Polinômios ortogonais

Prof. Doherty Andrade

www.metodosnumericos.com.br

Dizemos que uma família de polinômios não nulos $p_0(x), p_1(x), \ldots, p_n(x), \ldots$ é uma família de polinômios ortogonais, relativamente ao produto interno $\langle \cdot, \cdot \rangle$, se verifica o seguinte

$$\langle p_i(x), p_j(x) \rangle = \begin{cases} 0, & i \neq j \\ C_i \neq 0, & i = j. \end{cases}$$

No estudo dos polinômios, utiliza-se comumente produtos internos da forma

$$\langle f, g \rangle = \int_a^b \omega(x) f(x) g(x) dx,$$

onde $\omega(x) \geq 0$ e $\omega(x) \not\equiv 0$ em cada subintervalo de $[a, b]$ e integrável em $[a, b]$ é chamada de função peso.

Os seguintes produtos internos são os mais comumente utilizados na determinação de polinômios ortogonais.

(1i) $\langle f, g \rangle = \int_{-1}^{1} f(x) g(x) dx$, isto é, $\omega(x) \equiv 1$ e $a = -1, b = 1$.

Este produto interno dará origem aos polinômios de Legendre.

Os polinômios de Legendre podem ser determinados pela seguinte lei de recorrência

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ (n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x). \end{aligned} \tag{1}$$

Outra forma de determiná-los é pela igualdade:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n.$$

(2i) $\langle f, g \rangle = \int_{-1}^{1} \frac{1}{\sqrt{1 - x^2}} f(x) g(x) dx$, isto é, $\omega(x) = \frac{1}{\sqrt{1 - x^2}}$ e $a = -1, b = 1$.

Este produto interno dará origem aos polinômios de Tchebycheff.

Uma forma alternativa para construir os polinômios de Tchebyscheff é dada por, para $n \geq 1$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

onde $T_0(x) = 1$, $T_1(x) = x$.

(3i) $\langle f, g \rangle = \int_0^\infty \mathrm{e}^{-x} f(x) g(x) dx$, isto é, $\omega(x) = \mathrm{e}^{-x}$ e $a = 0, b = \infty$.

Este produto interno dará origem aos polinômios de Laguerre.

(4i) $\langle f, g \rangle = \int_{-\infty}^\infty \mathrm{e}^{-x^2} f(x) g(x) dx$, isto é, $\omega(x) = \mathrm{e}^{-x^2}$ e $a = -\infty, b = \infty$.

Este produto interno dará origem aos polinômios de Hermite.

Vamos utilizar a biblioteca Python chamada mpmath para trabalhar com polinômios ortogonais.

1. Polinômios de Legendre

O código gera os primeiros polinômios de Legendre.

In [1]:
```
from sympy import symbols, legendre, init_printing

# Inicializa a impressão em LaTeX
init_printing()

# Símbolo
x = symbols('x')

# Gera e imprime as expressões para os polinômios de Legendre até o grau N
N = 10
for n in range(N+1):
    expression = legendre(n, x)
    latex_expression = f'P_{n}(x) = {expression}'
    print(latex_expression)
```

```
P_0(x) = 1
P_1(x) = x
P_2(x) = 3*x**2/2 - 1/2
P_3(x) = 5*x**3/2 - 3*x/2
P_4(x) = 35*x**4/8 - 15*x**2/4 + 3/8
P_5(x) = 63*x**5/8 - 35*x**3/4 + 15*x/8
P_6(x) = 231*x**6/16 - 315*x**4/16 + 105*x**2/16 - 5/16
P_7(x) = 429*x**7/16 - 693*x**5/16 + 315*x**3/16 - 35*x/16
P_8(x) = 6435*x**8/128 - 3003*x**6/32 + 3465*x**4/64 - 315*x**2/32 + 35/128
P_9(x) = 12155*x**9/128 - 6435*x**7/32 + 9009*x**5/64 - 1155*x**3/32 + 315*x/128
P_10(x) = 46189*x**10/256 - 109395*x**8/256 + 45045*x**6/128 - 15015*x**4/128 + 34
65*x**2/256 - 63/256
```

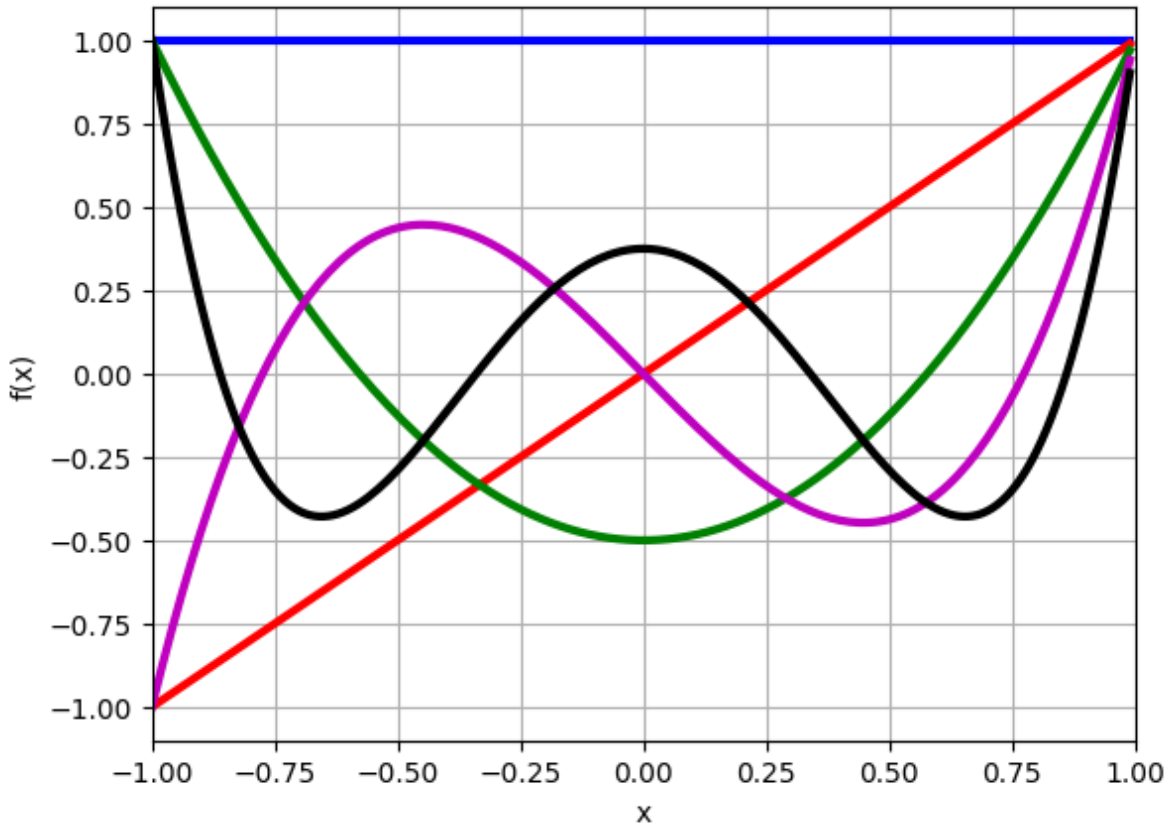
O código gera os polinômios de Legendre no formato LaTeX.

