

# Bab 1

## Akar Persamaan

### 1.1 Metode bisection

Metode bisection sering juga disebut metode bagi-dua atau metode setengah-selang.

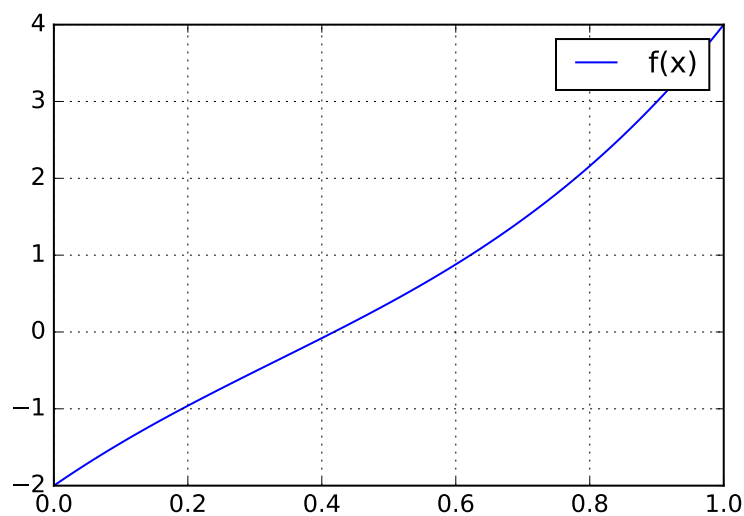
Ide dari metode bisection fakta bahwa jika tanda dari  $f(x_1)$  dan  $f(x_2)$  berbeda, maka setidaknya ada satu akar dari persamaan  $f(x) = 0$  ada dalam selang  $(x_1, x_2)$ .

Misalkan kita ingin mencari akar dari fungsi berikut:

$$f(x) = 5x^3 - 5x^2 + 6x - 2$$

Untuk memperoleh gambaran mengenai akar dari persamaan  $f(x) = 0$ , kita akan membuat plot dari  $f(x)$  terlebih dahulu.

```
def func_01(x):  
    return 5*x**3 - 5*x**2 + 6*x - 2  
  
x = np.linspace(0,1,100)  
y = func_01(x)  
plt.clf()  
plt.plot(x, y, label="f(x)")  
plt.legend()  
plt.grid()  
plt.savefig("figures/bisection_01_1.pdf")
```



Dari plot di atas dapat dilihat bahwa akar dari  $f(x)$  terletak di sekitar  $x = 0.5$ . Sebagai ilustrasi untuk metode bisection kita akan menggunakan interval  $x_1 = 0$  dan  $x_2 = 1$ .

Mari kita cek bahwa tanda  $f(x_1)$  dan  $f(x_2)$  memiliki tanda yang berbeda, atau  $f(x_1)f(x_2) < 0$ .

```
x1 = 0.0
x2 = 1.0
f1 = func_01(x1)
f2 = func_01(x2)
print("f1 = ", f1)
print("f2 = ", f2)
print("f1*f2 is negative = ", f1*f2 < 0)
```

```
f1 = -2.0
f2 = 4.0
f1*f2 is negative = True
```

Fungsi `np.sign()` juga bisa digunakan untuk mengecek tanda positif atau negatif dari suatu bilangan.

```
np.sign(-2.0), np.sign(2.1)
```

```
| (-1.0, 1.0)
```

Sekarang, kita perlu menentukan tebakan akar dari selang  $x_1$  dan  $x_2$ . Dengan metode bisection, tebakan akar dihitung tepat berada di tengah selang yaitu:

$$x_r = \frac{x_1 + x_2}{2}$$

```
xr = 0.5*(x1 + x2)
xr
```

```
| 0.5
```

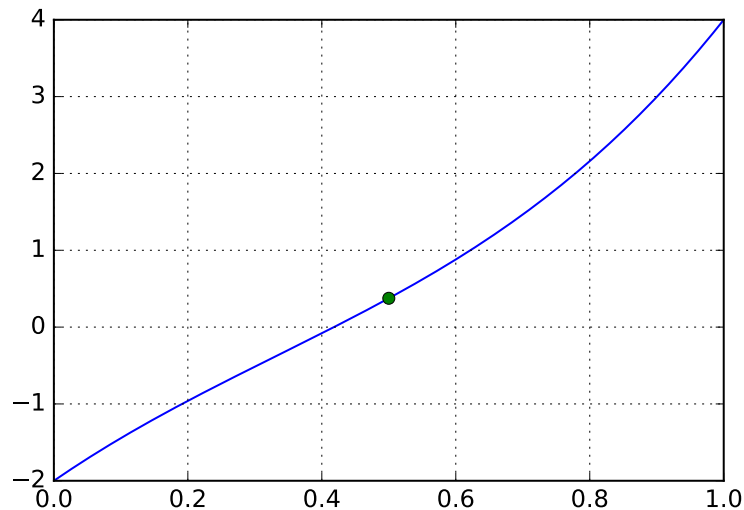
OK, sepertinya kita sudah berada dekat di akar sebenarnya. Mari kita cek nilai  $f(x_r)$ :

```
fxr = func_01(xr)
fxr
```

```
| 0.375
```

Ternyata nilai dari  $f(x_r)$  tidak tepat pada 0. Mari buat plot  $f(x)$  pada domain (interval) yang lebih sempit.

```
x = np.linspace(0.0, 1.0, 500)
y = func_01(x)
plt.clf()
plt.plot(x, y)
plt.plot(xr, fxr, marker="o") # Tandai nilai fx pada xr, yaitu (xr,fxr)
plt.grid()
plt.savefig("figures/bisection_01_2.pdf")
```



Kita dapat memperbaiki tebakan akar dengan memilih rentang baru di mana kita akan mengaplikasikan lagi metode bisection. Kita sekarang memiliki 3 titik yaitu  $x_1$ ,  $x_2$ , dan  $x_r$ , dengan nilai fungsi pada titik-titik tersebut adalah:

```
f1, f2, fxr
```

```
| (-2.0, 4.0, 0.375)
```

Dengan informasi tersebut, kita dapat menggunakan  $x_r$  sebagai pengganti dari  $x_2$  karena selang ini lebih kecil dan diharapkan nilai tebakan akar dapat menjadi lebih dekat dengan akar sebenarnya.

```
x2 = xr
f2 = fxr
```

Cek apakah nilai fungsi pada interval baru ini berbeda tanda (hasil kali  $f(x_1)$  dan  $f(x_2)$  adalah negatif.

```
print(f1*f2 < 0)
```

```
| True
```

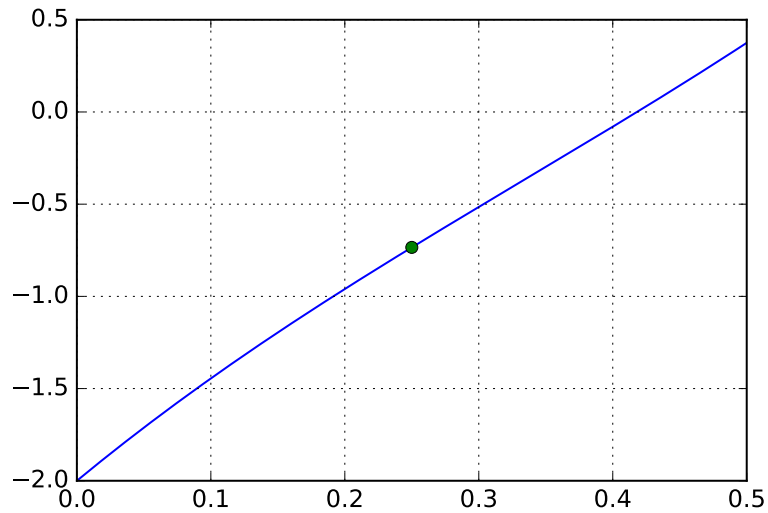
Kita hitung lagi tebakan akar  $x_r$  sebagai nilai tengah antara  $x_1$  dan  $x_2$ :

```
xr = 0.5*(x1 + x2)
fxr = func_01(xr)
fxr
```

```
| -0.734375
```

Sepertinya nilai akar yang kita dapatkan masih belum baik.

```
x = np.linspace(x1, x2, 500)
y = func_01(x)
plt.clf()
plt.plot(x, y)
plt.plot(xr, fxr, marker="o") # Tandai nilai fx pada xr, yaitu (xr,fxr)
plt.grid()
plt.savefig("figures/bisection_01_3.pdf")
```



```
f1, f2, fxr
```

```
| (-2.0, 0.375, -0.734375)
```

