MEE445 - INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS

Cem Çavuş - 180412025

Comparison of Steepest-Descent and Conjugate-Gradient Methods

---

Question

Example: Solve the following unconstrained optimization problem via CG Method by determining the initial condition as $\mathbf{x}_0 = [-4.5 \quad -3.5]^T$.

$$\min_{\mathbf{x}} f(x_1, x_2) = 3 + (x_1 - 1.5x_2)^2 + (x_2 - 2)^2 \qquad 0$$

check the velidity of CG method in accordance with Golden Section method.

$$-5 \le \mathbf{x} \le 5$$

Listings

Steepest-Descent Algorithm

Step 1 Determine an initial condition ($\mathbf{x}_0$) and maximum number of iteration ($N_{\max}$).

Determine ε_1, ε_2 and ε_3 values as termination criteria.

$k \leftarrow 0$

Step 2 Compute the gradient vector $\nabla f(\mathbf{x}_k)$ at $\mathbf{x}_k$ point.

Select the forward direction as $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.

Find the value of step size (s_k) that makes $f(\mathbf{x}_k + s_k\mathbf{p}_k)$ value minimum via single dimensional optimization.

Use $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k\mathbf{p}_k$ update rule.

$k \leftarrow k + 1$

Step 3 Terminate the algorithm if one of the following conditions is satisfied, unless go to Step 2.

C1: $k > N$: maximum number of iteration is reached.

C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$: function does not change.

C3: $|\Delta x| = \|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon_2$: function does not change.

C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$: algorithm converges to local minimum.

Fig 1. Define of Steepest-Descent Method

Listing 1: Steepest-Descent Method

```
% Steepest Descent

% define the function and variables
syms x y % x -> x1 , y -> x2 for given function
f = 3 + (x - 1.5*y)^2 + (y - 2)^2;
fi = inline(f); % use inline function for give a numerical
    value back

iter = 20; k = 0; % define maximum number of iteration and step

% define sk and pk for next point -> x(k+1) = x(k) + s(k)*p(k)
syms sk pk;
pk = -gradient(f); pki = inline(pk); % use inline function for
    give a numerical value back

x0 = [-4.5; -3.5]; % start point

pk_value = pki(x0(1),x0(2)); % find pk value for start point
sk = vpasolve(diff(fi(x0(1) + sk*pk_value(1),x0(2) + sk*
    pk_value(2))));
```

```matlab

tol = 1e-4; % set the tolerans

x1 = [x0(1) + sk*pk_value(1);x0(2) + sk*pk_value(2)]; % next point

% draw the function 3d using meshgrid and mesh matlab functions - define all points
a = linspace(5,-5); b = linspace(5,-5); [a,b] = meshgrid(a,b); c = 3 + (a - 1.5.*b).^2 + (b - 2).^2;
mesh(a,b,c); hold on;

% draw first point - fi(x0(1),x0(2)) -> value of f(-4.5, -3.5)
plot3(x0(1),x0(2),fi(x0(1),x0(2)),'.','Color', [0, 0, 0],'MarkerSize',15)
% draw line for connecting the points
plot3([x0(1),x1(1)],[x0(2),x1(2)],[fi(x0(1),x0(2)),fi(x1(1),x1(2))],'r','linewidth',3)

% define a loop for finding a minimum point and use some criterias
while x1(1) - x0(1) > tol || norm(fi(x1(1),x1(2)) - fi(x0(1),x0(2))) > tol || k < iter
    % if the condition is true, continue with next point x0 -> x1 x1 -> x2
    x0 = x1;
    % draw the next point
    plot3(x0(1),x0(2),fi(x0(1),x0(2)),'.','Color', [0, 0, 0],'MarkerSize',15)
    % calculate again sk pk value for next point
    pki_solution = pki(x0(1),x0(2));
    syms sk; sk = vpasolve(diff(fi(x0(1) + sk*pki_solution(1), x0(2) + sk*pki_solution(2))));
    x1 = [x0(1) + sk*pki_solution(1);x0(2) + sk*pki_solution(2)]; % find next point
    plot3([x0(1),x1(1)],[x0(2),x1(2)],[fi(x0(1),x0(2)),fi(x1(1),x1(2))],'r','linewidth',3)
    % continue next step
    k = k + 1;
end

```

```
% display end point (minimum point)
disp(x1);
```

```
>> steepestdescentmethod
 2.9850046903039538071566739905194
  1.992926350460623726962182765183
```

Fig 2. End point

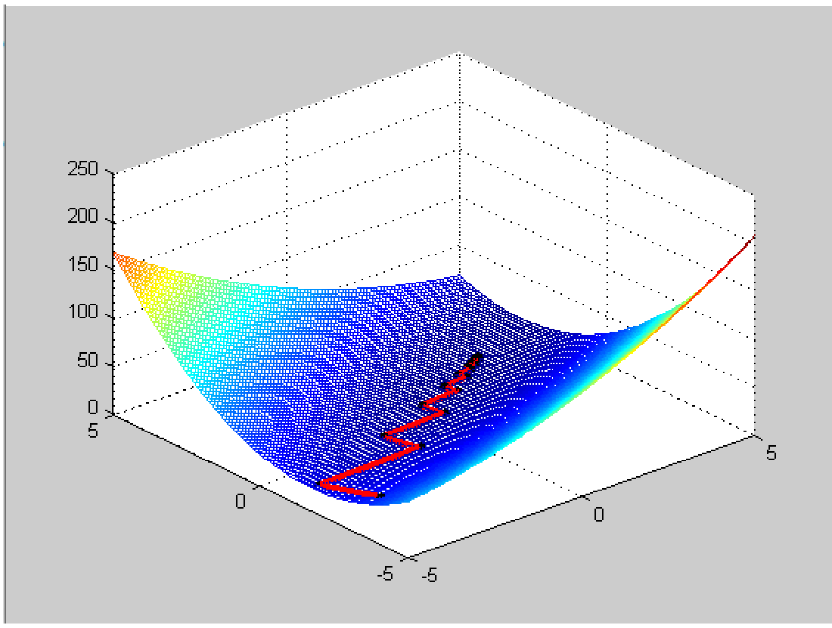


Fig 3. Result0

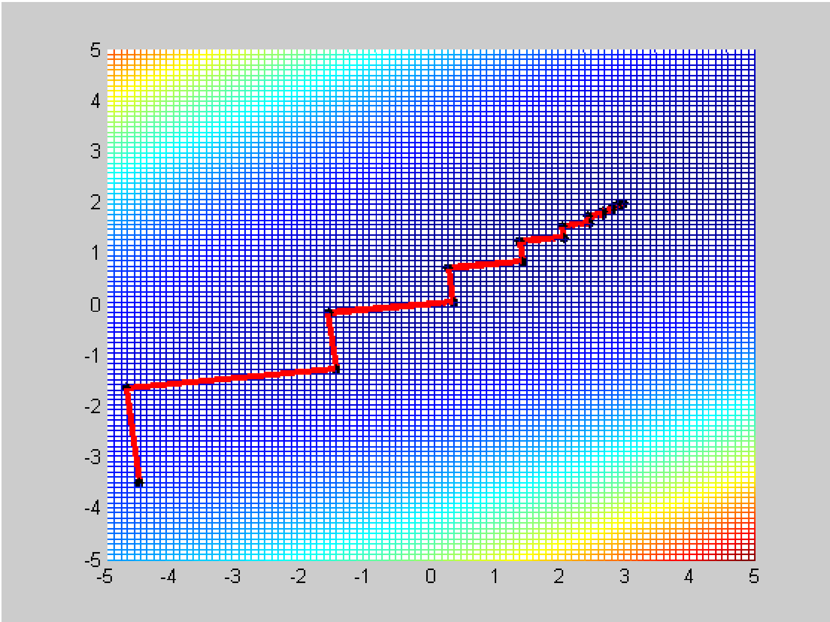


Fig 4. Result1

Conjugate-Gradient Algorithm

Step 1 Determine the initial condition ($\mathbf{x}_0$) and maximum number of iteration ($N_{\max}$).

Determine ε_1, ε_2 and ε_3 as ending criteria.

$k \leftarrow 0$

Step 2 Compute the gradient vector $\nabla f(\mathbf{x}_k)$ at $\mathbf{x}_k$ point.

Cond. statement

If $k = 0$ determine the forward direction as $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.

If $k \neq 0$ determine the forward direction as $\mathbf{p}_k = -\nabla f(\mathbf{x}_k) + \beta \mathbf{p}_{k-1}$ where $\beta = \frac{\nabla^T f(\mathbf{x}_k) \nabla f(\mathbf{x}_k)}{\nabla^T f(\mathbf{x}_{k-1}) \nabla f(\mathbf{x}_{k-1})}$.

Determine step-size (s_k) that makes $f(\mathbf{x}_k + s_k \mathbf{p}_k)$ value minimum.

Use the update rule $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{p}_k$.

$k \leftarrow k + 1$

Step 3 If one of the following conditions is satisfied terminate the algorithm, unless go to Step 2.

$C1$: $k > N$

$C2$: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$

$C3$: $|\Delta x| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$

$C4$: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$

Fig 5. Define of Conjugate-Gradient Method

Listing 2: Conjugate-Gradient Method

```matlab
% Conjugate Gradient

% define the function and variables
syms x y % x -> x1 , y -> x2
f = 3 + (x - 1.5*y)^2 + (y - 2)^2;
fi = inline(f); % use inline function for give a numerical value back

% define maximum number of iteration and step
iter = 20; k = 0;

% define sk and pk for next point -> x(k+1) = x(k) + s(k)*p(k)
syms sk pk;
pk = -gradient(f);
pki = inline(pk); % use inline function for give a numerical value back

% start point
x0 = [-4.5; -3.5];
% previous pk
pki_p = pki(x0(1),x0(2));
% find pk value
pk_value = pki(x0(1),x0(2));
sk = vpasolve(diff(fi(x0(1) + sk*pk_value(1),x0(2) + sk*pk_value(2))));

% set the tolerans
tol = 1e-5;

% next point
x1 = [x0(1) + sk*pk_value(1);x0(2) + sk*pk_value(2)];
% draw the function 3d using meshgrid - define all points

a = linspace(5,-5);
b = linspace(5,-5);
[a,b] = meshgrid(a,b);
c = 3 + (a - 1.5.*b).^2 + (b - 2).^2;
mesh(a,b,c);
hold on;

% draw first point
```

```
% fi(x0(1),x0(2)) -> value of f(-4.5, -3.5)
plot3(x0(1),x0(2),fi(x0(1),x0(2)),'.','Color', [0, 0, 0],'MarkerSize',15)
plot3([x0(1),x1(1)],[x0(2),x1(2)],[fi(x0(1),x0(2)),fi(x1(1),x1(2))],'r','linewidth',3)

% define a loop for finding a minimum point and use some criterias
while norm(fi(x1(1),x1(2)) - fi(x0(1),x0(2))) > tol && k < iter
    % if the condition is true, continue with next point x0 -> x1 x1 -> x2
    x0 = x1;
    % draw the next point
    plot3(x0(1),x0(2),fi(x0(1),x0(2)),'.','Color', [0, 0, 0],'MarkerSize',15)
    % calculate again sk pk value for next point
    syms beta
    beta = (pki(x0(1),x0(2))'*pki(x0(1),x0(2)))/(pki_p'*pki_p);
    pki_solution = pki(x0(1),x0(2)) + beta*pki_p;
    pki_p = pki(x0(1),x0(2));
    syms sk;
    sk = vpasolve(diff(fi(x0(1) + sk*pki_solution(1),x0(2) + sk*pki_solution(2))));
    % find next point
    x1 = [x0(1) + sk*pki_solution(1);x0(2) + sk*pki_solution(2)];
    plot3([x0(1),x1(1)],[x0(2),x1(2)],[fi(x0(1),x0(2)),fi(x1(1),x1(2))],'r','linewidth',3)
    % continue next step
    k = k + 1;
end

% minimum point
disp(x1);
```

```
>> conjugategradient
  3.0
  2.0
```

