> *Производная - это основной инструмент математической аналитики.*

Карл Гаусс

Лабораторная работа №2

Вариант №31

Товмасян Арман М3132

Функция:

$$f(x) = \cos\left(\frac{\pi}{6} + x\right)$$

Аналитический Метод

1. Получим формулу производной n-ого порядка для $f(x)$

Для начала найдем первые 4 производные

$$f'(x) = -\sin\left(\frac{\pi}{6} + x\right) = \cos\left(\frac{\pi}{6} + x + \frac{\pi}{2}\right)$$

$$f''(x) = -\cos\left(\frac{\pi}{6} + x\right) = \cos\left(\frac{\pi}{6} + x + \pi\right)$$

$$f'''(x) = \sin\left(\frac{\pi}{6} + x\right) = \cos\left(\frac{\pi}{6} + x + \frac{3\pi}{2}\right)$$

$$f^4(x) = \cos\left(\frac{\pi}{6} + x\right) = \cos\left(\frac{\pi}{6} + x + 2\pi\right)$$

Заметим некоторую закономерность в наших действиях. Далее воспользуемся методом математической индукции

Докажем, что $f^{(n)}(x) = \cos\left(\frac{\pi}{6} + x + \frac{\pi n}{2}\right)$ индукцией по n

Пусть $n = 1$:

$$f'(x) = -\sin\left(\frac{\pi}{6} + x\right) = \cos\left(\frac{\pi}{6} + x + \frac{\pi}{2}\right)$$

Истина

Пусть для $n = m$ истина. Докажем для случая $n = m + 1$

$$f^{(m+1)}(x) = (f^{(m)}(x))' = \left(\cos\left(\frac{\pi}{6} + x + \frac{\pi m}{2}\right)\right)' =$$

$$= -\sin\left(\frac{\pi}{6} + x + \frac{\pi m}{2}\right) = \cos\left(\frac{\pi}{6} + x + \frac{\pi m}{2} + \frac{\pi}{2}\right) =$$

$$= \cos\left(\frac{\pi}{6} + x + \frac{\pi(m+1)}{2}\right)$$

Что является также истиной

---

2. Распишем многочлен Тейлора n-ого порядка по степеням x

По формуле Тейлора:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^k(x_0)\,(x - x_0)^k}{k!}$$

Для нашей функции имеем многочлен Тейлора порядка n вида:

$$P_n(x) = \sum_{k=0}^{n} \cos\left(\frac{\pi}{6} + \frac{\pi k}{2}\right) \frac{x^k}{k!}$$

---

3. Выведем многочлен Тейлора n-ого порядка и сравним с П.2

$$\cos\left(\frac{\pi}{6} + x\right) = \cos\left(\frac{\pi}{6}\right)\cos(x) - \sin\left(\frac{\pi}{6}\right)\sin(x) = \frac{\sqrt{3}}{2}\cos(x) - \frac{1}{2}\sin(x) =$$

Воспользуемся уже известным нам разложением функций $\cos$ и $\sin$

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

Однако заметим что:

$$\frac{\sqrt{3}}{2}\cdot(-1)^k=\cos\left(\frac{\pi}{6}+\frac{\pi\,(2k)}{2}\right)$$

$$-\frac{1}{2}\cdot(-1)^k=\cos\left(\frac{\pi}{6}+\frac{\pi\,(2k+1)}{2}\right)$$

$$(*)=\sum_{k=0}^{\left[\frac{n}{2}\right]}\cos\left(\frac{\pi}{6}+\frac{\pi\,(2k)}{2}\right)\cdot\frac{x^{2k}}{(2k)!}+\sum_{k=0}^{\left[\frac{n-1}{2}\right]}\cos\left(\frac{\pi}{6}+\frac{\pi\,(2k+1)}{2}\right)\cdot\frac{x^{(2k+1)}}{(2k+1)!}=$$

$$=\sum_{k=0}^{n}\cos\left(\frac{\pi}{6}+\frac{\pi k}{2}\right)\cdot\frac{x^k}{k!}$$

Получили тоже самое что и в П.2

---

4. Оценим остаточный член формулы Тейлора

Используем остаточный член формулы Тейлора в форме Лагранжа:

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

Имеем следующие данные:

$$f(x)=\cos\left(\frac{\pi}{6}+x\right)$$

$$a=-0.2\quad \Delta_1=10^{-3}\quad \Delta_2=10^{-6}\quad x_0=0\quad x=a\quad \xi\in(0,a)$$

Тогда:

$$f^{(n+1)}(\xi)=\cos\left(\frac{\pi}{6}+\xi+\frac{\pi\,(n+1)}{2}\right)$$

Оценим $|R_n(a)|$ сверху:

$$\left|\cos\left(\frac{\pi}{6}+\xi+\frac{\pi\,(n+1)}{2}\right)\cdot\frac{a^{(n+1)}}{(n+1)!}\right|=(*)$$

Учитывая что:

$$\left|\cos\left(\frac{\pi}{6}+\xi+\frac{\pi\,(n+1)}{2}\right)\right|\leq 1$$

Получим:

$$(*)\leq\frac{a^{(n+1)}}{(n+1)!}$$

Найдем такие n_1 и n_2 чтобы выполнялось неравенство для Δ_1 и Δ_2:

$$|R_{n_1}(a)|<\Delta_1$$

$$n=1\Rightarrow\frac{0.2^2}{2!}=\frac{2}{10^2}>\frac{1}{10^3}$$

$$n=2\Rightarrow\frac{0.2^3}{3!}=\frac{4}{3\cdot 10^3}>\frac{1}{10^3}$$

$$n=3\Rightarrow\frac{0.2^4}{4!}=\frac{2}{3\cdot 10^4}<\frac{1}{10^3}$$

Следовательно $n_1=3$. Найдем значение n_2

$$|R_{n_2}(a)|<\Delta_2$$

$$n=4\Rightarrow\frac{0.2^5}{5!}=\frac{4}{15\cdot 10^5}>\frac{1}{10^6}$$

$$n=5\Rightarrow\frac{0.2^6}{6!}=\frac{4}{45\cdot 10^6}<\frac{1}{10^6}$$

Следовательно $n_2=5$

Получим

$$n_1=3\quad n_2=5$$

Численный Метод

1. Построим графики $f(x)$ и многочленов Тейлора $1, 2, \ldots n_2$

In [1]:
```
import numpy as np  # Библиотека для работы с массивами чисел
import matplotlib.pyplot as plt  # Библиотека для построения графиков
%matplotlib inline
```

In [2]:
```
plt.style.use(["science", "notebook", "grid", "high-vis"])  # Стиль координатной плоскос
```

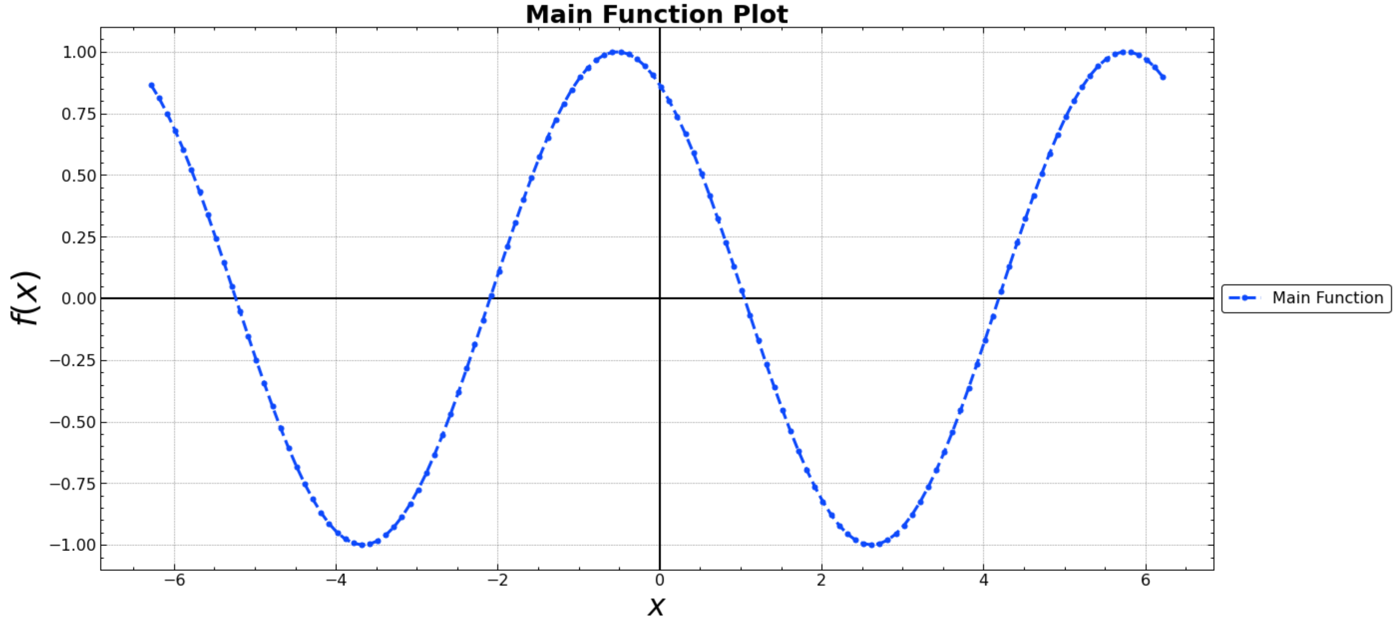
In [3]:
```
# Функция возвращающая значения по данной нам функции
def function(x):
    return np.cos((np.pi / 6) + x)
```

In [4]:
```
# Функция возвращающая n-ую производную нашей функции
def get_n_derivative(n):
    return np.cos((np.pi / 6) + ((np.pi * n) / 2))
```

In [5]:
```
# Функция возвращающая многочлен Тейлора n-ого порядка
def taylor_polynom(x, n):
    result_sum = 0
    for k in range(n + 1):
        result_sum += (1 / np.math.factorial(k)) * get_n_derivative(k) * np.power(x, k)
    return result_sum
```

In [6]:
```
X = np.arange(-2*np.pi, 2*np.pi, 0.1) # Область определения
Y = function(X) # Область значений
```

In [7]:
```
plt.figure(figsize=(20, 10))
plt.axvline(x=0, lw=2, color="black")  # Построим x=0
plt.axhline(y=0, lw=2, color="black")  # Построим y=0
plt.xlabel("$x$", fontsize=30)  # Подпишем ось абсцисс
plt.ylabel("$f(x)$", fontsize=34)  # Подпишем ось ординат
# Построим график основной функции
X_main = np.arange(-2*np.pi, 2*np.pi, 0.1) # Возьмем побольше область определения
Y_main = function(X_main)
plt.plot(X_main, Y_main, "o--", lw=3, ms=5, label="Main Function")
plt.legend(loc="center left", bbox_to_anchor=(1, 0.5), edgecolor="black")  # Отобразим л
plt.title("Main Function Plot", fontsize=25, fontweight="bold", verticalalignment="cente
plt.show()
```

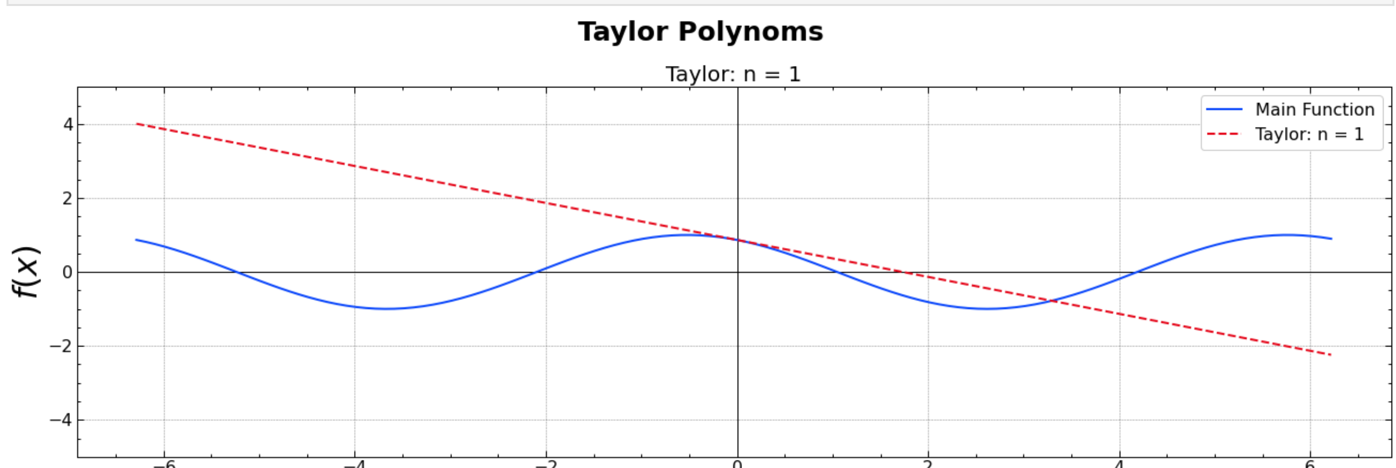


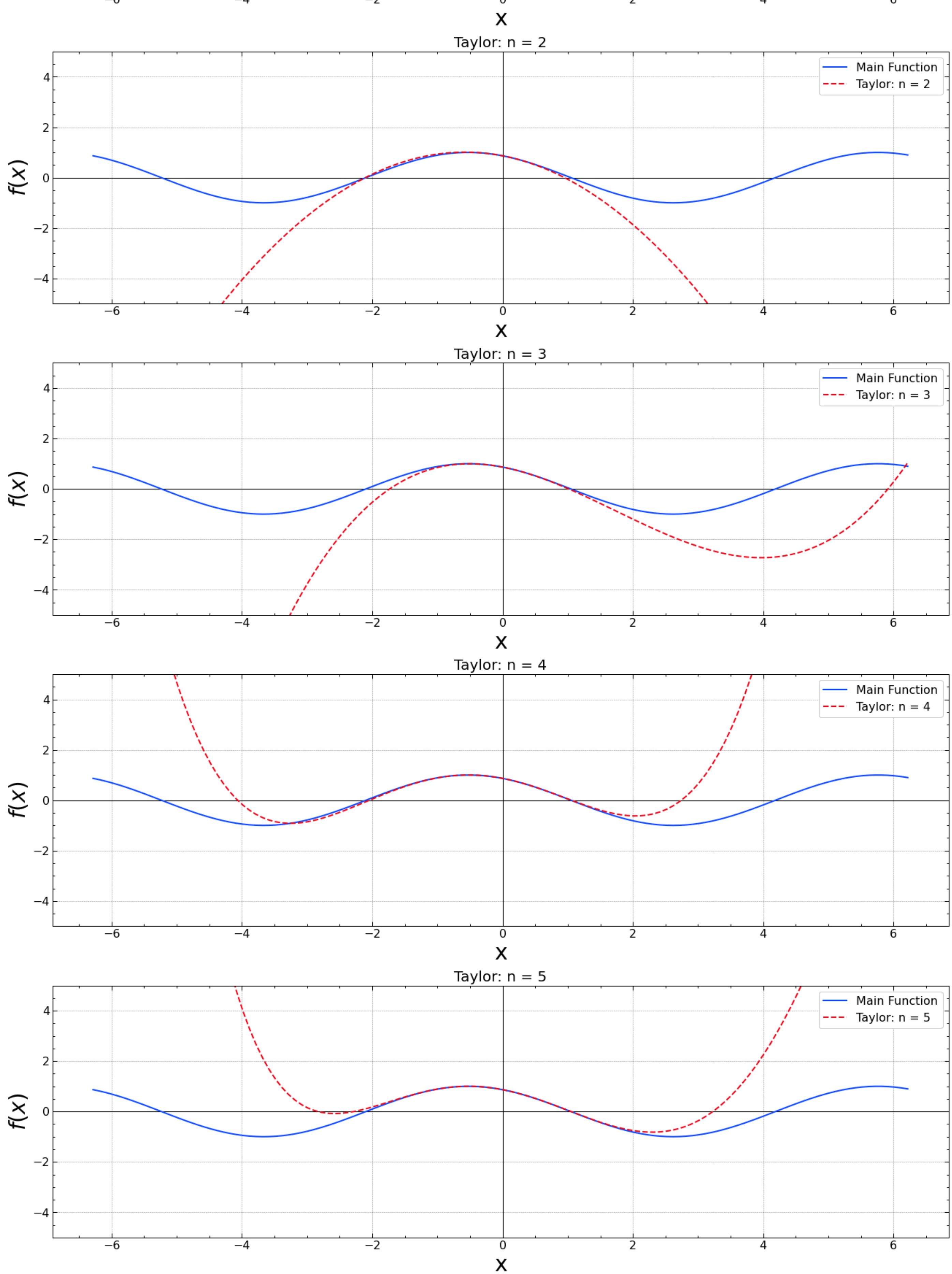
In [8]:

```python
# Функция строящая график основной функции и многочлена Тейлора n-ого порядка
def get_nth_Taylor_plot(X_i, Y_i, n, i):
    global axes
    local_X = np.arange(-2*np.pi, 2*np.pi, 0.1)
    ax_i = axes[i]
    ax_i.set_title(f"Taylor: n = {n}", fontsize=20)  # Название у графика

    ax_i.set_xlabel("x", fontsize=30)  # Лэйбл по оси Ox
    ax_i.set_ylabel(r"$f(x)$", fontsize=30)  # Лэйбл по оси Oy
    ax_i.axvline(x=0, lw=1, color="black")  # Построим x=0
    ax_i.axhline(y=0, lw=1, color="black")  # Построим y=0
    ax_i.plot(local_X, function(local_X), label="Main Function") # График основной функц
    ax_i.plot(X_i, Y_i, label=f"Taylor: n = {n}") # График многочлена Тейлора n-ого поря
    ax_i.set_ylim(-5, 5)
    ax_i.legend()
```

In [9]:

```python
fig, axes = plt.subplots(5, constrained_layout=True)
fig.suptitle("Taylor Polynoms", fontsize=25, fontweight="bold", verticalalignment="cente
fig.set_figwidth(18)
fig.set_figheight(30)

get_nth_Taylor_plot(X, taylor_polynom(X, 1), 1, 0)

get_nth_Taylor_plot(X, taylor_polynom(X, 2), 2, 1)

get_nth_Taylor_plot(X, taylor_polynom(X, 3), 3, 2)

get_nth_Taylor_plot(X, taylor_polynom(X, 4), 4, 3)

get_nth_Taylor_plot(X, taylor_polynom(X, 5), 5, 4)
```

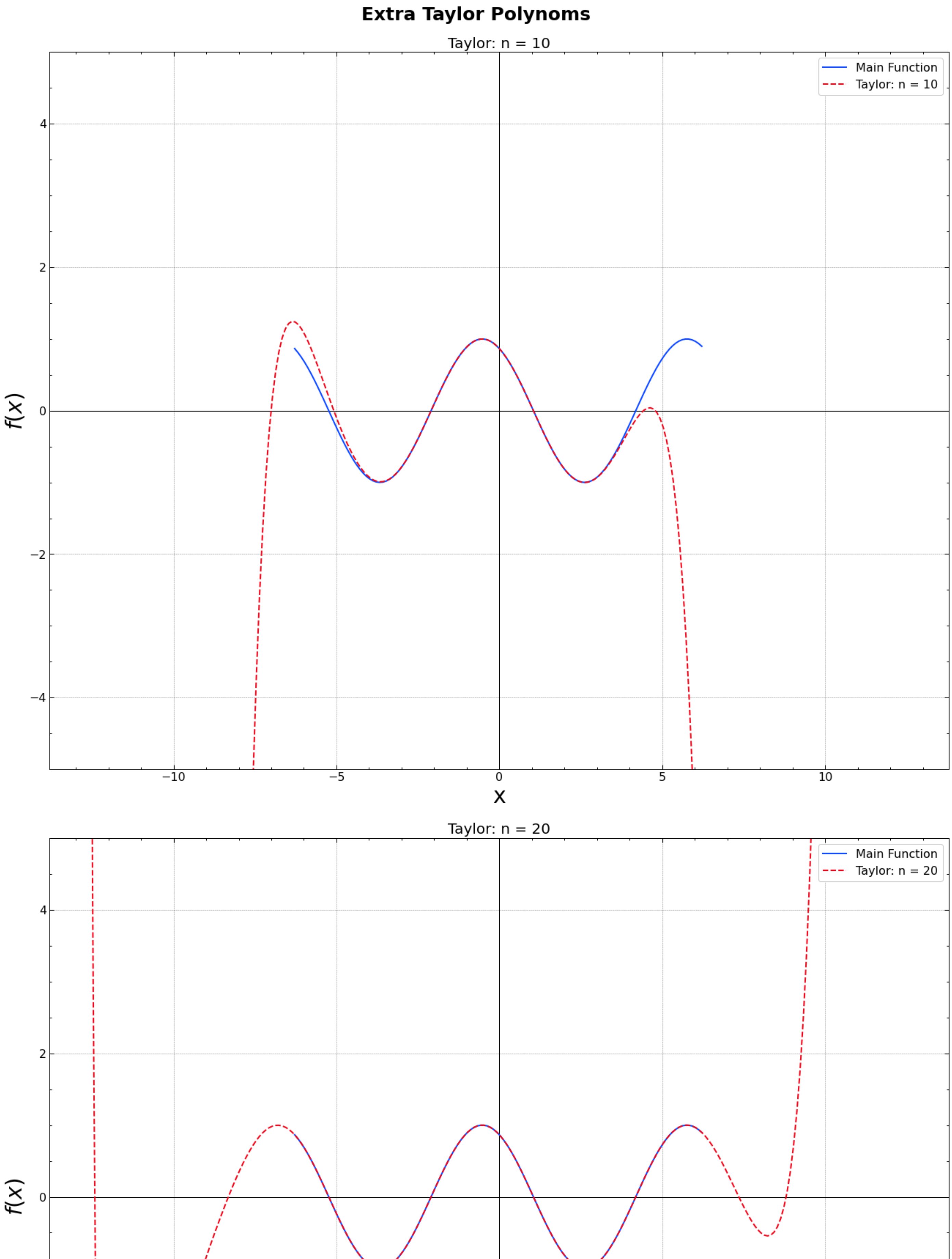


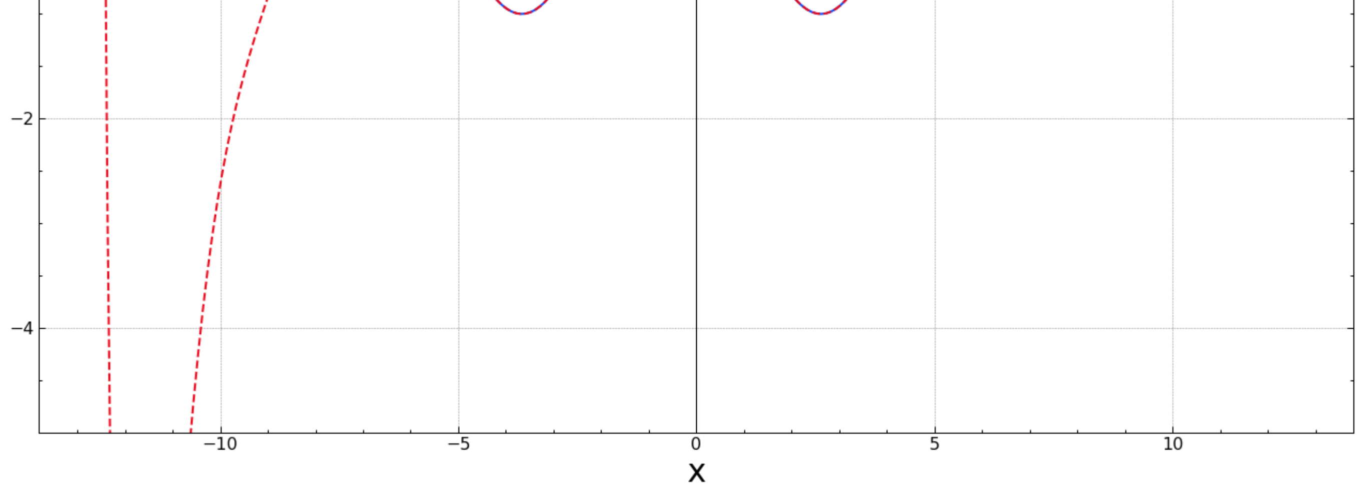


Хотелось бы нарисовать еще несколько графиков, да бы показать нагляднее мощь данного инструмента

In [10]:
```
fig, axes = plt.subplots(2, constrained_layout=True)
```

```
fig.suptitle("Extra Taylor Polynoms", fontsize=25, fontweight="bold", verticalalignment=
fig.set_figwidth(18)
fig.set_figheight(30)
X_extra = np.arange(-4*np.pi, 4*np.pi, 0.1)

get_nth_Taylor_plot(X_extra, taylor_polynom(X_extra, 10), 10, 0)

get_nth_Taylor_plot(X_extra, taylor_polynom(X_extra, 20), 20, 1)
```





2. Вычислим приближенные значения $f(a)$

In [11]:
```
a = -0.2
```

In [16]:
```
taylor_polynom(a, 3)
```

Out[16]:
```
0.9480382290420832
```

In [17]:
```
taylor_polynom(a, 5)
```

Out[17]:
```
0.9480972974023355
```

---

3. Сравним приближенные значения с точным значением (вычисленным компьютером)

Для полинома 3-ого порядка, точность относительно машинного значения достигает 5 знаков после запятой

In [12]:
```
R_n1 = abs(function(a) - taylor_polynom(a, 3))
R_n1
```

Out[12]:
```
5.8990166041605896e-05
```

Для полинома 5-ого порядка, точность относительно машинного значения достигает 8 знаков после запятой

In [13]:
```
R_n2 = abs(function(a) - taylor_polynom(a, 5))
R_n2
```

Out[13]:
```
7.819421066201926e-08
```

Убедились что требуемая точность достигнута

In [14]:
```
R_n1 < 10**(-3)
```

Out[14]:
```
True
```

In [15]:
```
R_n2 < 10**(-6)
```

```
Out[15]: True
```