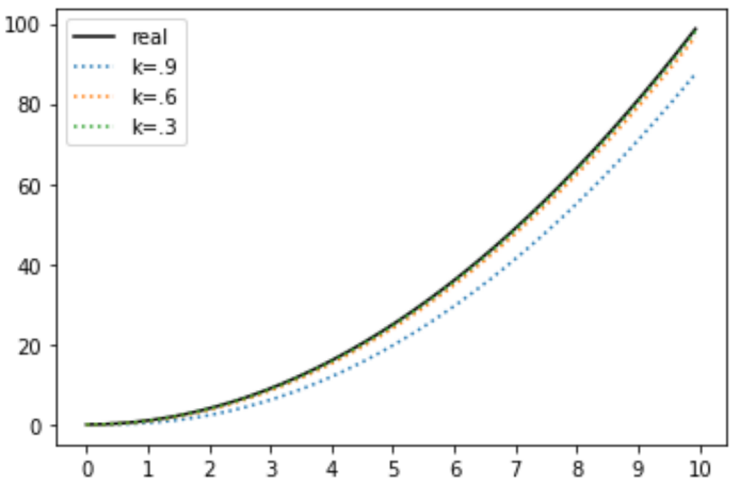
In [1]:
```
# setup
import numpy as np
import matplotlib.pyplot as plt

def media_pesata(array, k):
    shape = array.shape
    res = np.zeros(shape)

    res[0] = array[0]
    for i in range(shape[0]):
        if i == 0:
            continue
        res[i] = k * res[i-1] + (1-k) * array[i]
    return res



fps = 15
t = 10
n_range = fps * t

l1 = np.array([0 for _ in range(n_range)])
l1[0] = 100

l2 = np.asarray([(x/fps)**2 for x in range(n_range)])

l=l2

k = .8

toll = .1
```

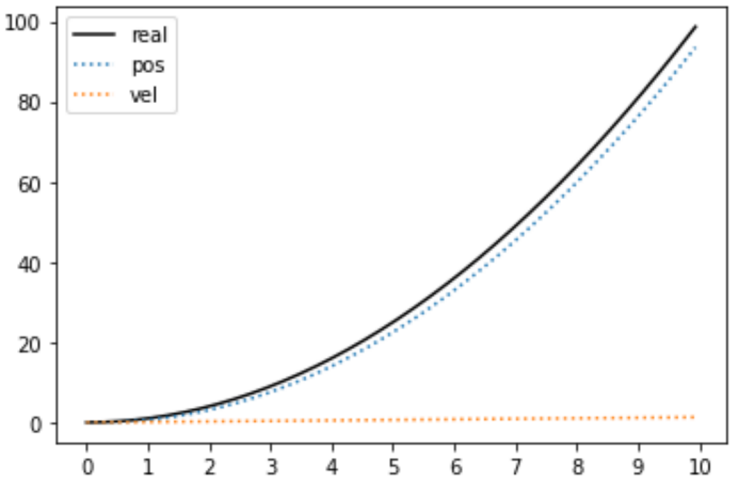
In [2]:
```
# plot some values of k

plt.xticks([x * fps for x in range(t+1)], range(t+1))
plt.plot(l, 'k-', label='real')
plt.plot(media_pesata(l, .9), ':', label='k=.9')
plt.plot(media_pesata(l, .6), ':', label='k=.6')
plt.plot(media_pesata(l, .3), ':', label='k=.3')
plt.legend()
plt.figure()
```

Out[2]:
```
<Figure size 432x288 with 0 Axes>
```



In [3]:
```
# plot the velocity (change from n-1 to n) for the curve

def velocity(array):
    res = np.zeros(array.shape)
    for i in range(array.shape[0]):
        if i == 0:
            continue
        res[i] = array[i] - array[i-1]
    return res

m = media_pesata(l, k)

plt.xticks([x * fps for x in range(t+1)], range(t+1))
plt.plot(l, 'k-', label='real')
plt.plot(m, ':', label='pos')
plt.plot(velocity(m), ':', label='vel')
plt.legend()
plt.figure()
```

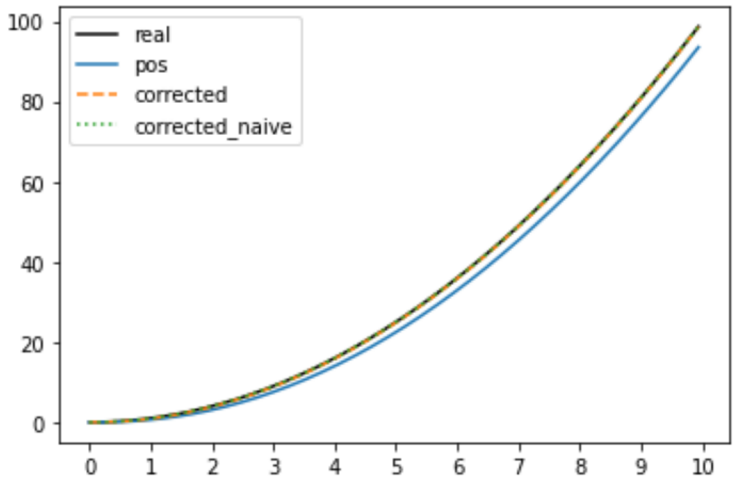
Out[3]:
```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

In [4]:
```
# try to correct the position using the velocity

m = media_pesata(l, k)
noisy = np.random.normal(size=len(l))
m_noisy = media_pesata(l + noisy, k)

corr = 1 / (1 - k) - 1

plt.xticks([x * fps for x in range(t+1)], range(t+1))
plt.plot(l, 'k-', label='real')
plt.plot(m, '-', label='pos')

plt.plot(m + (media_pesata(velocity(m) * corr, k)), '--', label='corrected'

plt.plot(m + (velocity(m) * corr), ':', label='corrected_naive')

plt.legend()
plt.figure()

print(min(m + (media_pesata(velocity(m) * corr, k))))
```

```
0.0
```

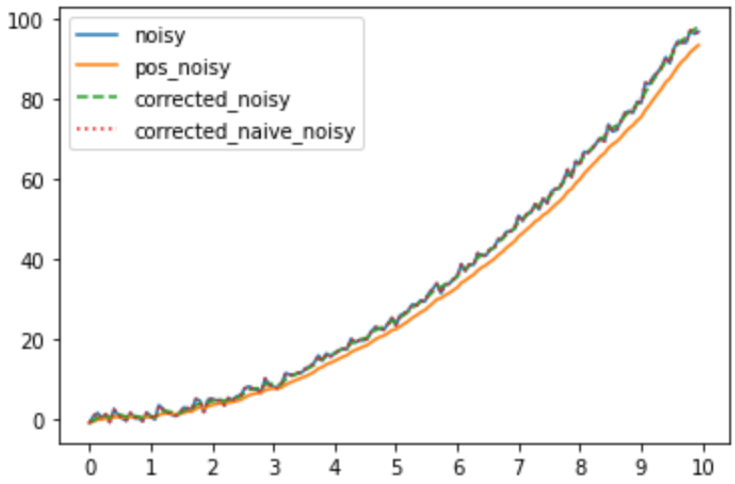


<Figure size 432x288 with 0 Axes>

In [5]:

```
# same as above but on noisy signal

m = media_pesata(l, k)
noisy = np.random.normal(size=len(l))
m_noisy = media_pesata(l + noisy, k)

corr = 1 / (1 - k) - 1

plt.xticks([x * fps for x in range(t+1)], range(t+1))
plt.plot(l + noisy, '-', label='noisy')
plt.plot(m_noisy, '-', label='pos_noisy')

plt.plot(m_noisy + (media_pesata(velocity(m_noisy) * corr, k)), '--', label

plt.plot(m_noisy + (velocity(m_noisy) * corr), ':', label='corrected_naive_

plt.legend()
plt.figure()

print(min(m_noisy + (media_pesata(velocity(m_noisy) * corr, k))))
```

```
-1.0813014461438968
```




```
<Figure size 432x288 with 0 Axes>
```

In [6]:
```
# test the formula to get the factor for the velocity when correcting
n = 10

def foo(k):
    f = True
    corr = 1 / (1-k) - 1
    m = media_pesata(l, k)
    for x, y in zip(l, m + (velocity(m) * corr)):
        if abs(x - y) > 0.01:
            f = False
    return f

print(list(zip([x / n for x in range(1, n)], [1 / (1-k) - 1 for k in [x / r
print(list(map(foo, [x / n for x in range(1, n)])))
```

```
[(0.1, 0.11111111111111116), (0.2, 0.25), (0.3, 0.4285714285714286), (0.4,
0.6666666666666667), (0.5, 1.0), (0.6, 1.5), (0.7, 2.333333333333333), (0.
8, 4.000000000000001), (0.9, 9.000000000000002)]
[True, True, True, True, True, True, True, True, True]
```

In [7]:
```
# check after how much time the position converges without correction

def limit(array, tollerance):
    for i in range(array.shape[0]):
        if abs(array[i]) < tollerance:
            return i
    return -1

ticks = [limit(x, toll) for x in [
    media_pesata(l, .9),
    media_pesata(l, .6),
    media_pesata(l, .3),
]]

print(ticks)

plt.xticks(ticks, [f"{x / fps:.1f}" for x in ticks])

plt.plot(l, 'k-', label='real')
plt.plot(media_pesata(l, .9), ':', label='k=.9')
plt.plot(media_pesata(l, .6), ':', label='k=.6')
plt.plot(media_pesata(l, .3), ':', label='k=.3')
plt.legend()
plt.figure()
```

```
[0, 0, 0]
```

Out[7]:
```
<Figure size 432x288 with 0 Axes>
```

In [8]:

```
# plot convergence time for some values of k

n = 10
values = [x / n for x in range(n)]

res = np.asarray([limit(media_pesata(l, v), toll) / fps for v in values])

plt.xticks(range(n), values)
plt.plot(res, '.-')
plt.figure

print(res[-1])
```

```
0.0
```