# Fixpunkt-Iteration

January 5, 2021

*Analysis I Winter 2020/21 - Vorlesung Woche 10*

## 1 Fixpunkt-Iteration nach Bannachschem Fixpunktsatz

**Jörn Behrens** (joern.behrens@uni-hamburg.de)

### 1.1 Einführung

Der Bannachsche Fixpunktsatz im $\mathbb{R}$ lautet:

Sei $f : I \to I$ eine Funktion die $I \subset \mathbb{R}$ in sich abbildet. Weiter gelte für alle $x, y \in I$

$$|f(x) - f(y)| \leq K|x - y|$$

mit einer von $x, y$ unabhängigen Konstanten $|K| < 1$. Dann hat $f$ genau einen Fixpunkt $\bar{x} \in I$ und die durch $x_{n+1} = f(x_n)$ definierte Iterationsfolge $(x_n)$ konvergiert für jeden beliebigen Anfangspunkt $x_0 \in I$ gegen diesen Fixpunkt.

Wir wollen diesen Satz anhand eines einfachen Beispiels demonstrieren und als Computerprogramm implementieren.

### 1.2 Gleichung

Betrachte das Polynom

$$p_3(x) = \frac{1}{4}x^3 - x + \frac{1}{5}.$$

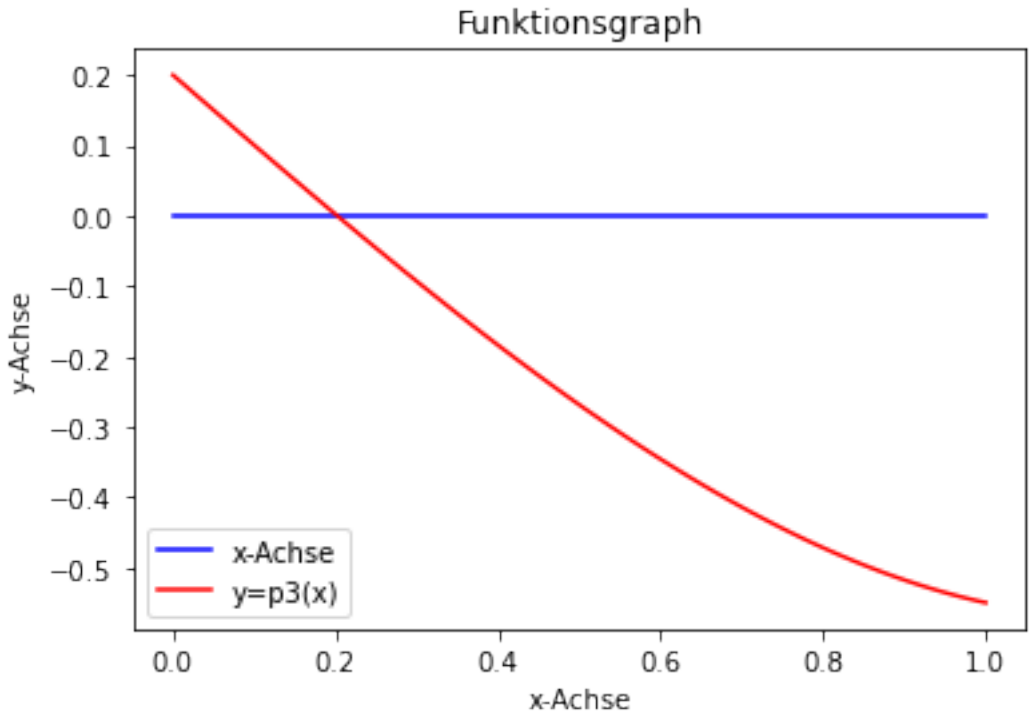
Wir wollen die Nullstellen des Polynoms mit Hilfe einer Fixpunkt-Iteration berechnen:

$$x = f(x) := \frac{1}{4}x^3 + \frac{1}{5}$$

### 1.3 Skizze

[1]:
```
from numpy import linspace, zeros, size
import matplotlib.pyplot as plt
%matplotlib inline
x=linspace(0,1,100,endpoint='true')
y=0.25*x**3-x+0.2
z=zeros(size(x))
plt.plot(x,z,'b-',label='x-Achse')
plt.plot(x,y,'r-',label='y=p3(x)')
```

```
plt.legend(loc='lower left')
plt.title('Funktionsgraph')
plt.xlabel('x-Achse')
plt.ylabel('y-Achse')
plt.show()
```




### 1.4 Voraussetzungen

Wir stellen zunächst fest, dass $f : [0, 1] \to [0, 1]$ das Einheitsintervall auf sich selbst abbildet. Außerdem entnimmt man der graphischen Skizze, dass auch $|K| < 1$. Damit sind die Voraussetzungen für den Bannachschen Fixpunktsatz erfüllt.

### 1.5 Fixpunkt-Iteration

Wir wollen jetzt ein Programm zur Fixpunkt-Iteration einführen und den Fixpunkt, also die Nullstelle von $p_3$ berechnen.

Wir definieren zunächst die Funktion $f$, welche für die Fixpunkt-Iteration benötigt wird.

[2]:
```
def ff(x):
    y= 0.25*x**3 + 0.2
    return y
```

Nun starten wir vom Anfangswert $x_0 = \frac{1}{2}$ und führen die erste Iteration durch.

[3]:
```
iter=10
i=1
xx=zeros(iter)
xx[0]= 0.5
xx[1]=ff(xx[0])
s = 'Iteration: '+repr(i)+', x-value: '+repr(xx[i])
print(s)
```

```
Iteration: 1, x-value: 0.23125
```

Jetzt wiederholen wir diese Prozedur in einer Schleife.

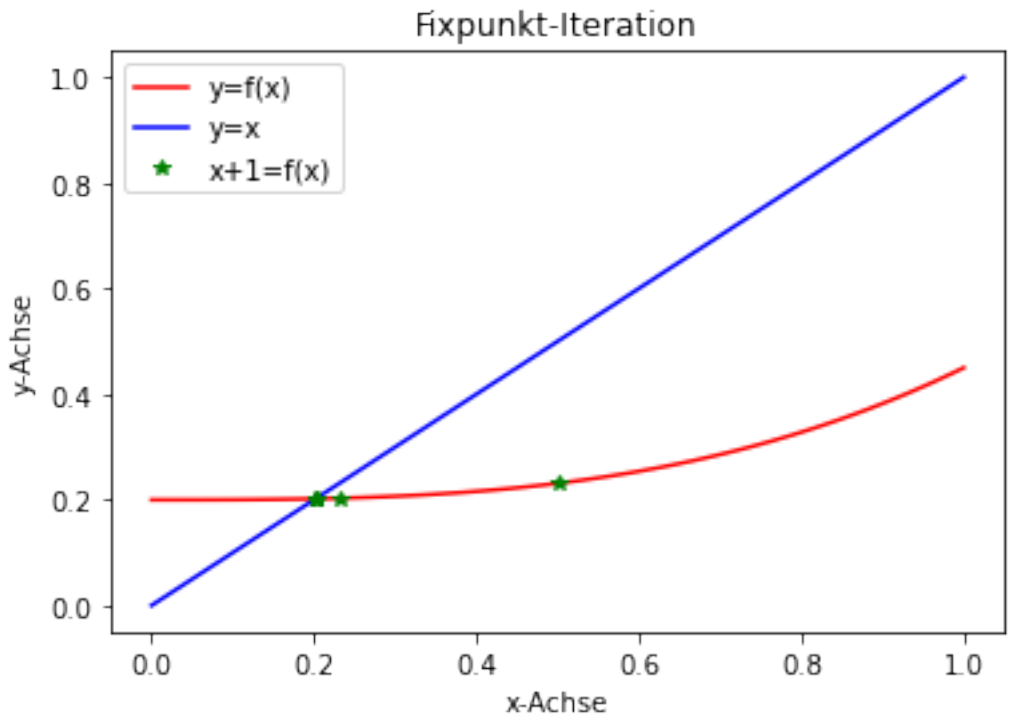
[4]:
```
for k in range(iter-1):
    xx[k+1]=ff(xx[k])
    s = 'Iteration: '+repr(k+1)+', x-value: '+repr(xx[k+1])
    print(s)
```

```
Iteration: 1, x-value: 0.23125
Iteration: 2, x-value: 0.20309161376953125
Iteration: 3, x-value: 0.20209418951191055
Iteration: 4, x-value: 0.20206348582589903
Iteration: 5, x-value: 0.20206254546740715
Iteration: 6, x-value: 0.20206251667165848
Iteration: 7, x-value: 0.2020625157898765
Iteration: 8, x-value: 0.20206251576287462
Iteration: 9, x-value: 0.20206251576204776
```

Wir sehen, dass in der 9. Iteration schon 11 Nachkommastellen unverändert bleiben.

Nun zeichnen wir noch einen Graphen.

[5]:
```
f=ff(x)
plt.plot(x,f,'r-',label='y=f(x)')
plt.plot(x,x,'b-',label='y=x')
plt.plot(xx[0:iter-1],xx[1:iter],'g*',label='x+1=f(x)')
plt.legend(loc='upper left')
plt.title('Fixpunkt-Iteration')
plt.xlabel('x-Achse')
plt.ylabel('y-Achse')
plt.show()
```


Fixpunkt-Iteration
y=f(x)
y=x
x+1=f(x)
y-Achse
x-Achse
0.0
0.2
0.4
0.6
0.8
1.0

[ ]: