Optymalna strategia podawania leku

Tomasz Kanas

3 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

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

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.

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:

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

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 $f(t, y, g) : \mathbb{R}^5 \to \mathbb{R}$ 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 [7].

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 [7].

1.4 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 $\hat{J}$, 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 $\hat{g}$ 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 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:

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

Oznaczane jako „Parametry” w tabelach z wynikami.

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

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

Oznaczane „aproks." tabelach z wynikami.

3.1.4 Siatki dyskretyzacji sterowania

- 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 $t_n = \begin{cases} 0.5 \cdot n \text{ if } n \leq 100 \\ 50 + (n - 50) \text{ if } 100 < n \leq 200 \\ 150 + 0.5 \cdot (n - 200) \text{ if } n > 200 \end{cases}$

Oznaczane „siatka" w tabelach z wynikami.

3.1.5 Kroki dyskretyzacji h metody R-K rozwiązania (4)

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

3.1.6 Sterowania startowe

- $g_0 \equiv 0$
- $g_3 \equiv g_{\max} = 3$
- $g_{\alpha,\beta} = \alpha \cdot \mathbb{1}_{[\beta,T]}$. Ograniczymy się do: $g_{0.4,42.5}$, $g_{0.55,42.5}$.
- $\mathfrak{g} = 3 \cdot \mathbb{1}_{[0,10]} + 0.5 \cdot \mathbb{1}_{[50,T]}$

Oznaczane „start" w tabelach z wynikami.

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 trzech wartości: wartości przybliżonego funkcjonału celu $\hat{J}$, liczby iteracji optymalizatora (oznaczaną przez „iter"), oraz liczbę wywołań funkcjonału celu w trakcie optymalizacji (oznaczaną przez $\#\hat{J}$).

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

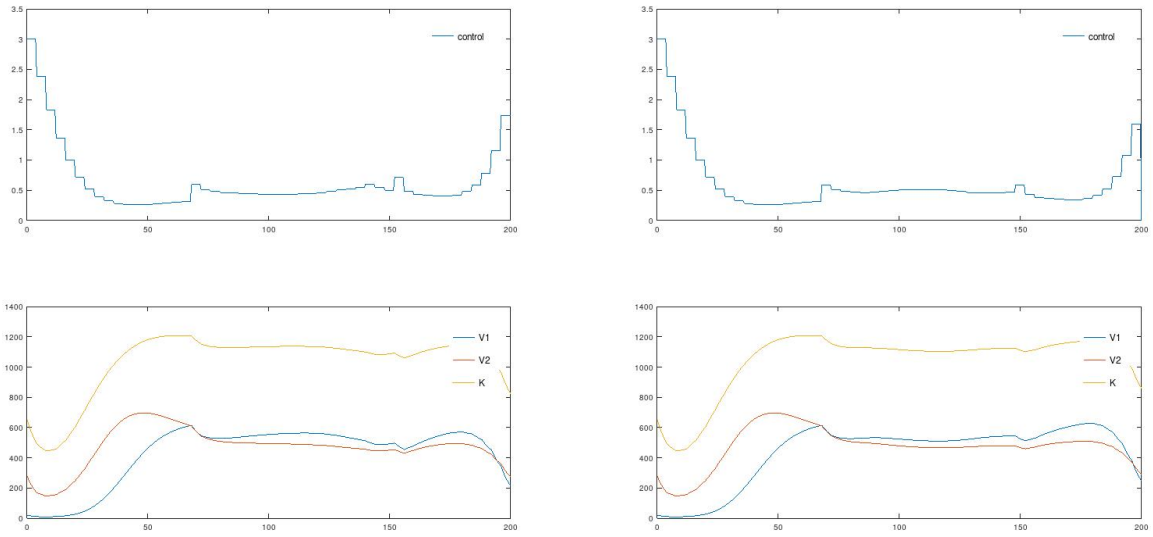
Pierwszy eksperyment ma na celu zaprezentowanie poprawności implementacji obliczania gradientu funkcji celu. Aby to sprawdzić porównamy naszą implementację, z domyślną implementacją przybliżającą gradient za pomocą różnic skończonych dostępną w FMINICON. Obliczanie wyników metodą domyślną wymaga wiele 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.

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

(a) Rozwiązanie przy użyciu różnic skończonych

(b) Rozwiązanie z liczeniem gradientu



Rysunek 1: Rozwiązania dla przypadku testowego

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.

Aby zyskać jeszcze większą pewność o poprawności naszej implementacji policzymy gradient $\hat{J}$ względem parametrów g_i, w punkcie będącym sterowaniem zaprezentowanym na wykresie (1b), Porównamy gradient G obliczony przy użyciu naszej implementacji, oraz gradient G_τ uzyskany za

pomocą metody różnic skończonych dla różnych wartości kroku różnic skończonych (FD)

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

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, ~~co też jest zachowaniem oczekiwanym~~. gdyż ....

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.

[Warto się zwrócić uwagę na to, że przy tym $\tau = 10^{-3}$ jesteśmy raczej dalelo od optimum? Czy dla $\tau = 10^{-7}$ wyniki będą bardziej zgodne?] ↳ Tabeli 1.

3.2.2 Parametry (CC)

Wyniki dla parametrów (CC) są dość jednoznaczne. Najlepszym znalezionym rozwiązaniem ~~jest~~ wydaje się $g \equiv g_{\max} = 3$, 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, algorytm *sqp* jest jedynym który zbiegł z zerowego sterowania do sterowania optymalnego. Różnice między pozostałymi metodami są nieznaczne.

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

3.2.3 Metoda dyskretyzacji

Porównując odpowiednie wartości Tabeli 4, możemy zaobserwować, że dla sterowania startowego g_0, dyskretyzacja P_1 osiąga zauważalnie lepsze rezultaty niż dyskretyzacja P_0. Dla algorytmu *lm* zastosowanie dyskretyzacji liniowej poprawiło wyniki, natomiast dla algorytmu *sqp* zmniejszyło zarówno liczbę iteracji jak i liczbę wywołań prawie dwukrotnie. Gdy punkt startowy to $g_{0.4,42.5}$, to obie dyskretyzacje osiągają bardzo podobne rezultaty.

Ale czy z "g_0" zbiegamy do lokalnego minimum? Czy jest ono globalne?
A dla "$g_{0.4,\ldots}$" – jak wtedy jest z ciągłością?

Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
(CC)	*lm*	P_0	S_1	0.1	g_0	2.2	3	8
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	2.19	3	8
(CC)	*lm*	P_0	N_{kon}	0.1	g_0	2.64	9	14
(CC)	*lm*	P_1	S_1	0.1	g_0	2.21	2	7
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_0	2.21	2	7
(CC)	*lm*	P_1	N_{kon}	0.1	g_0	2.66	2	7
(CC)	*sqp*	P_1	$S_{0.5}$	0.1	g_0	2.14	9	10
(CC)	*lm*	P_0	S_1	0.1	g_3	2.14	1	2
(CC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	2.14	1	2
(CC)	*lm*	P_0	N_{kon}	0.1	g_3	2.14	1	2
(CC)	*lm*	P_1	S_1	0.1	g_3	2.14	1	2
(CC)	*lm*	P_1	$S_{0.5}$	0.1	g_3	2.14	1	2
(CC)	*lm*	P_1	N_{kon}	0.1	g_3	2.14	1	2

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

Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	3.04	81	158
(DC)	*lm*	P_1	$S_{0.5}$	0.1	g_0	2.94	95	185
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	3.3	18	205
(DC)	*sqp*	P_1	$S_{0.5}$	0.1	g_0	3.28	9	105
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	120	236
(DC)	*lm*	P_1	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.74	103	202
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	10	123
(DC)	*sqp*	P_1	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	10	136

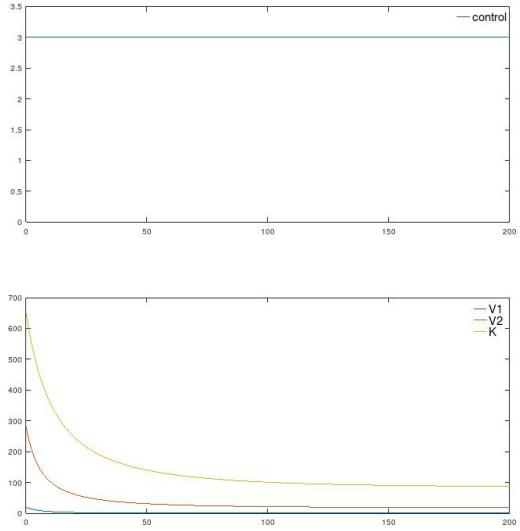
Tabela 4: ~~Eksperymenty badające~~ metody dyskretyzacji

3.2.4 Siatka dyskretyzacji

Pierwszym wnioskiem na temat siatki dyskretyzacji który możemy wyciągnąć z Tabeli 5 jest to, że dla algorytmu *lm* siatka dyskretyzacji nie ma zbyt dużego wpływu na końcowy wynik. Dla testów różniących się tylko siatką dyskretyzacji, różnica w wynikach dla algorytmu *lm* wynosi co najwyżej 0.09. Możemy zauważyć, że zwykle wynik poprawia się wraz z zagęszczaniem siatki, ale jak widzimy dla startu $g_{0.4,42.5}$ i siatki $S_{0.1}$, zagęszczenie satki może pogorszyć wynik. Siatka nieregularna dla tego algorytmu nie osiągała najlepszych wyników, ale zbiegała znacznie szybciej niż siatki regularne o podobnych wynikach.

Dla algorytmu *sqp* i startu g_0 siatka dyskretyzacji ma już większe znaczenie, różnice wynoszą nawet 0.23. Najlepszy wynik w tym przypadku osiąga siatka S_1, a siatka niejednorodna — najgorszy.

Wyniki dla algorytmu *sqp* siatki S_1 i startu $g_{0.4,42.5}$ są podejrzane. Wartość $\hat{J}$ dla sterowania startowego $g_{0.4,42.5}$ wynosi 30.1, więc jest lepsza niż wynik. Sprawdzenie wyników dla wywołań pośrednich ujawniło, że algorytm w trakcie optymalizacji osiągał wyniki podobne jak pozostałe testy



Rysunek 2: Rozwiązanie dla parametrów (CC)

Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
(DC)	lm	P_0	S_1	0.1	g_0	3.08	63	121
(DC)	lm	P_0	$S_{0.5}$	0.1	g_0	3.04	81	158
(DC)	lm	P_0	$S_{0.1}$	0.1	g_0	3.0	199	396
(DC)	lm	P_0	N_{sr}	0.1	g_0	3.02	25	51
(DC)	sqp	P_0	S_1	0.1	g_0	3.16	14	164
(DC)	sqp	P_0	$S_{0.5}$	0.1	g_0	3.3	18	205
(DC)	sqp	P_0	N_{sr}	0.1	g_0	3.39	15	172
(DC)	lm	P_0	S_1	0.1	$g_{0.4,42.5}$	2.78	118	231
(DC)	lm	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	120	236
(DC)	lm	P_0	$S_{0.1}$	0.1	$g_{0.4,42.5}$	2.81	38	81
(DC)	lm	P_0	N_{sr}	0.1	$g_{0.4,42.5}$	2.76	48	96
(DC)	sqp	P_0	S_1	0.1	$g_{0.4,42.5}$	4.07	24	85
(DC)	sqp	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	10	123
(DC)	sqp	P_0	N_{sr}	0.1	$g_{0.4,42.5}$	2.75	8	104

Tabela 5: Eksperymenty badające siatkę dyskretyzacji

dla tego sterowania startowego. Takie zachowanie byłoby teoretycznie możliwe dla tego algorytmu, ponieważ nie wymaga on spełnienia ograniczeń w trakcie optymalizacji, więc mógłby nie znaleźć lepszego punktu spełniającego ograniczenia. Jednakże punkt startowy spełnia ograniczenia, więc zachowanie takie wydaje się błędem optymalizatora.

