Optimizavimo metodai

Laboratorinis darbas nr. 2 – Optimizavimas be apribojimų

(Parengė Žygimantas Rimgaila, Informatika, III kursas, IVgr.)

Gradientinis metodas

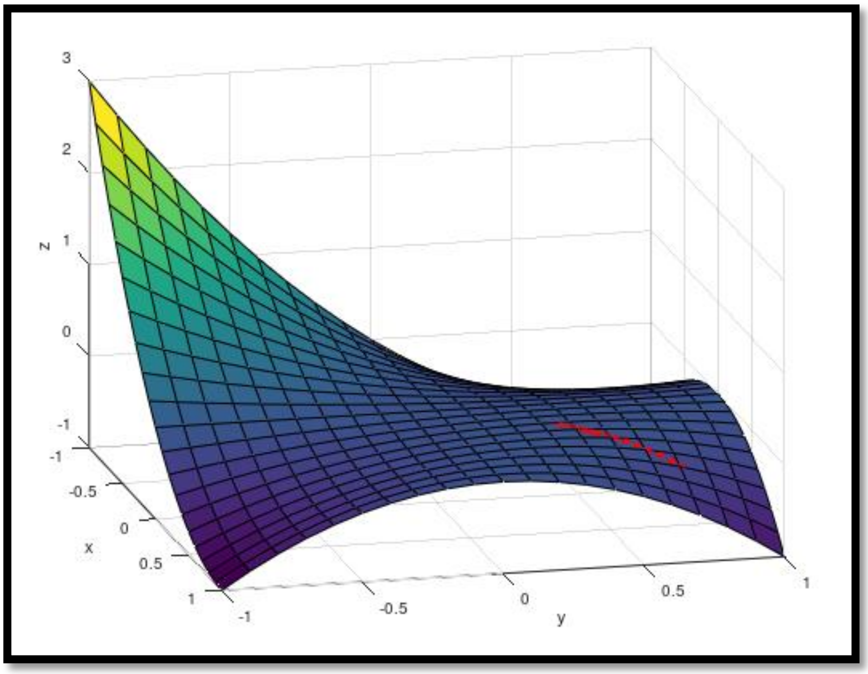
```
a = 7;
b = 1;
x = a/10;
y = b/10;
eps = 0.0001;
i = 0;
xi = [x,y];
gama = 0.9;
f = @(x,y) (-(1/8).*x.*y.*(1-x-y));
derx = @(x,y) (2*x*y+y^2-y)/8; % isvestine pagal x
dery = @(x,y) (2*x*y+x^2-x)/8; % isvestine pagal y
% Paruošiu 3d piešimui
X = -1:0.1:1;
Y = -1:0.1:1;
[X, Y] = meshgrid(X,Y);
Z = f(X,Y);
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('z');
hold on
while true
    plot3(xi(1), xi(1), f(xi(1),xi(1)), '*r');
    i = i + 1;
    xi1 = xi - gama * gradf(xi);
    gr_norma = gradf(xi1);
    if sqrt(gr_norma(1)^2 + gr_norma(2)^2) < eps
        xi = xi1;
        break;
    end
    xi = xi1;
end
hold off
i
```

X_min = (0.33503;   0.33165)

