Optymalna strategia podawania leku

Tomasz Kanas

9 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 [tu wypada zacytować wyjściową pracę]. Model ten przedstawia rozwój nowotworu w czasie w zależności od dawkowania leku za pomocą równania różniczkowego:

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

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

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

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 $g : [0, T] \to [0, g_{\max}]$ oraz $V_1, V_2, K : [0, T] \to (0, \infty)$ spełniających (1), oraz minimalizujących 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 uprościmy notację.

Problem 1. *Znaleźć funkcję kawałkami ciągłą*

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

i funkcję

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

spełniające równanie różniczkowe:

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

oraz 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 są $t_1, \ldots, t_n$, wtedy (4) oznacza ciąg równań różniczkowych postaci

$$\begin{aligned} \dot{y}|_{(t_i, t_{i+1})}(t) &= f|_{(t_i, t_{i+1})}(t, y, g) \\ y(t_i) &= \lim_{t \to t_i^-} y(t) \end{aligned} \qquad \text{gdzie } i \in \{0, \ldots, n\},\ t_0 = 0,\ t_{n+1} = T \tag{7}$$

Zwróćmy jeszcze uwagę na fakt, że funkcja $F(x) = -\ln(x)$ posiada osobliwość w 0, więc problem nie jest dobrze zdefiniowany 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 $y_3(t) = 0$ zadanie także nie jest dobrze określone.

1.2 Problem przybliżony

Analityczne rozwiązywanie problemu optymalnego sterowania rzadko kiedy jest możliwe, a nawet gdy jest możliwe, jest trudne. 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 za pomocą splajnu ~~interpolującego~~ g w punktach t_i. Dla prostoty ograniczymy się do splajnów stopnia 0 i 1. Ostatecznie przybliżone sterowanie ma postać:

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

lub

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

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

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

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

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

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

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

W ten sposób wyraziliśmy przybliżoną funkcję celu jako funkcję $g_1, \ldots, g_n$:

$$\hat{J}(g_1, \ldots, g_n) = \hat{J}(\hat{y}) \tag{15}$$

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

Problem 2.

$$\min_{g_1,\ldots,g_n} \hat{J}(g_1,\ldots,g_n) \text{ z ograniczeniami określone w (15) (16)} \tag{16}$$

$$\forall_{i\in\{1,\ldots,n\}} 0 \le g_i \le g_{\max} \tag{17}$$

Jest to problem optymalizacji nieliniowej z ograniczeniami i istnieją implementacje metod pozwalających uzyskać przybliżone rozwiązanie tego problemu. To podejście do numerycznego problemu optymalnego występuje w [1] pod nazwą „sequential", a w [3] pod nazwą „direct multiple shooting".

1.3 Plan rozwiązania

Aby obliczyć wynik problemu przybliżonego skorzystamy ze środowiska MATLAB/Octave wraz z dostarczonym 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 $\hat{J}(g_1,\ldots,g_n)$ 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 $g_1,\ldots,g_n$.

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 należy zróżniczkować równanie (9) lub (10):

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

lub

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

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 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):

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

gdzie

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

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:

$$\hat{J}(\hat{y}) = h(\alpha_0, \ldots, \alpha_N) \cdot j(\hat{y}(0, h, 2h, \ldots, T))^T \tag{22}$$

W implementacji w Octave

Korzystamy tu ze zwektoryzowanej implementacji funkcji $\hat{y}$, ~~założenia naszej implementacji są spełnione ponieważ krok kwadratury to h~~. 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:

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

co to?

gdzie

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

3 Eksperymenty

↖ w punkcie (tak jak po lewej)

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 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:

$$\alpha_{12} = 0.1 \qquad \alpha_{21} = 0.15 \qquad \omega = 1000 \tag{25}$$

$$\alpha_{12} = 0.5 \qquad \alpha_{21} = 0.75 \qquad \omega = 2000 \tag{26}$$

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:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{27}$$

Eksperymenty proponuję zacząć od weryfikacji poprawności liczenia gradientu np. przez porównanie z aproksymacją różnicami dielonymi.

Do przybliżania funkcjonału celu (3) użyjemy kwadratury trapezów. ~~Jest to nieskomplikowana kwadratura, ale przy niskiej długości kroku daje dobre przybliżenie~~. Wartości parametrów α_i we wzorze (14) to:

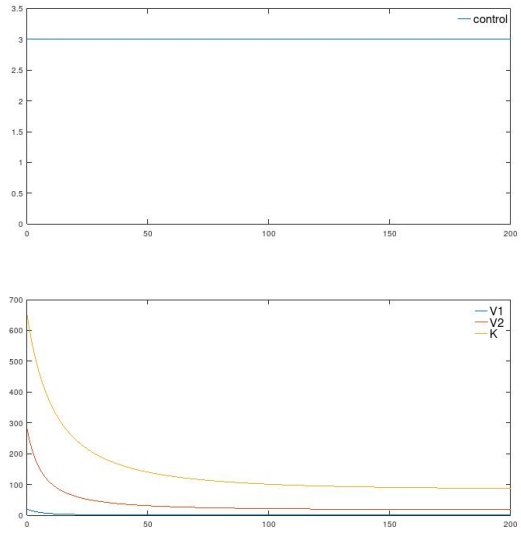
$$\alpha_i = \begin{cases} \frac{1}{2} \text{ gdy } i \in \{0, N\} \\ 1 \text{ w.p.p.} \end{cases} \tag{28}$$

Przeprowadzone eksperymenty obejmowały różne punkty startowe dla optymalizatora i siatki dyskretyzacji. Testowane było zarówno sterowanie kawałkami stałe (9), jak i kawałkami liniowe (10). Przykładowymi przetestowanymi punktami startowymi było sterowanie stale równe 0, 1 lub $g_{\max}$, sterowanie malejące liniowo od $g_{\max}$ do 0, czy sterowanie przyjmujące $g_{\max}$ na początku przedziału czasu, i 0 później. Przykładowymi siatkami dyskretyzacji są punkty rozłożone równo w odległości $\frac{1}{2}$, oraz siatka z odstępami $\frac{1}{2}$ w początkowej i końcowej $\frac{1}{4}$ długości przedziału i odstępami 1 na pozostałej części przedziału czasu.

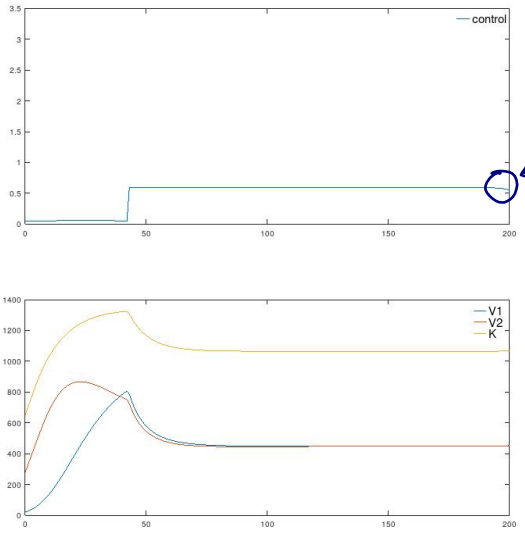
Do eksperymentów użyte zostało środowisko Octave. Optymalizator FMINICON na tym środowisku umożliwia użycie jednego z dwóch backendów do optymalizacji nieliniowej: *lm_feasable*, lub *sqp*. Jedną z różnic między tymi backendami jest to, że *lm_feasable* zachowuje ograniczenia w trakcie optymalizacji,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

3.1 Wyniki eksperymentów

(a) Rozwiązanie dla parametrów (25)



(b) Rozwiązanie dla parametrów (26)



Rysunek 1: Rozwiązania równania (1)

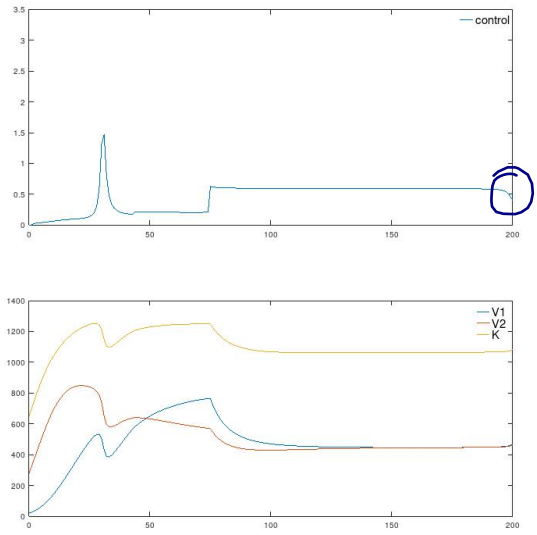
Dla parametrów (25) wyniki eksperymentów są zgodne: najlepsze rezultaty osiągamy dla sterowania stale równego $g_{\max}$. Optymalizator zaczynając w nim nie znajduje żadnego lepszego punktu,

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 backendów, 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 1a.

Dla parametrów (26) 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 2. Wartość funkcjonału celu dla tego rozwiązania to 295794.6, co jest o $8,6\%$ większe od rozwiązania optymalnego.

Rysunek 2: Nietypowe rozwiązanie dla parametrów (26)



3.2 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 (25) lub (26), 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

	backend	start	wynik	#iteracji	#wywołań
Parametry (25)	*lm_feasible*	$g(t) = 0$	234437.6	6	15
	lm_feasible	$g(t) = g_{max}$	213575.7	1	2
	sqp	$g(t) = 0$	213575.7	8	9
Parametry (26)	*lm_feasible*	$g(t) = 0$	324132.9	7	17
	lm_feasible	$g(t) = 0.4 \cdot \mathbb{1}_{[42.5,200]}$	283519.9	10	26
	lm_feasible	$g(t) = 0.55 \cdot \mathbb{1}_{[42.5,200]}$	272466.2	19	42
	sqp	$g(t) = 0$	340595.9	3	57
	sqp	$g(t) = 0.55 \cdot \mathbb{1}_{[42.5,200]}$	272502.7	7	110
Parametry (26), siatka niejednorodna	*lm_feasible*	$g(t) = 0$	295303.8	20	42
	lm_feasible	$g(t) = 0.4 \cdot \mathbb{1}_{[42.5,200]}$	277775.5	14	35
	lm_feasible	$g(t) = 0.55 \cdot \mathbb{1}_{[42.5,200]}$	272848.2	19	42
	sqp	$g(t) = 0$	348091.2	2	53

Tabela 1: Wyniki niektórych eksperymentów

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 różnice wartości na przedziałach z różnymi długościami kroku, które nie występowały przy siatce jednorodnej.

Dla parametrów (25) 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 stale maksymalnego, dla którego optymalizacja kończyła się w jednej iteracji.

Dla parametrów (26) 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 ~~tępa~~ zbieżności i czasu działania różnych metod najważniejszym elementem jest samodzielne liczenie pochodnej, co zmniejszyło czas działania wielokrotnie oraz znacząco poprawiło tępo zbieżności i wyniki optymalizatora. W eksperymentach korzystających z domyślnego algorytmu 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_feasable*.

Dalszy wpływ punktu startowego i metody na wyniki i czas zbieżności zaprezentowano na tabeli 1.

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 zestawie parametrów (26) parametr ω jest znacznie większy niż w (25), więc w tym przypadku druga całka ma większy wpływ na wynik funkcjonału. Jak przyjrzymy się rozwiązaniu z rysunku 1b), 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 (26) bardziej opłaca się utrzymywać stan $y_2(t) - y_1(t) < 0$, natomiast przy wartościach (25) 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

najmniej?

Wszystkie wyniki które udało się osiągnąć są co ~~najwyżej~~ minimami lokalnymi. Wynika to głównie 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 może 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 tępa zbieżności, ale nie udało się jej tu z sukcesem zastosować. Podobnie przybliżanie sterowania za pomocą splajnów wyższego rzędu mogłoby poprawić tępo zbieżności. Być może zastosowanie tych metod oraz automatyzacja procesu znajdowania siatki dyskretyzacji np. metodą zastosowaną w [2] 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 1a to 14.9, więc i ta nie jest zbyt bliska 0. Okazuje się jednak, że przy parametrach (26), 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.

W rozdziale 3 warto byłoby zbadać zależność wyników od "h" (liczba dyskretyzacji) i od wyboru dyskretyzacji (P_0 vs P_1 — to już częściowo jest).

Verte!

Literatura

[1] M. Diehl and S. Gros. *Numerical Optimal Control*. 2017.

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

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

Eksperymenty należy przeprowadzić w bardziej usystematyzowany sposób. Z grubsza widzę to tak:

- wybierać "szerokie" (ale sprecyzowany) zestaw testów, uwzględniający
  - wiele solverów, backendów, tolerancje
  - wiele parametrów rodzina
  - itd.

  } można im nadać jakieś oznaczenia

  (Ten zbiór np. wylosuje Pan do policzenia, ie dla (25) zawsze wychodzi to samo)

- w kolejnych eksperymentach definiować "węższe" podzbiory w/w zbioru na których będzie Pan badał specjalne cechy np. rozbieżności lub solvera.

Oczywiście w/w to zarys ogólnej strategii, którą może Pan dopasować do swoich upodobań.