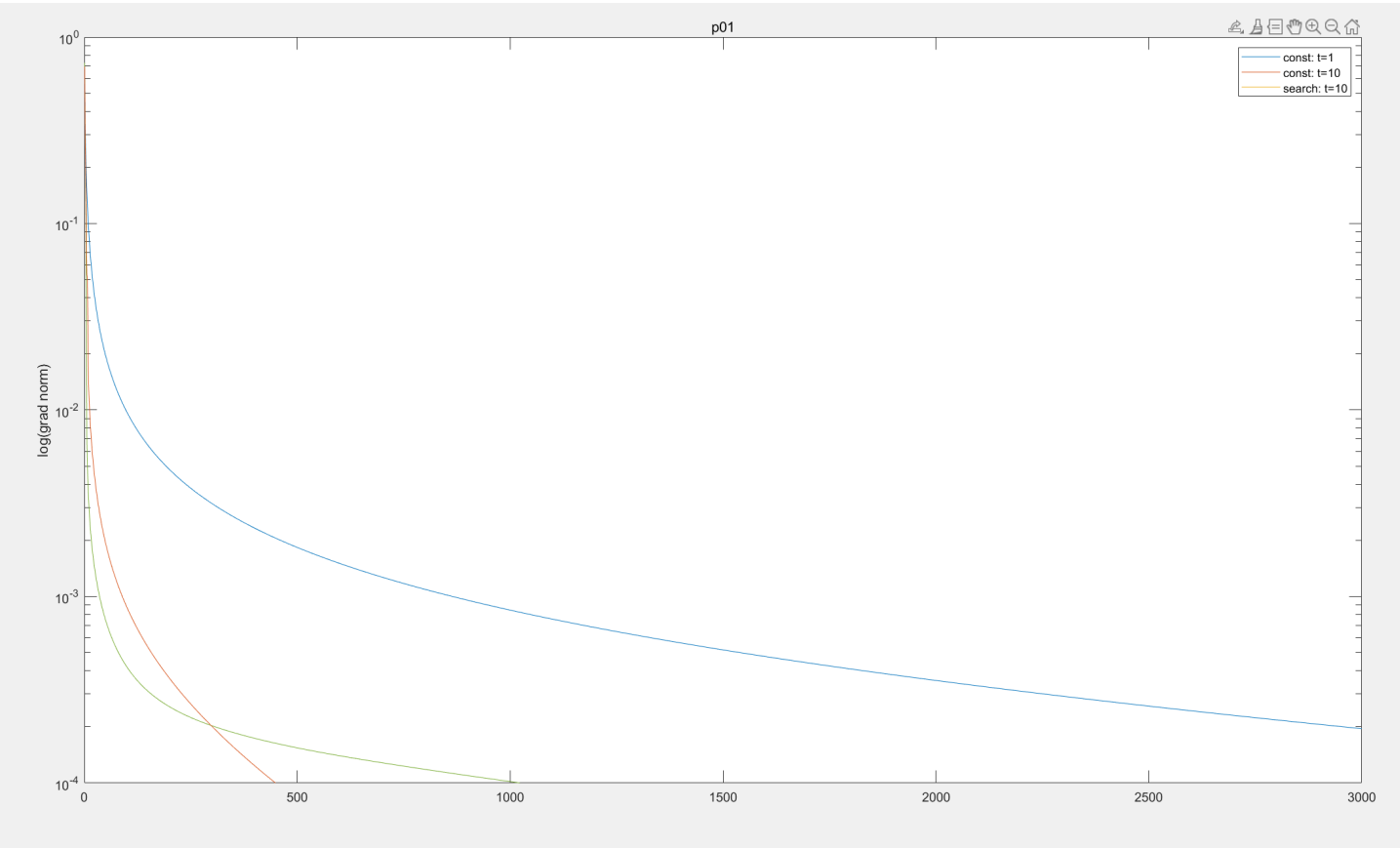
Assignment10

- **姓名：戴琪润**
- **学号：21307140003**
- **专业：人工智能（大数据班）**

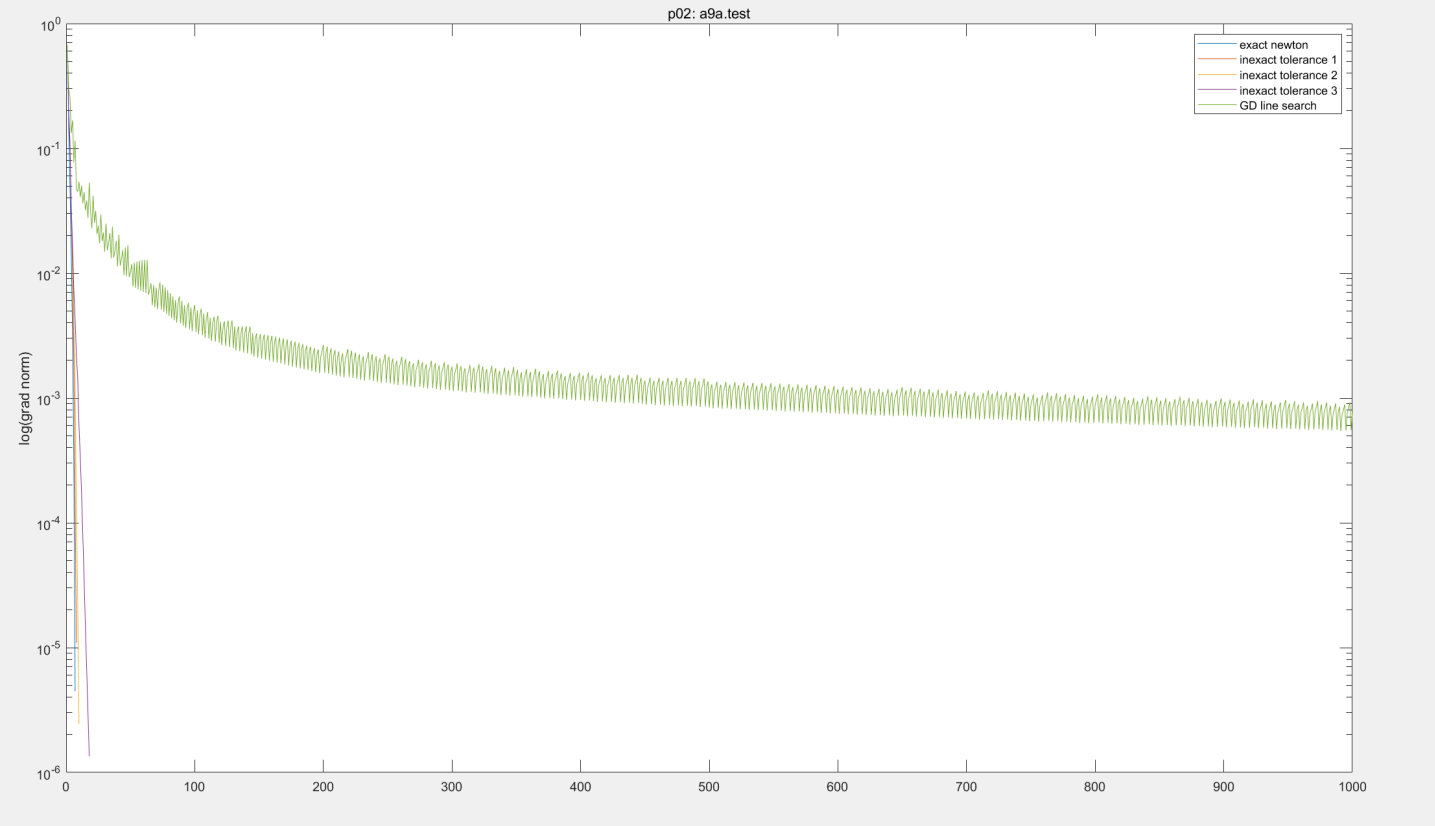
1. 问题1

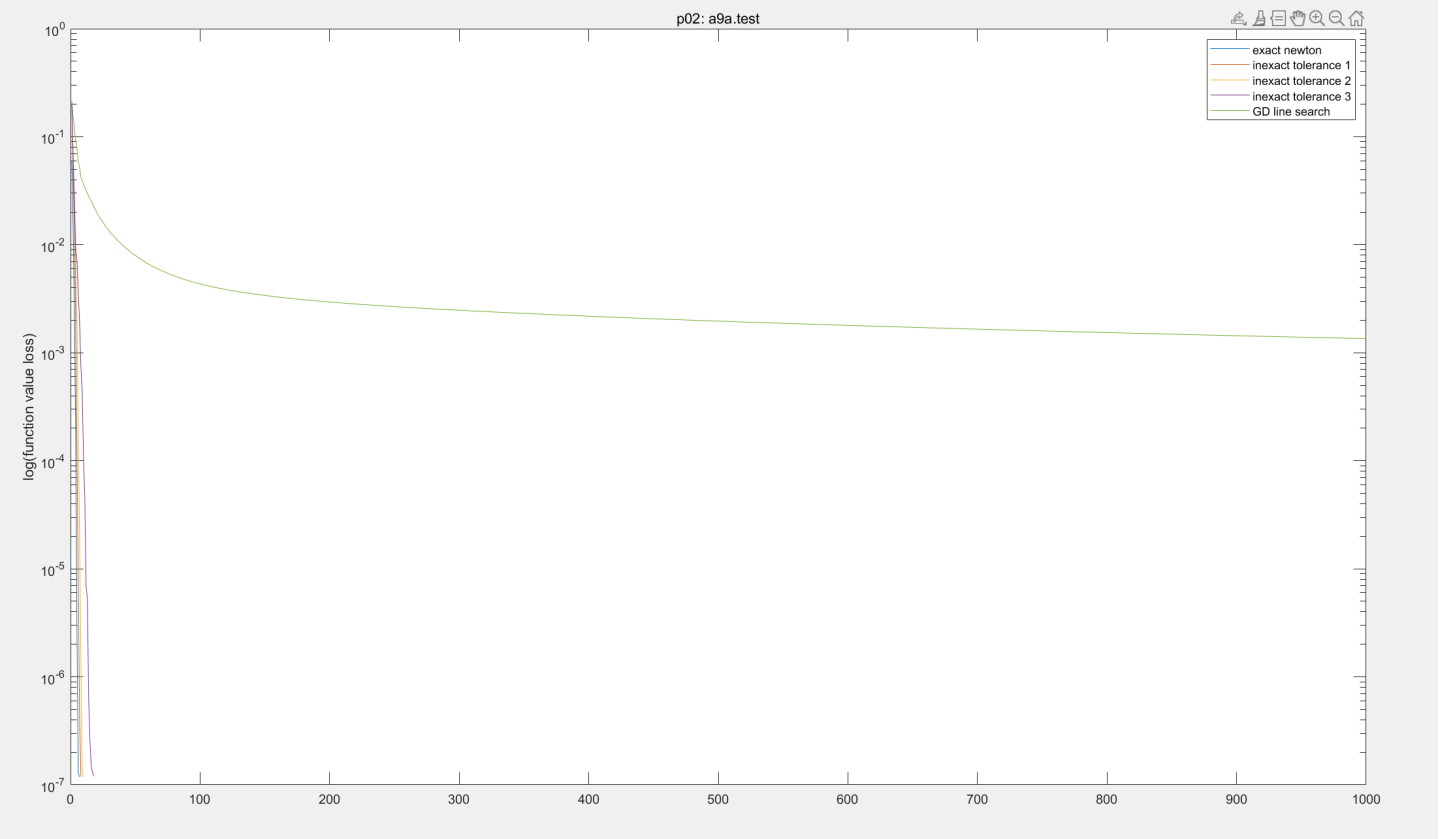
1. 当 const_step_size = 1, 有 Iteration = 3000; grad_norm = 1.799591e-04; 即算法迭代 3000 步后尚未收敛;
2. 当 const_step_size = 10, 有 Iteration = 904; grad_norm = 9.994044e-05; 即算法迭代 904 次后收敛;
3. 当 Armijo line search 采用 alpha=0.3, beta=0.6, t0=10.0, 有 Iteration = 425; grad_norm = 9.964404e-05; 即算法迭代 425 次后收敛;
4. **综上可得, 对于随机生成的数据, 当 const_step_size 变大, 收敛速率变快; 保持初始 step_size 相同的条件下, 采用 line_search, 能进一步加快收敛速率.**
5. 对 `grad_norm` 作 `semilogy` 图(共三条曲线)如下:



