Emplee la extrapolación de Richardson para aproximar *g'(1.65)* mediante $N_7(h)$ si se sabe que: $g(x) = 7ln^2(x) + 3tan(5x)$. Utilice $h=1/80$. Además, obtenga el valor exacto y el error en la aproximación

Solución

Para encontrar la solución necesitamos la siguiente matriz:

$N_1(h)$	$N_2(h)$	$N_3(h)$	$N_4(h)$	$N_5(h)$	N6(h)	N7(h)
N_1h/2)	N_2(h/2)	N_3(h/2)	N_4(h/2)	N_5(h/2)	N6(h/2)	
N_1h/4)	N_2(h/4)	N_3(h/4)	N_4(h/4)	N_5(h/4)		
N_1h/8)	N_2(h/8)	N_3(h/8)	N_4(h/8)			
N_1h/16)	N_2(h/16)	N_3(h/16)				
N_1h/32)	N_2(h/32)					
N_1h/64)						

En donde encontramos nueestros valores iniciales (primera columna con):

$$N_1(h) = \frac{g(c+h) - g(c-h)}{2h}$$

$$\dots$$

$$N_7(h) = \frac{g(c+h/64) - g(c-h/64)}{2(h/64)}$$

In [1]:
```
from sympy import *
from numpy import zeros, float64

x = symbols("x")
g = 7* (ln(x))**2 + 3 * tan(5*x)
c = 1.65
h = 1/80
```

```
# Aquí declaro mi n para ocuparla en el resto del ejercicio
n=7

N = zeros((n, n), dtype=float64)

#Imprimo la columna y verifico mi matriz
print(N)
```

```
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]
```

In [2]:

```
# Ahora a tráves de un bucle for lleno mi primera columna

for i in range(n):
    N[i][0] = (g.subs(x, c+h) - g.subs(x, c-h)) / (2*h)
    h = h/2

# Reinicio H
h= 1/80
# Verifico N
print(N)
```

```
[[107.4989198    0.           0.           0.           0.
    0.           0.        ]
 [105.65409163   0.           0.           0.           0.
    0.           0.        ]
 [105.20366512   0.           0.           0.           0.
    0.           0.        ]
 [105.09171575   0.           0.           0.           0.
    0.           0.        ]
 [105.06376923   0.           0.           0.           0.
    0.           0.        ]
 [105.05678515   0.           0.           0.           0.
    0.           0.        ]
 [105.05503929   0.           0.           0.           0.
    0.           0.        ]]
```

In [3]:
```
# Verifico si los valores son aproximados al "valor exacto"
# Ya que de no ser así hay algún error en las fórmulas

valor_exacto = diff(g).subs(x, 1.65)
print(valor_exacto)
```

```
105.054457349555
```

In [19]:
```
# Para llenar el resto de la matriz, realizo un bucle for anidado
# para todo el resto de elementos

for j in range(1, n):
    for i in range(n-j):
        # La j es diferente a la fórmual por la forma de indexar de Python
        N[i][j] = (4**(j) * N[i+1][j-1] - N[i][j-1]) / (4**(j) - 1)

#imprimo de forma que se vea bien la matriz
for i in N:
    for j in i:
        print(str(j).ljust(18), end="  ")
    print("\n")
```

```
107.49891980447075  105.03914891065487  105.05448121735434  105.0544573402626   105.05445734955771  105.05445734955808  105.05445734955472

105.65409163410884  105.05352294818562  105.05445771334216  105.0544573495214   105.05445734955808  105.05445734955474  0.0

105.20366511966643  105.05439929051988  105.0544573552061   105.05445734955794  105.05445734955474  0.0                 0.0

105.09171574780652  105.05445372616322  105.05445734964619  105.05445734955475  0.0                 0.0                 0.0

105.06376923157404  105.05445712317851  105.05445734955619  0.0                 0.0                 0.0                 0.0

105.05678515027739  105.05445733540758  0.0                 0.0                 0.0                 0.0                 0.0

105.05503928912503  0.0                 0.0                 0.0                 0.0                 0.0                 0.0
```

In [11]:
```
# Valores aproximado, exacto y error

valor_aproximado = N[0][n-1]
print("El valor aproximado es: ", valor_aproximado)
print("El valor exacto es:", valor_exacto)
```

```
error = abs(valor_aproximado - valor_exacto)
print("El error es de: ", error)
```

```
El valor aproximado es:  105.05445734955472
El valor exacto es: 105.054457349555
El error es de:  6.82121026329696e-13
```