AppliedW5

March 21, 2024

[66]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.linalg import norm
```

[67]:
```
f = lambda x: x**2 - 2
df = lambda x: 2*x
x0 = 1
x1 = 2
```

[68]:
```
def convergence(X):
    return np.log(np.abs((X[-1] - X[-2])/(X[-2]-X[-3])))/np.log(np.abs((X[-2] -␣
 ↪X[-3])/(X[-3]-X[-4])))

def norm2(x):
    return norm(x, ord=2)
```

1 Newton

[69]:
```
def newton(f, df, x0, tol=1e-10, max_iter = 1e3):
    i = 0
    X = [x0]
    while abs(f(x0)) > tol and i < max_iter:
        print('Iteration: {0}, f({1})={2}'.format(i, x0, f(x0)))
        x0 = x0 - f(x0)/df(x0)
        X += [x0]
        i += 1
    if i >= max_iter:
        print("Maximum Iterations Exceeded")
    return x0, X
```

[70]:
```
x, X_newt = newton(f=f, df=df, x0=x0)
x
```

```
Iteration: 0, f(1)=-1
Iteration: 1, f(1.5)=0.25
Iteration: 2, f(1.4166666666666667)=0.006944444444444642
Iteration: 3, f(1.4142156862745099)=6.007304882871267e-06
```

[70]: 1.4142135623746899

[71]:
```
convergence(X_newt)
```

[71]: 1.9995089548529266

2 Seacant

[72]:
```
def seacant(f, x0, x1, tol=1e-10, max_iter = 1e3):
    i = 0
    x2 = x1
    X = [x1, x2]
    while abs(f(x2)) > tol and i < max_iter:
        print('Iteration: {0}, f({1})={2}'.format(i, x2, f(x2)))
        x2 = x1 - f(x1)*((x1 - x0)/(f(x1)- f(x0)))
        x0 = x1
        x1 = x2
        X += [x2]
        i += 1

    if i >= max_iter:
        print("Maximum Iterations Exceeded")
    return x2, X
```

[73]:
```
x, X_sec = seacant(f=f, x0=x0, x1=x1)
x
```

```
Iteration: 0, f(2)=2
Iteration: 1, f(1.3333333333333335)=-0.22222222222222188
Iteration: 2, f(1.4000000000000001)=-0.03999999999999959
Iteration: 3, f(1.4146341463414633)=0.0011897679952408424
Iteration: 4, f(1.41421143847487)=-6.007286838860537e-06
Iteration: 5, f(1.4142135620573204)=-8.931455575122982e-10
```

[73]: 1.4142135623730954

[74]:
```
convergence(X_sec)
```

[74]: 1.6649584093105616

3 Newtons Multi Dimension

[75]:
```
def norm2(x):
    return np.linalg.norm(x, ord=2)
```

[76]:
```
def newton_ND(f, J, x0, tol=1e-10, max_iter = 1e3):
    i = 0
    h = np.ones_like(x0)
    while norm2(h) > tol and i < max_iter:
        print('Iteration: {0}, f({1})={2}'.format(i, x0, f(x0)))
        h = np.dot(np.linalg.inv(J(x0)),f(x0))
        x0 = x0 - h
        i += 1
    if i >= max_iter:
        print("Maximum Iterations Exceeded")
    return x0
```

[77]:
```
f1 = lambda x: x[0]**2 + x[1]**2 - 4
f2 = lambda x: np.exp(x[0]) + np.exp(x[1]) - 2

F = lambda x: np.array([f1(x), f2(x)])

df1dx1 = lambda x: 2*x[0]
df1dx2 = lambda x: 2*x[1]
df2dx1 = lambda x: np.exp(x[0])
df2dx2 = lambda x: np.exp(x[1])

J = lambda x: np.array([
    [df1dx1(x), df1dx2(x)],
    [df2dx1(x), df2dx2(x)]
])

X0 = np.array([0, 1])
```

[78]:
```
sol = newton_ND(f=F, J=J, x0=X0, tol=1e-10, max_iter = 1e3)
```

```
Iteration: 0, f([0 1])=[-3.          1.71828183]
Iteration: 1, f([-5.79570457  2.5       ])=[35.84019148 10.18553455]
Iteration: 2, f([-3.06468326  1.66323878])=[8.15864674 3.32304085]
Iteration: 3, f([-2.08013206  1.02473408])=[1.37702932 0.91126812]
Iteration: 4, f([-1.91392     0.69023597])=[0.13951544 0.14168709]
Iteration: 5, f([-1.90337716  0.61840608])=[0.00527068 0.00503176]
Iteration: 6, f([-1.90288625  0.61565552])=[7.80654500e-06 7.03224353e-06]
Iteration: 7, f([-1.90288545  0.61565166])=[1.55724322e-11 1.38649092e-11]
```

[79]:
```
sol, F(sol)
```

[79]:
```
(array([-1.90288545,  0.61565166]), array([0., 0.]))