i = 43

V = 0.068041

krastine = 0.40826

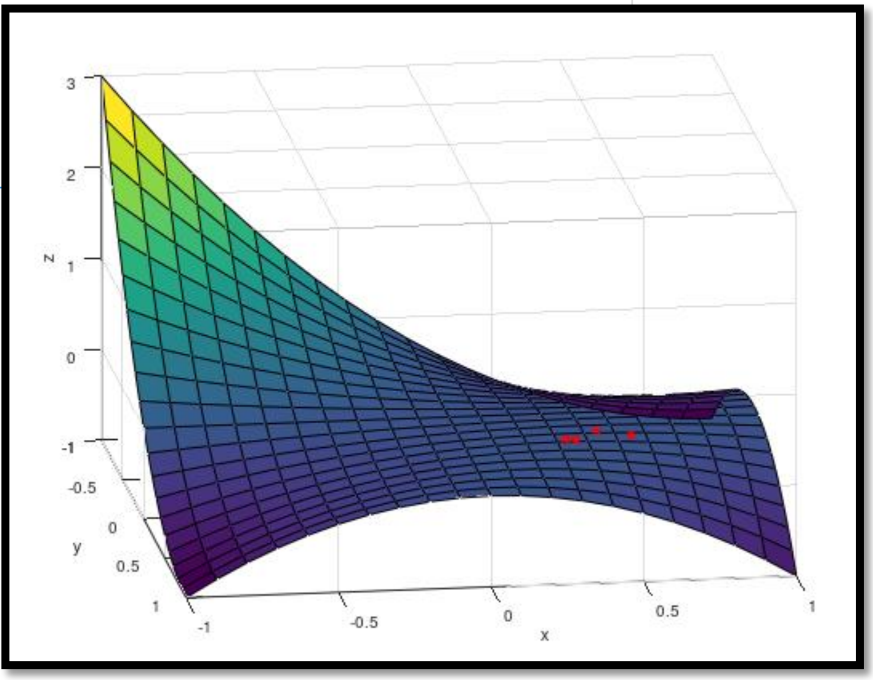


Greičiausio nusileidimo metodas

```matlab
function Greiciausias
a = 7;
b = 1;
eps = 0.0001; %tikslumas
k=1; %iteraciju skaitliukas
i=0; %funkciju kvietimu skaicius
k_max=100; % maksimalus iteraciju skaitliukas
i_max=100;
norma = Inf;
X0 = [a/10, b/10];
f = @(x1, x2) (x1.^2).*x2 + x1.*(x2.^2) - x1.*x2;
gradf = @(X) [2.*X(1,1).*X(1,2) + X(1,2).^2 - X(1,2), X(1,1).^2 + 2.*X(1,1).*X(1,2)-X(1,1)];
X = -1:0.1:1;
Y = -1:0.1:1;
[X, Y] = meshgrid(X,Y);
Z = f(X,Y);
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('z');
hold on
disp('    x1           x2          f(x1,x2)       k           i');
while norma >= eps
      grad = gradf(X0);
      norma = norm(grad);
      ats = AuksinisPj(X0, grad);
      gama = ats(1);
      i_sk = ats(2);
      i = i + 1 + i_sk;
      X1 = X0 - gama*grad;
      disp([X1, f(X1(1,1), X1(1,2)), k, i]);
      plot3(X1(1,1), X1(1,2),  '*r');
      if k == k_max
        disp(['Itercijų skaičius maksimalus, k = ', num2str(k_max)]);
        break
      end
      X0 = X1;
      k = k+1;
end
grid on
hold off
end
```

k = 5 (iteracijos)

i = 130 (tikslo funkcijos skačiavimų)



x1	x2	f(x1,x2)	k	i
0.554945	0.303076	0.023879	1.000000	26.000000
0.450642	0.228574	0.033042	2.000000	52.000000
0.369582	0.342055	0.036454	3.000000	78.000000
0.342818	0.322938	0.037004	4.000000	104.000000
0.334999	0.333882	0.037036	5.000000	130.000000

Auksinio pjūvio algoritmas

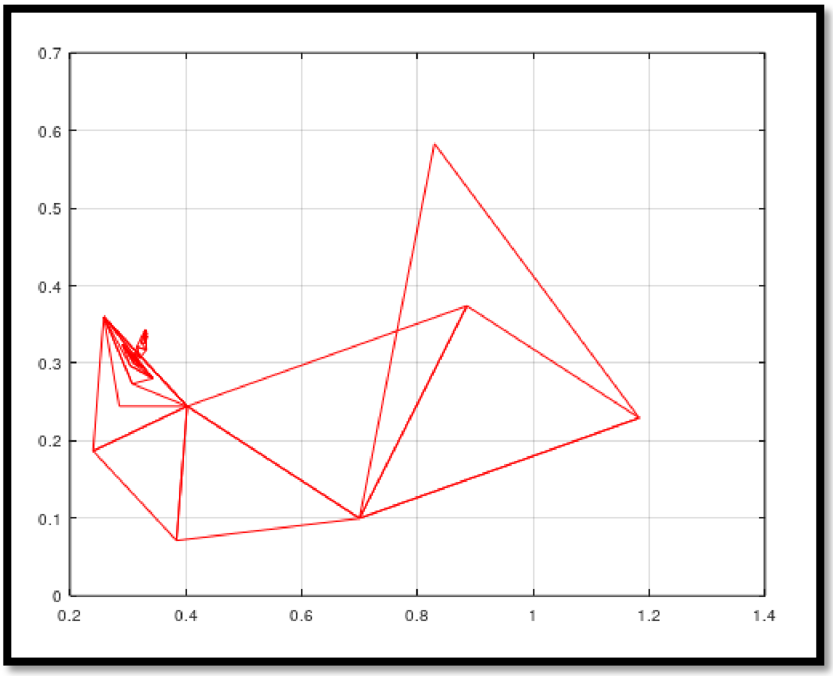
```
function ats = AuksinisPj(X0, grad)
% Randu f-jos f(x) minimumą intervale [l,r] dalijimo pusiau metodu
l = 0;    % apatinis intervalo rezis
r = 3.4;  % viršutinis intervalo rezis
eps = 0.0001;
k = 1; % iteraciju skaitliukas
k_max = 100; % max iteraciju
f = @(X) (X(1).^2).*X(2) + X(1).*(X(2).^2) - X(1).*X(2);
f1 = @(t) f(X0 - t*grad);
L = r-l; % intervalo ilgis
t = (sqrt(5)-1)/2; % pjuvis
x1 = r-t*L;
x2 = l+t*L;
y1 = f1(x1);
y2 = f1(x2);
format short
while L >= eps
      if y1 < y2
          r = x2;
          x2 = x1;
          y2 = y1;
          L = r-l;
          x1 = r-t*L;
          y1 = f1(x1);
      else
          l = x1;
          x1 = x2;
          y1 = y2;
          L = r-l;
          x2 = l+t*L;
          y2 = f1(x2);
      end
      if k == k_max
          break
      end
      k = k+1;
      L = r-l;
end
gama = x1;
i_sk = k+2;
ats = [gama, i_sk];
end
```

Skaičiuojant minimumą greičiausio nusileidimo metodo būdu naudoju auksinio pjūvio algoritmą gamai apskaičiuoti.

Simplekso algoritmas

```
function Simpleksas
a = 7;
b = 1;
eps = 0.0001; %tikslumas
k = 1; %iteraciju skaitliukas
i = 0;
i_max = 100;
k_max = 100;
X0=[a/10, b/10];
f=@(X) (X(1,1).^2).*X(1,2) + X(1,1).*(X(1,2).^2) - X(1,1).*X(1,2);
norma=Inf;
alfa = 1/2;
beta = 0.5;
gama = 2;
eta = -0.5;
teta = 1;
n = 2;
delta1 = alfa*(sqrt(n+1)+n-1)/(n*sqrt(2));
delta2 = alfa*(sqrt(n+1)-1)/(n*sqrt(2));
X1 = [X0(1)+delta2, X0(2)+delta1];
X2 = [X0(1)+delta1, X0(2)+delta2];
y0=f(X0);
y1=f(X1);
y2=f(X2);
Y=[y0, y1, y2];
X=[X0; X1; X2];
[Y1, nr] = sort(Y);
yl=Y1(1);%Y(nr(1));
yg=Y1(2);%Y(nr(2));
yh=Y1(3);%Y(nr(3));
Xl=X(nr(1),:);
Xg=X(nr(2),:);
Xh=X(nr(3),:);
x = [X0(1),X0(1),X1(1); X1(1), X2(1), X2(1)];
y = [X0(2),X0(2),X1(2); X1(2), X2(2), X2(2)];
plot(x, y, 'r');
hold on
i=3;
disp('X1        X2        y        k   i');
```

```
while norma>= eps
      Xc = (Xl+Xg)/2;
      Xnew = Xh+(1+teta)*(Xc-Xh);
      ynew = f(Xnew);
      i = i+1;
      if Xnew(1) <= 0 || Xnew(2) <= 0
          teta = eta;
          Xnew = Xh+(1+teta)*(Xc-Xh);
          ynew = f(Xnew);
          i = i+1;
      end
      if(yl < ynew && ynew < yg)
          teta = 1;
      elseif(ynew < yl)
          teta = gama;
          Z= Xh +(1+teta)*(Xc-Xh);
          i = i+1;
          yz = f(Z);
          if(yz < ynew)
              Xnew = Z;
              ynew = yz;
          end
      elseif(ynew > yh)
          teta = eta;
          Xnew = Xh+(1+teta)*(Xc-Xh);
          ynew = f(Xnew);
          i = i+1;
      elseif(yg < ynew && ynew < yh)
          teta = beta;
          Xnew = Xh+(1+teta)*(Xc-Xh);
          ynew = f(Xnew);
          i = i+1;
      end
      if Xnew(1) <= 0 || Xnew(2) <= 0
          teta = eta;
          Xnew = Xh+(1+teta)*(Xc-Xh);
          ynew = f(Xnew);
          i = i+1;
      end
```



k = 30 (iteracijų)

i = 56 (tikslo funkcijos skaičiavimų)

Išvados

Geičiauso nusileidimo metodas yra greičiausias, nes suskaičiuoja minimumą per 5 iteracijas, tačiau yra sudėtingiausias, nes tikslo funkcija yra skaičiuojama 130 kartus naudojant auskinio pjūvio algoritmą.

Gradientinis algoritmas – 43 iteracijos.

Simplekso algoritmas – 30 iteracijų ir 56 tikslo funkcijos skaičiavimai.

Kai pradinis taškas [0, 0]

Gradientiniu metodu algoritmas nekonverguoja į minimumą, nes gradientas yra lygus 0 (lygiai taip pat elgiasi ir greičiausio nusileidimo algoritmas).

Simplekso metodu algoritmas konverguoja į minimumą per 17 iteracijų, apskaičiuodamas 36 kartus tikslo funkciją.

Kai pradinis taškas [1, 1]

Gradientiniu metodu algoritmas koverguoja į minimumą per 18 iteracijų.

Greičiausio nusileidimo metodu algoritmas konverguoja į minimumą per 2 iteracijas suskaičiuodamas tikslo funkciją 48 kartus.

Simplekso metodu algoritmas konverguoja į minimumą per 45 iteracijas suskaičiuodamas tikslo funkciją 87 kartus.