KÖK BULMA

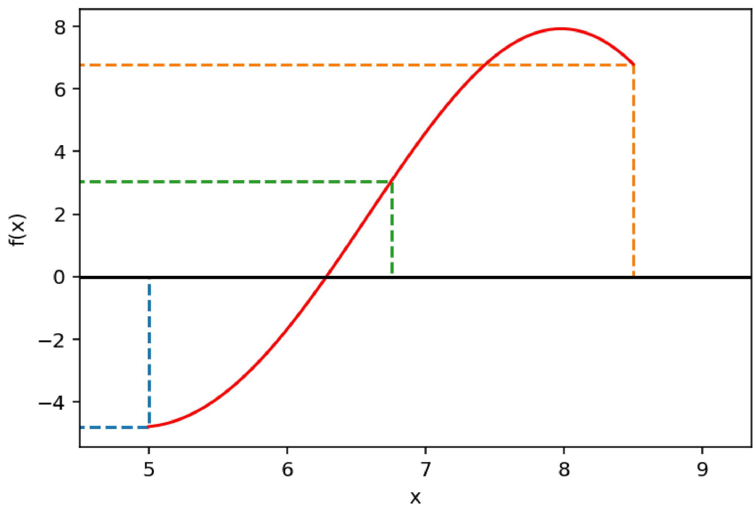
[a,b]∈x aralığında tanımlı bir f(x) fonksiyonunu sıfır yapan x değerini bulma işlemi;
o fonksiyonun köklerini bulmak olarak adlandırılır.

Diğer bir ifadeyle $f(x) = 0$ eşitliğini sağlayan x değerleri aranılan köklerdir.

YARILAMA YÖNTEMİ (INTERVAL / BISECTION METHOD)

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

isaretle = lambda x, y: plt.plot([0,x,x],[y, y,0], '--')

a = 5.0
b = 8.5
xo = (a+b)/2.

x = np.linspace(a, b, 100)
f = lambda x: x*np.sin(x)

isaretle(a, f(a))
isaretle(b, f(b))
isaretle(xo, f(xo))

plt.plot(x,f(x), color="red")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.axhline(color='black')
plt.xlim(a*.9,b*1.1)
plt.show()
```

Out[1]:



In [2]:
```
a0 = 5.0
b0 = 8.5
tol = 1e-6

a = a0
b = b0
dkok = abs(b-a)

f = lambda x: x*np.sin(x)


while dkok>tol:
    xo = (a+b)/2.
    if f(b)*f(xo)>0:
        b = xo
    else:
        a=xo
    dkok = abs(b-a)

print 'aranan kök = %s'%xo
```

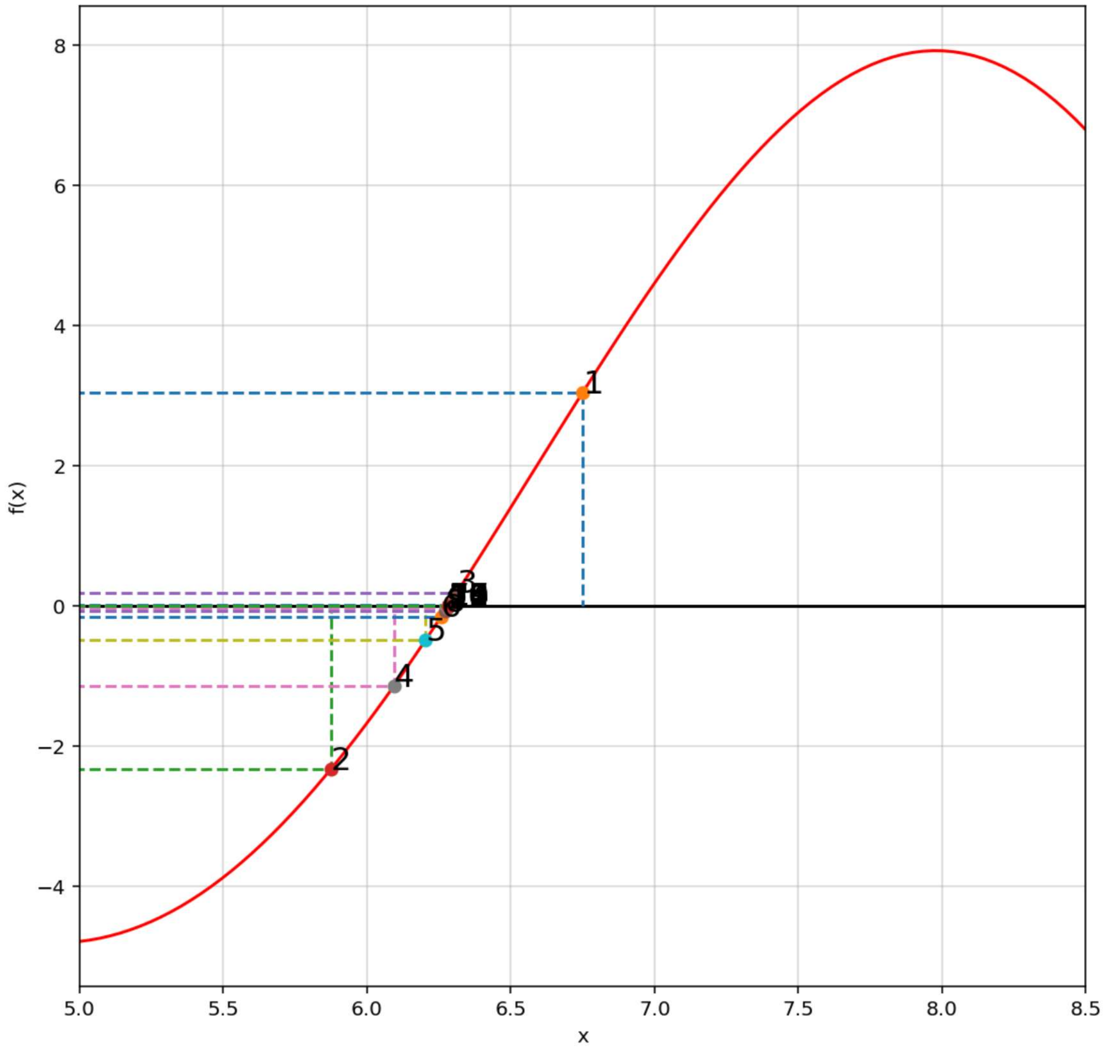
```
aranan kök = 6.28318607807
```

In [3]:

```
a0 = 5.0
b0 = 8.5
tol = 1e-6

a = a0
b = b0
dkok = abs(b-a)
x = np.linspace(a, b, 100)
f = lambda x: x*np.sin(x)

plt.figure(figsize=(9,9))
plt.plot(x,f(x), color="red")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(alpha=.5)
plt.axhline(color='black')
plt.xlim(a0, b0)

it = 0
while dkok>tol:
    xo = (a+b)/2.
    if f(b)*f(xo)>0:
        b = xo
    else:
        a=xo
    # a ve b bir birine ne kadar yakın
    dkok = abs(b-a)
    # döngü sayısı
    it += 1
    fxo = f(xo)
    isaretle(xo, fxo)
    plt.plot(xo, fxo,'o')
    plt.text(xo,fxo, str(it), fontsize=16)

print "%s denemede aşağıdaki kök bulundu"%it
print 'aranan kök = %s'%xo
print 'f(%s)=%s'%(xo, f(xo))
plt.show()
```

```
22 denemede aşağıdaki kök bulundu
aranan kök = 6.28318607807
f(6.28318607807)=4.843657930862331e-06
```

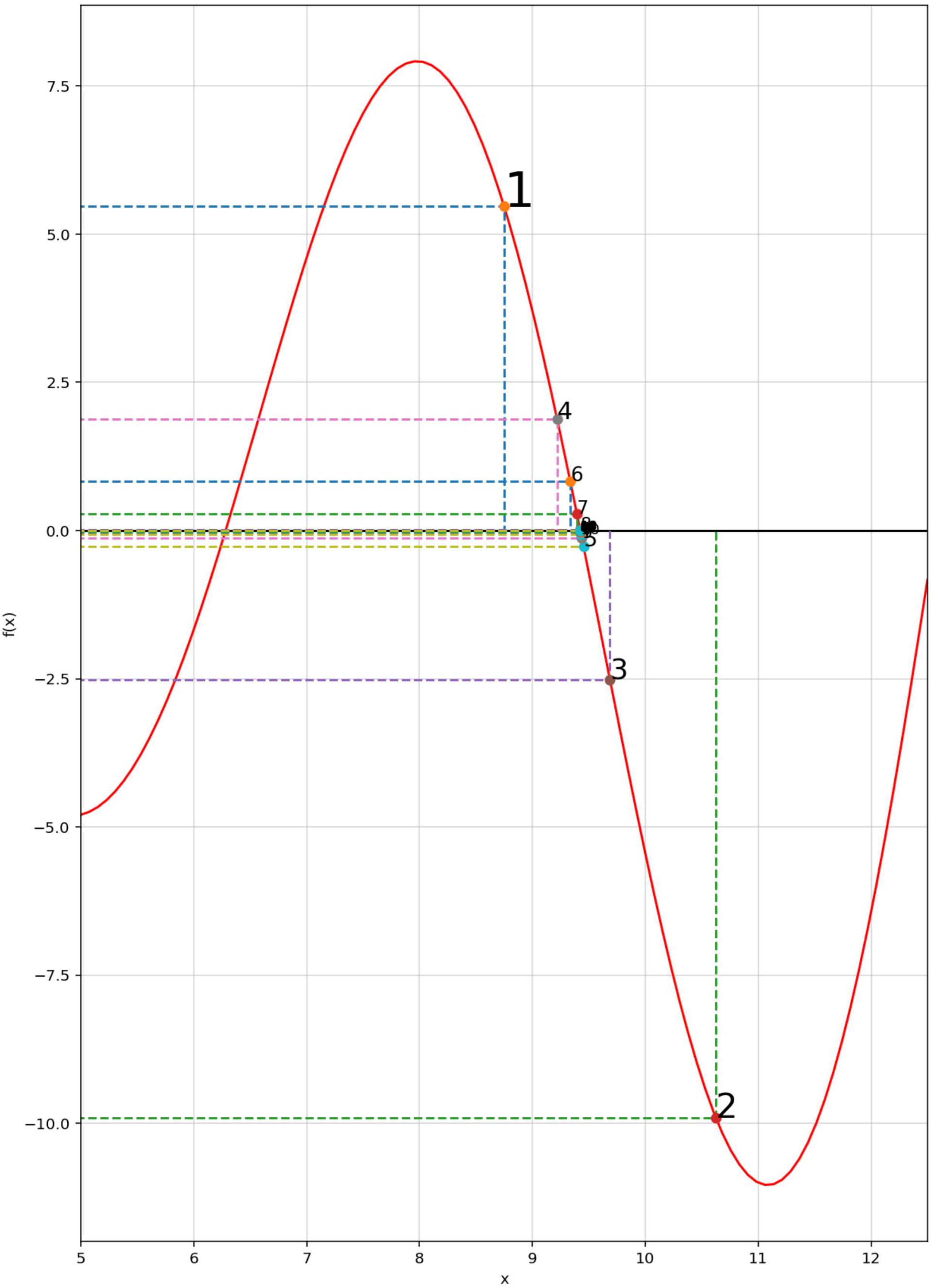
Out[3]:



In [4]:
```
a0 = 5.0
b0 = 12.5
tol = 1e-16

a = a0
b = b0
dkok = abs(b-a)
x = np.linspace(a, b, 100)
f = lambda x: x*np.sin(x)

plt.figure(figsize=(10,15))
plt.plot(x,f(x), color="red")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(alpha=.5)
plt.axhline(color='black')
plt.xlim(a0, b0)

it = 0
itmax = 300
while dkok>tol and it<itmax:
    xo = (a+b)/2.
    if f(b)*f(xo)>0:
        b = xo
    else:
        a=xo
    # a ve b bir birine ne kadar yakın
    dkok = abs(b-a)
    # döngü sayısı
    it += 1
    fxo = f(xo)
    isaretle(xo, fxo)
    plt.plot(xo, fxo,'o')
    plt.text(xo,fxo, str(it), fontsize=32/it**.5)

print "%s denemede aşağıdaki kök bulundu"%it
print 'aranan kök = %s'%xo
print 'f(%s)=%s'%(xo, f(xo))
plt.show()
```

```
300 denemede aşağıdaki kök bulundu
aranan kök = 9.42477796077
f(9.42477796077)=3.4626072486992214e-15
```

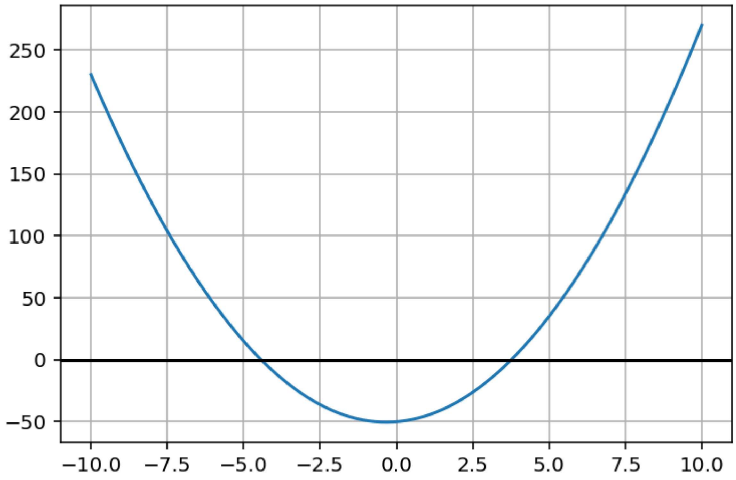
Out[4]:



2. dereceden polinomun kökü

In [5]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

a1 = 3.
b1 = 2.
c1 = -50.
f = lambda x: a1*x**2 + b1*x + c1

x = np.linspace(-10, 10, 100)
y = f(x)

plt.plot(x, y)
plt.grid()
plt.axhline(color='black')
plt.show()
```

Out[5]:



Yarılama yöntemiyle sayısal çözüm

In [6]:
```python
a0 = -10.
b0 = 0.
tol = 1e-6

a = a0
b = b0
dkok = abs(b-a)


while dkok>tol:
    xo = (a+b)/2.
    if f(b)*f(xo)>0:
        b = xo
    else:
        a=xo
    dkok = abs(b-a)

print 'aranan kök = %s'%xo
print 'f(%s)=%s'%(xo, f(xo))
```

```
aranan kök = -4.42940175533
f(-4.42940175533)=-3.78022612324e-06
```

Analitik çözüm

In [7]:
```
delta = (b1**2. - 4*a1*c1)**.5
xt1 = (-b1 + delta)/(2.*a1)
xt2 = -(b1 + delta)/(2.*a1)
print "%s %s"%(xt1, xt2)
```

```
3.76273524248 -4.42940190915
```

SymPy kütüphanesiyle analitik çözüm

In [8]:
```
import sympy as sm
a,b,c,x = sm.symbols('a,b,c,x')

denklem  = a*x**2 + b*x + c

cozumler = sm.solve(denklem, x)

print denklem
print cozumler
```

```
a*x**2 + b*x + c
[(-b + sqrt(-4*a*c + b**2))/(2*a), -(b + sqrt(-4*a*c + b**2))/(2*a)]
```

In [9]:
```
print "*"*64
print "reel sayılarla"
print "birinci kök = ", cozumler[0].subs({a:a1, b:b1, c:c1})
print "ikinci kök = ", cozumler[1].subs({a:a1, b:b1, c:c1})
print "*"*64
print "tam (rasyonel) sayılarla"
print "birinci kök = ", cozumler[0].subs({a:3, b:2, c:-50})
print "birinci kök = ", cozumler[1].subs({a:3, b:2, c:-50})
print "*"*64
```

```
****************************************************************
reel sayılarla
birinci kök =  3.76273524248150
ikinci kök =  -4.42940190914817
****************************************************************
tam (rasyonel) sayılarla
birinci kök =  -1/3 + sqrt(151)/3
birinci kök =  -sqrt(151)/3 - 1/3
****************************************************************
```

