ATIVIDADE 03 – Cálculo Numérico

Nome: João Pedro Rosa Cezarino
R.A: 22.120.021-5

```
[+] Exercício 1 [+]
```

```
[=] Interpolação Linear com Dois Pontos [=]
```

In [12]:
```python
import sympy as sy
import numpy as np
from scipy.interpolate import lagrange
from numpy.polynomial.polynomial import Polynomial
x=[7,14]
y=[17.7,35.2]
x = np.array(x)
y = np.array(y)
# valor 1 representa o grau do polinomio
z = np.polyfit(x, y, 1)
print("[=]", z)
p = np.poly1d(z)
print("[=] Resultado da Interpolação Linear com dois pontos: ", round(p(9),2))
```

```
[=] [2.5 0.2]
[=] Resultado da Interpolação Linear com dois pontos:  22.7
```

```
[=] Interpolação de Lagrange [=]
```

In [9]:
```python
import sympy as sy
import numpy as np
from scipy.interpolate import lagrange
from numpy.polynomial.polynomial import Polynomial
from scipy.interpolate import lagrange
x=[1,7,14]
y=[10.5,17.7,35.2]
x = np.array(x)
y = np.array(y)
poly = lagrange(x, y)
Polynomial(poly).coef
print("[=] Resultado da Interpolação de Lagrange: ",round(poly(9),1))
```

```
[=] Resultado da Interpolação de Lagrange:  21.7
```

[+] Exercício 2 [+]

[=] Método de Simpson [=]

In [5]:
```python
import numpy as np

def f(x):
    return x*np.exp(x/2)


def simpson13(x0,xn,n):
    # calculating step size
    h = (xn - x0) / n

    # Finding sum
    integration = f(x0) + f(xn)

    for i in range(1,n):
        k = x0 + i*h

        if i%2 == 0:
            integration = integration + 2 * f(k)
        else:
            integration = integration + 4 * f(k)

    # Finding final integration value
    integration = integration * h/3

    return integration
```

In [6]:
```python
result = simpson13(2, 3, 4)
print("[=] Resultado da integração por Simpson é: %0.2f" % (result) )
```

[=] Resultado da integração por Simpson é: 8.96

[+] Regra do Trapézio [+]

In [10]:
```python
import numpy as np
x = np.linspace(2.0, 3.0, 4)
y = x * np.exp(x/2)
I = np.trapz(y,x)
print("[=]", I)
```

[=] 9.016742584604309

In [11]:
```python
import scipy.integrate as si
import numpy as np
i = si.quad(lambda x: x*np.exp(x/2),2,3)
print("[=]", i)
```

[=] (8.963378140676129, 9.951348790200464e-14)