Numerical Analysis
Assignment 5

Chuan Lu

September 22, 2017

Problem 1. Problem 2.21, Page 121

Solution. According to Theorem 2.7, we should choose c so as to have $|g'(\alpha)| < 1$. Thus

$$|g'(\alpha)| = |1 + cf'(\alpha)| < 1.$$

So c satisfies $-2 < cf'(\alpha) < 0$. For a good rate of convergence, pick c s.t. $cf'(\alpha) \sim 0$.

Problem 2. Problem 2.24, Page 121

Solution. **(a)**

$$g(x) = -16 + 6x + \frac{12}{x}.$$

Thus $g'(2) = 3 > 1$. So the iteration does not converge.
(b)

$$g(x) = \frac{2}{3}x + \frac{1}{x^2}.$$

Thus $g'(3^{\frac{1}{3}}) = \frac{2}{3} - 2\alpha^{-3} = 0$. So the iteration converges. Since

$$\begin{aligned}\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} &= \lim_{n\to\infty} \frac{-2x_n^3 + 3\alpha x_n^2 - 3}{3x_n^2(\alpha - x_n)^2} = \lim_{x_n\to\alpha} \frac{-2x_n^3 + 3\alpha x_n^2 - 3}{3x_n^2(\alpha - x_n)^2} \\ &= \lim_{x_n\to\alpha} \frac{-6x_n^2 + 6\alpha x_n}{6x_n(\alpha - x_n)^2 - 6x_n^2(\alpha - x_n)} = \lim_{x_n\to\alpha} \frac{1}{\alpha - 2x_n} \\ &= -\frac{1}{\alpha}.\end{aligned}$$

We know the iteration is second-order convergent.
(c)

$$g(x) = \frac{12}{1 + x}.$$

Thus $g'(\alpha) = -\frac{3}{4}$, so the iteration converges. Since

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = \frac{(1 + x_n)\alpha - 12}{(1 + x_n)(\alpha - x_n)} = \frac{\alpha}{\alpha - 2x_n - 1} = -\frac{3}{4},$$

We know the iteration is linear convergence, and the rate is $\frac{3}{4}$.

Problem 3. Problem 2.28, Page 122

Proof. Assume $f(x) = (x - \alpha)h(x),\ h(\alpha) \neq 0$. Then

$$\begin{aligned}g(x) = x - \frac{f(x)}{D(x)} &= x - \frac{(x - \alpha)^2 h^2(x)}{(x - \alpha)(h(x) + 1)h((x - \alpha)h(x) + x) - (x - \alpha)h(x)} \\ &= x - \frac{(x - \alpha)h^2(x)}{(h(x) + 1)h((x - \alpha)h(x) + x) - h(x)}.\end{aligned}$$

Thus

$$\begin{aligned}g'(\alpha) &= 1 - \frac{(h^2(x) + 2(x - \alpha)h(x)h'(x))((h(x) + 1)h((x - \alpha)h(x) + x) - h(x)) - (x - \alpha)M}{((h(x) + 1)h((x - \alpha)h(x) + x) - h(x))^2}\Bigg|_{x=\alpha} \\ &= 1 - \frac{h^4(\alpha)}{h^4(\alpha)} = 0.\end{aligned}$$

And we can know that $g''(\alpha) \neq 0$. Accoding to Theorem 2.8, this iteration is a second-order method.

Problem 4. Problem 2.48, Page 126

Proof. According to the continuity of $\|\cdot\|_\infty$, we can find an ϵ and a closed set $B(\alpha, \delta)$, s.t. $\forall x \in B(\alpha, \delta)$, $\|G(x)\|_\infty \leq 1 - \epsilon$. Then $\forall x_0 \in B = B(\alpha, \delta)$,

$$\|\alpha - x_{n+1}\| = \|g(\alpha) - g(x_n)\| = \|G(\xi)\| \cdot \|\alpha - x_n\| \leq (1-\epsilon)\|\alpha - x_n\| \leq (1-\epsilon)^n \|\alpha - x_0\|.$$

Thus $\|\alpha - x_{n+1}\| \to 0,\ t \to \infty$. So the iteration converges to α. And since $g(B) \subset B$ according to that $\|\alpha - g(x)\| < \|\alpha - x\|$, and $\max\|G(x)\| < 1$, we know that it satisfies the condition of Theorem 2.9.