KİRİŞ YÖNTEMİ (SECANT METHOD)

In [10]:
```
def kiris(x0,x1, tol, itermax=50):
    dx = x1-x0
    iter = 1
    while abs(dx)>tol:
        x2 = x1 - (x1-x0)*f(x1)/(f(x1)-f(x0))
        x0, x1 = x1, x2
        dx = x1 - x0
        if iter>itermax:
            print "%s denemede kök bulunamadı"%itermax
            break
        iter += 1
    return x2
```

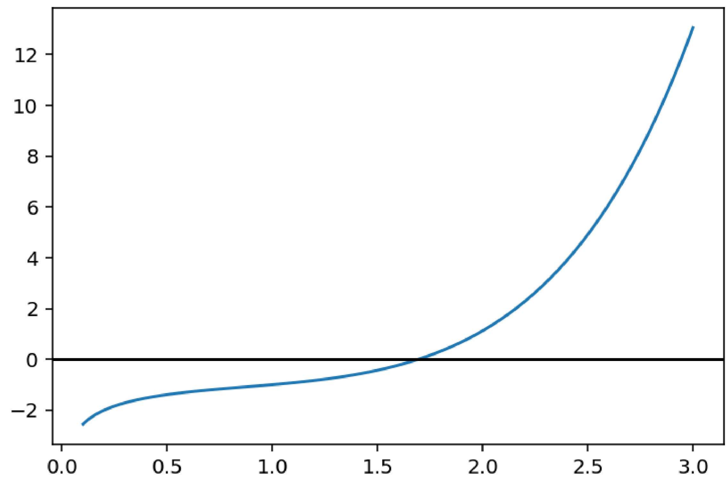
In [11]:
```
def kiris_kokListesi(x0,x1, tol, itermax=50):
    dx = x1-x0
    xkok = [x0, x1]
    fxkok = [f(x0), f(x1)]
    iter = 1
    while abs(dx)>tol:
        x2 = x1 - (x1-x0)*f(x1)/(f(x1)-f(x0))
        xkok += [x2]
        fxkok += [f(x2)]
        x0, x1 = x1, x2
        dx = x1 - x0
        if iter>itermax:
            print "%s denemede kök bulunamadı"%itermax
            break
        iter += 1
    return x2, xkok, fxkok
```

In [12]:
```
from math import *

f = lambda x: exp(x)*log(x) - x**2

a = 3.0
b = 5.0
tol = 1.0e-8
itermax = 100
x = kiris(a,b,tol, itermax=itermax)
print x
```

```
1.6946009205
```

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

#f = lambda x: np.exp(x)*np.log(x) - x**2
def f(x):
    return np.exp(x)*np.log(x) - x**2

a = 0.1
b = 3.0
x = np.linspace(a, b, 100)
y = f(x)

plt.plot(x, y)
plt.axhline(color='black')
plt.show()
```

Out[13]:

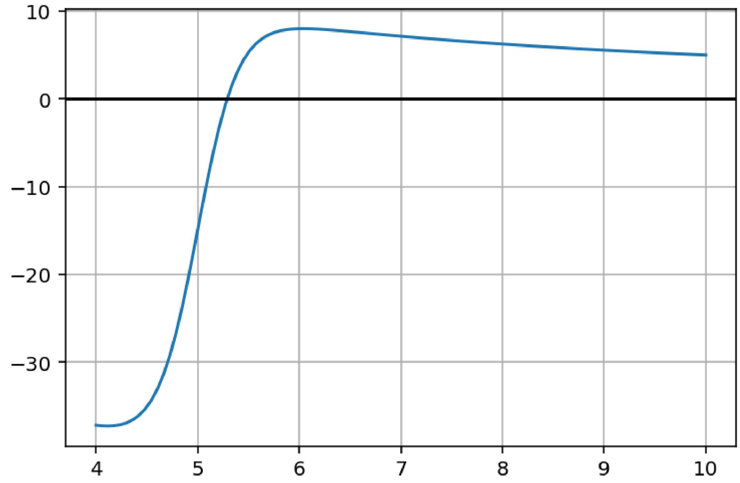


Kiriş yönteminin ıraksar durumu

In [14]:

```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

#f = lambda x: np.exp(x)*np.log(x) - x**2
def f(x):
    return -50/(1+np.exp((x-5)/.2)) + 50./x

a = 4.0
b = 10.0
x = np.linspace(a, b, 100)
y = f(x)

plt.plot(x, y)
plt.axhline(color='black')
plt.grid()
plt.show()
```

Out[14]:



In [15]:

```python
def f(x):
    return -50/(1+np.exp((x-5)/.2)) + 50./x

a = 4.0
b = 10.0

tol = 1.0e-8
itermax = 100
x = kiris(a,b,tol, itermax=itermax)
print x
```

```
100 denemede kök bulunamadı
8.865941311725278e+21

/ext/sage/sage-8.9_1804/local/lib/python2.7/site-packages/ipykernel/__main__.py:2: R
untimeWarning: overflow encountered in exp
  from ipykernel import kernelapp as app
```

In [16]:
```
def f(x):
    return -50/(1+np.exp((x-5)/.2)) + 50./x

a = 4.0
b = 5.0 #10.0

tol = 1.0e-8
itermax = 100
x = kiris(a,b,tol, itermax=itermax)
print x
```

```
5.291318820618074
```

Kiriş yönteminde ıraksayan durumun, Yarılama yöntemiyle, çözümünün bulunması

In [17]:
```
def f(x):
    return -50/(1+np.exp((x-5)/.2)) + 50./x

