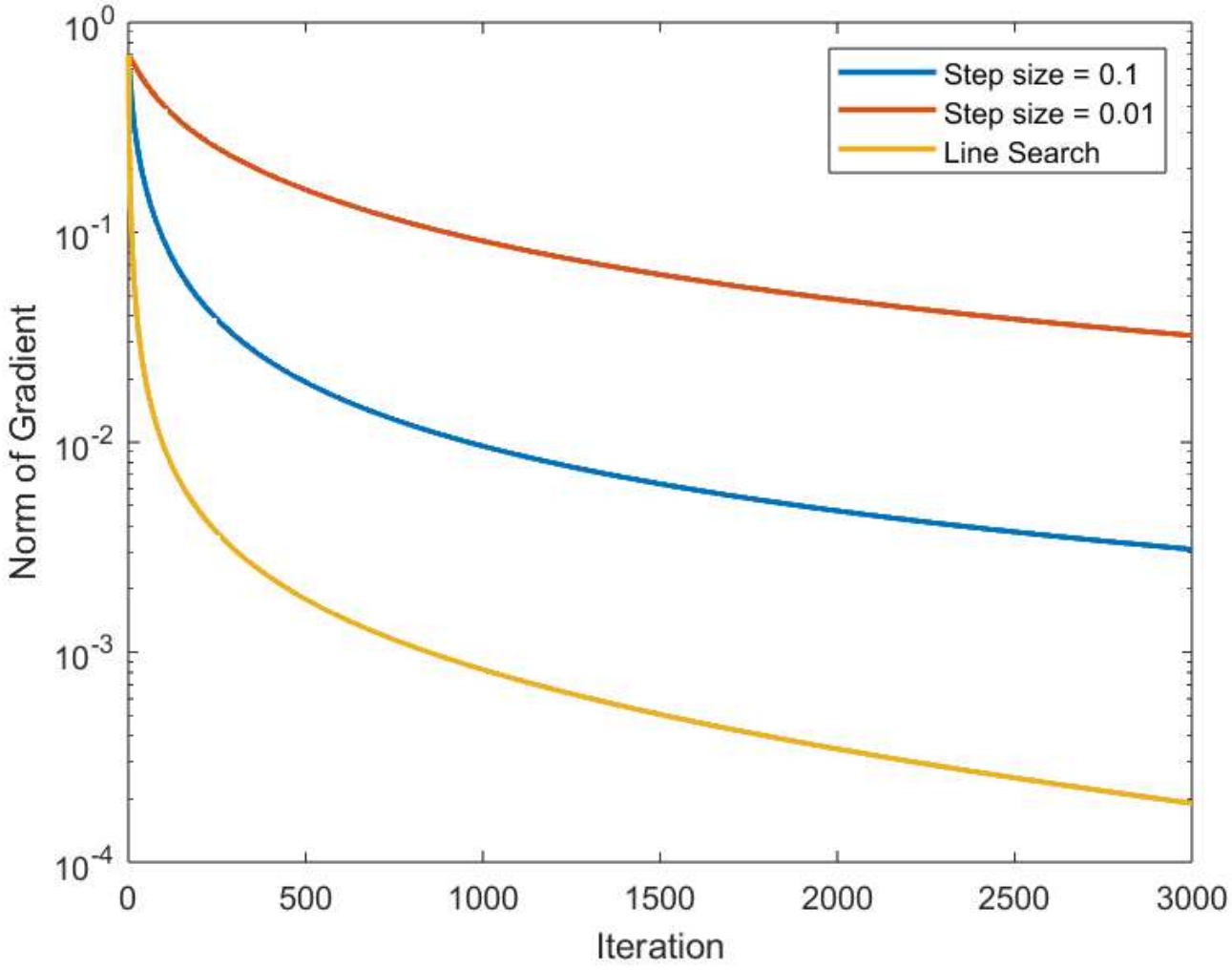
```
clear; clc;

rand('seed',xxx);

m = 500; n = 1000;
A = randn(n,m); b = sign(rand(m,1)-0.5);

e = ones(m,1);
p = @(x) 1./(1+exp(-b.*(A'*x)));
l = @(x) -1/m*e'*log(p(x))+1/(100*m)*sum_square(x);
g = @(x) -1/m*A*(b.*(e-p(x)))+1/(50*m)*x;

max_iter = 3000;

% constant step size gradient descent(step size = 0.1)
w = zeros(n,1); alpha1 = 0.1;
for iter_number = 1:max_iter
    grad = g(w);
    ng = norm(grad);
    ng_alpha1(iter_number) = ng;
    funcv = l(w);
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-4
        break
    end
    w = w-alpha1*grad;
end

% constant step size gradient descent(step size = 0.01)
w = zeros(n,1); alpha2 = 0.01;
for iter_number = 1:max_iter
    grad = g(w);
    ng = norm(grad);
    ng_alpha2(iter_number) = ng;
    funcv = l(w);
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-4
        break
    end
    w = w-alpha2*grad;
end

% line search gradient descent
w = zeros(n,1); alpha = 1e-4; beta = 0.5;
for iter_number = 1:max_iter
    grad = g(w);
    ng = norm(grad);
    ng_ls(iter_number) = ng;
    funcv = l(w);
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-4
        break
    end
    t = 1;
    while l(w-t*grad) > l(w)-alpha*t*sum_square(grad)
        t = t * beta;
    end
    w = w-t*grad;
end

figure;
semilogy(ng_alpha1,'DisplayName','Step size = 0.1','LineWidth',1.8);
```

```
hold on;
semilogy(ng_alpha2,'DisplayName','Step size = 0.01','LineWidth',1.8);
semilogy(ng_ls,'DisplayName','Line Search','LineWidth',1.8);
xlabel('Iteration');
ylabel('Norm of Gradient');
legend;
hold off;
```

运行结果：




**第二题**

matlab代码：

```
clear; clc;

% dataset = 'a9a.test';
% dataset = 'CINA.test';
dataset = 'ijcnn1.test';
[b,A] = libsvmread(dataset);
[m,n] = size(A);
mu = 1e-2/m;

e = ones(m,1);
p = @(x) 1./(1+exp(-b.*(A*x)));
l = @(x) -1/m*e'*log(p(x))+1/(100*m)*sum_square(x);
g = @(x) -1/m*A'*(b.*(e-p(x)))+1/(50*m)*x;
h = @(x) 1/m*A'*(spdiags(p(x),0,m,m)*spdiags(e-p(x),0,m,m))*A+1/(50*m)*eye(n);

w0 = zeros(n,1);
alpha = 1e-4;
beta = 0.5;

% optimal value l(x*) is obtained by running Newton's method
% opt_val = 3.187971e-01; % for 'a9a.test'
% opt_val = 1.593068e-01; % for 'CINA.test'
opt_val = 1.956580e-01; % for 'ijcnn1.test'

% Exact Newton's Method
w = w0; iter_number = 1;
while iter_number >= 1
    grad = g(w);
    ng = norm(grad);
    ng_exact_newton(iter_number) = ng;
    funcv = l(w);
    funcv_diff_exact_newton(iter_number) = funcv-opt_val;
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-6
        break
    end
    hessian = h(w);
    nt_direction = -hessian\grad;
    t = 1;
    while l(w+t*nt_direction) > l(w)+alpha*t*nt_direction'*grad
        t = t*beta;
    end
    w = w+t*nt_direction;
    iter_number = iter_number+1;
end

% Newton CG with tolerance rule 1: η = min(0.5, ‖grad‖) * ‖grad‖
w = w0; iter_number = 1;
while iter_number >= 1
    grad = g(w);
    hessian = h(w);
    ng = norm(grad);
    ng_newton_cg1(iter_number) = ng;
    funcv = l(w);
    funcv_diff_newton_cg1(iter_number) = funcv-opt_val;
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-6
        break
    end

    nt_direction = zeros(n,1); CG_tol = min(0.5,ng)*ng;
    r = grad; s = -r;
```

```
    CG_maxiter = 3000;
    for iter = 1 : CG_maxiter
        rr = r' * r;
        Hs = hessian * s;
        lambda = rr / (s' * Hs);
        nt_direction = nt_direction + lambda * s;
        r = r + lambda * Hs; % r is residual
        nr1 = norm(r);
        if nr1 <= CG_tol
            break;
        end
        gamma = nr1^2 / rr;
        s = -r + gamma * s;
    end

    t = 1;
    while l(w+t*nt_direction) > l(w) + alpha*t*nt_direction'*grad
        t = t*beta;
    end
    w = w+t*nt_direction;
    iter_number = iter_number+1;
end

% Newton CG with tolerance rule 2: η = min(0.5, sqrt( ‖ grad ‖ )) *  ‖ grad ‖
w = w0; iter_number = 1;
while iter_number >= 1
    grad = g(w);
    hessian = h(w);
    ng = norm(grad);
    ng_newton_cg2(iter_number) = ng;
    funcv = l(w);
    funcv_diff_newton_cg2(iter_number) = funcv-opt_val;
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-6
        break
    end

    nt_direction = zeros(n,1); CG_tol = min(0.5,sqrt(ng))*ng;
    r = grad; s = -r;
    CG_maxiter = 3000;
    for iter = 1 : CG_maxiter
        rr = r' * r;
        Hs = hessian * s;
        lambda = rr / (s' * Hs);
        nt_direction = nt_direction + lambda * s;
        r = r + lambda * Hs; % r is residual
        nr1 = norm(r);
        if nr1 <= CG_tol
            break;
        end
        gamma = nr1^2 / rr;
        s = -r + gamma * s;
    end

    t = 1;
    while l(w+t*nt_direction) > l(w) + alpha*t*nt_direction'*grad
        t = t*beta;
    end
    w = w+t*nt_direction;
    iter_number = iter_number+1;
end
```

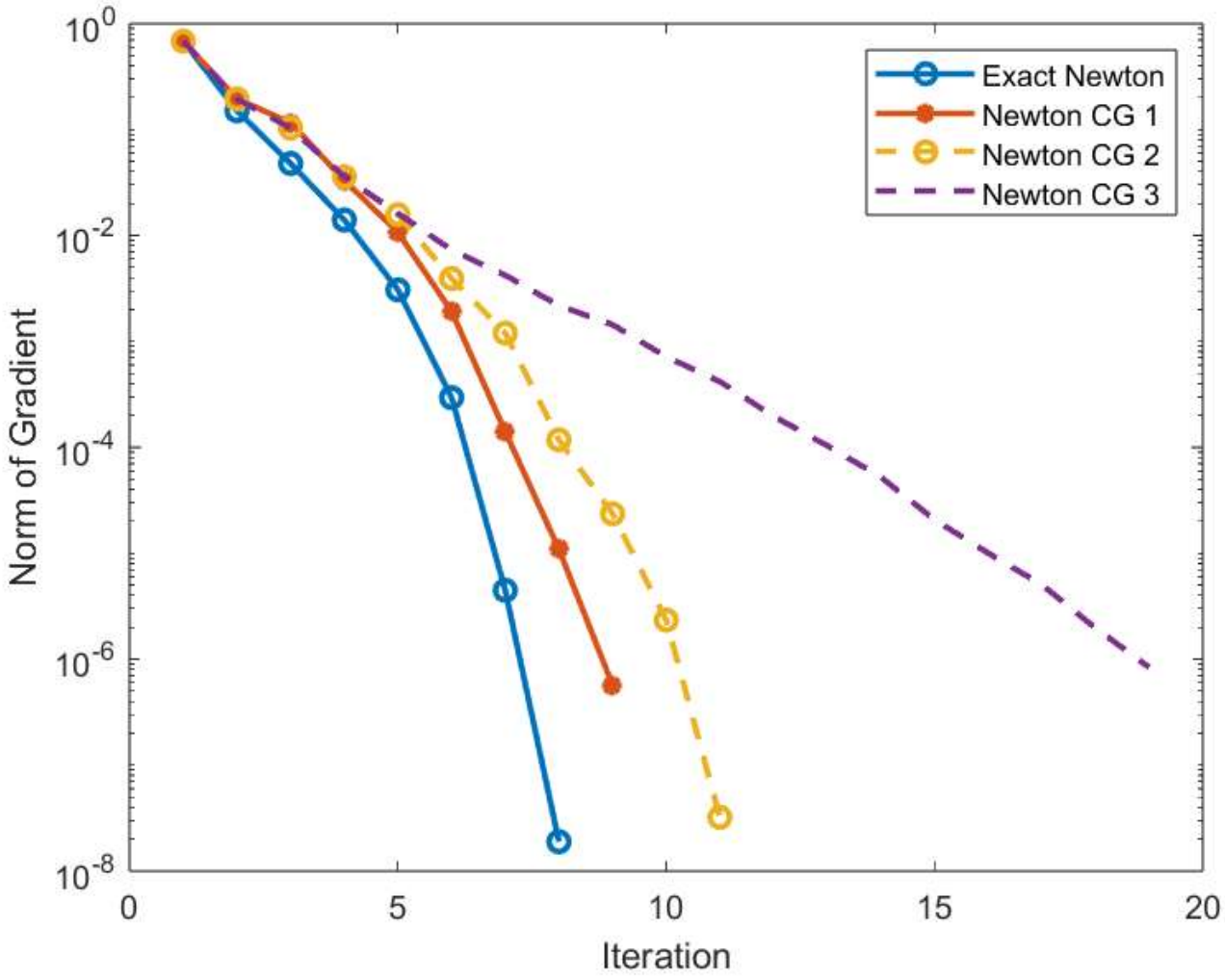
```
% Newton CG with tolerance rule 3: η = 0.5 * ‖grad‖
w = w0; iter_number = 1;
while iter_number >= 1
    grad = g(w);
    hessian = h(w);
    ng = norm(grad);
    ng_newton_cg3(iter_number) = ng;
    funcv = l(w);
    funcv_diff_newton_cg3(iter_number) = funcv-opt_val;
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    if ng < 1e-6
        break
    end

    nt_direction = zeros(n,1); CG_tol = 0.5*ng;
    r = grad; s = -r;
    CG_maxiter = 3000;
    for iter = 1 : CG_maxiter
        rr = r' * r;
        Hs = hessian * s;
        lambda = rr / (s' * Hs);
        nt_direction = nt_direction + lambda * s;
        r = r + lambda * Hs; % r is residual
        nr1 = norm(r);
        if nr1 <= CG_tol
            break;
        end
        gamma = nr1^2 / rr;
        s = -r + gamma * s;
    end

    t = 1;
    while l(w+t*nt_direction) > l(w)+alpha*t*nt_direction'*grad
        t = t*beta;
    end
    w = w+t*nt_direction;
    iter_number = iter_number+1;
end

% gradient descent with Armijo rule line search
w = w0;
for iter_number = 1 : 1000
    grad = g(w);
    ng = norm(grad);
    ng_gd_ls(iter_number) = ng;
    funcv = l(w);
    funcv_diff_gd_ls(iter_number) = funcv-opt_val;
    fprintf("iter_number = %d norm_gradient = %e fun_val = %e\n",iter_number,ng,funcv);
    t = 1;
    while l(w-t*grad) > l(w)-alpha*t*sum_square(grad)
        t = t*beta;
    end
    w = w-t*grad;
end

figure;
semilogy(ng_exact_newton, '-o', 'DisplayName', 'Exact Newton', 'LineWidth', 1.8);
hold on;
semilogy(ng_newton_cg1, '-*', 'DisplayName', 'Newton CG 1', 'LineWidth', 1.8);
semilogy(ng_newton_cg2, '--o', 'DisplayName', 'Newton CG 2', 'LineWidth', 1.8);
semilogy(ng_newton_cg3, '--', 'DisplayName', 'Newton CG 3', 'LineWidth', 1.8);
semilogy(ng_gd_ls, '-|', 'DisplayName', 'Gradient Descent', 'LineWidth', 1.8);
```

