

# Optymalna strategia podawania leku

Tomasz Kanas

30 sierpnia 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]. Model ten przedstawia rozwój nowotworu w czasie w zależności od dawkowania leku za pomocą równania różniczkowego:

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

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

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

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

$$\lambda_1, \lambda_2, \alpha_{12}, \alpha_{21}, \beta_1, \beta_2, \beta, \mu, d, V_{10}, V_{20}, K_0 \geq 0$$

są zadanymi 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  $t$ .

Zadanie polega na znalezieniu funkcji  $g : [0, T] \rightarrow [0, g_{\max}]$  oraz  $V_1, V_2, K : [0, T] \rightarrow (0, \infty)$  spełniających (1), oraz minimalizujących funkcjonal:

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

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 uprościmy notację.

**Problem 1.** *Znaleźć funkcję kawałkami ciągłą*

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

i funkcję

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

spełniające równanie różniczkowe:

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

oraz minimalizujące funkcjonal

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

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

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

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  $f$  są  $\xi_1, \dots, \xi_n$ , wtedy (4) oznacza ciąg równań różniczkowych postaci

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

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 obciążone 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 skończonej wymiarowej. W tym celu ustalimy siatkę dyskretyzacji przedziału  $[0, T]$ :

$$0 = t_0 < t_1 < \dots < t_{n-1} < t_n = T \quad (8)$$

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:

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

lub

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

i jest jednoznacznie zdefiniowane przez wartości  $g_i$  dla  $i = 0, \dots, n$ . Te wartości będą optymalizowanymi zmiennymi.

Korzystając z przybliżonej funkcji sterowania, przybliżymy rozwiązanie  $y$  równania różniczkowego (4). Użyjemy do tego metody Rungego-Kutty rzędu  $r$  ze stałym krokiem długości  $h$ , wtedy przybliżone rozwiązanie  $\hat{y}(t) \simeq y(t)$  w punkcie  $t = t_n + (m+1)h$  dla  $m = 0, \dots, \frac{t_{n+1}-t_n}{h} - 1$  wyraża się przez:

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

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

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

$$J(y) = \int_0^T j(y(t)) dt \quad (12)$$

gdzie

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

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

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

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

Zatem możemy zdefiniować przybliżoną funkcję celu jako funkcję  $g_1, \dots, g_n$ :

$$\hat{J}(g_1, \dots, g_n) = Q(\hat{y}) \quad (15)$$

ponieważ  $\hat{y}$  jest jednoznacznie wyznaczony przez  $\hat{g}$  za pomocą (11), a  $\hat{g}$  jest wyznaczone jednoznacznie przez  $g_1, \dots, g_n$ . Podsumowując, problem przybliżony to:

**Problem 2.** Znaleźć  $g_1, \dots, g_n$  minimalizujące

$$\hat{J}(g_1, \dots, g_n) \quad (16)$$

i spełniające

$$\forall_{i \in \{1, \dots, n\}} 0 \leq g_i \leq g_{\max} \quad (17)$$

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 [2] i [5].

### 1.3 Alternatywne podejście

Innym popularnym podejściem jest „direct collocation”. W tym podejściu zaczyna się tak samo od dyskretyzacji czasu:

$$0 = t_0 < t_1 < \dots < t_n = T$$

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 w powyższym podejściu, 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

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

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

$$\int_{t_i}^{t_{i+1}} \dot{y}(t) dt = \int_{t_i}^{t_{i+1}} f(t, y(t), g(t)) dt$$
$$y_{i+1} - y_i \simeq \frac{1}{2}(t_{i+1} - t_i)(f(t_{i+1}, y_{i+1}, g(t_{i+1})) - f(t_i, y_i, g(t_i)))$$

Ograniczenia występujące w sformułowaniu oryginalnego problemu można zwykle bez problemu przekształcić na ograniczenia na  $g_i$  oraz  $y_i$ . Można też zdefiniować przybliżony funkcjonal celu jako numeryczne przybliżenie oryginalnego funkcjonułu celu zastosowane do przybliżonego stanu  $\hat{y}$ .