a0 = 4.0
b0 = 10.0
tol = 1e-6

a = a0
b = b0
dkok = abs(b-a)
while dkok>tol:
    xo = (a+b)/2.
    if f(b)*f(xo)>0:
        b = xo
    else:
        a=xo
    dkok = abs(b-a)

print 'aranan kök = %s'%xo
```

```
aranan kök = 5.29131913185
```

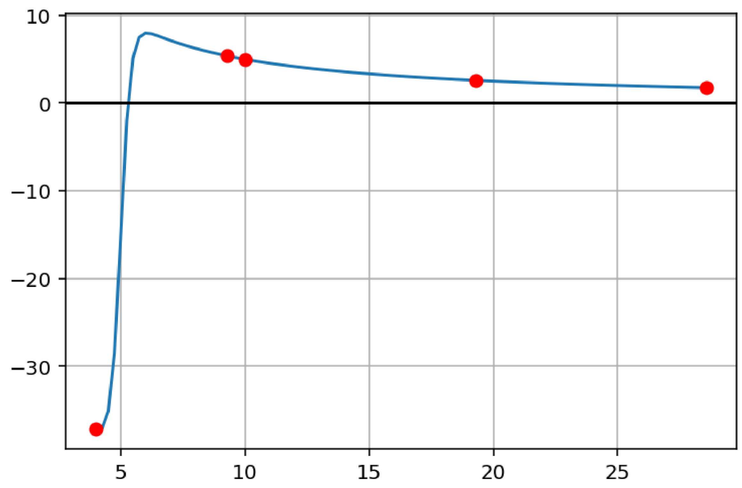
Kiriş yöntemindeki ıraksar durumun grafiği

In [18]:
```
def f(x):
    return -50/(1+np.exp((x-5)/.2)) + 50./x

a = 4.0
b = 10.0

tol = 1.0e-8
itermax = 100
xkok, xkok_liste, fxkok_liste = kiris_kokListesi(a,b,tol, itermax=itermax)
print xkok
```

```
100 denemede kök bulunamadı
8.865941311725278e+21

/ext/sage/sage-8.9_1804/local/lib/python2.7/site-packages/ipykernel/__main__.py:2: R
untimeWarning: overflow encountered in exp
  from ipykernel import kernelapp as app
```

In [19]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

#f = lambda x: np.exp(x)*np.log(x) - x**2
def f(x):
    return -50/(1+np.exp((x-5)/.2)) + 50./x

a = 4.0
b = xkok_liste[4]#10.0
x = np.linspace(a, b, 100)
y = f(x)

plt.plot(x, y)
plt.plot(xkok_liste[:5], fxkok_liste[:5], "ro")
plt.axhline(color='black')
plt.grid()
plt.show()
```

Out[19]:



NEWTON-RAPHSON YÖNTEMİ

In [20]:
```
# ilk yazdığımız newton_raphson kök bulma fonksiyonu
def newton_eski(x1, tol):
    x2 = 3.*tol
    dkok = abs(x2-x1)
    iter=1
    while dkok>tol:
        x1 = x2
        x2 = x1 - f(x1)/f1(x1)
        dkok = abs(x2-x1)
        iter += 1
        if iter>5:
            print ("kök bulunamadı")
            break
    return x2
```

In [21]:
```python
# yeniden düzenlenmiş olan
def newton(x2, tol=1.0e-8, itermax=5):
    iter=1
    while iter<itermax:
        x1 = x2
        x2 = x1 - f(x1)/f1(x1)
        dkok = abs(x2-x1)
        iter += 1
        if dkok<=tol: break
    if dkok>tol:
        print "%s denemede, aranan köke %s kadar yaklaşılabilmiştir."%(iter, dkok)
        print "İstenilen hassasiyet için itermax veya x başlangıç değeri değiştirilme
lidir."
    return x2
```

In [22]:
```python
from math import *

f = lambda x: cos(x) - x
f1 = lambda x: -sin(x) - 1.

newton(1.,1.0e-8)
```

Out[22]: 0.7390851332151607

In [23]:
```python
from math import *

f = lambda x: cos(x) - x
f1 = lambda x: (f(x+h) - f(x))/h

h = 0.01
newton(1.,1.0e-8)
```

```
5 denemede, aranan köke 1.2047086706e-07 kadar yaklaşılabilmiştir.
İstenilen hassasiyet için itermax veya x başlangıç değeri değiştirilmelidir.
```

Out[23]: 0.7390851334803596

In [24]:
```python
import sympy as sm
x = sm.symbols('x')


f = lambda x: sm.cos(x) - x
f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)

newton(1.,1.0e-8)
```

