4. Wykład IV: Metody gradientowe

Rozważmy zadanie poszukiwania minimum w przestrzeni *n*-wymiarowej

$F(x): R^n \to R^1,\ x \in X_0 \subset R^n; X_0$ - zbiór otwarty.

Niech $F(x) \in C^1$, a zatem w każdym punkcie zbioru $x \in X_0$ można określić gradient:

$$\nabla F(x) = \left[\frac{\partial F(x)}{\partial x_1} \quad \frac{\partial F(x)}{\partial x_2} \quad \dots \quad \frac{\partial F(x)}{\partial x_n} \right]^T .$$

Kierunek: $-\nabla F(x)|_{\hat{x}}$ jest nazywany **kierunkiem najszybszego spadku** w punkcie $\hat{x}$.

Przykład 4.1

$$F(x) = x_1^2 + 4x_2^2$$

Gradient $\nabla F(x) = \begin{bmatrix} 2x_1 \\ 8x_2 \end{bmatrix}$ pozwala określić kierunek najszybszego spadku w dowolnym punkcie $x \in R^2$.

Krzywą najszybszego spadku można opisać równaniem różniczkowym (na ogół nieliniowym) o postaci:

$$\frac{dx(s)}{ds} = -k\nabla F(x(s)), \quad x(0) = x_0 .$$

Trudności w rozwiązaniu takiego równania doprowadziły do procedur numerycznych aproksymujących punktowo krzywą najszybszego spadku.

Dla przykładu 4.1 krzywa najszybszego spadku jest opisana równaniem

$$x_2 = e^c x_1^4 ,$$

gdzie *c* jest stałą.

4.1 Elementarny algorytm gradientowy

Alg 4.1:

1. Określ kierunek poszukiwań $d^k = \nabla F(x^k)$.

2. Wykonaj krok: $x^{k+1} = x^k + \lambda^k d^k$ przy $\lambda^k \in R^1$, przy czym:

   $\lambda^k > 0$ dla maksymalizacji,

   $\lambda^k < 0$ dla minimalizacji.

   3. Sprawdź kryterium STOP-u. Gdy nie spełnione: $x^k = x^{k+1}$ i wróć do pkt. 1. Gdy spełnione kryterium STOP-u, zakończ iteracje.

Wybranie stałej długości kroku λ^k nie jest dobrym rozwiązaniem. Jedną z popularnych metod racjonalnego doboru długości kroku λ^k jest zastosowanie „poszukiwania wzdłuż kierunku".

Przykład 4.2 Analityczna metoda poszukiwania na kierunku

Znaleźć minimum formy kwadratowej:

$$F(x) = \frac{1}{2} x^T A x, \text{ gdzie: } A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix},$$

w punkcie: $x^0 = [10;-5]^T$ i na kierunku: $d^0 = -\nabla F(x^0) = [-5;0]^T$.

$$F(x^0 + \lambda d^0) = \frac{1}{2}[10-5\lambda;-5]\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 10-5\lambda \\ -5 \end{bmatrix} = \frac{1}{2}[50 - 75\lambda + 50\lambda^2]$$

Z warunków optymalności zastosowanych do funkcji $F(\lambda)$ wynika:

$$\frac{dF(x^0 + \lambda d^0)}{d\lambda} = 0,$$

$$\frac{d^2 F(x^0 + \lambda d^0)}{d\lambda^2} = 100 > 0,$$

co oznacza, że jest to minimum, a optymalny krok wynosi: $\hat{\lambda} = 0.75$.

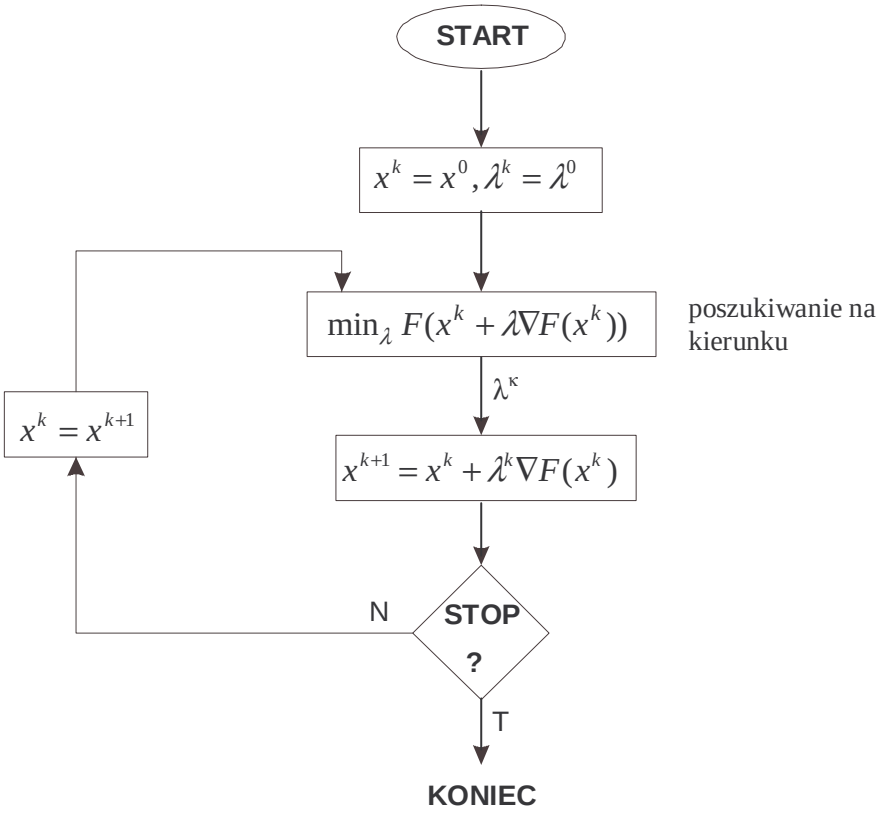
Zmieniając w kolejnych krokach kierunki poszukiwań d^k dobieramy wielkość kroku λ^k w taki sposób, aby funkcja celu była minimalizowana na tych kierunkach. Przedstawione w Przykładzie 4.2 rozwiązanie analityczne zazwyczaj zastępuje się przybliżonym algorytmem numerycznym.

Gradientowe metody poszukiwań różnią się od siebie sposobem wyboru kierunku d^k oraz stosowaną metodą minimalizacji kierunkowej.

4.2 Metoda najszybszego spadku.

Metoda ta wykorzystuje zlinearyzowany model funkcji celu. Rys.4.1 przedstawia jej schemat.