Problem 5. Problem 2.50, Page 127

Solution. The code and result are as follows;(In fact)

```
function [x, error, count] = Newton(f, F, x0, max_iter, tol)
% Newton's method for nonlinear system;
% f: the funtion to extract root;
% F: Jacobian of f;
% x0: initial guess;
% max_iter, tol: max iteration and tolerance;
count = 0;
x = zeros(100, 2);
error = zeros(100, 1);
while(1)
    count = count + 1;
    x(count,1:2) = x0;
    x1 = x0 - feval(F, x0)\feval(f, x0);
    error(count) = norm(x1-x0);
    if(error(count) < tol)
        return;
    elseif(count > max_iter)
        warning('Iteration count overflow, may not converge');
        return;
    end
    x0 = x1;
end
```

```
format long;
f = @(x)([x(1)^2+x(2)^2-4; x(1)^2-x(2)^2-1]);
F = @(x)([2*x(1), 2*x(2); 2*x(1), -2*x(2)]);
x0 = [1.6; 1.2];
max_it = 100;
tol = 1e-6;
[x, error, count] = Newton(f, F, x0, max_it, tol);
disp(x(1:count, 1:2));
```

```
>> prob50
   1.600000000000000   1.200000000000000
   1.581250000000000   1.225000000000000
   1.581138833992095   1.224744897959184
```

Problem 6. Problem 2.51, Page 127

Solution. We just use the code of Newton's method from Problem 2.50, and the executive script is as follows;

```
format long;
f = @(x)([x(1)^2+x(1)*x(2)^3-9; 3*x(1)^2*x(2)-x(2)^3-4]);
F = @(x)([2*x(1)+x(2)^3, 3*x(2)^2; 6*x(1)*x(2), 3*x(1)^2-3*x(2)^2]);
max_iter = 100;
```

```
tol = 1e-6;

%%%%%%%%% Case 1 %%%%%%%%
% figure(1);
% x0 = [1.2; 2.5];
% [x, error, count] = Newton(f, F, x0, max_iter, tol);
% disp(x(1:count, 1:2));
% disp(count);
% semilogy(1:count, error(1:count));
% legend('Convergence Rate');
% title('Convergence rate for init = (1.2, 2.5)');
% xlabel('Iteration Count');
% ylabel('Error');
% grid on;

%%%%%%%%% Case 2 %%%%%%%%
% figure(2);
% x0 = [-2; 2.5];
% [x, error, count] = Newton(f, F, x0, max_iter, tol);
% disp(x(1:count, 1:2));
% disp(count);
% semilogy(1:count, error(1:count));
% legend('Convergence Rate');
% title('Convergence rate for init = (-2, 2.5)');
% xlabel('Iteration Count');
% ylabel('Error');
% grid on;

%%%%%%%%% Case 3 %%%%%%%%%
% figure(3);
% x0 = [-1.2; -2.5];
% [x, error, count] = Newton(f, F, x0, max_iter, tol);
% disp(x(1:count, 1:2));
% disp(count);
% semilogy(1:count, error(1:count));
% legend('Convergence Rate');
% title('Convergence rate for init = (-1.2, -2.5)');
% xlabel('Iteration Count');
% ylabel('Error');
% grid on;

%%%%%%%% Case 4 %%%%%%%%%
figure(4);
x0 = [2; -2.5];
[x, error, count] = Newton(f, F, x0, max_iter, tol);
disp(x(1:count, 1:2));
disp(count);
semilogy(1:count, error(1:count));
legend('Convergence Rate');
title('Convergence rate for init = (2, -2.5)');
xlabel('Iteration Count');
ylabel('Error');
grid on;
```

Result. The result is shown in the following table 1, and the convergence speed shown in **??**. From the result we know that different initial values may lead to convergence to different roots, while when the iteration comes to very close to the root, the convergence order is first-order.

Initial Point	Convergence	Result	Number of iterates
(1.2, 2.5)	Yes	(1.336355248159268, 1.754234729131226)	11
(-2, 2.5)	Yes	(1.336355247361990, 1.754234726236878)	19
(-1.2, -2.5)	Yes	(1.336355235838950, 1.754234684404958)	24
(2, -2.5)	Yes	(2.998365352785336, 0.148430977265564)	9

Table 1: The result of convergence and iteration count.

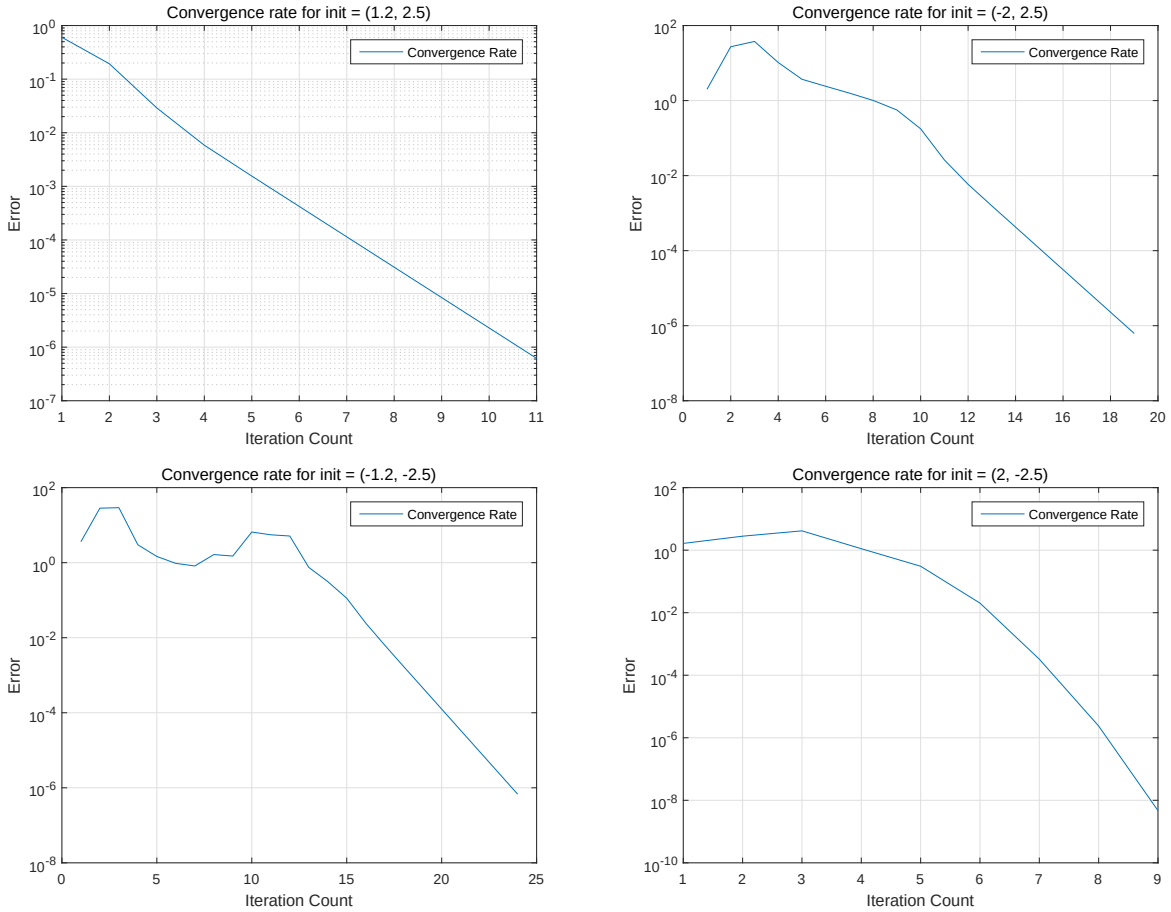

Figure 1: Convergence Order