Out[24]: 0.739085133215161

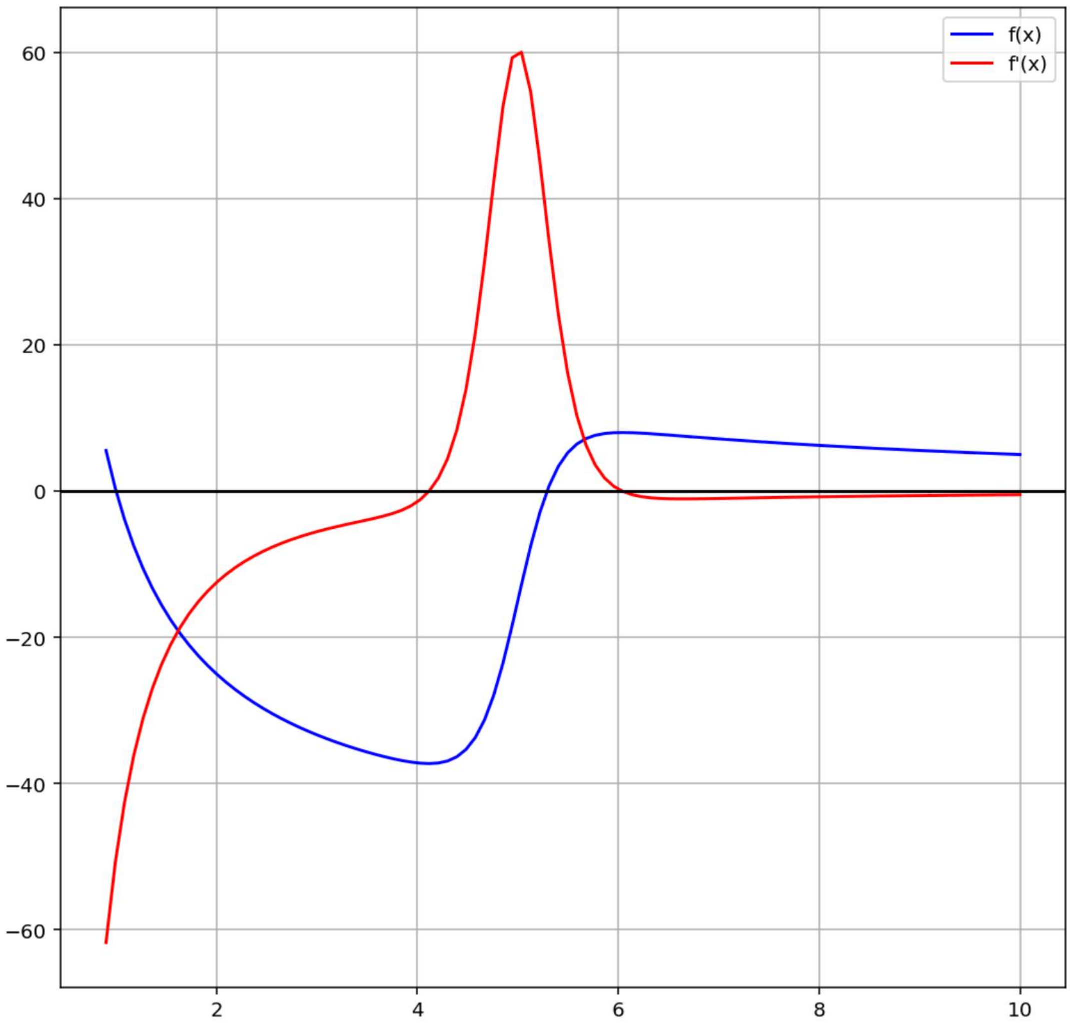
In [25]:
```python
import sympy as sm
x = sm.symbols('x')


f = lambda x:-50/(1+sm.exp((x-5)/.2)) + 50./x

f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)

newton(1.2,1.0e-8)
```

```
5 denemede, aranan köke 2.55999460918588e-6 kadar yaklaşılabilmiştir.
İstenilen hassasiyet için itermax veya x başlangıç değeri değiştirilmelidir.
```

Out[25]: 1.0000000020546

In [26]:
```
import numpy as np
import sympy as sm
import matplotlib.pyplot as plt
%matplotlib inline

x = sm.symbols('x')

f = lambda x:-50/(1+sm.exp((x-5)/.2)) + 50./x
f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)

fn = sm.lambdify(x, f(x), "numpy")
f1n = sm.lambdify(x, f1(x), "numpy")

a = 0.9
b = 10.0
x = np.linspace(a, b, 100)
y = fn(x)
y1 = f1n(x)

plt.figure(figsize=(9,9))
plt.plot(x, y, color="blue", label="f(x)")
plt.plot(x, y1, color="red", label="f'(x)")
plt.axhline(color='black')
plt.grid()
plt.legend()
plt.show()
```

Out[26]:



Newton-Raphson yönteminde seçilen başlangıç değeri önemlidir.

Yukarıdaki grafikten anlaşılacağı üzere;

- 2 ile 4 arasında seçilen başlangıç değerleri fonksiyonun her iki köküne de yaklaşmayacaktır.
- 6 ve üzerinde ise yine kök her iki köke de yakınsamayacaktır.

In [27]:
```
import sympy as sm
x = sm.symbols('x')


f = lambda x:-50/(1+sm.exp((x-5)/.2)) + 50./x

f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)

print newton(2.0,1.0e-8,5)
print
print newton(4.0,1.0e-8,5)
print
print newton(6.0,1.0e-8,10)
```

```
5 denemede, aranan köke 4.405284334936840e-5 kadar yaklaşılabilmiştir.
İstenilen hassasiyet için itermax veya x başlangıç değeri değiştirilmelidir.
-8.81037462166934e-5

5 denemede, aranan köke 63469827346.8788 kadar yaklaşılabilmiştir.
İstenilen hassasiyet için itermax veya x başlangıç değeri değiştirilmelidir.
-63470079278.5668

10 denemede, aranan köke 4.46881970862620e+354 kadar yaklaşılabilmiştir.
İstenilen hassasiyet için itermax veya x başlangıç değeri değiştirilmelidir.
-4.46881970862620e+354
```

