Lecture 2

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

1 Solving Nonlinear Equations

We are interested in finding the solutions x of the equation

$$f(x) = 0$$

such as

$$e^x - \sin x = 0$$

The following methods will be introduced: 1. Bisection Method 2. Fixed-Point Iteration 3. Newton's Method 4. Secant Method and Variants

In this lecture, we will focus on bisection and fixed-point interation.

1.1 Bisection Method

Definition 1.1 *The function $f(x)$ has a **root** at $x = r$ if $f(r) = 0$.*

The Bisection Method is based on the following theorem:

Theorem 1.1 *Let f be a continuous function on $[a, b]$, satisfying $f(a)f(b) < 0$. Then f has a root between a and b, that is, there exists a number r satisfying $a < r < b$ and $f(r) = 0$.*

The bisection method starts with checking if a solution exists for sure on the given interval by the theorem above, then the solution is refined gradually by reducing the interval where the solution potentially exists.

The algorithm for the Bisection Method is:

```
Given initial interval [a,b] such that f(a)f(b)<0
while (b-a)/2 > TOL
    c=(a+b)/2
    if f(c)=0,
        stop
    end
    if f(a)f(c)<0
        b=c
```

```
    else
        a=c
    end
end
The final interval [a,b] contains a root.
The approximate root is (a+b)/2.
```

What the algorithm does is:

1. Check the value of the function at the midpoint $c = (a+b)/2$ of the interval.
2. Since $f(a)$ and $f(b)$ have opposite signs, either $f(c) = 0$ (in which case we have found a root and are done), or the sign of $f(c)$ is opposite the sign of either $f(a)$ or $f(b)$.
3. If $f(c)f(a) < 0$, for example, we are assured a solution in the interval $[a, c]$, whose length is half that of the original interval $[a, b]$. If instead $f(c)f(b) < 0$, we can say the same of the interval $[c, b]$. In either case, one step reduces the problem to finding a root on an interval of one-half the original size. This step can be repeated to locate the function more and more accurately.

Example 1.1 *Find a root of the function* $f(x) = x^3 + x - 1$ *by using the Bisection Method on the interval* $[0, 1]$.

Solution:

i	a_i	$f(a_i)$	c_i	$f(c_i)$	b_i	$f(b_i)$
0	0.0000	-	0.5000	-	1.0000	+
1	0.5000	-	0.7500	+	1.0000	+
2	0.5000	-	0.6250	-	0.7500	+
3	0.6250	-	0.6875	+	0.7500	+
4	0.6250	-	0.6562	-	0.6875	+
5	0.6562	-	0.6719	-	0.6875	+
6	0.6719	-	0.6797	-	0.6875	+
7	0.6797	-	0.6836	+	0.6875	+
8	0.6797	-	0.6816	-	0.6836	+
9	0.6816	-	0.6826	+	0.6836	+

The table shows that the solution is bracketed between a_9 and c_9. Taking the average, the approximate solution is 0.6821. Taking into the error into consideration, the root is $r = 0.6821 \pm 0.0005$.

[2]:
```
# Bisection Method
def bisect(f, a, b, tol):
    """
    f: the function
    a: left end
    b: right end
    tol: tolerance to control when to stop the algo
    """
    # Check if the algorithm needs to proceed
```