Kita akan melakukan kembali prosedur bisection. Untuk interval baru kita akan ganti  $x_1$  dengan  $x_r$ .

```
x1 = xr
f1 = fxr
print(f1*f2 < 0)
```

```
| True
```

Hitung kembali tebakan akar pada selang  $x_1$  dan  $x_2$

```
xr = 0.5*(x1 + x2)
fxr = func_01(xr)
fxr
```

```
| -0.189453125
```

Nilai ini sudah lebih dekat dari tebakan-tebakan kita sebelumnya. Untuk mendapatkan tebakan akar yang lebih baik kita akan lakukan sekali lagi metode bisection.

```
f1, f2, fxr
```

```
| (-0.734375, 0.375, -0.189453125)
```

```
x1 = xr
f1 = fxr
print(f1*f2 < 0)
```

```
| True
```

```
xr = 0.5*(x1 + x2)
fxr = func_01(xr)
fxr
```

```
| 0.086669921875
```

Tebakan ini lebih baik dari tebakan sebelumnya karena  $f(x_r)$  yang diperoleh lebih dekat dengan 0. Kita dapat melakukan prosedur bisection sekali lagi.

```
f1, f2, fxr
```

```
| (-0.189453125, 0.375, 0.086669921875)
```

```
x2 = xr
f2 = fxr
xr = 0.5*(x1 + x2)
fxr = func_01(xr)
fxr
```

```
| -0.052459716796875
```

Setelah melakukan iterasi metode bisection secara manual, sekarang kita akan membuat prosedur bisection dalam suatu subrutin (fungsi). Fungsi ini menerima masukan **f** sebagai fungsi yang akan dicari akarnya, **x1** dan **x2** sebagai input selang di mana akar akan dicari. Fungsi ini juga menggunakan **TOL** dengan nilai default  $1e-10$  untuk menentukan akurasi hasil yang diperoleh dan juga **NiterMax** dengan nilai default 100 sebagai jumlah maksimum iterasi yang dilakukan.

```
def bisection(f, x1, x2, TOL=1e-10, NiterMax=100):

    f1 = f(x1)
    f2 = f(x2)

    if f1*f2 > 0:
        raise RuntimeError("f1 dan f2 memiliki tanda yang sama")

    for i in range(1, NiterMax+1):

        xr = 0.5*(x1 + x2)
        fxr = f(xr)

        if abs(fxr) <= TOL:
            print("Iterasi konvergen: akar ditemukan")
            return xr

        print("Iter = %5d, xr = %18.10f, abs(fxr) = %15.5e" % (i, xr, abs(fxr)))

        # f1 dan fxr berbeda tanda
        if f1*fxr < 0.0:
            x2 = xr
            f2 = fxr
        else:
            x1 = xr
            f1 = fxr

    print("WARNING: Konvergensi tidak diperoleh setelah %d iterasi" % NiterMax)
    print("WARNING: Nilai tebakan akhir akan dikembalikan")
    return xr
```

Contoh penggunaan:

Cari akar pada selang  $[0.0, 1.0]$

```
xr = bisection(func_01, 0.0, 1.0)
```

```
Iter = 1, xr = 0.5000000000, abs(fxr) = 3.75000e-01
Iter = 2, xr = 0.2500000000, abs(fxr) = 7.34375e-01
Iter = 3, xr = 0.3750000000, abs(fxr) = 1.89453e-01
Iter = 4, xr = 0.4375000000, abs(fxr) = 8.66699e-02
Iter = 5, xr = 0.4062500000, abs(fxr) = 5.24597e-02
Iter = 6, xr = 0.4218750000, abs(fxr) = 1.67809e-02
Iter = 7, xr = 0.4140625000, abs(fxr) = 1.79133e-02
Iter = 8, xr = 0.4179687500, abs(fxr) = 5.85616e-04
Iter = 9, xr = 0.4199218750, abs(fxr) = 8.09266e-03
Iter = 10, xr = 0.4189453125, abs(fxr) = 3.75230e-03
Iter = 11, xr = 0.4184570312, abs(fxr) = 1.58304e-03
Iter = 12, xr = 0.4182128906, abs(fxr) = 4.98635e-04
Iter = 13, xr = 0.4180908203, abs(fxr) = 4.35092e-05
Iter = 14, xr = 0.4181518555, abs(fxr) = 2.27558e-04
Iter = 15, xr = 0.4181213379, abs(fxr) = 9.20233e-05
Iter = 16, xr = 0.4181060791, abs(fxr) = 2.42567e-05
Iter = 17, xr = 0.4180984497, abs(fxr) = 9.62632e-06
Iter = 18, xr = 0.4181022644, abs(fxr) = 7.31519e-06
Iter = 19, xr = 0.4181003571, abs(fxr) = 1.15557e-06
Iter = 20, xr = 0.4181013107, abs(fxr) = 3.07981e-06
Iter = 21, xr = 0.4181008339, abs(fxr) = 9.62119e-07
Iter = 22, xr = 0.4181005955, abs(fxr) = 9.67255e-08
Iter = 23, xr = 0.4181007147, abs(fxr) = 4.32697e-07
Iter = 24, xr = 0.4181006551, abs(fxr) = 1.67986e-07
Iter = 25, xr = 0.4181006253, abs(fxr) = 3.56301e-08
Iter = 26, xr = 0.4181006104, abs(fxr) = 3.05477e-08
Iter = 27, xr = 0.4181006178, abs(fxr) = 2.54121e-09
Iter = 28, xr = 0.4181006141, abs(fxr) = 1.40032e-08
Iter = 29, xr = 0.4181006160, abs(fxr) = 5.73101e-09
Iter = 30, xr = 0.4181006169, abs(fxr) = 1.59490e-09
Iter = 31, xr = 0.4181006174, abs(fxr) = 4.73154e-10
Iter = 32, xr = 0.4181006171, abs(fxr) = 5.60874e-10
Iterasi konvergen: akar ditemukan
```

Cari akar pada selang  $[0.0, 1.0]$ , dengan toleransi  $10^{-9}$  dan jumlah iterasi maksimum 10.

```
xr = bisection(func_01, 0.0, 0.5, TOL=1e-9, NiterMax=10)
```

```
Iter = 1, xr = 0.2500000000, abs(fxr) = 7.34375e-01
Iter = 2, xr = 0.3750000000, abs(fxr) = 1.89453e-01
Iter = 3, xr = 0.4375000000, abs(fxr) = 8.66699e-02
Iter = 4, xr = 0.4062500000, abs(fxr) = 5.24597e-02
Iter = 5, xr = 0.4218750000, abs(fxr) = 1.67809e-02
Iter = 6, xr = 0.4140625000, abs(fxr) = 1.79133e-02
Iter = 7, xr = 0.4179687500, abs(fxr) = 5.85616e-04
Iter = 8, xr = 0.4199218750, abs(fxr) = 8.09266e-03
Iter = 9, xr = 0.4189453125, abs(fxr) = 3.75230e-03
Iter = 10, xr = 0.4184570312, abs(fxr) = 1.58304e-03
WARNING: Konvergensi tidak diperoleh setelah 10 iterasi
WARNING: Nilai tebakan akhir akan dikembalikan
```

Sama seperti sebelumnya namun menggunakan nilai iterasi maksimum default.

```
xr = bisection(func_01, 0.0, 0.5, TOL=1e-9)
```

```
Iter = 1, xr = 0.2500000000, abs(fxr) = 7.34375e-01
Iter = 2, xr = 0.3750000000, abs(fxr) = 1.89453e-01
Iter = 3, xr = 0.4375000000, abs(fxr) = 8.66699e-02
Iter = 4, xr = 0.4062500000, abs(fxr) = 5.24597e-02
Iter = 5, xr = 0.4218750000, abs(fxr) = 1.67809e-02
```

```

Iter = 6, xr = 0.4140625000, abs(fxr) = 1.79133e-02
Iter = 7, xr = 0.4179687500, abs(fxr) = 5.85616e-04
Iter = 8, xr = 0.4199218750, abs(fxr) = 8.09266e-03
Iter = 9, xr = 0.4189453125, abs(fxr) = 3.75230e-03
Iter = 10, xr = 0.4184570312, abs(fxr) = 1.58304e-03
Iter = 11, xr = 0.4182128906, abs(fxr) = 4.98635e-04
Iter = 12, xr = 0.4180908203, abs(fxr) = 4.35092e-05
Iter = 13, xr = 0.4181518555, abs(fxr) = 2.27558e-04
Iter = 14, xr = 0.4181213379, abs(fxr) = 9.20233e-05
Iter = 15, xr = 0.4181060791, abs(fxr) = 2.42567e-05
Iter = 16, xr = 0.4180984497, abs(fxr) = 9.62632e-06
Iter = 17, xr = 0.4181022644, abs(fxr) = 7.31519e-06
Iter = 18, xr = 0.4181003571, abs(fxr) = 1.15557e-06
Iter = 19, xr = 0.4181013107, abs(fxr) = 3.07981e-06
Iter = 20, xr = 0.4181008339, abs(fxr) = 9.62119e-07
Iter = 21, xr = 0.4181005955, abs(fxr) = 9.67255e-08
Iter = 22, xr = 0.4181007147, abs(fxr) = 4.32697e-07
Iter = 23, xr = 0.4181006551, abs(fxr) = 1.67986e-07
Iter = 24, xr = 0.4181006253, abs(fxr) = 3.56301e-08
Iter = 25, xr = 0.4181006104, abs(fxr) = 3.05477e-08
Iter = 26, xr = 0.4181006178, abs(fxr) = 2.54121e-09
Iter = 27, xr = 0.4181006141, abs(fxr) = 1.40032e-08
Iter = 28, xr = 0.4181006160, abs(fxr) = 5.73101e-09
Iter = 29, xr = 0.4181006169, abs(fxr) = 1.59490e-09
Iterasi konvergen: akar ditemukan

```

Cari akar pada selang  $[0.3, 0.5]$ ,

```
xr = bisection(func_01, 0.3, 0.5, NiterMax=100)
```

```

Iter = 1, xr = 0.4000000000, abs(fxr) = 8.00000e-02
Iter = 2, xr = 0.4500000000, abs(fxr) = 1.43125e-01
Iter = 3, xr = 0.4250000000, abs(fxr) = 3.07031e-02
Iter = 4, xr = 0.4125000000, abs(fxr) = 2.48340e-02
Iter = 5, xr = 0.4187500000, abs(fxr) = 2.88452e-03
Iter = 6, xr = 0.4156250000, abs(fxr) = 1.09868e-02
Iter = 7, xr = 0.4171875000, abs(fxr) = 4.05420e-03
Iter = 8, xr = 0.4179687500, abs(fxr) = 5.85616e-04
Iter = 9, xr = 0.4183593750, abs(fxr) = 1.14926e-03
Iter = 10, xr = 0.4181640625, abs(fxr) = 2.81773e-04
Iter = 11, xr = 0.4180664063, abs(fxr) = 1.51934e-04
Iter = 12, xr = 0.4181152344, abs(fxr) = 6.49166e-05
Iter = 13, xr = 0.4180908203, abs(fxr) = 4.35092e-05
Iter = 14, xr = 0.4181030273, abs(fxr) = 1.07035e-05
Iter = 15, xr = 0.4180969238, abs(fxr) = 1.64029e-05
Iter = 16, xr = 0.418099756, abs(fxr) = 2.84972e-06
Iter = 17, xr = 0.4181015015, abs(fxr) = 3.92688e-06
Iter = 18, xr = 0.4181007385, abs(fxr) = 5.38581e-07
Iter = 19, xr = 0.4181003571, abs(fxr) = 1.15557e-06
Iter = 20, xr = 0.4181005478, abs(fxr) = 3.08494e-07
Iter = 21, xr = 0.4181006432, abs(fxr) = 1.15043e-07
Iter = 22, xr = 0.4181005955, abs(fxr) = 9.67255e-08
Iter = 23, xr = 0.4181006193, abs(fxr) = 9.15899e-09
Iter = 24, xr = 0.4181006074, abs(fxr) = 4.37832e-08
Iter = 25, xr = 0.4181006134, abs(fxr) = 1.73121e-08
Iter = 26, xr = 0.4181006163, abs(fxr) = 4.07657e-09
Iter = 27, xr = 0.4181006178, abs(fxr) = 2.54121e-09
Iter = 28, xr = 0.4181006171, abs(fxr) = 7.67679e-10
Iter = 29, xr = 0.4181006175, abs(fxr) = 8.86765e-10
Iterasi konvergen: akar ditemukan

```

Contoh yang akan menghasilkan error:

```
xr = bisection(func_01, 0.3, 0.4)
```

```
-----RuntimeError
Traceback (most recent call last)<ipython-input-1-2f68945ac062> in
<module>
----> 1 xr = bisection(func_01, 0.3, 0.4)
<ipython-input-1-3b4f53df89d5> in bisection(f, x1, x2, TOL, NiterMax)
      5
      6     if f1*f2 > 0:
----> 7         raise RuntimeError("f1 dan f2 memiliki tanda yang
sama")
      8
      9     for i in range(1,NiterMax+1):
RuntimeError: f1 dan f2 memiliki tanda yang sama
```

### 1.1.1 Contoh 2

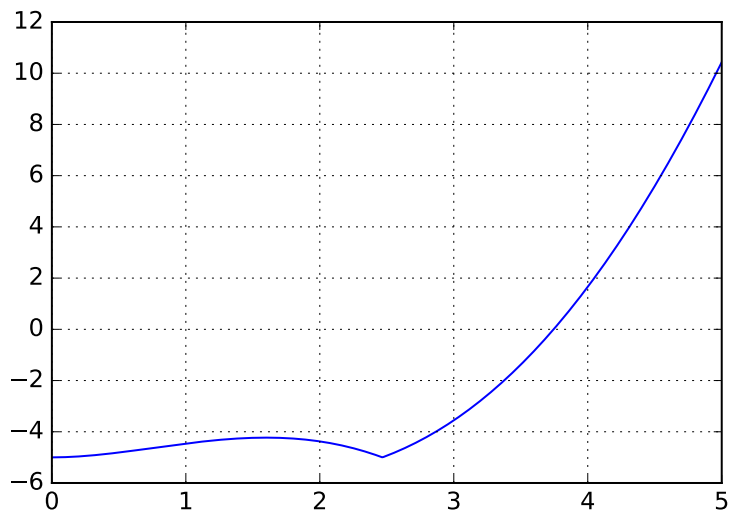
Contoh dengan fungsi yang berbeda:

$$f = x^2 \left| \cos(\sqrt{x}) \right| - 5$$

```
def func_02(x):
    return x**2 * np.abs(np.cos(np.sqrt(x))) - 5
```

Kita perlu selang untuk tebakan awal akar. Untuk memperoleh informasi tersebut kita akan buat plot dari  $f(x)$  terlebih dahulu.

```
x = np.linspace(0,5,500)
y = func_02(x)
plt.clf()
plt.plot(x, y)
plt.grid()
plt.savefig("figures/func_02_1.pdf")
```



Akar terletak antara  $x = 0$  dan  $x = 5$ .

```
xr = bisection(func_02, 0, 5)
```

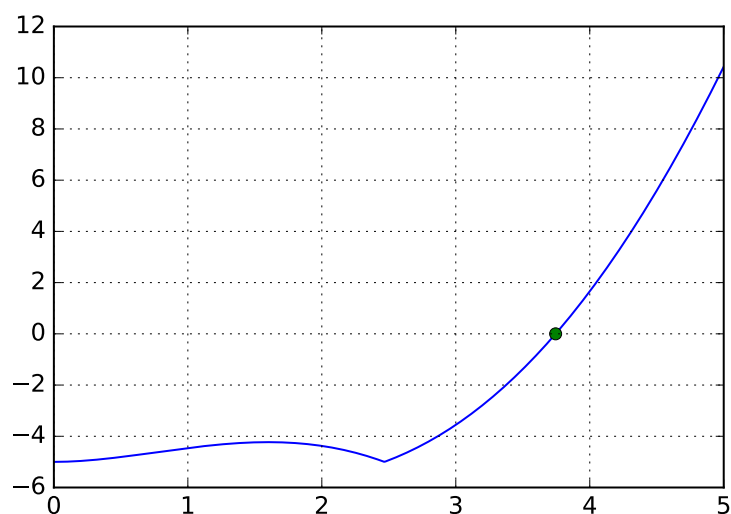
Iter =	1, xr =	2.5000000000, abs(fxr) =	4.93536e+00
Iter =	2, xr =	3.7500000000, abs(fxr) =	2.87324e-02
Iter =	3, xr =	3.1250000000, abs(fxr) =	3.08887e+00
Iter =	4, xr =	3.4375000000, abs(fxr) =	1.69754e+00



Iter = 5, xr =	3.5937500000, abs(fxr) =	8.77063e-01
Iter = 6, xr =	3.6718750000, abs(fxr) =	4.34912e-01
Iter = 7, xr =	3.7109375000, abs(fxr) =	2.05785e-01
Iter = 8, xr =	3.7304687500, abs(fxr) =	8.92016e-02
Iter = 9, xr =	3.7402343750, abs(fxr) =	3.04035e-02
Iter = 10, xr =	3.7451171875, abs(fxr) =	8.77797e-04
Iter = 11, xr =	3.7475585938, abs(fxr) =	1.39167e-02
Iter = 12, xr =	3.7463378906, abs(fxr) =	6.51683e-03
Iter = 13, xr =	3.7457275391, abs(fxr) =	2.81886e-03
Iter = 14, xr =	3.7454223633, abs(fxr) =	9.70364e-04
Iter = 15, xr =	3.7452697754, abs(fxr) =	4.62426e-05
Iter = 16, xr =	3.7451934814, abs(fxr) =	4.15787e-04
Iter = 17, xr =	3.7452316284, abs(fxr) =	1.84775e-04
Iter = 18, xr =	3.7452507019, abs(fxr) =	6.92668e-05
Iter = 19, xr =	3.7452602386, abs(fxr) =	1.15123e-05
Iter = 20, xr =	3.7452650070, abs(fxr) =	1.73651e-05
Iter = 21, xr =	3.745262228, abs(fxr) =	2.92641e-06
Iter = 22, xr =	3.7452614307, abs(fxr) =	4.29294e-06
Iter = 23, xr =	3.7452620268, abs(fxr) =	6.83264e-07
Iter = 24, xr =	3.7452623248, abs(fxr) =	1.12157e-06
Iter = 25, xr =	3.7452621758, abs(fxr) =	2.19155e-07
Iter = 26, xr =	3.7452621013, abs(fxr) =	2.32055e-07
Iter = 27, xr =	3.7452621385, abs(fxr) =	6.44991e-09
Iter = 28, xr =	3.7452621572, abs(fxr) =	1.06352e-07
Iter = 29, xr =	3.7452621479, abs(fxr) =	4.99512e-08
Iter = 30, xr =	3.7452621432, abs(fxr) =	2.17507e-08
Iter = 31, xr =	3.7452621409, abs(fxr) =	7.65038e-09
Iter = 32, xr =	3.7452621397, abs(fxr) =	6.00232e-10
Iter = 33, xr =	3.7452621391, abs(fxr) =	2.92484e-09
Iter = 34, xr =	3.7452621394, abs(fxr) =	1.16230e-09
Iter = 35, xr =	3.7452621396, abs(fxr) =	2.81036e-10
Iter = 36, xr =	3.7452621396, abs(fxr) =	1.59599e-10

Iterasi konvergen: akar ditemukan

```
x = np.linspace(0,5,500)
y = func_02(x)
plt.clf()
plt.plot(x, y)
plt.plot(xr, func_02(xr), marker="o")
plt.grid()
plt.savefig("figures/func_02_2.pdf")
```



Mari kita coba gunakan selang  $x = 3$  dan  $x = 4$ .

```
xr = bisection(func_02, 3, 4)
```

```
Iter = 1, xr = 3.5000000000, abs(fxr) = 1.37950e+00
Iter = 2, xr = 3.7500000000, abs(fxr) = 2.87324e-02
Iter = 3, xr = 3.6250000000, abs(fxr) = 7.02772e-01
Iter = 4, xr = 3.6875000000, abs(fxr) = 3.43907e-01
Iter = 5, xr = 3.7187500000, abs(fxr) = 1.59314e-01
Iter = 6, xr = 3.7343750000, abs(fxr) = 6.57229e-02
Iter = 7, xr = 3.7421875000, abs(fxr) = 1.86034e-02
Iter = 8, xr = 3.7460937500, abs(fxr) = 5.03748e-03
Iter = 9, xr = 3.7441406250, abs(fxr) = 6.78970e-03
Iter = 10, xr = 3.7451171875, abs(fxr) = 8.77797e-04
Iter = 11, xr = 3.7456054688, abs(fxr) = 2.07942e-03
Iter = 12, xr = 3.7453613281, abs(fxr) = 6.00706e-04
Iter = 13, xr = 3.7452392578, abs(fxr) = 1.38572e-04
Iter = 14, xr = 3.7453002930, abs(fxr) = 2.31060e-04
Iter = 15, xr = 3.7452697754, abs(fxr) = 4.62426e-05
Iter = 16, xr = 3.7452545166, abs(fxr) = 4.61651e-05
Iter = 17, xr = 3.7452621460, abs(fxr) = 3.86710e-08
Iter = 18, xr = 3.7452583313, abs(fxr) = 2.30632e-05
Iter = 19, xr = 3.7452602386, abs(fxr) = 1.15123e-05
Iter = 20, xr = 3.7452611923, abs(fxr) = 5.73681e-06
Iter = 21, xr = 3.7452616692, abs(fxr) = 2.84907e-06
Iter = 22, xr = 3.7452619076, abs(fxr) = 1.40520e-06
Iter = 23, xr = 3.7452620268, abs(fxr) = 6.83264e-07
Iter = 24, xr = 3.7452620864, abs(fxr) = 3.22296e-07
Iter = 25, xr = 3.7452621162, abs(fxr) = 1.41813e-07
Iter = 26, xr = 3.7452621311, abs(fxr) = 5.15708e-08
Iter = 27, xr = 3.7452621385, abs(fxr) = 6.44991e-09
Iter = 28, xr = 3.7452621423, abs(fxr) = 1.61106e-08
Iter = 29, xr = 3.7452621404, abs(fxr) = 4.83032e-09
Iter = 30, xr = 3.7452621395, abs(fxr) = 8.09795e-10
Iter = 31, xr = 3.7452621399, abs(fxr) = 2.01026e-09
Iter = 32, xr = 3.7452621397, abs(fxr) = 6.00232e-10
Iter = 33, xr = 3.7452621396, abs(fxr) = 1.04781e-10
Iter = 34, xr = 3.7452621397, abs(fxr) = 2.47725e-10
Iterasi konvergen: akar ditemukan
```

## 1.2 Metode Regula-Falsi

Metode regula-falsi mirip dengan metode bisection, namun dengan persamaan yang berbeda untuk menentukan aproksimasi akar. Pada metode bisection, tebakan akar diberikan sebagai nilai tengah dari  $x_1$  dan  $x_2$  sedangkan pada metode regula-falsi digunakan interpolasi linear antara  $f(x_1)$  dan  $f(x_2)$ . Tebakan akar adalah perpotongan antara garis interpolasi linear ini dengan sumbu  $x$ . Hasil akhirnya adalah:

$$x_r = x_2 - \frac{f(x_2)}{f(x_1) - f(x_2)}(x_1 - x_2)$$

```
def regula_falsi(f, x1, x2, TOL=1e-10, NiterMax=100):

    f1 = f(x1)
    f2 = f(x2)

    if f1*f2 > 0:
        raise RuntimeError("f1 dan f2 memiliki tanda yang sama")

    for i in range(1, NiterMax+1):

        xr = x2 - f2*(x1 - x2)/(f1 - f2)
        fxr = f(xr)
```

```

if abs(fxr) <= TOL:
    print("Iterasi konvergen: akar ditemukan")
    return xr

print("Iter = %5d, xr = %18.10f, abs(fxr) = %15.5e" % (i, xr, abs(fxr)))

# f1 dan fxr berbeda tanda
if f1*fxr < 0.0:
    x2 = xr
    f2 = fxr
else:
    x1 = xr
    f1 = fxr

print("WARNING: Konvergensi tidak diperoleh setelah %d iterasi" % NiterMax)
print("WARNING: Nilai tebakan akhir akan dikembalikan")
return xr