```

4 Bounded Growth

Adding and subtracting N_k gives:

$$N_{k+1} = \frac{\lambda N_k}{1 + aN_k} + N_k - N_k$$

$$N_{k+1} = N_k + \frac{\lambda N_k}{1 + aN_k} - N_k$$

$$N_{k+1} = N_k + N_k \left(\frac{\lambda}{1 + aN_k} - 1 \right)$$

Thus

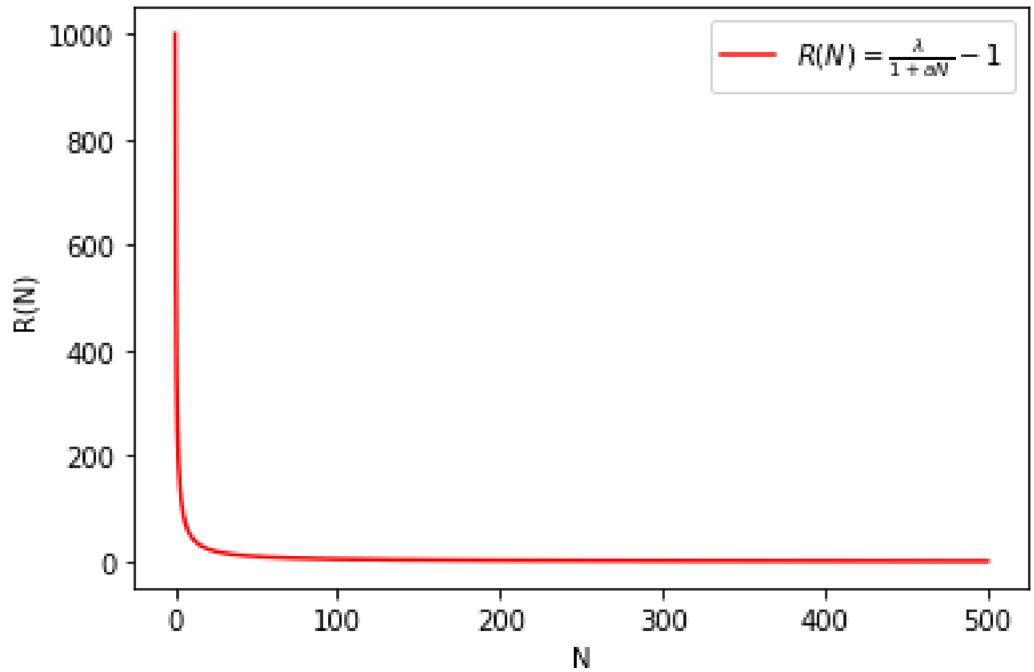
$$R(N\_k) = \frac{\lambda}{1 + aN_k} - 1$$

5 General Shape

As an example, let's choose $\lambda = 1001$ and $a = 2$. Play around with these values and see how the growth rate function changes.

[80]:
```
l = 1001
a = 2
R = lambda N: l/(1 + a * N) - 1
N = np.arange(0, 500, 0.01)
plt.plot(N, R(N), label=r'$R(N) = \frac{\lambda}{1+aN} -1$', c='r')
plt.xlabel('N')
plt.ylabel('R(N)')
plt.legend()
```

[80]: <matplotlib.legend.Legend at 0x7fecc5699280>



Unrestricted growth rate occurs when $N_k = 0$, giving:

$r = R(0) = \lambda - 1$.

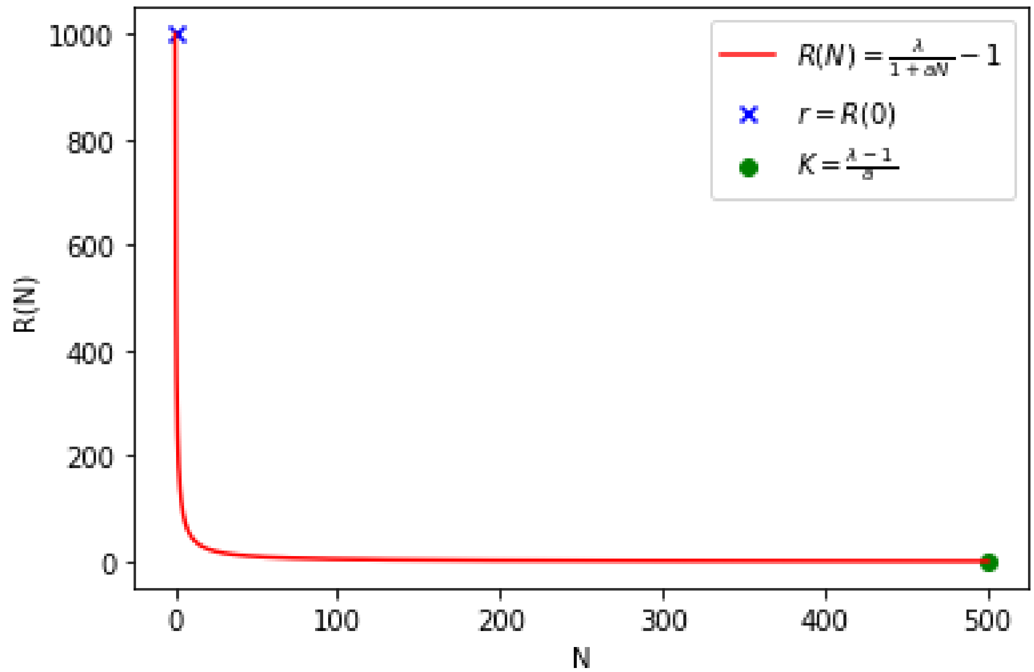
Carrying capacity occurs when $R(K) = 0$.

