In [1]:
```python
from sympy import init_session
import math
import matplotlib.pyplot as plt
```

In [2]:
```python
init_session(use_latex=True)
```

```
IPython console for SymPy 1.6.2 (Python 3.7.4-64-bit) (ground types: python)

These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()

Documentation can be found at https://docs.sympy.org/1.6.2/
```

In [3]:
```python
# Variables
theta_2,theta_3,theta_4 = symbols("theta_2,theta_3,theta_4")
theta_2,theta_3,theta_4
```

Out[3]: $(\theta_2, \theta_3, \theta_4)$

In [4]:
```python
# Parametros
a = 2
b = 7
c = 9
d = 6

a, b, c, d
```

Out[4]: $(2, 7, 9, 6)$

In [5]:
```python
# GDL
theta_2 = math.radians(30)
theta_2
```

Out[5]: 0.5235987755982988

In [6]:
```python
# Funciones
F1 = a*cos(theta_2) + b*cos(theta_3) - c*cos(theta_4) - d
F2 = a*sin(theta_2) + b*sin(theta_3) - c*sin(theta_4)
F1,F2
```

Out[6]: $(7\cos(\theta_3) - 9\cos(\theta_4) - 4.26794919243112,\ 7\sin(\theta_3) - 9\sin(\theta_4) + 1.0)$

In [58]:
```python
# Condiciones iniciales
Ini = Matrix([1,2])
Ini
```

Out[58]: $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$

In [59]:
```python
# Vector de funciones
Fun = Matrix([F1,F2])
Fun
```

Out[59]: $\begin{bmatrix} 7\cos(\theta_3) - 9\cos(\theta_4) - 4.26794919243112 \\ 7\sin(\theta_3) - 9\sin(\theta_4) + 1.0 \end{bmatrix}$

In [60]:
```python
# Variables
Var = Matrix([theta_3,theta_4])
Var
```

Out[60]: $\begin{bmatrix} \theta_3 \\ \theta_4 \end{bmatrix}$

In [62]:
```python
# Jacobiano
J = Fun.jacobian(Var)
J
```

Out[62]: $\begin{bmatrix} -7\sin(\theta_3) & 9\sin(\theta_4) \\ 7\cos(\theta_3) & -9\cos(\theta_4) \end{bmatrix}$

In [63]:
```python
# Inversión Jacobiano
Inv = J.inv()
Inv
```

Out[63]: $\begin{bmatrix} -\frac{\cos(\theta_4)}{7\sin(\theta_3)\cos(\theta_4) - 7\sin(\theta_4)\cos(\theta_3)} & -\frac{\sin(\theta_4)}{7\sin(\theta_3)\cos(\theta_4) - 7\sin(\theta_4)\cos(\theta_3)} \\ \frac{\cos(\theta_3)}{-9\sin(\theta_3)\cos(\theta_4) + 9\sin(\theta_4)\cos(\theta_3)} & \frac{\sin(\theta_3)}{-9\sin(\theta_3)\cos(\theta_4) + 9\sin(\theta_4)\cos(\theta_3)} \end{bmatrix}$

In [64]:
```python
# Vector Final
Fin = Matrix([0,0])
Fin
```

Out[64]: $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

In [65]:
```python
# Tolerancia
tol = 1e-6
tol
```

Out[65]: $1e - 06$

In [66]:
```python
# Función error
def error(Ini,Fin):
    err = sum(abs(Ini-Fin))
    return err
```

In [67]:
```python
# Error
err = error(Ini,Fin)
err
```

Out[67]: 3

In [68]:
```python
# Newton-Raphson MV
con = 0
while err>tol:
    plt.plot(con,Ini[0].evalf(),"o", color = "blue")
    plt.plot(con,Ini[1].evalf(),"v", color = "red")
    plt.grid()
    plt.show()

    Fin = Ini - Inv.subs([(theta_3,Ini[0]),(theta_4,Ini[1])])*Fun.subs([(theta_3,Ini[0]),(theta_4,Ini[1
])])
    err = error(Ini.evalf(),Fin.evalf())
    Ini = Fin.evalf()
    con+=1
    print(con)
    print(err)

Fin
```

```
1
0.518778000207701
2
0.272860506677624
3
0.0166091434433497
4
0.000170723787521698
5
2.73841571640787e-9
```

Out[68]: $\begin{bmatrix} 1.55050235907826 \\ 2.04702805088061 \end{bmatrix}$

In [21]:
```python
# Respuesta en grados
math.degrees(Fin[0]),math.degrees(Fin[1])
```

Out[21]: $(88.83724130026172,\ 117.28606786035022)$