```

Contoh penggunaan:

```
xr = regula_falsi(func_01, 0.0, 1.0)
```

```

Iter =    1, xr =    0.3333333333, abs(fxr) =    3.70370e-01
Iter =    2, xr =    0.3898305085, abs(fxr) =    1.24648e-01
Iter =    3, xr =    0.4082699418, abs(fxr) =    4.35410e-02
Iter =    4, xr =    0.4146417183, abs(fxr) =    1.53464e-02
Iter =    5, xr =    0.4168789156, abs(fxr) =    5.42383e-03
Iter =    6, xr =    0.4176685323, abs(fxr) =    1.91870e-03
Iter =    7, xr =    0.4179477285, abs(fxr) =    6.78967e-04
Iter =    8, xr =    0.4180465103, abs(fxr) =    2.40291e-04
Iter =    9, xr =    0.4180814678, abs(fxr) =    8.50444e-05
Iter =   10, xr =    0.4180938398, abs(fxr) =    3.00995e-05
Iter =   11, xr =    0.4180982185, abs(fxr) =    1.06531e-05
Iter =   12, xr =    0.4180997683, abs(fxr) =    3.77043e-06
Iter =   13, xr =    0.4181003168, abs(fxr) =    1.33446e-06
Iter =   14, xr =    0.4181005109, abs(fxr) =    4.72306e-07
Iter =   15, xr =    0.4181005796, abs(fxr) =    1.67163e-07
Iter =   16, xr =    0.4181006039, abs(fxr) =    5.91639e-08
Iter =   17, xr =    0.4181006125, abs(fxr) =    2.09398e-08
Iter =   18, xr =    0.4181006156, abs(fxr) =    7.41122e-09
Iter =   19, xr =    0.4181006167, abs(fxr) =    2.62305e-09
Iter =   20, xr =    0.4181006170, abs(fxr) =    9.28374e-10
Iter =   21, xr =    0.4181006172, abs(fxr) =    3.28579e-10
Iter =   22, xr =    0.4181006172, abs(fxr) =    1.16294e-10
Iterasi konvergen: akar ditemukan

```

Contoh lain:

```
xr = regula_falsi(func_02, 3, 4)
```

```

Iter =    1, xr =    3.6819027315, abs(fxr) =    3.76606e-01
Iter =    2, xr =    3.7407725061, abs(fxr) =    2.71536e-02
Iter =    3, xr =    3.7449486892, abs(fxr) =    1.89809e-03
Iter =    4, xr =    3.7452402785, abs(fxr) =    1.32391e-04
Iter =    5, xr =    3.7452606151, abs(fxr) =    9.23275e-06
Iter =    6, xr =    3.7452620333, abs(fxr) =    6.43873e-07
Iter =    7, xr =    3.7452621322, abs(fxr) =    4.49024e-08
Iter =    8, xr =    3.7452621391, abs(fxr) =    3.13140e-09
Iter =    9, xr =    3.7452621396, abs(fxr) =    2.18379e-10
Iterasi konvergen: akar ditemukan

```

### 1.3 Metode fixed-point

Pada metode ini, persamaan  $f(x)$  yang ingin kita cari akarnya diubah menjadi  $x = g(x)$ . Contoh: untuk mencari akar dari persamaan  $f(x) = e^{-x} - x = 0$  kita mengubah persamaan tersebut menjadi  $x = g(x) = e^{-x}$ .

Iterasi dimulai dengan suatu tebakan awal  $x_0$ . Nilai tebakan akar berikutnya dihitung dengan persamaan

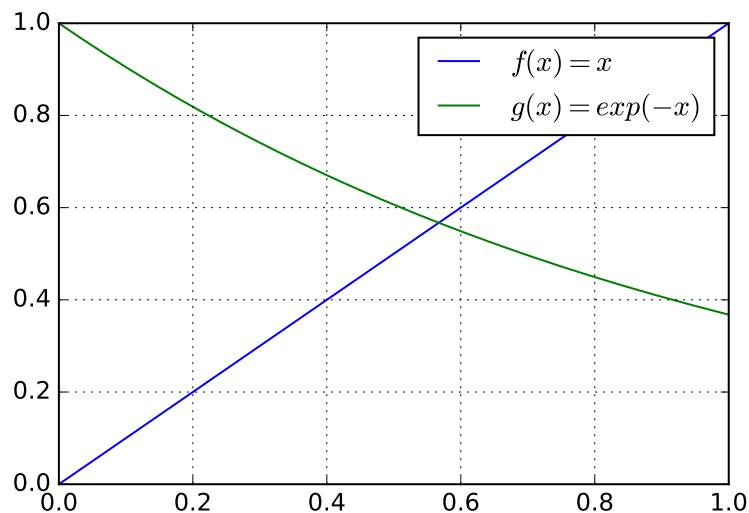
$$x_{i+1} = g(x_i)$$

Jika iterasi ini konvergen, maka  $x_{i+1}$  adalah akar dari persamaan  $f(x) = 0$ .

Perhatikan bahwa metode ini tidak selalu konvergen.

```
# definisi fungsi g(x) = exp(-x)
def func_03(x):
    return np.exp(-x)
```

```
plt.clf()
x = np.linspace(0,1,500)
plt.plot(x, x, label="$f(x) = x$")
plt.plot(x, func_03(x), label="$g(x) = exp(-x)$")
plt.legend()
plt.grid()
plt.savefig("figures/func_03_1.pdf")
```



```
def fixed_point(g, x, TOL=1e-10, NiterMax=100):

    for i in range(1,NiterMax+1):
        gx = g(x)
        print("Iter = %5d, x = %18.10f, g(x) = %18.10f abs(x-g(x)) %15.5e" % (i, x, g(x), abs(x-gx)))
        if abs(x - gx) <= TOL:
            print("Iterasi konvergen: akar ditemukan")
            return x
        x = gx

    print("WARNING: Konvergensi tidak diperoleh setelah %d iterasi" % NiterMax)
    print("WARNING: Nilai tebakan akhir akan dikembalikan")
    return x
```

```
x0 = 0.0
xr = fixed_point(func_03, x0)
```

```

Iter = 1, x = 0.0000000000, g(x) = 1.0000000000
abs(x-g(x)) 1.00000e+00
Iter = 2, x = 1.0000000000, g(x) = 0.3678794412
abs(x-g(x)) 6.32121e-01
Iter = 3, x = 0.3678794412, g(x) = 0.6922006276
abs(x-g(x)) 3.24321e-01
Iter = 4, x = 0.6922006276, g(x) = 0.5004735006
abs(x-g(x)) 1.91727e-01
Iter = 5, x = 0.5004735006, g(x) = 0.6062435351
abs(x-g(x)) 1.05770e-01
Iter = 6, x = 0.6062435351, g(x) = 0.5453957860
abs(x-g(x)) 6.08477e-02
Iter = 7, x = 0.5453957860, g(x) = 0.5796123355
abs(x-g(x)) 3.42165e-02
Iter = 8, x = 0.5796123355, g(x) = 0.5601154614
abs(x-g(x)) 1.94969e-02
Iter = 9, x = 0.5601154614, g(x) = 0.5711431151
abs(x-g(x)) 1.10277e-02
Iter = 10, x = 0.5711431151, g(x) = 0.5648793474
abs(x-g(x)) 6.26377e-03
Iter = 11, x = 0.5648793474, g(x) = 0.5684287250
abs(x-g(x)) 3.54938e-03
Iter = 12, x = 0.5684287250, g(x) = 0.5664147331
abs(x-g(x)) 2.01399e-03
Iter = 13, x = 0.5664147331, g(x) = 0.5675566373
abs(x-g(x)) 1.14190e-03
Iter = 14, x = 0.5675566373, g(x) = 0.5669089119
abs(x-g(x)) 6.47725e-04
Iter = 15, x = 0.5669089119, g(x) = 0.5672762322
abs(x-g(x)) 3.67320e-04
Iter = 16, x = 0.5672762322, g(x) = 0.5670678984
abs(x-g(x)) 2.08334e-04
Iter = 17, x = 0.5670678984, g(x) = 0.5671860501
abs(x-g(x)) 1.18152e-04
Iter = 18, x = 0.5671860501, g(x) = 0.5671190401
abs(x-g(x)) 6.70100e-05
Iter = 19, x = 0.5671190401, g(x) = 0.5671570440
abs(x-g(x)) 3.80039e-05
Iter = 20, x = 0.5671570440, g(x) = 0.5671354902
abs(x-g(x)) 2.15538e-05
Iter = 21, x = 0.5671354902, g(x) = 0.5671477143
abs(x-g(x)) 1.22241e-05
Iter = 22, x = 0.5671477143, g(x) = 0.5671407815
abs(x-g(x)) 6.93280e-06
Iter = 23, x = 0.5671407815, g(x) = 0.5671447133
abs(x-g(x)) 3.93189e-06
Iter = 24, x = 0.5671447133, g(x) = 0.5671424834
abs(x-g(x)) 2.22995e-06
Iter = 25, x = 0.5671424834, g(x) = 0.5671437481
abs(x-g(x)) 1.26470e-06
Iter = 26, x = 0.5671437481, g(x) = 0.5671430308
abs(x-g(x)) 7.17265e-07
Iter = 27, x = 0.5671430308, g(x) = 0.5671434376
abs(x-g(x)) 4.06792e-07
Iter = 28, x = 0.5671434376, g(x) = 0.5671432069
abs(x-g(x)) 2.30709e-07
Iter = 29, x = 0.5671432069, g(x) = 0.5671433378
abs(x-g(x)) 1.30845e-07
Iter = 30, x = 0.5671433378, g(x) = 0.5671432636
abs(x-g(x)) 7.42080e-08
Iter = 31, x = 0.5671432636, g(x) = 0.5671433056
abs(x-g(x)) 4.20866e-08
Iter = 32, x = 0.5671433056, g(x) = 0.5671432818

