Metody obliczeniowe poszukiwania minimum

Przypomnienie – poszukiwanie minimum funkcji wielu zmiennych

$2)\ J : R^n \rightarrow R$

$J(x) = J(x_1, x_2, \ldots, x_n)$

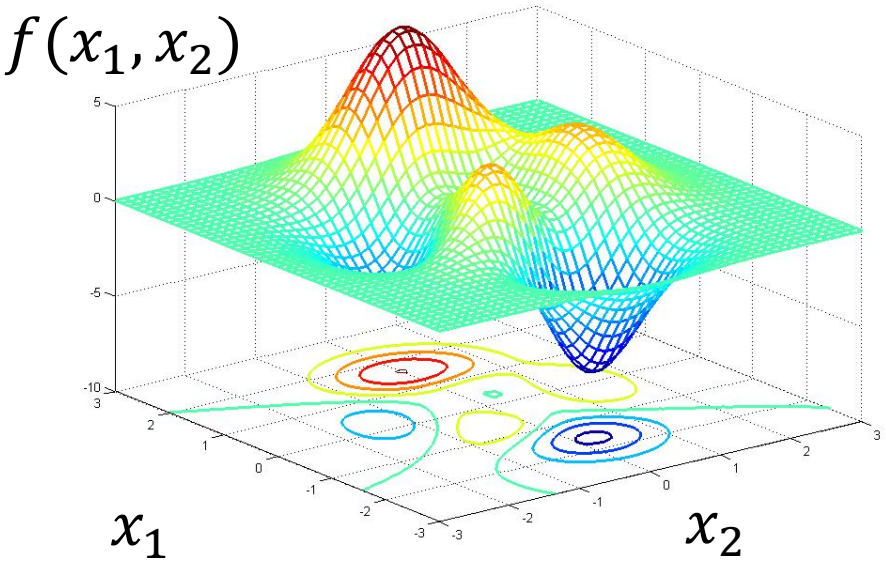
Warunek konieczny

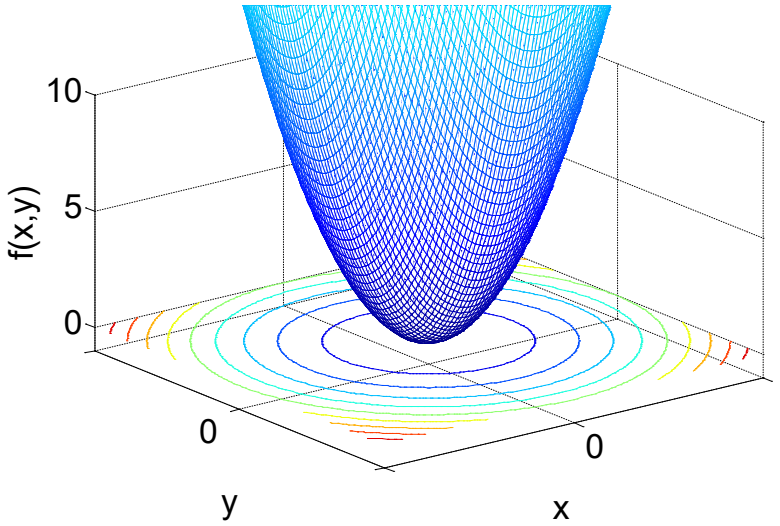
$$\nabla_x J = \frac{\partial J}{\partial x} = 0$$

Jeśli rozwiążemy rozwiązanie (dla funkcji wypukłej), to mamy rozwiązanie.

?

Interpretacja graficzna





Metody obliczeniowe (niektóre)

- Bezgradientowe
  - Algorytm Gaussa-Seidela
  - Algorytm Powell;
- Gradientowe:
  - Algorytm najszybszego spadku
  - Algorytm gradientów sprzężonych;
- Inne…

Algorytm Gaussa- Seidela

$$\min_x f(x) \qquad x = [x_1, \ldots, x_n]^T$$

Założenie:

- Funkcja celu może być przybliżona w pobliżu minimum ściśle wypukłą funkcją kwadratową.

Kierunki przeszukiwać (stałe, ortogonalne):

$$e_j = [\underbrace{0 \ldots 0}_{j-1 \text{ elementów}} 1 \, 0 \ldots 0]^T$$

W kroku j:

$$\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j-1)} + \hat{\alpha}^{(j)} e_j \qquad \hat{\alpha}^{(j)} = \arg\min_{\alpha^{(j)}} f\left(\boldsymbol{x}^{(j)} + \alpha^{(j)} \boldsymbol{e}_j\right)$$

Przykład

$$e_j = [\underbrace{0 \ldots 0}\, 1\, 0 \ldots 0]^T$$

$$\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j-1)} + \hat{\alpha}^{(j)} e_j \qquad \hat{\alpha}^{(j)} = \arg\min_{\alpha^{(j)}} f\left(\boldsymbol{x}^{(j)} + \alpha^{(j)} \boldsymbol{e}_j\right)$$

$$f(\boldsymbol{x}) = x_1^2 + x_1^2 \qquad x^{(0)} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

I iteracja, krok 1: $e_1 = [1 \quad 0]^T$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \hat{\alpha}^{(0)} e_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix} + \hat{\alpha}^{(0)} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 + \hat{\alpha}^{(0)} \\ 2 \end{bmatrix}$$

$$f\left(\begin{bmatrix} 3 + \hat{\alpha}^{(0)} \\ 2 \end{bmatrix}\right) = \left(3 + \hat{\alpha}^{(0)}\right)^2 + 2^2 \qquad \rightarrow \hat{\alpha}^{(0)} = -3$$

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

Przykład – c.d.