UYGULAMALAR

Karacisim Işıması

$\mathbf{u(\lambda, T) = \frac{8\pi hc}{\lambda^5} \frac{1}{e^{hc/k_B T\lambda} - 1}}$ denklemi için $\mathbf{x = hc/k_B T \lambda}$ değişken dönüşümü yapılırsa;

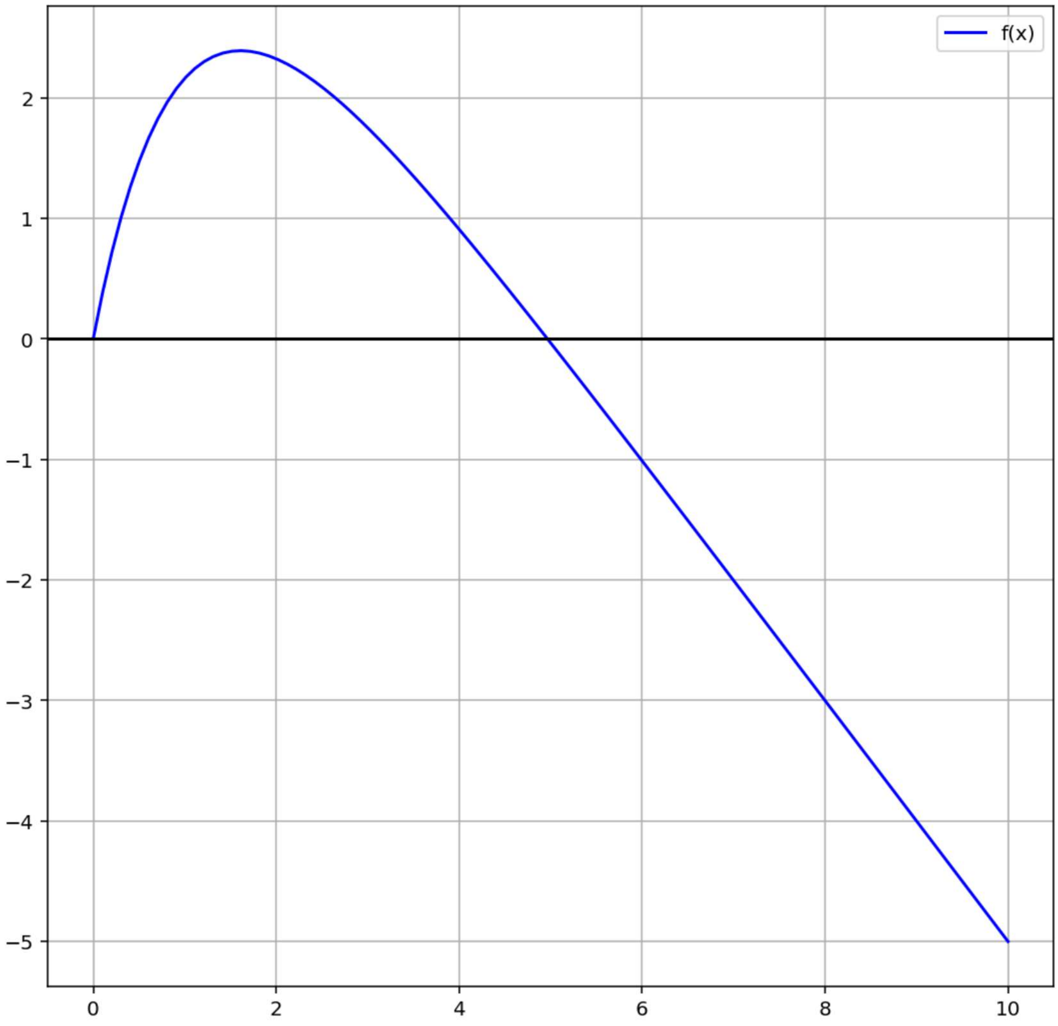
$\mathbf{u(x) = A \frac{x^5}{e^x - 1}}$ elde edilir. Burada A gerekli dönüşüm yapıldıktan sonra geriye kalan sabitleri temsil etmektedir.

Türevinin sıfır olduğu x değeri bulunarak λ_{max} bulunur.

In [28]:

```
import numpy as np
import sympy as sm
import matplotlib.pyplot as plt
%matplotlib inline

x = sm.symbols('x')
f = lambda x: (5. - x) - 5.*sm.exp(-x)
fn = sm.lambdify(x, f(x), "numpy")

a = 0.0
b = 10.0
x = np.linspace(a, b, 100)
y = fn(x)

plt.figure(figsize=(9,9))
plt.plot(x, y, color="blue", label="f(x)")
plt.axhline(color='black')
plt.grid()
plt.legend()
plt.show()
```

Out[28]:



In [29]:
```
import sympy as sm
x = sm.symbols('x')


f = lambda x: (5. - x) - 5.*sm.exp(-x)

f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)

print "birinci kök %s"%newton(1.0,1.0e-8,15)
print
print "ikinci kök %s"%newton(4.0,1.0e-8,15)
```

```
birinci kök 9.41605232733861e-17

ikinci kök 4.96511423174428
```

ALIŞTIRMALAR

3.4 van der Waals denklemi:

$$\mathbf{(P + \frac{a}{v^2})(v - b) = RT}$$

R = 0.08207 lt atm/mol
CO (Karbon monoksit) gazı için;
a = 3.592
b = 0.04267
T = 320 K
P = 2.2 atm

ise ilk formülden ve ideal gaz denkleminden elde edilen
hacim değerlerini ($v = V/n$) karşılaştırınız.