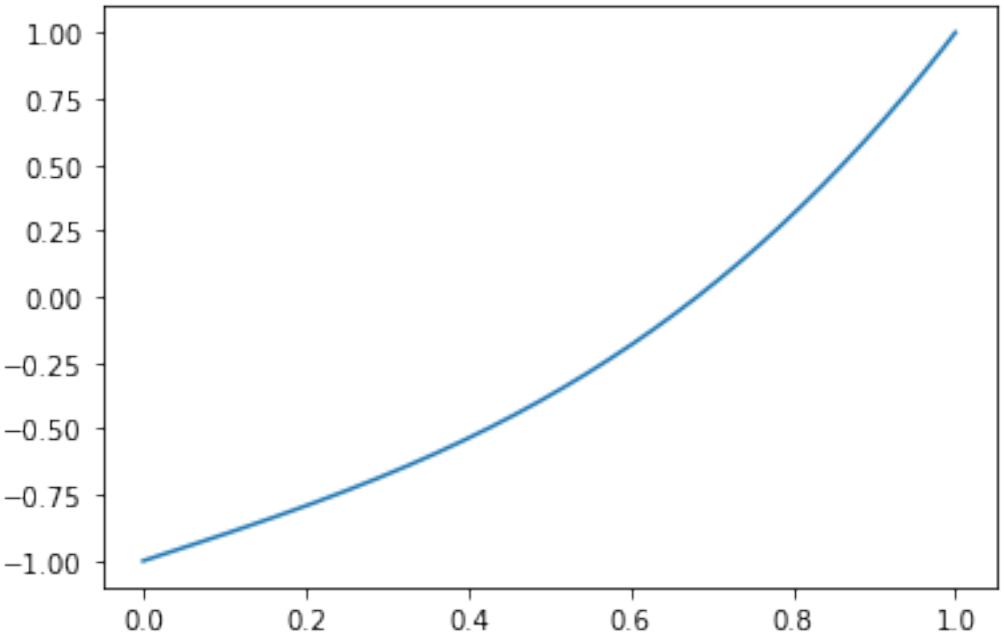
```
    if f(a) == 0:
        print('The solution is ', a)
        stop

    if f(b) == 0:
        print('The solution is ', b)
        stop

    if np.sign(f(a))*np.sign(f(b)) == 1:
        print('f(a)f(b) is not satisfied')
        stop

    # Start the algorithm
    fa = f(a)
    fb = f(b)
    while (b-a)/2 > tol:
        c = (a+b)/2
        fc = f(c)
        if fc == 0:
            break
        if np.sign(fc)*np.sign(fa)<0:
            # a and c form the new interval
            b = c
            fb = fc
        else:
            a = c
            fa = fc

    return (a+b)/2
```

Example 1.2 *Use the bisection method to find a solution of*

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

on the interval $[0, 1]$.

[3]:
```
# First define the function:
def f_ex2(x):
    """
    The f from example 2
    """
    return x**3+x-1

# check what the graph looks like
x = np.linspace(0, 1, 1000)
plt.plot(x, f_ex2(x));
```



[4]:
```
# Call the function, suppose tol=1e-5
tol = 1e-5
xc = bisect(f_ex2, 0, 1, tol)
print('The solution is ', xc)
# Check if the solution is right
print('The value of the function at x = ',
      xc, ' is ', f_ex2(xc))
```

```
The solution is  0.6823348999023438
The value of the function at x =  0.6823348999023438  is  1.7007361593268655e-05
```

1.1.1 Speed of convergence of the solution

- Let $[a, b]$ be the starting interval. After n bisection steps, the interval $[a_n, b_n]$ has length $(b - a)/2^n$
- Choosing the midpoint $x_c = (a_n + b_n)/2$ as the approximate solution r, then the true solution is between r and a_n, or r and b_n. Thus,

$$\text{Solution error } = |x_c - r| < \frac{b-a}{2^{n+1}}$$

- After n steps, the computational cost, represented by the number of function evaluations, is $n + 2$.

Definition 1.2 *A solution is* ***correct within*** p ***decimal places*** *if the error is less than* 0.5×10^{-p}.

Since we know the error of the approximate solution with the bisection method, we can estimate how many bisection steps are needed to achieve a given accuracy.

Example 1.3 *How many steps are needed to find a root of $f(x) = \cos x - x$ in the interval $[0, 1]$ within 6 correct places using Bisection Method?*

Solution

We know

$$|x_c - r| < \frac{b-a}{2^{n+1}}$$

So we want

$$\frac{b-a}{2^{n+1}} < 0.5 \times 10^{-6}$$

i.e.,

$$\frac{1}{2^{n+1}} < 0.5 \times 10^{-6}$$

So

$$2^{n+1} > 2 \times 10^6$$

and

$$n > \log_2(2 \times 10^6) - 1 = 19.9$$

So $n = 20$ steps are needed.