I iteracja, krok 2: $e_2 = [0 \quad 1]^T$

$$\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} + \hat{\alpha}^{(1)} e_2 \quad = \begin{bmatrix} 0 \\ 2 \end{bmatrix} + \hat{\alpha}^{(1)} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad = \begin{bmatrix} 0 \\ 2 + \hat{\alpha}^{(1)} \end{bmatrix}$$

$$f\left(\begin{bmatrix} 0 \\ 2 + \hat{\alpha}^{(1)} \end{bmatrix}\right) = 0 + \left(2 + \hat{\alpha}^{(1)}\right)^2 \qquad \rightarrow \hat{\alpha}^{(1)} = -2$$

II iteracja, krok 1: $e_1 = [1 \quad 0]^T$ $\qquad \boldsymbol{x}^{(2)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$\boldsymbol{x}^{(3)} = \boldsymbol{x}^{(2)} + \hat{\alpha}^{(2)} e_1 \quad = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \hat{\alpha}^{(2)} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad = \begin{bmatrix} \hat{\alpha}^{(2)} \\ 0 \end{bmatrix}$$

$$f\left(\begin{bmatrix} \hat{\alpha}^{(2)} \\ 0 \end{bmatrix}\right) = \hat{\alpha}^{(2)^2} \qquad \rightarrow \hat{\alpha}^{(0)} = 0 \qquad \boldsymbol{x}^{(3)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

II iteracja, krok 2: $e_2 = [0 \quad 1]^T$ - to samo i koniec obliczeń

Kryteria stopu

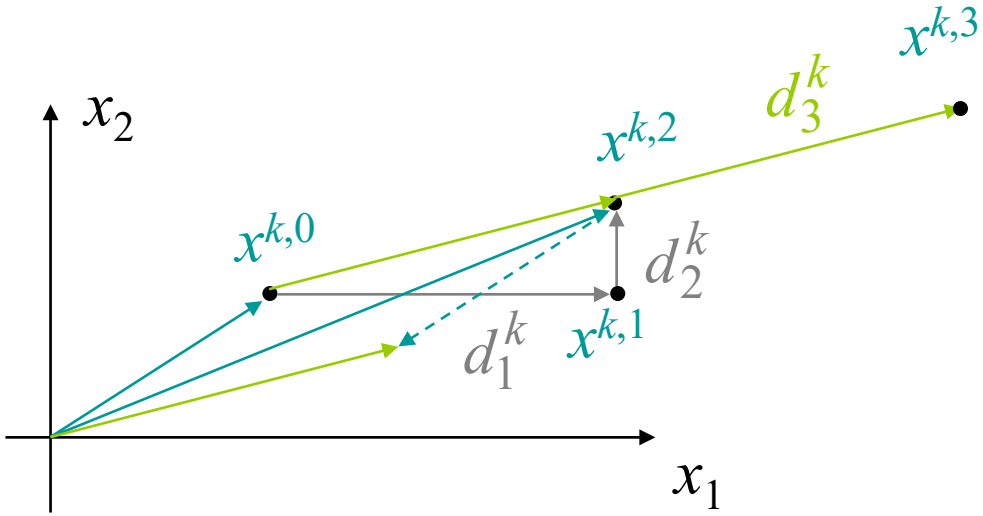
$$\left|f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right)\right| \leq \varepsilon$$

$$\left\|x^{(k+1)} - x^{(k)}\right\| \leq \varepsilon$$

Uwaga: (k) – numer pełnej interacji (nie kroku w iteracji)

Metoda Powella

$$d_{n+1}^k = x^{k,n} - x^{k,0}$$



$$x^{k,n+1} = x^{k,n} + \hat{\alpha}_j \; d_{n+1}^k$$

$$\hat{\alpha}_j = \arg\min_{\alpha_j} f\left(x^{k,n} + \alpha_j d_{n+1}^k\right)$$

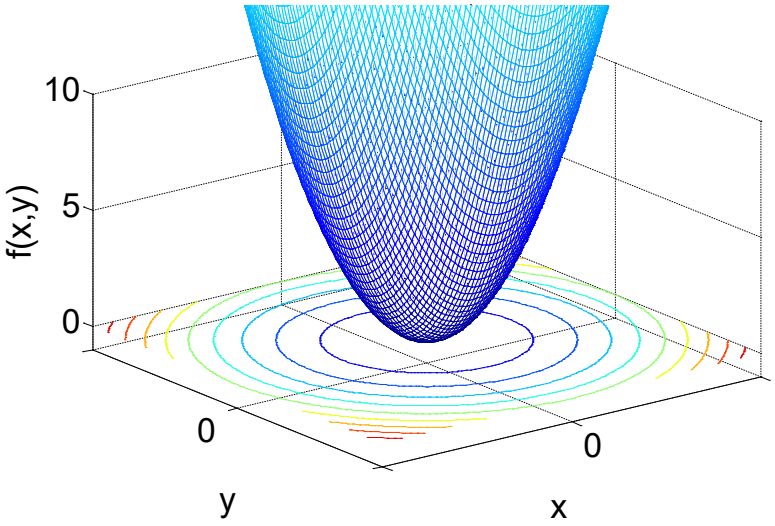
Kryterium stopu:

$$\left\| x^{k,n+1} - x^{k,0} \right\| \overset{?}{\leq} \varepsilon$$

Jeśli nie $\longrightarrow$

$$d_1^{k+1} \leftarrow d_2^k$$
$$d_2^{k+1} \leftarrow d_3^k$$
$$\vdots$$
$$d_n^{k+1} \leftarrow d_{n+1}^k$$