In [24]:
```
from sympy import symbols, legendre, latex

# Símbolo
x = symbols('x')

# Gera e imprime as expressões LaTeX compiladas para os polinômios de Legendre até
N = 10
for n in range(N+1):
    expression =  legendre(n, x)
    latex_expression = latex(expression)
    print(f'P_{n}(x) = {latex_expression}')
```

```
P_0(x) = 1
P_1(x) = x
P_2(x) = \frac{3 x^{2}}{2} - \frac{1}{2}
P_3(x) = \frac{5 x^{3}}{2} - \frac{3 x}{2}
P_4(x) = \frac{35 x^{4}}{8} - \frac{15 x^{2}}{4} + \frac{3}{8}
P_5(x) = \frac{63 x^{5}}{8} - \frac{35 x^{3}}{4} + \frac{15 x}{8}
P_6(x) = \frac{231 x^{6}}{16} - \frac{315 x^{4}}{16} + \frac{105 x^{2}}{16} - \fra
c{5}{16}
P_7(x) = \frac{429 x^{7}}{16} - \frac{693 x^{5}}{16} + \frac{315 x^{3}}{16} - \fra
c{35 x}{16}
P_8(x) = \frac{6435 x^{8}}{128} - \frac{3003 x^{6}}{32} + \frac{3465 x^{4}}{64} -
\frac{315 x^{2}}{32} + \frac{35}{128}
P_9(x) = \frac{12155 x^{9}}{128} - \frac{6435 x^{7}}{32} + \frac{9009 x^{5}}{64} -
\frac{1155 x^{3}}{32} + \frac{315 x}{128}
P_10(x) = \frac{46189 x^{10}}{256} - \frac{109395 x^{8}}{256} + \frac{45045 x^{6}}
{128} - \frac{15015 x^{4}}{128} + \frac{3465 x^{2}}{256} - \frac{63}{256}
```

In [25]:
```
from mpmath import *
mp.dps = 15; mp.pretty = True
```

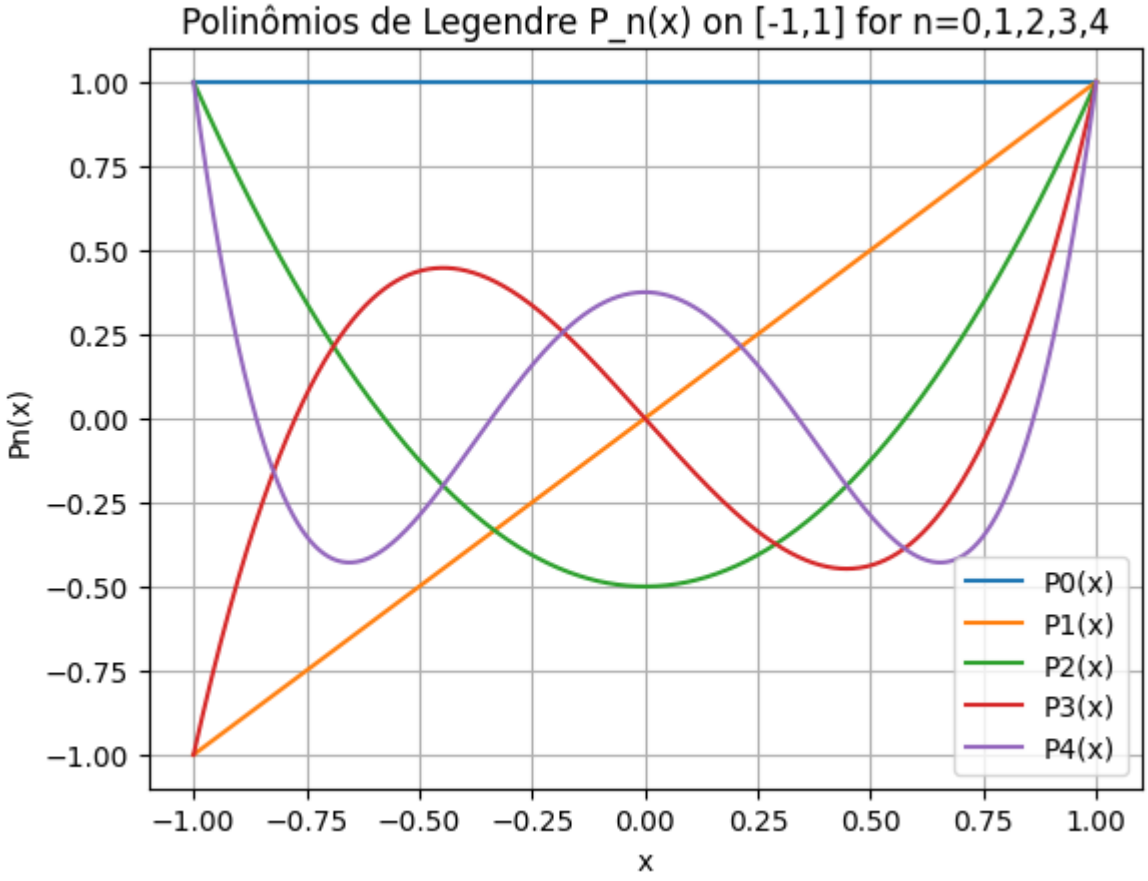
In [26]:
```
# Legendre polynomials P_n(x) on [-1,1] for n=0,1,2,3,4
f0 = lambda x: legendre(0,x)
f1 = lambda x: legendre(1,x)
f2 = lambda x: legendre(2,x)
f3 = lambda x: legendre(3,x)
f4 = lambda x: legendre(4,x)
plot([f0,f1,f2,f3,f4],[-1,1])
```



Um gráfico um pouco melhor com legendas.

In [6]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import legendre

# Definindo as funções
f0 = lambda x: legendre(0)(x)
```

```
f1 = lambda x: legendre(1)(x)
f2 = lambda x: legendre(2)(x)
f3 = lambda x: legendre(3)(x)
f4 = lambda x: legendre(4)(x)

# Gerando o gráfico
x_values = np.linspace(-1, 1, 1000)

plt.plot(x_values, f0(x_values), label='P0(x)')
plt.plot(x_values, f1(x_values), label='P1(x)')
plt.plot(x_values, f2(x_values), label='P2(x)')
plt.plot(x_values, f3(x_values), label='P3(x)')
plt.plot(x_values, f4(x_values), label='P4(x)')

# Adicionando Legendas
plt.legend()

# Exibindo o gráfico
plt.xlabel('x')
plt.ylabel('Pn(x)')
plt.title('Polinômios de Legendre P_n(x) on [-1,1] for n=0,1,2,3,4')
plt.grid(True)
plt.show()
```



Alguns polinômios de Legendre (não normalizados).

Grau	Polinômios de Legendre
0	1
1	x
2	$3x^2/2 - 1/2$
3	$5x^3/2 - 3x/2$
4	$\frac{35}{8}x^4 - \frac{15}{4}x^2 + 3/8$
5	$\frac{63}{8}x^5 - \frac{35}{4}x^3 + \frac{15}{8}x$
6	$\frac{231}{16}x^6 - \frac{315}{16}x^4 + \frac{105}{16}x^2 - \frac{5}{16}$
7	$\frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x$
8	$\frac{6435}{128}x^8 - \frac{3003}{32}x^6 + \frac{3465}{64}x^4 - \frac{315}{32}x^2 + \frac{35}{128}$
9	$\frac{12155}{128}x^9 - \frac{6435}{32}x^7 + \frac{9009}{64}x^5 - \frac{1155}{32}x^3 + \frac{315}{128}x$
10	$\frac{46189}{256}x^{10} - \frac{109395}{256}x^8 + \frac{45045}{128}x^6 - \frac{15015}{128}x^4 + \frac{3465}{256}x^2 - \frac{63}{256}$

Vamos testar a ortogonalidade de alguns polinômios de Legendre.

In [17]:
```
#precisamos de mpmath
from mpmath import *
mp.dps = 15; mp.pretty = True
```

In [18]:
```
for j in range(11):
    for k in range(11):
        R = chop(quad(lambda x: legendre(j,x)*legendre(k,x), [-1, 1]))
        print(j,k, R)
```

```
0 0 2.0
0 1 0.0
0 2 0.0
0 3 0.0
0 4 0.0
0 5 0.0
0 6 0.0
0 7 0.0
0 8 0.0
0 9 0.0
0 10 0.0
1 0 0.0
1 1 0.666666666666667
1 2 0.0
1 3 0.0
1 4 0.0
1 5 0.0
1 6 0.0
1 7 0.0
1 8 0.0
1 9 0.0
1 10 0.0
2 0 0.0
2 1 0.0
2 2 0.4
2 3 0.0
2 4 0.0
2 5 0.0
2 6 0.0
2 7 0.0
2 8 0.0
2 9 0.0
2 10 0.0
3 0 0.0
3 1 0.0
3 2 0.0
3 3 0.285714285714286
3 4 0.0
3 5 0.0
3 6 0.0
3 7 0.0
3 8 0.0
3 9 0.0
3 10 0.0
4 0 0.0
4 1 0.0
4 2 0.0
4 3 0.0
4 4 0.222222222222222
4 5 0.0
4 6 0.0
4 7 0.0
4 8 0.0
4 9 0.0
4 10 0.0
5 0 0.0
5 1 0.0
5 2 0.0
5 3 0.0
5 4 0.0
5 5 0.181818181818182
5 6 0.0
5 7 0.0
5 8 0.0
```

