Лабораторная Работа 2

Назаров Рустам M3132 368563

Аналитический метод

Лабораторная работа №2

Вариант 25

Назаров Рустам 368563 М3132

$$f(x) = \sin\left(\frac{\pi}{3} + x\right) \qquad x = 0{,}2$$

1) Формула производной n-ого порядка

Найдем первые 8 производных

(каждую производную выразим в вид положительных синусов с помощью формул приведения)

$$f'(x) = \cos\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + x + \frac{\pi}{2}\right)$$

$$f''(x) = -\sin\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + x + \pi\right)$$

$$f'''(x) = -\cos\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + x + \frac{3\pi}{2}\right)$$

$$f^{4}(x) = \sin\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + x + 2\pi\right)$$

$$f^{5}(x) = \cos\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + \frac{5\pi}{2}\right)$$

$$f^{6}(x) = -\sin\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + 3\pi\right)$$

$$f^{7}(x) = -\cos\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + \frac{7\pi}{2}\right)$$

$$f^{8}(x) = \sin\left(\frac{\pi}{3} + x\right) = \sin\left(\frac{\pi}{3} + 4\pi\right)$$

Итак, мы видим, что каждая производная (число в синусе (аргумент)) больше предыдущей на $\frac{\pi}{2}$, а первая больше функции $\frac{\pi}{2}$, то есть можно находить производные, увеличивая функцию на $\frac{\pi}{2}$.

Отсюда формула производной n-го порядка для $f(x)$

$$f^n(x)=\sin\left(\frac{\pi}{3}+x+n\cdot\frac{\pi}{2}\right)$$

Докажем по математической индукцией

База: $n=1$

$$f'(x)=\left(\sin\left(\frac{\pi}{3}+x\right)\right)'=\cos\left(\frac{\pi}{3}+x\right)$$

$$f^1(x)=\sin\left(\frac{\pi}{3}+x+\frac{\pi}{2}\right)=\cos\left(\frac{\pi}{3}+x\right)$$

с помощью формулы приведения

Шаг: $n=k$

$$f^k(x)=\sin\left(\frac{\pi}{3}+x+k\cdot\frac{\pi}{2}\right)$$

$n=k+1$

$$f^{k+1}(x)\stackrel{?}{=}\sin\left(\frac{\pi}{3}+x+(k+1)\frac{\pi}{2}\right)$$

$$f^{k+1}(x)=\left(f^k(x)\right)'=\left(\sin\left(\frac{\pi}{3}+x+k\cdot\frac{\pi}{2}\right)\right)'=\cos\left(\frac{\pi}{3}+x+k\cdot\frac{\pi}{2}\right)=$$

$$\underset{\text{формула приведения}}{=}\sin\left(\frac{\pi}{3}+x+k\cdot\frac{\pi}{2}+\frac{\pi}{2}\right)=\sin\left(\frac{\pi}{3}+x+(k+1)\cdot\frac{\pi}{2}\right)$$

$$\Downarrow$$

$$f^{k+1}(x)=\sin\left(\frac{\pi}{3}+x+(k+1)\cdot\frac{\pi}{2}\right)$$

доказали

2) Многочлен Тейлора n-го порядка по степеням x

формула Тейлора: $P_n(x)=\sum\limits_{k=0}^{n}\frac{f^k(x_0)\,(x-x_0)^k}{k!}$

так как $f^k(x_0)=\sin\left(\frac{\pi}{3}+x_0+k\cdot\frac{\pi}{2}\right)$; $x_0=0$ получаем:

$$P_n(x)=\sum_{k=0}^{n}\sin\left(\frac{\pi}{3}+k\cdot\frac{\pi}{2}\right)\cdot\frac{x^k}{k!}$$

3) Получим многочлен Тейлора n-го порядка

Так как $\sin\left(\frac{\pi}{3}+x+\overset{\text{четность}}{2k}\cdot\frac{\pi}{2}\right)=(-1)^k\cdot\frac{\sqrt{3}}{2}$ из формул приведения

$\sin\left(\frac{\pi}{3}+x+\underset{\text{нечетность}}{(2k+1)}\cdot\frac{\pi}{2}\right)=(-1)^k\cdot\frac{1}{2}$

И так как по разложениям Тейлора

$$\cos x = 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\boxed{(-1)^m\cdot\frac{x^{2m}}{(2m)!}}+\ldots$$

$$\sin x = x-\frac{x^3}{3!}+\frac{x^5}{5!}-\boxed{(-1)^{m}\cdot\frac{x^{2m+1}}{(2m+1)!}}+\ldots$$

Итак,

$$f(x)=\sin\left(\frac{\pi}{3}+x\right)=\sin\frac{\pi}{3}\cdot\cos x+\sin x\cdot\cos\frac{\pi}{3}=$$

$$=\frac{\sqrt{3}}{2}\cdot\cos x+\frac{1}{2}\cdot\sin x=\frac{\sqrt{3}}{2}\sum_{k=0}^{\left[\frac{n}{2}\right]}(-1)^k\frac{x^{2k}}{(2k)!}+\frac{1}{2}\sum_{k=0}^{\left[\frac{n-1}{2}\right]}(-1)^k\cdot\frac{x^{2k+1}}{(2k+1)!}=$$

(из разложений Тейлора)

$$=\sum_{k=0}^{\left[\frac{n}{2}\right]}\underbrace{(-1)^k\cdot\frac{\sqrt{3}}{2}}_{\text{формулы приведения}}\cdot\frac{x^{2k}}{(2k)!}+\sum_{k=0}^{\left[\frac{n-1}{2}\right]}\underbrace{(-1)^k\cdot\frac{1}{2}}_{\text{формулы приведения}}\cdot\frac{x^{2k+1}}{(2k+1)!}=$$

$$=\sum_{k=0}^{\left[\frac{n}{2}\right]}\sin\left(\frac{\pi}{3}+2k\cdot\frac{\pi}{2}\right)\cdot\frac{x^{2k}}{(2k)!}+\sum_{k=0}^{\left[\frac{n-1}{2}\right]}\sin\left(\frac{\pi}{3}+(2k+1)\frac{\pi}{2}\right)\cdot\frac{x^{2k+1}}{(2k+1)!}=$$

$$=\sum_{k=0}^{n}\sin\left(\frac{\pi}{3}+k\cdot\frac{\pi}{2}\right)\cdot\frac{x^k}{k!}$$

Вывод: то же, что и во 2 (втором) пункте

4 Остаточный член формулы Тейлора

Остаточный член формулы Лагранжа

$$R_n(x,x_0)=\frac{f^{n+1}(\xi)\cdot(x-x_0)^{n+1}}{(n+1)!}\qquad \xi\in(0,x)$$

Так как: $f(x)=\sin(\frac{\pi}{3}+x)$

$a=0,2;\ \Delta_1=10^{-3};\ \Delta_2=10^{-6};\ x_0=0;\ x=a;\ \xi\in(0,a)$

Тогда

$$f^{n+1}(\xi)=\sin\left(\frac{\pi}{3}+\xi+(n+1)\frac{\pi}{2}\right).$$

$$R_n(a,0)=\sin\left(\frac{\pi}{3}+\xi+(n+1)\cdot\frac{\pi}{2}\right)\cdot\frac{a^{n+1}}{(n+1)!}$$

Оценим $|R_n(a,0)|$ сверху

Так как $|\sin\alpha|\le 1 \Rightarrow \left|\sin\left(\frac{\pi}{3}+\xi+(n+1)\frac{\pi}{2}\right)\right|\le 1 \Rightarrow$

$$\Rightarrow |R_n(a,0)|=\left|\sin\left(\frac{\pi}{3}+\xi+(n+1)\frac{\pi}{2}\right)\cdot\frac{a^{n+1}}{(n+1)!}\right|\le\frac{a^{n+1}}{(n+1)!}$$

т.к. ≤ 1.
синус уменьшает или не изменяет

Найдем n_1, n_2 для неравенства Δ_1 и Δ_2

$\Delta_1=10^{-3}$

$|R_n(a,0)|<10^{-3}$

$n_1=1\Rightarrow\frac{(0,2)^2}{2!}=0,02=2\cdot10^{-2}>10^{-3}$

$n=2\Rightarrow\frac{(0,2)^3}{3!}=\frac{4}{3}\cdot10^{-3}>10^{-3}$

$$\boxed{n=3\Rightarrow\frac{(0,2)^4}{4!}=\frac{2}{3}\cdot10^{-4}<10^{-3}}$$

Δ_1

$n_1=3$

$\Delta_2=10^{-6}$

$|R_n(a,0)|<10^{-6}$

$n=4\Rightarrow\frac{(0,2)^5}{5!}=\frac{4}{15}\cdot10^{-5}$

$$\boxed{n=5\Rightarrow\frac{(0,2)^6}{6!}=\frac{4}{45}\cdot10^{-6}<10^{-6}}$$

Δ_2

$n_2=5$

Численный метод

Язык Программирования: Python Версия языка: 3.8

1) Построение графиков f(x) и многочленов Тейлора порядков 1,2,...,n_2

In [1]:
```
# Импортируем необходимые библиотеки
import numpy as np
import matplotlib.pyplot as plt
import math
%matplotlib inline
```

In [2]:
```
# Данная нам функция
def func(X):
    f_x = np.sin(math.pi / 3 + X)
    return f_x
```

In [3]:
```
x = np.arange(-10, 10, 0.1)       # Беру точки x от -10 до 10 с шагом 0.1
f_x = func(x)                   # Получаем значения данной функции при разных x
```

In [4]:
```
# Значения порядков
n_1 = 3
n_2 = 5
```

In [5]:
```
fig = plt.figure(figsize=(100,60))    # Размер графика
# Цвет окантовки
fig.patch.set_facecolor('#DDA0DD')

# Заголовок
ax = fig.add_subplot()
fig.subplots_adjust(top=0.93)
fig.suptitle('Данная функция f(x)', fontsize=150, fontweight='bold')

# Размер координат осей абсцисс и ординат
plt.xticks(fontsize = 120)
plt.yticks(fontsize = 120)
# Разметка на графике
plt.grid(axis = 'both', linewidth = 4)

# Выводим график функции
plt.plot(x, f_x, color='#000080', linewidth=60, label='Функция')

# Выведем Легенду
plt.legend(loc=4, prop={'size': 100})

# Вывод полученного графика
plt.show()
```

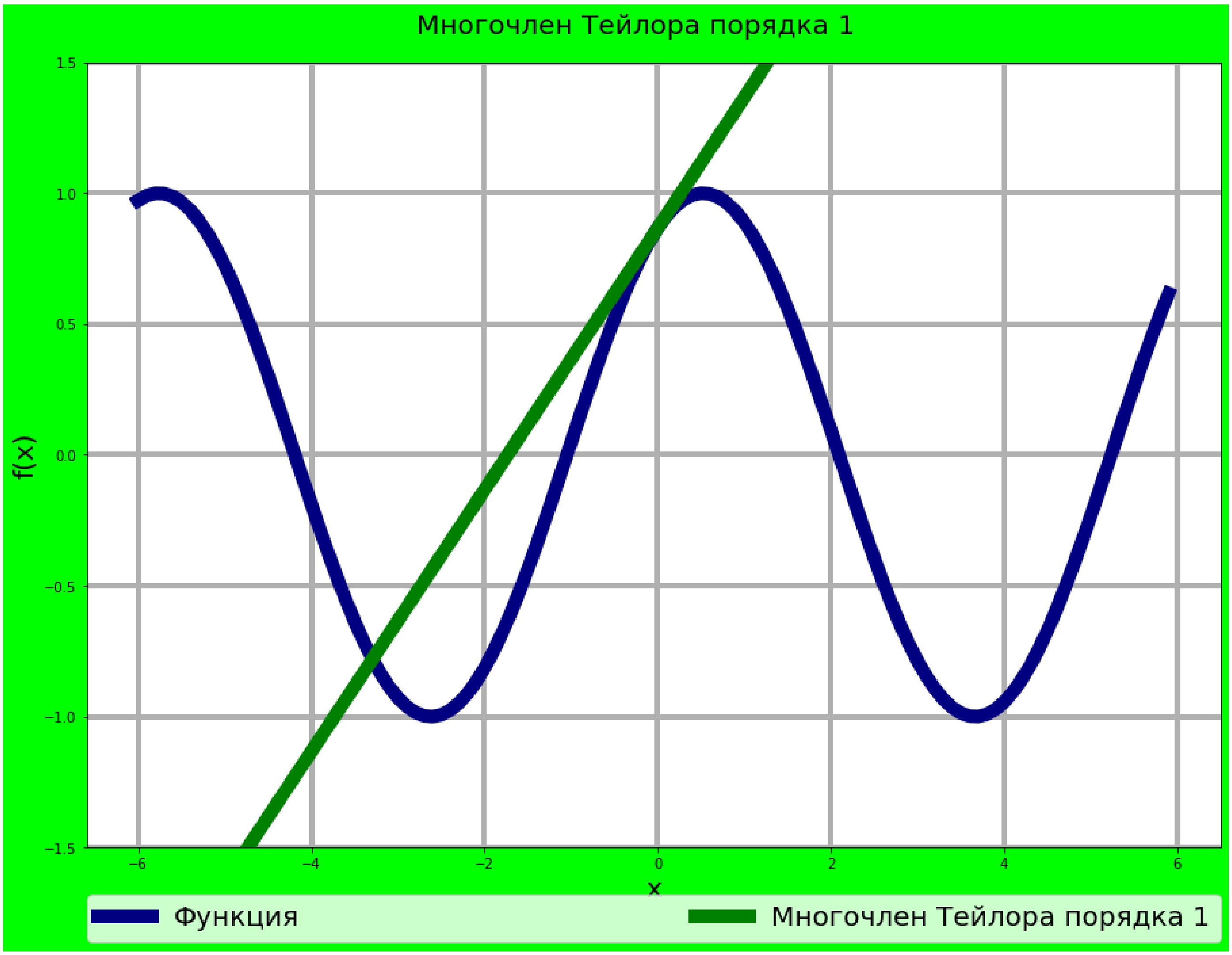
In [6]:
```
# Многочлен Тейлора n-ого порядка по степеням x
def polinom_Taylor_n_order(X, n):
    Y = 0;
    # Считаем сумму
    for k in range(n + 1):
        Y += math.sin(math.pi / 3 + k * math.pi / 2) * X**k / math.factorial(k)
    return Y
```

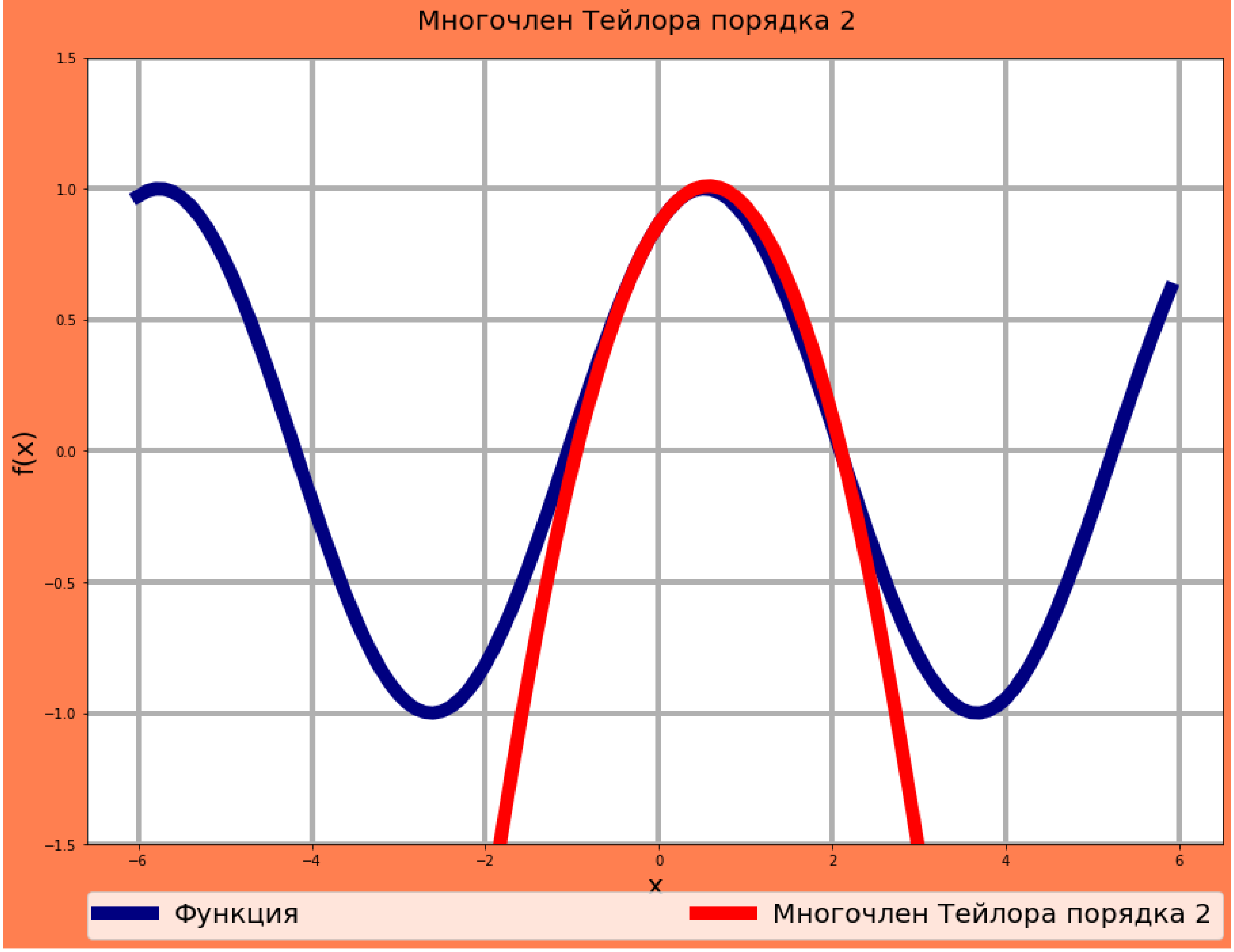
In [7]:
```
# Библиотека цветов
colors = ["red", 'green', '#FF0000', 'darkred', '#FAC205', '#F97306', "yellow"]
colors_back = ['lime', 'coral', 'tan', 'gold', '#FFA500']
```