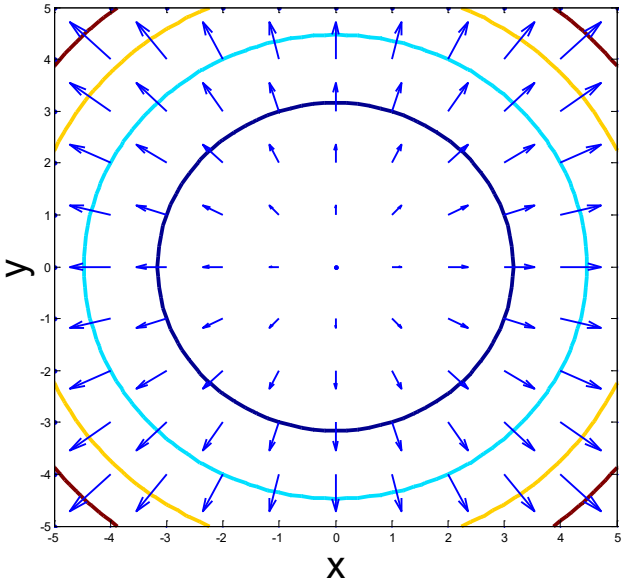
Interpretacja graficzna gradientu



$$f(\boldsymbol{x}) = x_1^2 + x_1^2$$

$$\nabla f = [2x_1 \quad 2x_2]$$

Kierunek najszybszego wzrostu funkcji



$$\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j-1)} - \hat{\alpha}^{(j)} \nabla f\Big|_{\boldsymbol{x}^{(j-1)}}$$

Metoda najszybszego spadku - przykład

$$\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j-1)} - \hat{\alpha}^{(j)} \nabla f\Big|_{\boldsymbol{x}^{(j-1)}}$$

$$\hat{\alpha}^{(j)} = \arg\min_{\alpha^{(j)}} f\left(\boldsymbol{x}^{(j)} - \alpha^{(j)} \nabla f\Big|_{\boldsymbol{x}^{(j-1)}}\right)$$

$$f(\boldsymbol{x}) = x_1^2 + x_1^2 \qquad x^{(0)} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\nabla f = [2x_1 \quad 2x_2]$$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \hat{\alpha}^{(0)} \nabla f\Big|_{\boldsymbol{x}^{(j-1)}} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} + \hat{\alpha}^{(0)} \begin{bmatrix} 2x_1^{(0)} \\ 2x_2^{(0)} \end{bmatrix} = \begin{bmatrix} 3 + 6\hat{\alpha}^{(0)} \\ 2 + 4\hat{\alpha}^{(0)} \end{bmatrix}$$

$$f\left(\begin{bmatrix} 3 + 6\hat{\alpha}^{(0)} \\ 2 + 4\hat{\alpha}^{(0)} \end{bmatrix}\right) = \left(3 + 6\hat{\alpha}^{(0)}\right)^2 + \left(2 + 4\hat{\alpha}^{(0)}\right)^2 \rightarrow \hat{\alpha}^{(0)} = -0.5$$

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Optymalizacja dynamiczna bez ograniczeń

Z poprzedniego wykładu

$x_{i+1} = f_i(x_i, u_i), \;\; i = 0,1, \ldots, N-1$ $\qquad$ x_0 – zadane

$$Min \leftarrow J = \sum_{i=0}^{N-1} L_i(x_i, u_i) + \Phi(x_N)$$

$$H_i(x_i, u_i, p_{i+1}) = L_i(x_i, u_i) + p_{i+1}^T g_i(x_i, u_i)$$

$$H_i(x_i, u_i, p_{i+1}) = L_i(x_i, u_i) + p_{i+1}^T g_i(x_i, u_i)$$

Warunki konieczne:

(1) $$\frac{\partial H_i(x_i, u_i, p_{i+1})}{\partial u_i} = 0, \qquad i = 0,1, \dots, N-1$$

(Warunek na minimum hamiltonianu)

(2) $$p_i = \left(\frac{\partial H_i(x_i, u_i, p_{i+1})}{\partial x_i}\right)^T, i = 1,2, \dots, N-1$$

(2a) $$p_N = \left(\frac{\partial \Phi\,(x_N)}{\partial x_N}\right)^T$$

(równanie sprzężone)

Oczywiście, musi być spełnione równanie stanu:

(3) $x_{i+1} = g_i(u_i, x_i),\ i = 0,1, \dots, N-1$ (3a) $x_0 = x(0)$

Problem obliczeniowy - TBVP

Algorytmy dla zadań bez ograniczeń

$x_{i+1} = f_i(x_i, u_i),\ \ i = 0,1, \dots, N-1$ $\quad$ x_0– zadane

$$Min \leftarrow J = \sum_{i=0}^{N-1} L_i(x_i, u_i) + \Phi(x_N)$$

- Inicjalizacja: przyjmij sekwencję sterowań $u^{(0)} = [u_0^{(0)}, u_1^{(0)}, \dots, u_{N-1}^{(0)}]$
- Oblicz wynikającą z przyjętej sekwencji sterowań trajektorię w przestrzeni stanu (i zmiennych sprzężonych)
- Dopóki nie jest spełnione kryterium stopu poprawiaj sekwencję sterowań $u^{(k+1)} = u^{(k)} + \Delta u^{(k)}$