Solving this expression gives:

$K = (\lambda - 1)/a$.

[81]:
```
plt.plot(N, R(N), label=r'$R(N) = \frac{\lambda}{1+aN} -1$', c='r')
plt.scatter([0], [R(0)], label=r'$r=R(0)$', marker='x', color='b')
plt.scatter((l-1)/a, [0], label=r'$K=\frac{\lambda-1}{a}$', marker='o',␣
 ↪color='g')
plt.xlabel('N')
plt.ylabel('R(N)')
plt.legend()
```

[81]:
```
<matplotlib.legend.Legend at 0x7fecabf44dc0>
```



6 Cobwebbing

The first thing to note is that the fixed points of the dynamics will correspond to solutions of:

$x = x^2 + c$

So we can start by finding the solutions of:

$x^2 + c - x = 0.$

These are:

$\frac{1}{2} \pm \frac{\sqrt{1-4c}}{2}.$

This already tell us the behaviour will be qualitatively different when $c > \frac{1}{4}$ and $0 < c < \frac{1}{4}$. In the former case there is no real solutions, and thus no fixed points. In the second case there are two fixed points.

The cobweb diagram for $0 < c < 0.25$. Reveals only one point is stable, x_1^*. Any positive x_0 less than x_2^* converges to x_1^*. Starting points outside of this range will lead to a system that simply keeps growing at a quadratic rate.

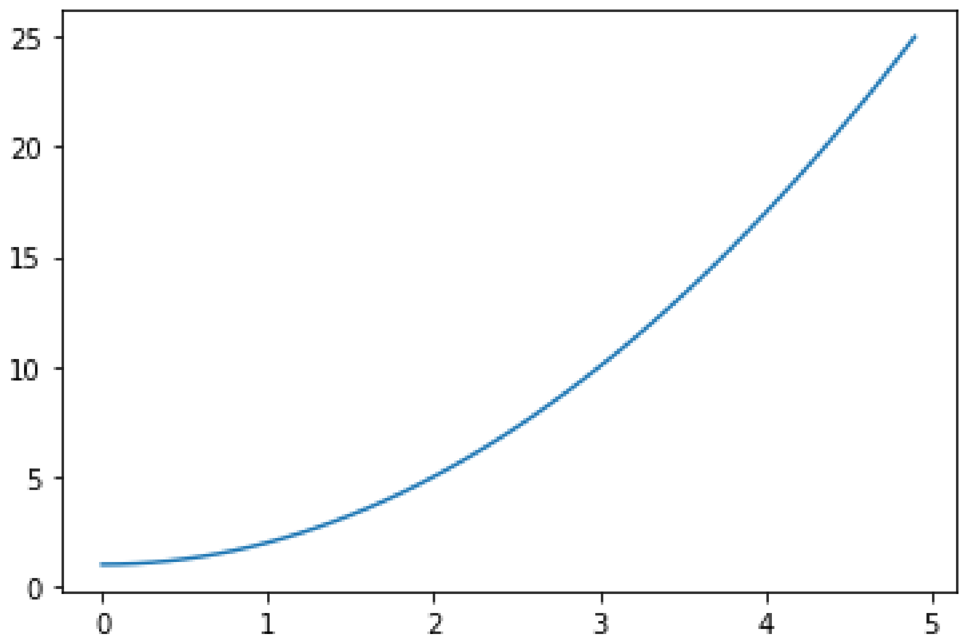
Note that for $c = \frac{1}{4}$, a single fixed point $x^* = \frac{1}{2}$ emerges.

The cobweb diagram for $c > 0.25$ shows that the variable of interest simply keeps growing at a quadratic rate, regardless of initial conditions.

