MATH 4513 Numerical Analysis

Chapter 2. Solutions of Equations in One Variable

Xu Zhang

Assistant Professor

Department of Mathematics

Oklahoma State University

Text Book: Numerical Analysis (10th edition)

R. L. Burden, D. J. Faires, A. M. Burden

Chapter 2. Solutions of Equations in One Variable

Contents

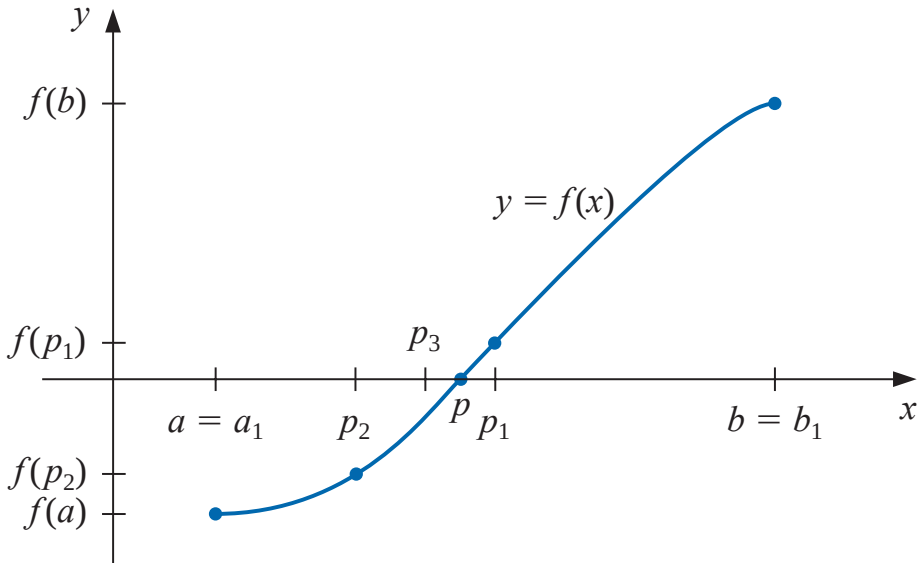
Section 2.1 The Bisection Method

- Starting from this section, we study the most basic mathematics problem: **root-finding problem**

$$f(x) = 0.$$

- The first numerical method, based on the Intermediate Value Theorem (IVT), is called the **Bisection Method**.
- Suppose that $f(x)$ is continuous on $[a, b]$. $f(a)$ and $f(b)$ have opposite sign. By IVT, there exists a number $p \in (a, b)$ such that $f(p) = 0$. That is, $f(x)$ has a root in (a, b).
- **Idea of Bisection Method:** repeatedly halve the subinterval of $[a, b]$, and at each step, locating the half containing the root.

- Set $a_1 \leftarrow a$, $b_1 \leftarrow b$. Calculate the midpoint $p_1 \leftarrow \frac{a_1+b_1}{2}$.



- If $f(p_1) = 0$, then $p \leftarrow p_1$, done.
- If $f(p_1) \neq 0$, then $f(p_1)$ has the same sign as either $f(a)$ or $f(b)$.
  - If $f(p_1)$ and $f(a)$ have the same sign, then $p \in (p_1, b_1)$. Set $a_2 \leftarrow p_1$, and $b_2 \leftarrow b_1$.
  - If $f(p_1)$ and $f(b)$ have the same sign, then $p \in (a_1, p_1)$. Set $a_2 \leftarrow a_1$, and $b_2 \leftarrow p_1$.
- Repeat the process on $[a_2, b_2]$.

ALGORITHM – Bisection (Preliminary Version)

USAGE: to find a solution to $f(x) = 0$ on the interval $[a, b]$.

$$p = \textbf{bisect0}\ (f, a, b)$$

For $n = 1, 2, 3, \cdots, 20$, do the following

- **Step 1** Set $p = (a + b)/2$;
- **Step 2** Calculate $FA = f(a)$, $FB = f(b)$, and $FP = f(p)$.
- **Step 3** If $FA \cdot FP > 0$, set $a = p$
  If $FB \cdot FP > 0$, set $b = p$.
  Go back to **Step 1**.

Remark

This above algorithm will perform 20 times bisection iterations. The number 20 is artificial.

Example 1.

Show that $f(x) = x^3 + 4x^2 - 10 = 0$ has a root in $[1, 2]$ and use the Bisection method to find the approximation root.

Solution.

Because $f(1) = -5$ and $f(2) = 14$, the IVT ensures that this continuous function has a root in $[1, 2]$.

To proceed with the Bisection method, we write a simple MATLAB code.

Matlab Code for Bisection (Preliminary Version)

```
function p = bisect0(fun,a,b)
% This is a preliminary version of Bisection Method

for n = 1:20 % Set max number of iterations to be 20
    p = (a+b)/2;
    FA = fun(a);
    FB = fun(b);
    FP = fun(p);
    if FA*FP > 0
        a = p;
    elseif FB*FP > 0
        b = p;
    end
end
```

A "Driver" File

```
% Driver File: Example 2.1.1 in the Textbook

%% Inputs
fun = @(x) x^3+4*x^2-10;
a = 1;
b = 2;

%% Call the subroutine: bisect0.m
p = bisect0(fun,a,b)
```

- After 20 iterations, we obtain the solution $p \approx 1.365229606628418$.
- To display more information from the whole iteration process, we modify the MATLAB subroutine file.

Matlab Code for Bisection (Preliminary Version with more outputs)

```
function p = bisect1(fun,a,b)
% This is a preliminary version of Bisection Method
% This version displays intermediate outputs nicely

disp('Bisection Methods')
disp('------------------------------------------------------------------')
disp(' n     a_n             b_n             p_n             f(p_n)')
disp('------------------------------------------------------------------')
formatSpec = '%2d    % .9f    % .9f    % .9f    % .9f \n';

for n = 1:20 % Set max number of iterations to be 20
    p = (a+b)/2;
    FA = fun(a);
    FB = fun(b);
    FP = fun(p);
    fprintf(formatSpec,[n,a,b,p,fun(p)]) % Printing output
    if FA*FP > 0
        a = p;
    elseif FB*FP > 0
        b = p;
    end
end
```

Some Remarks on Bisection Method

- To start, an interval $[a, b]$ must be found with $f(a) \cdot f(b) < 0$. Otherwise, there may be no solutions in that interval.
- It is good to set a maximum iteration number "$maxit$", in case the the iteration enters an endless loop.
- It is good to set a tolerance or stopping criteria to avoid unnecessary computational effort, such as
  1. $\dfrac{b_n - a_n}{2} < tol$
  2. $|p_n - p_{n+1}| < tol$
  3. $\dfrac{|p_n - p_{n+1}|}{|p_n|} < tol$
  4. $|f(p_n)| < tol$

A more robust Matlab code for Bisection method

```
function [p,flag] = bisect(fun,a,b,tol,maxIt)
%% This is a more robust version of Bisection Method than bisect1.m

flag = 0; % Use a flag to tell if the output is reliable
if fun(a)*fun(b) > 0 % Check f(a) and f(b) have different sign
    error('f(a) and f(b) must have different signs');
end

disp('Bisection Methods')
disp('------------------------------------------------------------------')
disp(' n     a_n             b_n             p_n             f(p_n)')
disp('------------------------------------------------------------------')
formatSpec = '%2d    % .9f    % .9f    % .9f    % .9f \n';

for n = 1:maxIt
    p = (a+b)/2;
    FA = fun(a);
    FP = fun(p);
    fprintf(formatSpec,[n,a,b,p,fun(p)]) % Printing output
    if abs(FP) <= 10^(-15) || (b-a)/2 < tol
        flag = 1;
        break; % Break out the for loop.
    else
        if FA*FP > 0
            a = p;
        else
            b = p;
        end
    end
end
```

Example 2.

Use Bisection method to find a root of $f(x) = x^3 + 4x^2 - 10 = 0$ in the interval $[1, 2]$ that is accurate to at least within 10^{-4}.