Podejście to jest dokładnie opisane w [3], występuje też w [2] i [5].

### 1.4 Plan rozwiązania

Aby obliczyć wynik problemu przybliżonego skorzystamy ze środowiska MATLAB/Octave wraz z dostarczoną z nim optymalizatorem problemu optymalizacji nieliniowej z ograniczeniami (FMINICON). W tym celu zaimplementujemy przejście od problemu optymalnego sterowania. 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 naszym przypadku jedynie (17). Domyślnie optymalizator wyznacza gradient za pomocą różnic skończonych, co wymaga uruchomienia funkcji celu liniowo wiele razy względem liczby parametrów. Aby poprawić wydajność optymalizacji zaimplementujemy liczenie gradientu przybliżonej funkcji celu ze względu na parametry  $g_1, \dots, g_n$ .

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  $g_1, \dots, g_n$  należy zróżniczkować równanie (9) lub (10):

$$\frac{\partial \hat{g}}{\partial g_i}(t) = \begin{cases} 1 & \text{gdy } t \in [t_{i-1}, t_i) \\ 0 & \text{w.p.p.} \end{cases} \quad (18)$$

lub

$$\frac{\partial \hat{g}}{\partial g_i}(t) = \begin{cases} \frac{t_i - t}{t_i - t_{i-1}} & \text{gdy } t \in [t_{i-1}, t_i) \\ \frac{t - t_i}{t_{i+1} - t_i} & \text{gdy } t \in [t_i, t_{i+1}) \\ 0 & \text{w.p.p.} \end{cases} \quad (19)$$

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):

$$\begin{aligned}\frac{\partial k_1}{\partial g_i} &= D_f \cdot \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + \frac{\partial f}{\partial g_i}(t_n + mh, \hat{y}(t_n + mh), \hat{g}) \\ \frac{\partial k_l}{\partial g_i} &= D_f \cdot \left( \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + h \sum_{j=1}^{l-1} a_{lj} \frac{\partial k_j}{\partial g_i} \right) + \frac{\partial f}{\partial g_i}(t_n + mh, \hat{y}(t_n + mh), \hat{g}) \\ \frac{\partial \hat{y}}{\partial g_i}(t_n + (m+1)h) &= \frac{\partial \hat{y}}{\partial g_i}(t_n + mh) + h \sum_{j=1}^r b_j \frac{\partial k_j}{\partial g_i}\end{aligned}\quad (20)$$

gdzie

$$D_f = \left[ \frac{\partial f_i}{\partial y_j} \right]_{i,j=1,\dots,3}(t_n + ah, \hat{y}(t_n + mh), \hat{g}) \quad (21)$$

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:

$$Q(\hat{y}) = h(\alpha_0, \dots, \alpha_N) \cdot j(\hat{y}(0, h, 2h, \dots, T))^T \quad (22)$$

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:

$$\left[ \frac{\partial J}{\partial g_i} \right]_{i=0,\dots,n} = (h(\alpha_0), \dots, h(\alpha_N)) \cdot \left[ \frac{\partial j(\hat{y}(hi))}{\partial g_k} \right]_{i=0,\dots,N, k=1,\dots,n} \quad (23)$$

gdzie

$$\frac{\partial j(y(t))}{\partial g_i} = \frac{\partial y_1(t)}{\partial g_i} + \frac{\partial y_2(t)}{\partial g_i} + \omega G' \left( \frac{y_2(t) - y_1(t)}{\epsilon} \right) \frac{\frac{\partial y_2(t)}{\partial g_i} - \frac{\partial y_1(t)}{\partial g_i}}{\epsilon}, \quad G'(x) = \frac{1}{2 \cosh^2(x)} \quad (24)$$

## 3 Eksperymenty

W eksperymentach użyjemy wartości parametrów podanych w wyjściowej pracy ??:

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

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:

1.	$\alpha_{12} = 0.1$	$\alpha_{21} = 0.15$	$\omega = 1000$
2.	$\alpha_{12} = 0.5$	$\alpha_{21} = 0.75$	$\omega = 2000$

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. Długość kroku obierzemy na  $h = 0.1$ , co daje 2000 kroków na całym przedziale  $[t_0, T]$ . Wartości stałych  $c_l$ ,  $a_{li}$ ,  $b_i$  z wzoru (11) podane na tabeli Butchera:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & 0 & \frac{1}{2} & \\ 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \quad (25)$$

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

$$Q(\hat{y}) = h \left( \frac{1}{2}(j(\hat{y}(0)) + j(\hat{y}(T))) + \sum_{i=1}^{N-1} j(\hat{y}(ih)) \right) \quad (26)$$

Do eksperymentów użyte zostało środowisko Octave. Optymalizator FMINICON na tym środowisku umożliwia użycie jednego z dwóch algorytmów do optymalizacji nieliniowej: *lm\_feasible*, lub *sqp*. Jedną z różnic między tymi algorytmami jest to, że *lm\_feasible* zachowuje ograniczenia w trakcie optymalizacji, natomiast *sqp* wymusza ograniczenia jedynie na koniec procesu optymalizacji. W eksperymentach przetestowane zostały oba z tych algorytmów.

### 3.1 Zbiór testowy

Aby móc zbadać wpływ różnych czynników na ostateczny wynik nasz zbiór testowy będzie zawierał

- Zestawy parametrów
  - 1.
  - 2.
- Algorytmy
  - *lm\_feasible* oznaczany w skrócie *lm*
  - *sqp*
- Metody dyskretyzacji
  - Stała (18)

- Liniowa (19)
- Siatki dyskretyzacji
  - Stała krok=4 ( $S_4$ )  $t_n = 4n$
  - Stała krok=1 ( $S_1$ )  $t_n = n$
  - Stała krok=0.5 ( $S_{0.5}$ )  $t_n = 0.5 \cdot n$
  - Niejednorodna, gęstsza w środku ( $N_{sr}$ )  $t_n = \begin{cases} n & \text{if } n \leq 25 \\ 25 + 0.1 \cdot (n - 25) & \text{if } 25 < n \leq 525 \\ 75 + (n - 525) & \text{if } n > 525 \end{cases}$
  - Niejednorodna, gęstsza na końcach ( $N_{kon}$ )  $t_n = \begin{cases} 0.5 \cdot n & \text{if } n \leq 100 \\ 50 + (n - 50) & \text{if } 100 < n \leq 200 \\ 150 + 0.5 \cdot (n - 200) & \text{if } n > 200 \end{cases}$
- Kroki dyskretyzacji ( $h$ )
  - 0.5
  - 0.1
  - 0.02
- Sterowania startowe
  - $g_0$ :  $g \equiv 0$
  - $g_3$ :  $g \equiv g_{\max} = 3$
  - $g_{0,0.4}$ :  $g = 0.4 \cdot \mathbb{1}_{[42.5, T]}$
  - $g_{0,0.55}$ :  $g = 0.55 \cdot \mathbb{1}_{[42.5, T]}$
  - $g_{3,0,0.5}$ :  $g = g_{\max} \cdot \mathbb{1}_{[0, 10]} + 0.5 \cdot \mathbb{1}_{[50, T]}$

Jako, że zbiór ten jest duży, przeprowadzimy kilka eksperymentów. Każdy eksperyment będzie przeprowadzony na podzbiorze tego zbioru wybranym tak, aby zaprezentować wpływ konkretnego parametru na wyniki.

## 3.2 Wyniki eksperymentów

Wartości funkcji celu są duże, więc dla zwiększenia czytelności będziemy podawać wartości dla  $10^{-5} \hat{J}$  zaokrąglone do 2 miejsc 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. Obliczanie wyników metodą domyślną wymaga wiele czasu, więc eksperyment obejmie tylko jeden przypadek, opisany na Tabeli 1.