Fig 6. End point

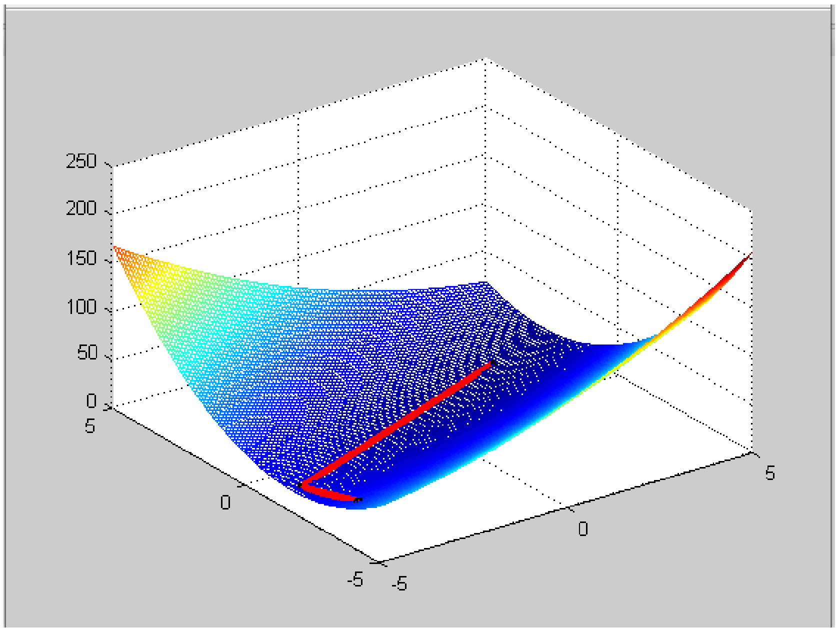


Fig 7. Result0

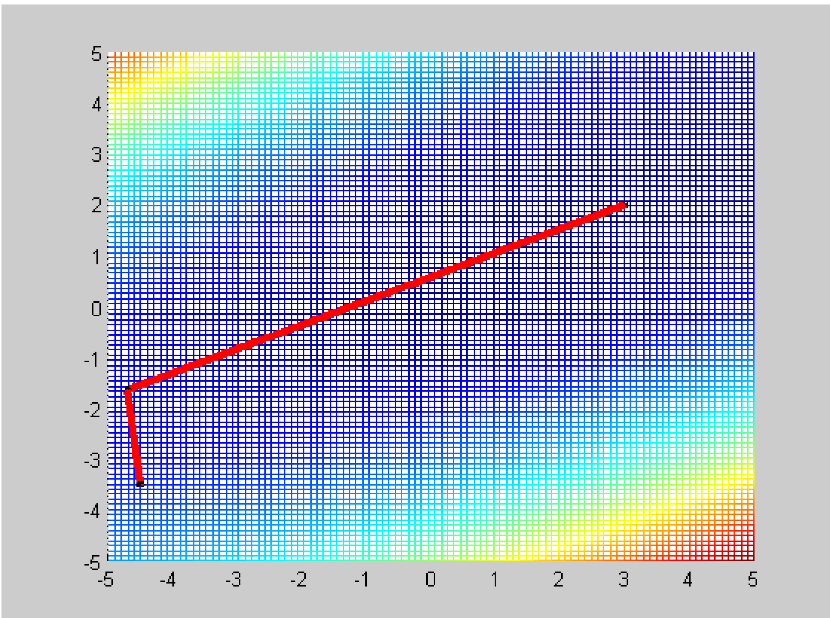


Fig 8. Result1

Combined Steepest-Descent Method and Conjugate-Gradient Method

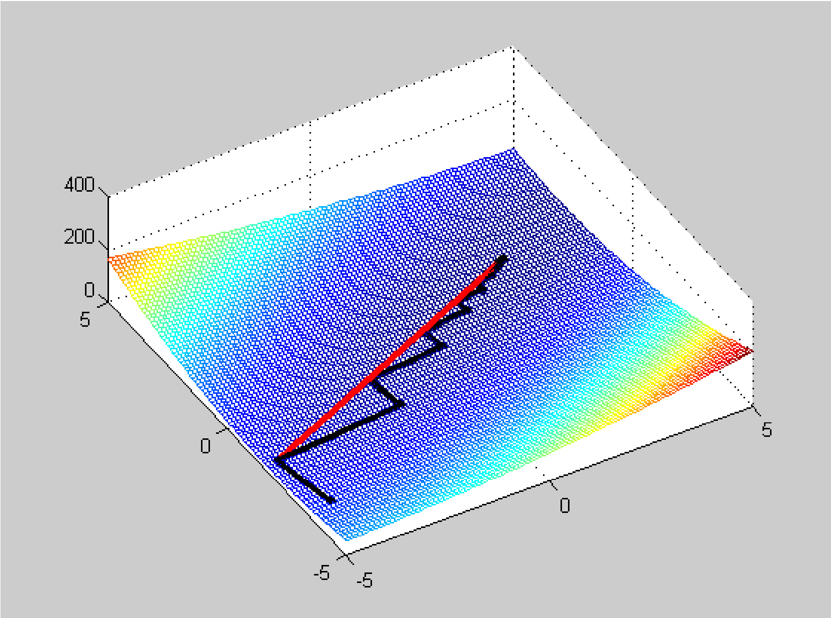


Fig 9. Red-Conjugate Descent Black-Steepest Descent

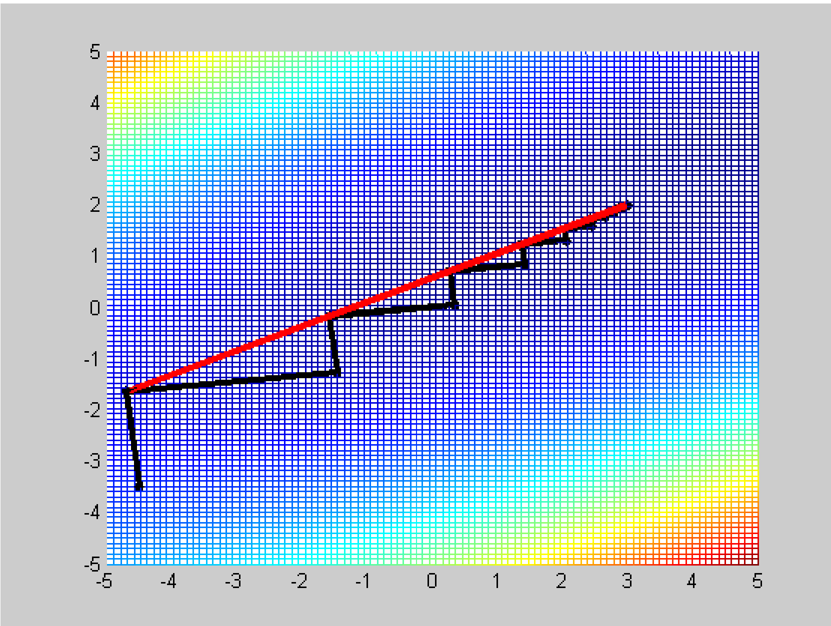
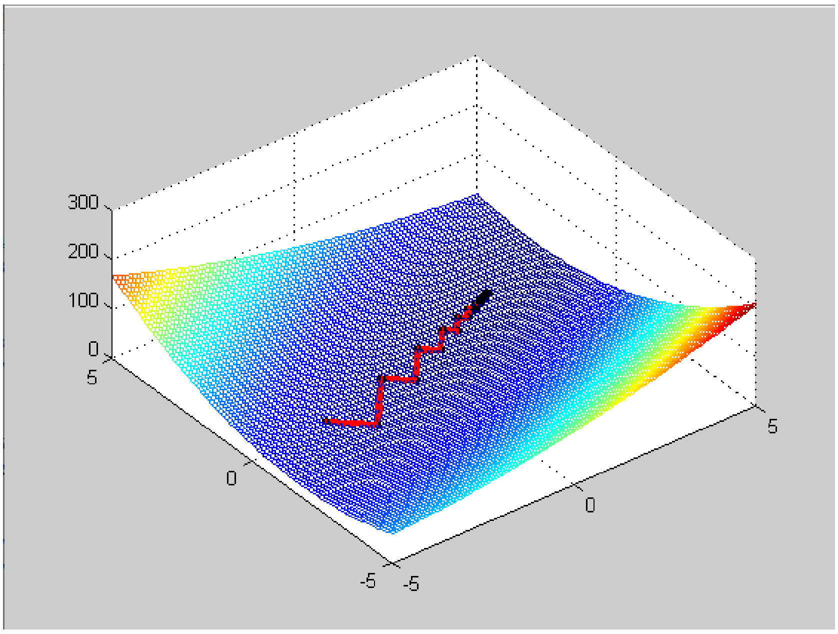