Alg.4.2



Rys. 4.1 Metoda najszybszego spadku

4.3 Kryteria STOP-u

Ważnym elementem tej metody – jak i innych numerycznych algorytmów optymalizacji – są „kryteria stopu", czyli testy sprawdzające zaistnienie warunków zakończenia iteracji zadania. Najczęściej stosowane kryteria to:

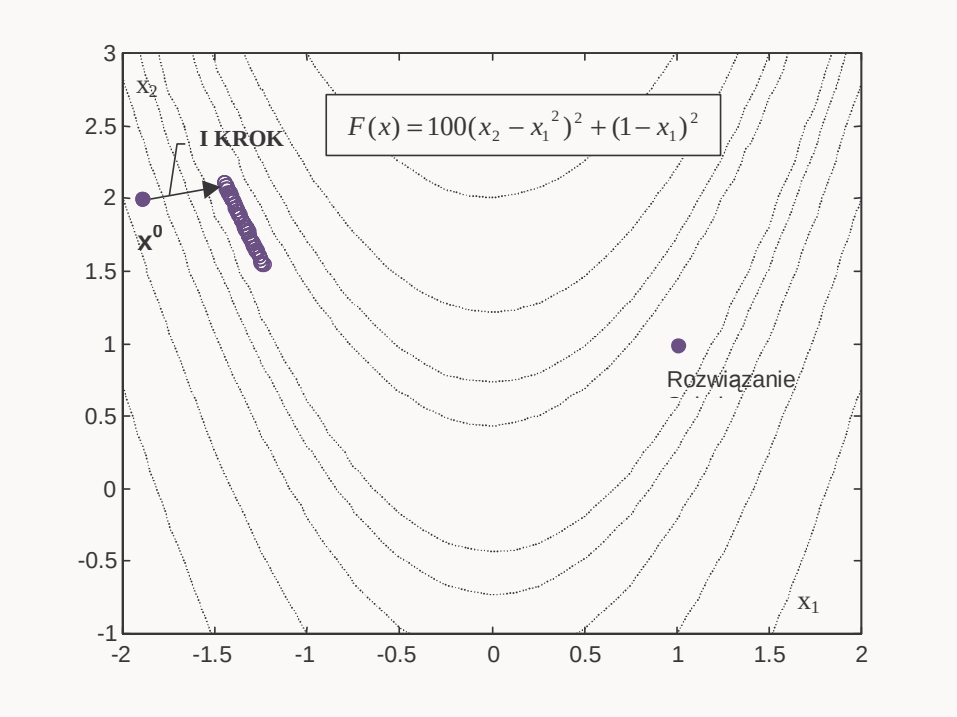
1. $\left\|\nabla F(x^k)\right\| < \varepsilon_1$ - test stacjonarności,
2. $\left\|x^k - x^{k-1}\right\| < \varepsilon_2$ - test szybkości zbieżności, często zastępowany przez: $\left|F(x^k) - F(x^{k-1})\right| \le \varepsilon_2$,
3. $k > K_0$ - test liczby iteracji.\,
4. test odległości pomiędzy rozwiązaniem prymalnym i dualnym (patrz rozdział 8).

Kryteria mogą być ze sobą łączone w ramach jednego algorytmu poszukiwania ekstremum.

Metoda najszybszego spadku charakteryzuje się dobrą zbieżnością w dużej odległości od punktu rozwiązania, ale przy zbliżeniu się do rozwiązania jej zbieżność staje się wolna.

Przykład 4.3

Rys.4.2 przedstawia wynik zastosowania metody najszybszego spadku dla minimalizacji funkcji Rosenbrocka [MATLAB].



Rys. 4.2 57 iteracji metody najszybszego spadku

4.4 Poszukiwanie na kierunku

Jak wynika z Rys.4.1 rozwiązanie zadania poszukiwania na kierunku $\min_\lambda F(\lambda) = \min_\lambda (x^k + \lambda \nabla F(x^k))$ jest podstawową „cegiełką" metody najszybszego spadku, ale także wielu innych gradientowych i bezgradientowych algorytmów optymalizacji, z których niektóre są omówione w kolejnych rozdziałach. Generalnie, polega ona na znalezieniu takiej długości kroku w zadanym kierunku, aby funkcja $F(\lambda)$ osiągała minimum. W praktyce poszukiwanie na kierunku jest metodą iteracyjną, która generuje ciąg $\{\Lambda_i\}$ oszacowań długości kroku spełniających założone warunki zakończenia poszukiwania na kierunku. Poszukiwanie zostaje zakończone, gdy przedział zmniejszy się poniżej zadanej wartości lub też, gdy długość kroku spełniająca zadane warunki nie istnieje.

Zadania poszukiwania na kierunku są rozwiązywane zarówno dla zadań bez ograniczeń jak i zadań z ograniczeniami. Odpowiednio są zatem formułowane zadania poszukiwania na półprostej jaki zadania poszukiwania na odcinku.

Znane i stosowane są liczne metody numeryczne rozwiązania problemu poszukiwania na kierunku [Find 80, Wit 86, Kor 92 Stach 00]. Ponieważ są to procedury wielokrotnie wywoływane (najgłębiej „zagnieżdżone w pętlach algorytmu optymalizacji) ich efektywność numeryczna jest decydująca, jeśli chodzi o efektywność całego algorytmu optymalizacji.

Dla zadania: $F(x): R^n \to R^1$, $x \in X_0 \subseteq R^n$, gdzie X_0 - zbiór dopuszczalny, gdy dane są $x_0 \in X_0$ oraz kierunek $d \in X_0$ to formułuje się dwa rodzaje zadań poszukiwania na kierunku.

Poszukiwanie na prostej

$x \in R^n = X_0$

$x^p = \{x : x = x_0 + \lambda \cdot d\}$

$\lambda \in R^1$

Minimalizacja $F(x)$ na prostej:

$$\min_{\lambda \in R^1} F(x_0 + \lambda \cdot d)$$

Poszukiwanie na odcinku domkniętym

X_0 - zbiór zwarty

$\bar{x}^p = \{x : x = x_0 + \lambda \cdot d\}$

$\lambda \in \Lambda_0 \subset R^1, \quad \Lambda_0 = \{\lambda : x \in X_0\}$

Minimalizacja $F(x)$ na odcinku

$$\min_{\lambda \in \Lambda_0 \subset R^1} F(x_0 + \lambda \cdot d)$$

$\Lambda_0 = [0, \lambda_M]$

Zadaniem algorytmu numerycznego jest znalezienie podzbioru oszacowań optymalnych wartości kroku $\Lambda \subset \Lambda_0$, $\Lambda_0 = [0, \lambda_M]$. Elementy tego podzbioru ten nie tylko muszą spełniać warunki poprawy $F(x^k + \lambda^k d^k) < F(x^k)$, ale także pewne dodatkowe warunki zwane **testami**. Dokładniej mówiąc, elementy podzbioru Λ są określane w sposób pośredni za pomocą testów. Testy umieszczone w algorytmie poszukiwania na kierunku pełnią rolę kryterium stopu.

Metody poszukiwania na kierunku klasyfikuje się według zapotrzebowania na informację o minimalizowanej funkcji. Są metody wymagające tylko znajomości wartości funkcji (**bezgradientowe**), w odróżnieniu od metod wymagających informacji o pochodnej funkcji (**gradientowych**). Te ostatnie umożliwiają stosowanie testów zbieżności omówionych w rozdziale następnym (np. testu Goldstaina).