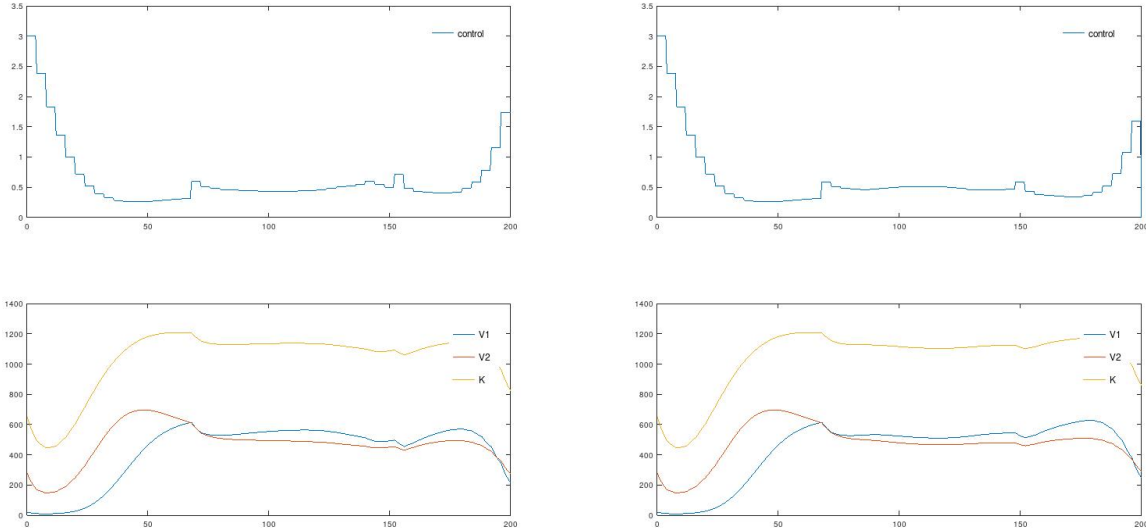


Gradient	Parametry	algorytm	dyskret.	siatka	$h$	start	wynik	#iteracji	#wywołań $\hat{J}$
nie	1.	$lm$	stała	$S_4$	0.1	$g_0$	3.26	20	38
tak	1.	$lm$	stała	$S_4$	0.1	$g_0$	3.25	20	37

Tabela 1: Test implementacji gradientu

(a) Rozwiązanie przy użyciu różnic skończonych

(b) Rozwiązanie z liczeniem gradientu



Rysunek 1: Rozwiązania dla przypadku testowego

Jak widzimy na Tabeli 1 i Rysunku 1, rozwiązania uzyskane obiema metodami są prawie identyczne, oraz nasza implementacja osiągnęła minimalnie lepszy wynik. Daje nam to dużą dozę pewności o poprawności naszej implementacji liczenia gradientu.

Warto tu jeszcze dodać, że liczba wywołań  $\hat{J}$  podana na Tabeli 1 nie uwzględnia wywołań potrzebnych do oszacowania 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 numeryczne. Zauważmy, że w okolicy punktu przecięcia się krzywych  $y_1$  i  $y_2$  wartość i pochodna wyrażenia  $G((y_2(t) - y_1(t))/\epsilon) = \frac{1+\tanh(x)}{2}$  zmienia się bardzo szybko w czasie, co powoduje znaczne błędy przy zbyt dużym kroku metody różnic skończonych.

### 3.2.2 Parametry 1.

Przypadek ten okazał się mniej skomplikowany od Parametrów 2., więc poświęcimy mu tylko jeden eksperyment.