```
5 9 0.0
5 10 0.0
6 0 0.0
6 1 0.0
6 2 0.0
6 3 0.0
6 4 0.0
6 5 0.0
6 6 0.153846153846154
6 7 0.0
6 8 0.0
6 9 0.0
6 10 0.0
7 0 0.0
7 1 0.0
7 2 0.0
7 3 0.0
7 4 0.0
7 5 0.0
7 6 0.0
7 7 0.133333333333333
7 8 0.0
7 9 0.0
7 10 0.0
8 0 0.0
8 1 0.0
8 2 0.0
8 3 0.0
8 4 0.0
8 5 0.0
8 6 0.0
8 7 0.0
8 8 0.117647058823529
8 9 0.0
8 10 0.0
9 0 0.0
9 1 0.0
9 2 0.0
9 3 0.0
9 4 0.0
9 5 0.0
9 6 0.0
9 7 0.0
9 8 0.0
9 9 0.105263157894737
9 10 0.0
10 0 0.0
10 1 0.0
10 2 0.0
10 3 0.0
10 4 0.0
10 5 0.0
10 6 0.0
10 7 0.0
10 8 0.0
10 9 0.0
10 10 0.0952380952380952
```

2. Polinômios de Tchebycheff

Uma forma alternativa para construir os polinômios de Tchebyscheff é dada por, para $n \geq 1$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

onde $T_0(x) = 1$, $T_1(x) = x$.

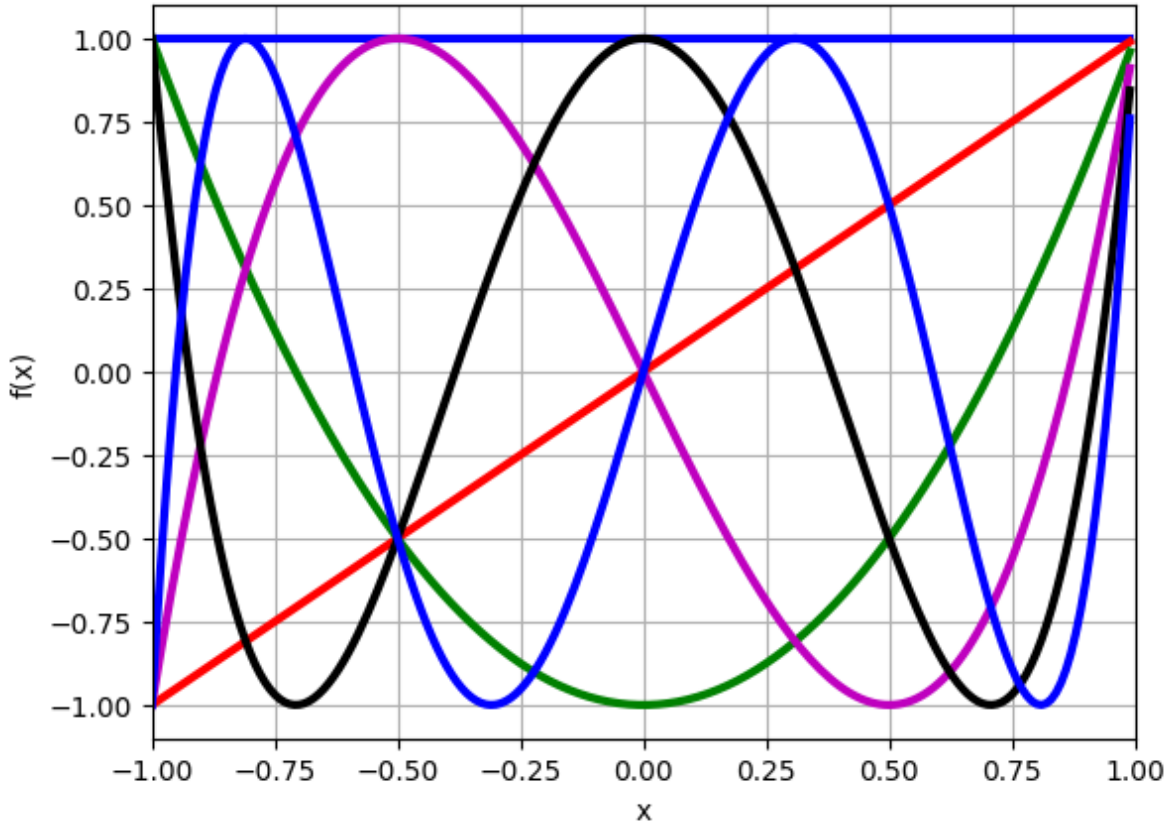
Veja o código abaixo que gera os primeiros polinômios.

In [27]:
```
from sympy import symbols, Eq, solve
from sympy import symbols, expand

# Símbolos
x, n = symbols('x n')

# Condições iniciais
T = {0: 1, 1: x}

# Relação recursiva para T_{n+1}(x)
for i in range(2, 6):  # Altere o valor de 6 conforme necessário para obter mais te
    T[i] = expand(2 * x * T[i - 1] - T[i - 2])

# Exibir os polinômios de Chebyshev até o termo desejado
for i in range(6):  # Altere o valor de 6 conforme necessário para obter mais termo
    print(f'T_{i}(x) = {T[i]}')
```

