HW3

Textbook §2.2 #9.

- Refer to the solutions in the textbook to prove the existence and uniqueness of fixed point by Thm.2.3.
- Run the following code. Then the numerical result shows that the approximation solution is 3.626995622438735 with 0.010000 accuracy after 3 iterations.

```
%fixed point iteration
p0 = pi; TOL = 10^(-2); N0 = 100;
g = @(x) pi + 0.5*sin(x/2);
i = 1;
while (i <= N0)
    p = g(p0);
    if (abs(p-p0) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
         after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    p0 = p;
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

- For $p_0 = \pi$, Cor.2.5 implies that

$$|p_n - p| \le \left(\frac{1}{4}\right)^n \pi < 0.01 \Rightarrow n > 4.1477... \Rightarrow n \ge 5$$

  and

$$|p_n - p| \le \frac{4}{3}\left(\frac{1}{4}\right)^n \frac{1}{2} < 0.01 \Rightarrow n > 3.0294... \Rightarrow n \ge 4.$$

Textbook §2.2 #14a.

- Let $g(x) = 2 + \sin x$. Then $g \in C[2, 3]$. Note that
  - $x \in [2, 3] \subset [\pi/2, \pi] \Rightarrow 0 \leq \sin x \leq 1 \Rightarrow g([2, 3]) \subset [2, 3]$.
  - $|g'(x)| = |\cos x| \leq |\cos 3| < 1, \ \forall\, x \in (2, 3)$.

  By Thm.2.4, for any $p_0 \in [2, 3]$, the sequence $p_n = g(p_{n-1})$ converges to the unique fixed point $p \in [2, 3]$.

- For $p_0 = 2.5$, Cor.2.5 implies that

$$|p_n - p| \leq k^n \max\{p_0 - a, b - p_0\} = |\cos 3|^n \cdot 0.5 < 10^{-5}$$

$$\Rightarrow n > 1075.747628... \Rightarrow n \geq 1076$$

  and

$$|p_n - p| \leq \frac{|\cos 3|^n}{1 - |\cos 3|} |p_1 - p_0| < 10^{-5}$$

$$\Rightarrow n > 1371.990509... \Rightarrow n \geq 1372.$$

- Run the following code. Then the numerical result shows that the approximation solution is 2.554192102747867 with 0.000010 accuracy after 52 iterations.

```
%fixed point iteration
p0 = 2.5; TOL = 10^(-5); N0 = 100;
g = @(x) 2 + sin(x);
i = 1;
while (i <= N0)
    p = g(p0);
    if (abs(p-p0) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
        after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    p0 = p;
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

Textbook §2.2 #20.

- $p = \lim_{n\to\infty} p_n = lim_{n\to\infty} g(p_{n-1}) = 2p - Ap^2 \Rightarrow p = \frac{1}{A}$.
- Any closed subinterval $[a, b]$ of $(\frac{1}{2A}, \frac{3}{2A})$ containing $\frac{1}{A}$ suffices. $i.e.$ $\frac{1}{2A} < a \leq \frac{1}{A} \leq b < \frac{3}{2A}$. To prove the convergence, first note that $g \in C[a, b]$. Then note that
  - g is increasing on $[a, \frac{1}{A}]$ and decreasing on $[\frac{1}{A}, b]$. Furthermore,

$$a \leq \frac{1}{A} \Rightarrow g(a) \geq a$$

$$b \geq \frac{1}{A} \Rightarrow g(b) \leq b$$

$$g(\frac{1}{A}) = \frac{1}{A} \text{ is the maximum.}$$

    Therefore, $g([a, b]) \subset [a, b]$.
  - $|g'(x)| = 2A|\frac{1}{A} - x| \leq 2A \max\{\frac{1}{A} - a, b - \frac{1}{A}\} < 2A\frac{1}{2A} = 1,\ \forall\, x \in (a, b)$.

HW #2.

Let $g(x) = 2 + \sin x$ and $\bar{g}(x) = \alpha x + (1-\alpha)g(x)$. Consider

$$p_{n+1} = \bar{g}(p_n) = \alpha p_n + (1-\alpha)g(p_n)$$

$$p_* = \bar{g}(p_*) = \alpha p_* + (1-\alpha)g(p_*).$$

$$\overset{MVT}{\Rightarrow} p_{n+1} - p_* = [\alpha + (1-\alpha)g'(\xi_n)](p_n - p_*) \text{ for some } \xi_n \text{ between } p_n \text{ and } p_*.$$

To accelerate the convergence, we choose α such that

$$\alpha + (1-\alpha)g'(\xi_n) = 0$$

$$\Rightarrow \alpha = \frac{g'(\xi_n)}{g'(\xi_n) - 1}.$$

Reasonable guesses for $g'(\xi_n)$ include $g'(2.5)$, $\frac{g(3)-g(2)}{3-2}$, and $\frac{g'(2)+g'(3)}{2}$. Both of them work well in this example.

For $g'(\xi_n) = g'(2.5)$, the approximation solution is 2.554196096669195 with 0.000010 accuracy after 3 iterations.

For $g'(\xi_n) = \frac{g(3)-g(2)}{3-2}$, the approximation solution is 2.554195880784590 with 0.000010 accuracy after 4 iterations.

For $g'(\xi_n) = \frac{g'(2)+g'(3)}{2}$, the approximation solution is 2.554196074187307 with 0.000010 accuracy after 5 iterations.

```
%generalized fixed point iteration
p0 = 2.5; TOL = 10^(-5); N0 = 100;
k = cos(2.5);
al = k / (k-1);
g = @(x) al*x + (1-al)*(2 + sin(x));
i = 1;
while (i <= N0)
    p = g(p0);
    if (abs(p-p0) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
        after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    p0 = p;
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

Textbook §2.3 #13a.

For $p_0 = -1, p_1 = 0$, the numerical result shows that the approximation solution is -0.040658499043342 with 0.000001 accuracy after 17 iterations.

For $p_0 = 0, p_1 = 1$, the numerical result shows that the approximation solution is 0.962398384238757 with 0.000001 accuracy after 9 iterations.

```
%method of false point
p0 = -1; p1 = 0; TOL = 10^(-6); N0 = 100;
%p0 = 0; p1 = 1; TOL = 10^(-6); N0 = 100;
f = @(x) 230*x^4 + 18 *x^3 + 9*x^2 - 221*x - 9;
i = 2;
q0 = f(p0);
q1 = f(p1);
while (i <= N0)
    p = p1 - q1*(p1-p0)/(q1-q0);
    if (abs(p-p1) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
        after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    q = f(p);
    if (q*q1 < 0)
        p0 = p1;
        q0 = q1;
    end
    p1 = p;
    q1 = q;
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

Textbook §2.3 #13b.

For $p_0 = -1, p_1 = 0$, the numerical result shows that the approximation solution is -0.040659288315725 with 0.000001 accuracy after 5 iterations.

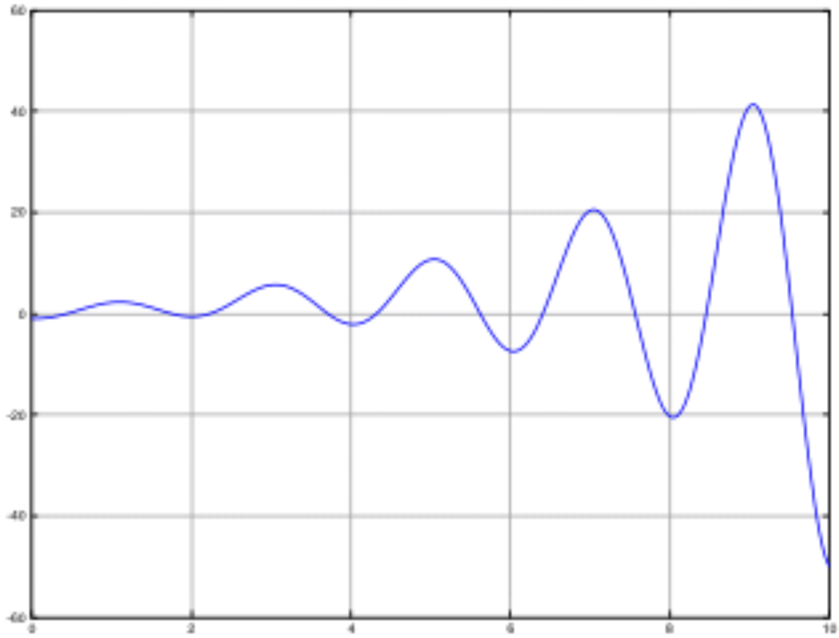
For $p_0 = 0, p_1 = 1$, the numerical result shows that the approximation solution is -0.040659288315572 with 0.000001 accuracy after 12 iterations.

```
%secant method
p0 = -1; p1 = 0; TOL = 10^(-6); N0 = 100;
%p0 = 0; p1 = 1; TOL = 10^(-6); N0 = 100;
f = @(x) 230*x^4 + 18 *x^3 + 9*x^2 - 221*x - 9;
i = 2;
q0 = f(p0);
q1 = f(p1);
while (i <= N0)
    p = p1 - q1*(p1-p0)/(q1-q0);
    if (abs(p-p1) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
        after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    p0 = p1;
    q0 = q1;
    p1 = p;
    q1 = f(p);
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

Textbook §2.3 #17c.

Plot the graph of f for an observation.



Test some initial points for more observations.

For $p_0 = 0.5$, the numerical result shows that the approximation solution is 0.450656747890593 with 0.000010 accuracy after 3 iterations.

For $p_0 = 1.5$, the numerical result shows that the approximation solution is 1.744738053368350 with 0.000010 accuracy after 3 iterations.

```
%Newton's method
p0 = 0.5; TOL = 10^(-5); N0 = 100;
%p0 = 1.5; TOL = 10^(-5); N0 = 100;
f = @(x) log(x^2+1) - exp(0.4*x) * cos(pi*x);
df = @(x) 2*x/(1+x^2) - exp(0.4*x) * (0.4*cos(pi*x)-pi*sin(pi*x));
i = 1;
while (i <= N0)
    p = p0 - f(p0)/df(p0);
    if (abs(p-p0) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
        after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    p0 = p;
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

Finally, we conclude that $n - \frac{1}{2}$ is a reasonable initial approximation to find the nth smallest positive zero.

Textbook §2.3 #18.

For $p_0 = \pi/2$, the numerical result shows that the approximation solution is 1.895488418971447 with 0.000010 accuracy after 15 iterations.

For $p_0 = 5\pi$, the numerical result shows that the approximation solution is 1.895489001382098 with 0.000010 accuracy after 19 iterations.

For $p_0 = 10\pi$, the numerical result shows that the method failed after N_0 iterations, $N_0 = 100$.

```
%Newton's method
p0 = pi/2; TOL = 10^(-5); N0 = 100;
%p0 = 5*pi; TOL = 10^(-5); N0 = 100;
%p0 = 10*pi; TOL = 10^(-5); N0 = 100;
f = @(x) 0.5 + 0.25*x^2 - x*sin(x) - 0.5*cos(2*x);
df = @(x) 0.5*x - sin(x) - x*cos(x) + sin(2*x);
i = 1;
while (i <= N0)
    p = p0 - f(p0)/df(p0);
    if (abs(p-p0) < TOL)
        fprintf('The approximation solution is %.15f with %f accuracy ...
        after %d iterations.\n', p, TOL, i);
        return;
    end
    i += 1;
    p0 = p;
end
printf('The method failed after N0 iterations, N0 = %d\n', N0);
return;

end
```

The results do not indicate the fast convergence usually associated with Newton's method.