Parametry	algorytm	dyskret.	siatka	$h$	start	wynik	#iteracji	#wywołań $\hat{J}$
1.	$lm$	stała	$S_1$	0.1	$g_0$	2.2	3	8
1.	$lm$	stała	$S_1$	0.1	$g_3$	2.14	1	2
1.	$lm$	stała	$S_{0.5}$	0.1	$g_0$	2.19	3	8
1.	$lm$	stała	$S_{0.5}$	0.1	$g_3$	2.14	1	2
1.	$lm$	stała	$N_{kon}$	0.1	$g_0$	2.64	9	14
1.	$lm$	stała	$N_{kon}$	0.1	$g_3$	2.14	1	2
1.	$lm$	liniowa	$S_1$	0.1	$g_0$	2.21	2	7
1.	$lm$	liniowa	$S_1$	0.1	$g_3$	2.14	1	2
1.	$lm$	liniowa	$S_{0.5}$	0.1	$g_0$	2.21	2	7
1.	$lm$	liniowa	$S_{0.5}$	0.1	$g_3$	2.14	1	2
1.	$lm$	liniowa	$N_{kon}$	0.1	$g_0$	2.66	2	7
1.	$lm$	liniowa	$N_{kon}$	0.1	$g_3$	2.14	1	2

Tabela 2: Eksperymenty z parametrami 1.

### 3.2.3 Metoda dyskretyzacji

Dla parametrów (??) wyniki eksperymentów są zgodne: najlepsze rezultaty osiągamy dla sterowania stale równego  $g_{\max}$ . Optymalizator zaczynając w nim nie znajduje żadnego lepszego punktu, natomiast dla wielu innych punktów startowych (n.p. dla sterowania stałego równego 0 lub 1) zbiega do sterowania niewiele różniącego się od stale równego  $g_{\max}$ . Stosowanie różnych algorytmów, siatek optymalizacji oraz metod przybliżania sterowania nie pozwoliło znaleźć istotnie różnego rozwiązania dającego obiecujące wyniki. Żaden eksperyment nie zbiegł do punktu dającego mniejszą wartość przybliżonego funkcjonału celu.

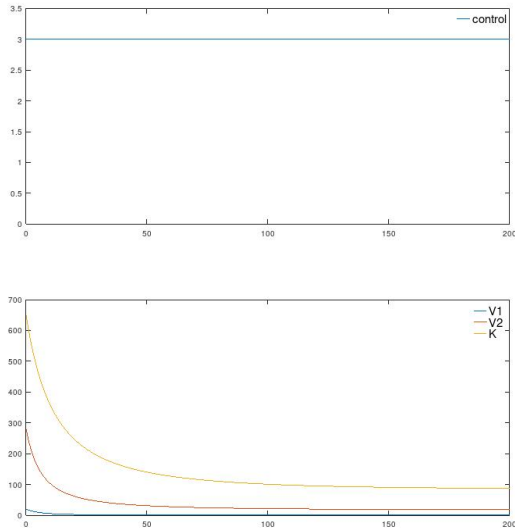
Wartość przybliżonego funkcjonału celu dla sterowania stale równego  $g_{\max}$  to ok. 213575,7. Dla porównania wartość przy braku leczenia, czyli sterowaniu stale równemu 0, to ok. 567688,3. Przybliżone rozwiązania równania (4) przedstawiono na rysunku 2a.

Dla parametrów (??) wyniki są zauważalnie inne. Sterowanie stale maksymalnie nie osiąga już dobrych rezultatów (wynik to 407071.7), natomiast najlepsze wyniki osiąga sterowanie mające na początku przedziału wartość ok. 0.05, a w pozostałych punktach wartość ok. 0.59. Funkcjonał celu dla tego sterowania osiąga wartość 272466.2. Warto jeszcze dodać, że przy tych wartościach parametrów wyniki nie były tak zgodne, w trakcie eksperymentów znaleziono wiele rozwiązań o niewiele gorszych wynikach, niektóre z nich istotnie różne od przedstawionego. Przykład takiego rozwiązania zaprezentowano na rysunku ???. Wartość funkcjonału celu dla tego rozwiązania to 295794.6, co jest o 8,6% większe od rozwiązania optymalnego.