Można też mówić o metodach „dokładnych", które doprowadzają nas do bliskiego otoczenia rozwiązania zadania poszukiwania na kierunku (np. interpolacja kwadratowa, złoty podział), jak i o metodach przybliżonych, które tylko poprawiają wartość funkcji celu na kierunku.

Praktyczne algorytmy numeryczne poszukiwania na kierunku są zazwyczaj kombinacja oszacowań, testów i metod dokładnych.

4.5 Oszacowanie długości kroków i testy zbieżność algorytmów gradientowych

O zbieżności algorytmu 4.1 lub 4.2 decyduje **wybór kierunku** *d*, w którym wykonywany jest krok jak również **długość tego kroku** λ.

Musi być spełnione warunek poprawy :

$$F(x^{k+1}) < F(x),$$

lub

$$F(x^k + \lambda^k d^k) < F(x^k).$$

Rozwijając w szereg Taylora otrzymujemy

$$F(x^k) + \lambda^k \nabla F(x^k)^T d^k < F(x^k),$$

lub

$$\lambda^k \nabla F(x^k)^T d^k < 0 .$$

Ponieważ λ^k jest zawsze przyjmowane jako dodatnie, to jednym ze wskaźników umożliwiających badanie zbieżności (czyli zachodzenia powyższego warunku)jest iloczyn:

$$\cos^2 \theta^k \left\| \nabla F(x^k) \right\|^2 , \tag{4.1}$$

gdzie θ jest kątem pomiędzy kierunkiem *d*, a kierunkiem $-\nabla F$ (antygradientem).

Jeśli ma być zachowany warunek $\cos^2 \theta^k \left\| \nabla F(x^k) \right\|^2 \underset{k\to\infty}{\Rightarrow} 0$, a wybór kierunku poszukiwań jest taki, że:

$$\cos\theta^k \geq \delta > 0 , \text{ dla każdego } k, \text{ gdzie } \delta \text{ - stała dodatnia,}$$

to z warunku $\cos^2 \theta^k \left\| \nabla F(x^k) \right\|^2 \underset{k\to\infty}{\Rightarrow} 0$ wynika:

$$\lim_{k\to\infty} \left\| \nabla F(x^k) \right\| = 0 .$$

Warunki na długość kroku λ określają odpowiednie testy poprawy (Armijo, Wolfa i Goldsteina). Problem polega na tym, że prosty warunek w postaci:

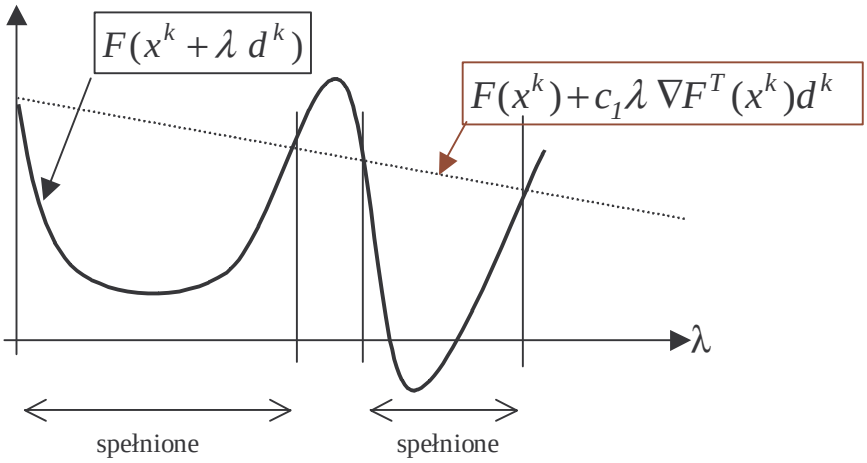
$$F(x^k + \lambda^k d^k) < F(x^k)$$

nie zawsze zapewnia zbieżność do lokalnego minimum (mogą być inne niż minimum lokalne punkty skupienia ciągu $\{F(x^k)\}_{k=0}^{\infty}$ przy niedostatecznie dużej redukcji wartości funkcji $F(x)$). Konieczna jest zatem kontrola górnej i dolnej wartości długości kroku λ.

Warunek poprawy Armijo jest w postaci:

$$F(x^k + \lambda^k d^k) \leq F(x^k) + c_1 \lambda^k \nabla F^T(x^k) d^k \tag{4.2}$$

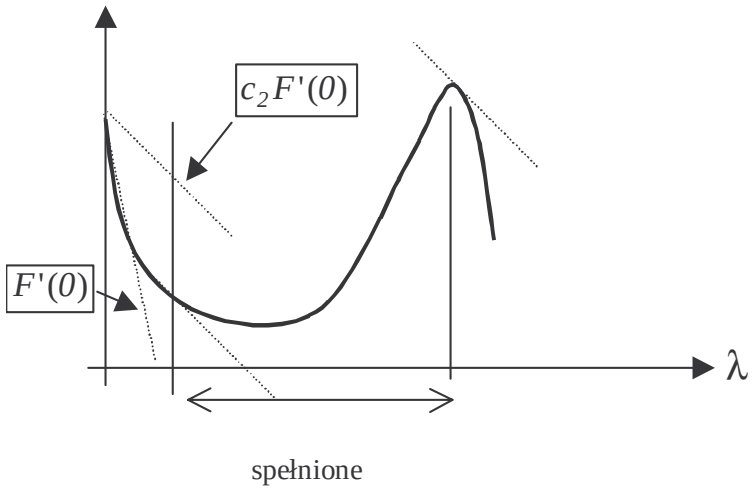
Oznacza on tylko tyle, że redukcja wartości funkcji celu powinna być proporcjonalna zarówno do długości kroku jak i pochodnej kierunkowej $\nabla F^T(x^k) d^k$.



Rys. 4.3 Warunek dostatecznej poprawy Armijo

Warto zwrócić uwagę, że warunek ten nie gwarantuje szybkości zbieżności algorytmu, bowiem jak wynika z rys. 4.3 zawsze może być spełniony dla dostatecznie małego λ. Aby zapobiec nadmiernej redukcji kroku, test powyższy uzupełnia się zatem testem „krzywizny" funkcji w postaci (rys.4.4):

$$\nabla F(x^k + \lambda^k d^k)^T d^k \geq c_2 \nabla F^T(x^k) d^k \quad \text{dla} \quad c_2 \in (c_1, 1), \text{ dla } c_1 \in (0,1). \qquad (4.3)$$



Rys. 4.4 Warunek dostatecznej krzywizny

Test ten oznacza, że jeśli w punkcie początkowym spadek jest duży,to nie dopuszcza się małych kroków.

Połączone testy poprawy i krzywizny, tworzą **test Wolfa**:

$$F(x^k + \lambda^k d^k) \le F(x^k) + c_1 \lambda^k \nabla F^T(x^k) d^k \text{, dla } c_1 \in (0,1),$$

$$\nabla F(x^k + \lambda^k d^k)^T d^k \ge c_2 \nabla F^T(x^k) d^k \text{ dla } \quad c_2 \in (c_1,1).$$

Tw.4.1 (o zbieżności)

Załóżmy dla dowolnej iteracji w postaci: $x^k + \lambda^k d^k$ (d^k - kierunek spadku), że funkcja F jest ograniczona w R^n i jest klasy C^1 w otwartym zbiorze N zawierającym zbiór poziomicowy $L = \{x : F(x) \le F(x_0)\}$, gdzie x_0 jest punktem początkowym iteracji. Załóżmy także, że gradient ∇F spełnia warunek Lipschitza na N, tzn. istnieje dodatnia stała l taka, że:

$\|\nabla F(x) - \nabla F(x')\| \le l\|x - x'\|$ dla każdego $x, x' \in N$.

O ile λ^k spełnia warunki Wolfe'a, to zachodzi:

$$\sum_k \cos^2 \theta^k \left\|\nabla F(x^k)\right\|^2 < \infty. \tag{4.4}$$

Z nierówności (4.4) wynika $\cos^2 \theta^k \left\|\nabla F(x^k)\right\|^2 \Rightarrow 0$ dla $k \to \infty$ **(**w.k. zbieżności szeregów**),** a zatem Twierdzenie 4.1 gwarantuje zatem zbieżność w sensie $\lim_{k\to\infty} \left\|\nabla F(x^k)\right\| = 0$ (do otoczenia lokalnego minimum), o ile zastosowana metoda poszukiwania w kierunku spełnia warunki poprawy Wolfa (Goldsteina) oraz kierunek poszukiwań spełnia w każdym kroku $\cos\theta^k \ge \delta > 0$ (bowiem $lim\{\cos^2 \theta^k \left\|\nabla F(x^k)\right\|^2\} = lim \cos^2 \theta^k \, lim \left\|\nabla F(x^k)\right\|^2$**)**.

Przykładowo, w metodzie najszybszego spadku w każdym kroku jest $\cos\theta = 1$, a zatem o ile długość kroku będzie spełniała warunki testu Wolfa, to twierdzenie 4.1 gwarantuje zbieżność tej metody.

Z Twierdzenia 4.1 wynika również, że dla uzyskania zbieżności ciągu rozwiązań zadania poszukiwania na kierunku, minima na kierunku nie muszą być „dokładnymi" rozwiązaniami

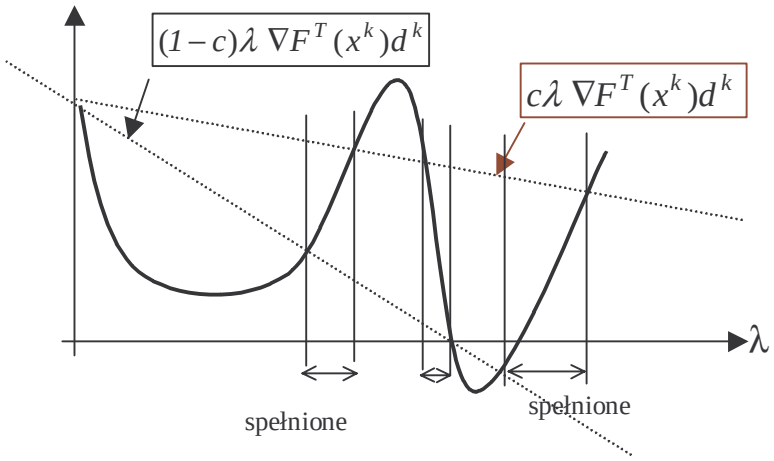
zadań minimalizacji, a jedynie krokami w kierunku rozwiązania, spełniającymi odpowiednie warunki.

Test Goldsteina, podobnie jak test Wolfe'a gwarantuje zmniejszanie się wartości funkcji celu, przy równoczesnym zabezpieczeniu przed wykonywaniem zbyt małych kroków. Warunki te są w postaci:

$$F(x^k)+(1-c)\lambda^k \nabla F^T(x^k)d^k \le F(x^k+\lambda^k d^k) \le F(x^k)+c\ \lambda^k \nabla F^T(x^k)d^k .$$

gdzie $0 < c < \frac{1}{2}$.

Prawa część nierówności jest testem spadku znanym z kryterium Wolfe'a, a zatem kontroluje górną granicę długości kroku. Lewa część jest wprowadzona dla kontroli dolnej granicy długości kroku (Rys.4.5).



Rys. 4.5 Test Goldsteina

Należy zwrócić uwagę, że test Goldsteina może zostać wykorzystany jako kryterium stopu dla algorytmu poszukiwania na kierunku. Wystarczy w tym celu przyjąć odpowiednie wartości stałej *c*.

Typowy numeryczny poszukiwania optymalnej długości kroku składa się z następujących etapów:

1. Wyznaczenie przedziału $\Lambda_0 = [\lambda_a, \lambda_b]$ zawierającego pożądany krok, oraz określenie kroku początkowego λ_0,

2. Redukcja przedziału Λ_0, w celu znalezienia kroku o długości spełniającej warunki testu (np. kontrakcja)
3. Interpolacja, wykonana w celu określenia $\hat{\lambda}$ bliskiego minimum.

Metoda kontrakcji

Alg 4.3:

1. Wybierz $\lambda^0 > 0,\ \rho, c \in (0,1),$ podstaw $\lambda \Leftarrow \lambda^0$.
2. Sprawdź czy $F(x^k + \lambda d^k) \le F(x^k) + c\lambda\, \nabla F^T(x^k) d^k$ Gdy spełnione, STOP.

Gdy nie, idź do pkt.3

3. Zredukuj krok: $\lambda \Leftarrow \rho\lambda$. Wróć do pkt.2.

W metodzie tej krok początkowy jest wybierany $\lambda^0 = 1$, jak ma to miejsce w metodzie Newtona, lub może przyjmować inną wartość. Metoda kontrakcji gwarantuje, że albo krok pozostanie równy wartości początkowej, albo ulegnie skróceniu tak, aby spełniony został warunek dostatecznej poprawy. Krok nigdy nie ulegnie nadmiernemu skróceniu, a zatem nie istnieje konieczność stosowania dodatkowych testów.

Interpolacja

Interpolacja jest zazwyczaj jedną z faz metody algorytmu minimalizacji na kierunku. Na podstawie znajomości wartości funkcji i jej pochodnej na końcach przedziału określamy kwadratową lub sześcienną aproksymację funkcji dla punktów wewnętrznych przedziału. Można zatem podać odpowiednie zależności wykorzystujące wartości funkcji (pochodnej funkcji) w punktach brzegowych dla określenia przedziału.

Oznaczmy: $\Phi(\lambda) = F(x^k + \lambda\ d^k)$

Niech celem poszukiwań będzie znalezienie kroku spełniającego warunek Armijo dla kierunku gradientu

$$\Phi(\lambda^k) \le \Phi(0) + c\lambda^k \Phi'(0) .$$

Przyjmijmy, że początkowa długość przedziału λ^0 jest dana. Gdy jest spełnione:

$$\Phi(\lambda^0) \leq \Phi(0) + c\lambda^0\Phi'(0),$$

to kończymy poszukiwania. W przeciwnym przypadku wiemy, że $[0, \lambda^0]$ zawiera akceptowalną długość przedziału poszukiwań (Rys.4.2).