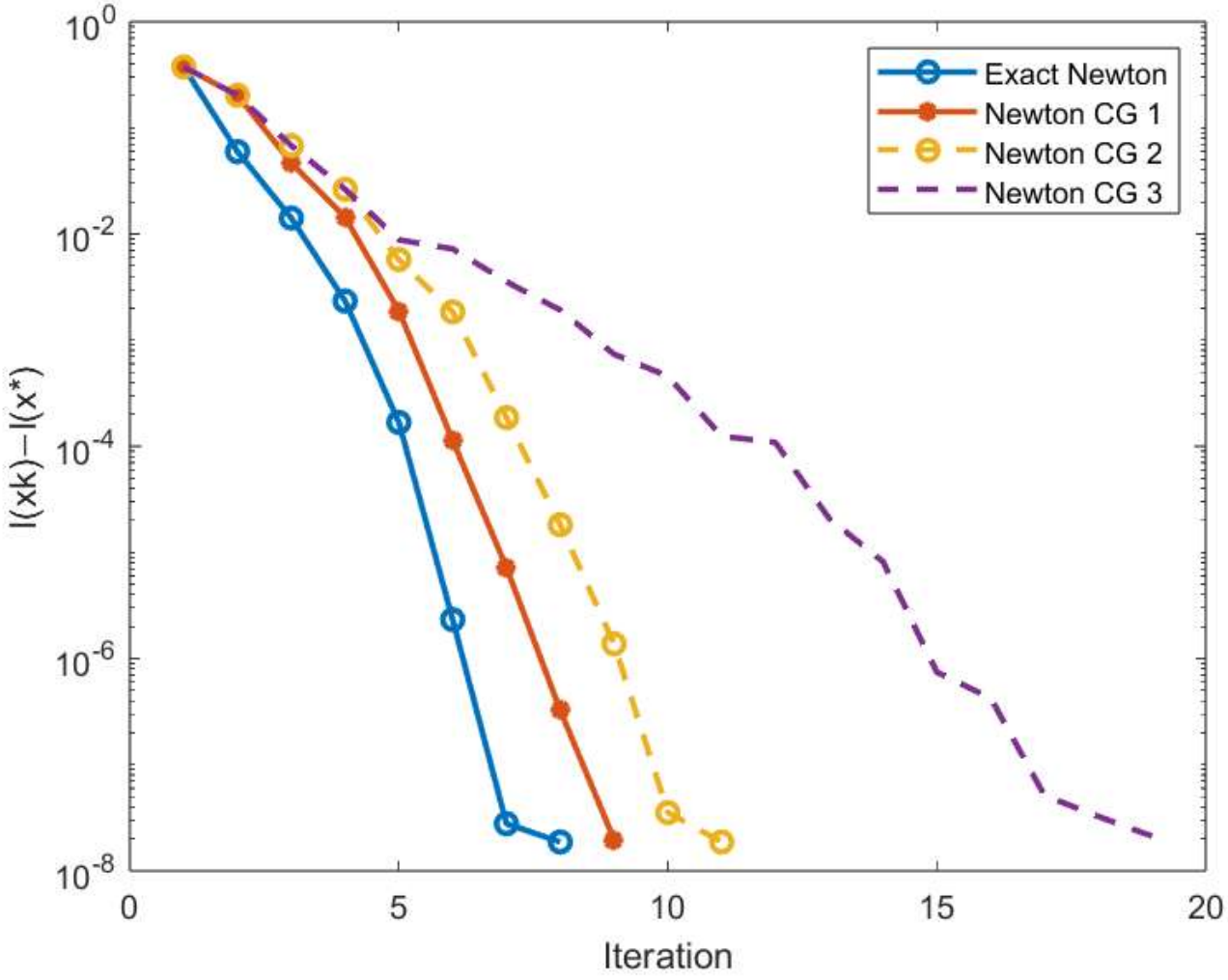
```
xlabel('Iteration');
ylabel('Norm of Gradient');
legend;
hold off;

figure;
semilogy(funcv_diff_exact_newton, '-o', 'DisplayName', 'Exact Newton', 'LineWidth', 1.8);
hold on;
semilogy(funcv_diff_newton_cg1, '-*', 'DisplayName', 'Newton CG 1', 'LineWidth', 1.8);
semilogy(funcv_diff_newton_cg2, '--o', 'DisplayName', 'Newton CG 2', 'LineWidth', 1.8);
semilogy(funcv_diff_newton_cg3, '--', 'DisplayName', 'Newton CG 3', 'LineWidth', 1.8);
semilogy(funcv_diff_gd_ls, '-|', 'DisplayName', 'Gradient Descent', 'LineWidth', 1.8);
xlabel('Iteration');
ylabel('l(xk)-l(x*)');
legend;
hold off;
```

运行结果：

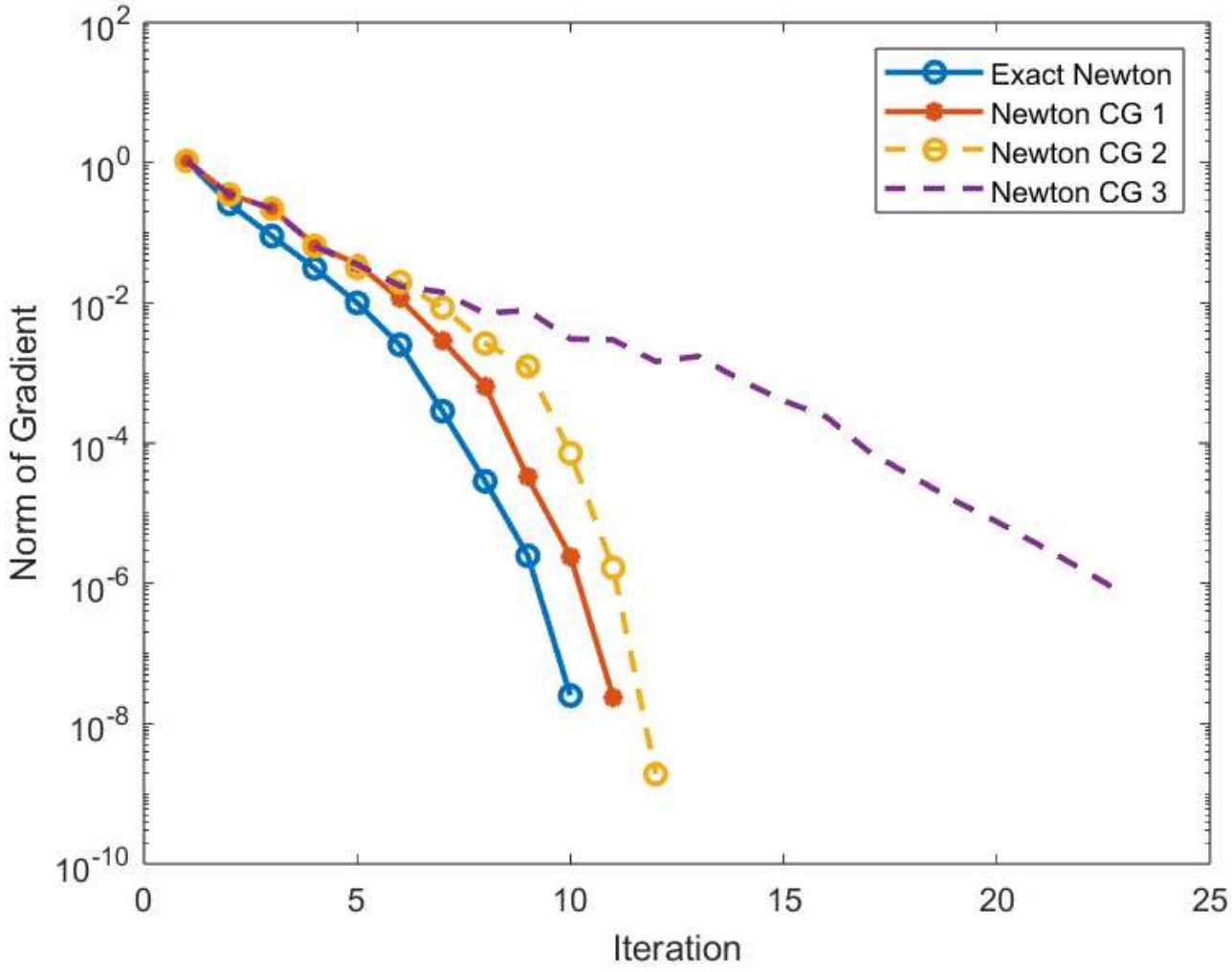
**第一小问：**

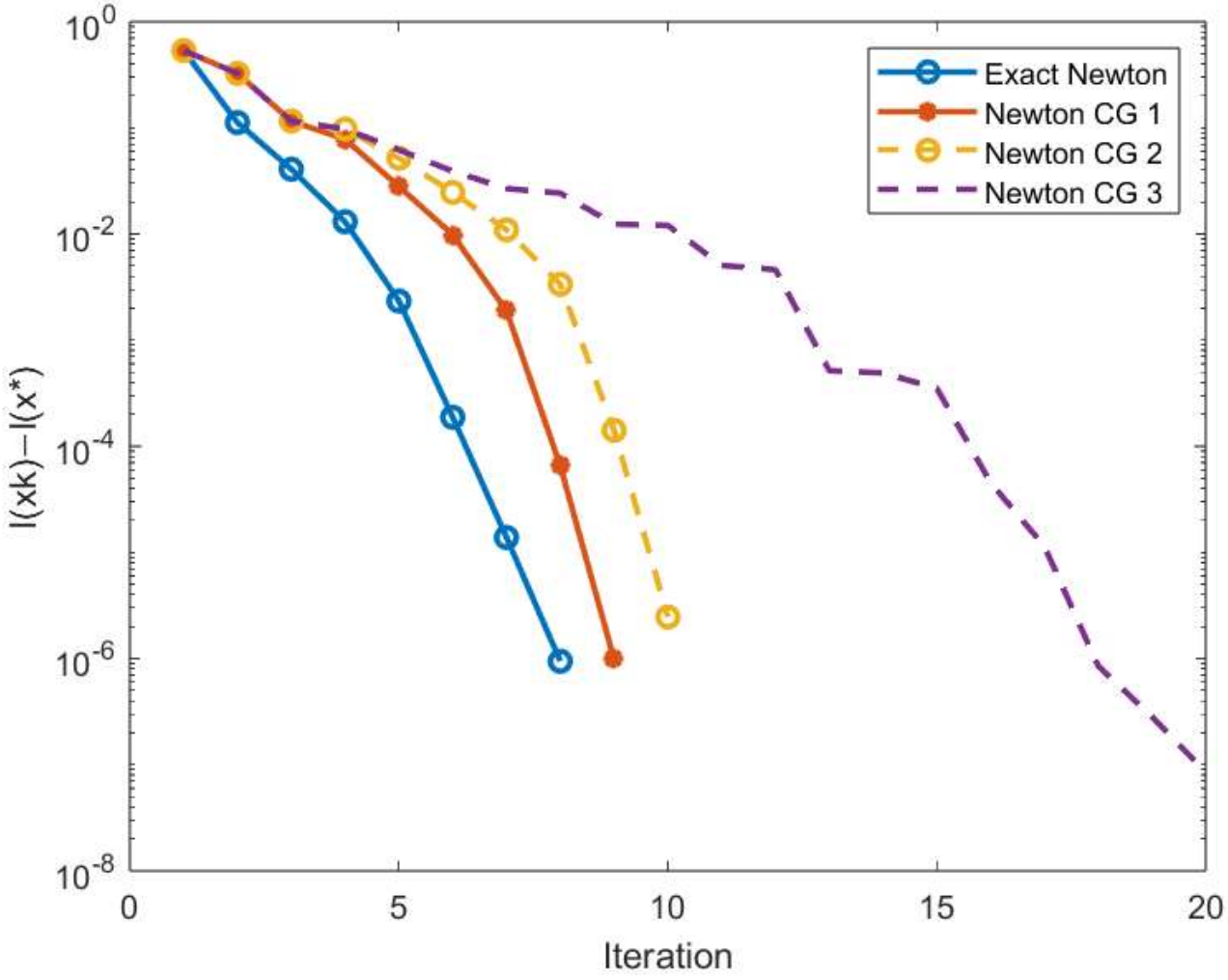
**dataset = 'a9a.test'**



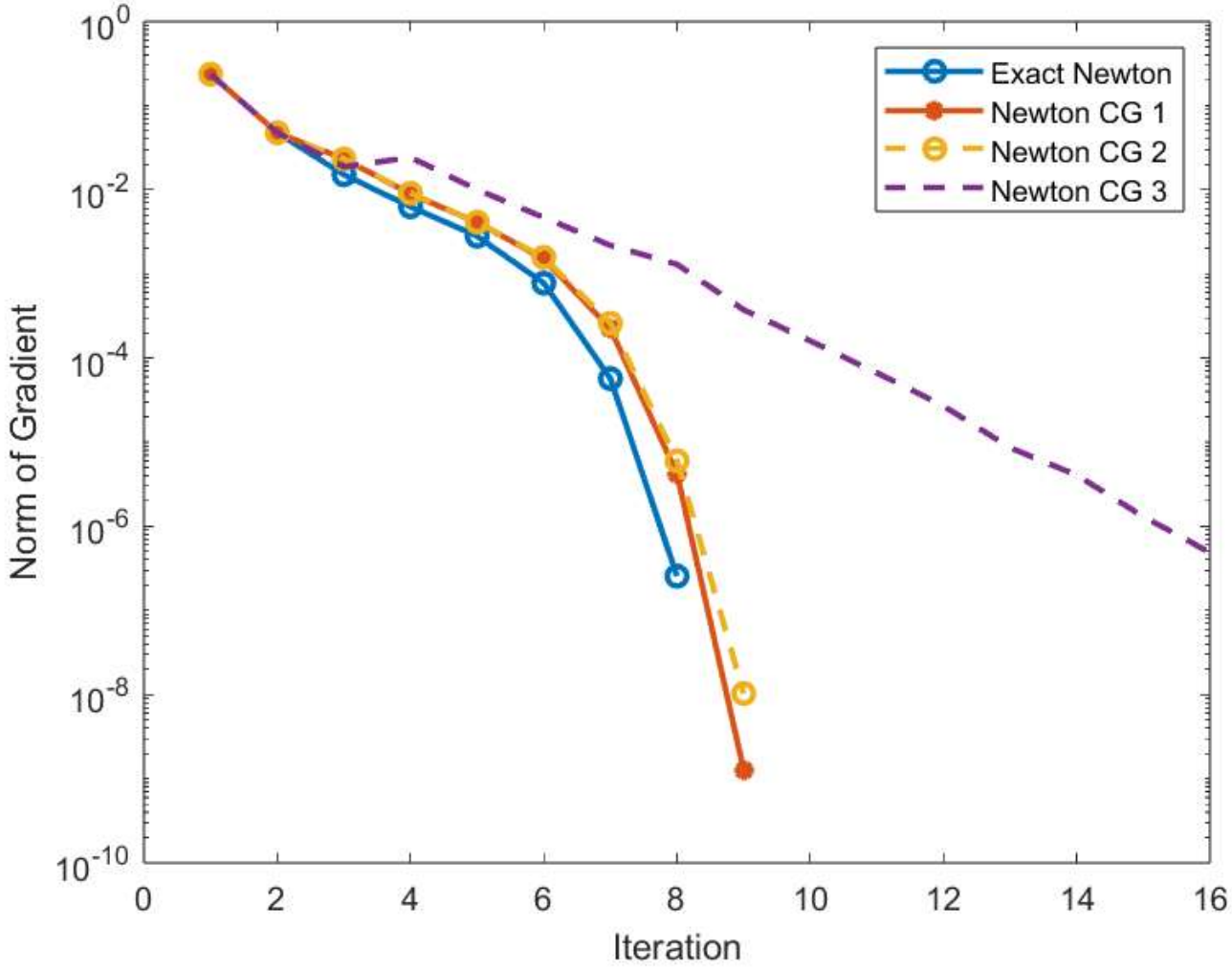


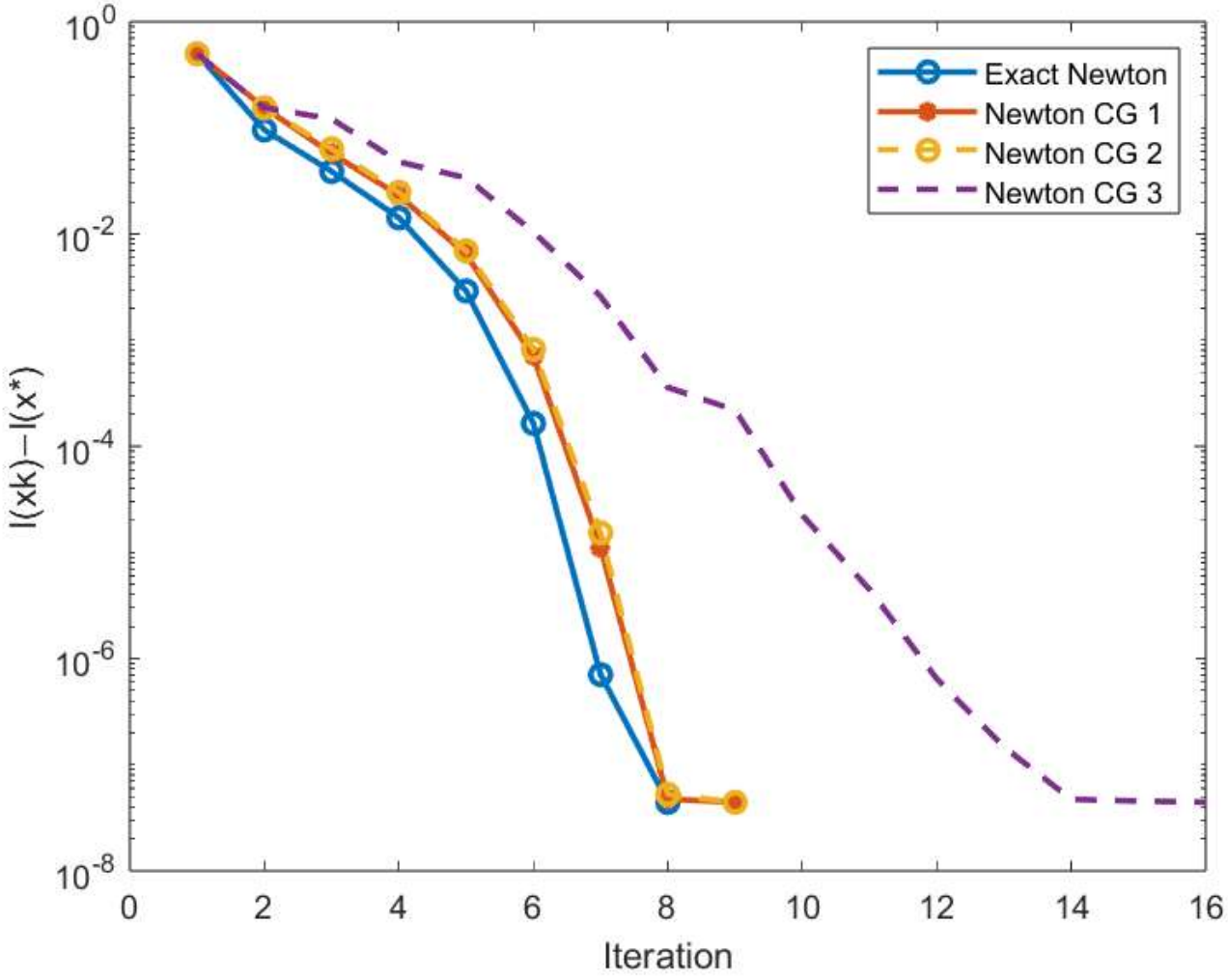

**dataset = 'CINA.test'**


Exact Newton
Newton CG 1
Newton CG 2
Newton CG 3
Norm of Gradient
Iteration
$10^{2}$
$10^{0}$
$10^{-2}$
$10^{-4}$
$10^{-6}$
$10^{-8}$
$10^{-10}$
0
5
10
15
20
25



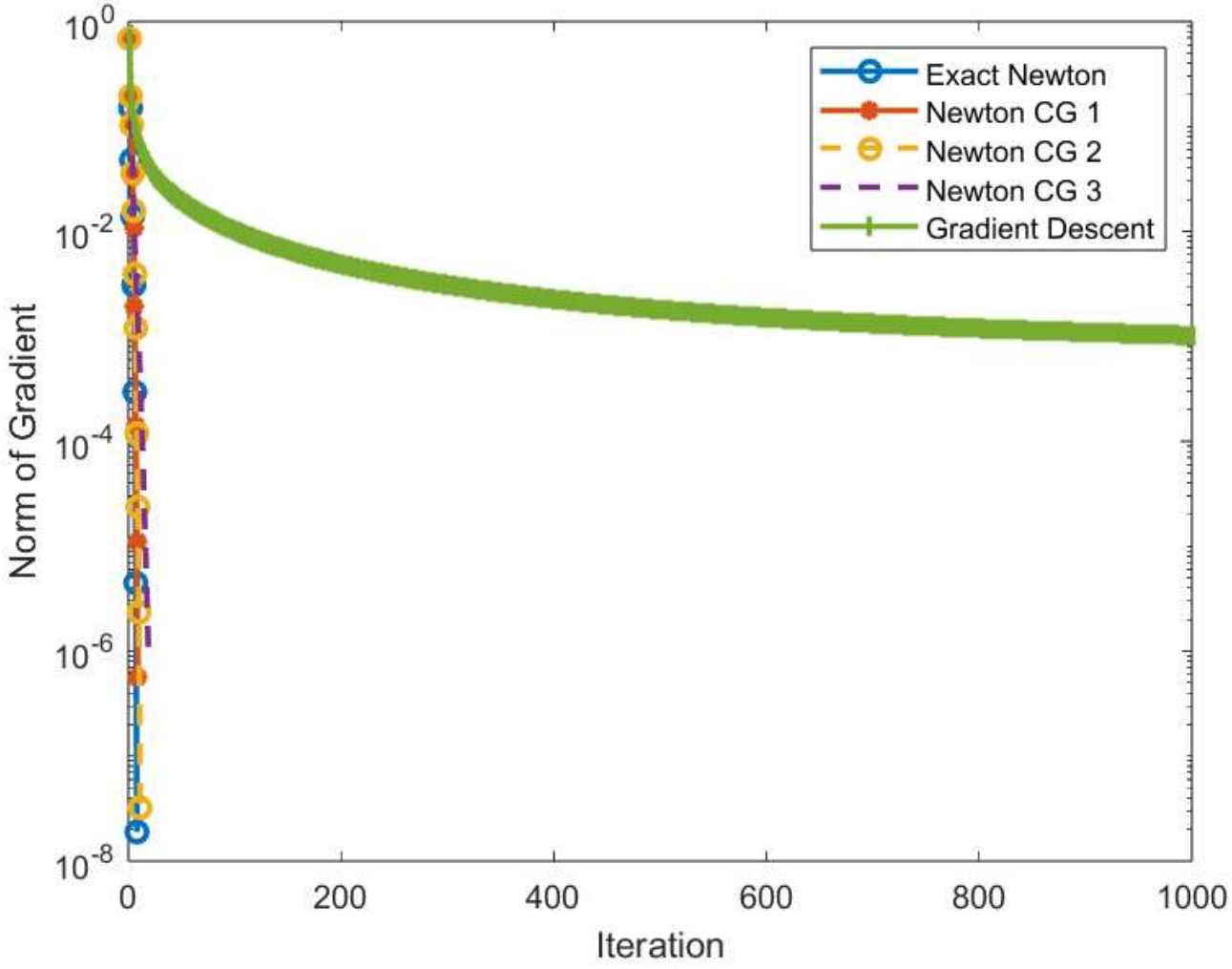

**dataset = 'ijcnn1.test'**

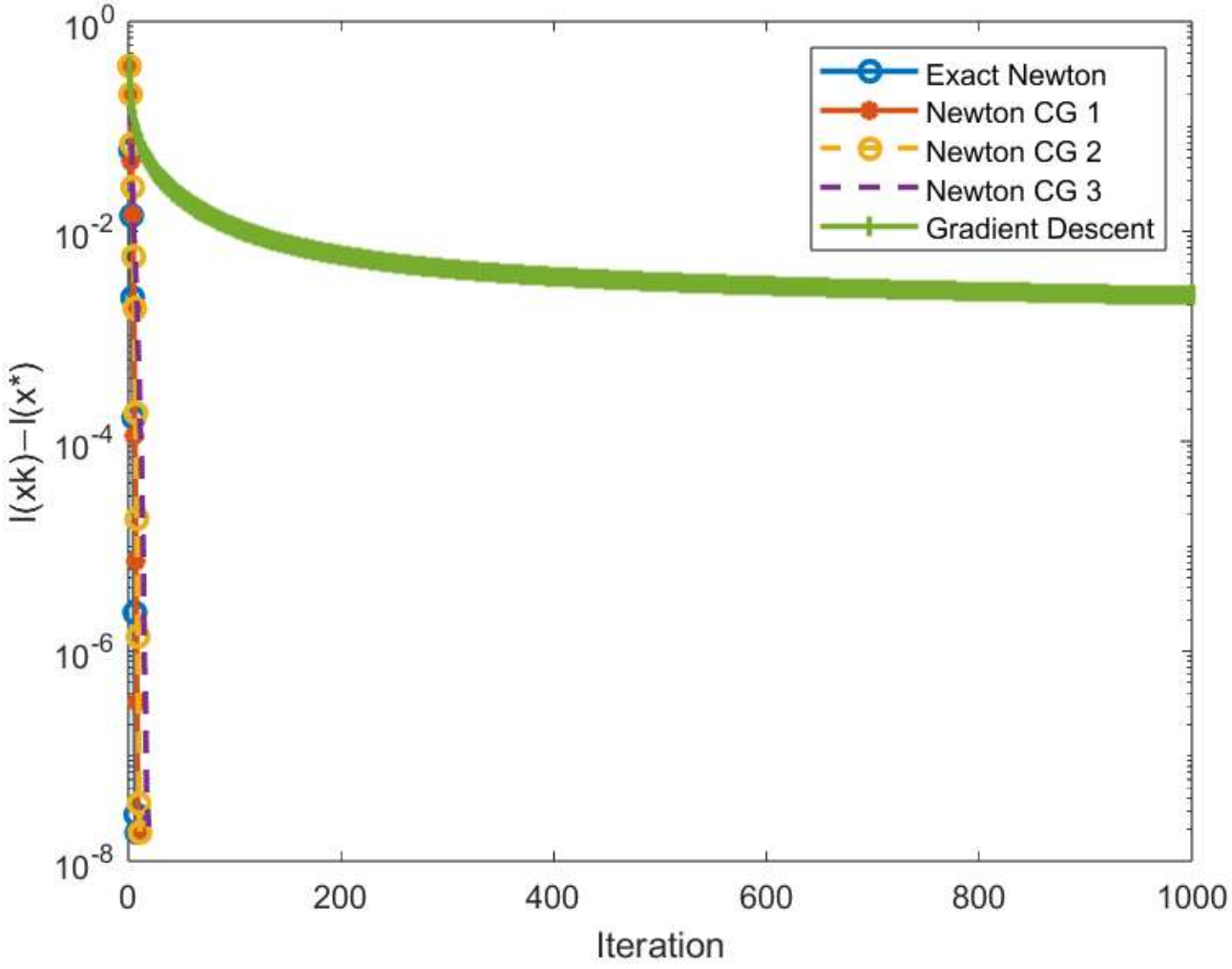

Exact Newton
Newton CG 1
Newton CG 2
Newton CG 3
Norm of Gradient
Iteration
$10^{0}$
$10^{-2}$
$10^{-4}$
$10^{-6}$
$10^{-8}$
$10^{-10}$
0
2
4
6
8
10
12
14
16



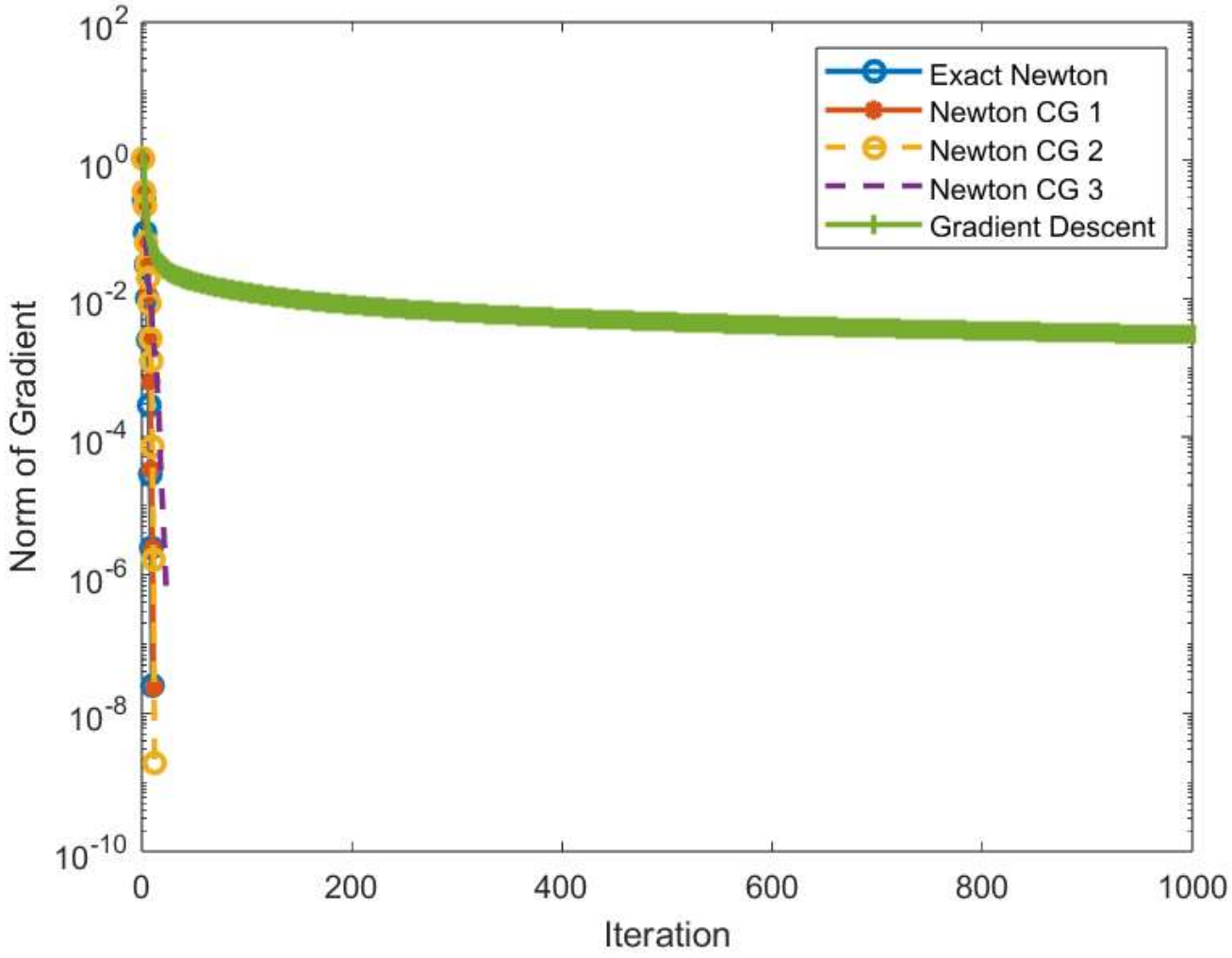

**第二小问：**

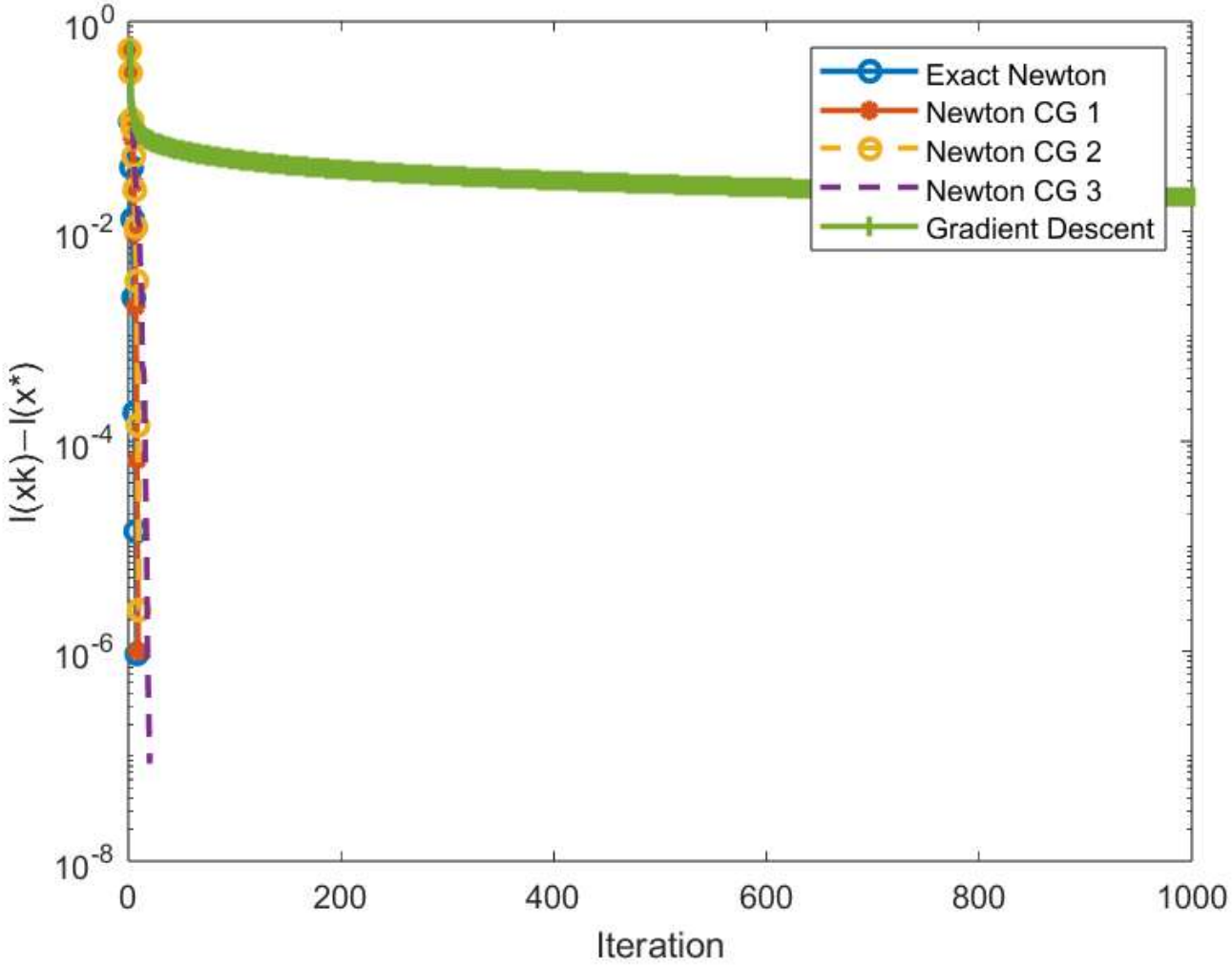
**dataset = 'a9a.test'**


Exact Newton
Newton CG 1
Newton CG 2
Newton CG 3
Gradient Descent
Norm of Gradient
Iteration
10^0
10^-2
10^-4
10^-6
10^-8
0
200
400
600
800
1000