Solution.

- We write a Matlab driver file for this test problem

```
% Example 2.1.1 in the Textbook

fun = @(x) x^3+4*x^2-10;
a = 1;
b = 2;
tol = 1E-4;
maxIt = 40;

[p,flag] = bisect(fun,a,b,tol,maxIt);
```

- In this driver file, we
  - specify all five **inputs**: fun, a, b, tol, maxIt
  - call the Bisection method code **bisect.m**.

Outputs from the Matlab Command Window

```
>> ex2_1_1
Bisection Methods
-------------------------------------------------------------------
 n     a_n              b_n              p_n              f(p_n)
-------------------------------------------------------------------
 1     1.000000000      2.000000000      1.500000000      2.375000000
 2     1.000000000      1.500000000      1.250000000     -1.796875000
 3     1.250000000      1.500000000      1.375000000      0.162109375
 4     1.250000000      1.375000000      1.312500000     -0.848388672
 5     1.312500000      1.375000000      1.343750000     -0.350982666
 6     1.343750000      1.375000000      1.359375000     -0.096408844
 7     1.359375000      1.375000000      1.367187500      0.032355785
 8     1.359375000      1.367187500      1.363281250     -0.032149971
 9     1.363281250      1.367187500      1.365234375      0.000072025
10     1.363281250      1.365234375      1.364257812     -0.016046691
11     1.364257812      1.365234375      1.364746094     -0.007989263
12     1.364746094      1.365234375      1.364990234     -0.003959102
13     1.364990234      1.365234375      1.365112305     -0.001943659
14     1.365112305      1.365234375      1.365173340     -0.000935847
>>
```

The approximation p_n converges to the true solution $p = 1.365230013...$

Theorem 3 (Convergence of Bisection Method).

Suppose that $f \in C[a,b]$ and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with

$$|p_n - p| \leq \frac{b-a}{2^n}, \quad \textit{when } n \geq 1.$$

Proof.

For $n \geq 1$, we have $p \in (a_n, b_n)$ and

$$b_n - a_n = \frac{1}{2^{n-1}}(b-a).$$

Since $p_n = \frac{1}{2}(a_n + b_n)$ for all $n \geq 1$, then

$$|p_n - p| \leq \frac{1}{2}(b_n - a_n) = \frac{b-a}{2^n}.$$

Example 4.

Determine the number of iteration necessary to solve $f(x) = x^3 + 4x^2 - 10 = 0$ with accuracy 10^{-3} using $a_1 = 1$ and $b_1 = 2$.

Solution.

By the convergence theorem (Theorem 2.3), we have

$$|p_n - p| \leq \frac{b-a}{2^n} = \frac{1}{2^n} < 10^{-3}.$$

That is

$$2^n > 10^3 \quad \Longrightarrow \quad n > 3\frac{\log 10}{\log 2} \approx 9.96.$$

Hence, 10 iterations are required.

2.2 Fixed-Point Iteration

A **fixed point** for a function is a number at which the value of the function does not change when the function is applied.

Definition 5 (fixed point).

The point p is a **fixed point** for a function $g(x)$, if $g(p) = p$.

Root-finding problems and fixed-point problems are equivalent:

- Given a root-finding problem $f(p) = 0$, we can define functions $g(x)$ with a fixed point at p in many ways such as

$$g(x) = x - f(x), \qquad g(x) = x - \frac{f(x)}{f'(x)}, \quad \text{if } f'(p) \neq 0.$$

- Given a function g has a fixed point at p, the function f defined by

$$f(x) = g(x) - x$$

has a root at p.

Example 6.

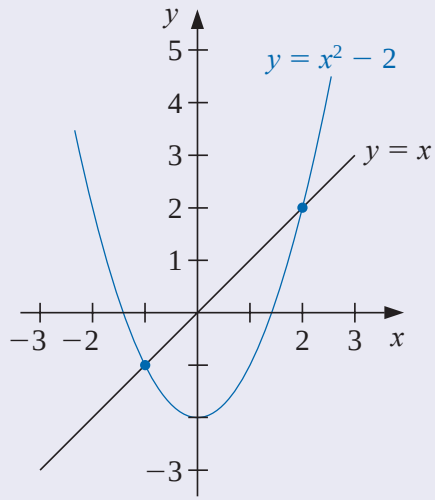
Determine any fixed points of the function $g(x) = x^2 - 2$

Solution

- If p is a fixed point of g, then

$$p = p^2 - 2 \implies p^2 - p - 2 = (p-2)(p+1) = 0$$
$$\implies p = -1 \text{ or } p = 2.$$

- $g(x)$ has two fixed points $p = -1$ and $p = 2$.



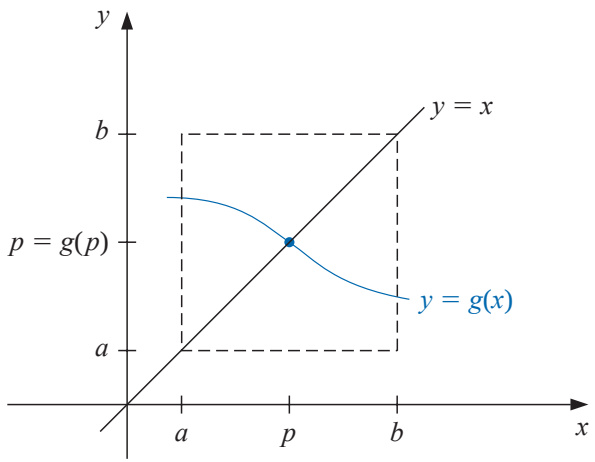
- The fixed point of $g(x)$ is the intersection of $y = g(x)$ and $y = x$.

Theorem 7 (Sufficient Conditions for Fixed Points).

- **(i) (Existence)** *If $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$, then g has at least one fixed point in $[a,b]$.*
- **(ii) (Uniqueness)** *If, in addition, $g'(x)$ exists and satisfies*

$$|g'(x)| \leq k < 1, \quad \text{for all } x \in (a,b),$$

for some positive constant k, there is exactly one fixed-point in $[a,b]$.



Note: the proof of existence uses the Intermediate Value Theorem, and the proof of uniqueness uses the Mean Value Theorem.

Example 8.

Show that $g(x) = \frac{1}{3}(x^2 - 1)$ has a unique fixed-point on $[-1, 1]$.

Proof (1/2)

(1. Existence). We show that $g(x)$ has at least a fixed point $p \in [-1, 1]$.

- Taking the derivative,

$$g'(x) = \frac{2x}{3}, \quad \text{only one critical point } x = 0, \quad g(0) = -\frac{1}{3}.$$

- At endpoints, $x = -1$ and 1, we have $g(-1) = 0$, and $g(1) = 0$.
- Then we have the global extreme values

$$\min_{x \in [-1,1]} g(x) = -\frac{1}{3}, \quad \text{and} \quad \max_{x \in [-1,1]} g(x) = 0.$$

- Therefore, $g(x) \in [-\frac{1}{3}, 0] \subset [-1, 1]$. By the first part of Theorem 2.7, the function g has at least one fixed-point on $[-1, 1]$.

Proof (2/2)

(2. Uniqueness). We show that $g(x)$ has exactly one fixed point.

- Note that

$$|g'(x)| = \left|\frac{2x}{3}\right| \leq \frac{2}{3}, \quad \forall x \in (-1, 1).$$

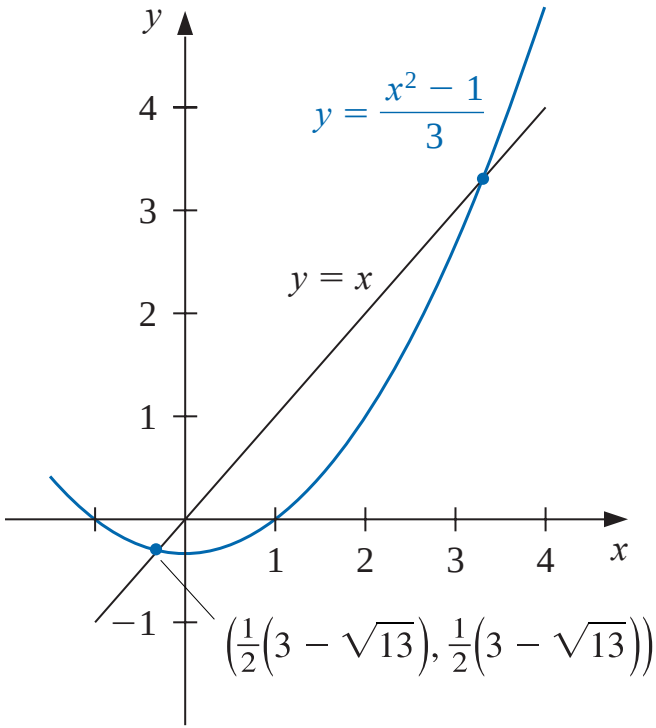
- By part (ii) of Theorem 2.7, g has a unique fixed-point on $[-1, 1]$.

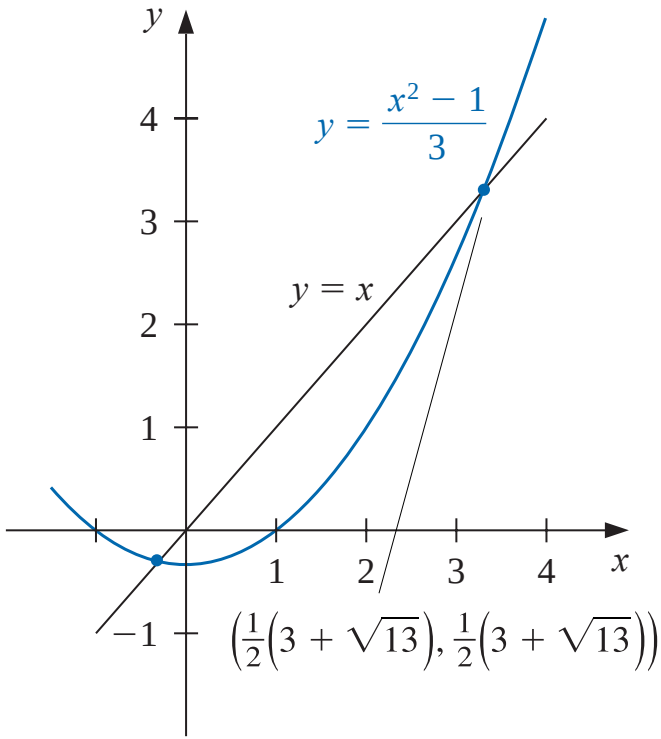
Remark

- In fact, $p = \dfrac{3 - \sqrt{13}}{2}$ is the fixed-point on the interval $[-1, 1]$.

Remark

- The function g has another fixed point $q = \frac{3+\sqrt{13}}{2}$ on the interval $[3, 4]$. However, it does not satisfy the hypotheses of Theorem 2.7 **(why? exercise)**.
- The hypotheses in Theorem 2.7 are sufficient but not necessary.





Fixed-Point Iteration

- If $g(x)$ is continuous, we can approximate the fixed point of g (if any) by

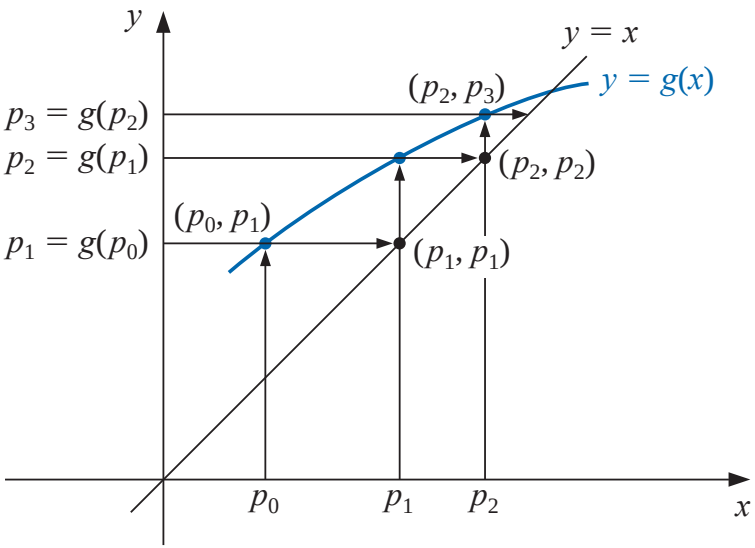
  Step 1 choose an initial approximation p_0

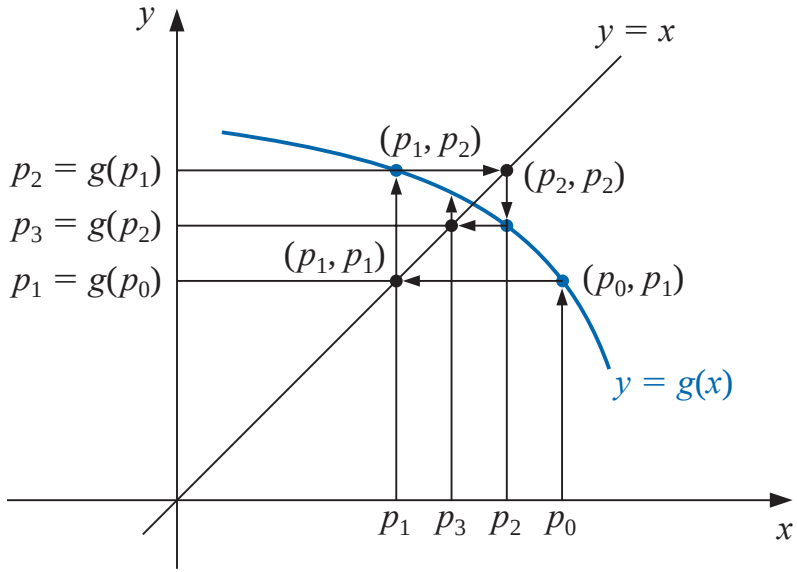
  Step 2 for $n \geq 1$, do $p_n = g(p_{n-1})$

- If $\{p_n\}$ converges to a number p, then

$$p = \lim_{n\to\infty} p_n = \lim_{n\to\infty} g(p_{n-1}) = g\left(\lim_{n\to\infty} p_{n-1}\right) = g(p).$$

  Thus, the number p is a fixed-point of g.





Matlab Code of Fixed-Point Iteration

```
function [p,flag] = fixedpoint(fun,p0,tol,maxIt)

n = 1; flag = 0; % Initialization

disp('Fixed Point Iteration')
disp('-----------------------------------')
disp(' n     p                f(p_n)')
disp('-----------------------------------')
formatSpec = '%2d    % .9f    % .9f \n';
fprintf(formatSpec,[n-1,p0,fun(p0)]) % printing output

while n <= maxIt
    p = fun(p0);
    fprintf(formatSpec,[n,p,fun(p)]) % printing output
    if abs(p-p0) < tol
        flag = 1;
        break;
    else
        n = n+1;
        p0 = p;
    end
end
```

Note: unlike Bisection method, we don't need to input an interval $[a, b]$ to start the fixed-point iteration, but we need an initial guess p_0.

Example 9.

The equation $x^3 + 4x^2 - 10 = 0$ has a unique solution in $[1, 2]$. There are many ways to change the equation to a fixed-point problem $x = g(x)$. For example,

- $g_1(x) = x - x^3 - 4x^2 + 10$
- $g_2(x) = \sqrt{\dfrac{10}{x} - 4x}$
- $g_3(x) = \dfrac{1}{2}\sqrt{10 - x^3}$
- $g_4(x) = \sqrt{\dfrac{10}{4 + x}}$
- $g_5(x) = x - \dfrac{x^3 + 4x^2 - 10}{3x^2 + 8x}$

Which one is better?