Metoda gradientu prostego

- Inicjalizacja: przyjmij sekwencję sterowań $u^{(0)} = [u_0^{(0)}, u_1^{(0)}, \dots, u_{N-1}^{(0)}]$
- Oblicz wynikającą z przyjętej sekwencji sterowań trajektorię w przestrzeni stanu $x^{(k)} = [x_0^{(k)}, x_1^{(k)}, \dots, x_N^{(k)}]$
- Wyznacz wartości zmiennych sprzężonych $p^{(k)} = [p_0^{(k)}, p_1^{(k)}, \dots, p_N^{(k)}]$
- Wyznacz normę gradientu $\left\| \frac{\partial H_i^{(k)}(x_i, u_i, p_{i+1})}{\partial u_i} \right\|$
- Jeśli nie jest spełnione kryterium stopu, wyznacz sterowanie w następnej iteracji

$$u_i^{(k+1)} = u_i^{(k)} - t \left. \frac{\partial H_i^{(k)}(x_i, u_i, p_{i+1})}{\partial u_i} \right|_{(x_i, u_i, p_{i+1})^{(k)}} = 0,$$

$$i = 0,1, \dots, N-1$$

Metoda gradientu sprzężonego w przestrzeni sterowań

W metodzie gradientu prostego kierunek poprawy sterowania w iteracji k dany był przez gradient Hamiltonianu:

$$d_i^{(k)} = -b_i^{(k)} = -\left.\frac{\partial H_i^{(k)}(x_i,u_i,p_{i+1})}{\partial u_i}\right|_{(x_i,u_i,p_{i+1})^{(k)}}$$

Dla uzyskania szybszej zbieżności można wykorzystać kierunki sprzężone, tzn. podobnie, jak wcześniej

$$u_i^{(k+1)} = u_i^{(k)} + t d_i^{(k)} = 0, \qquad i = 0,1,\dots,N-1$$

$$d_i^{(k)} = -b_i^{(k)} + c^{(k)} d_i^{(k-1)},\ c^{(0)} = 0$$

$$c^{(k)} = \frac{\left\|b^{(i)}\right\|^2}{\left\|b^{(i-1)}\right\|^2}$$

Optymalizacja dynamiczna z ograniczeniami – metoda funkcji kary

Optymalizacja statyczna (przypomnienie)
Algorytmy numeryczne

Metody kierunków dopuszczalnych

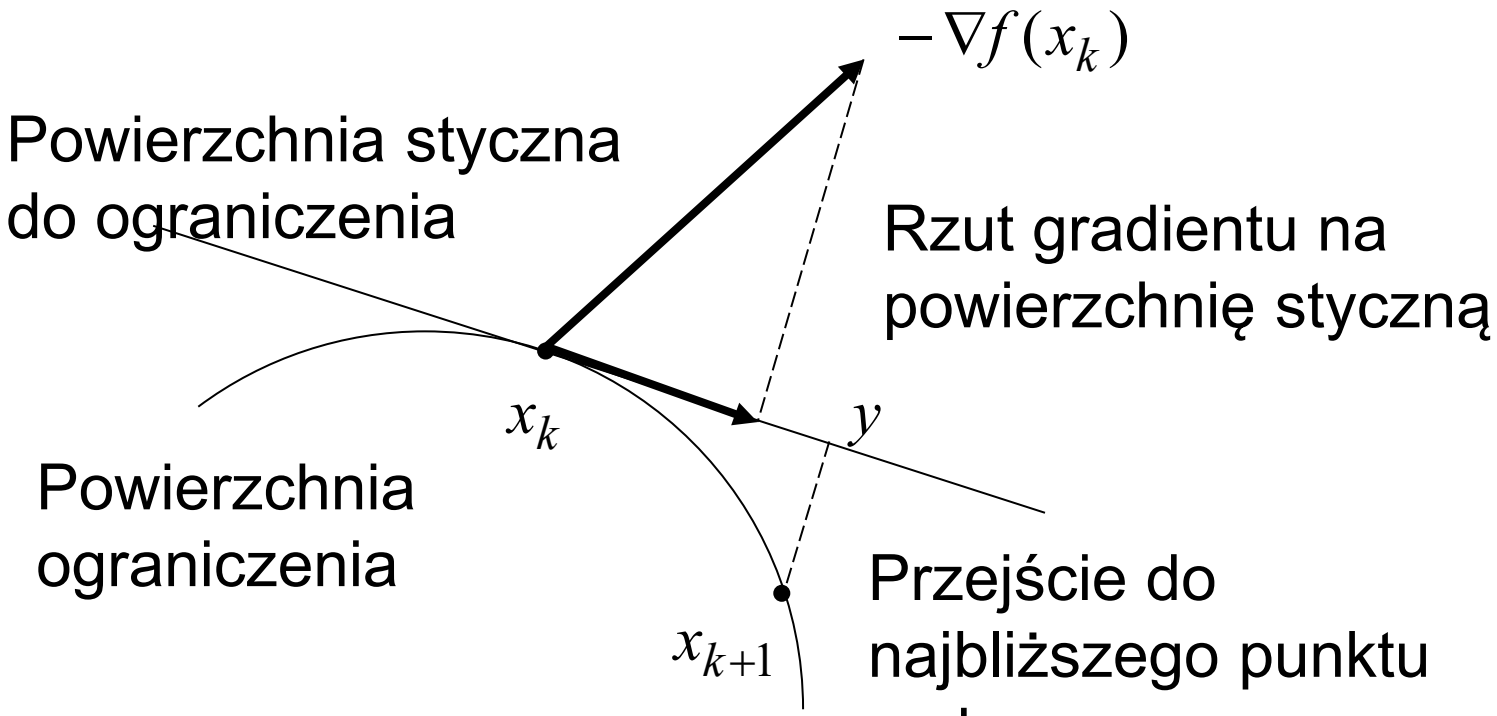
Metoda rzutowania gradientu

Metody funkcji kary

Połączenie metod funkcji kary z metodami gradientowymi

Metoda rzutowania gradientu

Po znalezieniu się na granicy ograniczeń:



Sformułowanie problemu

$$x_{i+1} = f_i(x_i, u_i), \;\; i = 0,1, \dots, N-1 \qquad x_0 - \text{zadane}$$

$$Min \leftarrow J = \sum_{i=0}^{N-1} L_i(x_i, u_i) + \Phi(x_N)$$

Zazwyczaj w praktycznych zadaniach występują ograniczenia (patrz tablica).

Ogólna postać ograniczeń:

$$h_i(x_i, u_i) \leq a_i$$

$$g_i(x_i, u_i) = b_i$$

(indeks i odnosi się do konkretnego etapu na horyzoncie $[0, N]$;
a_i, b_i są w ogólnym przypadku wektorami)

Funkcja kary

$$x = [x_0^T, x_1^T, \dots, x_N^T]^T \qquad x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{in} \end{bmatrix}$$

$$u = [u_0^T, u_1^T, \dots, u_{N-1}^T]^T \qquad u_i = \begin{bmatrix} u_{i1} \\ \vdots \\ u_{im} \end{bmatrix}$$

$$min \leftarrow \overline{J(x,u,t)} = J(x,u) + t \cdot R(x,u)$$

współczynnik kary

funkcja kary

$R(x,u) = 0$, jeśli ograniczenie jest spełnione

$R(x,u) > 0$, jeśli ograniczenie nie jest spełnione

Dobór funkcji kary – główna idea

(Uzasadnienie i przykłady – patrz tablica)

(etap decyzyjny, zdyskretyzowany czas – nie krok iteracji)

Dla ograniczenia równościowego w dyskretnej chwili i:

$$g_i(x_i, u_i) = a_i$$

$$g_i = \begin{bmatrix} g_{i1} \\ \vdots \\ g_{iq} \end{bmatrix}$$

$$R_i^g(x_i, u_i) = \sum_{j=1}^{q} \|g_i(x_i, u_i) - ai\|^2$$

Dla ograniczenia nierównościowego w dyskretnej chwili i:

$$h_i = \begin{bmatrix} h_{i1} \\ \vdots \\ h_{ik} \end{bmatrix}$$

$$h_i(x_i, u_i) \leq a_i$$

$$R_i^h(x_i, u_i) = \sum_{j=1}^{k} \big(h_{ij}(x_i, u_i) - a_i\big) \max\big(0, h_{ij}(x_i, u_i) - a_i\big)$$

Ostatecznie:

$$R(x, u) = \sum_{i=0}^{N} \left(R_i^g(x_i, u_i) + R_i^h(x_i, u_i)\right)$$

Dobór funkcji kary – główna idea

Podsumowując – modyfikujemy zadanie optymalizacji, w dość intuicyjny sposób – dlaczego to nie wystarczy?

Potrzebna modyfikacja:

- Rozwiązanie powyższego zadania stanowi tylko pierwszą iterację; kolejne będą podobne, z tym, że
  - Należy (?) modyfikować współczynnik kary t tak, by przyspieszać znajdowanie rozwiązań dopuszczalnych
  - Dla dużych wartości współczynnika kary t zadanie optymalizacyjne jest źle uwarunkowane numerycznie
  - Należy uwzględniać, jak daleko jesteśmy od spełnienia ograniczeń – stąd wprowadzenie miary przekroczenia ograniczeń
  - Warunkiem stopu będzie właściwie znalezienie rozwiązań dopuszczalnych (?)

Inicjalizacja

Znajdujemy rozwiązanie dla zadania

$$min \leftarrow \overline{J(x,u,t)} = J(x,u) + t \cdot R(x,u)$$

1. Jak wybrać wartość współczynnika kary t?
2. Większość algorytmów wymaga inicjalizacji – określenia trajektorii sterowania

- Całkowicie arbitralnie (np. samymi zerami) lub losowo – nie spełniamy ograniczeń
- Z obszaru dopuszczalnego
  - Jak go znaleźć?

Iteracja pośrednia – rozwiązanie nie spełnia ograniczeń

Można zwiększyć współczynnik kary

- O stałą wartość $t^{(j+1)} = t^{(j)} + \beta$
  - Wolno dochodzimy do ograniczeń
- Proporcjonalnie do poprzedniej wartości $t^{(j+1)} = \beta t^{(j)}$
  - Czy zbyt duży współczynnik kary może powodować problemy?

Teoretycznie nie, bo $min \leftarrow \overline{J(x,u,t)} = J(x,u) + t \cdot R(x,u)$
$R(x,u) = 0$, jeśli ograniczenie jest spełnione
$R(x,u) > 0$, jeśli ograniczenie nie jest spełnione

Ale zazwyczaj określa się dopuszczalną odchyłkę od ograniczeń – wtedy dopuszczamy, że w rozwiązaniu końcowym $R(x,u) \neq 0$

Alternatywa

- Można uzależnić zmianę współczynnika kary od tego, czy odpowiednio szybko zbliżamy się do ograniczeń – jeśli tak, to nie będziemy go zwiększać
- Ale wtedy musimy wprowadzić inną modyfikację, bo jeśli nie zmienimy współczynnika kary, to pozostaniemy przy rozwiązaniu, które nie jest dopuszczalne
  - Przesuwny funkcjonał kary

Przesuwny funkcjonał kary - idea

- Wprowadzamy osobną, liniową miarę przekroczenia ograniczeń
- W każdej iteracji (czyli po wyznaczeniu sekwencji sterowań i odpowiadającej jej trajektorii stanu) sprawdzamy, jak zmieniła się wartość tej miary (w jednej z wersji: jak bardzo przybliżyliśmy się do ograniczeń)
  - Jeśli zmniejsza się ona szybciej, niż w postępie geometrycznym o wybranym ilorazie $\alpha \in (0,1)$ (kolejny parametr algorytmu), to nie zmieniamy współczynnika kary, tylko przesuwamy funkcjonał kary oraz zwiększamy wymaganą dokładność dla następnej iteracji
  - Jeśli nie zmniejsza się odpowiednio szybko, to należy zwiększyć współczynnik kary β razy ($\beta > 1$ – kolejny parametr algorytmu) oraz zmniejszyć przesunięcie funkcjonału kary β razy

Przesuwny funkcjonał kary - idea

Uwaga:

Warunek, który jest sprawdzany, nie jest bezpośrednio sformułowany jako sprawdzenie, jak blisko jesteśmy spełnienia oryginalnych ograniczeń, ale jak szybko do nich zdążamy !