[82]:
```
def DE(x, c):
    return x**2 + c
```

[83]:
```
X = np.arange(0, 5, 0.1)
X_next = DE(X, c=1)
plt.plot(X, X_next)
```

[83]: [<matplotlib.lines.Line2D at 0x7fecabe8e6d0>]

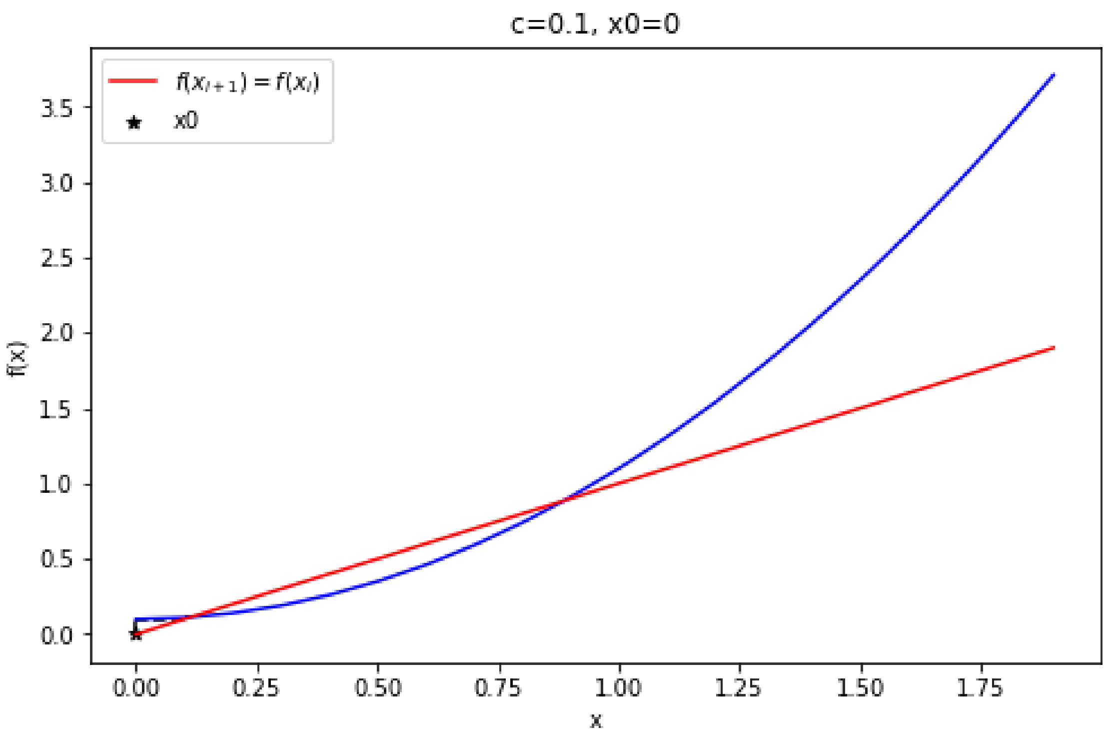


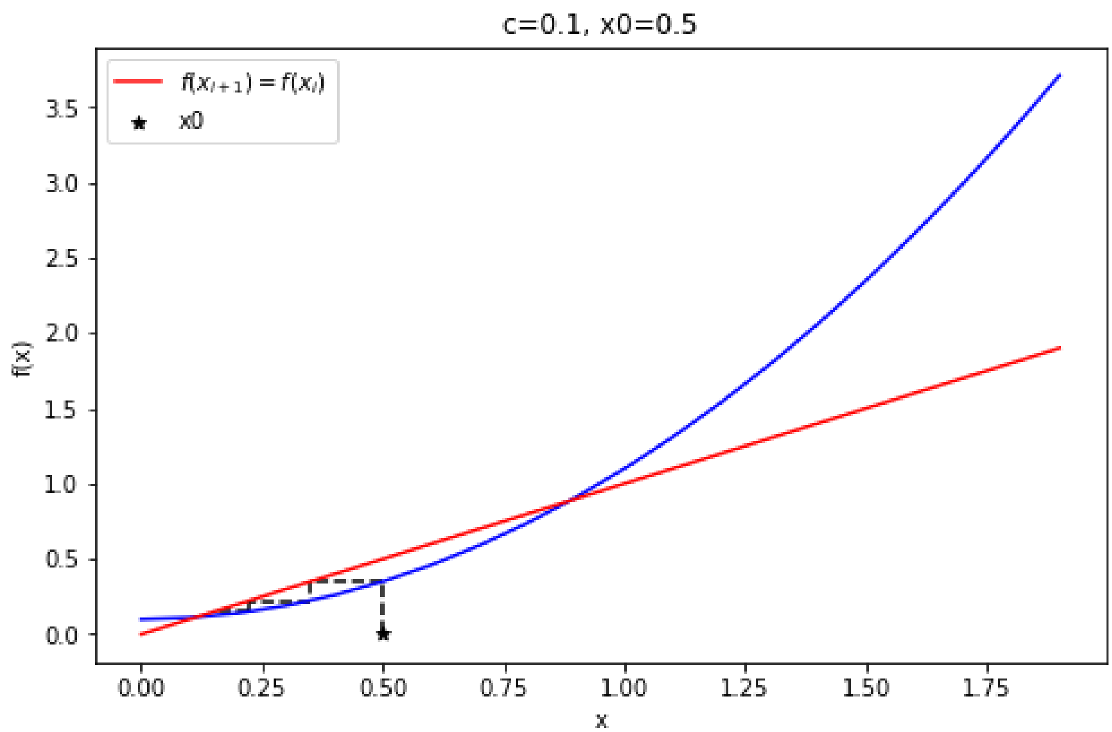
[84]:
```
def cobweb(c, x0, max_iter=3, x_max=2):
    fig, axs = plt.subplots(1, 1, sharex=True, sharey=True)
    fig.set_figheight(5)
    fig.set_figwidth(8)
    X = np.arange(0, x_max, 0.1)
    X_next = DE(x=X, c=c)
    axs.plot(X, X_next, c='b')
    axs.plot(X, X, c='r', label=r'$f(x_{i+1})=f(x_i)$')
    axs.set_title('c={0}, x0={1}'.format(str(c), str(x0)))
    axs.set_xlabel('x')
    axs.set_ylabel('f(x)')

    axs.scatter([x0], 0, label='x0', color='k', marker='*')
    for i in range(max_iter):
        x1 = DE(x=x0, c=c)
        plt.vlines(x=x0, ymin=x0 if i != 0 else 0, ymax=x1, linestyles='dashed',␣
 ↪colors='k')
```

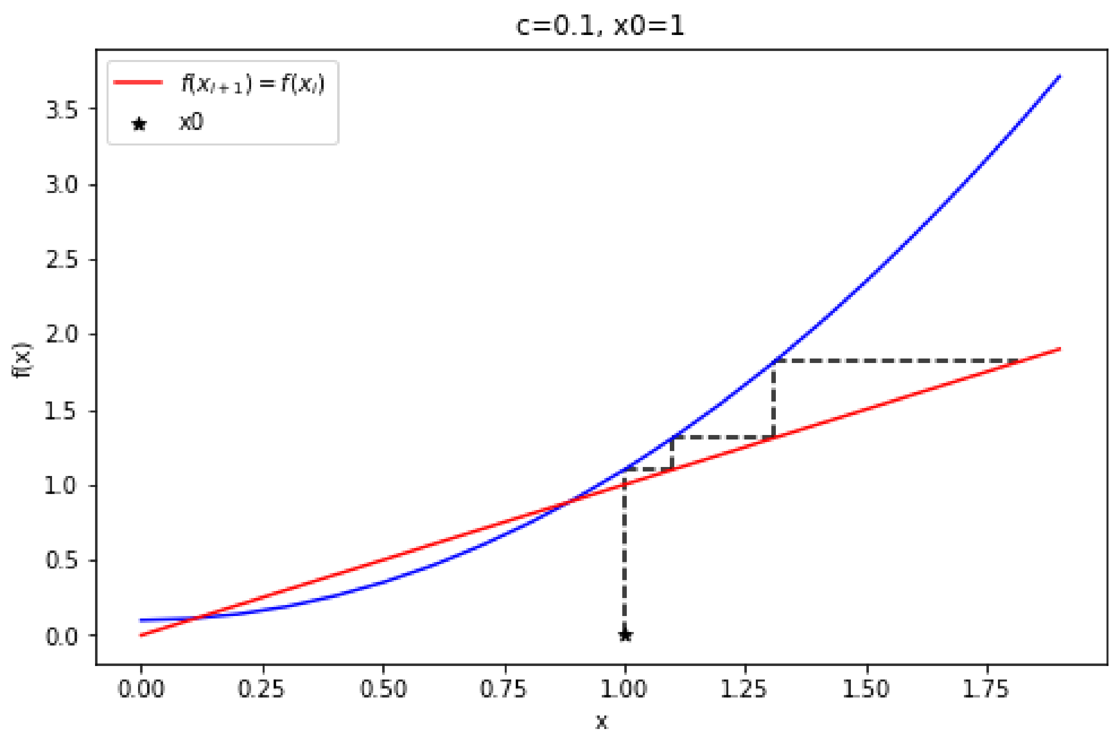
```
        plt.hlines(y=x1, xmin=x0, xmax=x1, linestyles='dashed', colors='k')
        x0 = x1

    plt.legend()
```

7 C<0.25, x0=[0, 0.5, 1]

[85]:
```
cobweb(c=0.1, x0=0)
cobweb(c=0.1, x0=0.5)
cobweb(c=0.1, x0=1)
```

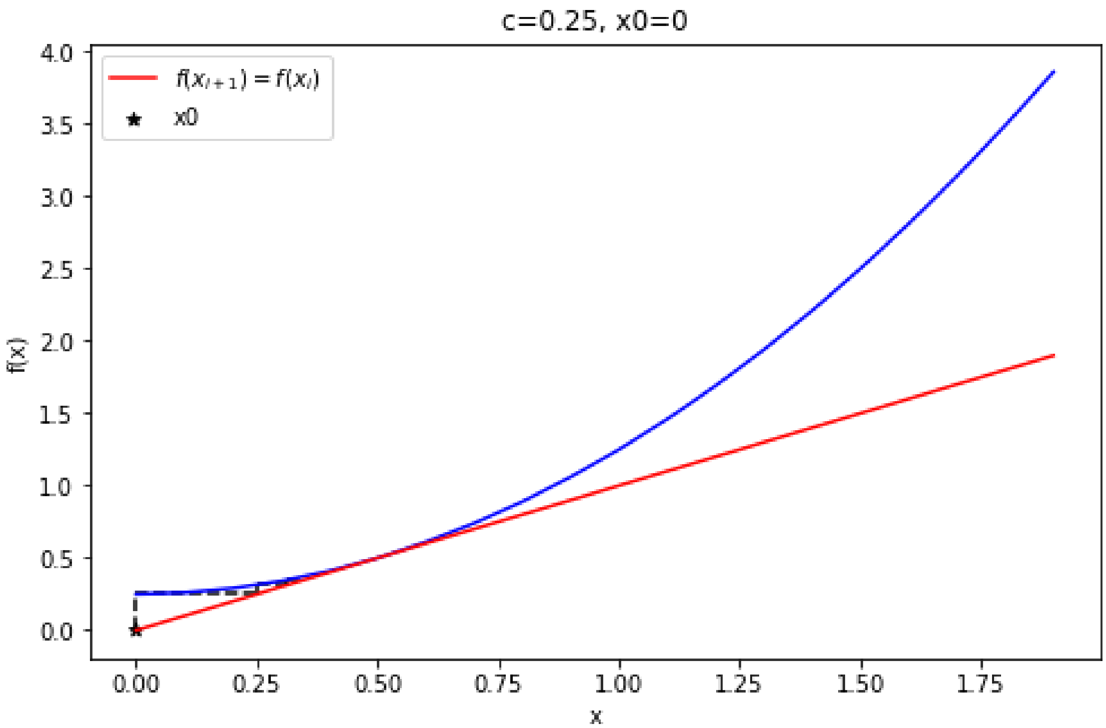


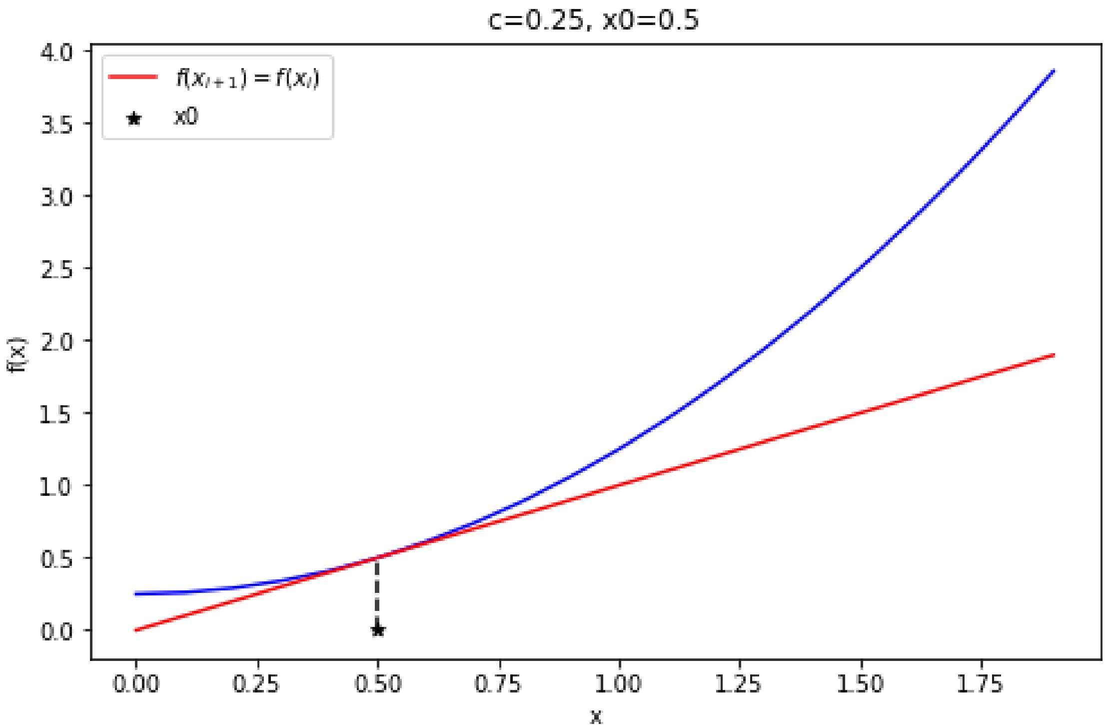

c=0.1, x0=0.5
$f(x_{i+1}) = f(x_i)$
x0
f(x)
x

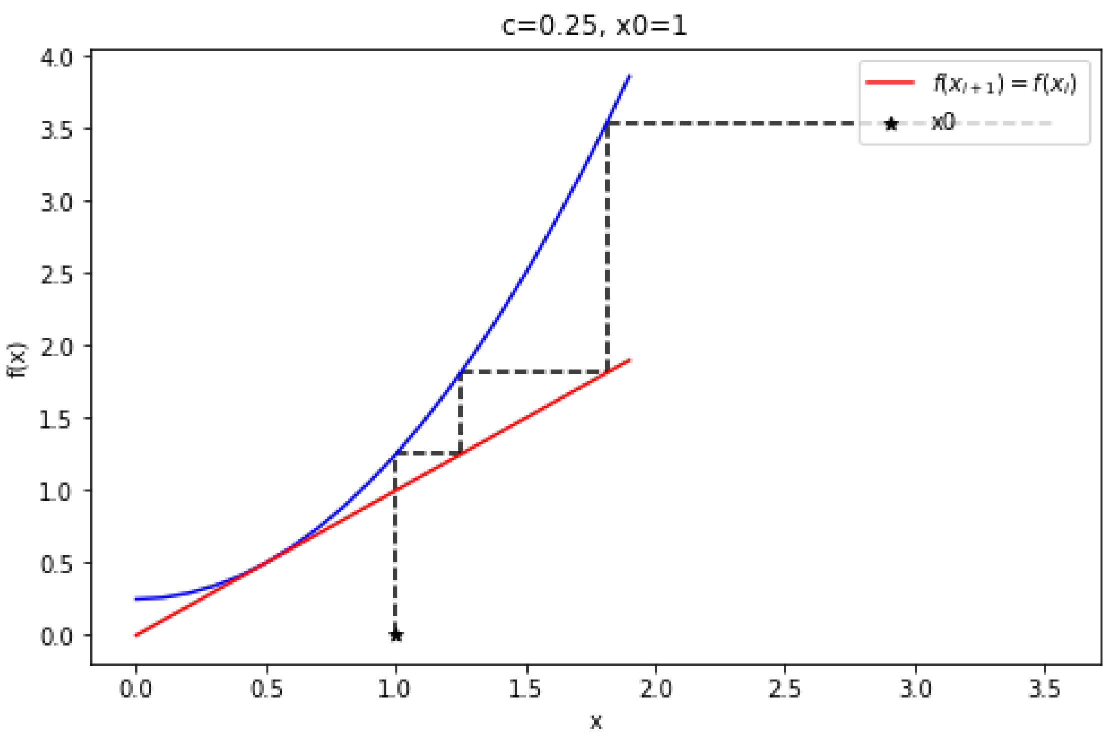

c=0.1, x0=1
$f(x_{i+1}) = f(x_i)$
x0
f(x)
x

8 c=0.25, x=[0, 0.5, 1]

[86]:
```
cobweb(c=0.25, x0=0)
cobweb(c=0.25, x0=0.5)
cobweb(c=0.25, x0=1)
```

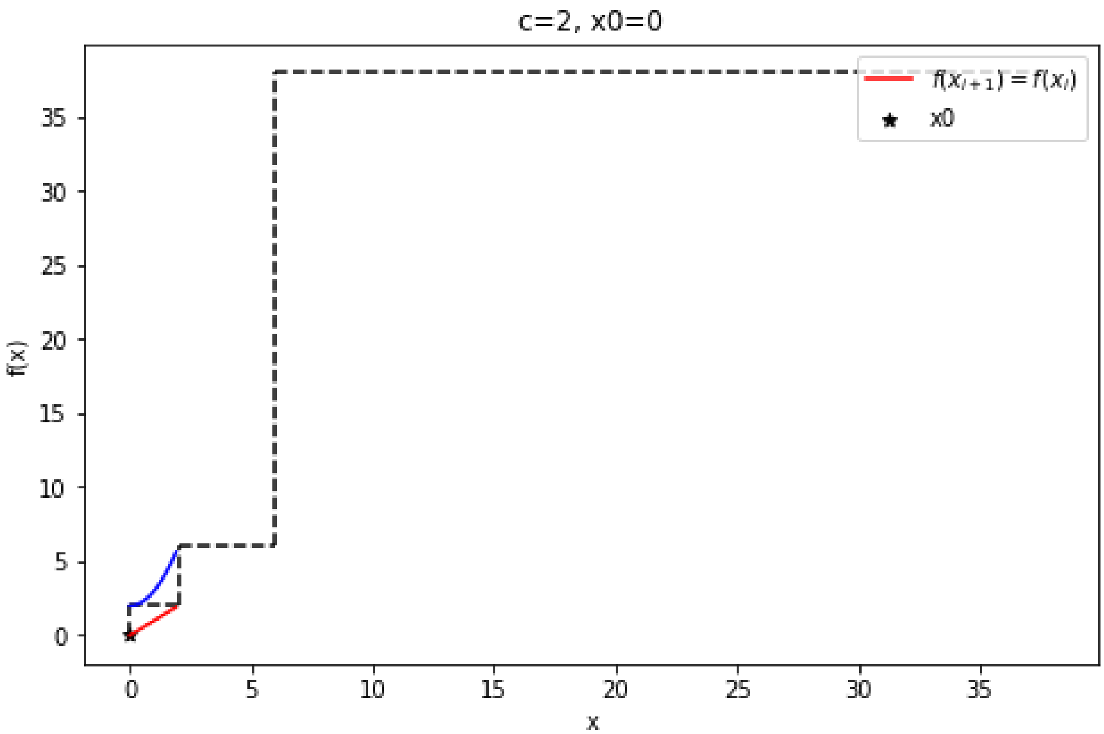


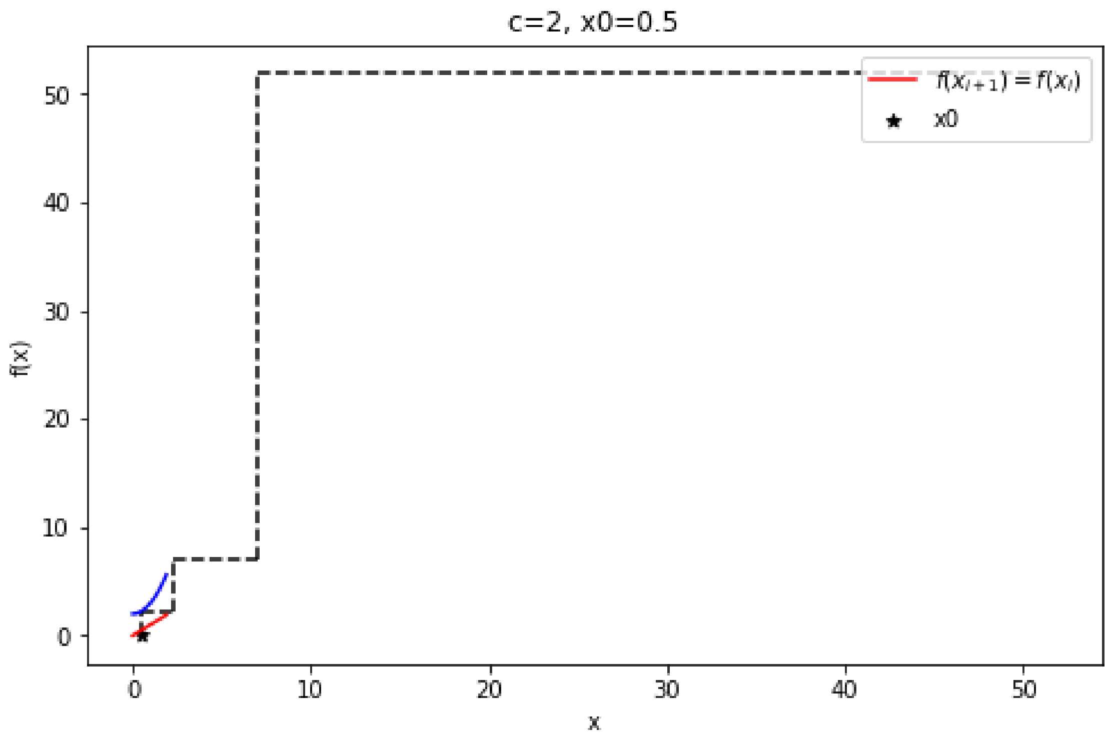

c=0.25, x0=0.5
$f(x_{i+1}) = f(x_i)$
x0
f(x)
x
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

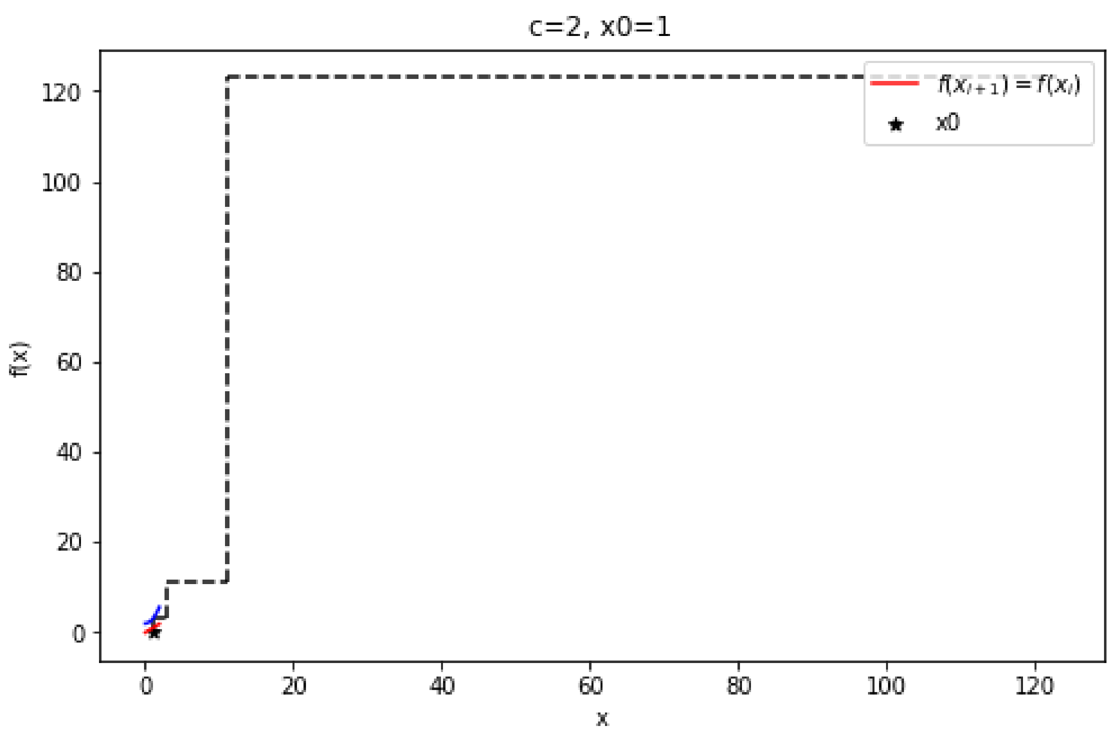

c=0.25, x0=1
$f(x_{i+1}) = f(x_i)$
x0
f(x)
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

9 C>0.25, x=[0, 0.5, 1]

[87]:
```
cobweb(c=2, x0=0)
cobweb(c=2, x0=0.5)
cobweb(c=2, x0=1)
```




c=2, x0=0.5
$f(x_{i+1}) = f(x_i)$
x0
f(x)
x
0
10
20
30
40
50


c=2, x0=1
$f(x_{i+1}) = f(x_i)$
x0
f(x)
x
0
20
40
60
80
100
120

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```