Лабораторная работа №1 задание 2

Доскоч Роман вариант 9

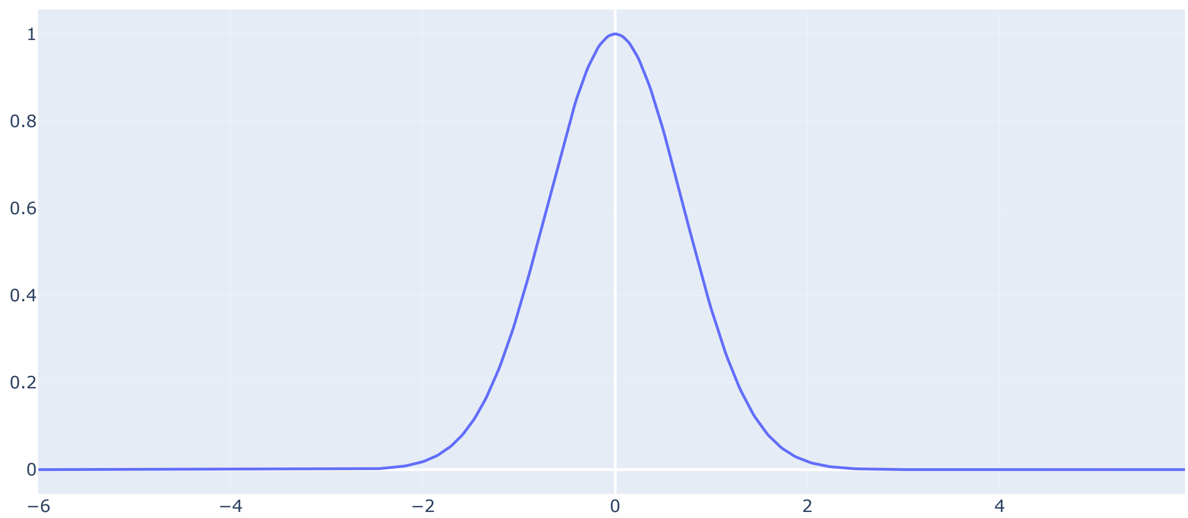
Приближение функции

$$y = e^{-x^2}, x \in [-5, 5]$$

In [1]:
```
import plotly as pl
import numpy as np
import plotly.graph_objs as go
import plotly.express as px
import numpy.random as rand
from numpy import linalg
import time
import pandas as pd
midl_sqwere_time, polinom_time=[],[]
```

In [2]:
```
f=lambda x:np.exp(-x**2)
```

In [3]:
```
x=np.linspace(-6,6,500)
fig = go.Figure(go.Scatter(x=x, y=f(x)))
fig.show()
```



Интерполяционный многочлен в форме Лагранжа по 6, 12, 18 узлам

$$P_n(x) = \sum_{i=0}^{n} y_i \prod_{i \neq j} \frac{x - x_j}{x_i - x_j}$$

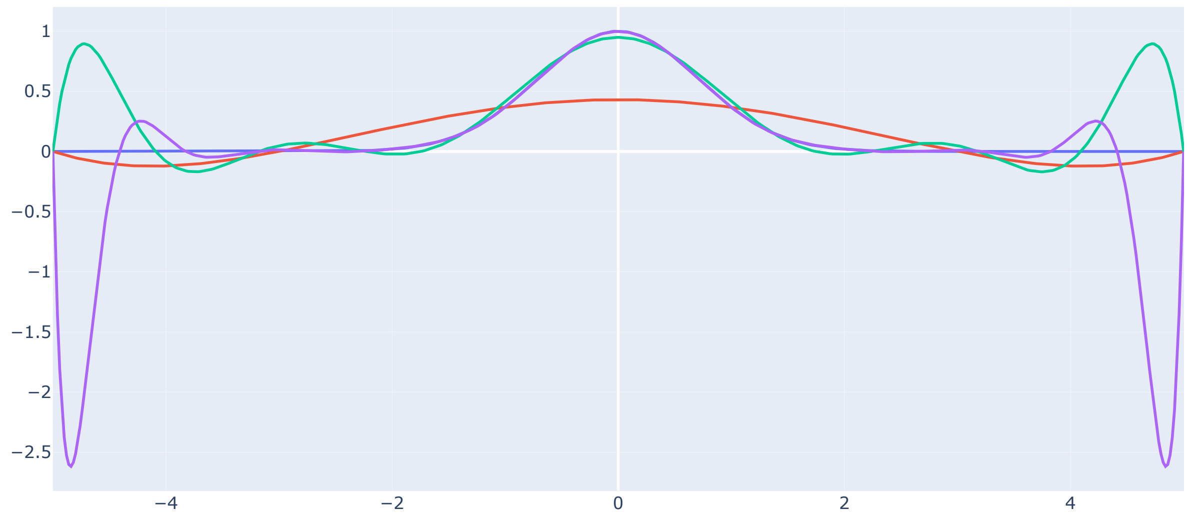
In [4]:
```
def P(x, N):
    X=np.linspace(-5,5, N)
    res=sum([L(X, x, i)*yi for i, yi in enumerate(f(X))])
    return res

def L(X, x_, i):
    return np.prod([(x_-xj)/(X[i] - xj) for j, xj in enumerate(X) if i != j])
```

In [5]:
```
def timePolinom(N):
    start = time.time()
    res=[P(i,N) for i in x]
    return res, time.time()-start
```

In [6]:
```
x=np.linspace(-5,5,500)
six, t6 =timePolinom(6)
twl, t12 =timePolinom(12)
nit, t18 =timePolinom(18)
polinom_time.append([t6,t12,t18])

fig = go.Figure(go.Scatter(x=x, y=f(x),name='real'))
fig.add_scatter(x=x, y=six,name='6')
fig.add_scatter(x=x, y=twl,name='12')
fig.add_scatter(x=x, y=nit,name='18')
fig.show()
```



Видно что с увеличением колличества узлов - растут "выбросы"(погрешность) на концах графика.

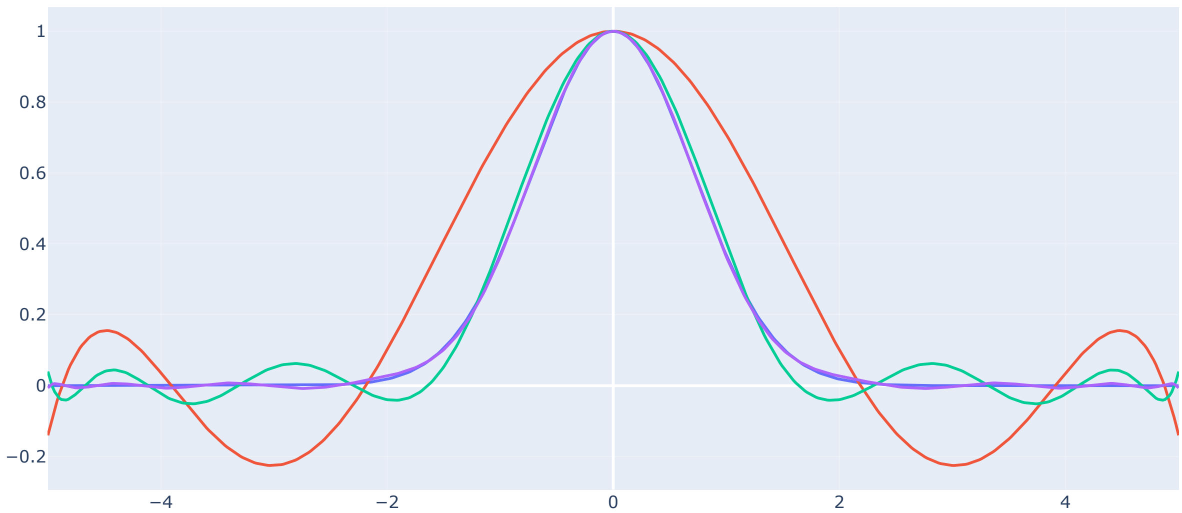
Интерполяционный многочлен в форме Лагранжа по 6, 12, 18 узлам Чебышова

In [7]:
```
def P(x, N):
    X=(a+b)/2 + (b-a)/2*np.array([np.cos(np.pi*(2*i+1)/(2*N+2)) for i in range(N+1)])
    return sum([L(X, x, i)*yi for i, yi in enumerate(f(X))])
```

In [8]:
```
def timePolinom(N):
    start = time.time()
    res=[P(i,N) for i in x]
    return res, time.time()-start
```

In [9]:
```
a,b = -5,5
x=np.linspace(a, b, 500)
six, t6 = timePolinom(6)
twl, t12 = timePolinom(12)
nit, t18 = timePolinom(18)
polinom_time.append([t6,t12,t18])

fig = go.Figure(go.Scatter(x=x, y=f(x),name='real'))
fig.add_scatter(x=x, y=six,name='6')
fig.add_scatter(x=x, y=twl,name='12')
fig.add_scatter(x=x, y=nit,name='18')
fig.show()
```



Как видно с использованием оптимальных узлов проблема прошлого подхода пропадает.