```
T_0(x) = 1
T_1(x) = x
T_2(x) = 2*x**2 - 1
T_3(x) = 4*x**3 - 3*x
T_4(x) = 8*x**4 - 8*x**2 + 1
T_5(x) = 16*x**5 - 20*x**3 + 5*x
```

In [28]:
```
from mpmath import *
mp.dps = 15; mp.pretty = True
```

In [29]:
```
# Chebyshev polynomials T_n(x) on [-1,1] for n=0,1,2,3,4
f0 = lambda x: chebyt(0,x)
f1 = lambda x: chebyt(1,x)
f2 = lambda x: chebyt(2,x)
f3 = lambda x: chebyt(3,x)
f4 = lambda x: chebyt(4,x)
f5 = lambda x: chebyt(5,x)
plot([f0,f1,f2,f3,f4,f5],[-1,1])
```



Veja alguns dos polinômios ortogonais de Tchebycheff.

Grau	Polinômios de Tchebyceff
0	1
1	x
2	$2\,x^2 - 1$
3	$4\,x^3 - 3\,x$
4	$8\,x^4 - 8\,x^2 + 1$
5	$16\,x^5 - 20\,x^3 + 5\,x$
6	$32\,x^6 - 48\,x^4 + 18\,x^2 - 1$
7	$64\,x^7 - 112\,x^5 + 56\,x^3 - 7\,x$
8	$128\,x^8 - 256\,x^6 + 160\,x^4 - 32\,x^2 + 1$
9	$256\,x^9 - 576\,x^7 + 432\,x^5 - 120\,x^3 + 9\,x$
10	$512\,x^{10} - 1280\,x^8 + 1120\,x^6 - 400\,x^4 + 50\,x^2 - 1$

Verificando a ortogonalidade entre alguns polinômios de Tchebycheff.

In [30]:
```
for j in range(11):
    for k in range(11):
        R = chop(quad(lambda x: chebyt(j,x)*chebyt(k,x)/sqrt(1-x**2), [-1, 1]))
        print(j,k, R)
```

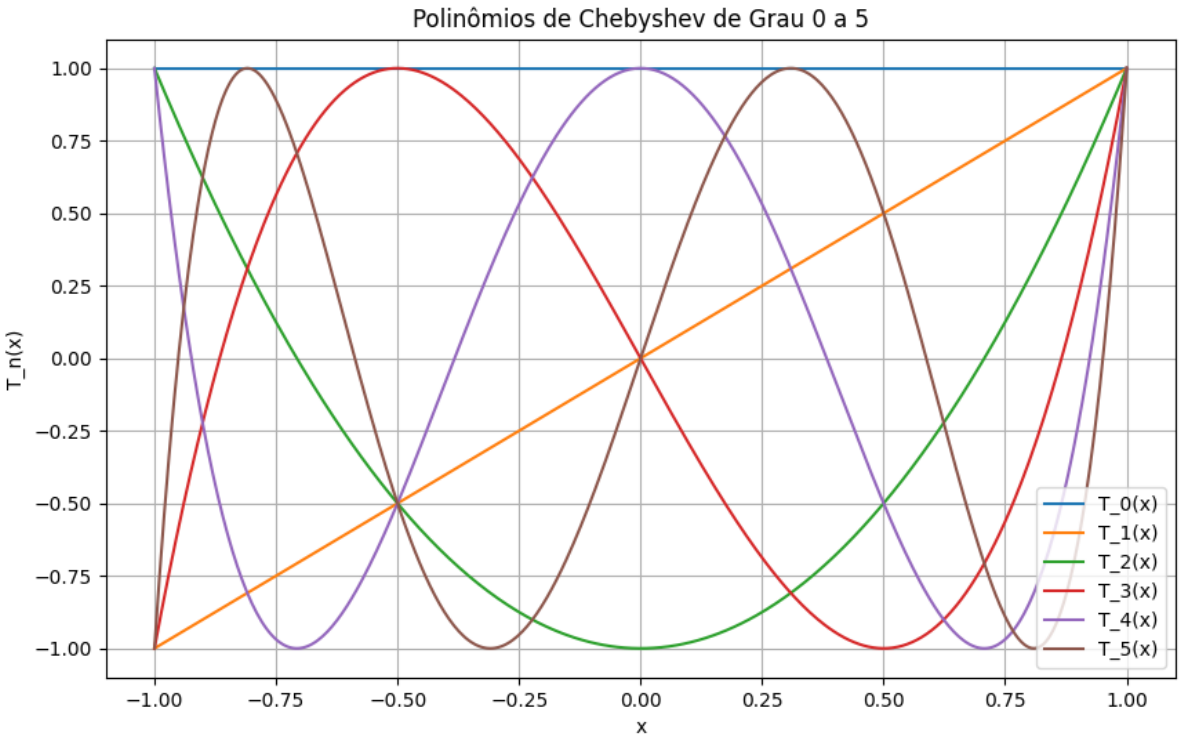
```
0 0 3.14159265252864
0 1 0.0
0 2 -1.33982266404492e-9
0 3 0.0
0 4 -1.06115030127495e-9
0 5 0.0
0 6 -1.06115030127496e-9
0 7 0.0
0 8 -8.3041953014951e-10
0 9 0.0
0 10 -8.30419530149473e-10
1 0 0.0
1 1 1.57079632545507
1 2 0.0
1 3 -1.06115030127492e-9
1 4 0.0
1 5 -1.06115030127492e-9
1 6 0.0
1 7 -8.30419530149598e-10
1 8 0.0
1 9 -8.3041953014953e-10
1 10 0.0
2 0 -1.33982266404492e-9
2 1 0.0
2 2 1.57079632573375
2 3 0.0
2 4 -1.06115030127495e-9
2 5 0.0
2 6 -8.30419530149546e-10
2 7 0.0
2 8 -8.30419530149531e-10
2 9 0.0
2 10 -8.30419530149513e-10
3 0 0.0
3 1 -1.06115030127492e-9
3 2 0.0
3 3 1.57079632573375
3 4 0.0
3 5 -8.3041953014956e-10
3 6 0.0
3 7 -8.30419530149534e-10
3 8 0.0
3 9 -8.30419530149521e-10
3 10 0.0
4 0 -1.06115030127495e-9
4 1 0.0
4 2 -1.06115030127495e-9
4 3 0.0
4 4 1.57079632596448
4 5 0.0
4 6 -8.30419530149533e-10
4 7 0.0
4 8 -8.3041953014955e-10
4 9 0.0
4 10 -8.30419530149552e-10
5 0 0.0
5 1 -1.06115030127492e-9
5 2 0.0
5 3 -8.3041953014956e-10
5 4 0.0
5 5 1.57079632596448
5 6 0.0
5 7 -8.30419530149508e-10
5 8 0.0
```

```
5 9 -8.3041953014953e-10
5 10 0.0
6 0 -1.06115030127496e-9
6 1 0.0
6 2 -8.30419530149546e-10
6 3 0.0
6 4 -8.30419530149533e-10
6 5 0.0
6 6 1.57079632596448
6 7 0.0
6 8 -8.30419530149533e-10
6 9 0.0
6 10 -1.06115030127497e-9
7 0 0.0
7 1 -8.30419530149598e-10
7 2 0.0
7 3 -8.30419530149534e-10
7 4 0.0
7 5 -8.30419530149508e-10
7 6 0.0
7 7 1.57079632596448
7 8 0.0
7 9 -1.06115030127497e-9
7 10 0.0
8 0 -8.3041953014951e-10
8 1 0.0
8 2 -8.30419530149531e-10
8 3 0.0
8 4 -8.3041953014955e-10
8 5 0.0
8 6 -8.30419530149533e-10
8 7 0.0
8 8 1.57079632573375
8 9 0.0
8 10 -1.06115030127497e-9
9 0 0.0
9 1 -8.3041953014953e-10
9 2 0.0
9 3 -8.30419530149521e-10
9 4 0.0
9 5 -8.3041953014953e-10
9 6 0.0
9 7 -1.06115030127497e-9
9 8 0.0
9 9 1.57079632573375
9 10 0.0
10 0 -8.30419530149473e-10
10 1 0.0
10 2 -8.30419530149513e-10
10 3 0.0
10 4 -8.30419530149552e-10
10 5 0.0
10 6 -1.06115030127497e-9
10 7 0.0
10 8 -1.06115030127497e-9
10 9 0.0
10 10 1.57079632555197
```

Gráfico dos polinômios de Tchebyvheff um pouco melhor com as legendas.

In [31]:
```
from scipy.special import chebyt
import matplotlib.pyplot as plt
import numpy as np
```

```
# Criando os polinômios de Chebyshev
T0 = chebyt(0)
T1 = chebyt(1)
T2 = chebyt(2)
T3 = chebyt(3)
T4 = chebyt(4)
T5 = chebyt(5)
# Gerando valores de x
x = np.linspace(-1, 1, 400)

# Calculando os valores dos polinômios nos pontos de x
y0 = T0(x)
y1 = T1(x)
y2 = T2(x)
y3 = T3(x)
y4 = T4(x)
y5 = T5(x)

# Plotando os resultados
plt.figure(figsize=(10, 6))
plt.plot(x, y0, label='T_0(x)')
plt.plot(x, y1, label='T_1(x)')
plt.plot(x, y2, label='T_2(x)')
plt.plot(x, y3, label='T_3(x)')
plt.plot(x, y4, label='T_4(x)')
plt.plot(x, y5, label='T_5(x)')


# Adicionando legendas
plt.legend()

# Exibindo o gráfico
plt.title("Polinômios de Chebyshev de Grau 0 a 5")
plt.xlabel("x")
plt.ylabel("T_n(x)")
plt.grid(True)
plt.show()
```



3. Polinômios de Hermite

Os polinômios de Hermite podem ser gerados por meio de fórmulas de recorrência. Os polinômios de Hermite são soluções da equação diferencial conhecida como a equação diferencial de Hermite, que está relacionada à função geradora de Hermite. Esses polinômios são denotados por $H_n(x)$, onde n é um número inteiro não negativo.

A fórmula de recorrência para os polinômios de Hermite pode ser expressa da seguinte forma:

$$H_{n+1}(x) = 2x \cdot H_n(x) - 2n \cdot H_{n-1}(x)$$

com condições iniciais:

$$H_0(x) = 1,$$

$$H_1(x) = 2x.$$

Essa fórmula de recorrência permite calcular os polinômios de Hermite de ordem superior $n+1$ com base nos polinômios de ordens inferiores n e $n-1$. Isso proporciona uma abordagem eficiente para gerar esses polinômios de forma recursiva.

Se estiver trabalhando em um programa ou implementação computacional, você pode usar essa fórmula para gerar os polinômios de Hermite de maneira iterativa, aproveitando os valores já calculados para ordens inferiores.

In [32]:
```
from sympy import symbols, expand