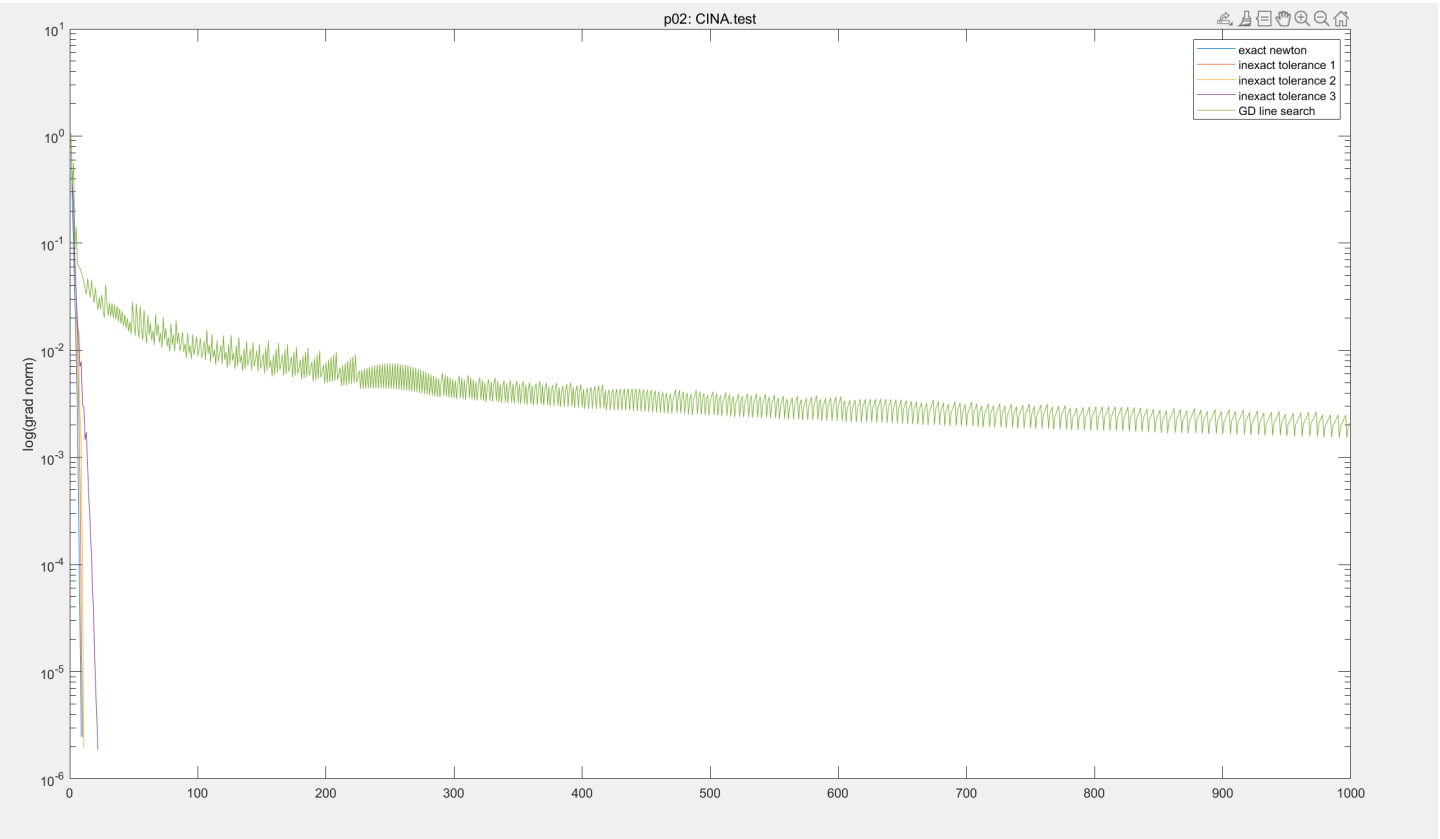
2. 问题2

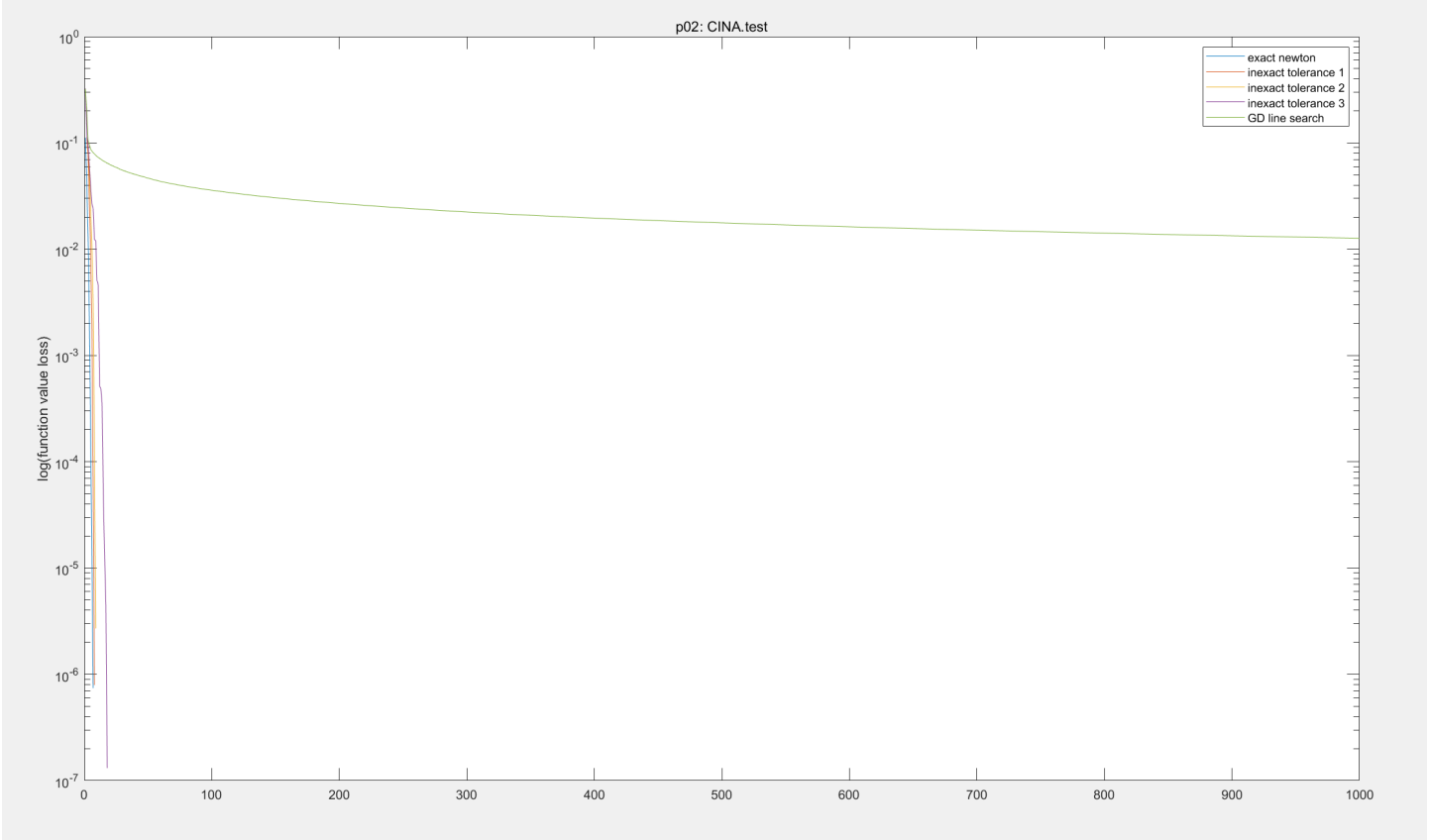
1. 对于 `newton` 和 `inexact_newton`, 不管哪个数据集, 参数都设置为: **alpha=0.3; beta=0.6; t0=1.0;**
2. 对于 `GD_line_search`, 对于三个数据集, **t0 均设置为 5;**
3. **在上述参数下, 解得: `a9a.test` 的最优值为 `0.318797`, `CINA.test` 的最优值为 `0.159307`, `ijcnn1.test` 的最优值为 `0.195658`;**
4. `a9a.test` 的 `grad_norm` 和 `function_loss` 作 `semilogy` 图如下(共 5 条曲线):