In [8]:
```
# Беру значения x от -6 до 6 с шагом 0.1,
# Чтобы вблизи наложить график функции на многочлен
x = np.arange(-6, 6, 0.1)
f_x = func(x)  # Получаю значения на Ординат
```

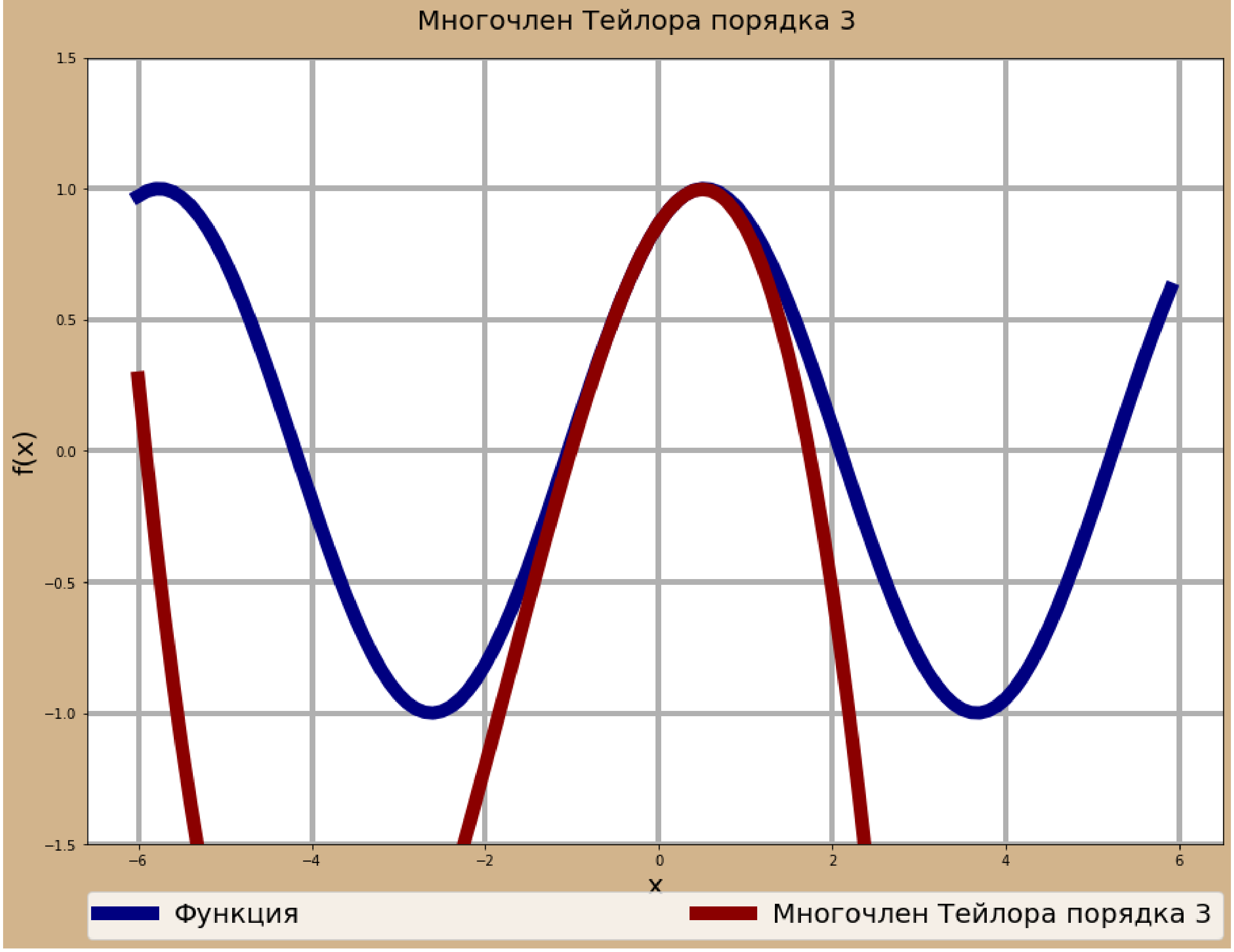
In [9]:
```
# Беру значения по оси Абсцисс от -6 до 6 с шагом 0.1 для многочлена Тейлора разных поря
x_n = np.arange(-6, 6, 0.1)

# Циклом перебираю необходимые порядки до n_2
for n in range(1, n_2 + 1):

    fig = plt.figure(figsize=(15,10))    # Размер графика
    # Цвет окантовки
    fig.patch.set_facecolor(colors_back[n - 1])

    # Считаю значения для текущего порядка по Ординат
    f_x_n = polinom_Taylor_n_order(x_n, n)

    # Вывожу график функции на текущее окно
    plt.plot(x, f_x, color='#000080', linewidth=10, label='Функция')

    # Вывожу многочлен Тейлора текущего порядка на его окно
    plt.plot(x_n, f_x_n, color=colors[n], linewidth=10, label=f'Многочлен Тейлора порядк
```