```

```

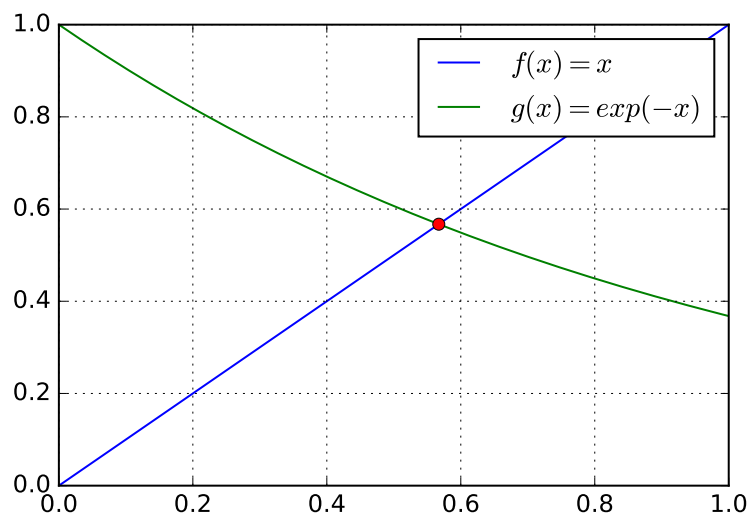
abs(x-g(x))    2.38691e-08
Iter =   33, x =    0.5671432818, g(x) =    0.5671432953
abs(x-g(x))    1.35372e-08
Iter =   34, x =    0.5671432953, g(x) =    0.5671432876
abs(x-g(x))    7.67754e-09
Iter =   35, x =    0.5671432876, g(x) =    0.5671432920
abs(x-g(x))    4.35427e-09
Iter =   36, x =    0.5671432920, g(x) =    0.5671432895
abs(x-g(x))    2.46949e-09
Iter =   37, x =    0.5671432895, g(x) =    0.5671432909
abs(x-g(x))    1.40056e-09
Iter =   38, x =    0.5671432909, g(x) =    0.5671432901
abs(x-g(x))    7.94316e-10
Iter =   39, x =    0.5671432901, g(x) =    0.5671432906
abs(x-g(x))    4.50491e-10
Iter =   40, x =    0.5671432906, g(x) =    0.5671432903
abs(x-g(x))    2.55493e-10
Iter =   41, x =    0.5671432903, g(x) =    0.5671432905
abs(x-g(x))    1.44901e-10
Iter =   42, x =    0.5671432905, g(x) =    0.5671432904
abs(x-g(x))    8.21796e-11
Iterasi konvergen: akar ditemukan

```

```

plt.clf()
x = np.linspace(0,1,500)
plt.plot(x, x, label="$f(x) = x$")
plt.plot(x, func_03(x), label="$g(x) = \exp(-x)$")
plt.plot(xr, func_03(xr), marker="o")
plt.legend()
plt.grid()
plt.savefig("figures/func_03_2.pdf")

```



```

x0 = 1.0
xr = fixed_point(func_03, x0)

```

```

Iter =    1, x =    1.0000000000, g(x) =    0.3678794412
abs(x-g(x))  6.32121e-01
Iter =    2, x =    0.3678794412, g(x) =    0.6922006276
abs(x-g(x))  3.24321e-01
Iter =    3, x =    0.6922006276, g(x) =    0.5004735006
abs(x-g(x))  1.91727e-01
Iter =    4, x =    0.5004735006, g(x) =    0.6062435351

```

```

abs(x-g(x))      1.05770e-01
Iter =      5, x =      0.6062435351, g(x) =      0.5453957860
abs(x-g(x))      6.08477e-02
Iter =      6, x =      0.5453957860, g(x) =      0.5796123355
abs(x-g(x))      3.42165e-02
Iter =      7, x =      0.5796123355, g(x) =      0.5601154614
abs(x-g(x))      1.94969e-02
Iter =      8, x =      0.5601154614, g(x) =      0.5711431151
abs(x-g(x))      1.10277e-02
Iter =      9, x =      0.5711431151, g(x) =      0.5648793474
abs(x-g(x))      6.26377e-03
Iter =     10, x =      0.5648793474, g(x) =      0.5684287250
abs(x-g(x))      3.54938e-03
Iter =     11, x =      0.5684287250, g(x) =      0.5664147331
abs(x-g(x))      2.01399e-03
Iter =     12, x =      0.5664147331, g(x) =      0.5675566373
abs(x-g(x))      1.14190e-03
Iter =     13, x =      0.5675566373, g(x) =      0.5669089119
abs(x-g(x))      6.47725e-04
Iter =     14, x =      0.5669089119, g(x) =      0.5672762322
abs(x-g(x))      3.67320e-04
Iter =     15, x =      0.5672762322, g(x) =      0.5670678984
abs(x-g(x))      2.08334e-04
Iter =     16, x =      0.5670678984, g(x) =      0.5671860501
abs(x-g(x))      1.18152e-04
Iter =     17, x =      0.5671860501, g(x) =      0.5671190401
abs(x-g(x))      6.70100e-05
Iter =     18, x =      0.5671190401, g(x) =      0.5671570440
abs(x-g(x))      3.80039e-05
Iter =     19, x =      0.5671570440, g(x) =      0.5671354902
abs(x-g(x))      2.15538e-05
Iter =     20, x =      0.5671354902, g(x) =      0.5671477143
abs(x-g(x))      1.22241e-05
Iter =     21, x =      0.5671477143, g(x) =      0.5671407815
abs(x-g(x))      6.93280e-06
Iter =     22, x =      0.5671407815, g(x) =      0.5671447133
abs(x-g(x))      3.93189e-06
Iter =     23, x =      0.5671447133, g(x) =      0.5671424834
abs(x-g(x))      2.22995e-06
Iter =     24, x =      0.5671424834, g(x) =      0.5671437481
abs(x-g(x))      1.26470e-06
Iter =     25, x =      0.5671437481, g(x) =      0.5671430308
abs(x-g(x))      7.17265e-07
Iter =     26, x =      0.5671430308, g(x) =      0.5671434376
abs(x-g(x))      4.06792e-07
Iter =     27, x =      0.5671434376, g(x) =      0.5671432069
abs(x-g(x))      2.30709e-07
Iter =     28, x =      0.5671432069, g(x) =      0.5671433378
abs(x-g(x))      1.30845e-07
Iter =     29, x =      0.5671433378, g(x) =      0.5671432636
abs(x-g(x))      7.42080e-08
Iter =     30, x =      0.5671432636, g(x) =      0.5671433056
abs(x-g(x))      4.20866e-08
Iter =     31, x =      0.5671433056, g(x) =      0.5671432818
abs(x-g(x))      2.38691e-08
Iter =     32, x =      0.5671432818, g(x) =      0.5671432953
abs(x-g(x))      1.35372e-08
Iter =     33, x =      0.5671432953, g(x) =      0.5671432876
abs(x-g(x))      7.67754e-09
Iter =     34, x =      0.5671432876, g(x) =      0.5671432920
abs(x-g(x))      4.35427e-09
Iter =     35, x =      0.5671432920, g(x) =      0.5671432895
abs(x-g(x))      2.46949e-09
Iter =     36, x =      0.5671432895, g(x) =      0.5671432909

```

```

abs(x-g(x))    1.40056e-09
Iter =   37, x =    0.5671432909, g(x) =    0.5671432901
abs(x-g(x))    7.94316e-10
Iter =   38, x =    0.5671432901, g(x) =    0.5671432906
abs(x-g(x))    4.50491e-10
Iter =   39, x =    0.5671432906, g(x) =    0.5671432903
abs(x-g(x))    2.55493e-10
Iter =   40, x =    0.5671432903, g(x) =    0.5671432905
abs(x-g(x))    1.44901e-10
Iter =   41, x =    0.5671432905, g(x) =    0.5671432904
abs(x-g(x))    8.21796e-11
Iterasi konvergen: akar ditemukan