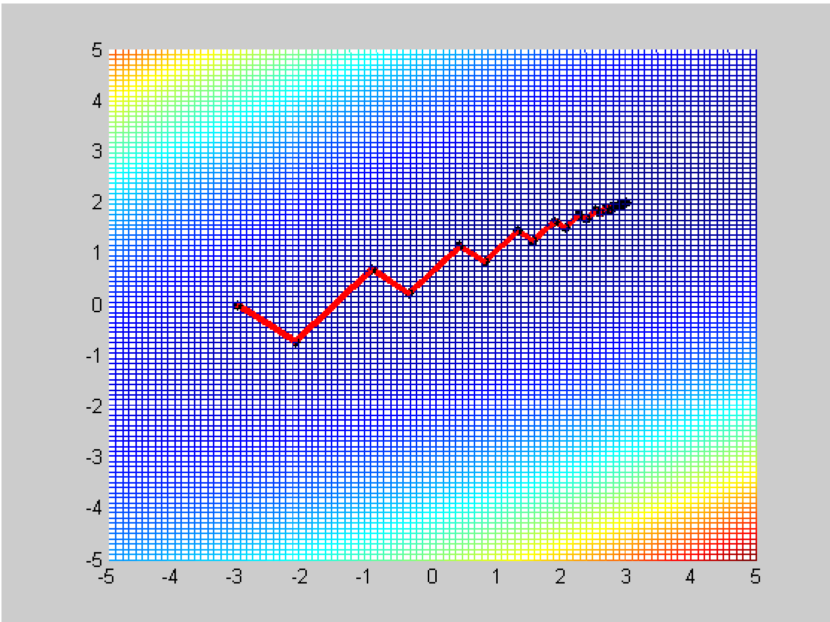
Fig 10. Red-Conjugate Descent Black-Steepest Descent

Select Different Three Point from -5 < x1 < 5 and -5 < x2 < 5

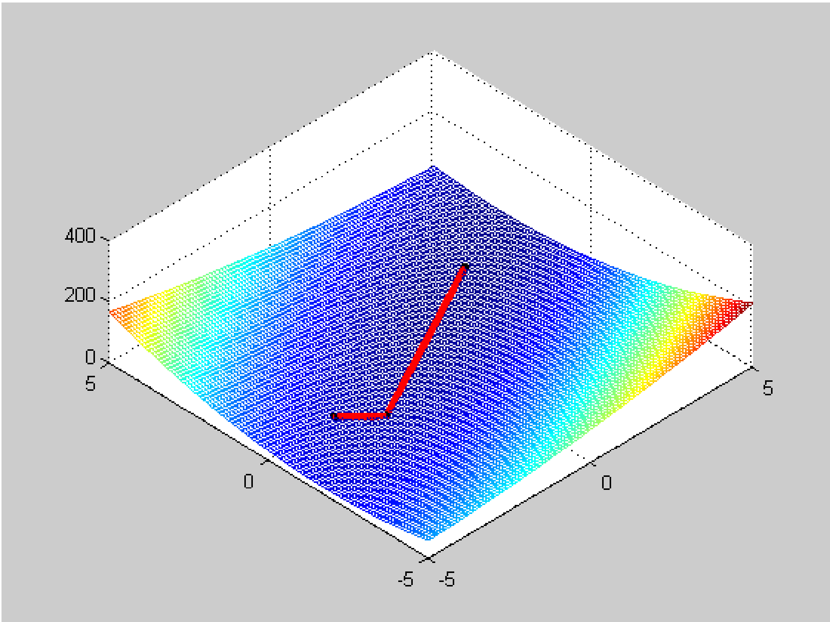
New Point x0 = [-3, 0]'

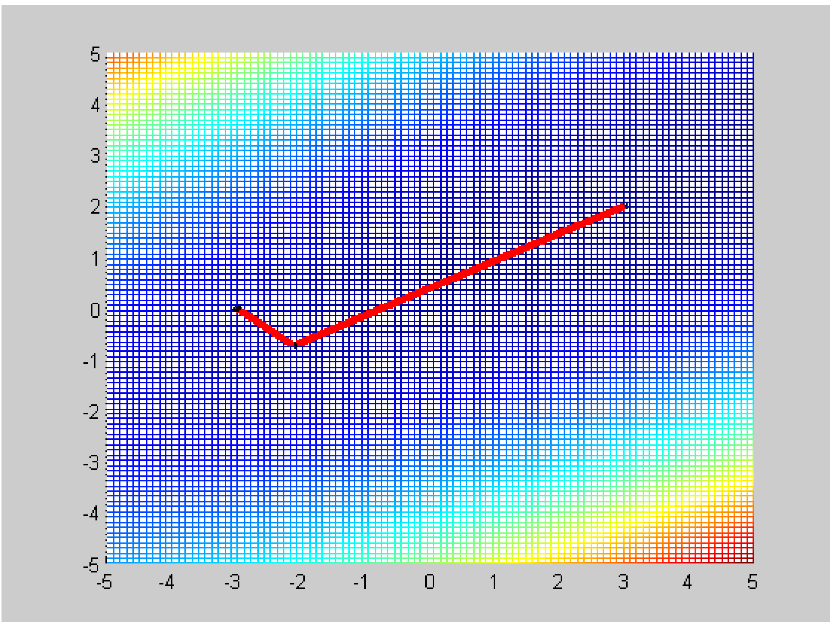
```
>> steepestdescentmethod
 2.99954043039512139061167695629l9
 1.99975428951818371379238173900 76
```

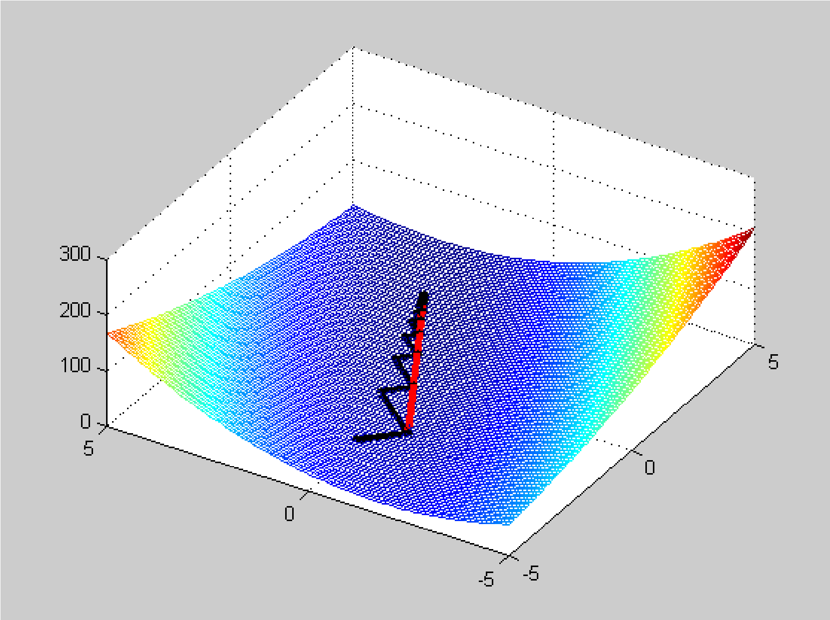


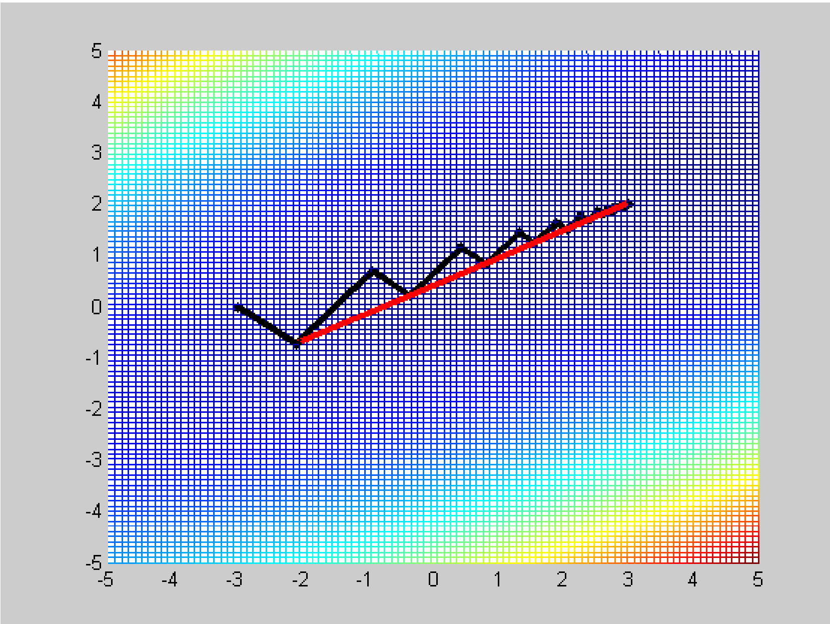


```
>> conjugategradient
 3.0
 2.0
```