Tworzymy teraz aproksymację kwadratową $\Phi_q(\lambda)$ funkcji wykorzystując dostępną informację, tzn. wartości funkcji $\Phi(0), \Phi(\lambda_0)$ oraz pochodną $\Phi'(0)$:

$$\Phi_q(\lambda) = a\lambda^2 + b\lambda + c,$$

gdzie:

$$a = \frac{\Phi(\lambda^0) - \Phi(0) - \lambda^0\Phi'(0)}{(\lambda^0)^2},$$

$$b = \Phi'(0),$$

$$c = \Phi(0).$$

Powyższa aproksymacja spełnia warunki brzegowe, tzn.

$$\Phi_q(0) = \Phi(0), \quad \Phi_q'(0) = \Phi'(0), \Phi_q(\lambda^0) = \Phi(\lambda^0).$$

Stąd można wyliczyć nowy krok, jako minimum powyższej funkcji kwadratowej:

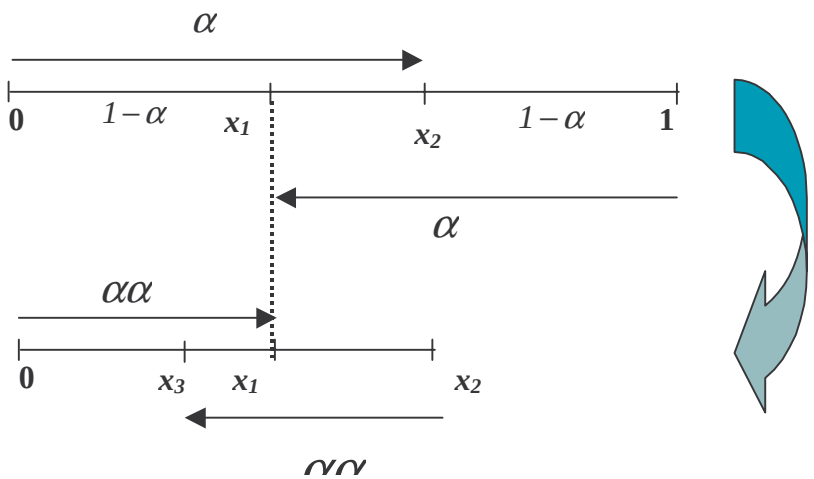
$$\lambda_1 = -\frac{\Phi(0)(\lambda^0)^2}{2[\Phi(\lambda^0) - \Phi(0) - \Phi'(0)\lambda^0]}.$$

Gdy warunek (4.2) nie jest spełniony podstawiamy $\lambda_0 = \lambda_1$ i powtarzamy obliczenia.

4.6 Metody bezgradientowe: metoda złotego podziału

Metody te są ciągle popularne w wielu algorytmach optymalizacji, choć rozwój możliwości różniczkowania symbolicznego znacznie zmniejszył ich znaczenie. Zaletą jest prostota algorytmów, zaś wadą brak ogólnej teorii oraz **założenie o unimodalności** funkcji celu.

Metoda złotego podziału polega na redukcji podprzedziału zawierającego poszukiwane minimum o stały współczynnik α, z zachowaniem w kolejnej iteracji informacji o wartości funkcji celu w jednym z punktów wewnętrznych. Można łatwo wyliczyć, ile taki współczynnik α powinien wynosić (Rys.4.6).



Rys. 4.6 Wyznaczenie współczynnika zmniejszania kroku dla metody złotego podziału. Punkt wewnętrzny x_1 zostaje zachowany do drugiej iteracji. Początkowa długość przedziału wynosi 1

Na rysunku 4.5 długość odcinka pierwotnego po jednej iteracji można określić jako:

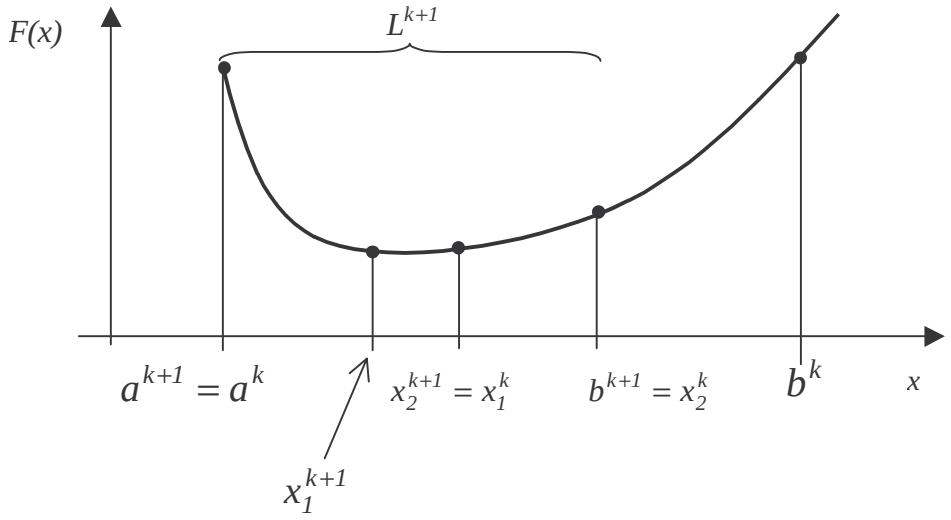
$$(1-\alpha)+\alpha-\alpha^2+(1-\alpha)=1.$$

Rozwiązanie tego równania daje: $\alpha = 0.618$

Ogólnie, w *k*-tej iteracji będzie:

$$\frac{x_2^k - a^k}{b^k - a^k} = \frac{b^k - x_1^k}{b^k - a^k} = 0.618,$$

gdzie a^k i b^k są końcami przedziału *k*-tej iteracji (Rys.4.7).



Rys. 4.7 Jedna iteracja algorytmu złotego podziału

Alg 4.4: (pojedyncza iteracja)

$$Gdy\ F(x_2^k) > F(x_1^k):$$
$$b^{k+1} = x_2^k$$
$$a^{k+1} = a^k$$
$$x_2^{k+1} = x_1^k$$
$$x_1^{k+1} = a^{k+1} + (1-\alpha)(b^{k+1} - a^{k+1})$$

$$Gdy\ F(x_1^k) > F(x_2^k):$$
$$b^{k+1} = b^k$$
$$a^{k+1} = x_1^k$$
$$x_1^{k+1} = x_2^k$$
$$x_2^{k+1} = b^{k+1} + (1-\alpha)(b^{k+1} - a^{k+1})$$

Dla algorytmu złotego podziału podaje się następujące oszacowanie przedziału zawierającego minimum po *n* iteracjach:

$$\frac{L^n}{L^1} = \alpha^{n-1}.$$

4.7 Algorytm Newtona – Rapsona

Rozważmy zadanie minimalizacji funkcji:

$$F(x): R^n \rightarrow R^1,$$

Aproksymacja $F(x)$ szeregiem Taylora (w otoczeniu x^k) jest w postaci:

$$\tilde{F}_k(x) = F(x^k) + \nabla^T F(x^k)(x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 F(x^k)(x - x^k) + ... \qquad (4.4)$$

Minimum formy kwadratowej (4.4) w punkcie x^{k+1} spełnia warunek:

$$\nabla \tilde{F}_k(x)\Big|_{x=x^{k+1}} = 0$$

$$\nabla \tilde{F}_k(x^{k+1}) = \nabla F(x^k) + \underbrace{\nabla^2 F(x^k)}_{H(x^k)} \cdot (x^{k+1} - x^k) = 0$$

$$x^{k+1} = x^k - H^{-1}(x^k) \cdot \nabla F(x^k) \qquad (4.5)$$

Ostatnia zależność definiuje algorytm Newtona – Rapsona.

Twierdzenie 4.2:

Ciąg punktów generowanych przez algorytm Newtona – Rapsona jest zbieżny z szybkością drugiego rzędu do $\hat{x}$ (minimum funkcji $F(x)$), o ile w każdym kroku hesjan $H(x^k)$ jest ściśle dodatnio określony, i spełnia (lokalnie) warunek Lipschitza.

Dowód patrz [Wit, 86].

Twierdzenie 4.3:

Minimum ściśle dodatnio określonej formy kwadratowej :

$$F(x) = \frac{1}{2} x^T A x + b^T x + c, \quad x \in R^n,$$

jest osiągane przez algorytm Newtona – Rapsona w jednym kroku.

Dowód:

Z warunku koniecznego optymalności minimum to wynosi:

$$\nabla F(\hat{x}) = A\hat{x} + b = 0$$
$$\hat{x} = -A^{-1}b$$

Z drugiej strony, stosując algorytm Newtona-Rapsona uzyskamy:

$$\nabla^2 F(x^0) = A$$

$$x^1 = x^0 - A^{-1} \cdot \nabla F(x^0) = x^0 - A^{-1} \cdot (Ax^0 + b) = -A^{-1} \cdot b = \hat{x}$$

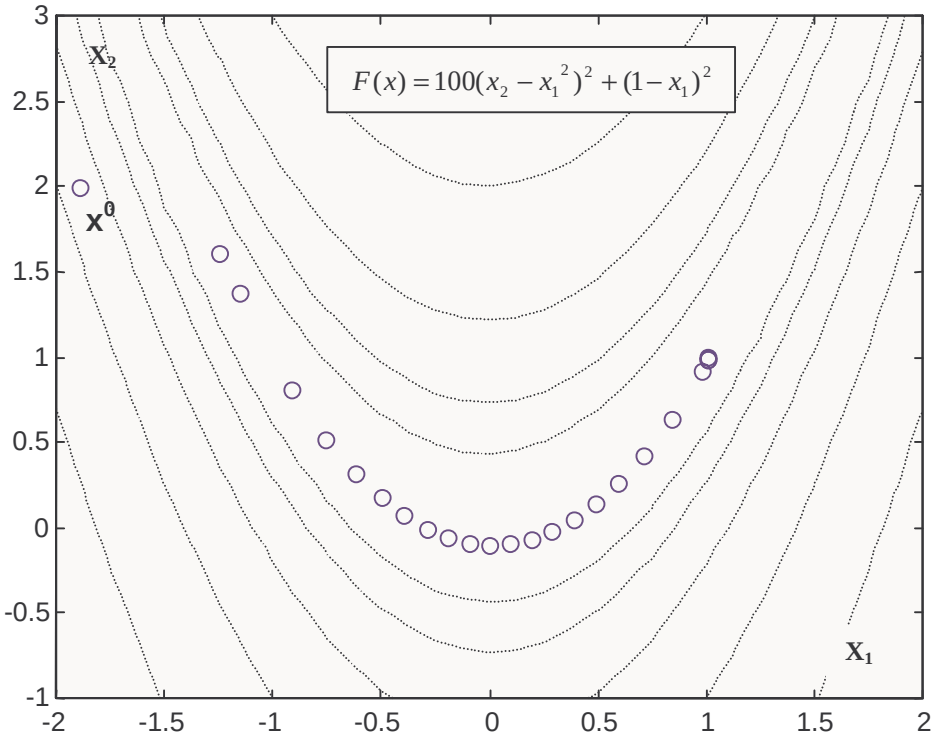
Zalety metody Newtona - Rapsona :

- szybka zbieżność (drugiego rzędu) w otoczeniu rozwiązania

Wady metody Newtona - Rapsona :

- silne założenia dotyczące gładkości $F(x)$,
- uzyskanie analitycznych wzorów na $H(x)$ jest trudne ,
- z powodu konieczności obliczania $H^{-1}(x^k)$ dla pewnych x^k zadanie może stać się źle uwarunkowane
- do obliczeń wymagana jest znaczna ilość pamięci

Przykład 4.4



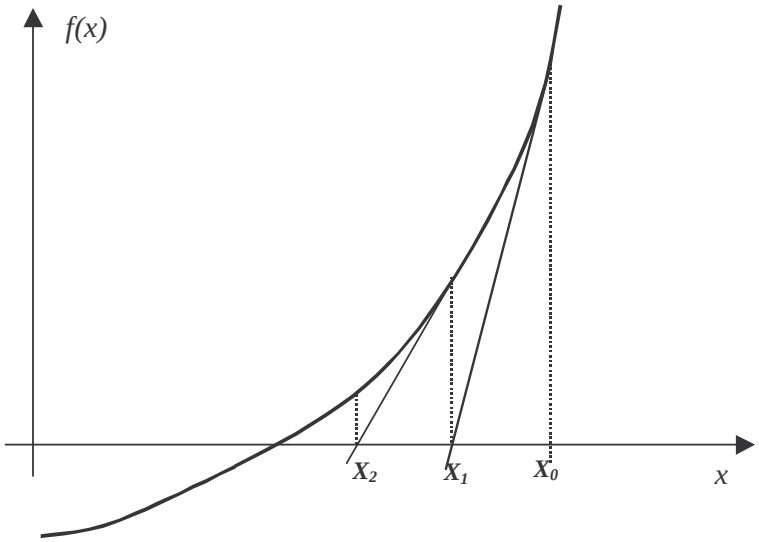
Rys. 4.8 23 iteracje metody Newtona dla funkcji Rosenbrocka

Przykład 4.5

Znana metoda (Newtona) numerycznego poszukiwania rozwiązania równania nieliniowego o postaci $f(x) = 0$ sprowadza się do wykonywania iteracji:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Ilustruje ja rys. 4.9. Jest to zgodne ze wzorem (4.5), jeśli przyjmiemy $f(x) = \nabla F$.



Rys. 4.9 Ilustracja metody Newtona rozwiązywania równań

Modyfikacje metody

Jedna z modyfikacji metody Newtona – Rapsona jest wprowadzenie poszukiwania na kierunku, celem zwiększenia obszaru zbieżności.

$$d^k = -H^{-1}\left(x^k\right)\cdot \nabla F\left(x^k\right),$$

$$x^{k+1} = x^k + \lambda^k d^k .$$

Rozwiązanie równania: $H\left(x^k\right)d^k = -\nabla F\left(x^k\right)$ np. metodą faktoryzacji wymaga wykonania n^3 operacji w każdej iteracji.

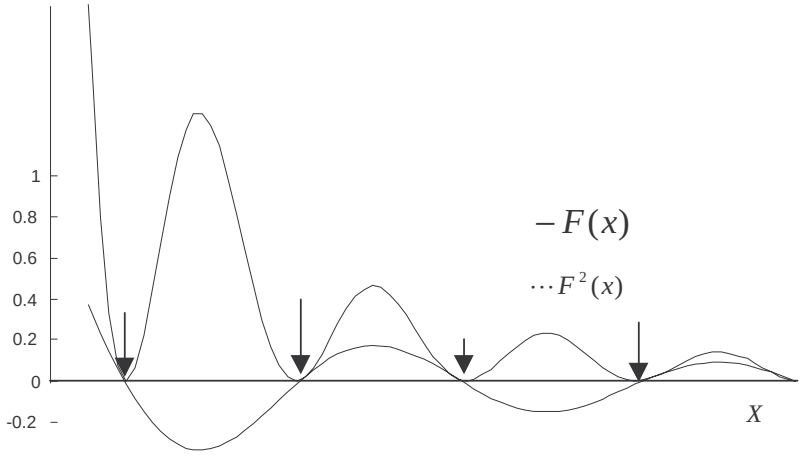
Przykład 4.6

Rozwiązać układ równań liniowych w postaci: $Ax = b,\quad x, b \in R^n$, gdzie A – macierz symetryczna, nieosobliwa.

Tworząc formę kwadratową $H(x) = \frac{1}{2}(Ax-b)^T(Ax-b)$, można zauważyć, że

$\nabla H(x) = A^T(Ax-b)$, a zatem w punkcie rozwiązania zachodzi: $\nabla H(\hat{x}) = 0$ i jest to minimum formy kwadratowej. Można zatem dla znalezienia rozwiązania układu równań liniowych zastosować dowolny algorytm gradientowy.

W innych przypadkach także jest możliwa zamiana zadania poszukiwanie zer funkcji nieliniowej na poszukiwanie minimum odpowiedniej funkcji przekształconej (Rys.4.10).



Rys. 4.10 Poszukiwanie zer wielomianu, a zadanie minimalizacji

4.7. Problemy zbieżności algorytmów

Działanie algorytmu rozpoczyna się od punktu początkowego (lub zbioru punktów początkowych) i może być opisane poprzez *odwzorowania algorytmiczne*, w postaci:

$$x^{k+1} = A\left(x^k, x^{k-1}, ..., x^{k-j}\right),\ k > j$$

lub

$$x^{k+1} = A_1\left(F\left(x^k\right), F\left(x^{k-1}\right), ..., F\left(x^{k-j}\right), x^k, x^{k-1}, ..., x^{k-j}\right),\ k > j$$

Typowe algorytmy optymalizacji charakteryzują się tym, że ciąg kolejno generowanych punktów zbiega x^k się do otoczenia punktu rozwiązania $\hat{x}$. Zbieżność ta może być charakteryzowana poprzez podanie odpowiednich oszacowań.

Def. 4.1:

Algorytm jest zbieżny według normy w przestrzeni R^n, gdy ciąg punktów generowanych w kolejnych iteracjach spełnia:

$$\lim_{k\to\infty}\left\|x^k - \hat{x}\right\| = 0\ .$$

Następująca definicja pozwala określić szybkość zbieżności algorytmów.

Def. 4.2: (Rząd zbieżności algorytmu)

Jeśli algorytm jest zbieżny według normy, to **rzędem zbieżności** nazywamy największą liczbę $p > 0$, dla której wartość wyrażenia :

$$\lim_{k\to\infty} \frac{\left\| x^{k+1} - \hat{x} \right\|}{\left\| x^{k} - \hat{x} \right\|^{p}} = q_p ,$$

jest skończona ($q_p < \infty$), gdzie q_p nazywamy współczynnikiem zbieżności.

- dla $p = 1$, $q_p > 0$ algorytm jest **zbieżny liniowo**,
- dla $p = 1$ i $q_p = 0$ algorytm jest **zbieżny superliniowo**,
- dla $p = 2$ i $q_p \geq 0$ algorytm jest zbieżny **z szybkością drugiego rzędu**.

Powyższą granicę często szacuje się przez wyrażenie:

$$\left\| x^{k+1} - \hat{x} \right\| \leq q_p \cdot \left\| x^{k} - \hat{x} \right\|^{p}$$

Przykładowo, zbieżność metody Newtona – Rapsona oszacowana jest wyrażeniem:

$$\left\| x^{k+1} - \hat{x} \right\| \leq c \cdot \left\| x^{k} - \hat{x} \right\|^{2}$$

Przykład 4.7

Zbieżność ciągu $\{e^{-k!}\}$ jest drugiego rzędu, co wynika z zależności:

$$\lim_{k\to\infty} \frac{(e^{-k!})^{k+1}}{e^{-k!}e^{-k!}} = \lim_{k\to\infty} (e^{-k!})^{k-1} = 0 .$$

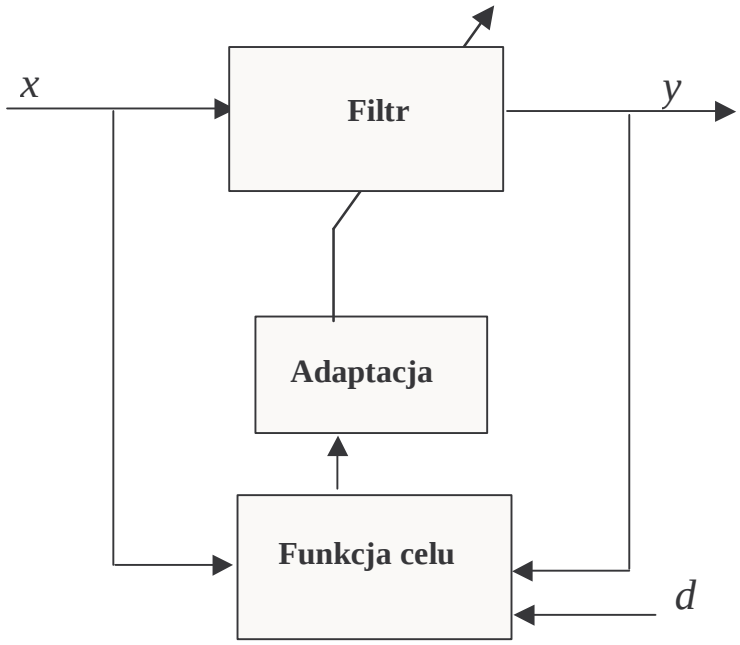
Przykład 4.8

Wykorzystanie metod gradientowych do optymalizacji filtrów adaptacyjnych (Stranneby D., Digital Signal Processing, Elsevier, 2001)

Zastosowania: tłumienie szumów, gdy nie jest znana funkcja filtracji.