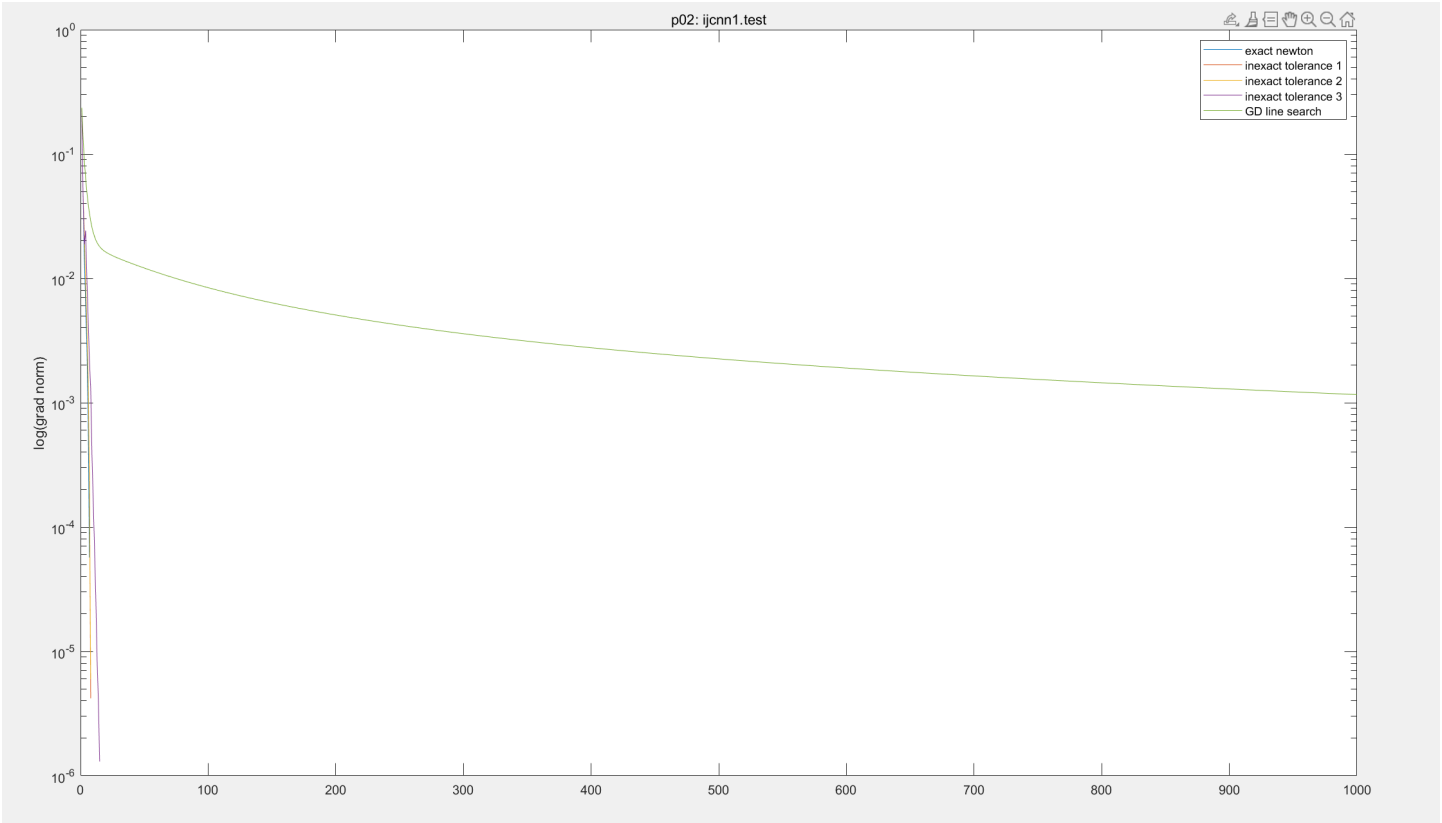


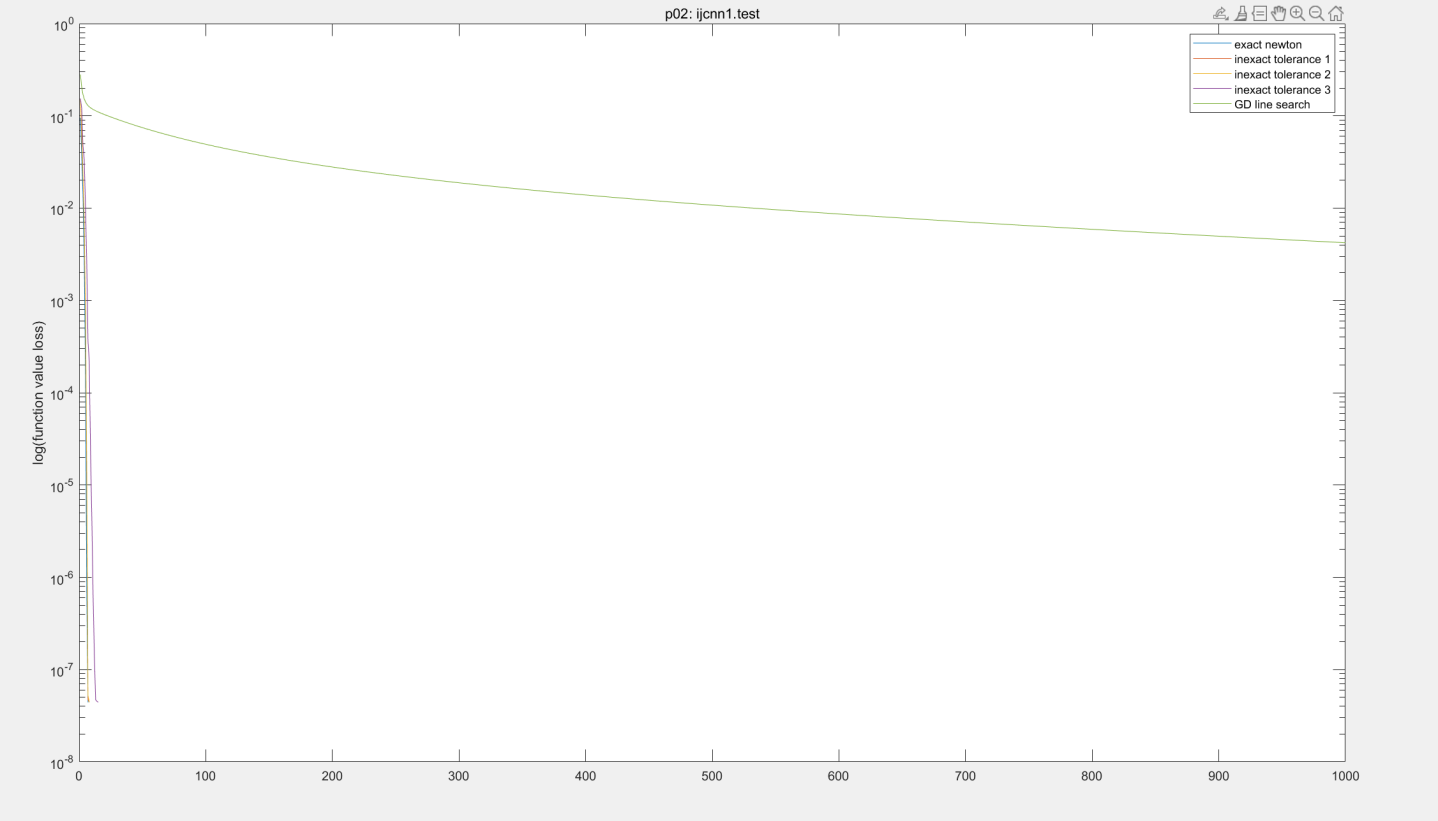
5. `CINA.test` 的 `grad_norm` 和 `function_loss` 作 `semilogy` 图如下(共 5 条曲线):



