In [1]:
```
import numpy as np
```

In [12]:
```
ALPHA = 0.01
BETA = 0.5
MAXITERS = 10
GRADTOL = 1e-3
```

In [13]:
```
p1 = np.matrix([[2],[1.3],[1]])
p2 = np.matrix([[2],[0.5],[1]])
p3 = np.matrix([[0.5],[1.3],[1]])
p4 = np.matrix([[0.5],[0.5],[1]])
```

In [14]:
```
def f(x1,x2):
    return np.log(x1+0.3*x2+1)/3+np.log(x1-x2/2+1)/6+np.log(-x1/2+0.3*x2+1)/
```

In [15]:
```
def a(x1,x2):
    return x1 + 0.3*x2 + 1
def b(x1,x2):
    return x1 - 0.5*x2 + 1
def c(x1,x2):
    return -0.5*x1 + 0.3*x2 + 1
def d(x1,x2):
    return -0.5*x1 - 0.5*x2 + 1
```

In [16]:
```
def grad(x1, x2):
    delta_x1 = (1/3)*x1/a(x1,x2)+(1/6)*x1/b(x1,x2)-(1/6)*x1/c(x1,x2)-(1/12)*
    delta_x2 = (1/10)*x2/a(x1,x2)+(-1/12)*x2/b(x1,x2)+(1/10)*x2/c(x1,x2)-(1/
    return np.matrix([[delta_x1],[delta_x2]])
```

In [17]:
```
def hess(x1, x2):
    x1x1_1 = ((1/3)*a(x1, x2) + (1/3)*x1)/(a(x1, x2)*a(x1, x2))
    x1x1_2 = ((1/6)*b(x1, x2) + (1/6)*x1)/(b(x1, x2)*b(x1, x2))
    x1x1_3 = ((-1/6)*c(x1, x2) + (1/12)*x1)/(c(x1, x2)*c(x1, x2))
    x1x1_4 = ((-1/12)*d(x1, x2) + (1/24)*x1)/(d(x1, x2)*d(x1, x2))
    delta_x1x1 = x1x1_1 + x1x1_2 + x1x1_3+ x1x1_4
    x2x2_1 = ((1/10)*a(x1, x2) + (0.03)*x2)/(a(x1, x2)*a(x1, x2))
    x2x2_2 = ((-1/12)*b(x1, x2) + (-1/12)*x2)/(b(x1, x2)*b(x1, x2))
    x2x2_3 = ((1/10)*c(x1, x2) + (0.03)*x2)/(c(x1, x2)*c(x1, x2))
    x2x2_4 = ((-1/12)*d(x1, x2) + (1/24)*x2)/(d(x1, x2)*d(x1, x2))
    delta_x2x2 = x2x2_1 + x2x2_2 + x2x2_3+ x2x2_4
    x1x2_1 = (0.1)*x1/(a(x1, x2)*a(x1, x2))
    x1x2_2 = (-1/12)*x1/(b(x1, x2)*b(x1, x2))
    x1x2_3 = (-1/20)*x1/(c(x1, x2)*c(x1, x2))
    x1x2_4 = (1/24)*x1/(d(x1, x2)*d(x1, x2))
    delta_x1x2 = x1x2_1 + x1x2_2 + x1x2_3 + x1x2_4
    x2x1_1 = (0.1)*x2/(a(x1, x2)*a(x1, x2))
    x2x1_2 = (-1/12)*x2/(b(x1, x2)*b(x1, x2))
    x2x1_3 = (-1/20)*x2/(c(x1, x2)*c(x1, x2))
    x2x1_4 = (1/24)*x2/(d(x1, x2)*d(x1, x2))
    delta_x2x1 = x2x1_1 + x2x1_2 + x2x1_3 + x2x1_4
    return np.matrix([[delta_x1x1, delta_x1x2],[delta_x2x1, delta_x2x2]])
```

```
In [18]: x = np.zeros((2,1))
         for i in range(1, MAXITERS):
             val = f(x[0,0],x[1,0])
             v = np.dot(-np.linalg.inv(hess(x[0,0],x[1,0])), grad(x[0,0],x[1,0]))
             lamba_s = np.dot(grad(x[0,0],x[1,0]).T, v)
             if abs(lamba_s) < 2 * GRADTOL:
                 break
             t = 1
             while np.min(f(x[0,0], x[1,0]) - val + ALPHA*t*lamba_s) > 0:
                 t = BETA*t;
             x = x + np.dot(t,v);
```

```
In [24]: x[0,0]

Out[24]: 0.49236374576347364
```

```
In [28]: x[1,0]

Out[28]: 0.19737483648723647
```

```
In [29]: 1 - x[0,0] - x[1,0]

Out[29]: 0.31026141774928989
```

```
In [30]: f(x[0,0],x[1,0])

Out[30]: 0.062265683519574108
```

```
In [19]: def hessian(x):
             x_grad = np.gradient(x)
             hessian = np.empty((x.ndim, x.ndim) + x.shape, dtype=x.dtype)
             for k, grad_k in enumerate(x_grad):
                 # iterate over dimensions
                 # apply gradient again to every component of the first derivative.
                 tmp_grad = np.gradient(grad_k)
                 for l, grad_kl in enumerate(tmp_grad):
                     hessian[k, l, :, :] = grad_kl
             return hessian
```