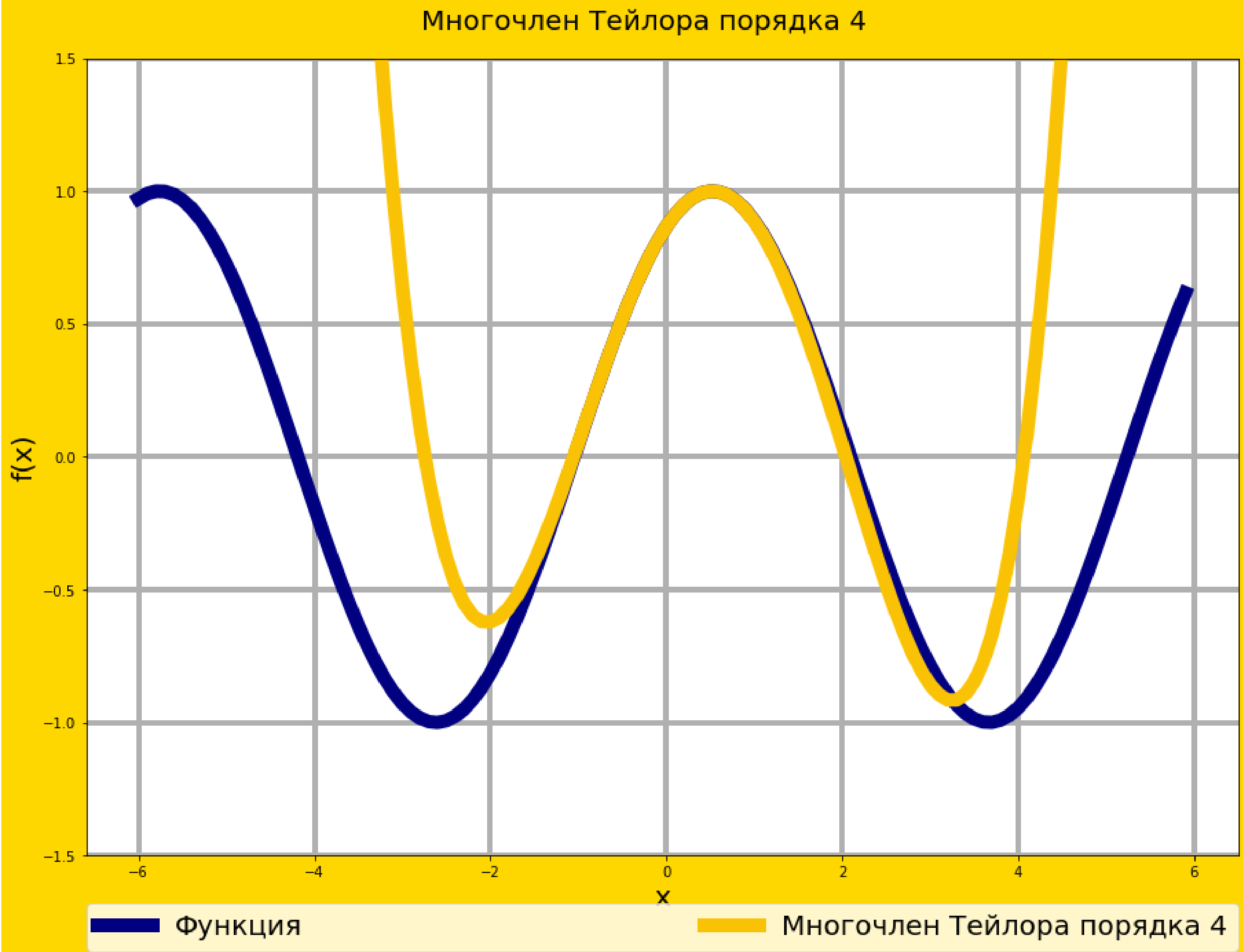
```
    fig.subplots_adjust(top=0.93)
    fig.suptitle(f'Многочлен Тейлора порядка {n}', fontsize=20)

    # Разметка
    plt.grid(axis = 'both', linewidth = 4)

    # Легенда
    plt.legend(bbox_to_anchor=(0., -0.12, 1., -0.12), loc='lower left',
                        ncol=2, mode="expand", borderaxespad=0., prop={'size': 20})

    # Название осей
    plt.xlabel("x", fontsize=20)
    plt.ylabel("f(x)", fontsize=20)

    # Ограничение для оси Ординат
    plt.ylim([-1.5, 1.5])

    # Вывод полученного графика
    plt.show()
```

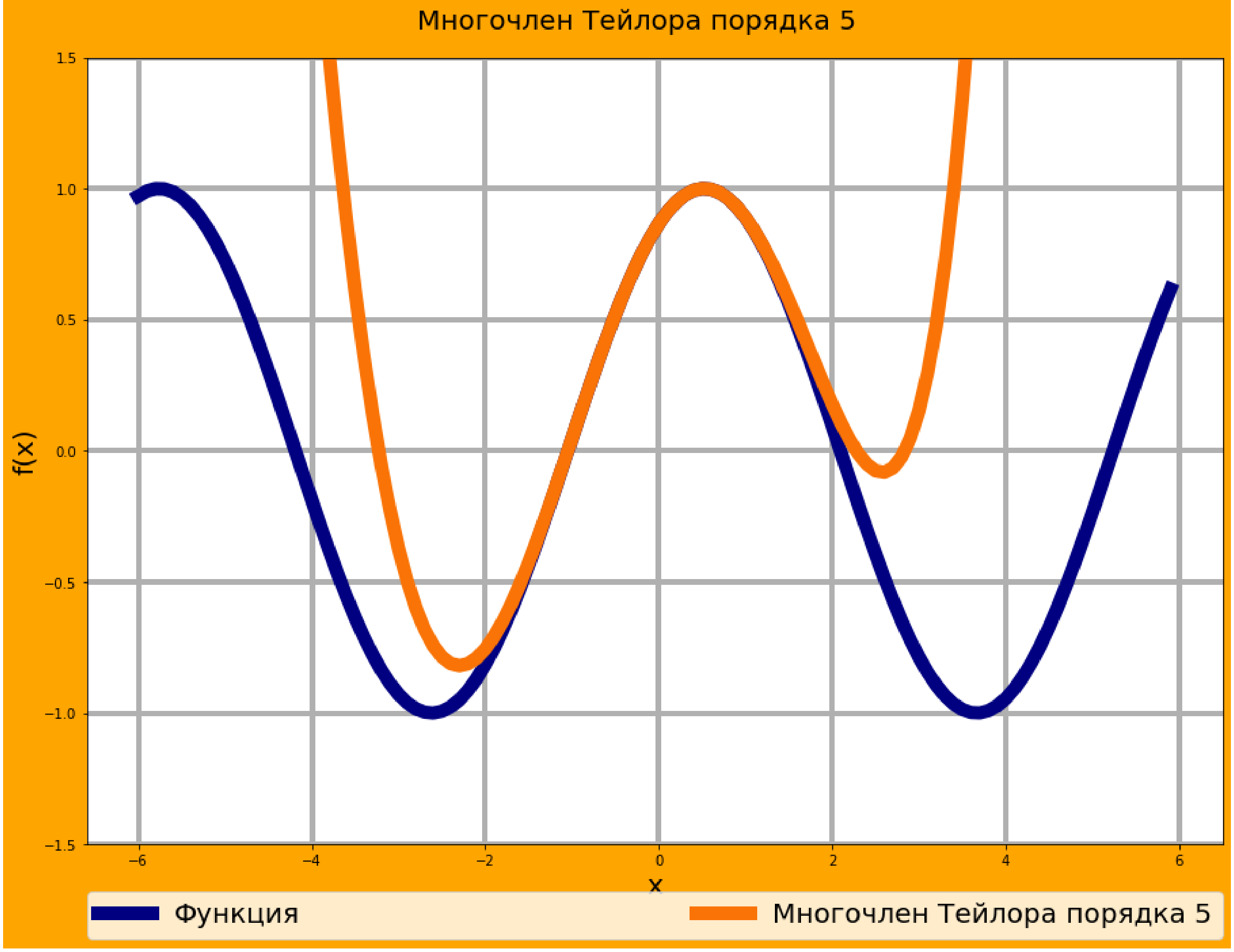




Многочлен Тейлора порядка 2
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−6
−4
−2
0
2
4
6
Функция
Многочлен Тейлора порядка 2


Многочлен Тейлора порядка 3
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−6
−4
−2
0
2
4
6
Функция
Многочлен Тейлора порядка 3


Многочлен Тейлора порядка 4
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−6
−4
−2
0
2
4
6
Функция
Многочлен Тейлора порядка 4



2) Приближенные значения f(a), заменяя на многочлены Тейлора порядков n_1, n_2

In [10]:
```
# Многочлен Тейлора n-ого порядка по степеням x
def polinom_Taylor_n_order(X, n):
    Y = 0;
    # Считаем сумму
    for k in range(n + 1):
        Y += math.sin(math.pi / 3 + k * math.pi / 2) * X**k / math.factorial(k)
    return Y
```

In [11]:
```
# Задаю значение точки a для f(a)
a = 0.2

# Задаю значения n_1, n_2
n_1 = 3
n_2 = 5
```

In [12]:
```
# Вывожу приблеженное значение f(a),
# Заменяя функцию многочленами Тейлора порядока n_1 = 3
print(polinom_Taylor_n_order(a, n_1))
```

```
0.9480382290420832
```

In [13]:
```
# Вывожу приблеженное значение f(a),
# Заменяя функцию многочленами Тейлора порядокаn_2 = 5
print(polinom_Taylor_n_order(a, n_2))
```

```
0.9480972974023355
```

3) Сравниваем приближенные значения с точным значением. Проверяем достигнута ли требуемая точность

In [14]:
```python
# Задаю значения для требуемой точности
exactly_1 = 10**(-3)
exactly_2 = 10**(-6)

# Задаю значения n_1, n_2 (Повторно, так как мы в новой части)
n_1 = 3
n_2 = 5

# Точное значение
exactly_func = func(a)

# Приближенные значения
polinom_T_1 = polinom_Taylor_n_order(a, n_1)
polinom_T_2 = polinom_Taylor_n_order(a, n_2)
```

In [15]:
```python
# Вывожу значения и разность между ними
print("Вывод:")

print(f"Точное значение = {exactly_func}")

print(f"Приближенное значение от n_1 = {polinom_T_1}; "
      f"Отличие с точным значение = {abs(exactly_func - polinom_T_1)}")

print(f"Приближенное значение от n_2 = {polinom_T_2}; "
      f"Отличие с точным значение = {abs(exactly_func - polinom_T_2)}")
```

```
Вывод:
Точное значение = 0.9480972192081248
Приближенное значение от n_1 = 0.9480382290420832; Отличие с точным значение = 5.8990166
041605896e-05
Приближенное значение от n_2 = 0.9480972974023355; Отличие с точным значение = 7.8194210
66201926e-08
```

In [16]:
```python
# Проверяю достигнута ли требуемая точность для n_1
print("Вывод:")

if (abs(exactly_func - polinom_T_1) < exactly_1):
    print("Требуемая точность_1 (10^-3) для n_1 выполняется", True)
else:
    print("Требуемая точность_1 (10^-6) для n_1 не выполняется", False)
```

```
Вывод:
Требуемая точность_1 (10^-3) для n_1 выполняется True
```

In [17]:
```python
# Проверяю достигнута ли требуемая точность для n_2
print("Вывод:")

if (abs(exactly_func - polinom_T_2) < exactly_2):
    print("Требуемая точность_2 (10^-3) для n_2 выполняется", True)
else:
    print("Требуемая точность_2 (10^-6) для n_2 не выполняется", False)
```

```
Вывод:
Требуемая точность_2 (10^-3) для n_2 выполняется True
```