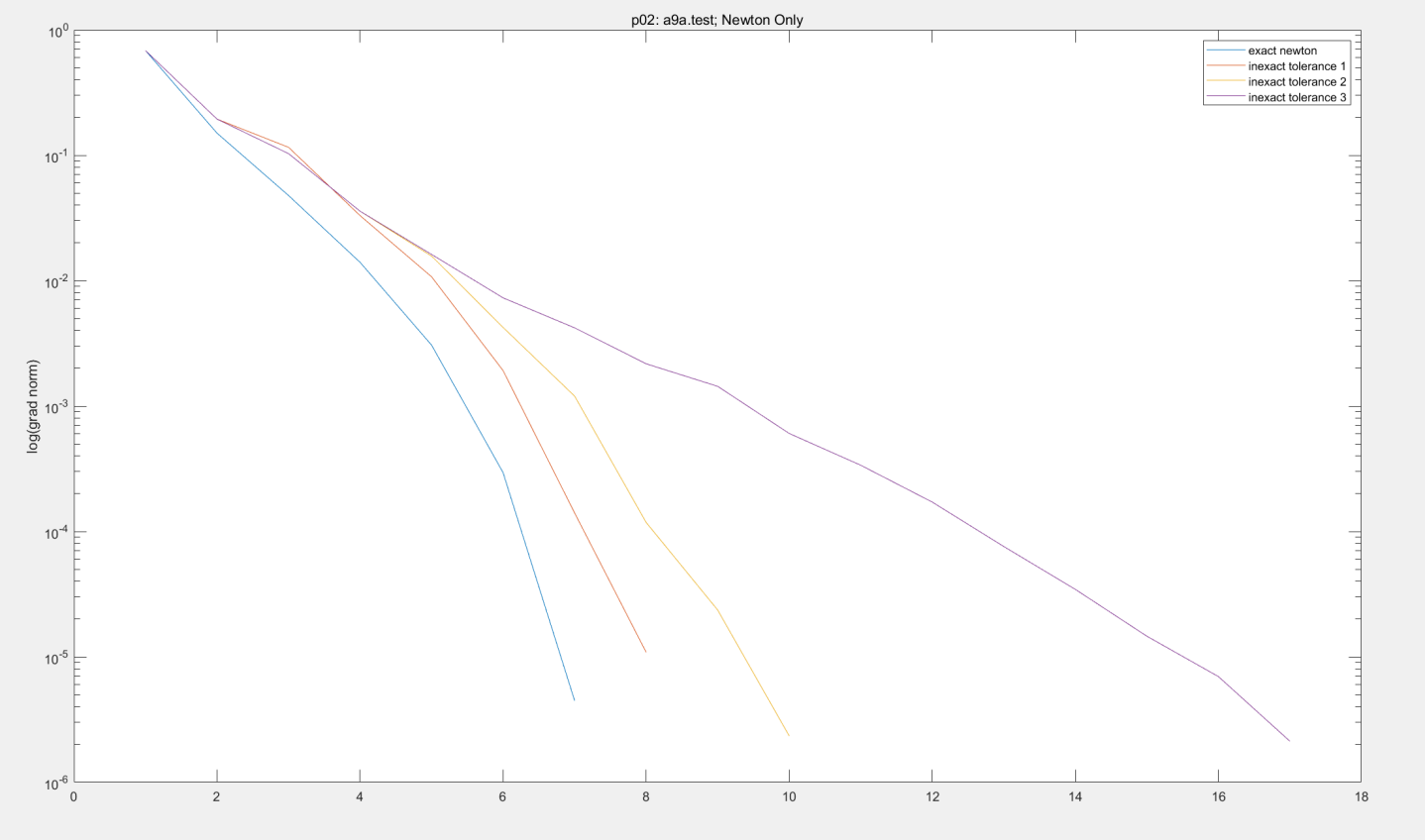


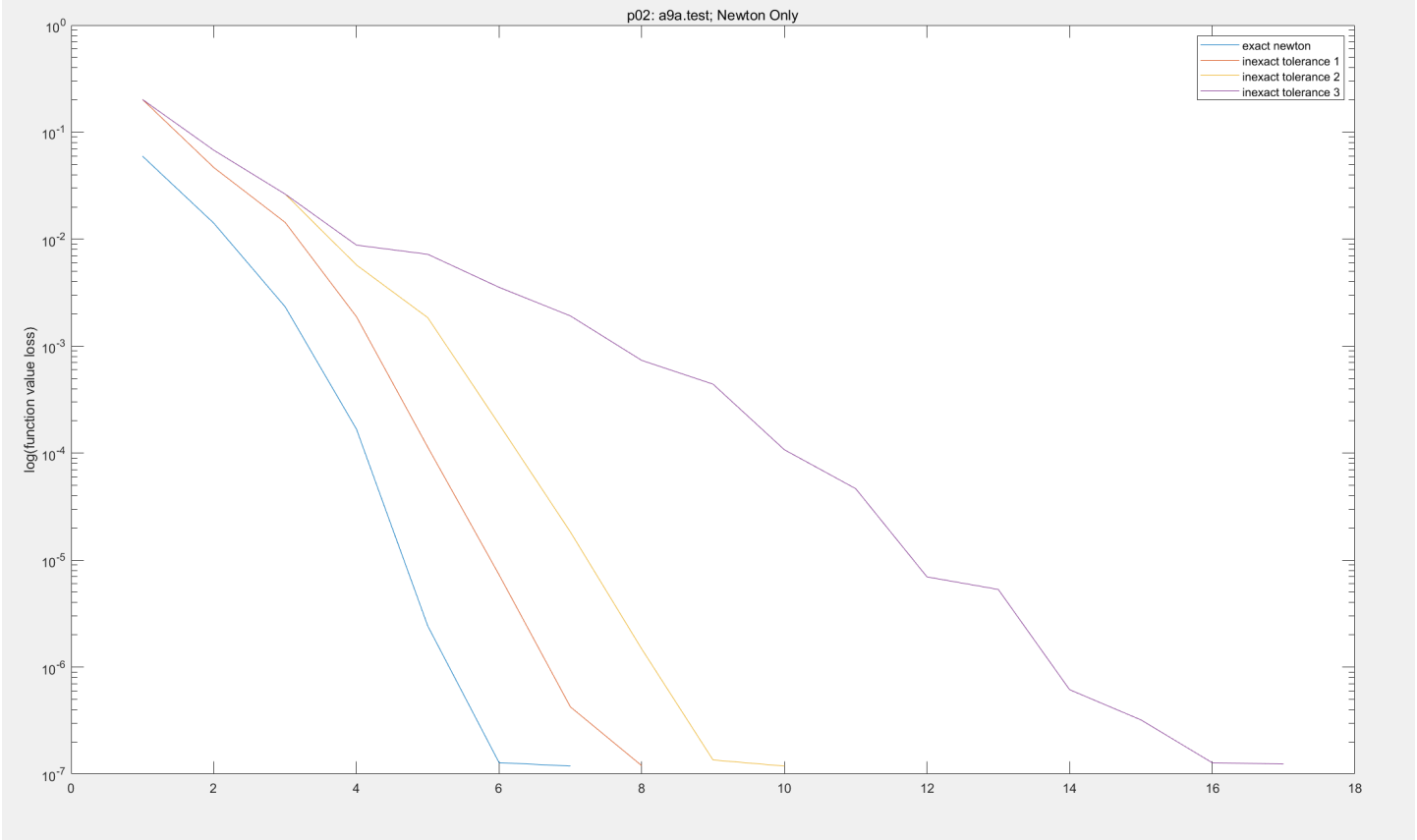
6. `ijcnn1.test` 的 `grad_norm` 和 `function_loss` 作 `semilogy` 图如下(共 5 条曲线):