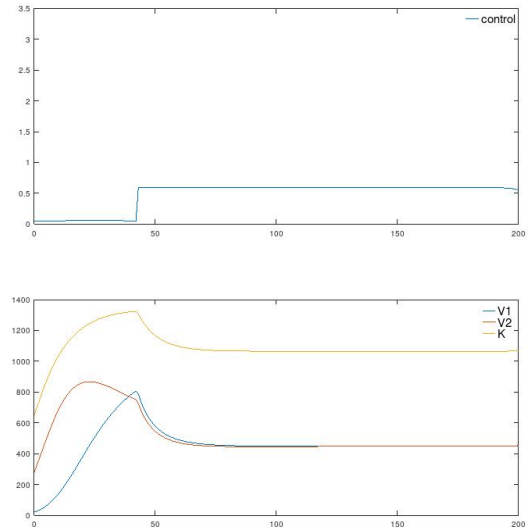
## 3.3 Analiza zastosowanych metod

Osiągnięcie przedstawionych wyników wymagało przeprowadzenia wielu eksperymentów, przetestowania różnych backendów optymalizatora, siatek optymalizacji, sterowania kawałkami stałego lub liniowego i różnych punktów startowych. Jak widzimy, dla różnych wartości parametrów (??) lub (??), wyniki mocno się różnią i każdy z tych zestawów parametrów wymagał różnego podejścia. Wspólne dla obu tych problemów było działanie siatki optymalizacji. W większości eksperymentów zastosowano siatkę jednorodną o kroku 0.5. Dalsze próby zagęszczania tej siatki nie poprawiły

(a) Rozwiązanie dla parametrów (??)



(b) Rozwiązanie dla parametrów (??)



Rysunek 2: Rozwiązania równania (1)

wyników, a tylko zwiększyły czas wykonania. Próby użycia siatki niejednorodnej przynosiły gorsze rezultaty niż te z siatką jednorodną o podobnej liczbie punktów. Jednym z powodów gorszych wyników siatki niejednorodnej mogą być fakt, że optymalizator zwykle dla takich siatek zbiegał do rozwiązania mającego znaczną różnicę wartości na przedziałach z różnymi długościami kroku, które nie występowały przy siatce jednorodnej.

Dla parametrów (??) najskuteczniejszym backendem był *sqp* który zbiegł do najlepszego rozwiązania zaczynając od zerowego sterowania, w przeciwieństwie do *lm\_feasable* który osiągnął trochę gorszy wynik 234437.6 (dla tego samego punktu startowego). Nie było tu znaczącej różnicy między sterowaniem kawałkami stałym a kawałkami liniowym. Używanie różnych punktów startowych nie przynosiło wyraźnych różnic, z wyjątkiem zaczynania ze sterowania stałe maksymalnego, dla którego optymalizacja kończyła się w jednej iteracji.

Dla parametrów (??) najtrudniej było dobrać odpowiedni punkt startowy i na znalezieniu odpowiedniego punktu startowego skupiła się większość eksperymentów. W tym przypadku backend *sqp* okazał się znacznie gorszy od *lm\_feasable*, za wyjątkiem punktu startowego będącego bardzo blisko najlepszego rozwiązania. Podobnie kawałkami liniowe osiągało gorsze wyniki od sterowania kawałkami stałego, co pewnie jest powodowane nieciągłością najlepszego rozwiązania. Znalezienie dobrego punktu początkowego wymagało postawienia hipotezy, że na początku wartość sterowania powinna być niska, a później wyższa. Zaczynając z takich pozycji optymalizacja zbiegała do punktu podobnego do najlepszego sterowania i wystarczyło już tylko wyznaczyć metodą prób i błędów najlepszy punkt nieciągłości, ponieważ optymalizator zaczynając z nieciągłego sterowania nie zmieniał punktu nieciągłości.

W kontekście tempa zbieżności i czasu działania różnych metod najważniejszym elementem jest samodzielne liczenie pochodnej, co zmniejszyło czas działania wielokrotnie oraz znacząco poprawiło tempo zbieżności i wyniki optymalizatora. W eksperymentach korzystających z domyślnego algorytmu

	backend	start	wynik	#iteracji	#wywołań $\tilde{J}$
Parametry (??)	<i>lm_feasible</i>	$g(t) = 0$	234437.6	6	15
	<i>lm_feasible</i>	$g(t) = g_{max}$	213575.7	1	2
	<i>sqp</i>	$g(t) = 0$	213575.7	8	9
Parametry (??)	<i>lm_feasible</i>	$g(t) = 0$	324132.9	7	17
	<i>lm_feasible</i>	$g(t) = 0.4 \cdot \mathbb{1}_{[42.5, 200]}$	283519.9	10	26
	<i>lm_feasible</i>	$g(t) = 0.55 \cdot \mathbb{1}_{[42.5, 200]}$	272466.2	19	42
	<i>sqp</i>	$g(t) = 0$	340595.9	3	57
	<i>sqp</i>	$g(t) = 0.55 \cdot \mathbb{1}_{[42.5, 200]}$	272502.7	7	110
Parametry (??), siatka niejednorodna	<i>lm_feasible</i>	$g(t) = 0$	295303.8	20	42
	<i>lm_feasible</i>	$g(t) = 0.4 \cdot \mathbb{1}_{[42.5, 200]}$	277775.5	14	35
	<i>lm_feasible</i>	$g(t) = 0.55 \cdot \mathbb{1}_{[42.5, 200]}$	272848.2	19	42
	<i>sqp</i>	$g(t) = 0$	348091.2	2	53

Tabela 3: Wyniki niektórych eksperymentów

liczenia pochodnej za pomocą różnic skończonych, trzeba było używać siatki posiadającej 4 razy mniej punktów, a i tak czas działania był wielokrotnie gorszy. Z tego powodu nie przeprowadzono wielu takich eksperymentów i nie będziemy dalej rozważać tego podejścia. Wypada też wspomnieć, że backend *sqp* wymagał zauważalnie więcej wywołań funkcji celu i czasu aby zbiec niż *lm\_feasible*. Dalszy wpływ punktu startowego i metody na wyniki i czas zbieżności zaprezentowano na tabeli 3.

## 4 Analiza i krytyka wyników

Jak widzimy wyniki dla różnych rodzajów parametrów znacznie się różnią. Dokładniejsze przyjrzenie się funkcjonalowi celu (5) pozwala wyjaśnić takie zjawisko. Jak widzimy parametr  $\omega$  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 zestawie parametrów (??) parametr  $\omega$  jest znacznie większy niż w (??), więc w tym przypadku druga całka ma większy wpływ na wynik funkcjonalu. Jak przyjrzymy się rozwiązaniu z rysunku 2b), 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 wartościach parametrów (??) bardziej opłaca się utrzymywać stan  $y_2(t) - y_1(t) < 0$ , natomiast przy wartościach (??) 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. Wynika to z zastosowanych technologii. Wyniki backendu *lm\_feasible* były mocno zależne od punktu startowego, natomiast backend *sqp*, który powinien szukać rozwiązania bardziej globalnie, miał nie najlepszą skuteczność. Być mo-

że zastosowanie innych backendów optymalizacji nieliniowej umożliwiłoby szukanie rozwiązania w sposób mniej zależący od znalezienia dobrego punktu startowego.

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 splajnów wyższego rzędu mogłoby poprawić tempo zbieżności. Być może zastosowanie tych metod oraz automatyzacja procesu znajdowania siatki dyskretyzacji np. metodą zastosowaną w [4] pozwoliłaby uzyskać lepsze wyniki.

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

## Literatura

- [1] P. Bajger, M. Bodzioch, and U. Foryś. Overcoming acquired chemotherapy resistance: insights from mathematical modelling. *niepublikowany manuskrypt*.
- [2] M. Diehl and S. Gros. Numerical optimal control. <https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf>, 2017. Accessed: 01-08-2020.
- [3] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59:849–904, 01 2017.
- [4] M. Patterson, W. Hager, and A. Rao. A ph mesh refinement method for optimal control. *Optimal Control Applications and Methods*, 36, 02 2014.
- [5] A. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135, 01 2010.