- (w zasadzie, poza pierwszą iteracją)
- Stąd konieczność wprowadzenia w algorytmie dwóch, a nie jednego parametru o charakterze wartości progowej
  - ε, wykorzystywanego jako warunek stopu (jesteśmy wystarczająco blisko ograniczeń, żeby zaakceptować rozwiązanie)
  - $c^{(j)}$, wykorzystywanego do określenia sposobu przesuwania ograniczeń i modyfikacji zmiennych parametrów ($c^{(j)}$ oraz $t^{(j)}$)

Przesuwanie ograniczeń

Oryginalne ograniczenia:

$$h_i(x_i, u_i) \leq a_i \qquad | \qquad g_i(x_i, u_i) = b_i$$

Liniowa miara tego, jak daleko jesteśmy od ograniczenia w pierwszej iteracji (po wyznaczeniu trajektorii stanu, odpowiadającej $u^{(0)}$):

$$r_i^{(1)} = \max\left(0, h_i\left(x_i^{(0)}, u_i^{(0)}\right) - a_i\right) \qquad | \qquad r_i^{(1)} = g_i\left(x_i^{(0)}, u_i^{(0)}\right) - b_i$$

Sprawdzamy, czy jesteśmy wystarczająco blisko ograniczeń, tzn. czy $\left\|r^{(1)}\right\| \leq c^{(1)}$. Jeśli warunek jest spełniony, to przesuwamy ograniczenia (bez zmiany współczynnika kary - $t^{(2)} = t^{(1)}$):

$$v_i^{(2)} = a_i - r_i^{(1)} \qquad | \qquad v_i^{(2)} = b_i - r_i^{(1)}$$

Zmieniamy również próg „bliskości” ograniczeń $c^{(2)} = \alpha c^{(1)}$

Przesuwanie ograniczeń – c.d.

Jeśli warunek $\left\|r^{(1)}\right\| \leq c^{(1)}$ nie jest spełniony, to inaczej przesuwamy ograniczenia i zwiększamy współczynnik kary, natomiast próg bliskości ograniczeń się nie zmienia ($c^{(2)} = c^{(1)}$)

$$v_i^{(2)} = a_i - \frac{1}{\beta} r_i^{(1)} \quad \Big| \quad v_i^{(2)} = b_i - \frac{1}{\beta} r_i^{(1)}$$

$$t^{(2)} = \beta t^{(1)}$$

Do następnej iteracji, zamiast oryginalnych ograniczeń, wstawiamy odpowiednio

$$R_i^h(x_i, u_i) = \left(h_{ij}(x_i, u_i) - v_i^{(2)}\right) \max\left(0, h_{ij}(x_i, u_i) - v_i^{(2)}\right)$$

Czyli tak, jakby $h_i(x_i, u_i) \leq v_i^{(2)}$

$$R_i^g(x_i, u_i) = \left\|g_i(x_i, u_i) - v_i^{(2)}\right\|^2$$

Czyli tak, jakby $g_i(x_i, u_i) = v_i^{(2)}$

Zbieżność do rozwiązań dopuszczalnych

Jak bardzo przybliżyliśmy się do /oddaliliśmy od ograniczeń?

Interpretację najłatwiej pokazać na przykładzie 1-wymiarowym, w którym narzucono wartość stanu końcowego. (kolejne slajdy, do końca przykładu, zawierają notację dostosowane do takiego przypadku) Przykładowo, niech $x_N = 10$

$$g_N(x_N) = x_N = 10 \qquad a_N = 10$$

g_N zależy tylko od x_N, ponieważ ostatnim sterowaniem jest u_{N-1}
(dlatego w skrypcie ograniczenia dla ostatniego kroku rozpatrywane jest osobno)

Funkcja kary $R(x,u) = |x_N - a_N|^2 = |x_N - 10|^2$

Jeśli w pierwszej iteracji otrzymano optymalną trajektorię, kończącą się np. w $x_N^{(1)} = 30$ (zbyt duża wartość) to przekroczenie ograniczenia

$$r_N^{(1)} = g_N\left(x_N^{(1)}\right) - a_N = x_N^{(1)} - 10 = 20$$

Przykład - przesunięcie funkcji kary

$g_N(x_N) = x_N = 10$ $\quad x_N^{(1)} = 30,$ $\quad r^{(1)} = 20$

Czy jesteśmy wystarczająco blisko ograniczeń?