```

```

x0 = -10.0
xr = fixed_point(func_03, x0)

```

```

Iter =    1, x =   -10.0000000000, g(x) =  22026.4657948067
abs(x-g(x))  2.20365e+04
Iter =    2, x =  22026.4657948067, g(x) =    0.0000000000
abs(x-g(x))  2.20265e+04
Iter =    3, x =    0.0000000000, g(x) =    1.0000000000
abs(x-g(x))  1.00000e+00
Iter =    4, x =    1.0000000000, g(x) =    0.3678794412
abs(x-g(x))  6.32121e-01
Iter =    5, x =    0.3678794412, g(x) =    0.6922006276
abs(x-g(x))  3.24321e-01
Iter =    6, x =    0.6922006276, g(x) =    0.5004735006
abs(x-g(x))  1.91727e-01
Iter =    7, x =    0.5004735006, g(x) =    0.6062435351
abs(x-g(x))  1.05770e-01
Iter =    8, x =    0.6062435351, g(x) =    0.5453957860
abs(x-g(x))  6.08477e-02
Iter =    9, x =    0.5453957860, g(x) =    0.5796123355
abs(x-g(x))  3.42165e-02
Iter =   10, x =    0.5796123355, g(x) =    0.5601154614
abs(x-g(x))  1.94969e-02
Iter =   11, x =    0.5601154614, g(x) =    0.5711431151
abs(x-g(x))  1.10277e-02
Iter =   12, x =    0.5711431151, g(x) =    0.5648793474
abs(x-g(x))  6.26377e-03
Iter =   13, x =    0.5648793474, g(x) =    0.5684287250
abs(x-g(x))  3.54938e-03
Iter =   14, x =    0.5684287250, g(x) =    0.5664147331
abs(x-g(x))  2.01399e-03
Iter =   15, x =    0.5664147331, g(x) =    0.5675566373
abs(x-g(x))  1.14190e-03
Iter =   16, x =    0.5675566373, g(x) =    0.5669089119
abs(x-g(x))  6.47725e-04
Iter =   17, x =    0.5669089119, g(x) =    0.5672762322
abs(x-g(x))  3.67320e-04
Iter =   18, x =    0.5672762322, g(x) =    0.5670678984
abs(x-g(x))  2.08334e-04
Iter =   19, x =    0.5670678984, g(x) =    0.5671860501
abs(x-g(x))  1.18152e-04
Iter =   20, x =    0.5671860501, g(x) =    0.5671190401
abs(x-g(x))  6.70100e-05
Iter =   21, x =    0.5671190401, g(x) =    0.5671570440
abs(x-g(x))  3.80039e-05
Iter =   22, x =    0.5671570440, g(x) =    0.5671354902
abs(x-g(x))  2.15538e-05
Iter =   23, x =    0.5671354902, g(x) =    0.5671477143
abs(x-g(x))  1.22241e-05

```



```

Iter = 24, x = 0.5671477143, g(x) = 0.5671407815
abs(x-g(x)) 6.93280e-06
Iter = 25, x = 0.5671407815, g(x) = 0.5671447133
abs(x-g(x)) 3.93189e-06
Iter = 26, x = 0.5671447133, g(x) = 0.5671424834
abs(x-g(x)) 2.22995e-06
Iter = 27, x = 0.5671424834, g(x) = 0.5671437481
abs(x-g(x)) 1.26470e-06
Iter = 28, x = 0.5671437481, g(x) = 0.5671430308
abs(x-g(x)) 7.17265e-07
Iter = 29, x = 0.5671430308, g(x) = 0.5671434376
abs(x-g(x)) 4.06792e-07
Iter = 30, x = 0.5671434376, g(x) = 0.5671432069
abs(x-g(x)) 2.30709e-07
Iter = 31, x = 0.5671432069, g(x) = 0.5671433378
abs(x-g(x)) 1.30845e-07
Iter = 32, x = 0.5671433378, g(x) = 0.5671432636
abs(x-g(x)) 7.42080e-08
Iter = 33, x = 0.5671432636, g(x) = 0.5671433056
abs(x-g(x)) 4.20866e-08
Iter = 34, x = 0.5671433056, g(x) = 0.5671432818
abs(x-g(x)) 2.38691e-08
Iter = 35, x = 0.5671432818, g(x) = 0.5671432953
abs(x-g(x)) 1.35372e-08
Iter = 36, x = 0.5671432953, g(x) = 0.5671432876
abs(x-g(x)) 7.67754e-09
Iter = 37, x = 0.5671432876, g(x) = 0.5671432920
abs(x-g(x)) 4.35427e-09
Iter = 38, x = 0.5671432920, g(x) = 0.5671432895
abs(x-g(x)) 2.46949e-09
Iter = 39, x = 0.5671432895, g(x) = 0.5671432909
abs(x-g(x)) 1.40056e-09
Iter = 40, x = 0.5671432909, g(x) = 0.5671432901
abs(x-g(x)) 7.94316e-10
Iter = 41, x = 0.5671432901, g(x) = 0.5671432906
abs(x-g(x)) 4.50491e-10
Iter = 42, x = 0.5671432906, g(x) = 0.5671432903
abs(x-g(x)) 2.55493e-10
Iter = 43, x = 0.5671432903, g(x) = 0.5671432905
abs(x-g(x)) 1.44901e-10
Iter = 44, x = 0.5671432905, g(x) = 0.5671432904
abs(x-g(x)) 8.21796e-11
Iterasi konvergen: akar ditemukan

```

## 1.4 Metode Newton-Raphson

Metode Newton-Raphson adalah salah satu metode yang paling sering digunakan untuk mencari akar persamaan nonlinear. Metode ini memerlukan informasi tebakan awal akar dan turunan pertama dari fungsi yang akan dicari akarnya.

Metode Newton-Raphson dapat diturunkan dari deret Taylor untuk  $f(x)$  disekitar  $x$ :

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \mathcal{O}(x_{i+1} - x_i)^2$$

Jika  $x_{i+1}$  adalah akar dari  $f(x) = 0$  maka diperoleh:

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \mathcal{O}(x_{i+1} - x_i)^2$$

Dengan mengasumsikan  $x_i$  dekat dengan  $x_{i+1}$ , suku  $\mathcal{O}(x_{i+1} - x_i)$  dapat dianggap nol sehingga diperoleh:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Sebagai contoh, kita akan menghitung akar dari persamaan  $f(x) = x^3 - 35$

```
def func_04(x):
    return x**3 - 35.0

def dfunc_04(x):
    return 3*x**2
```

```
plt.clf()
x = np.linspace(2,4,500)
plt.plot(x, func_04(x))
plt.grid(); plt.savefig("figures/func_04.pdf")
```

```
def newton_raphson(f, df, x, TOL=1e-10, NiterMax=100):

    SMALL = np.finfo(float).eps

    for i in range(1, NiterMax+1):
        fx = f(x)
        dfx = df(x)

        if abs(dfx) <= SMALL:
            raise RuntimeError("Turunan f(x) sangat kecil")

        xr = x - fx/dfx

        print("Iter = %5d, x = %18.10f, abs(f(x)) %15.5e" % (i, x, abs(fx)))
        if abs(fx) <= TOL:
            print("Iterasi konvergen: akar ditemukan")
            return x

        x = xr

    print("WARNING: Konvergensi tidak diperoleh setelah %d iterasi" % NiterMax)
    print("WARNING: Nilai tebakan akhir akan dikembalikan")
    return x
```

Kita akan coba mencari akar persamaan `func_04` dengan beberapa tebakan awal.

```
x0 = 3.0
xr = newton_raphson(func_04, dfunc_04, x0)
```

```
Iter =    1, x =      3.0000000000, abs(f(x))      8.00000e+00
Iter =    2, x =      3.2962962963, abs(f(x))      8.16136e-01
Iter =    3, x =      3.2712589288, abs(f(x))      6.18335e-03
Iter =    4, x =      3.2710663215, abs(f(x))      3.64060e-07
Iter =    5, x =      3.2710663102, abs(f(x))      0.00000e+00
Iterasi konvergen: akar ditemukan
```

```
x0 = 4.0
xr = newton_raphson(func_04, dfunc_04, x0)
```

```
Iter =    1, x =      4.0000000000, abs(f(x))      2.90000e+01
Iter =    2, x =      3.3958333333, abs(f(x))      4.15968e+00
Iter =    3, x =      3.2755942598, abs(f(x))      1.45547e-01
Iter =    4, x =      3.2710725664, abs(f(x))      2.00823e-04
Iter =    5, x =      3.2710663102, abs(f(x))      3.84091e-10
Iter =    6, x =      3.2710663102, abs(f(x))      0.00000e+00
Iterasi konvergen: akar ditemukan
```

```
x0 = 10.0
xr = newton_raphson(func_04, dfunc_04, x0)
```

```
Iter = 1, x = 10.000000000, abs(f(x)) 9.65000e+02
Iter = 2, x = 6.783333333, abs(f(x)) 2.77126e+02
Iter = 3, x = 4.7757703873, abs(f(x)) 7.39257e+01
Iter = 4, x = 3.6953637378, abs(f(x)) 1.54628e+01
Iter = 5, x = 3.3179190978, abs(f(x)) 1.52560e+00
Iter = 6, x = 3.2717248105, abs(f(x)) 2.11419e-02
Iter = 7, x = 3.2710664427, abs(f(x)) 4.25408e-06
Iter = 8, x = 3.2710663102, abs(f(x)) 1.70530e-13
Iterasi konvergen: akar ditemukan
```

Kita coba mencari akar dari func\_01. Kita perlu mendefinisikan turunan dari func\_01.

```
def func_01(x):
    return 5*x**3 - 5*x**2 + 6*x - 2

def dfunc_01(x):
    return 15*x**2 - 10*x + 6
```

```
xr = newton_raphson(func_01, dfunc_01, 0.0)
```

```
Iter = 1, x = 0.000000000, abs(f(x)) 2.00000e+00
Iter = 2, x = 0.333333333, abs(f(x)) 3.70370e-01
Iter = 3, x = 0.4188034188, abs(f(x)) 3.12185e-03
Iter = 4, x = 0.4181007594, abs(f(x)) 6.31253e-07
Iter = 5, x = 0.4181006173, abs(f(x)) 2.57572e-14
Iterasi konvergen: akar ditemukan
```

```
xr = newton_raphson(func_01, dfunc_01, 1.0)
```

```
Iter = 1, x = 1.000000000, abs(f(x)) 4.00000e+00
Iter = 2, x = 0.6363636364, abs(f(x)) 1.08189e+00
Iter = 3, x = 0.4469148796, abs(f(x)) 1.29143e-01
Iter = 4, x = 0.4183866706, abs(f(x)) 1.27050e-03
Iter = 5, x = 0.4181006407, abs(f(x)) 1.04260e-07
Iter = 6, x = 0.4181006173, abs(f(x)) 8.88178e-16
Iterasi konvergen: akar ditemukan
```

```
xr = newton_raphson(func_01, dfunc_01, 10.0)
```

```
Iter = 1, x = 10.000000000, abs(f(x)) 4.55800e+03
Iter = 2, x = 6.7581792319, abs(f(x)) 1.35352e+03
Iter = 3, x = 4.5873900492, abs(f(x)) 4.02992e+02
Iter = 4, x = 3.1261527539, abs(f(x)) 1.20650e+02
Iter = 5, x = 2.1317684971, abs(f(x)) 3.65069e+01
Iter = 6, x = 1.4409899718, abs(f(x)) 1.12244e+01
Iter = 7, x = 0.9473245383, abs(f(x)) 3.44759e+00
Iter = 8, x = 0.6021555613, abs(f(x)) 8.91659e-01
Iter = 9, x = 0.4375613295, abs(f(x)) 8.69457e-02
Iter = 10, x = 0.4182241073, abs(f(x)) 5.48453e-04
Iter = 11, x = 0.4181006216, abs(f(x)) 1.94077e-08
Iter = 12, x = 0.4181006173, abs(f(x)) 0.00000e+00
Iterasi konvergen: akar ditemukan
```

Sebagai perbandingan dengan metode fixed-point, kita akan menghitung akar dari persamaan  $f(x) = e^{-x} - x$ . Turunan pertama dari fungsi ini adalah  $f'(x) = -e^{-x} - 1$

```
def func_05(x):
    return np.exp(-x) - x

def dfunc_05(x):
    return -np.exp(-x) - 1
```

```
x0 = 0.0
xr = newton_raphson(func_05, dfunc_05, x0)
```

```
Iter =    1, x =      0.0000000000, abs(f(x))      1.00000e+00
Iter =    2, x =      0.5000000000, abs(f(x))      1.06531e-01
Iter =    3, x =      0.5663110032, abs(f(x))      1.30451e-03
Iter =    4, x =      0.5671431650, abs(f(x))      1.96480e-07
Iter =    5, x =      0.5671432904, abs(f(x))      4.44089e-15
Iterasi konvergen: akar ditemukan
```

Dapat diamati bahwa metode Newton-Raphson konvergen dengan cepat dibandingkan dengan metode fixed-point.

## 1.5 Metode secant

Metode secant menggunakan ide yang sama dengan metode Newton-Raphson. Perbedaannya adalah metode secant menggunakan aproksimasi terhadap turunan pertama dari  $f(x)$ .

$$f'(x) \approx \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

```
def secant(f, x, TOL=1e-10, NiterMax=100, DELTA=0.001):
    SMALL = np.finfo(float).eps
    # Untuk aproksimasi turunan pertama
    x_old = x + DELTA
    for i in range(1, NiterMax+1):
        fx = f(x)
        fx_old = f(x_old)
        dfx = (fx_old - fx) / (x_old - x)
        if abs(dfx) <= SMALL:
            raise RuntimeError("Turunan f(x) sangat kecil")
        xr = x - fx/dfx
        print("Iter = %5d, x = %18.10f, abs(f(x)) %15.5e" % (i, x, abs(fx)))
        if abs(fx) <= TOL:
            print("Iterasi konvergen: akar ditemukan")
            return x
        x_old = x
        x = xr
    print("WARNING: Konvergensi tidak diperoleh setelah %d iterasi" % NiterMax)
    print("WARNING: Nilai tebakan akhir akan dikembalikan")
    return x
```

```
x0 = 0.0
xr = secant(func_05, x0)
```

```
Iter =    1, x =      0.0000000000, abs(f(x))      1.00000e+00
Iter =    2, x =      0.5001249896, abs(f(x))      1.06330e-01
Iter =    3, x =      0.5596304161, abs(f(x))      1.17898e-02
Iter =    4, x =      0.5670511519, abs(f(x))      1.44397e-04
Iter =    5, x =      0.5671431650, abs(f(x))      1.96523e-07
Iter =    6, x =      0.5671432904, abs(f(x))      3.27660e-12
Iterasi konvergen: akar ditemukan
```

```
x0 = 0.0
xr = secant(func_01, x0)
```

```
Iter =    1, x =      0.0000000000, abs(f(x))    2.00000e+00
Iter =    2, x =      0.3336110645, abs(f(x))    3.69167e-01
Iter =    3, x =      0.4091296137, abs(f(x))    3.97425e-02
Iter =    4, x =      0.4182403537, abs(f(x))    6.20610e-04
Iter =    5, x =      0.4181002700, abs(f(x))    1.54234e-06
Iter =    6, x =      0.4181006172, abs(f(x))    6.17357e-11
Iterasi konvergen: akar ditemukan
```

```
x0 = 1.0
xr = secant(func_04, x0)
```

```
Iter =    1, x =      1.0000000000, abs(f(x))    3.40000e+01
Iter =    2, x =     12.3220075518, abs(f(x))    1.83587e+03
Iter =    3, x =      1.2058686147, abs(f(x))    3.32465e+01
Iter =    4, x =      1.4035942709, abs(f(x))    3.22348e+01
Iter =    5, x =      7.7034593939, abs(f(x))    4.22149e+02
Iter =    6, x =      1.8505185189, abs(f(x))    2.86630e+01
Iter =    7, x =      2.2226542773, abs(f(x))    2.40197e+01
Iter =    8, x =      4.1476654265, abs(f(x))    3.63528e+01
Iter =    9, x =      2.9885349159, abs(f(x))    8.30838e+00
Iter =   10, x =      3.2041693472, abs(f(x))    2.10375e+00
Iter =   11, x =      3.2772827235, abs(f(x))    1.99924e-01
Iter =   12, x =      3.2709375923, abs(f(x))    4.13163e-03
Iter =   13, x =      3.2710660659, abs(f(x))    7.84243e-06
Iter =   14, x =      3.2710663102, abs(f(x))    3.08610e-10
Iter =   15, x =      3.2710663102, abs(f(x))    0.00000e+00
Iterasi konvergen: akar ditemukan
```



Bab 2

Test pweave