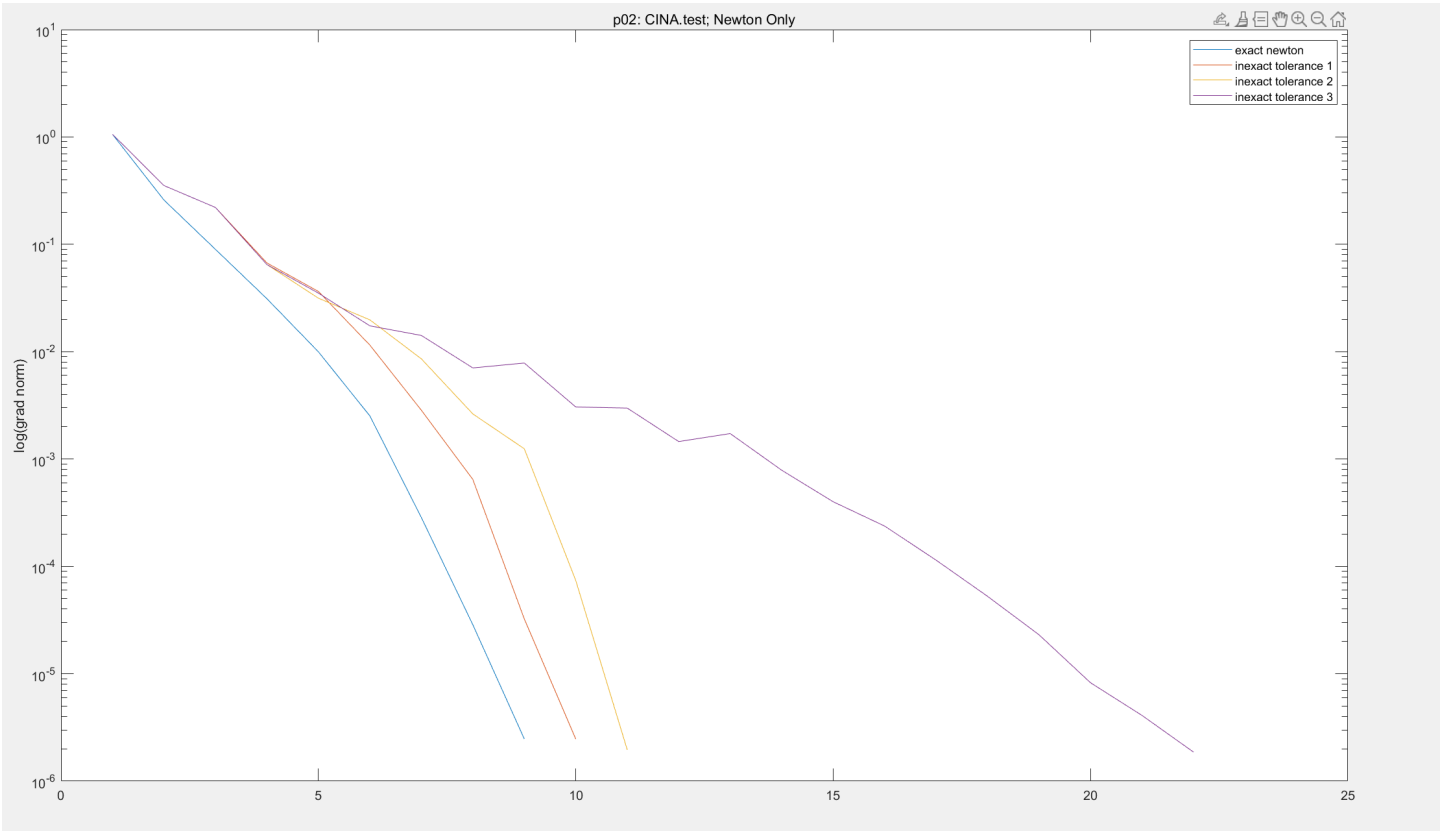


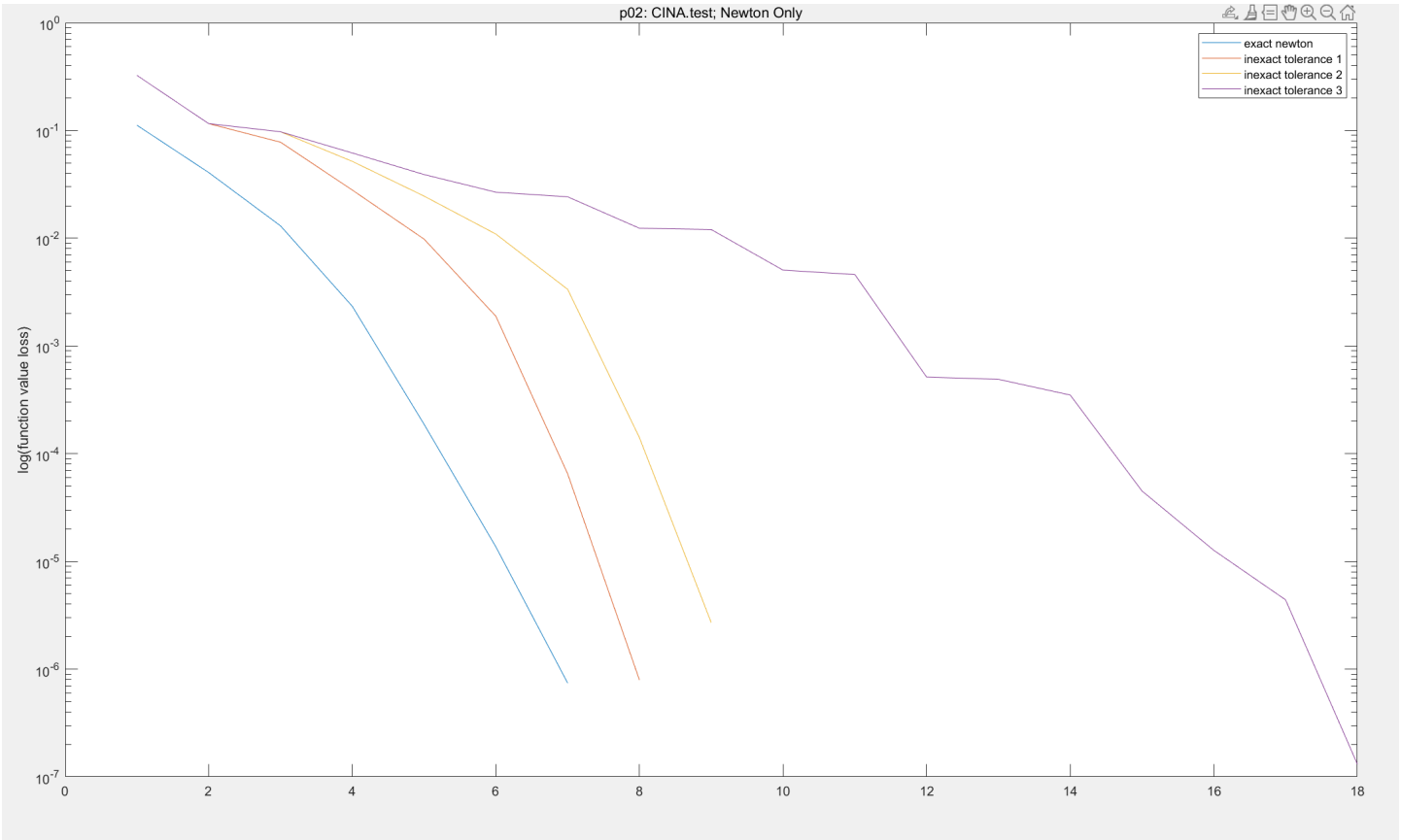


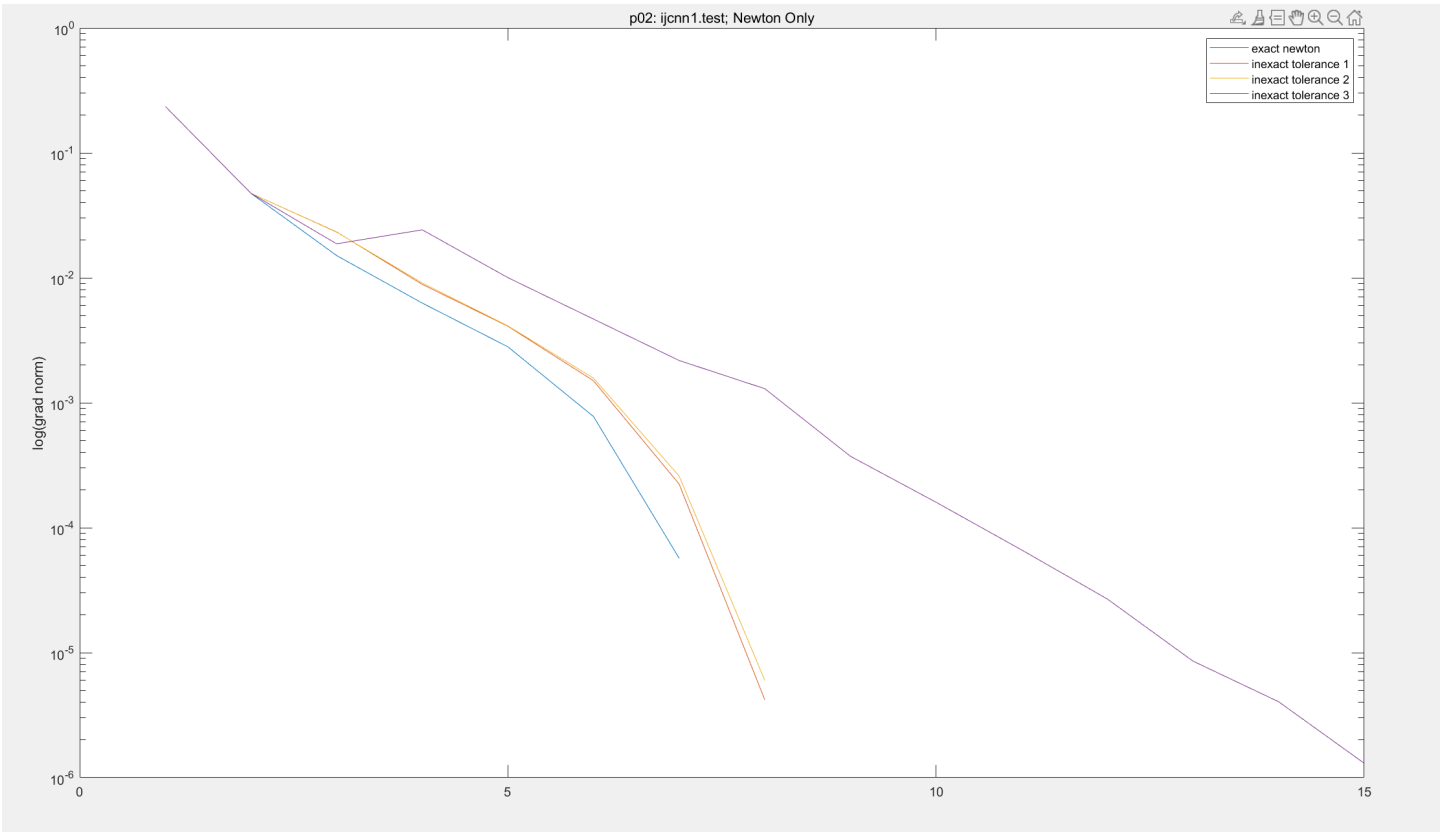
5. 比较下来可以发现, GD_line_search 的收敛速率显著慢于 Newton/inexact_newton, 且 a9a 和 CINA 两个数据集上, `grad norm` 在 GD_line_search 的过程中有持续的小幅震荡.
6. **进一步比较可以发现, exact_newton 的收敛速率最快, inexact_newton 随着 tolerance 的逐渐放松**(由 tolerance1 -> tolerance 3), **收敛速率逐渐变慢**, 但变慢程度不大, 额外需要的更多步数在5步以内. 具体图像如以下6张图(每个数据集各两张).

