Obraz zawierający tekst, Czcionka, logo, Grafika

Opis wygenerowany automatycznie

Laboratorium

Metody Optymalizacji

Temat ćwiczenia:

**„Ćwiczenie 5 – Metody bezpośrednie optymalizacji dynamicznej bez ograniczeń”**

Autorzy:

Wojciech Buchelt  
Filip Fidokowicz

Kierunek: Automatyka i Robotyka, spec. ROB

Semestr: 1

1. Cel ćwiczenia

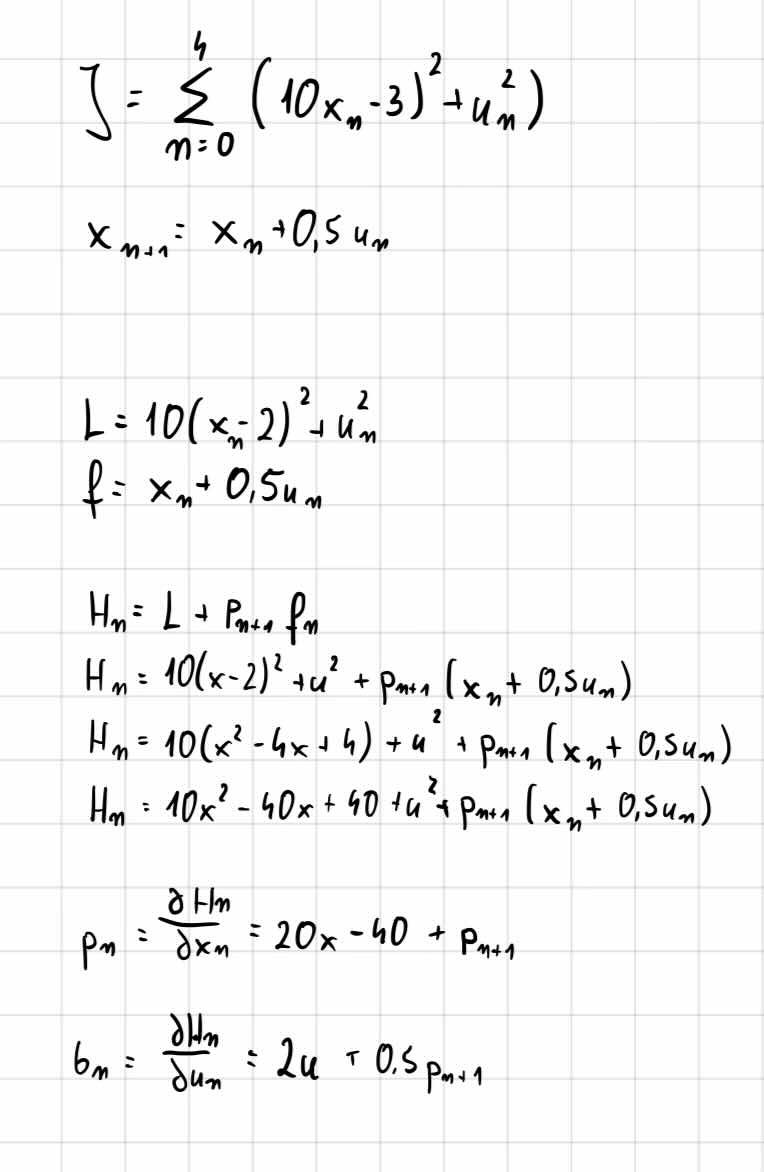
Celem ćwiczenia jest zapoznanie się z metodami bezpośredniej optymalizacji dynamicznej dla układów bez ograniczeń (z wyjątkiem równania stanu), ze szczególnym uwzględnieniem dwóch metod: metody gradientu prostego w przestrzeni sterowań (gradient prosty) i metody kierunków sprzężonych (gradient sprzężony).

1. Zadany problem

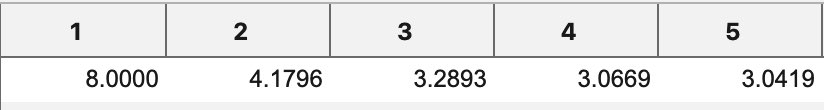
Obraz zawierający tekst, papier, pismo odręczne, list

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

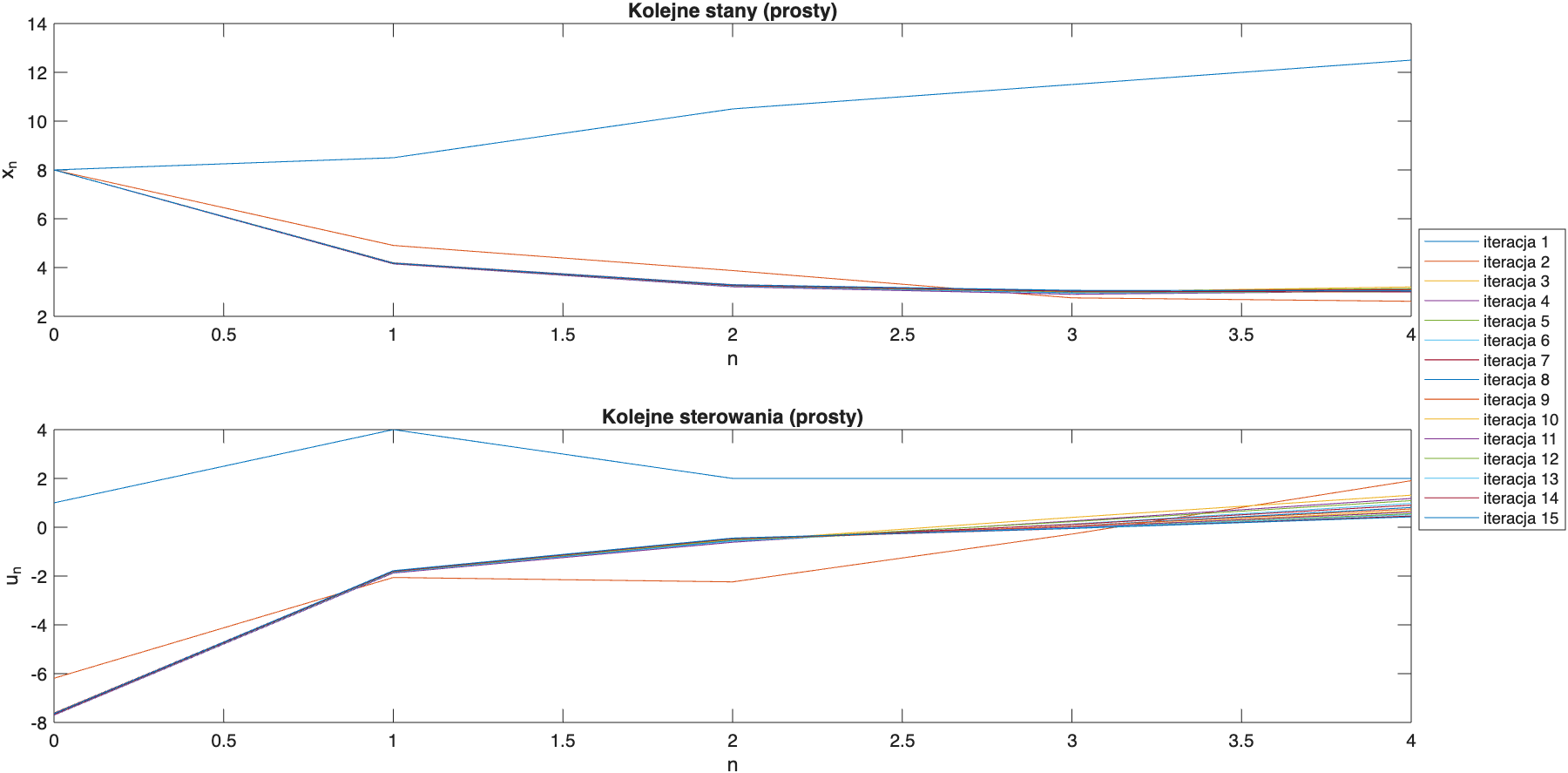
1. Analityczne wyznaczenie WK dla zadanego problemu

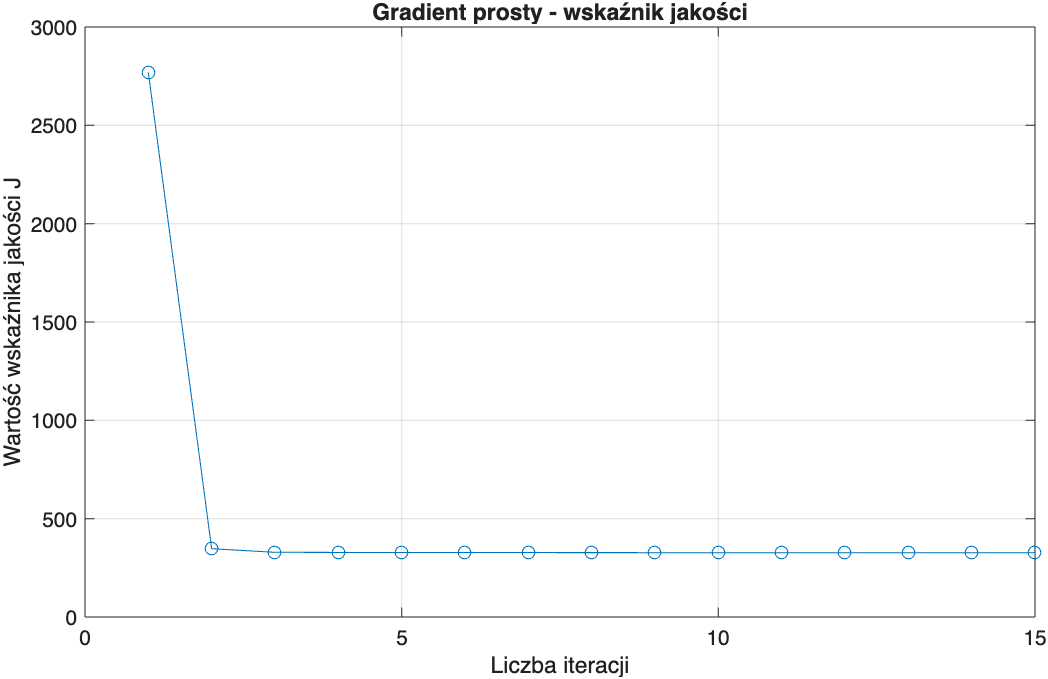


1. Obliczone wartości stanu x, u oraz wskaźnika jakości dla metody gradientu prostego
2. Kolejne wartości stanu x (gradient prosty)

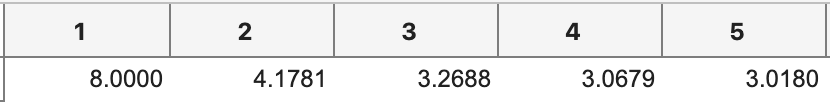


1. Kolejne wartości u (gradient prosty)
2. Kolejne wartości wskaźnika jakości (gradient prosty)
3. Otrzymane wykresy dla metody gradientu prostego





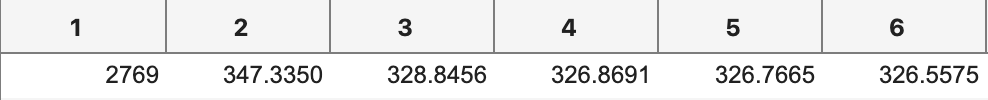
1. Obliczone wartości stanu x, u oraz wskaźnika jakości dla metody gradientu sprzężonego
2. Kolejne wartości stanu x (gradient sprzężony)



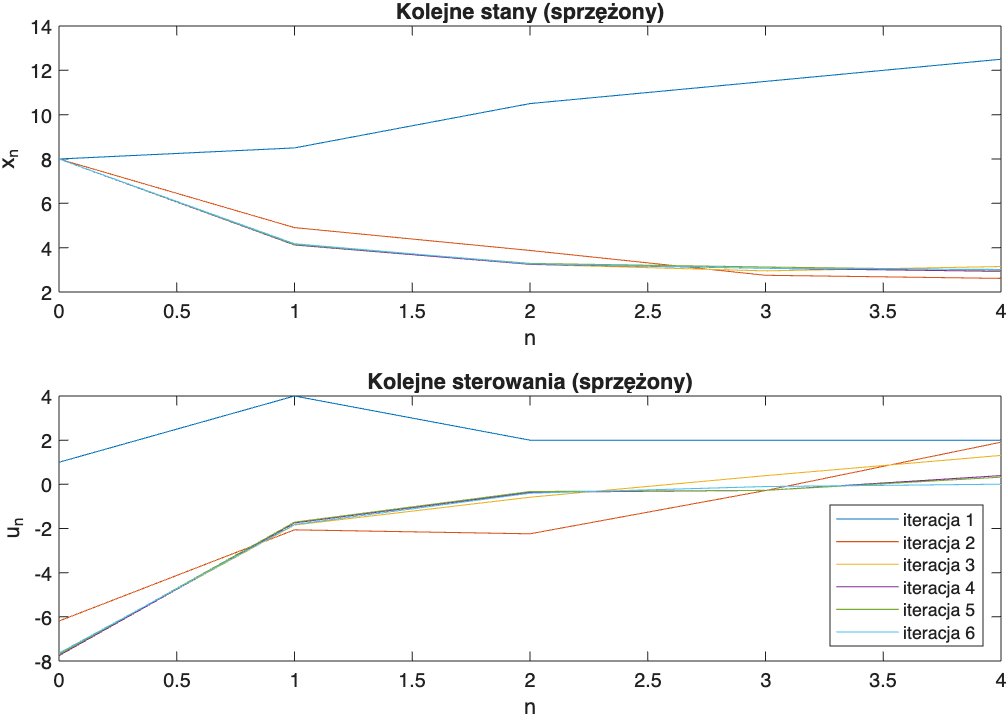
1. Kolejne wartości u (gradient sprzężony)

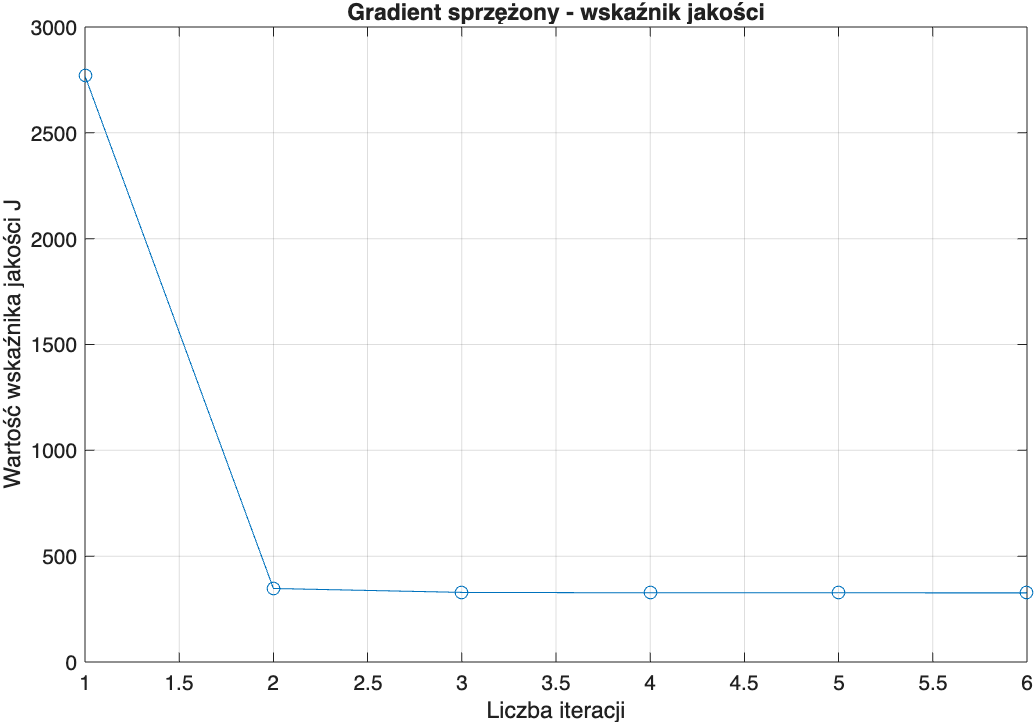


1. Kolejne wartości wskaźnika jakości (gradient sprzężony)

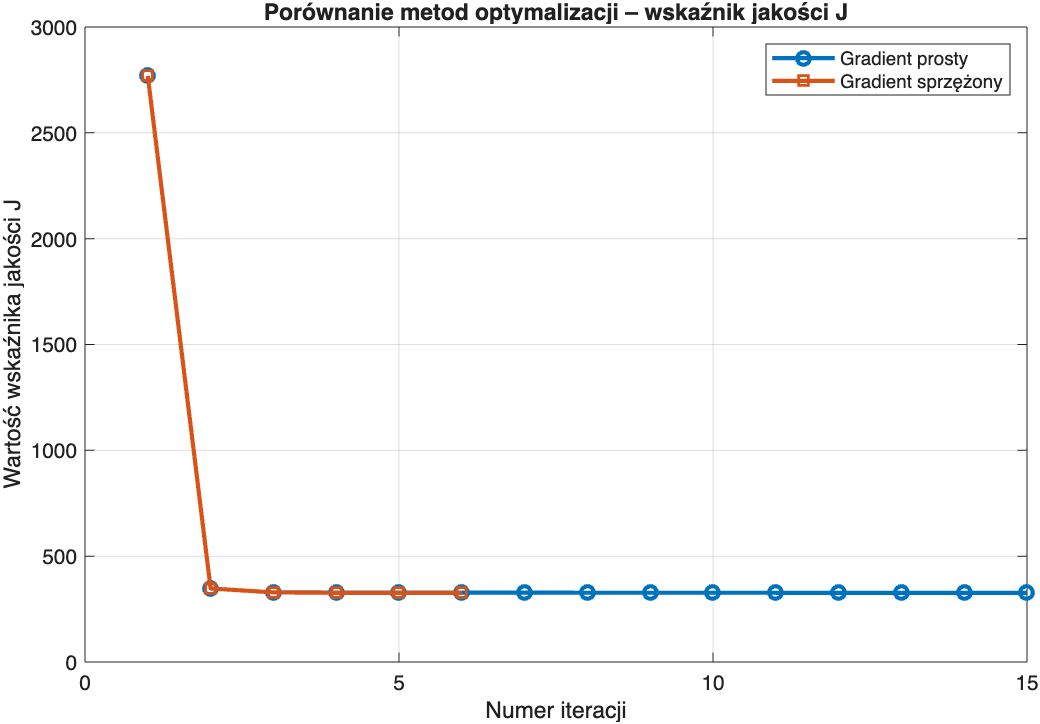


1. Otrzymane wykresy dla metody gradientu sprzężonego





1. Porównanie metod optymalizacji



1. Wnioski:

W zadaniu zastosowano dwie metody: gradient prosty i gradient sprzężony.

Gradient prosty kieruje się wyłącznie bieżącym kierunkiem największego spadku funkcji celu.

W przedstawionym przykładzie metoda ta wykonała 15 iteracji (maksymalną dozwoloną liczbę), ponieważ nie osiągnął dokładności mniejszej niż 𝜀=0,2. Z kolei gradient sprzężony bierze pod uwagę również poprzednie kierunki, dzięki czemu porusza się szybciej w stronę minimum. W tym samym przykładzie zakończył działanie już po 6 iteracjach, ponieważ osiągnął wartość gradientu mniejszą niż założony próg dokładności.

Ostateczne wyniki w obu metodach są dość podobne, jednak gradient sprzężony dotarł do nich znacznie szybciej, co potwierdza jego większą efektywność niż metoda gradientu prostego.

1. Kod:

clear all;

close all;

clc;

K = 15;

t = 0.001;

e = 0.2;

x(1) = 8;

u = [1 4 2 2 2];

N = length(u);

c = 0;

r\_stanu = @(x, u) x + 0.5\*u;

L = @(x, u) (10\*(x-3).^2 + u.^2);

J\_fun = @(x, u) sum(L(x, u));

p(N+1) = 0;

syms xs us ps

Hn(xs, us, ps) = L(xs, us) + ps \* r\_stanu(xs, us);

pn\_fun = matlabFunction(diff(Hn, xs));

bn\_fun = matlabFunction(diff(Hn, us));

legenda = {};

k = 0;

while 1

k = k + 1;

for i = 2:N

x(i) = r\_stanu(x(i-1), u(i-1));

end

legenda{end+1} = ['iteracja ', num2str(k)];

subplot(2,1,1);

plot(0:N-1, x); hold on; title('Kolejne stany (prosty)'); xlabel('n'); ylabel('x\_n');

subplot(2,1,2);

plot(0:N-1, u); hold on; title('Kolejne sterowania (prosty)'); xlabel('n'); ylabel('u\_n');

legend(legenda);

J\_plot\_p(k) = J\_fun(x, u);

for i = N:-1:1

p(i) = pn\_fun(x(i), u(i), p(i+1));

b(i) = bn\_fun(x(i), u(i), p(i+1));

end

t = fminsearch(@(t) wyznacz\_t(r\_stanu, J\_fun, x, u, N, b, t), t);

u = u - t \* b;

if norm(b) < e || k >= K

break

end

end

J\_plot\_p

figure;

plot(1:k, J\_plot\_p, '-o');

title('Gradient prosty - wskaźnik jakości');

xlabel('Liczba iteracji');

ylabel('Wartość wskaźnika jakości J');

grid on;

% Funkcja do minimalizacji po t (gradient prosty)

function J\_val = wyznacz\_t(r\_stanu, J\_fun, x, u, N, b, t)

x\_new = x;

for i = 2:N

x\_new(i) = r\_stanu(x\_new(i-1), u(i-1) - t\*b(i-1));

end

u\_new = u - t\*b;

J\_val = J\_fun(x\_new, u\_new);

end

% Gradient sprzężony

u = [1 4 2 2 2];

k = 0;

b\_old = zeros(1,N);

d = zeros(1,N);

legenda = {};

while 1

k = k + 1;

for i = 2:N

x(i) = r\_stanu(x(i-1), u(i-1));

end

legenda{end+1} = ['iteracja ', num2str(k)];

subplot(2,1,1);

plot(0:N-1, x); hold on; title('Kolejne stany (sprzężony)'); xlabel('n'); ylabel('x\_n');

subplot(2,1,2);

plot(0:N-1, u); hold on; title('Kolejne sterowania (sprzężony)'); xlabel('n'); ylabel('u\_n');

legend(legenda);

J\_plot\_sprz(k) = J\_fun(x, u);

for i = N:-1:1

p(i) = pn\_fun(x(i), u(i), p(i+1));

b(i) = bn\_fun(x(i), u(i), p(i+1));

end

if k == 1

d = -b;

else

c\_FR = (norm(b)^2) / (norm(b\_old)^2);

d = -b + c\_FR \* d;

end

b\_old = b;

t = fminsearch(@(t) wyznacz\_t\_sprz(r\_stanu, J\_fun, x, u, N, d, t), t);

u = u + t \* d;

if norm(b) < e || k >= K

break

end

end

J\_plot\_sprz

figure;

plot(1:k, J\_plot\_sprz, '-o');

title('Gradient sprzężony - wskaźnik jakości');

xlabel('Liczba iteracji');

ylabel('Wartość wskaźnika jakości J');

grid on;

figure;

plot(1:length(J\_plot\_p), J\_plot\_p, '-o', 'LineWidth', 2, 'DisplayName', 'Gradient prosty'); hold on;

plot(1:length(J\_plot\_sprz), J\_plot\_sprz, '-s', 'LineWidth', 2, 'DisplayName', 'Gradient sprzężony');

title('Porównanie metod optymalizacji – wskaźnik jakości J');

xlabel('Numer iteracji');

ylabel('Wartość wskaźnika jakości J');

legend('Location','northeast');

grid on;

% Funkcja do minimalizacji po t (gradient sprzężony)

function J\_val = wyznacz\_t\_sprz(r\_stanu, J\_fun, x, u, N, d, t)

x\_new = x;

for i = 2:N

x\_new(i) = r\_stanu(x\_new(i-1), u(i-1) + t\*d(i-1));

end

u\_new = u + t\*d;

J\_val = J\_fun(x\_new, u\_new);

end