Aproximación de Ecuaciones Diferenciales

Luis Eduardo Robles Jiménez

0224969

Input

In [15]:
```
#yp, a, b, n, c = 'y - t**2 + 1', 0, 2, 10, 0.5
#yp, a, b, n, c = '-2*t**3 + 12*t**2 - 20*t + 8.5', 0, 4, 8, 1
#yp, a, b, n, c = 'y - t**2 + 1', 0, 2, 10, 0.5
#yp, a, b, n, c = '-5*y + 5*t**2 + 2*t', 0, 1, 10, 1/3
yp, a, b, n, c = 't*exp(3*t)-2*y', 0, 1, 10, 0
```

Method

In [16]:
```
Euler(yp, a, b, n, c)
```

```
        f(x) = t*exp(3*t) - 2*y

0       0
0.1     0
0.2     0.0134985880757600
0.30000000000000004     0.0472412464684182
0.4     0.111581090509443
0.5     0.222069549317016
0.6     0.401740092970516
0.7     0.684370922241190
0.7999999999999999      1.11912863167269
0.8999999999999999      1.77715701578948
```

Out[16]:
```
(0.9999999999999999, 2.76090146787014)
```

In [ ]:
```
RunggeKutta(yp, a, b, n, c)
```

Euler

In [10]:
```
def Euler(fun, a, b, n, c):
    f = parse_expr(fun)
    print("\tf(x) =", f, end = "\n\n")
    h = (b - a)/n
    tT, yV, p = a, c, []
    for i in range(1, n+1):
        print(tT, yV, sep = "\t")
        yV += h*N(f.subs([(t, tT), (y, yV)]))
        tT += h
        p.append(yV)
    return (tT, yV)
```

Rungge Kutta (Cuarto Grado)

In [9]:
```
def RunggeKutta(fun, a, b, n, c):
    f = parse_expr(fun)
    print("\tf(x) =", f, end = "\n\n")
    h = (b - a)/n
    tT, yV, p = a, c, []
    for i in range(n):
        ku = h*N(f.subs([(t, tT), (y, yV)]))
        kd = h*N(f.subs([(t, tT + h/2), (y, yV + ku/2)]))
        kt = h*N(f.subs([(t, tT + h/2), (y, yV + kd/2)]))
        kc = h*N(f.subs([(t, tT + h), (y, yV + kt)]))
        yV += (ku + 2*kd + 2*kt + kc)/6
        tT += h
        p.append(yV)
        print(tT, yV, sep = "\t")
    return (tT, yV)
```

Run first

Run first

In [8]:
```
from sympy import *
t, y = symbols("t"), symbols("y")
```