- Trzeba wprowadzić odpowiedni parametr do algorytmu, $c^{(1)}$

Jeśli $\left|r^{(1)}\right| \leq c^{(1)}$, to pozostawiamy współczynnik kary bez zmian, wprowadzamy tylko przesunięcie kary i nowe $c^{(2)}$

$$v^{(2)} = a_N - r^{(1)} = -10 \qquad c^{(2)} = \alpha c^{(1)}$$

Nowa funkcja kary $R(x,u) = \left|x_N - v^{(2)}\right|^2 = |x_N + 10|^2$

Zamiast $R(x,u) = |x_N - a_N|^2 = |x_N - 10|^2$

W rezultacie, procedura optymalizacyjna będzie próbowała „ściągnąć” x_N do wartości $x_N = -10$!
(w pewnym sensie we właściwym kierunku, bo jest aktualnie za duże, ale trochę na wyrost; tym można się zająć w kolejnych iteracjach)

Przykład - przesunięcie funkcji kary

$g_N(x_N) = x_N = 10$

Załóżmy inny wynik pierwszej iteracji, $x_N^{(1)} = -15$ (zbyt mała wartość)

$r^{(1)} = -25$

$v^{(2)} = a_N - r^{(1)} = 35$

Nowa funkcja kary $R(x,u) = \left|x_N - v^{(2)}\right|^2 = |x_N - 35|^2$

Zamiast $R(x,u) = |x_N - a_N|^2 = |x_N - 10|^2$

W rezultacie, procedura optymalizacyjna będzie próbowała „ściągnąć” x_N do wartości $x_N = 35$!
(jak w poprzednim przykładzie - na wyrost, ale w dobrym kierunku)

Przykład – zmiana wsp. kary

Wróćmy do pierwszego wariantu i zobaczmy, co się dzieje, jeśli warunek $\left|r^{(1)}\right| \leq c^{(1)}$ nie jest spełniony (niech $\beta = 4$)

$g_N(x_N) = x_N = 10$ $\quad x_N^{(1)} = 30,$ $\quad r^{(1)} = 20$

$$v^{(2)} = a_N - \frac{1}{4} r^{(1)} = 5 \qquad t^{(2)} = 4t^{(1)} \qquad c^{(2)} = c^{(1)}$$

Nowa funkcja kary $R(x,u) = \left|x_N - v^{(2)}\right|^2 = |x_N - 5|^2$

W rezultacie, procedura optymalizacyjna będzie próbowała „ściągnąć” x_N do wartości $x_N = 5$, ale z większym współczynnikiem kary za niespełnienie ograniczeń

Przesunięcie funkcji kary w koleinych iteracjach

W kolejnych iteracjach

$$v^{(j+1)} = a_N - r^{(j)}$$

Nowa funkcja kary $R(x,u) = \left|x_N - v^{(j+1)}\right|^2$

Jeśli przyjmiemy, że $v^{(1)} = a_N$ (bo jeszcze nie ma przesunięcia), to nie ma potrzeby rozpatrywania osobno pierwszej i wszystkich kolejnych iteracji – wystarczy zawsze w funkcji kary wstawić $v^{(j)}$

Im bliżej ograniczeń, tym mniejsza wartość $r^{(j)}$,
a więc coraz mniejsze przesunięcie –
zbliżamy się do pierwotnie określonej funkcji kary

Przykład – druga iteracja

Założyliśmy dla przykładu, w którym $x_N^{(1)} = 30$, $\left|r^{(1)}\right| \leq c^{(1)}$, czyli np. (trochę absurdalnie) $c^{(1)} = 20$. Niech parametry algorytmu zostaną przyjęte jako $\alpha = 0.25$, $\beta = 4$.

$$c^{(2)} = \alpha c^{(1)} = 5$$

Załóżmy, że w efekcie przesunięcia kary, w drugiej iteracji otrzymano optymalną trajektorię, kończącą się w $x_N^{(2)} = 20$. Pamiętając, że $v^{(2)} = -10$

$$r_N^{(2)} = g_N\left(x_N^{(1)}\right) - v^{(2)} = x_N^{(1)} - v^{(2)} = 20 - (-10) = 30$$

Warunek $\left|r^{(2)}\right| \leq c^{(2)}$ nie jest spełniony, tak więc

$$v^{(3)} = a_N - \frac{1}{\beta} r^{(2)} = -10 = 10 - 7.5 = 2.5 \qquad t^{(3)} = \beta t^{(2)}$$

Wracamy do ogólnej postaci opisu problemu

Ograniczenia – równościowe (zawsze aktywne)
i nierównościowe

- Wektorowe
- Trzeba stosować odpowiednią normę (zamiast modułu, który był wykorzystany w prezentowanym przykładzie)

Miary przekroczenia ograniczeń

Dla ograniczeń równościowych

$$g_i(x_i, u_i) = a_i \qquad R_i^g(x_i, u_i) = \left\| g_i(x_i, ui) - v_i^{(j)} \right\|^2$$

$$r_i^{(j)} = g_i\left(x_i^{(j)}, u_i^{(j)}\right) - v_i^{(j)}$$

Wektory!

Dla ograniczeń nierównościowych

$$h_i(x_i, u_i) \leq a_i$$

$$r_i^{(j)} = \max\left(0, h_i\left(x_i^{(j)}, u_i^{(j)}\right) - v_i^{(j)}\right)$$

W obu przypadkach $x_i^{(j)}$ i $u_i^{(j)}$ oznaczają odpowiednio wartości zmiennych stanu i sterowania, znalezione w iteracji j

$$v_i^{(1)} = a_i$$

Wypadkowa wartość ograniczeń

W pierwszej iteracji, do wyznaczenia nowej wartości parametrów oraz przesunięcia funkcji kary, wykorzystywany jest warunek

$$\left\|r^{(1)}\right\| \leq c^{(1)}$$

W kolejnych iteracjach:

$$\left\|r^{(j)}\right\| \leq c^{(j)}$$

Można zmodyfikować ten warunek, wprowadzając dla $j \geq 2$

$a^{(j)} = v^{(j)} + r^{(j)}$ - „wypadkowa wartość ograniczeń”

W kolejnych iteracjach sprawdzany jest wtedy warunek

$$\left\|a^{(j)} - a\right\| \leq c^{(j)}$$

$\gamma = \left\|a^{(j)} - a\right\|$określa w takim razie miarę zbieżności

(Ta wersja jest w skrypcie)

Zbieżność do rozwiązań dopuszczalnych

$$\gamma = \left\| a_N^{(j)} - a \right\|$$

Wektor, złożony ze wszystkich wektorów a_i

Kryterium stopu:

$$\gamma \leq \varepsilon$$

Kolejny parametr algorytmu

Właściwie mogłoby być tak:

$$\left\| r^{(j)} \right\| \leq \varepsilon$$

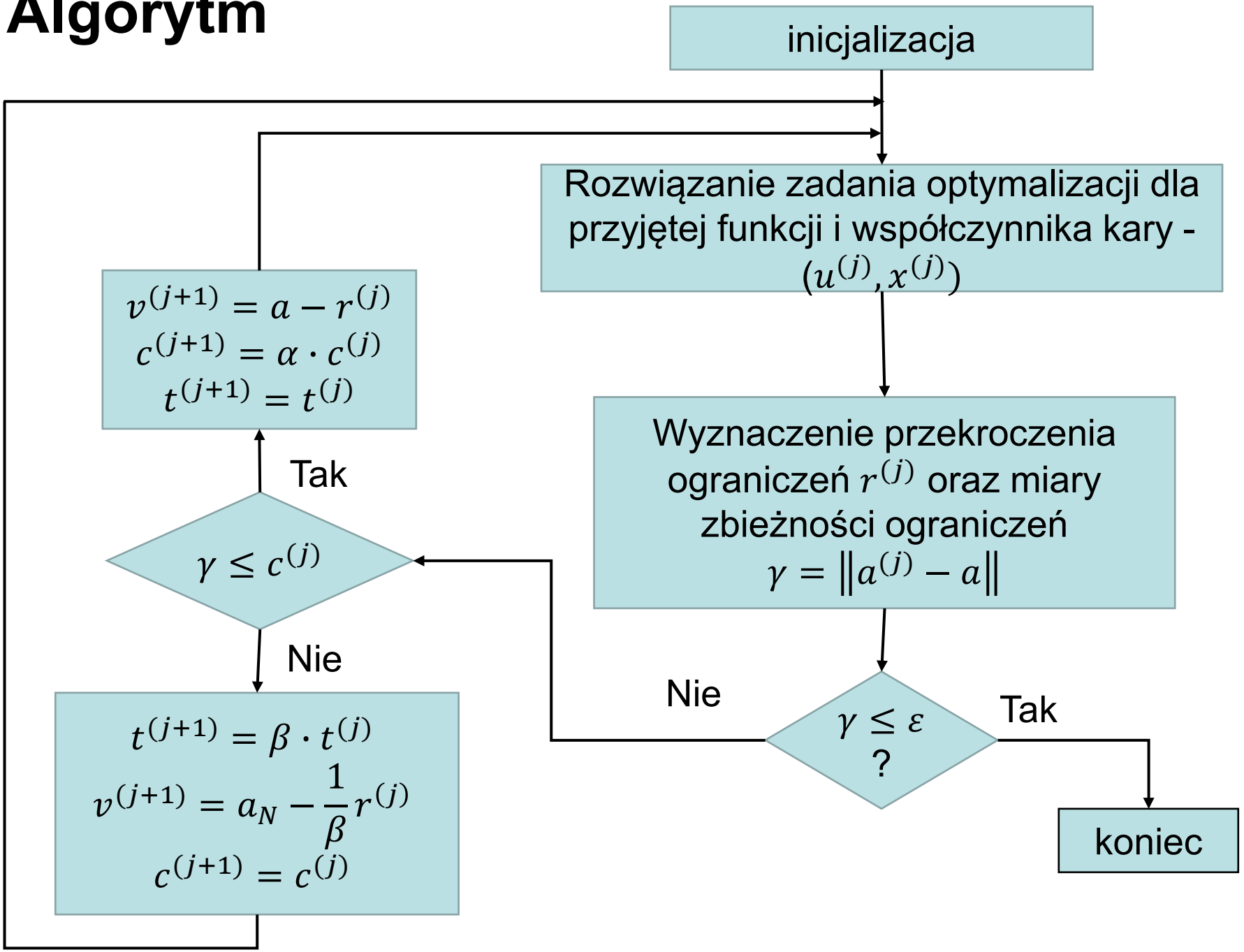
Tak sformułowane kryterium stopu powoduje zatrzymanie algorytmu w momencie spełnienia ograniczeń – dlaczego tak można (co z optymalnością?)

Algorytm

Inicjalizacja

- Sformułowanie zmodyfikowanego funkcjonału $\bar{J}$
- Przyjęcie wartości parametrów, które nie są zmieniane w trakcie działania algorytmu: $\alpha, \beta, \varepsilon$
- Przyjęcie wartości parametrów, które są zmieniane w trakcie działania algorytmu: $c^{(1)}, t^{(1)}$
- $v^{(1)} = a$ (wektor złożony z wartości po prawej stronie kolejnych ograniczeń)

Algorytm


inicjalizacja
Rozwiązanie zadania optymalizacji dla przyjętej funkcji i współczynnika kary - $(u^{(j)}, x^{(j)})$
Wyznaczenie przekroczenia ograniczeń $r^{(j)}$ oraz miary zbieżności ograniczeń $\gamma = \|a^{(j)} - a\|$
$\gamma \leq \varepsilon$?
Nie
Tak
koniec
$\gamma \leq c^{(j)}$
Tak
Nie
$v^{(j+1)} = a - r^{(j)}$
$c^{(j+1)} = \alpha \cdot c^{(j)}$
$t^{(j+1)} = t^{(j)}$
$t^{(j+1)} = \beta \cdot t^{(j)}$
$v^{(j+1)} = a_N - \frac{1}{\beta} r^{(j)}$
$c^{(j+1)} = c^{(j)}$