Additional Exercises

Exercise 1.1 *Use the Intermediate Value Theorem to find an interval of length one that contains a root of the equation*

(a) $x^3 = 9$

(b) $3x^3 + x^2 = x + 5$

(c) $\cos x^2 + 6 = x$

Exercise 1.2 *Consider the equations in Exercise 1. Apply two steps of the Bisection Method to find an approximate root.*

1.2 Fixed-Point Iteration

Now we will switch to the second method to solve linear equations–fixed-point iteration

Definition 1.3 *The real number r is a **fixed point** of the function g if $g(r) = r$.*

For example, the function $g(x) = x^3$ has three fixed points, $r = -1, 0, 1$.

If an equation $f(x) = 0$ can be written as $g(x) = x$, then Fixed-Point Iteration proceeds by starting with an initial guess x_0 and iterating the function g:

Fixed-Point Iteration

$$x_0 = \text{ initial guess}$$
$$x_{i+1} = g(x_i) \text{ for } i = 0, 1, 2, \ldots$$

i.e.,

$$x_1 = g(x_0)$$
$$x_2 = g(x_1)$$
$$x_3 = g(x_2)$$
$$\vdots$$

The sequence x_i may or may not converge as the number of steps goes to infinity. However, if g is continuous and the x_i converge, say, to a number r, then r is a fixed point. This is because:

$$g(r) = g(\lim_{i\to\infty} x_i) = \lim_{i\to\infty} g(x_i) = \lim_{i\to\infty} x_{i+1} = r$$

[5]:
```
# Function for fixed-point iteration
# if f(x)=0 can be written as g(x)=x

def fpi(g,x_0,k):
    """
    g: the function g(x) in g(x)=x
    x_0: the initial guess
    k: the maximum number of interation steps
    """
    x = np.zeros(k+1,)
    x[0] = x_0

    for i in range(k):
        x[i+1] = g(x[i])

    return x[k]
```

Example 1.4 *Solve the equation*

$$f(x) = \cos x - x = 0$$

using fixed-point iteration starting with $x_0 = 0$

Solution

The original equation can be written as

$$\cos x = x$$

So $g(x) = \cos x$.

[12]:
```
# Define g first
def g_ex4(x):
    return np.cos(x)

# Applying fixed-point iteration
x_0 = 0
k = 1000
```

```
x_fpi = fpi(g_ex4, x_0, k)
print('The approximate solution of f(x)=cosx-x after ',
     k, ' steps of fixed point iteration is ',
     x_fpi)

# Check if the result is a solution of f(x)
print()
print('The value of f at ', x_fpi, ' is: ',
      g_ex4(x_fpi)-x_fpi)
```

```
The approximate solution of f(x)=cosx-x after  1000  steps of fixed point
iteration is  0.7390851332151605

The value of f at  0.7390851332151605  is:  3.3306690738754696e-16
```

Example 1.5 *Find the solution of*

$$x^3 + x - 1 = 0$$

using fixed-point iteration.

Solution:

There are different ways to rewrite f as $g(x) = x$, such as

$$x = 1 - x^3$$
$$x = \sqrt[3]{1 - x}$$

A third way is obtained by adding $2x^3$ to both sides of $f(x) = 0$:

$$\begin{aligned} 3x^3 + x - 1 &= 2x^3 \Rightarrow \\ 3x^3 + x &= 1 + 2x^3 \Rightarrow \\ (3x^2 + 1)x &= 1 + 2x^3 \Rightarrow \\ x &= \frac{1 + 2x^3}{1 + 3x^2} \end{aligned}$$

We first try the first choice, and show the results for $0 - 12$ steps of iterations. We choose the initial value to be 0.5.