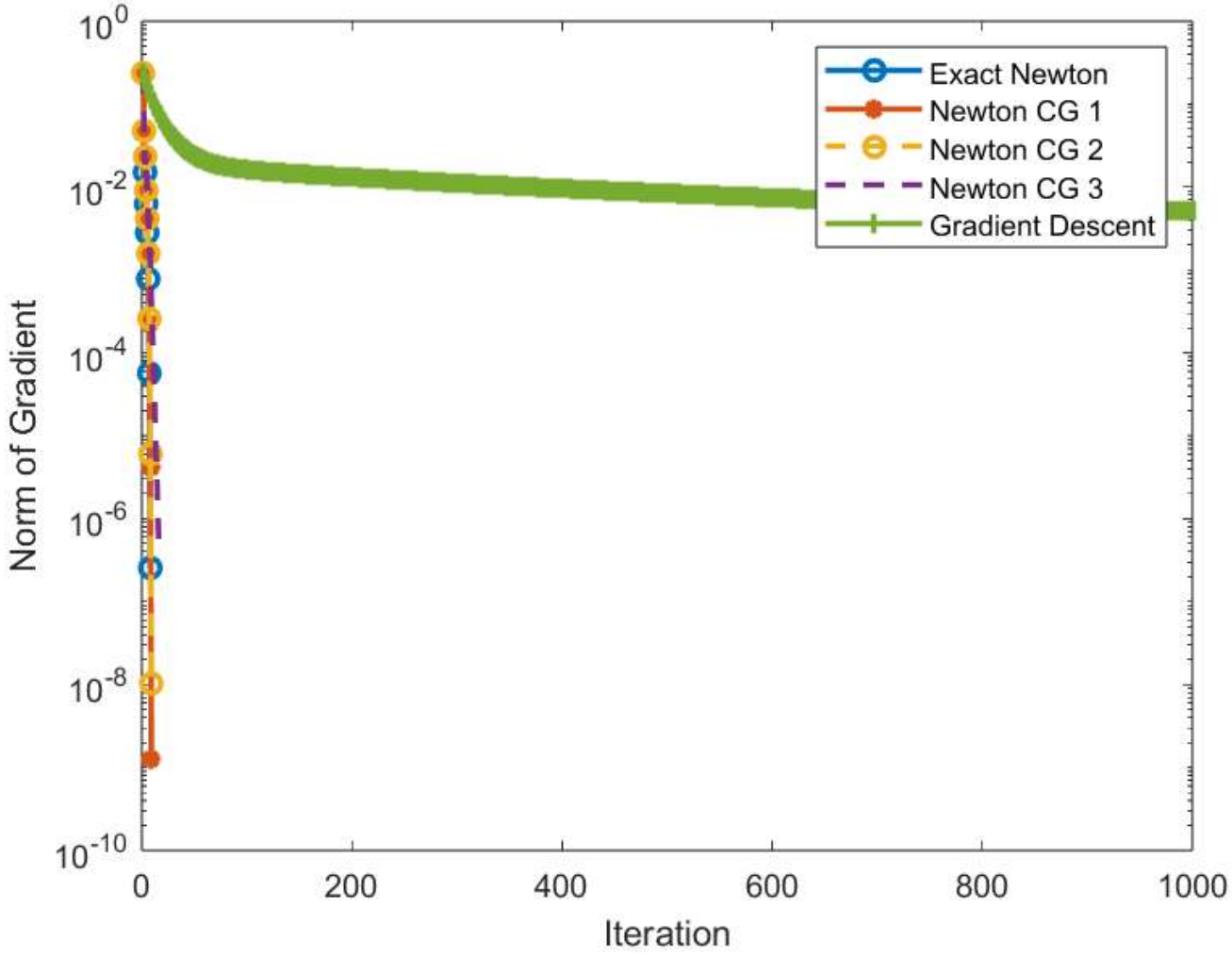


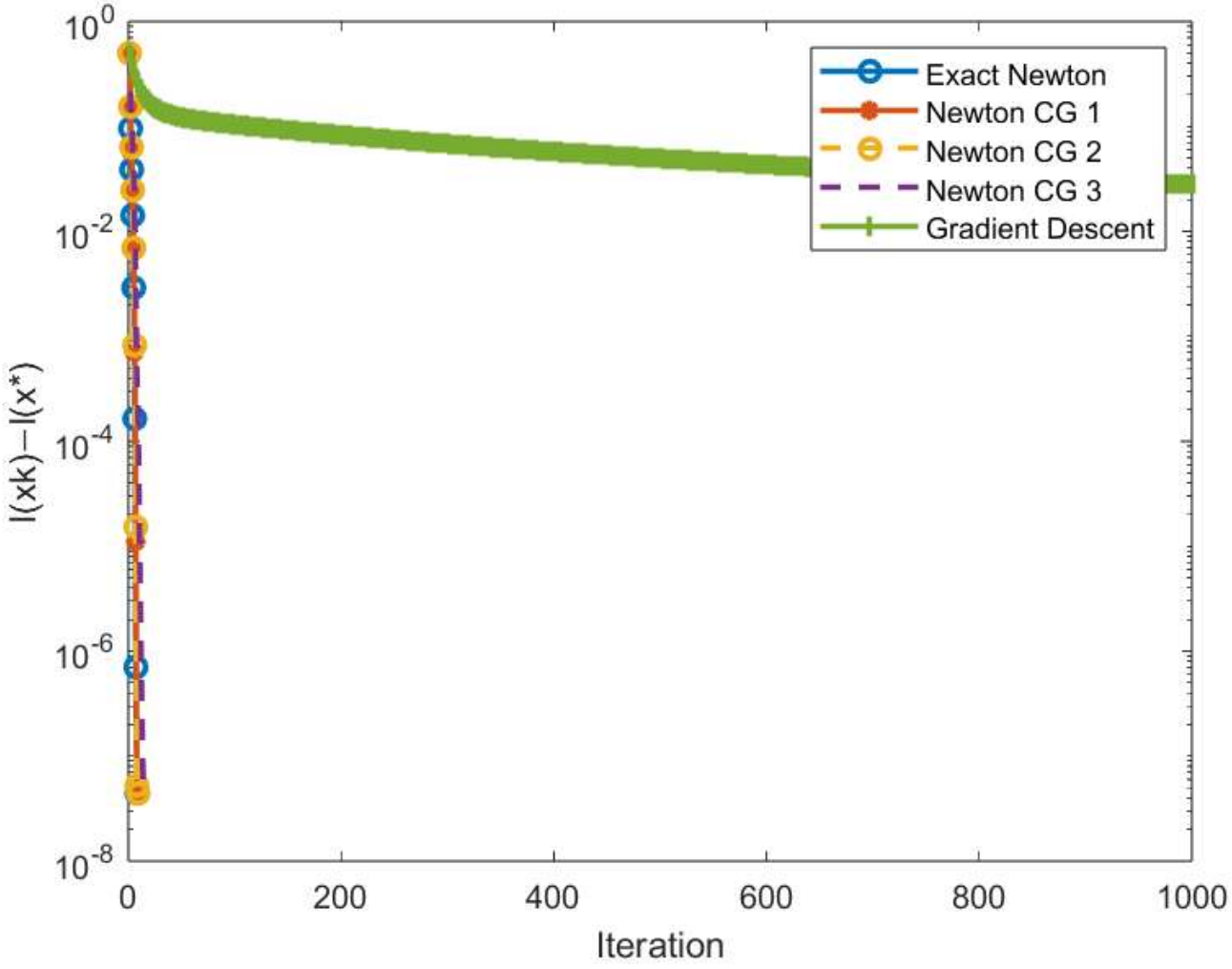

**dataset = 'CINA.test'**


Exact Newton
Newton CG 1
Newton CG 2
Newton CG 3
Gradient Descent
Norm of Gradient
Iteration
$10^{2}$
$10^{0}$
$10^{-2}$
$10^{-4}$
$10^{-6}$
$10^{-8}$
$10^{-10}$
0
200
400
600
800
1000




**dataset = 'ijcnn1.test'**


Exact Newton
Newton CG 1
Newton CG 2
Newton CG 3
Gradient Descent
Norm of Gradient
Iteration
$10^{0}$
$10^{-2}$
$10^{-4}$
$10^{-6}$
$10^{-8}$
$10^{-10}$
0
200
400
600
800
1000


Exact Newton
Newton CG 1
Newton CG 2
Newton CG 3
Gradient Descent
10^0
10^-2
10^-4
10^-6
10^-8
I(xk)−I(x*)
0
200
400
600
800
1000
Iteration