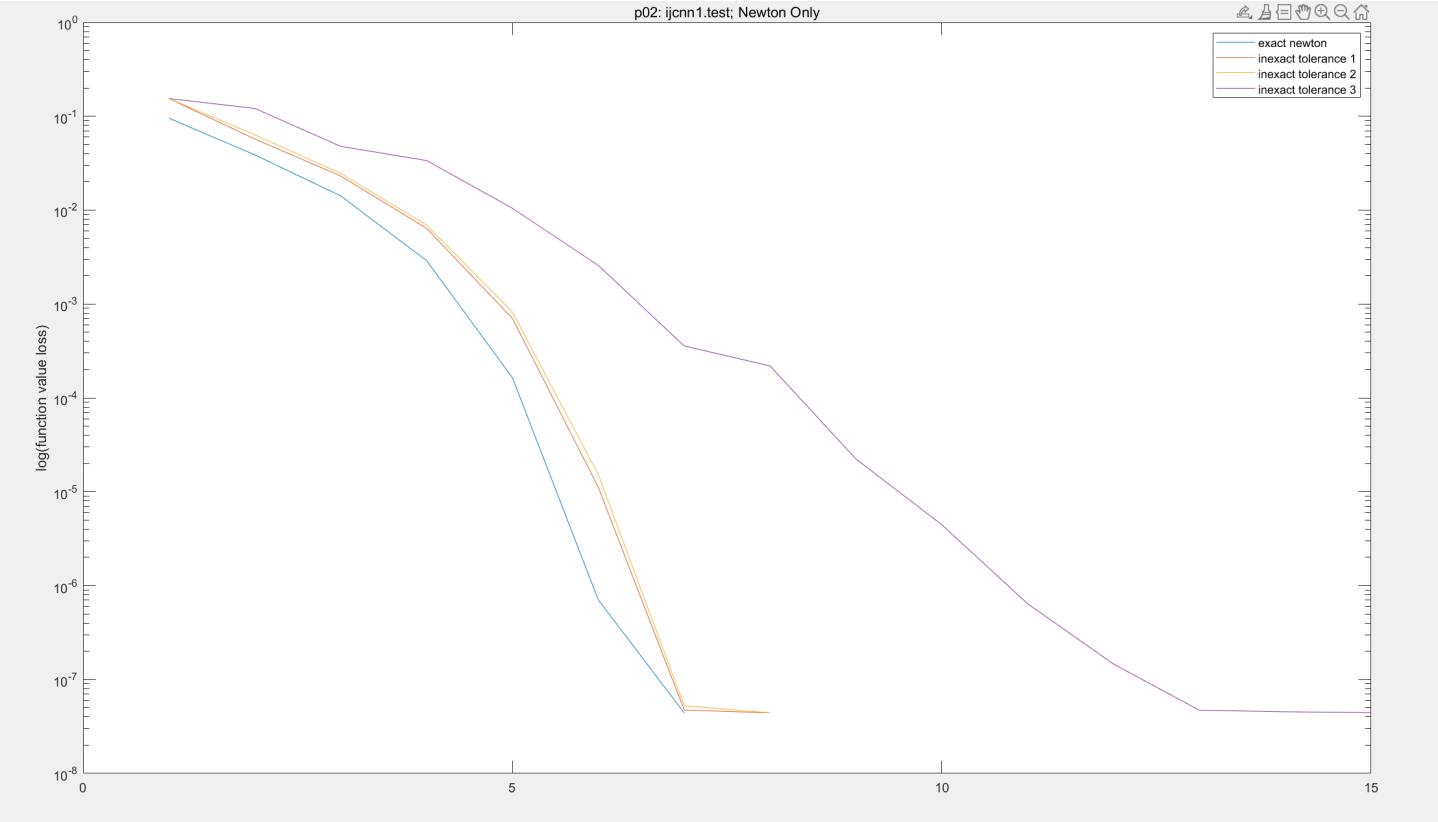



p02: a9a.test; Newton Only
exact newton
inexact tolerance 1
inexact tolerance 2
inexact tolerance 3
log(function value loss)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
2
4
6
8
10
12
14
16
18


p02: CINA.test; Newton Only
exact newton
inexact tolerance 1
inexact tolerance 2
inexact tolerance 3
log(grad norm)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
5
10
15
20
25


p02: CINA.test; Newton Only
exact newton
inexact tolerance 1
inexact tolerance 2
inexact tolerance 3
log(function value loss)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
2
4
6
8
10
12
14
16
18


p02: ijcnn1.test; Newton Only
exact newton
inexact tolerance 1
inexact tolerance 2
inexact tolerance 3
log(grad norm)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
5
10
15



3. 源代码

1. **工具函数**合集:
   - `gradient.m`; `Hessian.m`; `l_value.m`;
   - `backtracking.m`; `backtracking_with_input_d.m`;
   - `N_direction.m`;

```
% gradient.m
function [y] = gradient(m, n, A, b, x)
p = 1 ./ (1 + exp(-b.*(A*x)));
e = ones(m,1);
y = -A' * (b.*(e-p))/m + 1/(50*m)*x;

% Hessian.m
function H=Hessian(m,n,A,b,x)
e = ones(m,1);
p = 1 ./ (1+exp(-b.*(A*x)));
I = eye(n,n);
D_P = spdiags(p,[0],m,m);
% D_P = diag(p,0);
D_e_p = spdiags(e-p,[0],m,m);
% D_e_p = diag(e-p,0);
H= 1/m * A'*(D_P * D_e_p)*A + 1/(50*m)*I;
```

```
17
18 % l_value.m
19 function y = l_value(m,n,A,b,x)
20 p=1./(1+exp(-b.*(A*x)));
21 e = ones(m,1);
22 y = -e'*log(p)/m + 1/(100*m)*(norm(x,2)^2);
23
24 % backtracking.m
25 function t = backtracking(grad,m,n,A,b,x,t0,alpha,beta)
26 t = t0;
27 while l_value(m,n,A,b,x) - l_value(m,n,A,b,x-t*grad) ...
28     < alpha*t*(norm(grad,2)^2)
29     t = beta * t;
30 end
31
32 % backtracking_with_input_d.m
33 function t = backtracking_with_input_d(...
34             grad,d,m,n,A,b,x,t0,alpha,beta)
35 t = t0;
36 while l_value(m,n,A,b,x+t*d) - l_value(m,n,A,b,x) ...
37     > alpha*t*grad'*d
38     t = beta * t;
39 end
40
41 % N_direction.m
42 function d=Ndirection(m,n,A,b,g,x)
43 e = ones(m,1);
44 p = 1 ./ (1+exp(-b.*(A*x)));
45 I = eye(n,n);
46 D1 = spdiags(p,[0],m,m);
47 D2 = spdiags(e-p,[0],m,m);
48 H = 1/m * A'*(D1*D2)*A + 1/(50*m)*I;
49 d = -inv(H)*g;
50
```

2. **问题1** 的源代码

```
1 % GD_const.m
2 function grad_norm = GD_const(t)
3 fprintf("GD_const: t = %.2f\n",t)
4 rand('seed', 21307140003);
5 m = 500; n = 1000;
6 A = randn(m,n); b = sign(rand(m,1)-0.5);
7
8 x = zeros(n,1);
```

```
grad_norm = zeros(3000);

for i=1:3000
    grad = gradient(m,n,A,b,x);
    x = x-t*grad;
    grad_norm(i) = norm(grad,2);
    fprintf("Iteration = %d; grad_norm = %.6e;\n", ...
            i,grad_norm(i));
    if grad_norm(i) < 1e-4
        break
    end
end

fprintf("GD_const_t=%.2f; Num_Iter = %d\n", t, i);


% GD_linesearch.m
function grad_norm = GD_linesearch(alpha, beta, t0)
fprintf("GD_line_search: alpha=%.1f, beta=%.1f, t0=%.1f\n",...
        alpha, beta, t0)
rand('seed', 21307140003);
m=500; n=1000;
A=randn(m,n); b=sign(rand(m,1)-0.5);

x = zeros(n,1);
grad_norm = zeros(3000);

for i = 1:3000
    grad = gradient(m,n,A,b,x);
    ng = norm(grad,2);
    t = backtracking(grad,m,n,A,b,x,t0,alpha,beta);
    x = x - t*grad;
    grad_norm(i) = ng;
    fprintf("Iteration = %d; grad_norm = %.6e;\n",...
            i, grad_norm(i));
    if grad_norm(i) < 1e-4
        break
    end
end

fprintf("GD_line_search_t=%.2f; Num_Iter = %d\n", t, i);


% main_1.m
grad_norm_1 = GD_const(1);
semilogy(grad_norm_1);
hold on;
```

```

grad_norm_2 = GD_const(10);
semilogy(grad_norm_2);
hold on;

grad_norm_3 = GD_linesearch(0.3, 0.6, 10);
semilogy(grad_norm_3);
hold off;

title('p01')
ylabel('log(grad norm)')
legend('const: t=1', 'const: t=10', 'search: t=10');
```

3. **问题2** 的源代码

```
% newton.m
function [grad_norm, l_value_loss] = newton(dataset, alpha, beta, t0)
fprintf("exact_newton: alpha=%.1f, beta=%.1f, t0=%.1f\n",...
        alpha, beta, t0)
[b,A] = libsvmread(dataset);
[m,n] = size(A);
mu = 1e-2/m;

x = zeros(n,1);
grad_norm = [];
l_value_loss = [];

grad=gradient(m,n,A,b,x);
ng = norm(grad,2);
while ng >= 1e-6
    d = Ndirection(m,n,A,b,grad,x);
    t = backtracking_with_input_d(grad,d,m,n,A,b,x,t0,alpha,beta);
    x = x + t*d;
    grad_norm = [grad_norm,ng];
    l_value_loss = [l_value_loss, l_value(m,n,A,b,x)];
    grad = gradient(m,n,A,b,x);
    ng = norm(grad, 2);
end
l_value_loss = l_value_loss - 0.195658;
fprintf("l(x*) = %.6f\n", l_value(m,n,A,b,x))


% inexact_newton.m
function [grad_norm, l_value_loss] = inexact_newton(...
                dataset, alpha, beta, t0, tolerance)
```

```
fprintf("inexact_newton: alpha=%.1f, beta=%.1f, t0=%.1f\n",...
        alpha, beta, t0)
[b,A] = libsvmread(dataset);
[m,n] = size(A);

x = zeros(n,1);
grad_norm = [];
l_value_loss = [];

grad = gradient(m,n,A,b,x);
ng = norm(grad, 2);
while ng >= 1e-6
    H = Hessian(m,n,A,b,x);
    d = myCG(H, grad, ng, 1000, tolerance);
    t = backtracking_with_input_d(grad,d,m,n,A,b,x,t0,alpha,beta);
    x = x + t*d;
    grad_norm=[grad_norm,ng];
    l_value_loss = [l_value_loss, l_value(m,n,A,b,x)];
    grad = gradient(m,n,A,b,x);
    ng = norm(grad, 2);
end
l_value_loss = l_value_loss - 0.195658;


% GD_linesearch_2.m
function [grad_norm, l_value_loss] = GD_linesearch_2(...
                            dataset, alpha, beta, t0)
fprintf("GD_line_search: alpha=%.1f, beta=%.1f, t0=%.1f\n",...
        alpha, beta, t0)

[b,A] = libsvmread(dataset);
[m,n] = size(A);
mu = 1e-2/m;

x=zeros(n,1);
grad_norm = zeros(1000);
l_value_loss = zeros(1000);

for i = 1:1000
    grad = gradient(m,n,A,b,x);
    ng = norm(grad,2);
    t = backtracking(grad,m,n,A,b,x,t0,alpha,beta);
    x = x - t*grad;
    grad_norm(i) = ng;
    l_value_loss(i) = l_value(m,n,A,b,x);
    fprintf("Iteration = %d; grad_norm = %.6e;\n",...
            i, grad_norm(i));
```

```
    if grad_norm(i) < 1e-4
        break
    end
end
l_value_loss = max(l_value_loss - 0.195658, zeros(1000));
fprintf("GD_line_search_t=%.2f; Num_Iter = %d\n", t, i);


% main_2.m
dataset = 'ijcnn1.test';
% a9a: l(x*) = 0.318797; GD_linesearch_t0 = 5
% CINA: l(x*) = 0.159307; GD_linesearch_t0 = 5
% ijcnn1: l(x*) = 0.195658; GD_linesearch_t0 = 50

[grad_norm_exact, l_value_loss_exact] = newton(...
                            dataset, 0.3, 0.6, 1);

tolerance_1 = @(ng) min([0.5, ng]);
tolerance_2 = @(ng) min([0.5, sqrt(ng)]);
tolerance_3 = @(ng) 0.5;
[grad_norm_1, l_value_loss_1] = inexact_newton(...
                dataset, 0.3, 0.6, 1, tolerance_1);
[grad_norm_2, l_value_loss_2] = inexact_newton(...
                dataset, 0.3, 0.6, 1, tolerance_2);
[grad_norm_3, l_value_loss_3] = inexact_newton(...
                dataset, 0.3, 0.6, 1, tolerance_3);

[grad_norm_gd, l_value_loss_gd] = GD_linesearch_2(...
                            dataset, 0.3, 0.6, 50);

semilogy(grad_norm_exact)
hold on
semilogy(grad_norm_1)
hold on
semilogy(grad_norm_2)
hold on
semilogy(grad_norm_3)
hold on
semilogy(grad_norm_gd)
hold off
title(sprintf('p02: %s', dataset))
ylabel('log(grad norm)')
legend('exact newton', 'inexact tolerance 1',...
'inexact tolerance 2', 'inexact tolerance 3', 'GD_line_search')

figure

```

```
semilogy(l_value_loss_exact)
hold on
semilogy(l_value_loss_1)
hold on
semilogy(l_value_loss_2)
hold on
semilogy(l_value_loss_3)
hold on
semilogy(l_value_loss_gd)
hold off
title(sprintf('p02: %s', dataset))
ylabel('log(function value loss)')
legend('exact newton', 'inexact tolerance 1',...
'inexact tolerance 2', 'inexact tolerance 3', 'GD_line_search')
```