f -> $x^2 - 10x + y^2 + 8 = 0$

--> $\frac{\partial f}{\partial x} = 2x - 10$ --> $\frac{\partial f}{\partial y} = 2y$

g -> $xy^2 + x - 10y + 8 = 0$

--> $\frac{\partial g}{\partial x} = y^2 + 1$ --> $\frac{\partial g}{\partial y} = 2xy - 10$

Jacobian matrix:

$$J = \begin{bmatrix} 2x - 10 & 2y \\ y^2 + 1 & 2xy - 10 \end{bmatrix}$$

In [11]:
```
import numpy as np
```

In [12]:
```
def jacobian_matrix(x, y):
    return np.array([[2 * x - 10, 2 * y], [y * y + 1, 2 * x * y - 10]])

initial_x = [[4], [4]] # chosen because 4^2-10(4)+4^2 = -8 and at close to solution (at
previous_x = initial_x

for i in range(10):
    x_1 = initial_x[0][0]
    x_2 = initial_x[1][0]
    jacobian = jacobian_matrix(x_1, x_2)
    initial_x = initial_x - np.matmul(np.linalg.inv(jacobian), np.array([[x_1 * x_1 - 10
    if abs(initial_x[0][0] - previous_x[0][0]) < 0.0000001 and abs(initial_x[1][0] - pre
        break
    previous_x = initial_x

print(initial_x)
```

```
[[2.19343942]
 [3.02046647]]
```

So $x_1 = 2.1934394$ and $x_2 = 3.0204665$

Checking in the original equations to see if they are close enough to zero:

In [13]:
```
# Check

x_1 = initial_x[0][0]
x_2 = initial_x[1][0]
print(x_1 * x_1 - 10 * x_1 + x_2 * x_2 + 8)
print(x_1 * x_2 * x_2 + x_1 - 10 * x_2 + 8)
```

```
-3.552713678800501e-15
3.552713678800501e-15
```

Close to 0 ==> So the solution is correct