Кубический-сплайн на 6,12,18 узлах

$$S_i(x) = \alpha_i + \beta_i(x - x_i) + \frac{1}{2}\gamma_i(x - x_i)^2 + \frac{1}{6}\delta_i(x - x_i)^3$$

In [10]:
```
def method_vstr_prog(n, a, b, c, f, m=0):
        alpha, beta, mu, nu = ([0] * n for _ in range(4))
        alpha[0], beta[0] = b[0]/c[0], f[0]/c[0]
        mu[n-1],nu[n-1] = a[n-2]/c[n-1],f[n-1]/c[n-1]

        for i in range(n-2, m-1, -1):
            denom = c[i] - b[i] * mu[i+1]
            mu[i] = a[i-1] / denom
            nu[i] = (f[i] - b[i] * nu[i+1]) / denom

        y = [0] * n
        y[m] = (nu[m] - mu[m] * beta[m - 1]) / (1 - mu[m] * alpha[m-1])
        for i in range(m-1, -1, -1): y[i] = beta[i] - alpha[i] * y[i+1]
        for i in range(m, n-1): y[i+1] = nu[i+1] - mu[i+1] * y[i]

        return y
```

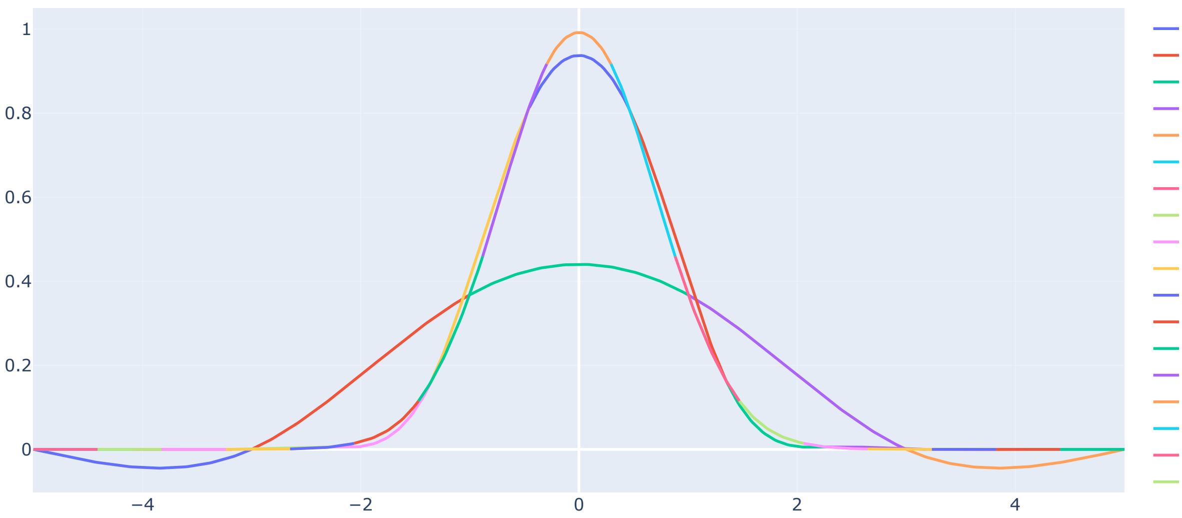
In [11]:
```
def coeffs(x, y):
    h = x[1] - x[0]
    b, g, d = [np.zeros(N) for _ in range(3)]
    c, e = [np.array([.5]*(N-3)) for _ in range(2)]
    b_ = 3*((y[2:] + y[:-2] - 2*y[1:-1])/h**2)

    g[1:-1] = method_vstr_prog(N-2, c, e, [2]*(N-2), b_)
    d[1:] = (g[1:] - g[:-1])/h
    b[1:] = (y[1:] - y[:-1])/h + (2*g[1:] + g[:-1]) / 6 * h
    return y, b, g, d
```

In [12]:
```
def Si(i, x, xi, a, b, g, d):
    return a[i] + b[i] * (x - xi) + g[i]/2 * (x - xi)**2  + d[i]/6 * (x - xi)**3
```

In [13]:
```
def show_graf(fig):
    start = time.time()
    x=np.linspace(a, b, N)
    coef=coeffs(x,f(x))
    for i in range(1,N):
        xi = np.linspace(x[i-1], x[i], int(500/N))
        fig.add_scatter(x=xi,y=Si(i, xi, x[i], *coef))
    return time.time()-start
```

In [14]:
```
fig=go.Figure()
N=6
t6=show_graf(fig)
N=12
t12=show_graf(fig)
N=18
t18=show_graf(fig)
polinom_time.append([t6,t12,t18])
fig.show()
```

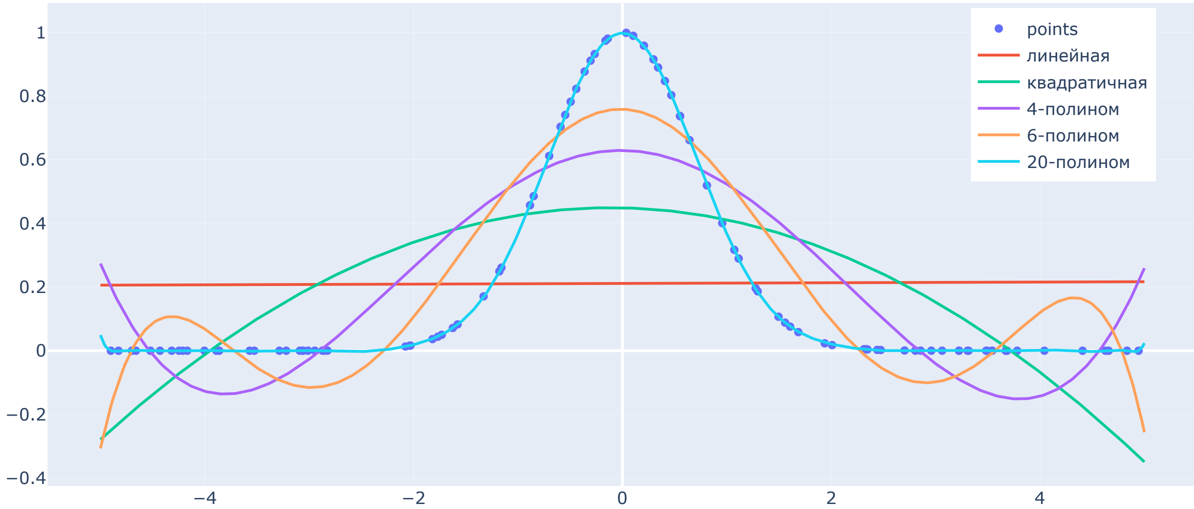


Средне Квадратичное приближение для базиса $\phi_i(x) = x^i, i = \overline{0, \mathrm{n}}$ при n = 1, 2, 4, 6.

In [15]:
```
def coefs(x, N):
    Gram=[[sum([xk**j*xk**i for xk in x]) for j in range(N+1)] for i in range(N+1)]
    b=[sum([yi*xi**i for xi,yi in zip(x,f(x))]) for i in range(N+1)]
    return np.linalg.solve(Gram, b)

def func(N):
    start=time.time()
    coef=coefs(x,N)
    res=[sum(coef*xi**np.arange(N+1)) for xi in X]
    midl_sqwere_time.append(time.time()-start)
    return res
```

In [16]:
```
x=rand.uniform(-5,5,100)
X=np.linspace(-5,5,500)
fig=go.Figure(go.Scatter(x=x,y=f(x),mode='markers', name='points'))
fig.add_scatter(x = X, y = func(1),  name = 'линейная')
fig.add_scatter(x = X, y = func(2),  name = 'квадратичная')
fig.add_scatter(x = X, y = func(4),  name = '4-полином')
fig.add_scatter(x = X, y = func(6),  name = '6-полином')
fig.add_scatter(x = X, y = func(20), name = '20-полином')
fig.update_layout(legend=dict(yanchor="top",y=0.99,xanchor="left",x=.8))
fig.show()
```



In [17]:
```
df = pd.DataFrame([['Лагранж',*polinom_time[0]],
                  ['чебышов-Лагранж',*polinom_time[1]],
                  ['Кубический Сплайн',*polinom_time[2]]]
                  , columns=['Тип', '6','12','18'])
df
```

Out[17]:

	Тип	6	12	18
0	Лагранж	0.041001	0.077000	0.135069
1	чебышов-Лагранж	0.038030	0.081999	0.124000
2	Кубический Сплайн	0.001986	0.003998	0.006002

Видно что метод чебышева немного медленне изаа дополнительного расчета оптимальных узлов. Скорость сплайнов обусловленна Алгоритмической сложностью $O(4n) = O(n)$ в то время как у ИМ. $O(n^2)$

In [18]:
```
df = pd.DataFrame([['Среднеквадартическое приближениие',*midl_sqwere_time]],columns=['Тип', '1','2','4','6','20'])
df
```

Out[18]:

	Тип	1	2	4	6	20
0	Среднеквадартическое приближениие	0.002978	0.003999	0.004999	0.003997	0.027002

Проблема этого подхода в расчете матрицы $n * n$