3.2.5 Krok dyskretyzacji h

Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
(DC)	*lm*	P_0	$S_{0.5}$	0.5	g_0	3.12	79	156
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	3.04	81	158
(DC)	*lm*	P_0	$S_{0.5}$	0.02	g_0	2.94	129	256
(DC)	*sqp*	P_0	$S_{0.5}$	0.5	g_0	3.28	8	92
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	3.3	18	205
(DC)	*sqp*	P_0	$S_{0.5}$	0.02	g_0	3.31	9	106
(DC)	*lm*	P_0	$S_{0.5}$	0.5	$g_{0.4,42.5}$	2.75	83	159
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	120	236
(DC)	*lm*	P_0	$S_{0.5}$	0.02	$g_{0.4,42.5}$	2.75	91	179
(DC)	*sqp*	P_0	$S_{0.5}$	0.5	$g_{0.4,42.5}$	2.75	10	127
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	10	123
(DC)	*sqp*	P_0	$S_{0.5}$	0.02	$g_{0.4,42.5}$	2.75	13	166

Tabela 6: Eksperymenty badające krok dyskretyzacji h

Krok dyskretyzacji, jak wnioskujemy z Tabeli 6, dla algorytmu *lm* zachowuje się przewidywalnie, to znaczy im niższy, tym lepsze wyniki. Co ciekawe, dla algorytmu *sqp* zależność jest odwrotna, choć różnice w wynikach są dość nieznaczne (rzędu 0.03). Dla sterowania startowego $g_{0.4,42.5}$ krok dyskretyzacji wydaje się nie mieć prawie żadnego wpływu na wynik.

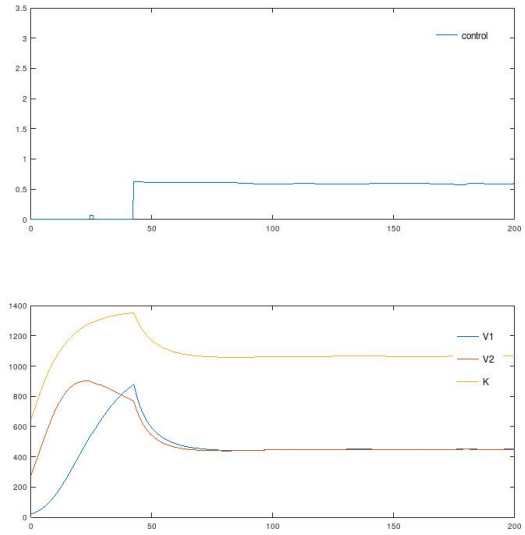
3.2.6 Sterowanie startowe

Parametry	algorytm	aproks.	siatka	h	start	$\hat{J}$	iter	$\#\hat{J}$
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_0	3.04	81	158
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_0	3.3	18	205
(DC)	*lm*	P_0	$S_{0.5}$	0.1	g_3	4.07	1	2
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	g_3	4.07	1	2
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	120	236
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.4,42.5}$	2.75	10	123
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$g_{0.55,42.5}$	2.72	31	64
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$g_{0.55,42.5}$	2.71	10	130
(DC)	*lm*	P_0	$S_{0.5}$	0.1	$\mathfrak{g}$	2.82	87	169
(DC)	*sqp*	P_0	$S_{0.5}$	0.1	$\mathfrak{g}$	2.81	100	1065

Tabela 7: Eksperymenty badające sterowanie startowe

Jak mogliśmy się już przekonać na poprzednich eksperymentach i co potwierdzają wyniki z Tabeli 7, dobór odpowiedniego sterowania startowego jest kluczowy.

Widzimy, 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



Rysunek 3: Rozwiązanie dla parametrów (DC)

pozostałych rezultatów. Najlepsze wyniki osiągane są dla sterowania startowego $g_{0.55,42.5}$. Ciekawe wyniki wychodzą sterowania startowego $\mathfrak{g}$. Jest to bardziej skomplikowane sterowanie startowe niż pozostałe i nie osiąga zby dobrych wyników. Ponadto wymaga stosunkowo dużo iteracji i wywołań funkcji $\hat{J}$, szczególne dla algorytmu *sqp* który potrzebował aż 1065 wywołań w 100 iteracjach, co było ustawione jako maksymalna liczba iteracji dla tego algorytmu.

Najlepszy wynik to 2.71 i osiągneliśmy go dla sterowania startowego $g_{0.55,42.5}$ i algorytmu *sqp*. Zaprezentowany jest on na Rysunku 3.

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 3, 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 tępo zbieżności. W literaturze rozważa też metody automatyzacji znajdowania siatki dyskretyzacji i odpowiedniego rzędu dyskretyzacji. Przykładem jest tu praca [6]. 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 2, 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] M. Patterson, W. Hager, and A. Rao. A ph mesh refinement method for optimal control. *Optimal Control Applications and Methods*, 36, 02 2014.

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