Metoda ekspansji i kontrakcji

Metody Optymalizacji

Wykonał Szymon Tokarz

Wstęp teoretyczny

Metody bezgradientowe wykorzystuje się w optymalizacji w przypadkach, gdy podany wzór funkcji jest zbyt skomplikowany, aby udało się obliczyć jego pochodną albo policzenie jej zajęłoby zbyt dużo czasu metodami analitycznymi. Jednymi z tych metod są metody ekspansji i kontrakcji.

Metoda ekspansji

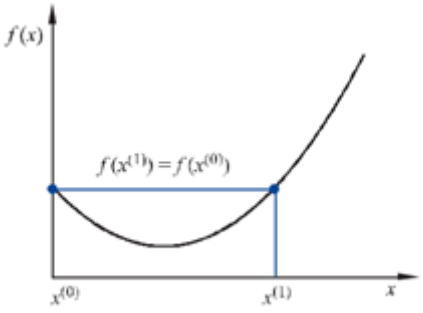
Pierwszym warunkiem, aby wykorzystać metodę ekspansji, jest to, że zadana funkcja musi być unimodalna. Gdyby tak nie było metoda może nie przynieść oczekiwanych rezultatów.

> **Definicja 3.1**
> Funkcja ciągła jednej zmiennej f jest *unimodalna*, gdy istnieje punkt c taki, że f jest rosnąca dla $x > c$ i malejąca dla $x < c$.

Metoda ekspansji polega na ciągłym zmniejszaniu przedziału [a,b] w jednym kierunku. Na początek należy wybrać punkt x(0)=0 oraz dowolny punkt x(1). W zależności od wartości funkcji celu można spotkać się z trzema przypadkami:

1. F(x(0)) = F(x(1))

Jeżeli wartości funkcji w obydwu punktach są takie same, to kończymy poszukiwania, a minimum znajduje się w przedziale [x(0),x(1)].

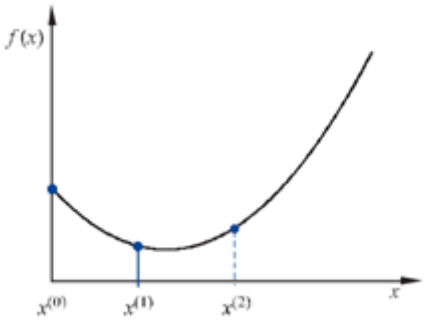


Rys.1 Wykres dla warunku 1.

2. F(x(0))<F(x(1))

Gdy zachodzi ten przypadek oznacza to, że kierunek poszukiwania jest dobry. Należy więc wykonać operację x(2)=α*x(1), gdzie alfa to współczynnik ekspansji mówiący nam o tym, jak

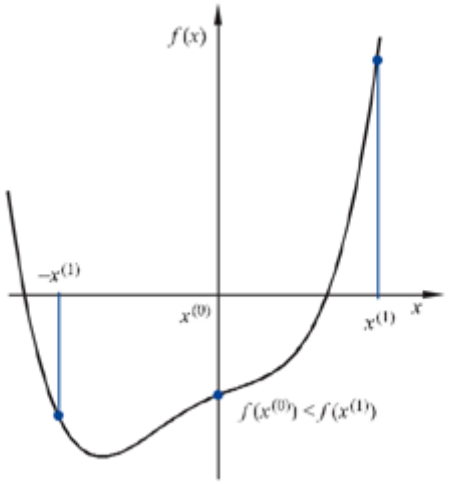
daleko szukamy nowego punktu. Alfa powinna być większa od 1, aby zapewnić przyrost długości kolejnych kroków. Jeśli F(x(2))>=F(x(1)) to kończymy proces i minimum znajduje się w przedziale [x(0),x(2)]. Jeśli jednak tak nie zajdzie to wykonuje się powyższą operację aż do skutku.



Rys.2 Wykres dla warunku 2

3. F(x(0))>F(x(1))

Jeżeli zachodzi ten warunek oznacza to, że następnego punktu należy szukać w lewo od punktu x(0). W tym wypadku należy wykonać podstawienie x(1) <- -x(1). Jeżeli F(x(1))>=F(x(0)) to kończymy poszukiwania i przedział to [x(1),x(0)]. Jeśli zaś nie to powtarzamy procedurę z powyższego punktu w analogiczny sposób. Jedyna różnica polega na tym, że poruszamy się w drugą stronę.