[18]:
```
# Define g for the first choice in Example 7
def g_ex5_ch1(x):
    return 1-x**3

x_0 = 0.5
k_max = 12
res = np.zeros(k_max, )
res[0] = x_0
for k in range(1, k_max):
    res[k] = fpi(g_ex5_ch1, x_0, k)
```

```
print('   i   ', '   xi   ')
for i in range(0, k_max):
    print("{0:4d} {1:10f}".format(i, res[i]))
```

```
   i       xi
   0   0.500000
   1   0.875000
   2   0.330078
   3   0.964037
   4   0.104054
   5   0.998873
   6   0.003376
   7   1.000000
   8   0.000000
   9   1.000000
  10   0.000000
  11   1.000000
```

So the iteration does not converge, and tends to alternate between the numbers 0 and 1. Neither 0 or 1 is a fixed point. So the method fails. For Bisection Method, we know the a root will be found since $f(0)f(1) < 0$. This is not the case for fixed-point iteration.

For the second choice:

[21]:
```
# Define g for the second choice in Example 7
def g_ex5_ch2(x):
    return (1-x)**(1/3)

x_0 = 0.5
k_max = 12
res = np.zeros(k_max, )
res[0] = x_0
for k in range(1, k_max):
    res[k] = fpi(g_ex5_ch2, x_0, k)

print('   i   ', '   xi   ')
for i in range(0, k_max):
    print("{0:4d} {1:10f}".format(i, res[i]))
```

```
   i       xi
   0   0.500000
   1   0.793701
   2   0.590880
   3   0.742364
   4   0.636310
   5   0.713801
   6   0.659006
   7   0.698633
   8   0.670448
```

```
  9   0.690729
 10   0.676259
 11   0.686646
```

So the method works.

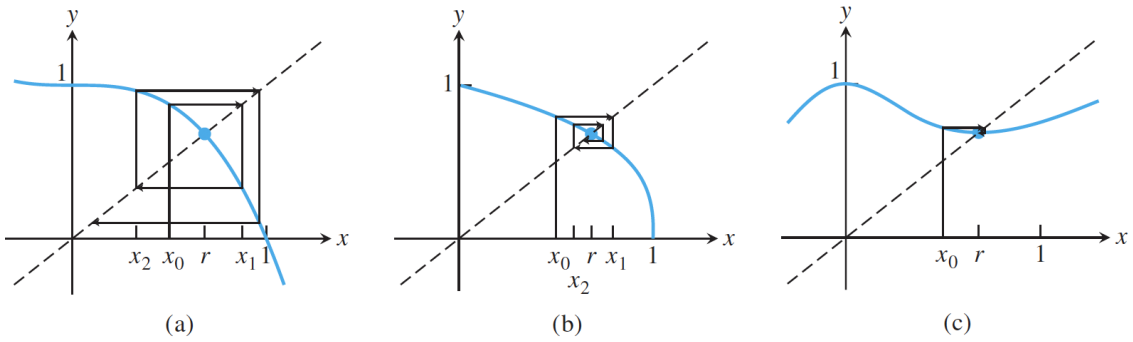
The third choice:

[23]:
```
# Define g for the second choice in Example 7
def g_ex5_ch3(x):
    return (1+2*x**3)/(1+3*x**2)

x_0 = 0.5
k_max = 12
res = np.zeros(k_max, )
res[0] = x_0
for k in range(1, k_max):
    res[k] = fpi(g_ex5_ch3, x_0, k)

print('   i    ', '   xi    ')
for i in range(0, k_max):
    print("{0:4d} {1:10f}".format(i, res[i]))
```

```
   i       xi
   0   0.500000
   1   0.714286
   2   0.683180
   3   0.682328
   4   0.682328
   5   0.682328
   6   0.682328
   7   0.682328
   8   0.682328
   9   0.682328
  10   0.682328
  11   0.682328
```

This method also works, and seems to converge faster.

We now look into why such difference exists:



The three figures show the three choices of g in Example 5 and their corresponding fixed-point iterations.

For example, in Figure (a), the path starts at $x_0 = 0.5$, and moves up to the function and horizontal to the point $(0.875, 0.875)$ on the diagonal, which is (x_1, x_1). Next, $x1$ should be substituted into $g(x)$. This is done the same way it was done for x_0, by moving vertically to the function. This yields $x_2 \approx 0.3300$, and after moving horizontally to move the y-value to an x-value, we continue the same way to get $x_3, x_4, \ldots$. The result is not successful—the iterates eventually tend toward alternating between 0 and 1.