# Símbolos
x, n = symbols('x n')

# Condições iniciais
H = {0: 1, 1: 2 * x}

# Relação recursiva para H_{n+1}(x)
for i in range(2, 11):  # Altere o valor de 6 conforme necessário para obter mais t
    H[i] = expand(2 * x * H[i - 1] - 2 * (i - 1) * H[i - 2])

# Exibir os polinômios de Hermite até o termo desejado
for i in range(11):  # Altere o valor de 11 conforme necessário para obter mais ter
    print(f'H_{i}(x) = {H[i]}')
```

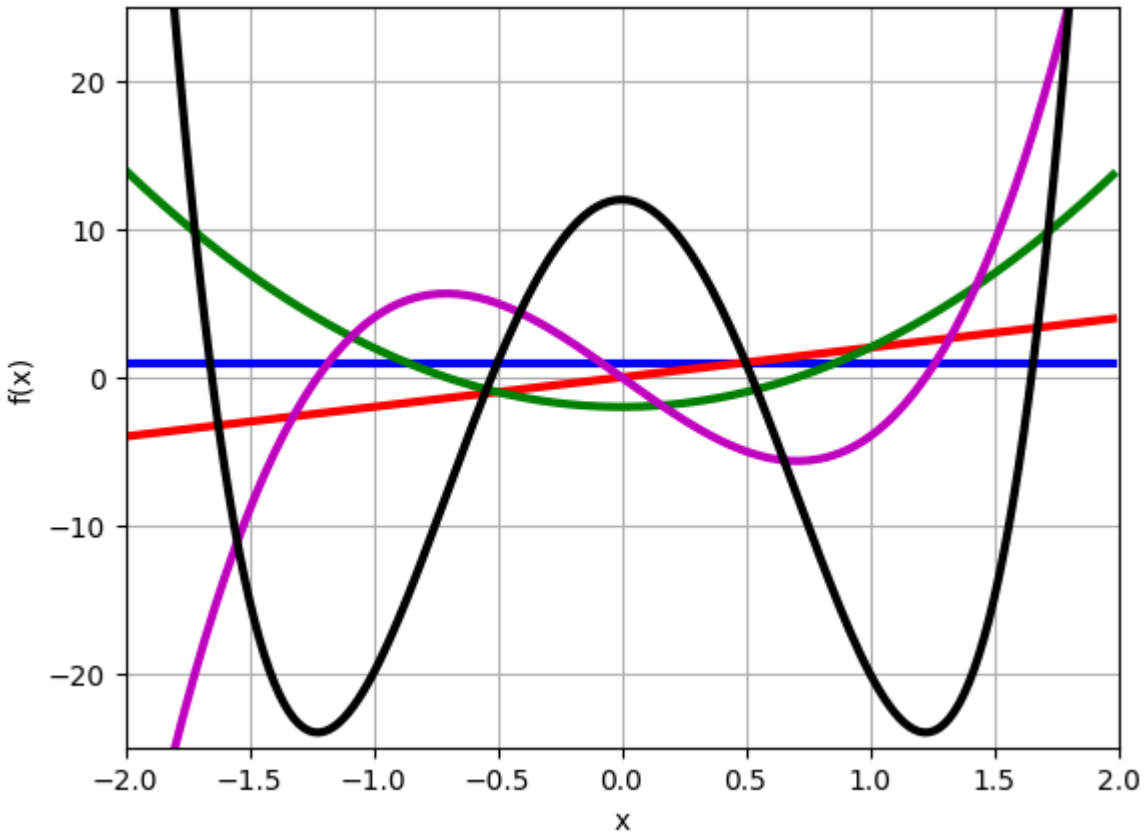
```
H_0(x) = 1
H_1(x) = 2*x
H_2(x) = 4*x**2 - 2
H_3(x) = 8*x**3 - 12*x
H_4(x) = 16*x**4 - 48*x**2 + 12
H_5(x) = 32*x**5 - 160*x**3 + 120*x
H_6(x) = 64*x**6 - 480*x**4 + 720*x**2 - 120
H_7(x) = 128*x**7 - 1344*x**5 + 3360*x**3 - 1680*x
H_8(x) = 256*x**8 - 3584*x**6 + 13440*x**4 - 13440*x**2 + 1680
H_9(x) = 512*x**9 - 9216*x**7 + 48384*x**5 - 80640*x**3 + 30240*x
H_10(x) = 1024*x**10 - 23040*x**8 + 161280*x**6 - 403200*x**4 + 302400*x**2 - 3024
0
```

Alguns polinômios de Hermite

Grau	Polinômios de Hermite
0	1
1	$2\,x$
2	$4\,x^2 - 2$
3	$8\,x^3 - 12\,x$
4	$16\,x^4 - 48\,x^2 + 12$
5	$32\,x^5 - 160\,x^3 + 120\,x$
6	$64\,x^6 - 480\,x^4 + 720\,x^2 - 120$
7	$\$128\,x^7 - 1344\,x^5 + 3360\,x^3 - 1680\,x$
8	$256\,x^8 - 3584\,x^6 + 13440\,x^4 - 13440\,x^2 + 1680$
9	$512\,x^9 - 9216\,x^7 + 48384\,x^5 - 80640\,x^3 + 30240\,x$
10	$1024\,x^{10} - 23040\,x^8 + 161280\,x^6 - 403200\,x^4 + 302400\,x^2 - 30240$

Gráficos

In [33]:
```
# Hermite polynomials H_n(x) on the real line for n=0,1,2,3,4
f0 = lambda x: hermite(0,x)
f1 = lambda x: hermite(1,x)
f2 = lambda x: hermite(2,x)
f3 = lambda x: hermite(3,x)
f4 = lambda x: hermite(4,x)
plot([f0,f1,f2,f3,f4],[-2,2],[-25,25])
```



Ortogonanlidade

In [34]:
```
for j in range(5):
    for k in range(5):
```

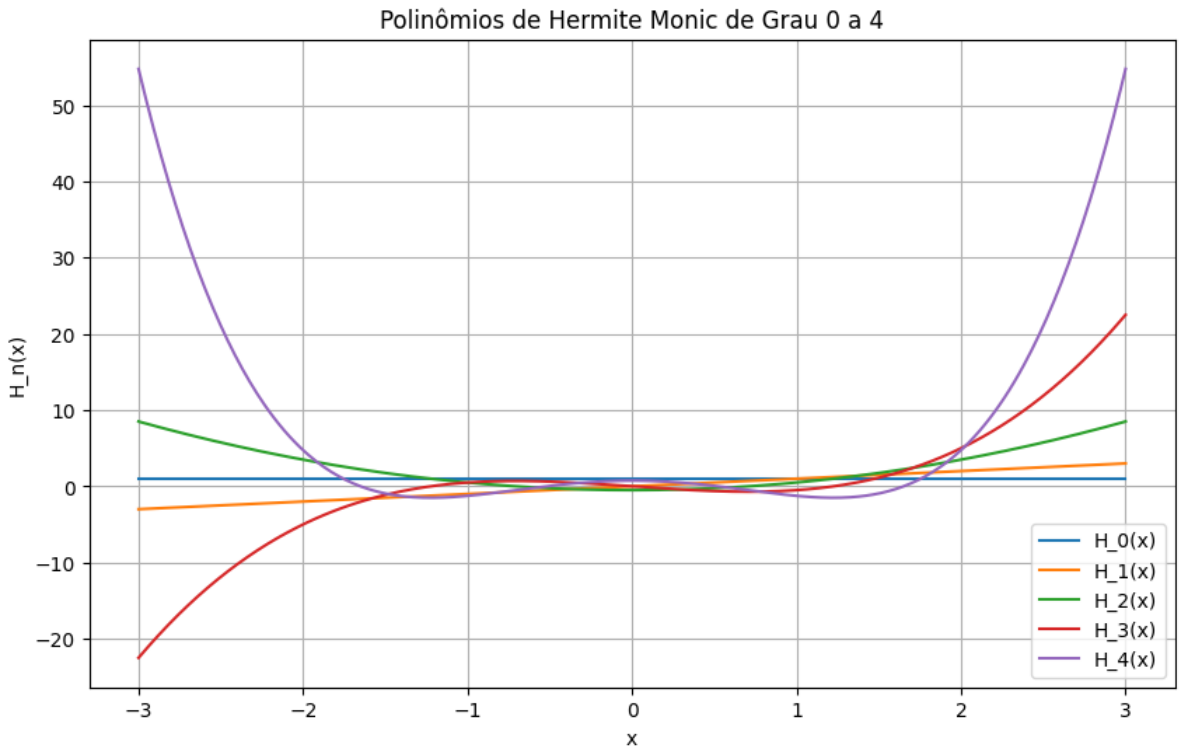
```
        R = chop(quad(lambda x: hermite(j,x)*hermite(k,x)*exp(-x**2), [-inf,inf]))
        print(j,k, R)
```

```
0 0 1.77245385090552
0 1 0.0
0 2 0.0
0 3 0.0
0 4 0.0
1 0 0.0
1 1 3.54490770181103
1 2 0.0
1 3 0.0
1 4 0.0
2 0 0.0
2 1 0.0
2 2 14.1796308072441
2 3 0.0
2 4 0.0
3 0 0.0
3 1 0.0
3 2 0.0
3 3 85.0777848434648
3 4 0.0
4 0 0.0
4 1 0.0
4 2 0.0
4 3 0.0
4 4 680.622278747718
```

In [42]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import hermite
```

In [43]:
```
# Criando os polinômios de Hermite
p_monic0 = hermite(0, monic=True)
p_monic1 = hermite(1, monic=True)
p_monic2 = hermite(2, monic=True)
p_monic3 = hermite(3, monic=True)
p_monic4 = hermite(4, monic=True)

# Gerando valores de x
x = np.linspace(-3, 3, 400)

# Calculando os valores dos polinômios nos pontos de x
y0 = p_monic0(x)
y1 = p_monic1(x)
y2 = p_monic2(x)
y3 = p_monic3(x)
y4 = p_monic4(x)

# Plotando os resultados
plt.figure(figsize=(10, 6))
plt.plot(x, y0, label='H_0(x)')
plt.plot(x, y1, label='H_1(x)')
plt.plot(x, y2, label='H_2(x)')
plt.plot(x, y3, label='H_3(x)')
plt.plot(x, y4, label='H_4(x)')

# Adicionando legendas
plt.legend()

# Exibindo o gráfico
plt.title("Polinômios de Hermite Monic de Grau 0 a 4")
plt.xlabel("x")
```

```
plt.ylabel("H_n(x)")
plt.grid(True)
plt.show()
```



4. Polinômios de Laguerre

Em matemática, os polinômios de Laguerre, nomeados em homenagem a Edmond Laguerre (1834–1886), são soluções não triviais da equação diferencial de Laguerre:

$$xy'' + (1 - x)y' + ny = 0, \quad y = y(x)$$

que é uma equação diferencial linear de segunda ordem. Essa equação tem soluções não singulares apenas se n for um número inteiro não negativo.

Às vezes, o nome "polinômios de Laguerre" é usado para soluções de:

$$xy'' + (\alpha + 1 - x)y' + ny = 0$$

onde $n$ainda é um número inteiro não negativo. Então, eles também são chamados de "polinômios de Laguerre generalizados", como será feito aqui (alternativamente, polinômios de Laguerre associados ou, raramente, polinômios de Sonine, em homenagem ao seu inventor Nikolay Yakovlevich Sonin).

De forma mais geral, uma função de Laguerre é uma solução quando n não é necessariamente um número inteiro não negativo.

Os polinômios de Laguerre também são usados para a quadratura de Gauss-Laguerre para calcular numericamente integrais da forma:

$$\int_0^\infty f(x)e^{-x}\,dx$$

Esses polinômios, geralmente denotados por $L_0, L_1, \ldots$, são uma sequência polinomial que pode ser definida pela fórmula de Rodrigues:

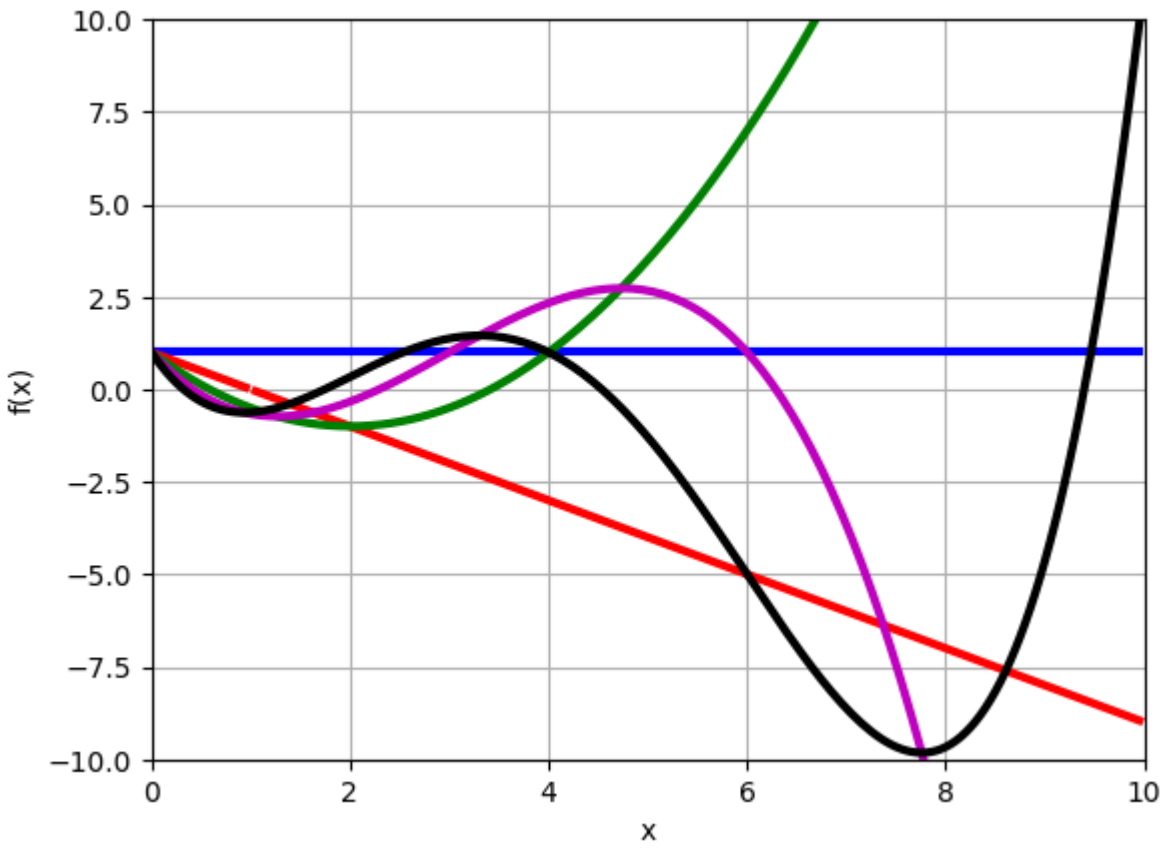
$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}\left(e^{-x}x^n\right) = \frac{1}{n!}\left(\frac{d}{dx} - 1\right)^n x^n.$$

Eles são polinômios ortogonais em relação a um produto interno:

$$\langle f, g\rangle = \int_0^\infty f(x)g(x)e^{-x}\,dx$$

Os polinômios de Laguerre surgem na mecânica quântica, na parte radial da solução da equação de Schrödinger para um átomo de um elétron. Eles também descrevem as funções estáticas de Wigner de sistemas osciladores em mecânica quântica no espaço de fase. Eles entram ainda na mecânica quântica do potencial Morse e do oscilador harmônico isotrópico 3D.

In [44]:
```
# laguerre polynomials L_n(x) on the real line for n=0,1,2,3,4
f0 = lambda x: laguerre(0,0,x)
f1 = lambda x: laguerre(1,0,x)
f2 = lambda x: laguerre(2,0,x)
f3 = lambda x: laguerre(3,0,x)
f4 = lambda x: laguerre(4,0,x)
plot([f0,f1,f2,f3,f4],[0,10],[-10,10])
```



Grau	Polinômios de Laguerre
0	1
1	$1 - x$
2	$1 - 2\,x + 1/2\,x^2$
3	$1 - 3\,x + 3/2\,x^2 - 1/6\,x^3$
4	$1 - 4\,x + 3\,x^2 - 2/3\,x^3 + 1/24\,x^4$
5	$1 - 5\,x + 5\,x^2 - 5/3\,x^3 + \frac{5}{24}\,x^4 - \frac{1}{120}\,x^5$
6	$1 - 6\,x + 15/2\,x^2 - 10/3\,x^3 + 5/8\,x^4 - 1/20\,x^5 + \frac{1}{720}\,x^6$
7	$1 - 7\,x + 21/2\,x^2 - \frac{35}{6}\,x^3 + \frac{35}{24}\,x^4 - \frac{7}{40}\,x^5 + \frac{7}{720}\,x^6 - \frac{1}{5040}\,x^7$
8	$1 - 8\,x + 14\,x^2 - \frac{28}{3}\,x^3 + \frac{35}{12}\,x^4 - \frac{7}{15}\,x^5 + \frac{7}{180}\,x^6 - \frac{1}{630}\,x^7 + \frac{1}{40320}\,x^8$
9	$1 - 9\,x + 18\,x^2 - 14\,x^3 + \frac{21}{4}\,x^4 - \frac{21}{20}\,x^5 + \frac{7}{60}\,x^6 - \frac{1}{140}\,x^7 + \frac{1}{4480}\,x^8 - \frac{1}{362880}\,x^9$
10	$1 - 10\,x + \frac{45}{2}\,x^2 - 20\,x^3 + \frac{35}{4}\,x^4 - \frac{21}{10}\,x^5 + \frac{7}{24}\,x^6 - 1/42\,x^7 + \frac{1}{896}\,x^8 - \frac{1}{36288}\,x^9$

Teste a ortogonalidade de alguns polinômios de Laguerre.

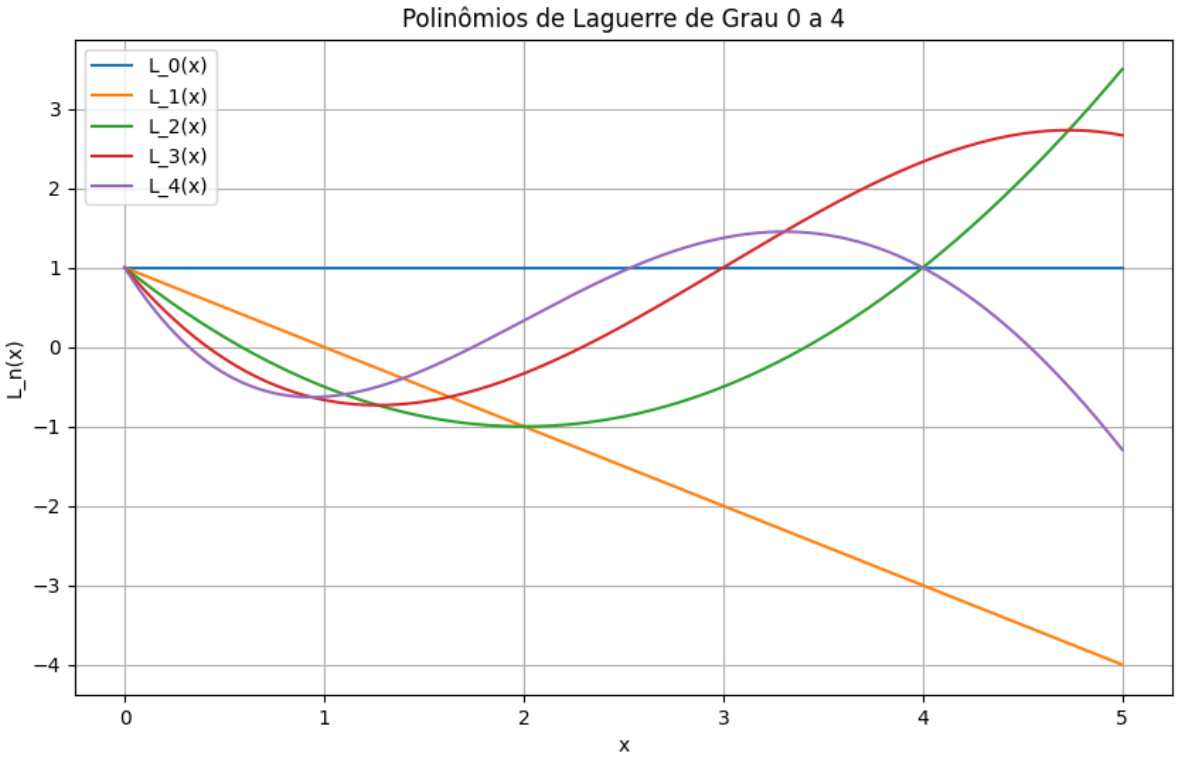
In [45]:
```
from mpmath import *
Lj = lambda x: laguerre(j,0,x)
Lk = lambda x: laguerre(k,0,x)
j = 5
k = 4
#calculando o produto interno
chop(quad(lambda x: exp(-x)*Lj(x)*Lk(x), [0,inf]))
```

Out[45]:
```
0.0
```

Gráficos

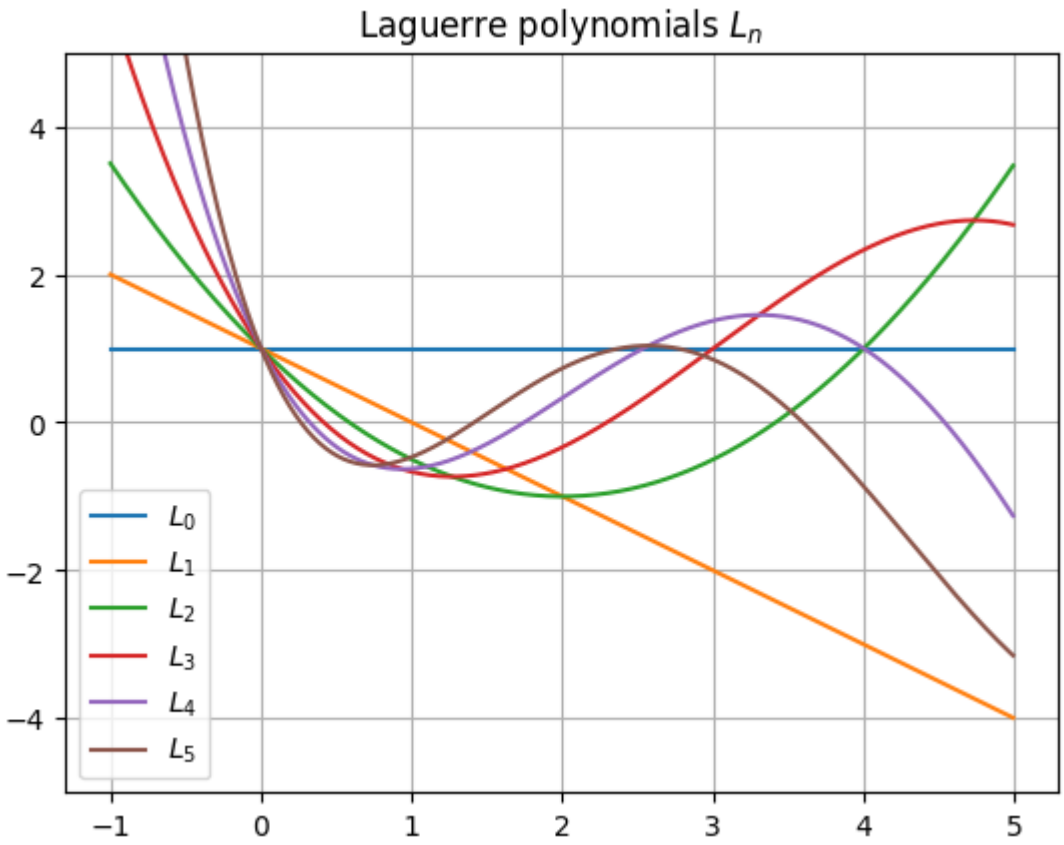
In [46]:
```
from scipy.special import laguerre
import matplotlib.pyplot as plt
import numpy as np

# Criando os polinômios de Laguerre
L0 = laguerre(0)
L1 = laguerre(1)
L2 = laguerre(2)
L3 = laguerre(3)
L4 = laguerre(4)

# Gerando valores de x
x = np.linspace(0, 5, 400)

# Calculando os valores dos polinômios nos pontos de x
y0 = L0(x)
y1 = L1(x)
y2 = L2(x)
y3 = L3(x)
y4 = L4(x)

# Plotando os resultados
plt.figure(figsize=(10, 6))
plt.plot(x, y0, label='L_0(x)')
plt.plot(x, y1, label='L_1(x)')
```

```
plt.plot(x, y2, label='L_2(x)')
plt.plot(x, y3, label='L_3(x)')
plt.plot(x, y4, label='L_4(x)')

# Adicionando legendas
plt.legend()

# Exibindo o gráfico
plt.title("Polinômios de Laguerre de Grau 0 a 4")
plt.xlabel("x")
plt.ylabel("L_n(x)")
plt.grid(True)
plt.show()
```



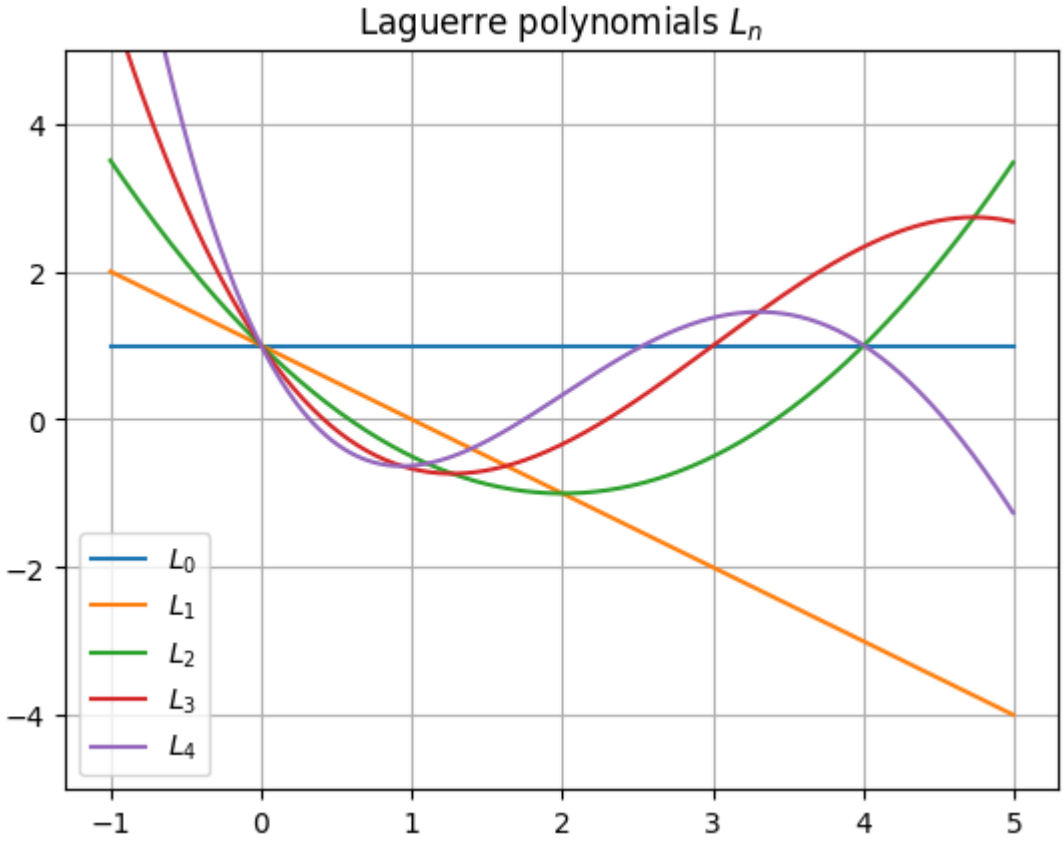
In [47]:
```
import numpy as np
from scipy.special import genlaguerre
from scipy.special import laguerre
```

In [48]:
```
import matplotlib.pyplot as plt
x = np.arange(-1.0, 5.0, 0.01)
fig, ax = plt.subplots()
ax.set_ylim(-5.0, 5.0)
ax.set_title(r'Laguerre polynomials $L_n$')
for n in np.arange(0, 6):
    ax.plot(x, laguerre(n)(x), label=rf'$L_{n}$')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```



In [49]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import laguerre
from sympy import symbols, laguerre as sym_laguerre, latex

# Símbolo simbólico
x_sym = symbols('x')

# Gerando valores de x
x_values = np.arange(-1.0, 5.0, 0.01)

fig, ax = plt.subplots()
ax.set_ylim(-5.0, 5.0)
ax.set_title(r'Laguerre polynomials $L_n$')

for n in np.arange(0, 5):
    laguerre_expr = sym_laguerre(n, x_sym)
    laguerre_lambda = lambda x: laguerre(n)(x)
    ax.plot(x_values, laguerre_lambda(x_values), label=rf'$L_{n}$')

    # Convertendo a expressão simbólica para LaTeX
    laguerre_latex = latex(laguerre_expr)
    print(f'L_{n}(x) = {laguerre_latex}')

plt.legend(loc='best')
plt.grid(True)
plt.show()
```

```
L_0(x) = 1
L_1(x) = 1 - x
L_2(x) = \frac{x^{2}}{2} - 2 x + 1
L_3(x) = - \frac{x^{3}}{6} + \frac{3 x^{2}}{2} - 3 x + 1
L_4(x) = \frac{x^{4}}{24} - \frac{2 x^{3}}{3} + 3 x^{2} - 4 x + 1
```



5. Aplicação

1.Propriedade: Se $p_0(x), p_1(x), \ldots, p_n(x)$ são polinômios ortogonais segundo o produto interno

$$\langle f, g \rangle = \int_a^b \omega(x) f(x) g(x) dx,$$

onde $\omega(x) \geq 0$ e $\omega(x) \not\equiv 0$ em cada subintervalo de $[a, b]$ é contínua em $[a, b]$, então $p_n(x)$ possui n raízes reais distintas pertencentes ao intervalo (a, b). Além disso, $p_n(x)$ e $p_{n+1}(x)$ possuem raízes entrelaçadas.

1. Integração numérica

Os métodos de integração numérica baseados nas fórmulas de Newton-Cotes considera pontos fixos igualmente espaçados. Gauss observou que a precisão poderia ser melhorada se as abscissas e os pesos não tiverem restrição.

Assim,

$$\int_a^b f(x) dx \approx \sum_{i=0}^{n} \omega_i f(x_i),$$

onde agora os pesos $\omega_0, \omega_1, \ldots, \omega_n$ e os nós $x_0, x_1, \ldots, x_n$ são determinados para obter a melhor precisão possível, isto é, deverá ser exata para polinômios de grau até $2n + 1$.

As fórmulas de quadratura de Gauss são baseadas no seguinte Teorema, que é a propriedade mais importante dos polinômios ortogonais.

TEOREMA: Sejam $p_0(x), p_1(x), p_2(x), \ldots, p_n(x), p_{n+1}(x) \ldots$ polinômios não nulos e ortogonais, segundo o produto interno

$$\langle f, g \rangle = \int_a^b \omega(x) f(x) g(x) dx,$$

onde $\omega(x) \geq 0$ e $\omega(x) \not\equiv 0$ em cada subintervalo de $[a, b]$ é contínua em $[a, b]$. Sejam $x_0, x_1, x_2, \ldots, x_n$ as raízes de $p_{n+1}(x)$. Se $f(x)$ é um polinômio de grau menor do que ou igual a $2n + 1$, então

$$\int_a^b \omega(x) f(x) dx = \sum_{k=0}^{n} \omega_k f(x_k),$$

onde

$$\omega_k = \int_a^b \omega(x) L_k(x) dx$$

com $P_n(x) = \sum_{k=0}^{n} f(x_k) L_k(x)$ é o polinômio interpolador de Lagrange que interpola f nos pontos $x_k, k = 0, 1, \ldots, n$.

Aproximações para π.

In [50]:
```
#pi- varias aproximacoes
from mpmath import *
mp.dps = 100; mp.pretty = True
print(mp.quad(lambda x: mp.exp(-x**2), [-mp.inf, mp.inf]) ** 2)
```

```
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170 68
```

In [ ]:
```

```