Adaptacja parametrów filtra jest realizowana według schematu z rys.4.11.

Sam filtr ma strukturę jak na rys. 4.12. Poszczególne wejścia filtru mogą być uzyskane z pojedynczego wejścia, w wyniku zastosowania linii opóźniającej.



Rys. 4.11 Adaptacja parametrów filtru, d – sygnał wzorcowy.

Funkcja celu jest określana jako:

$$J(W) = E[\varepsilon^2] = E[(d(k) - W^T X(k))(d(k) - W^T X(k))] = E[d^2(k) + W^T X(k) X^T(k) W - 2d(k) X^T(k) W],$$

gdzie odpowiedź filtru (rys.4.12) jest określona jako:

$$y(k) = W^T(k) X(k), \quad X(k) = [x_0(k), x_1(k) \ldots x_L(k)]^T,$$

a *k* oznacza poszczególne kroki modelu dyskretnego.

Zmienność parametrów filtru (adaptacja) jest przyjęta jako znacznie wolniejsza niż zmiany sygnału wejściowego, a zatem $W(k) = W$.

Wartość oczekiwaną błędu (względem *k*) liczymy z zależności:

$$J(W) = E[d^2(k) + W^T X(k) X^T(k) W - 2d(k) X^T(k) W] = E[d^2(k)] + W^T E[X(k) X^T(k)] W +$$
$$- 2E[d(k) X^T(k)] W = E[d^2(k)] + W^T R W - 2P^T W$$

gdzie***:*** ***R*** **–** jest macierzą korelacji pomiędzy poszczególnymi wejściami, ***P*** **–** jest macierzą korelacji wzajemnej wejść i wartości zadanej.

Zadaniem procedury optymalizacji jest określenie najlepszych nastaw współczynników wag filtru, a więc jak najlepsza adaptacja.

Funkcja celu (błąd) jest forma kwadratową, ściśle dodatnio określoną, a więc istnieje jedno globalne minimum. Jeśli wartość zadana i wejścia nie są skorelowane, to ***P****=0* i minimalna wartość błędu jest uzyskiwana dla *W=0* , czyli całkowicie wyłączając filtr. Oznacza to wtedy, że albo sygnał *d(k)* albo struktura filtru, albo oba te elementy są nieodpowiednie.

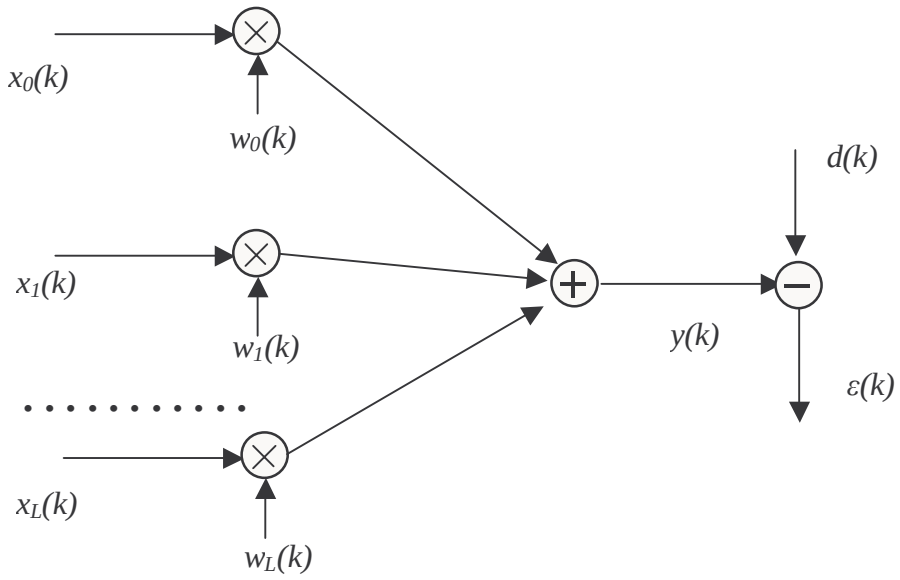
Gdy $P \neq 0$ to gradient tak zaproponowanej funkcji celu jest w postaci:

$$\nabla(J) = 2RW - 2P\,,$$

i spełnia warunek konieczny optymalności dla :

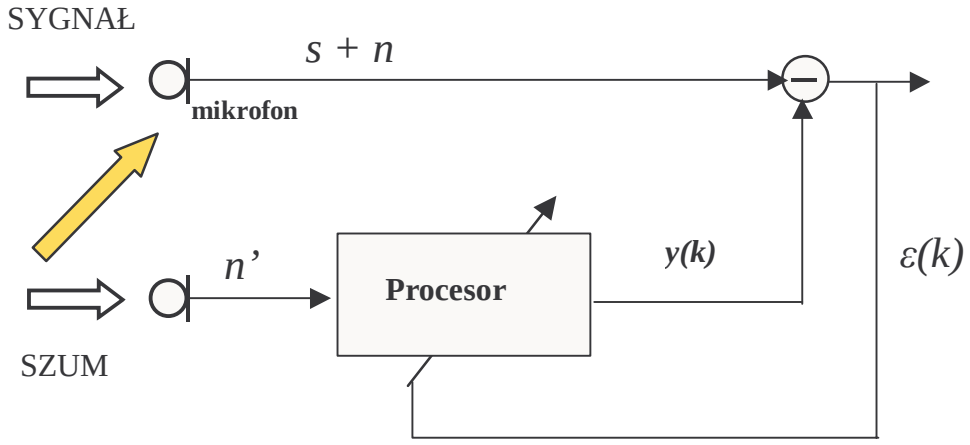
$$\hat{W} = R^{-1}P\,.$$

Rozwiązanie $\hat{W}$ może zostać osiągnięte w iteracjach metody najszybszego spadku, lub w jednym kroku z wykorzystaniem metody Newtona.



Rys. 4.12 Model filtru: liniowy sumator ważony z sygnałem zadanym d(k)

Przykładem technicznego zastosowania jest tłumienie tła w postaci szumów (rys. 4.13).



Rys. 4.13 Filtracja szumu, dostępny skorelowany szum n'.

Stosując dodatkowy mikrofon wychwytujący szum tła, można stłumić sygnał zakłócające sygnał użyteczny *s*. Warunkiem działania takiego filtru jest niezerowa macierz P, czyli istnienie korelacji pomiędzy szumem tła *n'*, a zakłócającym szumem *n*. Sygnał użyteczny *s* nie jest skorelowany z *n*. Wtedy:

$$\varepsilon = s + n - y\,,$$

$$J(\varepsilon^2) = E[\varepsilon^2] = E[\varepsilon^2 + (n-y)^2 + 2s(n-y)] = E[\varepsilon^2] + E[(n-y)^2],$$

(założony jest brak korelacji sygnału s i n oraz y).

A zatem minimalizacja wyrażenia $E[(n-y)^2]$ jest metodą adaptacji filtru, a sygnał wyjściowy filtru *y* jest estymatą nieznanego składnika *n* – szumu.