Solution(1/2): Write a Matlab driver file for this example

```
% Example 2.2.1 in the Textbook
% Compare the convergence of fixed point iteration for five functions

clc % clear the command window
fun = @(x) x^3+4*x^2-10;

funG1 = @(x) x-x^3-4*x^2+10;
funG2 = @(x) sqrt(10/x-4*x);
funG3 = @(x) (1/2)*sqrt(10-x^3);
funG4 = @(x) sqrt(10/(4+x));
funG5 = @(x) x-(x^3+4*x^2-10)/(3*x^2+8*x);

p0 = 1.5;
tol = 1E-9;
maxIt = 40;

disp('-------------Test #1-------------')
[p1,flag1] = fixedpoint(funG1,p0,tol,maxIt);
disp('-------------Test #2-------------')
[p2,flag2] = fixedpoint(funG2,p0,tol,maxIt);
disp('-------------Test #3-------------')
[p3,flag3] = fixedpoint(funG3,p0,tol,maxIt);
disp('-------------Test #4-------------')
[p4,flag4] = fixedpoint(funG4,p0,tol,maxIt);
disp('-------------Test #5-------------')
[p5,flag5] = fixedpoint(funG5,p0,tol,maxIt);

disp(' ')
disp('Converge or Not')
disp([flag1,flag2,flag3,flag4,flag5])
```

Solution(2/2)

Iterations of g_1 and g_2 diverge. Iterations of g_3, g_4, and g_5 converge:

```
---------------------------------
Fixed Point Iteration
---------------------------------
 n    p                f(p_n)
---------------------------------
 0    1.500000000      1.286953768
 1    1.286953768      1.402540804
 2    1.402540804      1.345458374
 3    1.345458374      1.375170253
 4    1.375170253      1.360094193
 5    1.360094193      1.367846968
 6    1.367846968      1.363887004
 7    1.363887004      1.365916733
 8    1.365916733      1.364878217
 9    1.364878217      1.365410061
10    1.365410061      1.365137821
11    1.365137821      1.365277209
12    1.365277209      1.365205850
13    1.365205850      1.365242384
14    1.365242384      1.365223680
15    1.365223680      1.365233256
16    1.365233256      1.365228353
17    1.365228353      1.365230863
18    1.365230863      1.365229578
19    1.365229578      1.365230236
20    1.365230236      1.365229899
21    1.365229899      1.365230072
22    1.365230072      1.365229984
23    1.365229984      1.365230029
24    1.365230029      1.365230006
25    1.365230006      1.365230017
26    1.365230017      1.365230011
27    1.365230011      1.365230014
28    1.365230014      1.365230013
29    1.365230013      1.365230014
30    1.365230014      1.365230013
---------------------------------
```

```
---------------------------------
Fixed Point Iteration
---------------------------------
 n    p                f(p_n)
---------------------------------
 0    1.500000000      1.348399725
 1    1.348399725      1.367376372
 2    1.367376372      1.364957015
 3    1.364957015      1.365264748
 4    1.365264748      1.365225594
 5    1.365225594      1.365230576
 6    1.365230576      1.365229942
 7    1.365229942      1.365230023
 8    1.365230023      1.365230012
 9    1.365230012      1.365230014
10    1.365230014      1.365230013
11    1.365230013      1.365230013
---------------------------------
Fixed Point Iteration
---------------------------------
 n    p                f(p_n)
---------------------------------
 0    1.500000000      1.373333333
 1    1.373333333      1.365262015
 2    1.365262015      1.365230014
 3    1.365230014      1.365230013
 4    1.365230013      1.365230013

Converge or Not
     0     0     1     1     1
```

Questions

- Why do iterations g_1 and g_2 diverge? but g_3, g_4, and g_5 converge?
- Why do g_4 and g_5 converge more rapidly than g_3?

Theorem 10 (Fixed-Point Theorem).

Let $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$. Suppose that g' exists on (a,b) and that a constant $0 < k < 1$ exists with

$$|g'(x)| \leq k < 1, \quad \forall x \in (a,b).$$

Then for any number $p_0 \in [a,b]$, the sequence

$$p_n = g(p_{n-1}), \quad n \geq 1$$

converges to the unique fixed point p in $[a,b]$.

Proof

The function g satisfies the hypotheses of Theorem 2.7, thus g has a unique fixed-point p in $[a, b]$. By Mean Value Theorem,

$$\begin{aligned} |p_n - p| &= |g(p_{n-1}) - g(p)| \\ &= |g'(\xi)||p_{n-1} - p| \\ &\leq k|p_{n-1} - p| \\ &\leq \cdots \\ &\leq k^n|p_0 - p|. \end{aligned}$$

Since $0 < k < 1$, then

$$\lim_{n\to\infty} |p_n - p| \leq \lim_{n\to\infty} k^n|p_0 - p| = 0.$$

Hence, the sequence $\{p_n\}$ converge to p.

Remark

- The rate of convergence of the fixed-point iteration depends on the factor k. The smaller the value of k, the faster the convergence.
- To be more precise, we have the following error bounds (Corollary 2.5 in textbook)

$$|p_n - p| \leq k^n \max\{p_0 - a, b - p_0\}.$$

and

$$|p_n - p| \leq \frac{k^n}{1-k}|p_1 - p_0|.$$

- We will see more in Section 2.4.

Proof (read if you like)

Since $p \in [a, b]$, then

$$|p_n - p| \leq k^n |p_0 - p| \leq k^n \max\{p_0 - a, b - p_0\}.$$

For $n \geq 1$,

$$|p_{n+1} - p_n| = |g(p_n) - g(p_{n-1})| \leq k|p_n - p_{n-1}| \leq \cdots \leq k^n |p_1 - p_0|.$$

For $m \geq n \geq 1$,

$$\begin{aligned}
|p_m - p_n| &= |p_m - p_{m-1} + p_{m-1} - \cdots - p_{n+1} + p_{n+1} - p_n| \\
&\leq |p_m - p_{m-1}| + |p_{m_1} - p_{m-2}| + \cdots + |p_{n+1} - p_n| \\
&\leq k^{m-1}|p_1 - p_0| + k^{m-2}|p_1 - p_0| + \cdots k^n |p_1 - p_0| \\
&\leq k^n |p_1 - p_0| \left(1 + k + k^2 + k^{m-n-1}\right).
\end{aligned}$$

Proof (2/2) (read if you like)

Let $m \to \infty$, we have

$$\begin{aligned}
|p - p_n| &= \lim_{m\to\infty} |p_m - p_n| \\
&\le \lim_{m\to\infty} k^n |p_1 - p_0| \left(1 + k + k^2 + k^{m-n-1}\right) \\
&= k^n |p_1 - p_0| \sum_{i=0}^{\infty} k^i \\
&= \frac{k^n}{1-k} |p_1 - p_0|.
\end{aligned}$$

The last equality is because of the convergence of geometric series when $0 < k < 1$.

A revisit of the fixed-point schemes g_1 to g_5 in Example 2.9.

- For $g_1(x) = x - x^3 - 4x^2 + 10$, we know that

$$g_1(1) = 6, \quad \text{and} \quad g_2(2) = -12,$$

so g_1 does not map $[1, 2]$ into itself. Moreover,

$$|g_1'(x)| = |1 - 3x^2 - 8x| > 1, \quad \text{for all} \quad x \in [1, 2].$$

There is no reason to expect convergence.

- For $g_2(x) = \sqrt{\dfrac{10}{x} - 4x}$, it does not map $[1, 2]$ into $[1, 2]$. Also, there is no interval containing the fixed point $p \approx 1.365$ such that $|g_2'(x)| < 1$, because $|g_2'(p)| \approx 3.4 > 1$. There is no reason to expect it to converge.

A revisit of the fixed-point schemes g_1 to g_5 in Example 2.9.