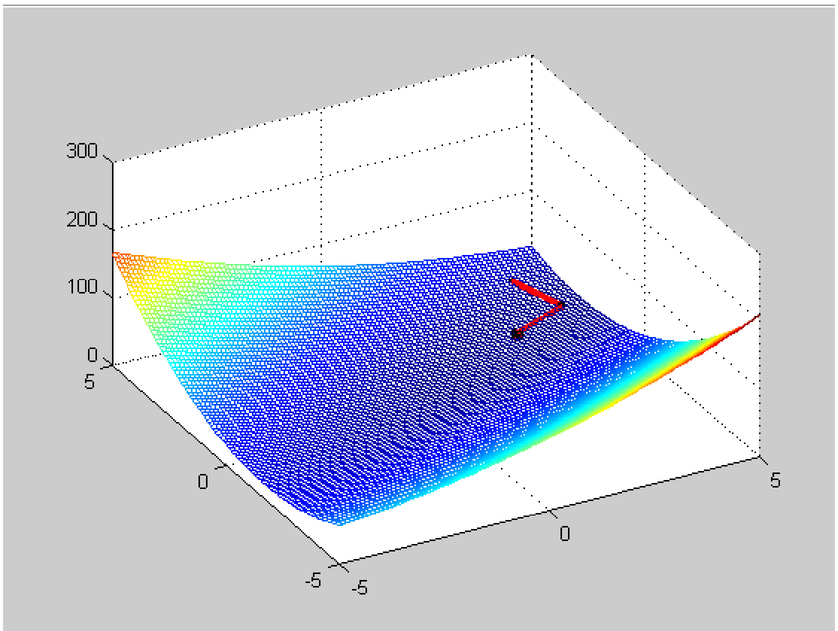


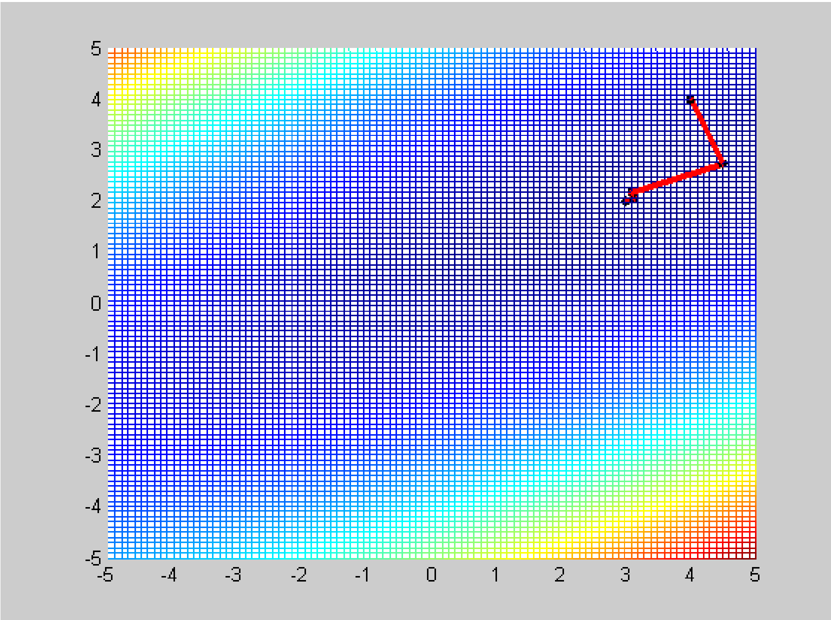




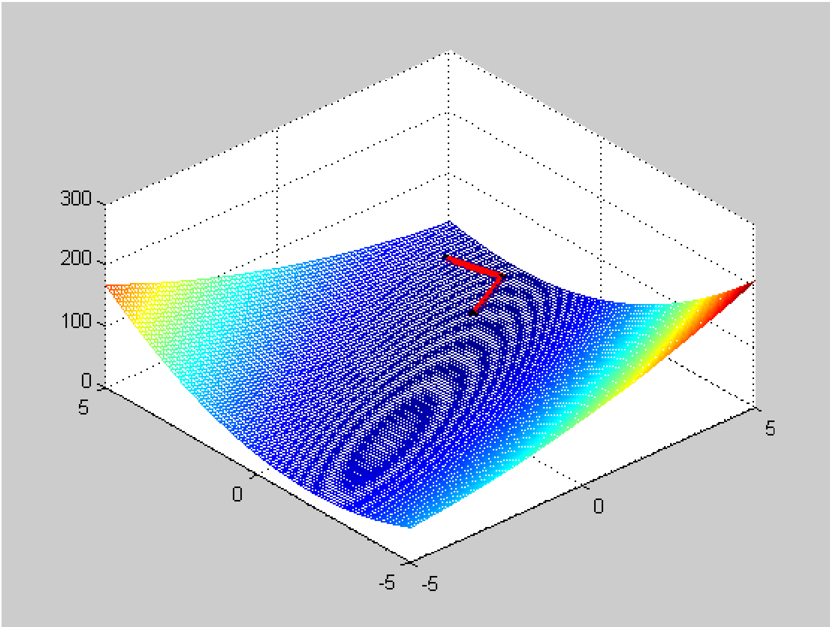
New Point x0 = [4, 4]'

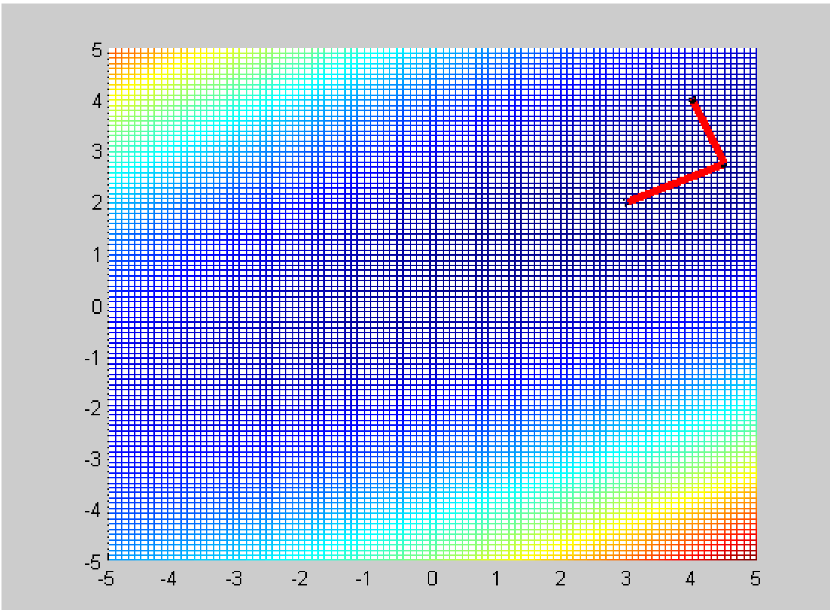
```
>> steepestdescentmethod
 3.00000000004116928148678067761 66
 2.00000000002031730774672293181 08
```

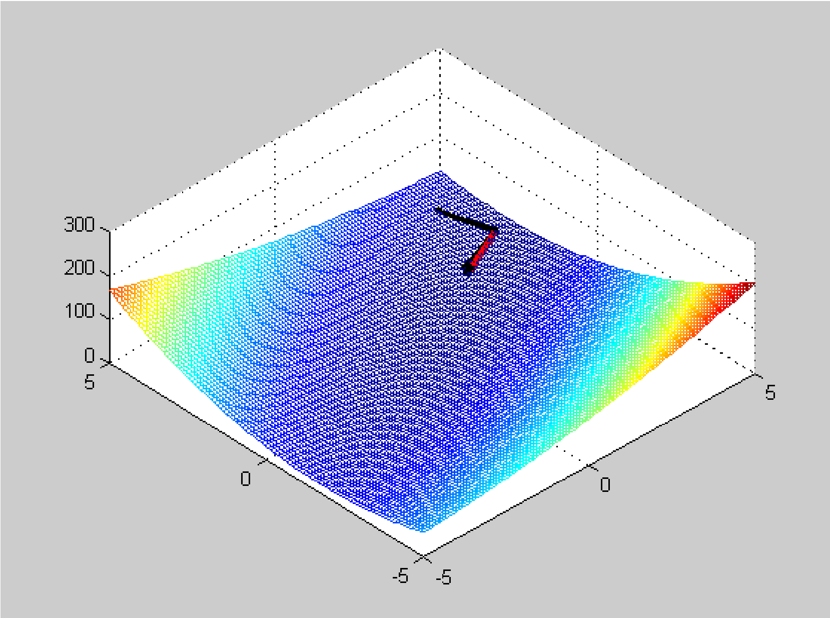


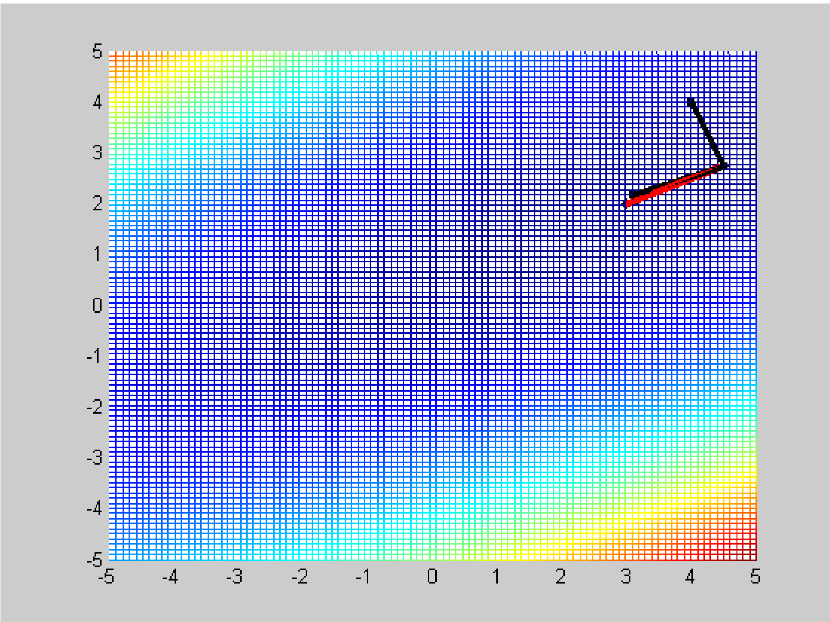



```
>> conjugategradient
 3.0
 2.0
```


300
200
100
0
5
0
-5
-5
0
5

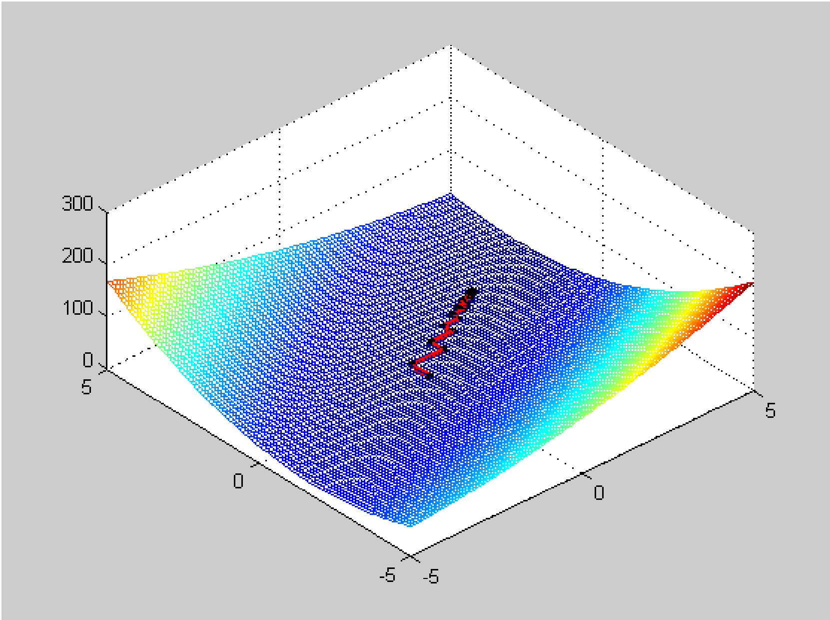

5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

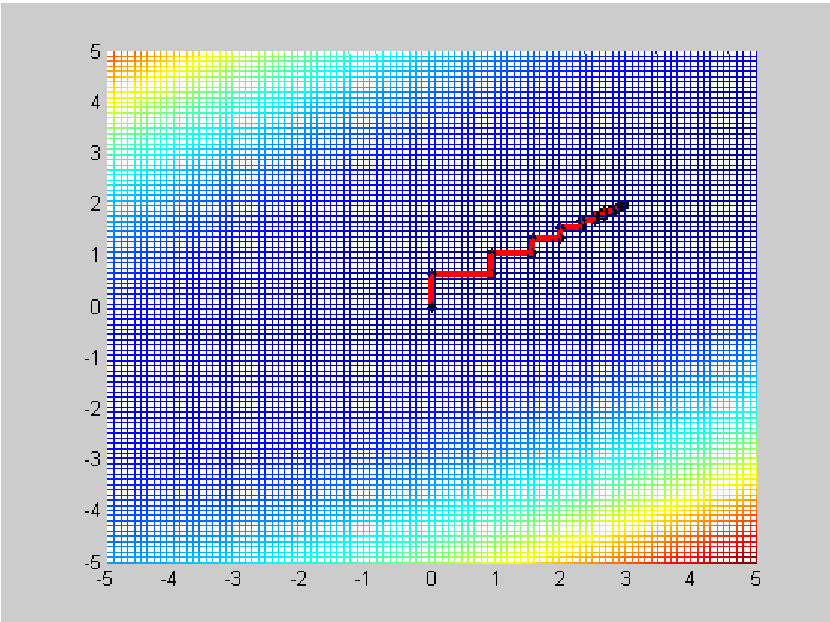




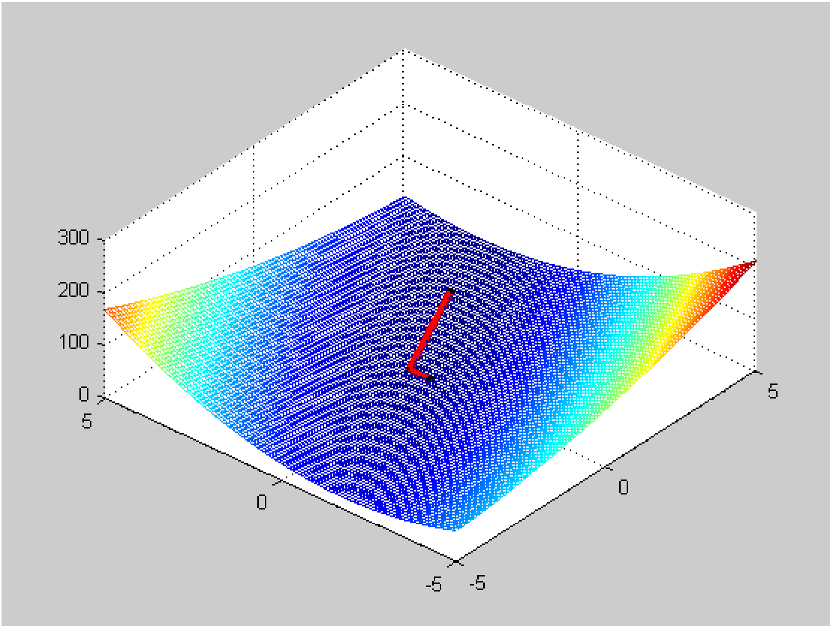
New Point x0 = [0, 0]'

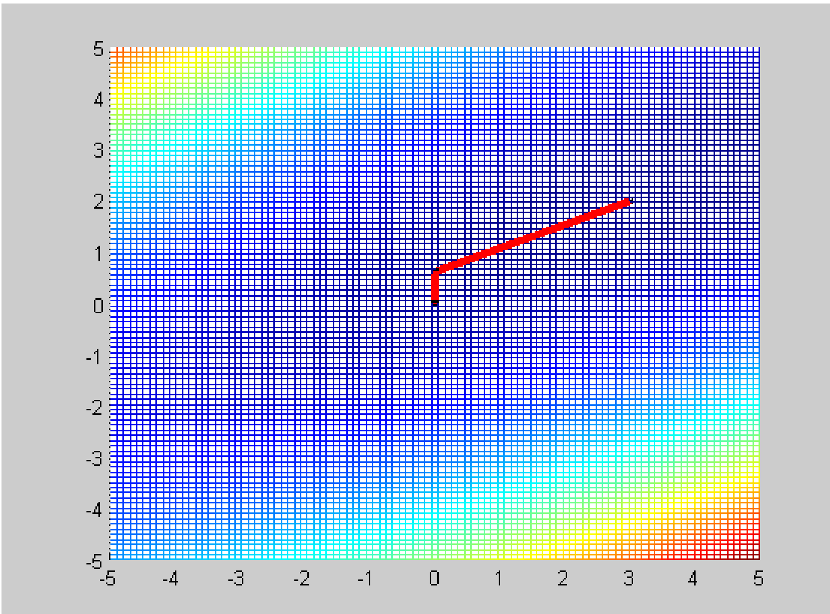
```
>> steepestdescentmethod
 2.97482263375559287692118497l4212
 1.98837967711796594319439306373 29
```

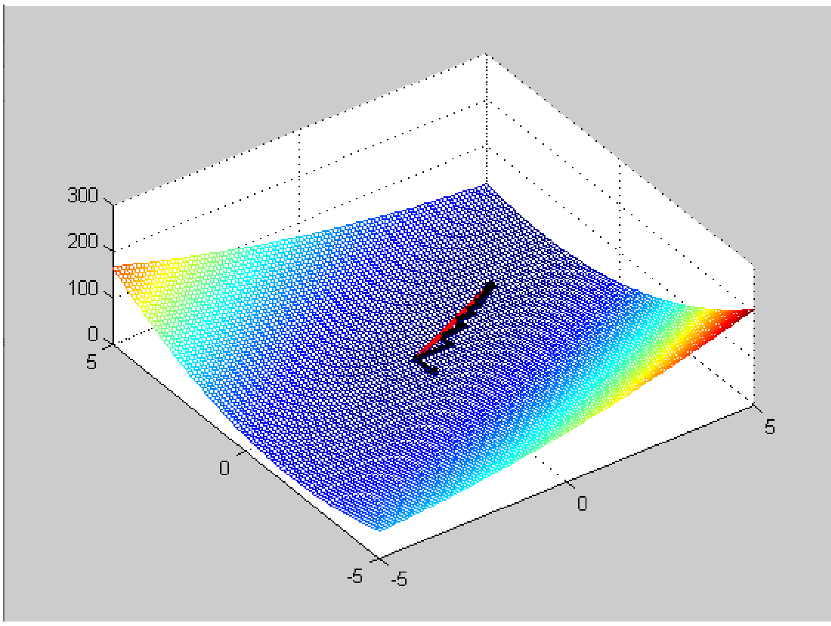


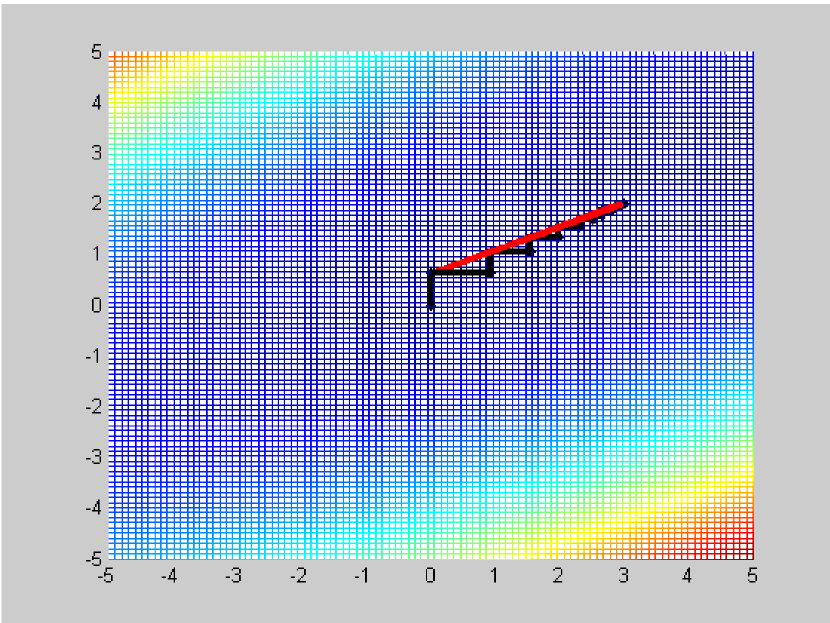



```
>> conjugategradient
 3.0
 2.0
```


300
200
100
0
5
0
-5
-5
0
5


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5


300
200
100
0
5
0
-5
-5
0
5


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5