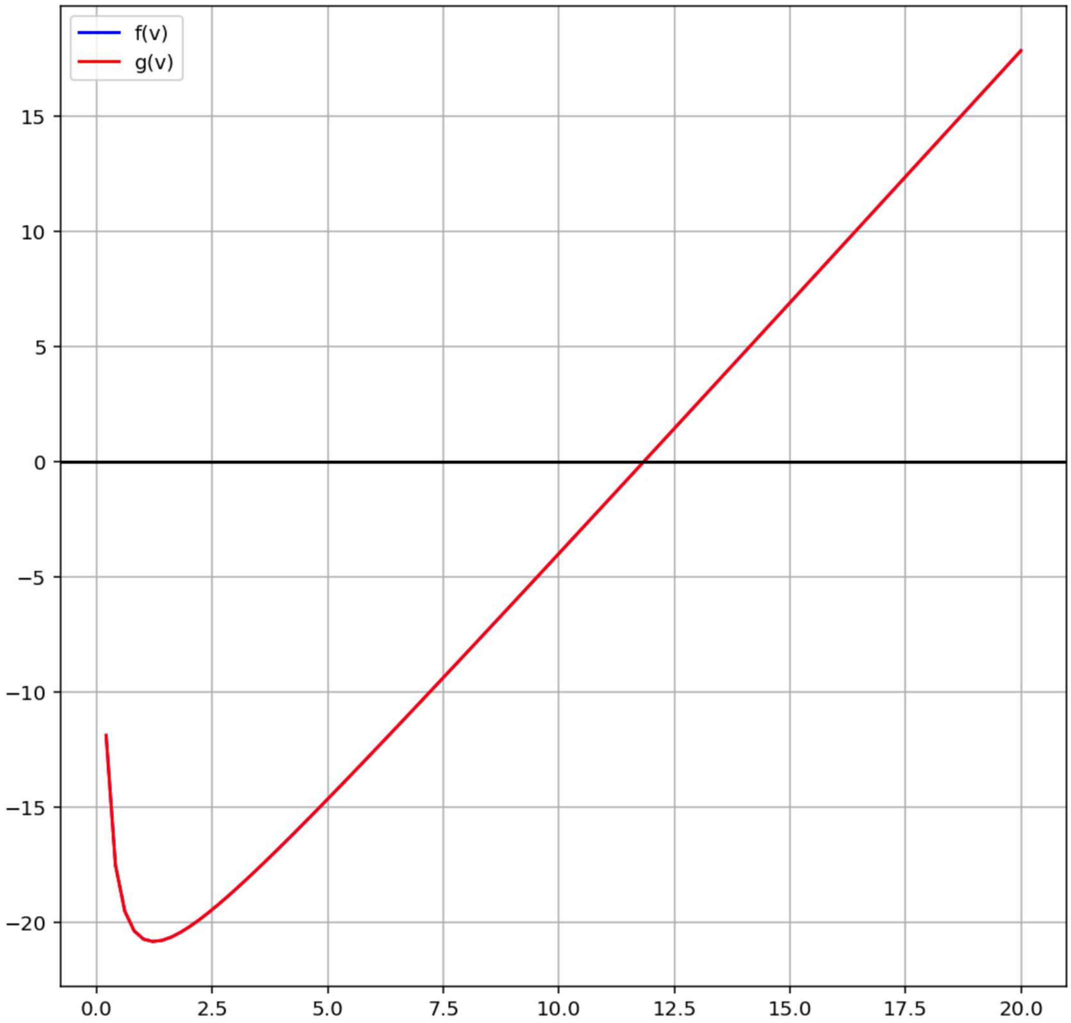
In [30]:
```
import sympy as sm
v = sm.symbols('v')

R = 0.08207 #lt atm/mol
#CO (Karbon monoksit) gazı için;
a = 3.592
b = 0.04267
T = 320. #K
P = 2.2 #atm

# van der Waals denklemi
f = lambda v: (P+a/v**2)*(v-b) - R*T
f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)
print "van der Waals denklemi %s"%newton(4.0,1.0e-8,15)

#ideal gaz denklemi
f = lambda v: P*v - R*T
f1 = lambda x0: sm.diff(f(x),x).subs(x,x0)
print "ideal gaz denklemi %s"%newton(4.0,1.0e-8,15)
```

```
van der Waals denklemi 11.8427540937255
ideal gaz denklemi 11.9374545454545
```

In [31]:
```
import numpy as np
import sympy as sm
import matplotlib.pyplot as plt
%matplotlib inline

v = sm.symbols('v')
R = 0.08207 #lt atm/mol
#CO (Karbon monoksit) gazı için;
a = 3.592
b = 0.04267
T = 320. #K
P = 2.2 #atm

f = lambda v: (P+a/v**2)*(v-b) - R*T
fn = sm.lambdify(v, f(v), "numpy")

g = lambda v: P*v - R*T
gn = sm.lambdify(v, f(v), "numpy")

a = 0.0
b = 20.0
v = np.linspace(a, b, 100)
yf = fn(v)
yg = gn(v)

plt.figure(figsize=(9,9))
plt.plot(v, yf, color="blue", label="f(v)")
plt.plot(v, yg, color="red", label="g(v)")
plt.axhline(color='black')
plt.grid()
plt.legend()
plt.show()
```

```
<string>:2: RuntimeWarning: divide by zero encountered in true_divide
<string>:2: RuntimeWarning: divide by zero encountered in true_divide
```

Out[31]:



In [ ]: