In [1]:

```
from sympy import *
from IPython.core.interactiveshell import InteractiveShell #print all elements, not onl
y last one
InteractiveShell.ast_node_interactivity = "all"
init_printing(use_unicode=True)
%matplotlib inline
```

In [4]:

```
s, q = var('s,q',real=True);
b, h, B, h0 = symbols('b, h, bbar, h0')
b0 = symbols ('b0', cls = Function)(q,s)
f = symbols ('f', cls = Function)(b) # density function of b
C = symbols ('C', cls = Function)(q) # cost of apprehension and conviction
g = symbols ('g', cls = Function)(s) # constraint
b0 = q*s
funct = (b-h)*f # welfare function
W = integrate(funct, (b, (b0, B))) - C
g = s
W
```

Out[4]:

$$-C(q) + \int\limits_{qs}^{\bar{b}} (b - h)\, f(b)\, db$$

In [5]:

```
lam = symbols('lambda', real = True)
L = W - lam*g # the Lagrangian equation
L
```

Out[5]:

$$-\lambda s - C(q) + \int\limits_{qs}^{\bar{b}} (b - h)\, f(b)\, db$$

In [6]:

```
gradL = [diff(L,k) for k in [s,q]] # gradient of Lagrangian w.r.t. (s,q)
KKT_eqs = gradL + [g]
KKT_eqs
```

Out[6]:

$$\left[-\lambda - q\left(-h + qs\right) f(qs),\ \ - s\left(-h + qs\right) f(qs) - \frac{d}{dq}C(q),\ s\right]$$

In [7]:

```
stationary_points = solve(KKT_eqs, [s, q, lam], dict=True) # solve the KKT equations
stationary_points
```

Out[7]:

$[]$

In [ ]:

```

```