In [9]:

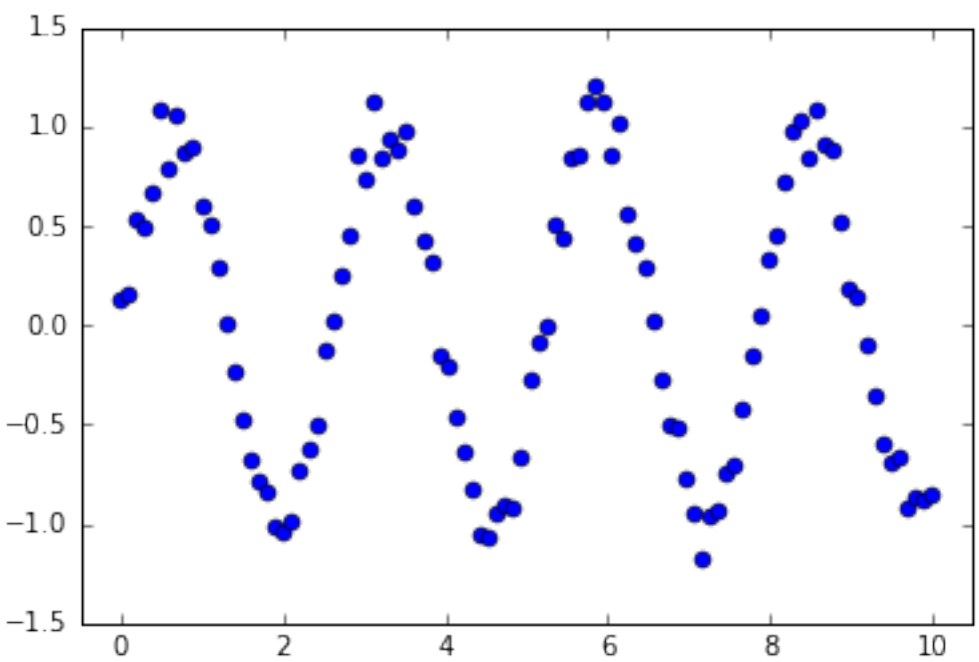
```
%matplotlib inline

import matplotlib
import numpy as np
import matplotlib.pyplot as pt
```

In [10]:

```
k = 2.4  #2pi/k is the frequency of sine wave
n = 100
x = np.linspace(0,10,n)
wave = np.sin(k*x)  #create pure sine function or wave
noise = np.random.normal(scale = 1.0/10, size = len(x)) #normally distributed ra
ndom numbers
y = wave + noise #add noise to curve
```

In [11]:

```
#plot the curve
pt.plot(x,y,'o')
pt.xlim([-.5,10.5])
#pt.plot(x,np.sin(x))
#pt.plot(x,np.sin(2*x))
#pt.plot(x,np.sin(3*x))
pt.show()

# What are some properties of the function that give clues to what function it m
ay be?
```

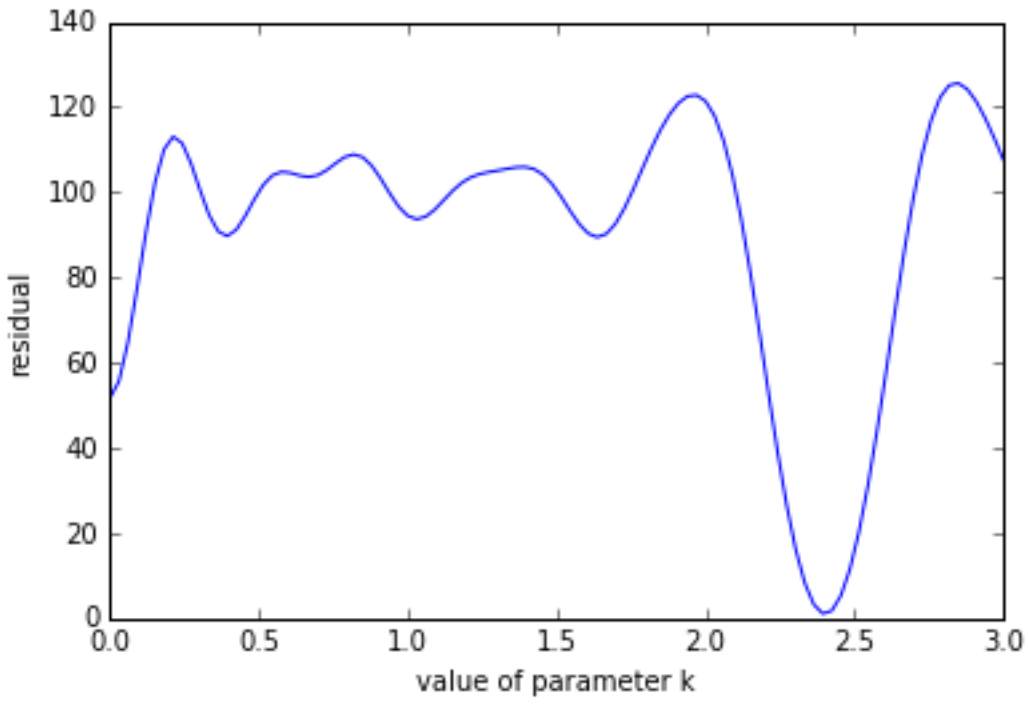


In [23]:

```
# define residual of data and sine function as function of parameter k (sum of s
quares)
def r(k):
    return sum((y - np.sin(k*x))**2)

# define derivative of residual with respect to k.  (k is the unknown variable)
def rprime(k):
    return -2*sum(x*np.cos(k*x)*(y-np.sin(k*x)))

# define 2nd derivative with respect to k
def r2prime(k):
    # this gets really nasty so I'm going to define it in steps
    s1 = (y - np.sin(k*x))*np.sin(k*x) + (np.cos(k*x))**2
    s2 = s1*x**2
    return 2*sum(s2)
```

In [24]:

```
#Let's look at a plot of the residual
k = np.linspace(0,3,n)#it looks like the true k falls in this range

res = np.zeros(len(k))
for i in range(len(k)):
    res[i] = r(k[i])

pt.plot(k,res)
pt.xlabel("value of parameter k")
pt.ylabel("residual")
pt.show()
```



In [27]:

```
# there are a lot of local minima, so we need to choose our initial guess carefu
lly
# should be between 2 and 3 by the "eyeball norm"

xk = 2.5 # initial guess - looks pretty close
tol = 1e-15 #tolerance to decide when to terminate

for i in range(100): #cap the number of iterations at 100
    h = -rprime(xk)/r2prime(xk)# define step size
    xk += h
    if (abs(h) < tol): # if step size is small, this is our last iteration
        print(i)
        break

print(xk)
print(r(xk))
print(rprime(xk))
```

```
4
2.40153218573
0.984708022249
3.65818486614e-14
```

In [ ]: