G\|_1$	$\frac{\|G\|_1}{\|G_0\|_1}$
(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	0.11	0.035
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	10^{-6}	3.3	18	205	32.34	10.81
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	10^{-9}	3.3	18	205	32.34	10.81
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-6}	4.07	1	2	0.13	1.0
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	10^{-9}	4.07	1	2	0.13	1.0
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_3	10^{-6}	4.07	1	2	0.13	1.0
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_3	10^{-9}	4.07	1	2	0.13	1.0
(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	0.79	0.391
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0,0.55,42.5}$	10^{-6}	2.71	10	130	3.63	1.804
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0,0.55,42.5}$	10^{-9}	2.71	10	130	3.63	1.804
(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	0.18	0.058
(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

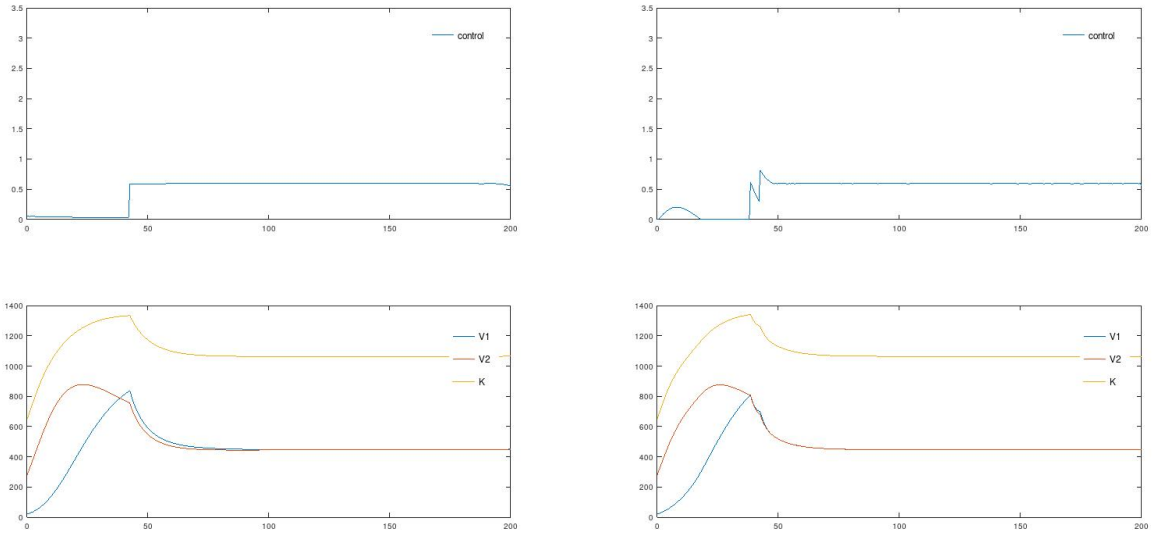
Tabela 4: Eksperymenty badające sterowanie startowe

wyników algorytmu *sqp*. Niestety dokumentacja tego algorytmu nie specyfikuje jakie są warunki zakończenia optymalizacji i na co wpływa parametr tolerancji, więc nie znamy powodu takiego zachowania.

Z Tabeli 4 możemy też wywnoskować, że przy parametrach (DC), podobnie jak przy parametrach (CC), sterowanie g_3 (czyli stale równe $3 = g_{\max}$) wydaje się być minimum lokalnym, ale w tym przypadku znacznie gorszym od pozostałych rezultatów. Stosuje się tu ta sama uwaga do wartości gradientu co przy eksperymencie z parametrami (CC).

Najlepsze wyniki osiągają są dla sterowania startowe $g_{0,0.55,42.5}$ 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 $\frac{\|G\|_1}{\|G_0\|_1}$ i tutaj zauważalnie lepiej wypada sterowanie startowe $g_{0.07,0.59,48.2}$. Warto też zauważyć, że pomimo podobnych wartości funkcj $\hat{J}$ dla tych czterech przypadków testowych, wynikowe sterowania zauważalnie się dla nich różnią, co pokazano na Rysunkach 3 i 4.

Widzimy też, że dla startu g_0 przy Tol = 10^{-9} i algorytmie *lm* udało się osiągnąć niską normę gradientu, 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 wynikowym sterowaniem dla tego przypadku, a wyżej omawianymi wynikami, które możemy zaobserwować porównując wykresy przedstawione na Rysunkach 5, 3b i 4b. Z tego powodu nie będziemy dalej zajmować się tym sterowaniem startowym.



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

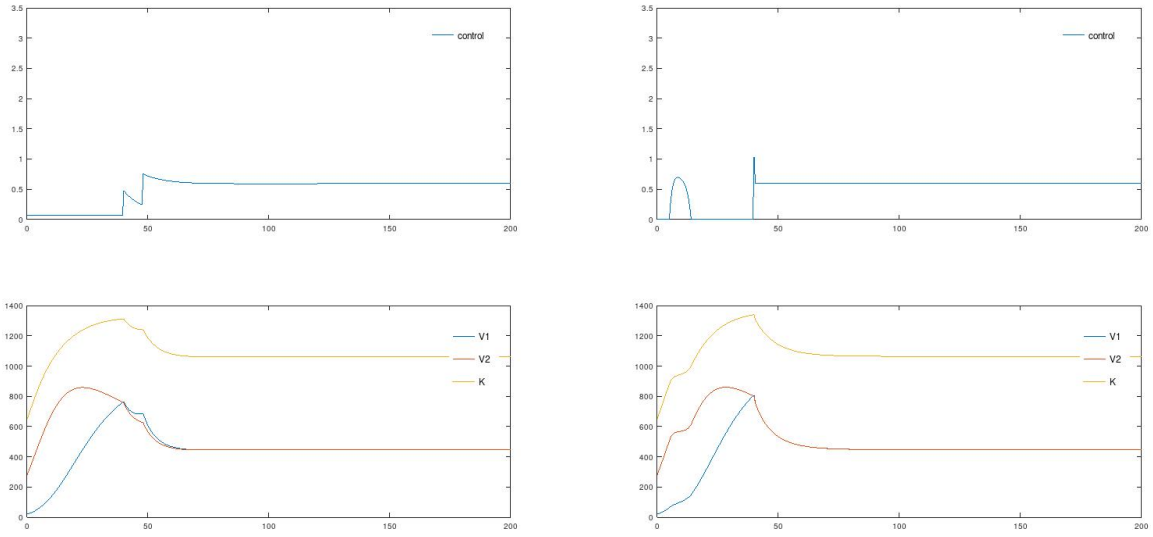
Rysunek 3: Rozwiązania dla startu $g_{0,0.55,42.5}$

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|G\|_1$	$\frac{\|G\|_1}{\|G_0\|_1}$
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.18	0.058
(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)	*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_1	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.76	5	101	7.06	1.556

Tabela 5: Eksperymenty badające metodę dyskretyzacji

3.2.4 Metoda dyskretyzacji

Jak widzimy na Tabeli 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 zapewne spowodowane brakiem znalezienia lepszego sterowania w otoczeniu sterowania startowego, co z kolei może być powodowane błędami wynikającymi z przybliżania sterowania nieciągłego sterowaniem kawałkami liniowym.



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

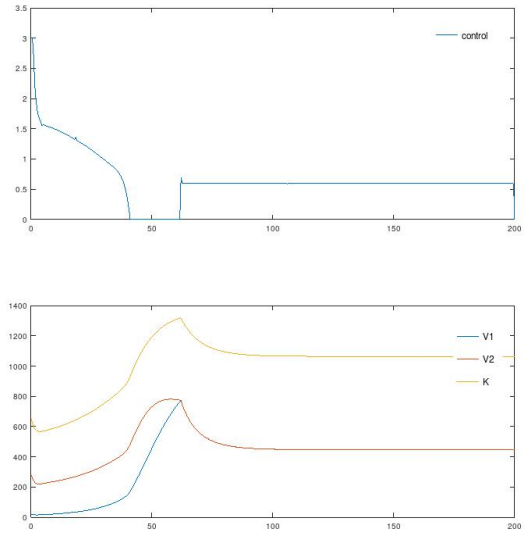
Rysunek 4: Rozwiązania dla startu $g_{0.07,0.59,48.2}$

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|G\|_1$	$\frac{\|G\|_1}{\|G_0\|_1}$
(DC)	*lm*	P_0	S_1	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.7	620	1110	0.39	0.13
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.18	0.058
(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

Tabela 6: Eksperymenty badające siatkę dyskretyzacji

3.2.5 Siatka dyskretyzacji

Zgodnie z oczekiwaniami gęstsza siatka jednorodna osiąga lepsze wyniki dla algorytmu *lm*, co widzimy na Tabeli 6, natomiast siatka niejednorodna okazuje się osiągać gorsze wyniki od 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 5: Rozwiązanie dla startu g_0 i Tol = 10^{-9}

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|G\|_1$	$\frac{\|G\|_1}{\|G_0\|_1}$
(DC)	*lm*	P_0	$S_{0.5}$	0.5	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	974	1741	0.17	0.051
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-9}	2.69	811	1439	0.18	0.058
(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

Tabela 7: Eksperymenty badające krok dyskretyzacji h

3.2.6 Krok dyskretyzacji h

Wyniki zaprezentowane na Tabeli 7 są dość nieoczekiwane. Wygląda na to, że zmniejszanie kroku dyskretyzacji h metody R-K pogarsza wyniki. Ponadto widzimy, że gdy h = 0.02, to algorytm *lm* wydaje się nie być w stanie znaleźć lepszego punktu w otoczeniu punktu startowego.

3.2.7 Warunek stopu

Na podstawie wyników z Tabeli 8 optymalnym warunkiem stopu wydaje się Tol = 10^{-9}, dla Tol = 10^{-6} norma $\|G\|_1$ jest duża, więc rozwiązanie uzyskane w ten sposób jest raczej odległe od minimum lokalnego. Dla niższych wartości Tol niż 10^{-9} nie widzimy istotnej poprawy wyników, natomiast znacznie rośnie liczba potrzebnych iteracji.

Ten eksperyment potwierdza nasze podejżenia, że dla algorytmu *sqp* parametr tolerancji nie ma wpływu na wynik w naszych eksperymentach, a wyniki osiągane przez ten algorytm wydają się być dalekie od minimum zarówno lokalnego jak i globalnego.

Param.	algorytm	aproks.	siatka	h	start	Tol	$\hat{J}$	iter	$\#\hat{J}$	$\|G\|_1$	$\frac{\|G\|_1}{\|G_0\|_1}$
(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	0.18	0.058
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-11}	2.69	1314	2313	0.17	0.058
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.07,0.59,48.2}$	10^{-13}	2.69	1401	2466	0.18	0.058
(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

Tabela 8: Eksperymenty badające wpływ warunku stopu

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^{-9}. Są one zaprezentowane na Rysunku 4b.

3.3 Metoda znajdowania sterowania startowego

Aby znaleźć sterowanie startowe które pozwoli uzyskać dobre wyniki posłużyliśmy się pewną heurystyczną metodą. Jak już pisaliśmy w rozdziale 3.1.5, spodziewaliśmy się, że optymalny wynik będzie nieciągły. Zauważmy, że sterowanie kawałkami stałe z jednym punktem nieciągłości można sparametryzować trzema wartościami, co nawet sugeruje notacja $g_{\alpha,\beta,\gamma}$. Aby uzyskać jak najlepszy punkt startowy tej postaci dostosowaliśmy naszą implementację rozwiązania problemu do tej parametryzacji, to znaczy optymalizowaliśmy problem

$$\min_{\alpha,\beta,\gamma} \check{J}(\alpha,\beta,\gamma) := \hat{J}(\alpha,\beta) \quad \text{z dyskretyzacją } t = (t_0,\gamma,T) \tag{32}$$

i z ograniczeniami

$$0 \le \alpha,\beta \le g_{\max}, \quad t_0 < \gamma < T. \tag{33}$$

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 różnic skończonych. Aby otrzymać dobre rezultaty trzeba było jeszcze uważnie ustawić parametry 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 (11) i kwadratury (14), co może skutkować znacznym błędem w wynikach tych metod na jednym przedziale długości h, więc zastosowano niższą wartość $h = 0.05$ niż w większości eksperymentów.

Pomimo heurystyczności tej metody, udało się nią uzyskać sterowanie $g_{0.07,0.59,48.2}$, które osiągnęło najlepsze rezultaty w eksperymentach.

4 Analiza i krytyka wyników

Jak widzimy, wyniki dla różnych rodzajów parametrów znacznie się różnią. Dokładniejsze przyjrzenie się funkcjonałowi celu (5) pozwala wyjaśnić takie zjawisko. 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 2-gim zestawie parametrów parametr ω jest znacznie większy niż w 1-szym, więc w tym przypadku druga całka ma większy wpływ na wynik funkcjonału. Jak przyjrzymy się rozwiązaniu z Rysunku **??**, 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 rakowych odpowiednio podatnych na działanie leku i odpornych na niego, a wartości sterowania jako dawki leku.

4.1 Krytyka wyników

Wszystkie wyniki które udało się osiągnąć są minimami lokalnymi. Jak już zauważyliśmy, wyniki były mocno zależne od punktu startowego, nawet w przypadku algorytmu *sqp*, który powinien szukać rozwiązania bardziej globalnie. Być może zastosowanie innych algorytmów optymalizacji nieliniowej umożliwiłoby szukanie rozwiązania w sposób mniej zależący od znalezienia dobrego punktu startowego.

Niejednorodna siatka dyskretyzacji jest metodą nierzadko pozwalającą na poprawę tępa zbieżności, ale nie udało się jej tu z sukcesem zastosować. Podobnie przybliżanie sterowania za pomocą Solana wyższego rzędu też mogłoby poprawić tempo zbieżności. W literaturze rozważa też metody automatyzacji znajdowania siatki dyskretyzacji i odpowiedniego rzędu dyskretyzacji. Przykładem jest tu praca [7]. Być może zastosowanie ich umożliwiłoby uzyskanie lepszych wyników.

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 2 to 14.9, więc i ta nie jest zbyt bliska 0. Okazuje się jednak, że przy parametrach (DC), w rozwiązaniu (4) dla sterowania stale równego $g_{\max}$ minimalna wartość y_3 wynosi ok. 0.02, więc ten eksperyment może być obarczony istotnym błędem numerycznym.

Literatura

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

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

[6] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951. University of California Press.

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

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