- For $g_3(x) = \frac{1}{2}\sqrt{10 - x^3}$, we have

$$g_3'(x) = -\frac{3}{4}x^2(10 - x^3)^{-1/2} < 0, \quad \text{on } [1, 2]$$

so g_3 is strictly decreasing on $[1, 2]$. If we start with $p_0 = 1.5$, it suffices to consider the interval $[1, 1.5]$. Also note that

$$1 < 1.28 \approx g_3(1.5) \leq g_3(x) \leq g_3(1) = 1.5,$$

so g_3 maps $[1, 1.5]$ into itself. Moreover, it is also true that

$$|g_3'(x)| \leq g_3'(1.5) \approx 0.66,$$

on the interval $[1, 1.5]$, so Theorem 2.10 guarantees its convergence. ($k \approx 0.66$)

A revisit of the fixed-point schemes g_1 to g_5 in Example 2.9.

- For $g_4(x) = \sqrt{\dfrac{10}{4+x}}$, it maps $[1,2]$ into itself. Moreover,

  $$|g_4'(x)| \le |\frac{\sqrt{10}}{2}(4+x)^{-3/2}| \le \frac{\sqrt{10}}{2}\cdot 5^{-3/2} = \frac{1}{5\sqrt{2}} < 0.15, \quad \text{for all } x \in [1,2].$$

  So g_4 converges much more rapidly than g_3 ($k \approx 0.15$).

- For $g_5(x) = x - \dfrac{x^3+4x^2-10}{3x^2+8x}$, it converges much more rapidly than other choices. This choice of the $g_5(x)$ is in fact the Newton's Method, and we will see where this choice came from and why it is so effective in the next section.

Concluding Remark

Question How can we find a fixed-point problem that produces a sequence that reliably and rapidly converges to a solution to a given root-finding problem?

Answer Manipulate the root-finding problem into a fixed point problem that satisfies the conditions of Fixed-Point Theorem (Theorem 2.10) and has a derivative that is as small as possible near the fixed point.

2.3 Newton's Method and Its Extensions

- In this section, we introduce one of the most powerful and well-known numerical methods for root-finding problems, namely **Newton's method (or Newton-Raphson method)**.
- Suppose $f \in C^2[a,b]$. Let $p_0 \in (a,b)$ be an approximation to a root p such that $f'(p_0) \neq 0$. Assume that $|p - p_0|$ is small. By Taylor expansion,

$$f(p) = f(p_0) + (p - p_0)f'(p_0) + \frac{(p-p_0)^2}{2} f''(\xi)$$

  where ξ is between p_0 and p.
- Since $f(p) = 0$,

$$0 = f(p_0) + (p - p_0)f'(p_0) + \frac{(p-p_0)^2}{2} f''(\xi)$$

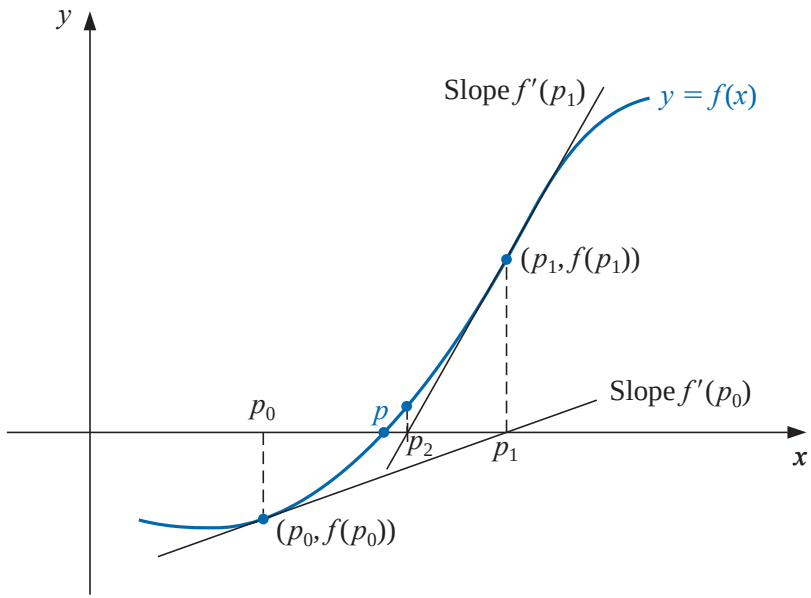
- Since $p - p_0$ is small, we drop the high-order term involving $(p-p_0)^2$,

$$0 \approx f(p_0) + (p - p_0)f'(p_0) \quad \Longrightarrow \quad p \approx p_0 - \frac{f(p_0)}{f'(p_0)} \equiv p_1.$$

Newton's Method

Given an initial approximation p_0, generate a sequence $\{p_n\}_{n=0}^{\infty}$ by

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad \text{for } n \geq 1.$$



- Note that p_n is the x-intercept of the tangent line to f at $(p_{n-1}, f(p_{n-1}))$.
- An animation:

  https://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif

- To program the Newton's method, the inputs should contain f, p_0, tol, $maxit$, as used in the fixed-point methods.
- In addition, we also need to include the derivative f' as an input.

Matlab Code of Newton's Method

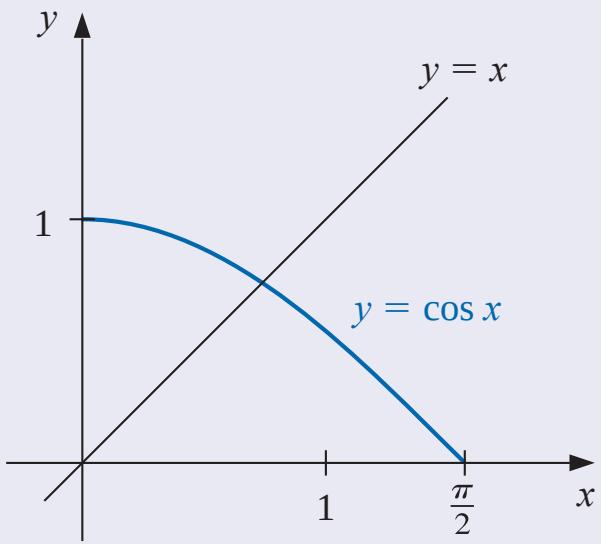
```
function [p,flag] = newton(fun,Dfun,p0,tol,maxIt)

n = 0; flag = 0; % Initializaiton

disp('-----------------------------------')
disp('Newton Method')
disp('-----------------------------------')
disp(' n    p_n              f(p_n)')
disp('-----------------------------------')
formatSpec = '%2d    %.10f    % .10f \n';
fprintf(formatSpec,[n,p0,fun(p0)])

while n<=maxIt
   p = p0 - fun(p0)/Dfun(p0);
   if abs(p-p0) < tol
        flag = 1; break;
   else
       n = n+1; p0 = p;
   end
   fprintf(formatSpec,[n,p,fun(p)])
end
```

Example 11.

Let $f(x) = \cos(x) - x$. Approximate a root of f using (i) the fixed-point method with $g(x) = \cos(x)$ and (ii) Newton's method.

Solution (1/3)

(i). Using the fixed-point function $g(x) = \cos(x)$, we can start the fixed-point iteration with $p_0 = \pi/4$.



Solution (2/3)

(ii). To apply Newton's method, we calculate $f'(x) = -\sin(x) - 1$. We again start with $p_0 = \pi/4$.

- **A MATLAB driver file for this example**

```
% Example 2.3.1 in the Textbook

fun = @(x) cos(x)-x;   % Function f(x)
Dfun = @(x) -sin(x)-1; % Derivative of f(x)
funF = @(x) cos(x);    % Function for fixed point iteration

tol = 1E-10;
maxIt = 20;

%% Fixed-Point Iteration
p0 = pi/4;
[pF,flagF] = fixedpoint(funF,p0,tol,maxIt);
disp(' ')

%% Newton Method
p0 = pi/4;
[p,flag] = newton(fun,Dfun,p0,tol,maxIt);
disp(' ')
```

Solution (3/3)

Fixed Point Iteration

n	p	f(p_n)
0	0.785398163	0.707106781
1	0.707106781	0.760244597
2	0.760244597	0.724667481
3	0.724667481	0.748719886
4	0.748719886	0.732560845
5	0.732560845	0.743464211
6	0.743464211	0.736128257
7	0.736128257	0.741073687
8	0.741073687	0.737744159
9	0.737744159	0.739987765
10	0.739987765	0.738476809
11	0.738476809	0.739494771
12	0.739494771	0.738809134
13	0.738809134	0.739271021
14	0.739271021	0.738959904
15	0.738959904	0.739169483
16	0.739169483	0.739028311
17	0.739028311	0.739123408
18	0.739123408	0.739059350
19	0.739059350	0.739102501
20	0.739102501	0.739073434

Newton Method

n	p_n	f(p_n)
0	0.7853981634	−0.0782913822
1	0.7395361335	−0.0007548747
2	0.7390851781	−0.0000000751
3	0.7390851332	−0.0000000000

- Comparing with the Fixed-point iteration, Newton method gives excellent approximation with only three iterations.

Remarks on Newton's Method

- Newton's method can provide **extremely accurate** approximations with very few iterations.
- Newton's method requires the **initial approximation to be sufficiently accurate**.
- In practical applications, an initial approximation can be obtained by other methods, such as bisection method. After the approximation is sufficient accurate, Newton's method is applied for rapid convergence.
- Newton's method requires evaluation of the derivative f' at each step. Usually f' is far more difficult to calculate than f.

Example 12.

Player A will shut out (win by a score of 21-0) player B in a game of racquetball with probability

$$P = \frac{1+p}{2}\left(\frac{p}{1-p+p^2}\right)^{21},$$

where p denotes the probability A will win any specific rally (independent of the server). Determine the minimum value of p that will ensure that player A will shut out player B in at least half the matches they play.

Solution

- The player A winning at least half of the matches means P is at least 0.5. We consider the root-finding problem

$$f(p) = \frac{1+p}{2}\left(\frac{p}{1-p+p^2}\right)^{21} - 0.5.$$

- The derivative f' is (verify by yourself)

$$f'(p) = \frac{1}{2}\left(\frac{p}{1-p+p^2}\right)^{21} + \frac{21}{2}(1+p)\left(\frac{p}{1-p+p^2}\right)^{20}\frac{1-p^2}{(1-p+p^2)^2}.$$

- Using Newton's method with $p_0 = 0.75$, and

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad \text{for } n \geq 1$$

we find that $p \approx 0.8423$ in three iterations.

- In last example, we see that the finding the derivative $f'(x)$ is not easy, and the evaluation of $f'(x)$ also requires more arithmetic operations than the evaluation of $f(x)$ itself.
- To circumvent this problem, we introduce a variation of Newton's method that does require the evaluation of derivative f'.
- Recall that in Newton's method we have

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad \text{for } n \geq 1$$

- By the definition of derivative,

$$f'(p_{n-1}) = \lim_{x \to p_{n-1}} \frac{f(x) - f(p_{n-1})}{x - p_{n-1}} \approx \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}}$$

since p_{n-2} is close to p_{n-1}.

Secant Method

- Replacing the derivative $f'(p_{n-1})$ in the Newton's formula by the difference quotient, we have

$$\begin{aligned} p_n &= p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \\ &\approx p_{n-1} - \frac{f(p_{n-1})}{\dfrac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}}} \\ &= p_{n-1} - \frac{f(p_{n-1})(p_{n-2} - p_{n-1})}{f(p_{n-2}) - f(p_{n-1})} \qquad n \geq 2. \end{aligned}$$

Secant Method

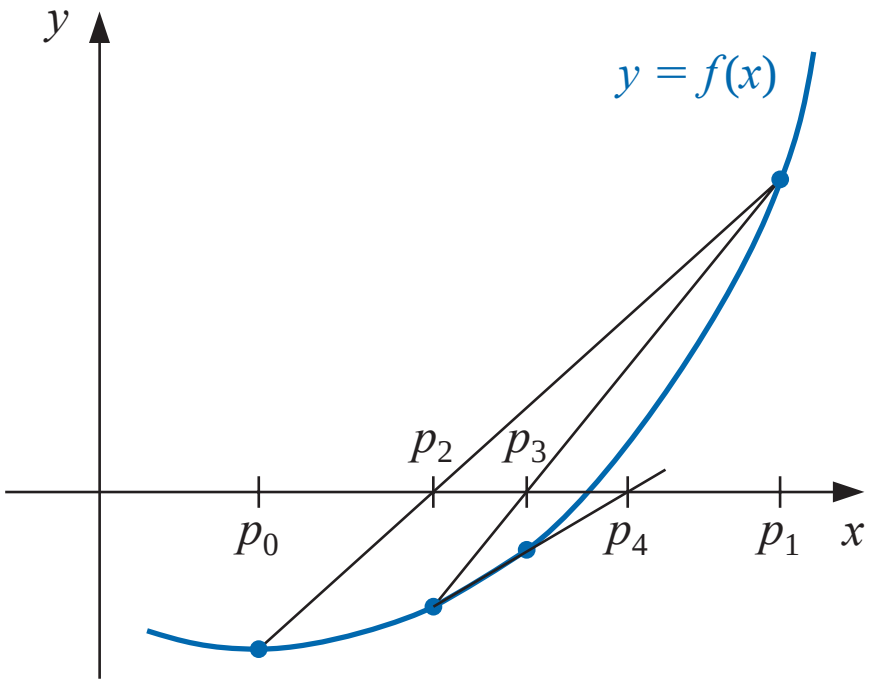
Given initial approximations p_0 and p_1, generate a sequence $\{p_n\}_{n=0}^{\infty}$ by

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-2} - p_{n-1})}{f(p_{n-2}) - f(p_{n-1})}, \quad n \geq 2.$$

Remark

- The Secant method requires two initial approximations.
- However, it does not require the evaluation of the derivative.

- An illustration of Secant method



- Starting with two initial approximations p_0 and p_1, the value p_2 is the x-intercept of the line joining $(p_0, f(p_0))$ and $(p_1, f(p_1))$.
- The approximation p_3 is the x-intercept of the line joining $(p_1, f(p_1))$ and $(p_2, f(p_2))$ and so on.

Matlab Code of Secant Method

```
function [p,flag] = secant(fun,p0,p1,tol,maxIt)

n = 1; flag = 0; % Initializaiton
q0 = fun(p0);  q1 = fun(p1);

disp('-----------------------------------')
disp('Secant Method')
disp('-----------------------------------')
disp(' n    p_n              f(p_n)')
disp('-----------------------------------')
formatSpec = '%2d    %.10f    % .10f \n';
fprintf(formatSpec,[n-1,p0,fun(p0)])
fprintf(formatSpec,[n,p1,fun(p1)])

while n<=maxIt
   p = p1 - q1*(p1-p0)/(q1-q0);
   if abs(p-p0) < tol
        flag = 1; break;
   else
       n = n+1;
       p0 = p1; q0 = q1; p1 = p; q1 = fun(p);
   end
   fprintf(formatSpec,[n,p,fun(p)])
end
```

Example 13.

Use the Secant method for find a solution to $x = \cos(x)$, and compare with the approximation with those given from Newton's method.

Solution (1/2)

Write a MATLAB driver file

```
% Example 2.3.2 in the Textbook

fun = @(x) cos(x)-x;
Dfun = @(x) -sin(x)-1;

tol = 1E-10;
maxIt = 40;
%% Newton
p0 = pi/4;
[pN,flagN] = newton(fun,Dfun,p0,tol,maxIt);
disp(' ')

%% Secant
p0 = 0.5; p1 = pi/4;
[pS,flagS] = secant(fun,p0,p1,tol,maxIt);
disp(' ')
```

Solution (2/2)

```
>> ex2_3_2
-----------------------------------
Newton Method
-----------------------------------
 n    p_n              f(p_n)
-----------------------------------
 0    0.7853981634    -0.0782913822
 1    0.7395361335    -0.0007548747
 2    0.7390851781    -0.0000000751
 3    0.7390851332    -0.0000000000

-----------------------------------
Secant Method
-----------------------------------
 n    p_n              f(p_n)
-----------------------------------
 0    0.5000000000     0.3775825619
 1    0.7853981634    -0.0782913822
 2    0.7363841388     0.0045177185
 3    0.7390581392     0.0000451772
 4    0.7390851493    -0.0000000270
 5    0.7390851332     0.0000000000
 6    0.7390851332     0.0000000000
```

- Secant method requires 5 iterations comparing with 3 iteration used in Newton's method.

Example 14.

A revisit of Example (Recquetball Winning Probability) use Secant method.

Solution

- The root-finding problem is

$$f(p) = \frac{1+p}{2}\left(\frac{p}{1-p+p^2}\right)^{21} - 0.5.$$

- Use Secant method with $p_0 = 0.5$, and $p_1 = 1$, we can find $p \approx 0.8423$ within accuracy of 10^{-5} in five iterations.

Remark

- Newton's method uses three iterations to reach this accuracy. However, it requires evaluations of the derivative f'.

Remark

- Secant Method converges slightly slower than Newton Method, but much faster than other Fixed-point iterations.
- Newton's method or the Secant method is often used to refine an answer obtained by another technique, such as the Bisection method, since these methods require good first approximations but generally give rapid convergence.

2.4 Error Analysis for Iterative Methods

- In this section we investigate the order of convergence of iteration schemes.
- For example, the following sequences all converge to 0 as $n \to \infty$

$$\left\{\frac{1}{n}\right\}, \quad \left\{\frac{1}{n^2}\right\}, \quad \left\{\frac{1}{e^n}\right\}, \quad \left\{\frac{1}{n!}\right\}.$$

  Clearly, the "speed" of the convergence is different.
- We will develop a procedure for measuring how rapidly a sequence converges.

Definition 15 (Order of Convergence).

Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p, with $p_n \neq p$ for all n.

- If $\lim_{n\to\infty} \dfrac{|p_{n+1} - p|}{|p_n - p|} = \lambda$, where $\lambda \in (0, 1)$, then $\{p_n\}$ is said to **converge linearly**, with asymptotic error constant λ.
- If $\lim_{n\to\infty} \dfrac{|p_{n+1} - p|}{|p_n - p|} = 0$, then $\{p_n\}$ is said to **converge superlinearly**.
- If $\lim_{n\to\infty} \dfrac{|p_{n+1} - p|}{|p_n - p|} = 1$, then $\{p_n\}$ is said to **converge sublinearly**.

Remark

To further distinguish superlinear convergences, we say the sequence $\{p_n\}$ converges to p of order $\alpha > 1$ if

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = M.$$

In particular,

- $\alpha = 2$ is called to **quadratic convergence**.
- $\alpha = 3$ is called to **cubic convergence**.

Example 16.

The following sequences all converge to 0. Find the convergence order of each sequence.

$$(a).\ \left\{\frac{1}{n}\right\} \quad (b).\ \left\{\frac{1}{n^2}\right\} \quad (c).\ \left\{\frac{1}{2^n}\right\} \quad (d).\ \left\{\frac{1}{n!}\right\} \quad (e).\ \left\{\frac{1}{2^{2^n}}\right\}$$

Solution (1/4)

(a). For $\left\{\frac{1}{n}\right\}$, the first few terms are $1,\ \frac{1}{2},\ \frac{1}{3},\ \frac{1}{4},\ \frac{1}{5}, \cdots$

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|} = \lim_{n\to\infty} \frac{\frac{1}{n+1}}{\frac{1}{n}} = \lim_{n\to\infty} \frac{n}{n+1} = 1.$$

The sequence $\left\{\frac{1}{n}\right\}$ converges to 0 **sublinearly**.

Solution (2/4)

(b). For $\left\{\frac{1}{n^2}\right\}$, the first few terms are $1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \cdots$

$$\lim_{n\to\infty}\frac{|p_{n+1}-p|}{|p_n-p|}=\lim_{n\to\infty}\frac{\frac{1}{(n+1)^2}}{\frac{1}{n^2}}=\lim_{n\to\infty}\frac{n^2}{(n+1)^2}=1.$$

The sequence $\left\{\frac{1}{n^2}\right\}$ converges to 0 **sublinearly**.

(c). For $\left\{\frac{1}{2^n}\right\}$, the first few terms are $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \cdots$

$$\lim_{n\to\infty}\frac{|p_{n+1}-p|}{|p_n-p|}=\lim_{n\to\infty}\frac{\frac{1}{2^{n+1}}}{\frac{1}{2^n}}=\lim_{n\to\infty}\frac{2^n}{2^{n+1}}=\frac{1}{2}.$$

The sequence $\left\{\frac{1}{2^n}\right\}$ converges to 0 **linearly**.

Solution (3/4)

(d). For $\left\{\frac{1}{n!}\right\}$, the first few terms are $1,\ \frac{1}{2},\ \frac{1}{6},\ \frac{1}{24},\ \frac{1}{120}, \cdots$

$$\lim_{n\to\infty}\frac{|p_{n+1}-p|}{|p_n-p|}=\lim_{n\to\infty}\frac{\frac{1}{(n+1)!}}{\frac{1}{n!}}=\lim_{n\to\infty}\frac{n!}{(n+1)!}=\lim_{n\to\infty}\frac{1}{n+1}=0.$$

The sequence $\left\{\frac{1}{n!}\right\}$ converges to 0 **superlinearly**.

Note that for any $a>1$,

$$\lim_{n\to\infty}\frac{|p_{n+1}-p|}{|p_n-p|^a}=\lim_{n\to\infty}\frac{(n!)^a}{(n+1)!}=\lim_{n\to\infty}\frac{(n!)^{a-1}}{n+1}\to\infty.$$

The convergence order of $\left\{\frac{1}{n!}\right\}$ is barely 1, but not any more.

Solution (4/4)

(e). For $\left\{\frac{1}{2^{2^n}}\right\}$, the first few terms are $\frac{1}{4}, \frac{1}{16}, \frac{1}{256}, \frac{1}{65536}, \frac{1}{4294967296}, \cdots$

$$\lim_{n\to\infty} \frac{\frac{1}{2^{2^{n+1}}}}{\frac{1}{2^{2^n}}} = \lim_{n\to\infty} \frac{2^{2^n}}{2^{2^{n+1}}} = \lim_{n\to\infty} \frac{2^{2^n}}{2^{2\cdot 2^n}} = \lim_{n\to\infty} \frac{2^{2^n}}{(2^{2^n})^2} = \lim_{n\to\infty} \frac{1}{2^{2^n}} = 0.$$

The sequence $\left\{\frac{1}{2^{2^n}}\right\}$ converges to 0 **superlinearly**.

Moreover, we note that

$$\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^2} = \lim_{n\to\infty} \frac{\frac{1}{2^{2^{n+1}}}}{(\frac{1}{2^{2^n}})^2} = \lim_{n\to\infty} \frac{(2^{2^n})^2}{2^{2^{n+1}}} = \lim_{n\to\infty} \frac{(2^{2^n})^2}{(2^{2^n})^2} = 1.$$

The sequence $\left\{\frac{1}{2^{2^n}}\right\}$ converges to 0 **quadratically**.

Comparison of Linear and Quadratic Convergences

n	Linear Convergence Sequence $\{p_n\}_{n=0}^{\infty}$ $(0.5)^n$	Quadratic Convergence Sequence $\{\tilde{p}_n\}_{n=0}^{\infty}$ $(0.5)^{2^n-1}$
1	5.0000×10^{-1}	5.0000×10^{-1}
2	2.5000×10^{-1}	1.2500×10^{-1}
3	1.2500×10^{-1}	7.8125×10^{-3}
4	6.2500×10^{-2}	3.0518×10^{-5}
5	3.1250×10^{-2}	4.6566×10^{-10}
6	1.5625×10^{-2}	1.0842×10^{-19}
7	7.8125×10^{-3}	5.8775×10^{-39}

- Quadratically convergent sequences are expected to converge much quicker than those that converge only linearly.
- It usually takes 5 or 6 iterations for a quadratic convergent sequence to reach the 64-bit machine precision.

Convergence Order of Bisection Method

- We have shown in Theorem 2.3 that the sequence $\{p_n\}$ of bisection method satisfies

$$|p_n - p| \leq \frac{b-a}{2^n}.$$

- The absolute error $e_n = |p_n - p|$ "behaves" like the sequence

$$e_n \approx \frac{1}{2^n}, \qquad \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|} \approx \frac{1}{2}.$$

- Bisection Method converges linearly with asymptotic constant $\frac{1}{2}$.

Convergence Order of Newton Method

Newton's Method $\quad p_{n+1} = p_n - \dfrac{f(p_n)}{f'(p_n)}.$

- Let $e_n \triangleq p_n - p$, by Taylor's theorem

$$f(p) = f(p_n - e_n) = f(p_n) - e_n f'(p_n) + \frac{e_n^2}{2} f''(\xi_n).$$

- Since $f(p) = 0$, $f'(p) \neq 0$ (so $f'(p_n) \neq 0$ when p_n is close to p), then

$$0 = \frac{f(p_n)}{f'(p_n)} - e_n + \frac{e_n^2}{2f'(p_n)} f''(\xi_n) = \frac{f(p_n)}{f'(p_n)} - p_n + p + \frac{e_n^2}{2f'(p_n)} f''(\xi_n)$$

$$\Longrightarrow \quad p_{n+1} \triangleq p_n - \frac{f(p_n)}{f'(p_n)} = p + \frac{e_n^2}{2f'(p_n)} f''(\xi_n)$$

- That is

$$e_{n+1} = \frac{f''(\xi_n)}{2f'(p_n)} e_n^2 \quad \Longrightarrow \quad |e_{n+1}| \leq M|e_n|^2, \quad \text{where} \quad M = \frac{|f''(p)|}{2|f'(p)|}.$$

Thus, Newton Method converges **quadratically**.

Convergence Order of Secant Method

Secant Method $$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f'(p_{n-1}) - f(p_{n-2})}.$$

It can be shown that

$$|e_n| \approx C|e_{n-1}|^{\alpha}, \quad \text{where } \alpha = \frac{\sqrt{5}+1}{2} \approx 1.618$$

Thus, Secant Method converges **superlinearly**, with an order of 1.618.

Remark

- For a complete proof, see

  `http://www1.maths.leeds.ac.uk/~kersale/2600/Notes/appendix_D.pdf`

- The Secant method converges much faster than Bisection method but slower than Newton method.

Convergence Order of Fixed-point Iteration

- Recall that a root-finding problem $f(x) = 0$ can be converted to a fixed-point iteration $g(p) = p$.
- The fixed-point iteration is given p_0,

$$p_n = g(p_{n-1}) \quad \text{for } n \geq 1$$

- It has been shown that

$$|p_n - p| \leq \frac{k^n}{1-k}|p_1 - p_0| \quad \text{where } 0 < k < 1.$$

- Thus, Fixed-point iteration (if it converges) converges at least linearly, with asymptotic constant at most k.

Multiple Roots

- Finally we consider problem with repeated roots such as

$$f(x) = (x-1)^3(x+2)(x-3)^2.$$

- When we apply Newton's method to find a multiple root, we can still expect convergence, but the convergence order is usually less than quadratic.

- A solution p of $f(x) = 0$ is a **zero of multiplicity** m of f if

$$f(x) = (x-p)^m g(x), \quad \text{where} \quad g(p) \neq 0.$$

- The function f has a **simple zero** if and only if $f(p) = 0$ and $f'(p) \neq 0$.

Example 17.

Let $f(x) = e^x - x - 1$. (a). Show that f has a zero of multiplicity 2 at $x = 0$. (b). Show that Newton's method with $p_0 = 1$ converges to this zero but not quadratically.

Solution(1/2)

(a). Note that

$$f(x) = e^x - x - 1, \quad f'(x) = e^x - 1, \quad f''(x) = e^x.$$

Thus

$$f(0) = e^0 - 0 - 1 = 0, \quad f'(0) = e^0 - 1 = 0, \quad f''(0) = e^0 = 1.$$

Thus, the root $p = 0$ is a zero of multiplicity 2.

Solution(2/2)

(b). We test the convergence of Newton's method

```
>> ex2_4_1
-----------------------------------
Newton Method
-----------------------------------
 n    p_n              f(p_n)
-----------------------------------
 0    1.0000000000     0.7182818285
 1    0.5819767069     0.2075956900
 2    0.3190550409     0.0567720087
 3    0.1679961729     0.0149359105
 4    0.0863488737     0.0038377257
 5    0.0437957037     0.0009731870
 6    0.0220576854     0.0002450693
 7    0.0110693875     0.0000614924
 8    0.0055449047     0.0000154014
 9    0.0027750145     0.0000038539
10    0.0013881490     0.0000009639
11    0.0006942351     0.0000002410
12    0.0003471577     0.0000000603
13    0.0001735889     0.0000000151
14    0.0000867970     0.0000000038
15    0.0000433991     0.0000000009
16    0.0000216997     0.0000000002
17    0.0000108499     0.0000000001
18    0.0000054250     0.0000000000
19    0.0000027125     0.0000000000
20    0.0000013563     0.0000000000
21    0.0000006782     0.0000000000
22    0.0000003390     0.0000000000
23    0.0000001700     0.0000000000
24    0.0000000851     0.0000000000
25    0.0000000408     0.0000000000
26    0.0000000190     0.0000000000
27    0.0000000073     0.0000000000
```

The convergence is much slower than quadratic, as we expect from Newton.

- To fix the problem for repeated roots, we consider the function

$$\mu(x) = \frac{f(x)}{f'(x)}.$$

- If p is a zero of $f(x)$ with multiplicity m, then $f(x) = (x-p)^m g(x)$, and

$$\begin{aligned} \mu(x) &= \frac{(x-p)^m g(x)}{m(x-p)^{m-1}g(x) + (x-p)^m g'(x)} \\ &= (x-p)\frac{g(x)}{mg(x) + (x-p)g'(x)}. \end{aligned}$$

Since $g(p) \neq 0$, then p is a simple zero of $\mu(x)$.

- Now to find the zero p, we apply Newton's method to $\mu(x)$,

$$\begin{aligned} g(x) &= x - \frac{\mu(x)}{\mu'(x)} \\ &= x - \frac{f(x)/f'(x)}{\dfrac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2}} \\ &= x - \frac{f(x)f'(x)}{[f'(x)]^2 - f(x)f''(x)}. \end{aligned}$$

Modified Newton's Method (for multiple roots)

Given an initial approximation p_0, generate a sequence $\{p_n\}_{n=0}^{\infty}$ by

$$p_n = p_{n-1} - \frac{f(p_{n-1})f'(p_{n-1})}{[f'(p_{n-1})]^2 - f(p_{n-1})f''(p_{n-1})}, \quad \text{for } n \geq 1.$$

Note: The modified Newton' method requires the second-order derivative $f''(x)$.

Example 18.

Solve $f(x) = e^x - x - 1$ by modified Newton's method.

Solution

We test the Modified Newton's method

```
-----------------------------------
Modified Newton Method
-----------------------------------
 n    p_n                 f(p_n)
-----------------------------------
 0     1.0000000000     0.7182818285
 1    -0.2342106136     0.0254057755
 2    -0.0084582799     0.0000356706
 3    -0.0000118902     0.0000000001
 4    -0.0000000000     0.0000000000
```

The quadratic convergence is recovered.