Optimizavimo metodai

2lab.

Optimizavimas be apribojimų

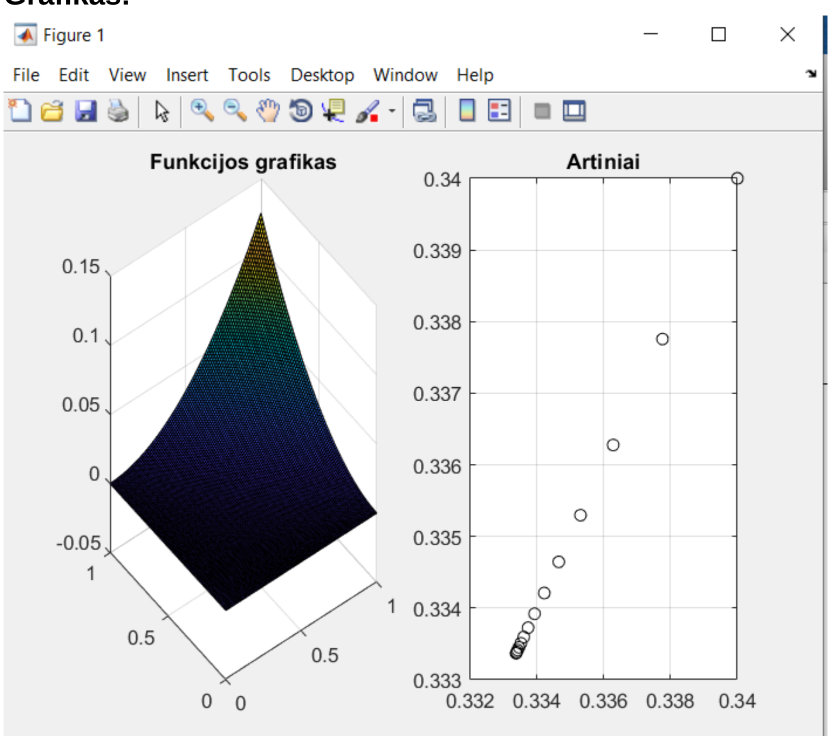
Rafal Michalkevič

Gradientinio nusileidimo metodas

```
function Gradientinio_nus

%duota funkcija
f = @(x1, x2) (1 / 8) * ((x1 .^ 2) .* x2 + x1 .* (x2 .^ 2)
- x1 .* x2);

%gradientine funkcija
gradient = @(X) [2 * X(1) * X(2) + X(2) .^ 2 - X(2), X(1) ^
2 + 2 * X(1) * X(2) - X(1)];

%pradiniai taskai
X_0 = [0, 0];
X_1 = [1, 1];
X_my = [9/10, 5/ 10];


%braizymas
subplot(1, 2, 1);
[x1, x2] = meshgrid(0:0.01:1, 0:0.01:1);
y = f(x1, x2);
surf(x1, x2, y);
title(['Funkcijos grafikas']);

%pasirenku gamma
gamma = 0.33;

epsilon = 10 ^ (- 4);

k = 1;
kmax = 100;

format long;

%pasirenkame kad praeitu nors karta per cikla
dist = Inf;

X0 = X_1;
while dist >= epsilon

    grad = gradient(X0);
    X0 = X0 - gamma .* grad;
    dist = norm(grad);
```

```
    fprintf('X0= %f %f f(x0)= %f k= %d\n', X0, f(X0(1),
X0(2)), k);

    subplot(1, 2, 2);
    title(['Artiniai']);
    plot(X0(1), X0(2), 'blao');
    hold on;

    if k == kmax
        format short;
        disp(['Pasiektas maksimalus iteraciju skaicius k=',
num2str(kmax)]);
        break
    end
    k = k + 1;

end
grid on;
hold off;
```

Grafikas:



Lentelė

x	Įvertis	Žingių skaičius	Funkcijų skaičiavimo skaičius
(0,0)	(0, 0)	1	1
(1,1)	(0.333369, 0.333369)	14	14
(0.9,0.5)	(0.333517,0.333149)	66	66

Greičiausio nusileidimo metodas

```
function Greiciausio_nus

%duota funkcija
f=@(X) (1 / 8) * (X(1)^2.*X(2)+X(1)*X(2)^2-X(1)*X(2));

%gradientine funkcija
gradient = @(X) [2 * X(1) * X(2) + X(2) .^ 2 - X(2), X(1) ^
2 + 2 * X(1) * X(2) - X(1)];

% pradiniai taskai
X_0 = [0, 0];
X_1 = [1, 1];
X_my = [9/10, 5/10];
%X_my2 = [5/10, 4/10];
X0 = X_my;

epsilon = 10 ^ (- 4);


i=0;
k=1;
kmax=100;


format short;

%priskiriame kad praeitu nors karta cikla
dist = Inf;
```

```
while dist>=epsilon
      grad=gradient(X0);
      res=Auksinio_pjuvio(f,X0,grad);
      gamma=res(1);
   %   fprintf("%f", gamma);
      i=i+res(2)+1;
      X0 = X0 - gamma .* grad; % naujas artinys
      dist = norm(grad);


      fprintf('X0= %f %f f(X0)= %f k= %d i= %d\n', X0,
f(X0), k, i);


      if k==kmax
          disp(['Pasiektas maksimalus iteraciju skaicius
k=', num2str(kmax)]);
          break
      end
      k=k+1;
end
end
```

Lentelė

x	Įvertis	Žingių skaičius	Funkcijų skaičiavimo skaičius
(0,0)	(0, 0)	1	22
(1,1)	(0.333365, 0.333365)	2	44
(0.9,0.5)	(0.333492,0.333175)	43	946

Simplekso metodas:

```
function Simplexo

f = @(X) (1 / 8) * ((X(1) .^ 2) .* X(2) + X(1) .* (X(2) .^
2) - X(1) .* X(2));
```

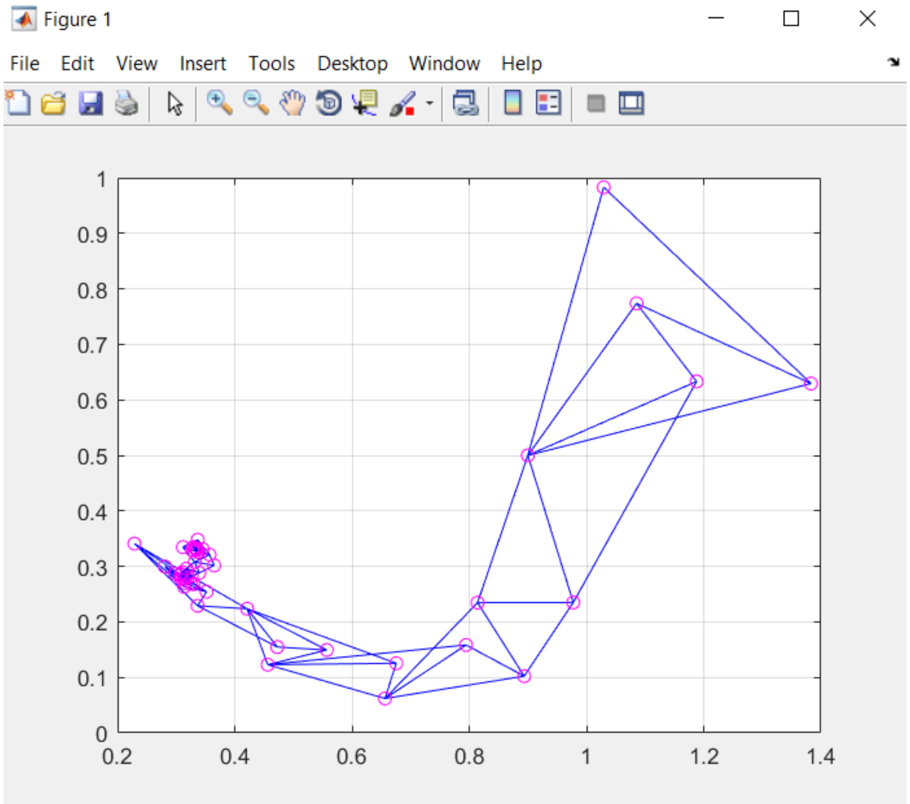
```
X_0 = [0, 0];
X_1 = [1, 1];
X_m = [9 / 10, 5 / 10];

X0 = X_m;

alpha = 1/2;
teta = 1;

gamma = 1.6;
beta = 0.5;
eta = - 0.5;

epsilon = 10 ^ (- 4);

n = 2;
delta1 = alpha * (sqrt(n + 1) + n - 1) / (n * sqrt(2));
delta2 = alpha * (sqrt(n + 1) - 1) / (n * sqrt(2));

X1 = [X0(1, 1) + delta2, X0(1, 2) + delta1];
X2 = [X0(1, 1) + delta1, X0(1, 2) + delta2];

y0 = f(X0);
y1 = f(X1);
y2 = f(X2);

X = [X0; X1; X2];

y = [y0, y1, y2];

deltax = [X0(1), X0(1), X1(1); X1(1), X2(1), X2(1)];
deltay = [X0(2), X0(2), X1(2); X1(2), X2(2), X2(2)];
plot(deltax, deltay, 'b');
grid on;
hold on;
plot(X(:, 1), X(:, 2), 'mo');
hold on;

k = 1;
i = 3;
kmax = 100;
imax = 100;

format short;
```

```
disp(['   x1    x2     f(x1,x2)     k     (f kv. sk.)']);

pasiekta = false;
%pradedame vykdyti iteracijas
while ~ pasiekta
    [~, nr] = sort(y);

    %didziausias y
    yhigh = y(nr(3));
    Xhigh = X(nr(3), :);

    %maziausias y
    ylow = y(nr(1));
    Xlow = X(nr(1), :);

    %vidurinis y
    yg = y(nr(2));
    Xg = X(nr(2), :);

    Xc = (Xg + Xlow) / 2;

    X_naujas = Xhigh + (1 + teta) * (Xc - Xhigh);
    y_naujas= f(X_naujas);
    i = i + 1;

    if X_naujas(1) <= 0 || X_naujas(2) <= 0
        teta = - 1 / 2;
        X_naujas = Xhigh + (1 + teta) * (Xc - Xhigh);
        y_naujas = f(X_naujas);
        i = i + 1;
    end

    if (ylow < y_naujas) && (y_naujas < yg)
        teta = 1;
    elseif y_naujas < ylow
        teta = gamma;
        Z = Xhigh + (1 + teta) * (Xc - Xhigh);
        yz = f(Z);
        i = i + 1;
        if yz < y_naujas
            y_naujas = yz;
            X_naujas = Z;
        end
```

```
    elseif y_naujas > yhigh
        teta = eta;
        Z = Xhigh + (1 + teta) * (Xc - Xhigh);
        X_naujas = Z;
        y_naujas = f(Z);
        i = i + 1;
    elseif (yg < y_naujas) && (y_naujas < yhigh)
        teta = beta;
        Z = Xhigh + (1 + teta) * (Xc - Xhigh);
        X_naujas = Z;
        y_naujas = f(Z);
        i = i + 1;
    end

    if X_naujas(1) <= 0 || X_naujas(2) <= 0
        teta = - 1 / 2;
        X_naujas = Xhigh + (1 + teta) * (Xc - Xhigh);
        y_naujas = f(X_naujas);
        i = i + 1;
    end

     fprintf('%f    %f   %f %d   %d', X_naujas, y_naujas,
k, i);

    count = 0;

    if max([norm(Xlow - Xg), norm(Xlow - Xhigh), norm(Xg -
Xhigh)]) < epsilon
        count = count + 1;
    end

    if max([abs(ylow - yg), abs(ylow - yhigh), abs(yg -
yhigh)]) < epsilon
        if ~count
          disp(' ')
        end
        count = count + 1;
    end

    if i >= imax
        count = count + 1;
        disp(['Pasiektas maksimalus funkciju kvietimu
skaicius i=', num2str(imax)]);
        if count == 3
```

```
            pasiekta = true;
        end
    end

    if k == kmax
        format short;
        disp(['Pasiektas maksimalus iteraciju skaicius k=',
num2str(kmax)]);
        break
    end

    k = k + 1;
    X = [Xlow; Xg; X_naujas];

    y = [ylow, yg, y_naujas];


    deltax = [Xlow(1), Xlow(1), Xg(1); Xg(1), X_naujas(1),
X_naujas(1)];
    deltay = [Xlow(2), Xlow(2), Xg(2); Xg(2), X_naujas(2),
X_naujas(2)];
    plot(deltax, deltay, 'b');
    hold on;
    plot(X(:, 1), X(:, 2), 'mo');
    hold on;

    if ~ count
        disp(' ');
    end
end
end
```

Grafikas:



Lentelė:

x	Įvertis	Žingių skaičius	Funkcijų skaičiavimo skaičius
(0,0)	(0.333333, 0.333333)	52	100
(1,1)	(0.333333, 0.333333)	55	100
(0.9,0.5)	(0.333262, 0.333337)	60	111

Apibendrinimas

- Pasirinkdami tašką X0 = [0,0] taikant Gradientinio bei Greičiausio nusileidimo algoritmus, nekonverguojama į minimumą. Tai įvyksta dėl to, kad gradientas tokio pradinio taško atvėjų yra lygus 0. Simplekso algoritmas su tokia problema nesusidūria, bei į minimum tašką konverguoja per 52 iteracijas tuo pačiu iškvietęs funkciją 100 kartų

- Pasirinkdami tašką X1 = [1,1] taikant Gradientinio nusileidimo algoritmą minimumui rasti reikia 14 iteracijų, o Greičiausio nusileidimo algoritmą užtenka tik 2 iteracijų – tiesą jų metu tikslo funkcija yra skaičiuojama net 44 kartus. Simplekso algoritmas tuo tarpų su ta pačią užduotimi susitvarko per 55 iteracijas iškviesdamas funkciją 100 kartų

- Pasirinkdami tašką Xm = [0.9,0.5] mažiausiai iteracijų atlieka Greičiausio nusileidimo metodas – 43 iteracijos, tačiau verta pažymėti, jog jo vykdymo metu labai daug kartų teko skaičiuoti tikslo funkcijos reikšmę – dažniausiai Auksinio pjūvio pritaikymo metu. Palyginimui – Simplekso ir Gradientinio nusileidimo algoritmai atliko atitinkamai 60 ir 66 iteracijas, bet Gradientinio nusileidimo algoritmui prireikė mažesnio funkcijų kvietimo skaičiaus.