Rys.3 Wykres dla warunku 3

Istnieje rozwinięcie tej metody opracowane przez Boxa i Daviesa polegające na tym, że gdy posiadamy wstępną lokalizację minimum nie trzeba przyjmować punktu x(0)=0, tylko bardziej odpowiadający punkt do szybszych obliczeń.

Algorytm metody ekspansji

Dane wejściowe: punkt startowy $x^{(0)} = 0$, punkt $x^{(1)} > 0$, współczynnik ekspansji $\alpha > 1$, maksymalna liczba iteracji $N_{\max}$

$i = 0$ ▷ licznik iteracji
if $f(x^{(1)}) = f(x^{(0)})$ **then**
  return $[x^{(0)}, x^{(1)}]$ ▷ minimum występuje w przedziale $[x^{(0)}, x^{(1)}]$
end if
if $f(x^{(1)}) > f(x^{(0)})$ **then**
  $x^{(1)} = -x^{(1)}$
  if $f(x^{(1)}) \geq f(x^{(0)})$ **then**
    return $[x^{(1)}, -x^{(1)}]$ ▷ minimum występuje w przedziale $[x^{(1)}, -x^{(1)}]$
  end if
end if
repeat
  if $i > N_{\max}$ **then**
    return error ▷ nie udało się ustalić przedziału w $N_{\max}$ krokach
  end if
  $i = i + 1$
  $x^{(i+1)} = \alpha^i x^{(1)}$
until $f(x^{(i)}) \leq f(x^{(i-1)})$
if $x^{(i-1)} < x^{(i+1)}$ **then**
  return $[x^{(i-1)}, x^{(i+1)}]$ ▷ minimum występuje w przedziale $[x^{(i-1)}, x^{(i+1)}]$
end if
return $[x^{(i+1)}, x^{(i-1)}]$ ▷ minimum występuje w przedziale $[x^{(i+1)}, x^{(i-1)}]$

Rys.4 Pseudokod dla metod ekspansji

Metoda kontrakcji

Rozważamy minimalizację funkcji unimodalnej Q:[z(0),z(1)]->R. Na początku należy wyznaczyć:

- wartości Q(z(0)) i Q(z(1)),

- współczynnik kontrakcji β z przedziału (0,1) zazwyczaj równy 0,5,

- maksymalną liczbę iteracji N.

Zakładamy, że Q(z(0))<Q(z(1)). Wartości wskaźnika jakości wyznacza się za pomocą wzoru:

$$z_i = z_0 + \beta^{i-1}(z_1 - z_0)$$

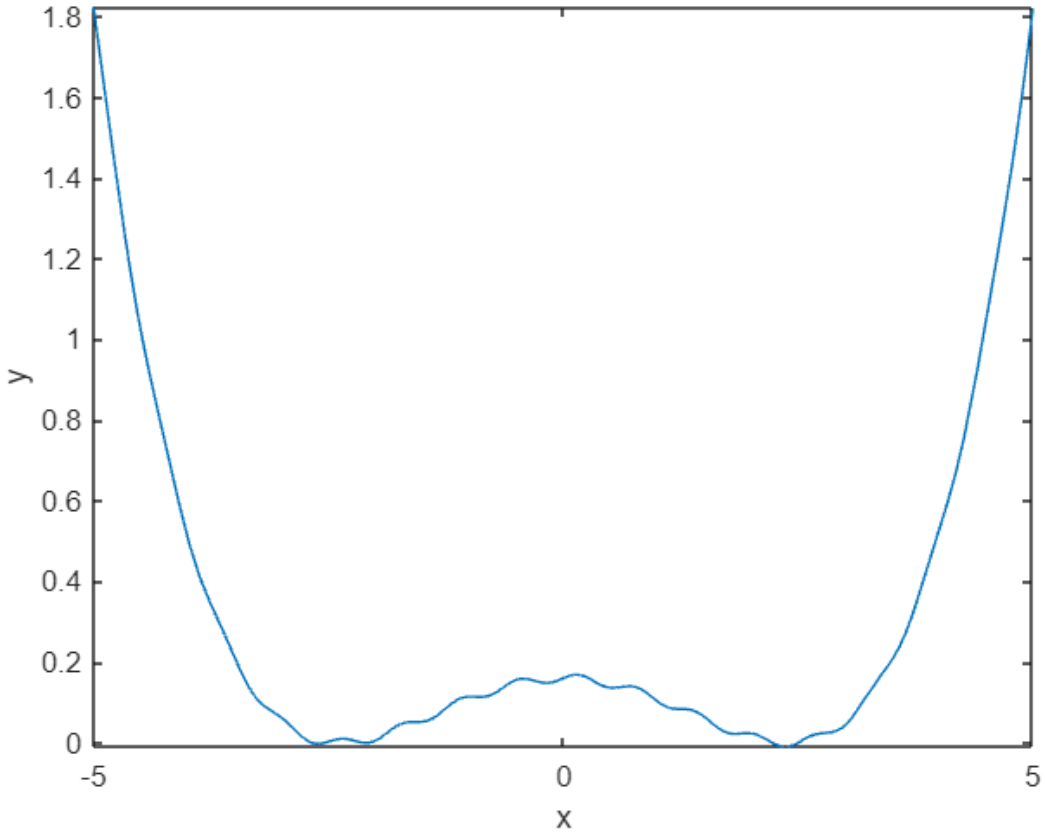
Gdzie i =2,3,4,… Algorytm przerywamy, gdy faktycznie Q(z(i))<Q(z(0)), jeśli krok pomyślny wystąpi przed przekroczeniem maksymalnej liczby iteracji. Wtedy przedziałem nieokreśloności jest po zakończeniu kontrakcji jest (z(0),z(i-1)), a najlepszym przybliżeniem minimum jest punkt (z(i),Q(z(i))). Jeśli nie uda się otrzymać kroku pomyślnego po N iteracjach, przerywamy algorytm bez polepszania aproksymacji minimum.

Kod w Matlab

Kod główny

```
f = @(x) 0.01*sin(10*x)+(0.07*x.^2-0.4).^2;


fplot(f);
xlabel('x');
ylabel('y');
```




```
x0 = -0.5;
x1 = x0 + 0.1;
alfa = 1.5;


[minimum, iteracje] = expanse(f, x0, x1, alfa, 100)
```

```
minimum = 1×2
   -0.4000    0.9000
iteracje = 1
```

```
alfa = 2;


[minimum, iteracje] = expanse(f, x0, x1, alfa, 100)
```

```
minimum = 1×2
   -0.4000    1.6000
iteracje = 1
```

```
x0 = 32;
x1 = x0 + 0.1;
```

```
alfa = 1.5;


[minimum, iteracje] = expanse(f, x0, x1, alfa, 100)
```
```
minimum = 1×2
  -32.1000   32.1000
iteracje = 0
```
```
alfa = 2;


[minimum, iteracje] = expanse(f, x0, x1, alfa, 100)
```
```
minimum = 1×2
  -32.1000   32.1000
iteracje = 0
```

4. Wyznaczyć przedział minimum funkcji metodą **ekspansji, następnie zawęzić przedział metodą kontrakcji** (uzupełnienie ekspansji)

5. Podać ilość kroków do osiągnięcia rozwiązań.

6. Przeanalizować warunek stopu, wnioski.

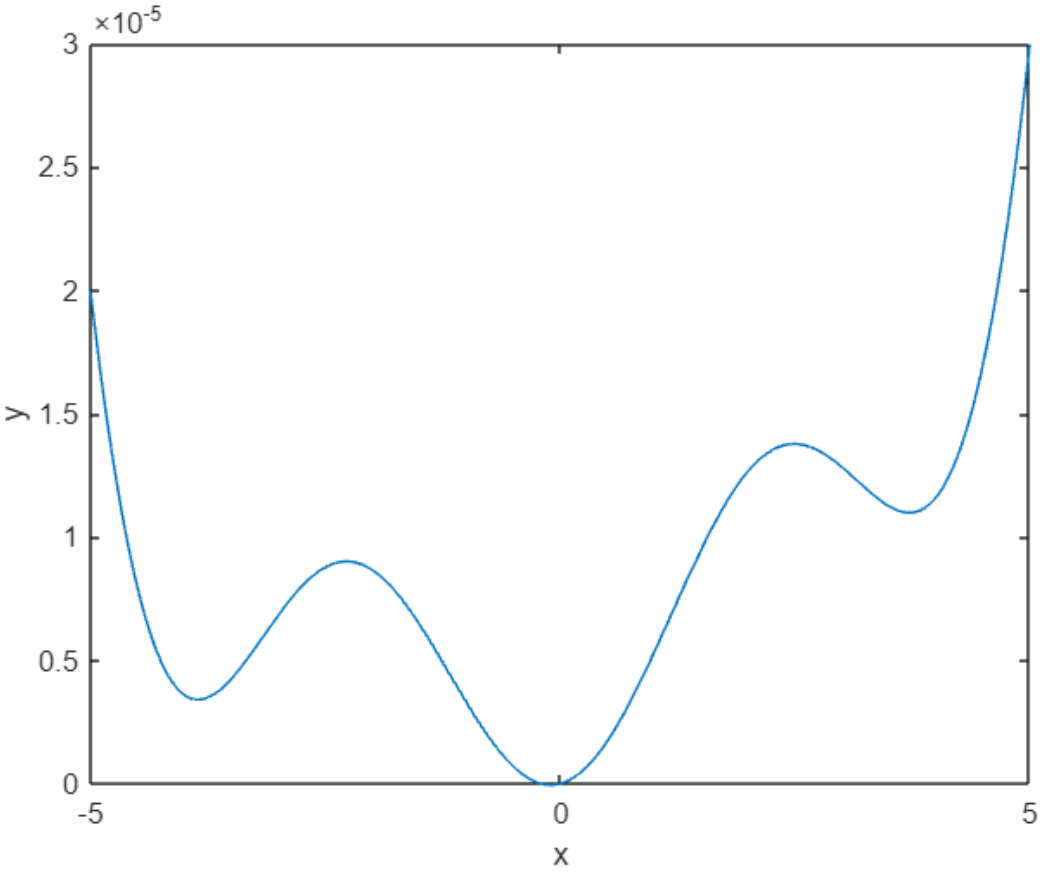
```
x0 = -0.5;
x1 = x0 + 0.1;
alfa = 2;
beta = 0.5;


[minimum, iteracje] = expanse(f, x0, x1, alfa, 100)
```
```
minimum = 1×2
   -0.4000    1.6000
iteracje = 1
```
```
[minimum_cont, iteracje] = contract(f, minimum(1), minimum(2), beta, 10000)
```
```
minimum_cont = "no_improv"
iteracje = 10000
```
```


x0 = 32;
x1 = x0 + 0.1;
[minimum, iteracje] = expanse(f, x0, x1, alfa, 100)
```
```
minimum = 1×2
  -32.1000   32.1000
iteracje = 0
```
```
[minimum_cont, iteracje] = contract(f, minimum(1), minimum(2), beta, 10000)
```
```
minimum_cont = 1×2
  -32.1000   32.1000
iteracje = 2
```

Druga funkcja

```
px = [-10 -5 0 5 10];
py = [0.0001 -0.1 0 0.1 -0.0001];
stopien=5;
W = polyfit(px, py, stopien);
```
```
Warning: Polynomial is not unique; degree >= number of data points.
```
```
fx=@(x)(0.05*sin(1*x)+W(1)*x.^stopien+W(2)*x.^(stopien-1)+W(3)*x.^(stopien-2)+W(4)*x.^(stopien-3)+W(5)*x.^(stopien-4)+W(6)*x.^(stopien-5))*(0.001*sin(x/10))+0.000001*x
```
```
fx = function_handle with value:
    @(x)(0.05*sin(1*x)+W(1)*x.^stopien+W(2)*x.^(stopien-1)+W(3)*x.^(stopien-2)+W(4)*x.^(stopien-3)+W(5)*x.^(stopien-4)+W(6)*x.^(stopien-5))*(0.001*sin(x/10))+0.000001*x
```
```
fplot(fx);
```
```
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly vectorize your function to return an output with the same size and shape as the input arguments.
```
```
xlabel('x');
ylabel('y');
```




```
x0 = -0.5;
x1 = x0 + 0.1;
```

```
alfa = 1.5;


[minimum, iteracje] = expanse(fx, x0, x1, alfa, 100)
```
```
minimum = 1×2
   -3.0375   -1.3500
iteracje = 4
```

```
alfa = 2;


[minimum, iteracje] = expanse(fx, x0, x1, alfa, 100)
```
```
minimum = 1×2
   -3.2000         0
iteracje = 2
```

```
x0 = 32;
x1 = x0 + 0.1;
alfa = 2;


[minimum, iteracje] = expanse(fx, x0, x1, alfa, 100)
```
```
minimum = 1×2
  -32.1000   32.1000
iteracje = 0
```

```
alfa = 2;


[minimum, iteracje] = expanse(fx, x0, x1, alfa, 100)
```
```
minimum = 1×2
  -32.1000   32.1000
iteracje = 0
```

Punkty 4,5,6

```
x0 = -0.5;
x1 = x0 + 0.1;
alfa = 1.5;
beta = 0.5;


[minimum, iteracje] = expanse(fx, x0, x1, alfa, 100)
```
```
minimum = 1×2
   -3.0375   -1.3500
iteracje = 4
```

```
[minimum_cont, iteracje] = contract(fx, minimum(1), minimum(2), beta, 10000)
```

```
minimum_cont = 1×2
         0   -0.4000
iteracje = 5
```

```
x0 = 32;
x1 = x0 + 0.1;
[minimum, iteracje] = expanse(fx, x0, x1, alfa, 100)
```

```
minimum = 1×2
  -32.1000   32.1000
iteracje = 0
```

```
[minimum_cont, iteracje] = contract(fx, minimum(1), minimum(2), beta, 10000)
```

```
minimum_cont = 1×2
  -32.1000   32.1000
iteracje = 2
```

```
minimum=[-28.4,-24.4];
beta=0.9;


[minimum_cont, iteracje] = contract(fx, minimum(1), minimum(2), beta, 10000)
```

```
minimum_cont = 1×2
  -28.4000  -24.4000
iteracje = 2
```

Funkcja expanse

```
function [min, i] = expanse(f, x0, x1, alfa, Nmax)
    x = zeros(Nmax+1);
    x(1) = x1;
    i = 0;

    if f(x0) == f(x1)
        min = [x0 x1];
        return
    end

    if f(x1) > f(x0)
        x1 = -x1;
        if f(x1) >= f(x0)
            min = [x1, -x1];
            return
        end
    end

    ok = 1;
    i = i + 1;
    while ok
        if i > Nmax + 1
            error("Nie udało się ustalić przedziału Nmax krokach");
            return;
        end
        i = i + 1;
```

```
        x(i + 1) = alfa^i*x1;
        ok = f(x(i)) <= f(x(i + 1));
    end

    if x(i - 1) < x(i + 1)
        min = [x(i - 1) x(i + 1)];
    else
        min = [x(i + 1) x(i - 1)];
    end
    i = i - 1;
end

Funkcja contract
function [min, it] = contract(f, x0, x1, beta, N)

    x = zeros(N);
    x(1) = x1;
    it = 0;

    if f(x0) > f(x1)
        temp = x0;
        x0 = x1;
        x1 = temp;
    end
    for it = 2:N
        x(it) = x0 + (beta^(it - 1))*(x1 - x0);

        if f(x(it)) < f(x0)
            min = [x0 x(it - 1)];
            return
        end
    end
    if it == N
        min = "no_improv";
    end
end
```

Wnioski

Metody ekspansji i kontrakcji są przydatnymi narzedziami pomocnymi w wyznaczeniu przybliżonych wartości funkcji. Ich wielką zaletą jest relatywnie mała liczba iteracji potrzebnych, aby otrzymać zadawalający wynik (maksymalnie 5). Można ich również używać bez potrzeby liczenia pochodnych lub gradientów, co może być problematyczne w bardziej skomplikowanych przypadkach. Przy korzystaniu z metody ekspansji i kontrakcji istotne jest uwzględnienie skali i kontekstu badanego problemu. Rozmiar obszaru ekspansji lub kontrakcji oraz okres analizy mają istotny wpływ na wyniki i interpretację. Dlatego ważne jest dostosowanie parametrów analizy do konkretnej sytuacji badawczej.