However, we can see the other two choices converge to the solution, and (c) converges much faster.

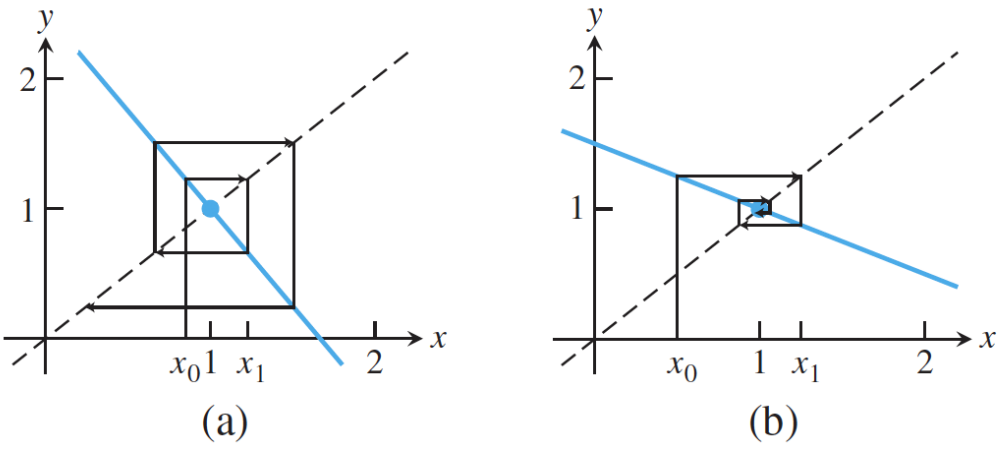
To see why this happens, we consider a simple equation:

$$f(x) = x - 1 = 0$$

For nontrivial choices of $g(x)$, we consider two choices:

$$g_1(x) = -\frac{3}{2}x + \frac{5}{2}$$
$$g_2(x) = -\frac{1}{2}x + \frac{3}{2}$$

Here $|g_1'(1)| = |-\frac{3}{2}| > 1$ while $|g_2'(1)| = |-\frac{1}{2}| < 1$.



Because the slope of g_1 at the fixed point is greater than one, the vertical segments, the ones that represent the change from x_n to x_{n+1}, are increasing in length as FPI proceeds. As a result, the iteration "spirals out". For g_2, the situation is reversed: FPI "spirals in" toward the solution.

We now see how the error $x_i - r$ evolves. Note

$$g_1(x) = -\frac{3}{2}x + \frac{5}{2}$$

Rewriting it in the form of $g_1(x) - r$ and $x - r$ ($r = 1$):

$$g_1(x) - 1 = -\frac{3}{2}x + \frac{3}{2} = -\frac{3}{2}(x - 1)$$

Let $x = x_i$, then

$$g_1(x_i) - 1 = x_{i+1} - 1 = -\frac{3}{2}(x_i - 1)$$

Let $e_i = |r - x_i|$ be the error at step i. Then the previous equation means

$$e_{i+1} = \frac{3}{2}e_i$$

which shows FPI diverges.

However, for g_2:

$$g_2(x) - 1 = -\frac{1}{2}(x - 1)$$

So

$$e_{i+1} = \frac{1}{2}e_i$$

and FPI converges.

Definition 1.4 *Let e_i denote the error at step i of an iterative method. If*

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = S < 1$$

*the method is said to obey **linear convergence** with rate S.*

For example, for fixed-point iteration, g_2 is linearly convergent to the root $r = 1$ with rate $S = \frac{1}{2}$.

Theorem 1.2 *Assume that g is continuously differentiable, that $g(r) = r$, and that $S = |g'(r)| < 1$. Then fixed-point iteration converges linearly with rate S to the fixed point r for initial guesses sufficiently close to r.*

Proof 1.1 *Suppose x_i is the result at step i. Then*

$$g(x_i) - r = g(x_i) - g(r) = x_{i+1} - r = g'(c_i)(x_i - r)$$

for some $c_i \in [x_i, r]$ by the Mean Value Theorem. So using the definition of e_i, we have

$$e_{i+1} = |g'(c_i)|e_i$$

Since $S = |g'(r)| < 1$, by the continuity of g', there is a small neighborhood around r for which

$$|g'(x)| < \frac{S + 1}{2}$$

where $S < \frac{S+1}{2} < 1$. *If* x_i *happens to lie in this neighborhood, so does* c_i. *So*

$$e_{i+1} \leq \frac{S+1}{2} e_i$$

That is, the error drops by a factor of $\frac{S+1}{2}$ *or better every step. So we have*

$$\lim_{i \to \infty} x_i = r$$

In addition,

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = \lim_{i \to \infty} |g'(c_i)| = |g'(r)| = S$$

Definition 1.5 *An iterative method is called* ***locally convergent*** *to* r *if the method converges to* r *for initial guesses sufficiently close to* r.

In other words, the method is locally convergent to the root r if there exists a neighborhood $(r - \epsilon, r + \epsilon)$, where $\epsilon > 0$, such that convergence to r follows from all initial guesses from the neighborhood.

So fixed-point iteration is a locally convergent method.

Example 1.6 *Explain why the first choice in Example 5 does not converge and the second and third choices converge.*

Solution:

Since $g_1'(x) = -3x^2$,

$$|g_1'(0.6823)| = |-3(0.6823)^2| \approx 1.3966 > 1$$

So the iteration does not converge.

For g_2, $g_2'(x) = -\frac{1}{3}(1-x)^{-\frac{2}{3}}$, *so*

$$|g_2'(0.6823)| \approx 0.716 < 1$$

So the iteration converges.

For g_3, $g_3'(x) = -\frac{6x^2(1+3x^2) - (1+2x^3)6x}{(1+3x^2)^2}$, *so*

$$|g_3'(0.6823)| = 0$$

So the iteration converges, and converges very fast.

Example 1.7 *Explain why the fixed-point iteration* $g(x) = \cos x$ *converges.*

Solution

$$|g'(r)| \approx |-\sin 0.74| \approx 0.67 < 1$$

Example 1.8 *Use fixed-point iteration to find a root of* $\cos x = \sin x$.

Solution

The simplest way to obtain a fixed-point iteration formula is:

$$x = x + \cos x - \sin x$$

Then

$$g(x) = x + \cos x - \sin x$$

We apply the fixed-point method to this particular $g(x)$, and show the intermediate results as well:

[34]:
```
def g_ex8(x):
    return x+np.cos(x)-np.sin(x)

r = np.pi/4
x_0 = 0
k_max = 20
res = np.zeros(k_max, )
res[0] = x_0
print(' i  ', '     xi  ' '   g(xi)     ',
     ' ei=|xi-r|', 'ei/ei-1 ')
print("{0:<4d} {1:10.7f} {2:10.7f} {3:10.7f} {4:7s} ".format(
        0, x_0, g_ex8(x_0),
        np.abs(x_0-r),
        '      '))

for k in range(1, k_max):
    res[k] = fpi(g_ex8, x_0, k)
    print("{0:<4d} {1:10.7f} {2:10.7f} {3:10.7f} {4:7f}".format(
        k, res[k], g_ex8(res[k]),
        np.abs(res[k]-r),
        np.abs(res[k]-r)/np.abs(res[k-1]-r)))
```

```
 i        xi     g(xi)       ei=|xi-r| ei/ei-1
0     0.0000000  1.0000000  0.7853982
1     1.0000000  0.6988313  0.2146018 0.273240
2     0.6988313  0.8211025  0.0865668 0.403384
3     0.8211025  0.7706197  0.0357043 0.412448
4     0.7706197  0.7915189  0.0147785 0.413913
5     0.7915189  0.7828629  0.0061207 0.414162
6     0.7828629  0.7864483  0.0025352 0.414205
7     0.7864483  0.7849632  0.0010501 0.414212
8     0.7849632  0.7855783  0.0004350 0.414213
9     0.7855783  0.7853235  0.0001802 0.414214
10    0.7853235  0.7854291  0.0000746 0.414214
11    0.7854291  0.7853854  0.0000309 0.414214
12    0.7853854  0.7854035  0.0000128 0.414214
13    0.7854035  0.7853960  0.0000053 0.414214
14    0.7853960  0.7853991  0.0000022 0.414214
15    0.7853991  0.7853978  0.0000009 0.414214
16    0.7853978  0.7853983  0.0000004 0.414214
17    0.7853983  0.7853981  0.0000002 0.414214
18    0.7853981  0.7853982  0.0000001 0.414214
19    0.7853982  0.7853982  0.0000000 0.414214
```

So as expected, the result converges to $\frac{\pi}{4} \approx 0.7854$. The rate of the convergence S, equal to $|g'(r)| = 1 - \sin r - \cos r = 1 - \sin \frac{\pi}{4} - \cos \frac{\pi}{4}| \approx 0.4142$, is the same as the observed rate (the last column).

Example 1.9 *Find the fixed points of* $g(x) = 2.8x - x^2$.

Solution

Note the fixed points are 0 *and* 1.8. *We apply the fixed-point iteration, starint with* 0.1. *We may expect the result converge to* 0.

[35]:
```
def g_ex9(x):
    return 2.8*x-x**2

x_0 = 0.1
k_max = 100
sol = fpi(g_ex9, x_0, k_max)
print('The approximate solution with FPI is ',
     sol)
```

```
The approximate solution with FPI is  1.8000000000782124
```

So, surprisingly the result converges to the other solution. The reason is $g'(0) = 2.8 > 0$; however, $g'(1.8) = -0.8 < 1$.

1.2.1 Stopping Criteria

- Unlike the case of bisection, the number of steps required for FPI to converge within a given tolerance is rarely predictable beforehand
- A **stopping criterion** should be specify to terminate the algorithm
- An example is the absolute error stopping criterion:

$$|x_{i+1} - x_i| < \text{TOL}$$

- If the solution is not too near zero, the relative error stopping criterion can be used:

$$\frac{|x_{i+1} - x_i|}{|x_{i+1}|} < \text{TOL}$$

- Or a hybrid absolute/relative stopping criterion can be used:

$$\frac{|x_{i+1} - x_i|}{\max\{|x_{i+1}|, \theta\}} < \text{TOL}$$

  for some $\theta > 0$.

The Bisection Method is guaranteed to converge linearly. Fixed-Point Iteration is only locally convergent, and when it converges it is linearly convergent. Both methods require one function evaluation per step. The bisection cuts uncertainty by $1/2$ for each step, compared with approximately $S = |g(r)|$ for FPI. Therefore, Fixed-Point Iteration may be faster or slower than bisection, depending on whether S is smaller or larger than $1/2$.

Additional Exercises

Exercise 1.3 *Find all fixed points of the following $g(x)$*

(a) $\frac{3}{x}$

(b) $x^2 - 2x + 2$

(c) $x^2 - 4x + 2$

Exercise 1.4 *For which of the following $g(x)$ is $r = \sqrt{3}$ a fixed point?*

(a) $g(x) = \frac{x}{\sqrt{3}}$

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

(c) $g(x) = x^2 - x$

(d) $g(x) = 1 + \frac{2}{x+1}$

Exercise 1.5 *Determine whether Fixed-Point Iteration of $g(x)$ is locally covergent to the given fixed point r.*

(a) $g(x) = (2x-1)^{1/3}, r = 1$

(b) $g(x) = (x^3 + 1)/2, r = 1$

(c) $g(x) = \sin x + x, r = 0$

Exercise 1.6 *Which of the following three Fixed-Point Iterations converge to $\sqrt{2}$? Rank the ones that converge from fastest to slowest.*

(a) $x \to \frac{1}{2}x + \frac{1}{x}$

(b) $x \to \frac{2}{3}x + \frac{2}{3x}$

(c) $x \to \frac{3}{4}x + \frac{1}{2x}$