МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

ОТЧЕТ

по Лабораторной работе № 2

Численная реализация оптического преобразования Фурье на основе быстрого преобразования Фурье

Вариант 21

Выполнил студент   
 группы 6402-010302D

Янкин И.Ю.

Самара 2024

1. Задание на лабораторную работу





1. Реализовать одномерное финитное преобразование Фурье с помощью применения алгоритма БПФ.
2. Построить график гауссова пучка . Здесь и далее для каждого графика следует строить отдельно графики амплитуды и фазы.
3. Убедиться в правильности реализации преобразования, подав на вход гауссов пучок – собственную функцию преобразования Фурье. На выходе тоже должен получиться гауссов пучок (построить график на правильной области определения [−𝑏̃, 𝑏̃]). Рекомендуемая входная область:
4. Реализовать финитное преобразование Фурье стандартным методом численного интегрирования (например, методом прямоугольников). Важно: необходимо вычислить интеграл для каждого дискретного значения , чтобы получить результат в виде вектора. На вход преобразования вновь следует подавать гауссов пучок.
5. Построить результаты двух разных реализаций преобразования на одном изображении (одно для амплитуды, одно для фазы) и убедиться, что они совпадают.
6. Используя первую реализацию преобразования, подать на вход световое поле, отличное от гауссова пучка, в соответствии со своим вариантом. Построить графики самого пучка и результата преобразования.
7. Рассчитать аналитически результат преобразования своего варианта поля и построить график на одной системе координат с результатом, полученным в предыдущем пункте. Аналитический расчёт должен быть помещён в отчёт. Прикрепление в отчёт фотографии или скана вывода формулы не является допустимым. В некоторых вариантах после преобразования могут появиться особые точки. Значения в них нужно вычислять отдельно, в том числе в коде программы.
8. Исследовать параметры и алгоритма БПФ. Для этого сначала необходимо варьировать , задавая при этом . Что происходит при увеличении и уменьшении ? После этого следует зафиксировать и изменять . Что происходит при увеличении ? Выводы следует поместить в таблицу 2. Подкрепить выводы графиками. Важно: число должно оставаться степенью двойки, чтобы график фазы содержал минимальное число ошибок.
9. Выполнить пункты 1-3 и 6 для двумерного случая. Графики изменятся на двумерные изображения, одномерные функции следует заменить на двумерные, равные произведению соответствующих одномерных функций. Например, гауссов пучок поменяется на . Изображение следует строить не в виде 3D-графиков, а в виде цветовой схемы (см. примеры двумерных пучков в лекциях). Рассчитать аналитически результат двумерного преобразования и нарисовать изображения его амплитуды и фазы. Двумерное преобразование следует реализовывать по предложенному выше алгоритму, а именно: применить одномерный алгоритм к строкам и столбцам матрицы. Если реализовывать алгоритм через двумерное БПФ, то добавление огромного числа нулей к матрице не будет оптимальным, сведёт на нет все преимущества БПФ, а потому будет поводом для снижения оценки за работу.
10. Результаты работы программы
    1. Графики пучка Гаусса

Изначально на входе имеем одномерную функцию Гаусса: . Графики амплитуды и фазы Гауссова пучка приведены на рисунке 1.

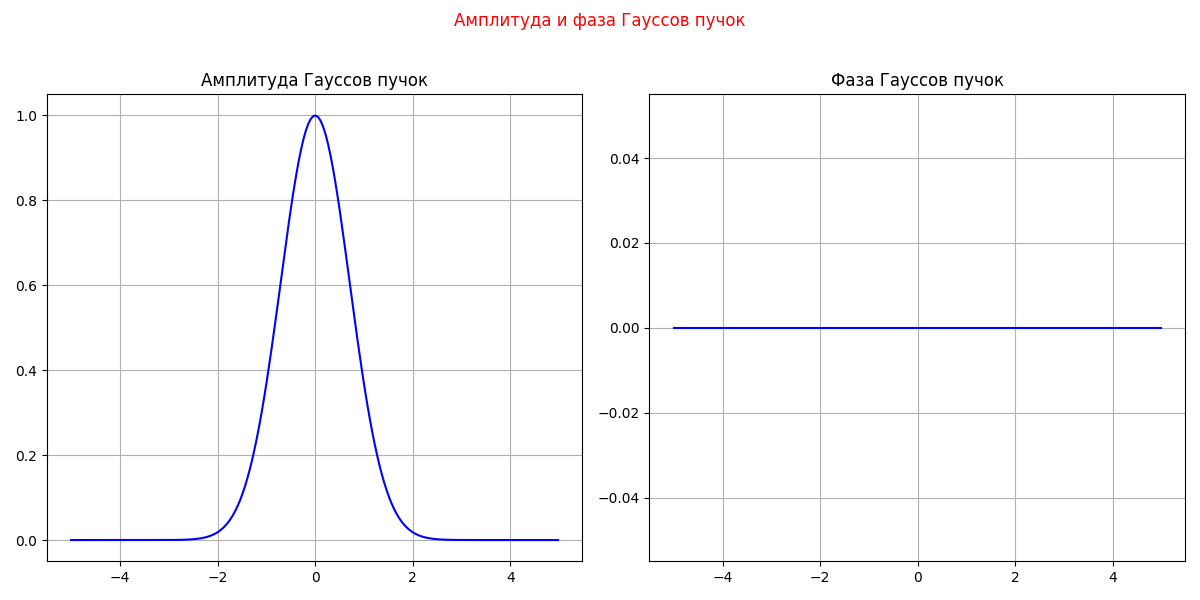


Рисунок 1 – Графики амплитуды и фазы пучка Гаусса

На рисунке 2 представлены графики амплитуды и фазы результата финитного быстрого преобразования Фурье Гауссова пучка.

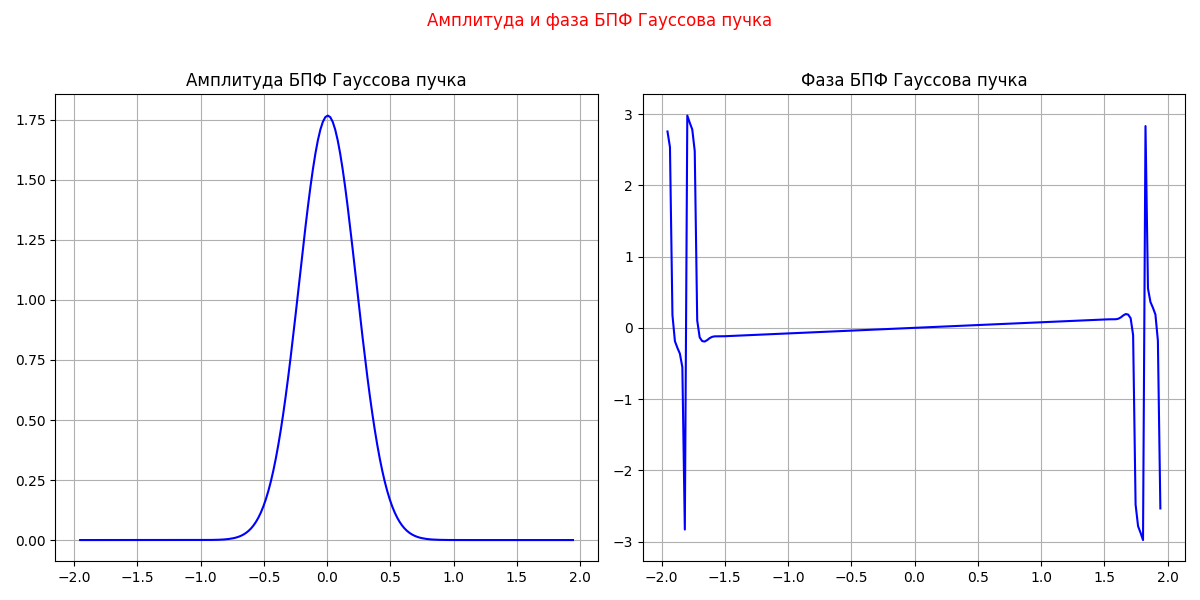


Рисунок 2 – Графики амплитуды и фазы БПФ пучка Гаусса

Чтобы убедиться в правильности реализации преобразования, необходимо выполнить преобразование Фурье методом численного интегрирования и сравнить графики результатов. Результаты преобразования представлены на рисунке 3.

Можно заметить, что графики БПФ и метода численного интегрирования полностью совпадают.

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

Рисунок 3 – Амплитуда и фаза пучка Гаусса в результате применения преобразования Фурье методом численного интегрирования

* 1. Функция согласно варианту

На рисунке 4 представлены амплитуда и фаза входного светового поля,

согласно варианту:

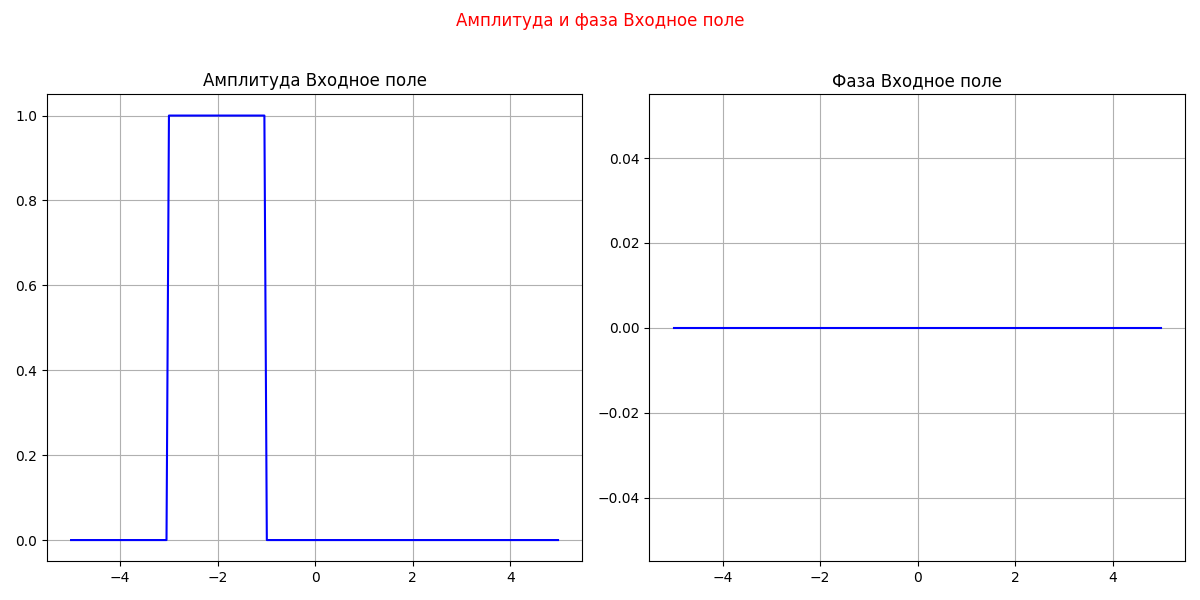


Рисунок 4 – Амплитуда и фаза входного поля

На рисунке 5 представлены графики амплитуды и фазы входного поля в результате применения быстрого преобразования Фурье.

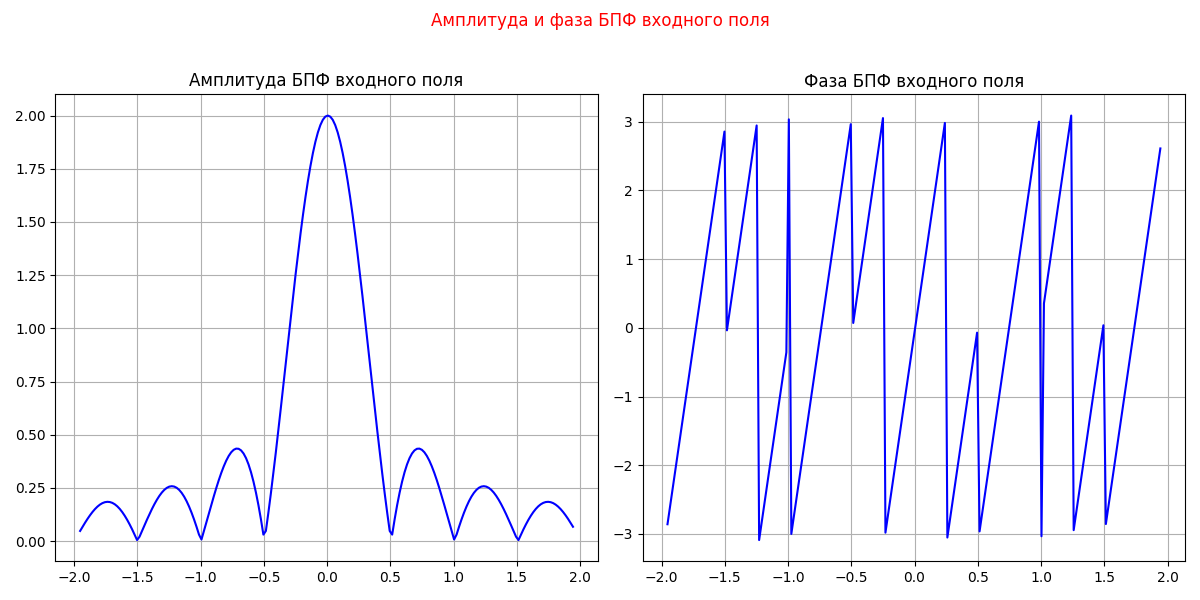


Рисунок 5 – Амплитуда и фаза входного поля в результате применения БПФ

Далее следует рассчитать аналитически результат преобразования своего варианта поля и построить график. Аналитически данное поле рассчитывается следующим образом:

Сравним графики аналитического решения и БПФ для входного сигнала Результаты представлены на рисунке 6.

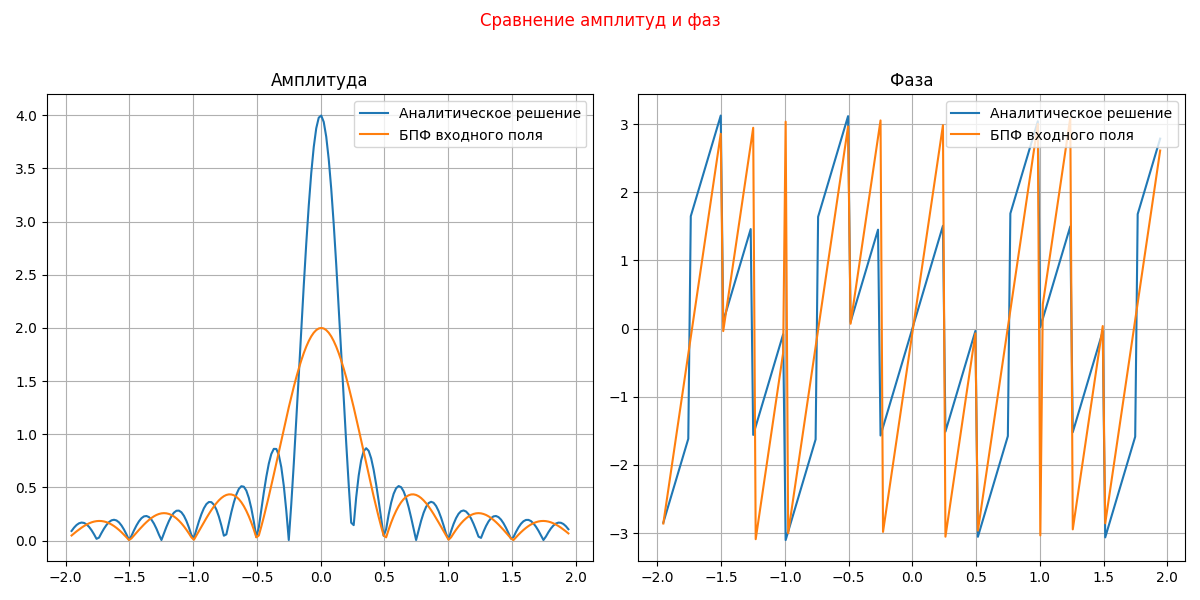


Рисунок 6 – Сравнение графиков амплитуды и фазы входного поля для БПФ и аналитического решений

* 1. Исследование параметров N и M

Перейдем к исследованию параметров алгоритма БПФ, для этого будем варьировать значения . На рисунках 7-12 представлены графики БПФ пучка Гаусса при различных значениях этих параметров, а выводы о влиянии параметров на графики представлены в таблице 1.

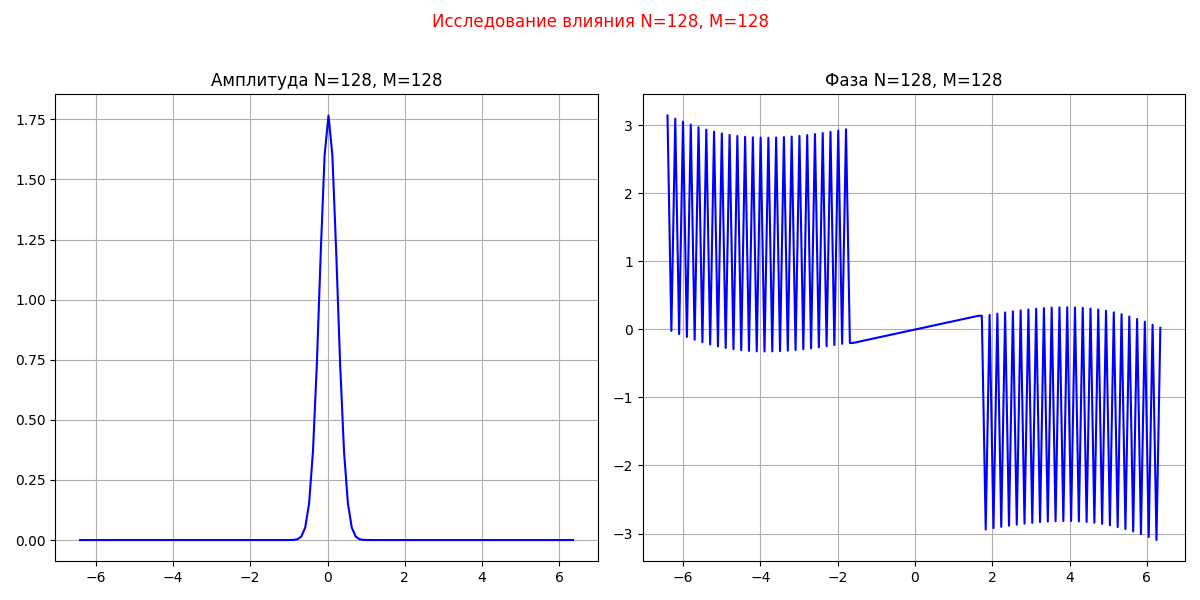


Рисунок 7 – Графики амплитуды и фазы пучка Гаусса

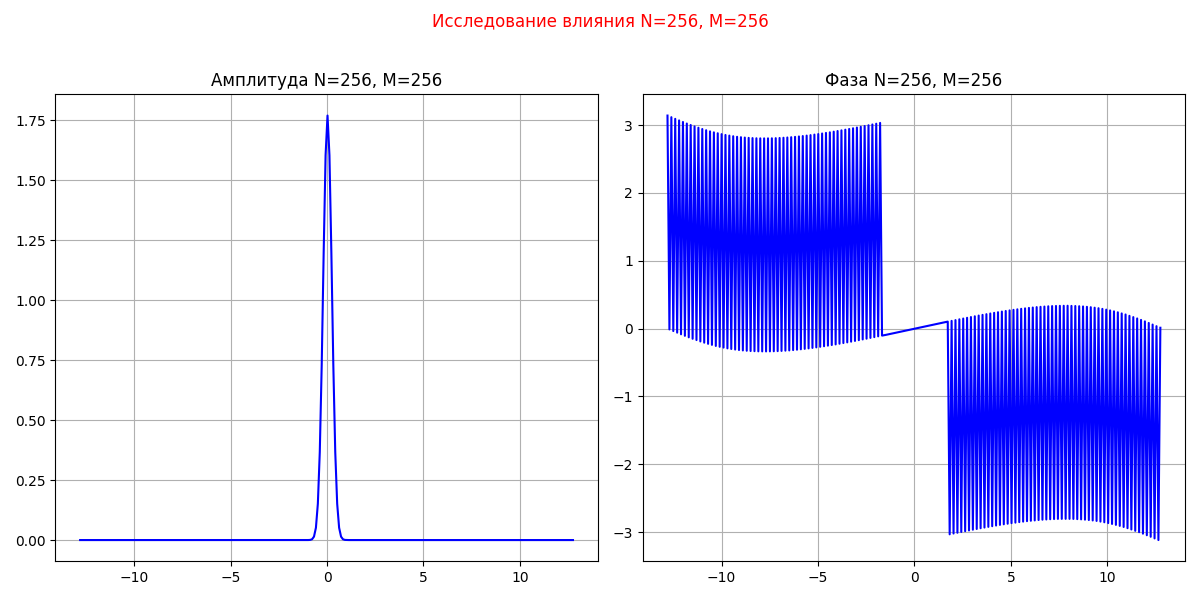


Рисунок 8 – Графики амплитуды и фазы пучка Гаусса

Рисунок 9 – Графики амплитуды и фазы пучка Гаусса

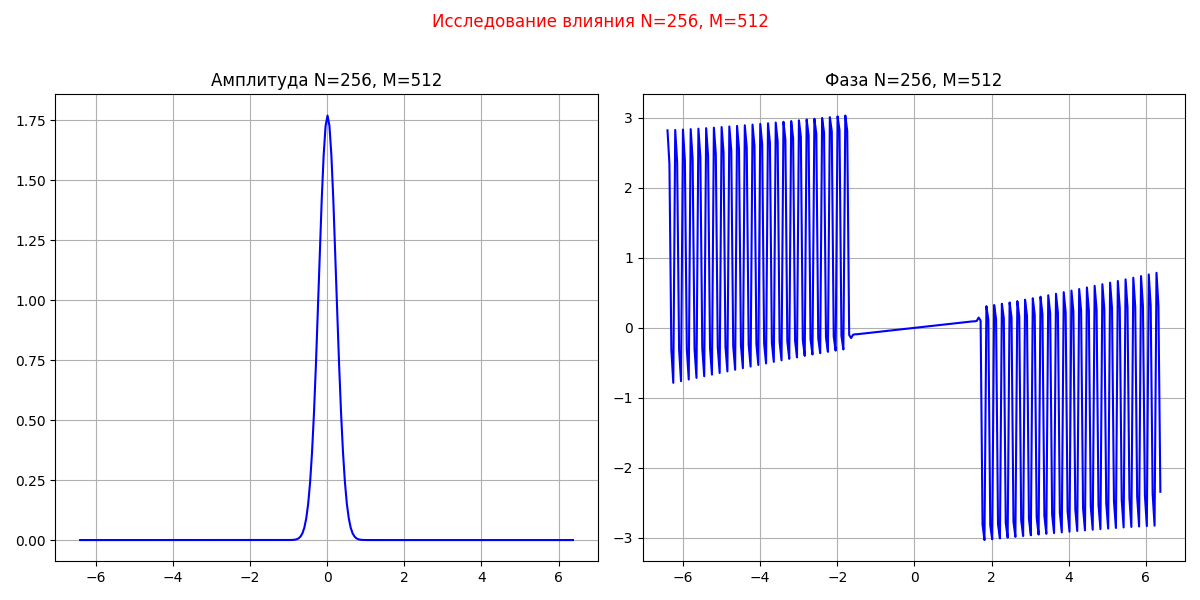
**

Рисунок 10 – Графики амплитуды и фазы пучка Гаусса

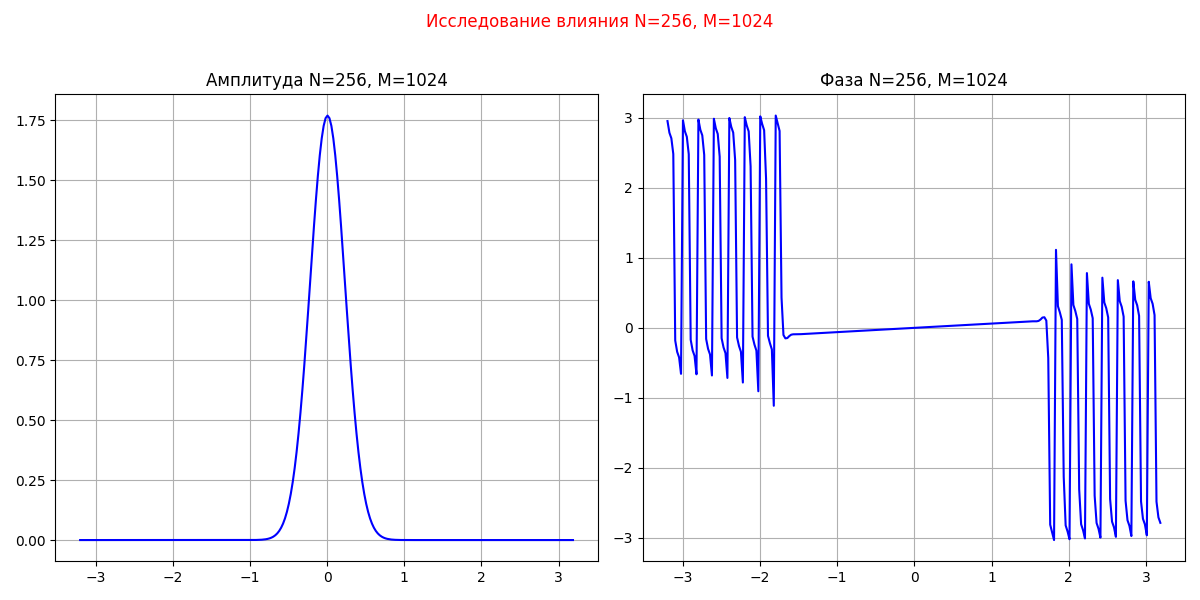
**

Рисунок 11 – Графики амплитуды и фазы пучка Гаусса

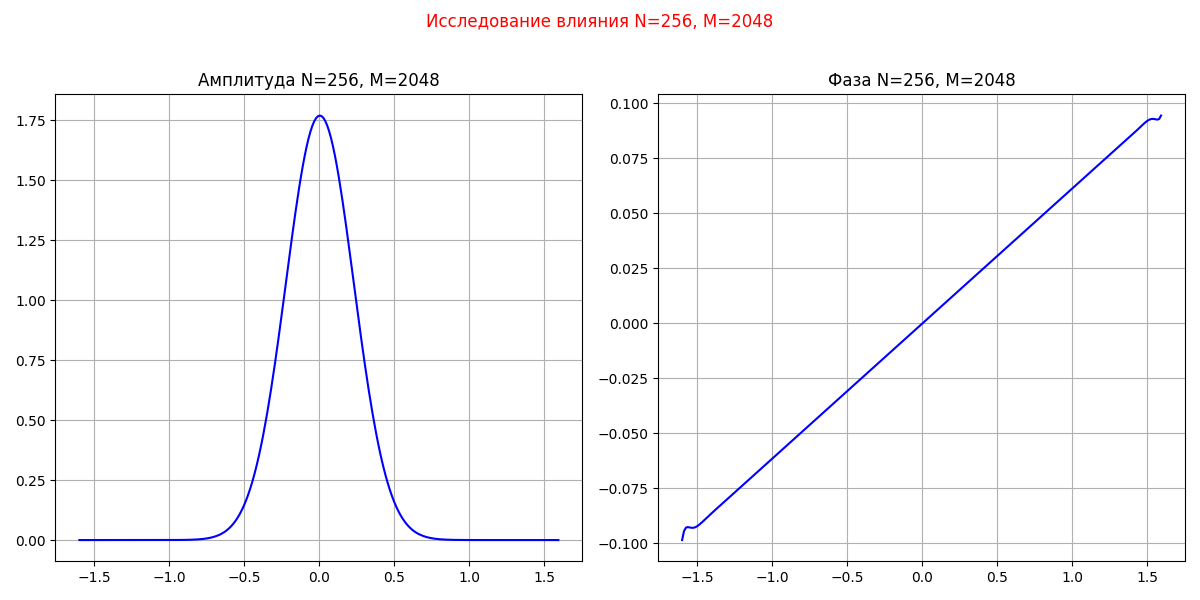
**

Рисунок 12 – Графики амплитуды и фазы пучка Гаусса

Таблица 1 – Влияние параметров на график результата преобразования

|  |  |
| --- | --- |
|  | При увеличении параметра на фиксированном промежутке увеличивается количество пиков, график становится более ломаным. Промежуток увеличивается, чем больше , тем меньше деталей будет видно на графике, при уменьшении параметра происходит обратная ситуация – график приближается, промежуток уменьшается. |
|  | При увеличении параметра график приближается, промежуток уменьшается, а при уменьшении – отдаляется, промежуток увеличивается. Чем меньше , тем меньше деталей будет видно на графике. |

1. Двумерный случай
   1. Двумерный пучок Гаусса

Рассмотрим теперь двумерный Гауссов пучок, который примет вид Графики амплитуды и фазы двумерного Гауссова пучка представлены на рисунке 13.

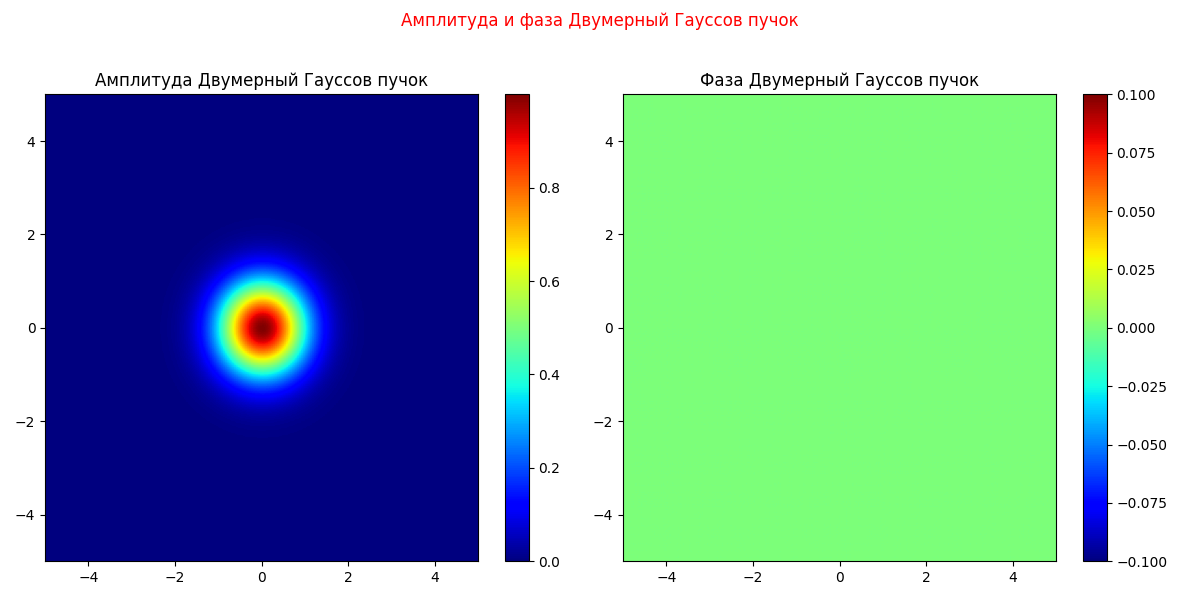


Рисунок 13 – Амплитуда и фаза двумерного Гауссова пучка

Теперь применим алгоритм быстрого преобразования Фурье для двумерного Гауссова пучка. Графики амплитуды и фазы представлены на рисунке 14.

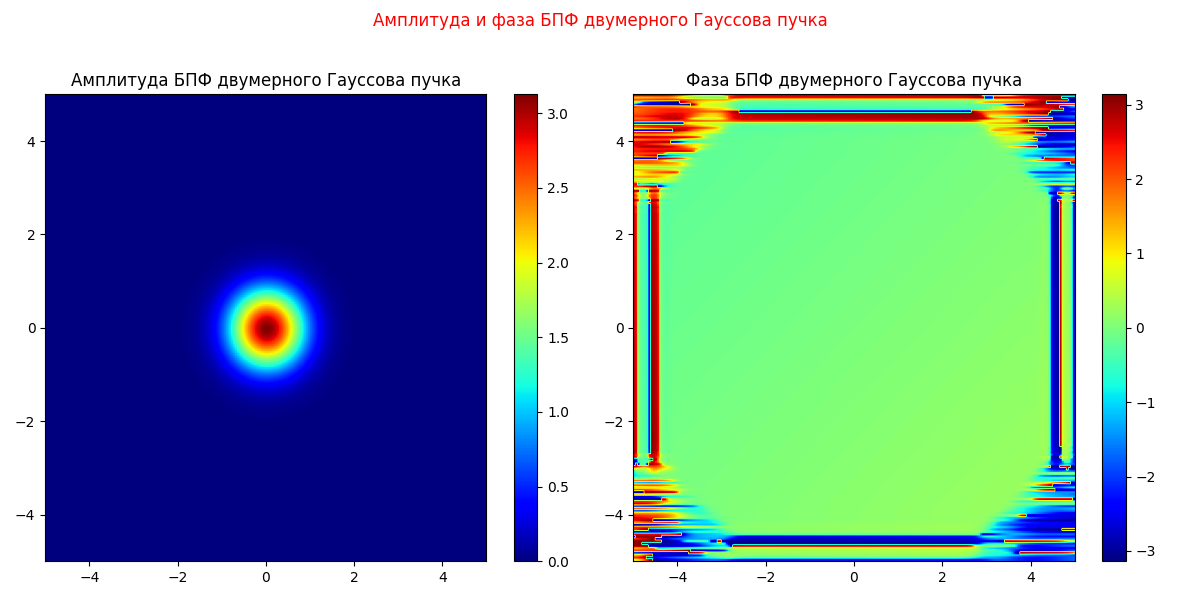


Рисунок 14 – Амплитуда и фаза БПФ двумерного Гауссова пучка

* 1. Функция согласно варианту

Также рассмотрим и двумерное входное поле, графики амплитуды и фазы которого представлены на рисунке 15. В результате применения БПФ были получены графики для амплитуды и фазы, которые представлены на рисунке 16.

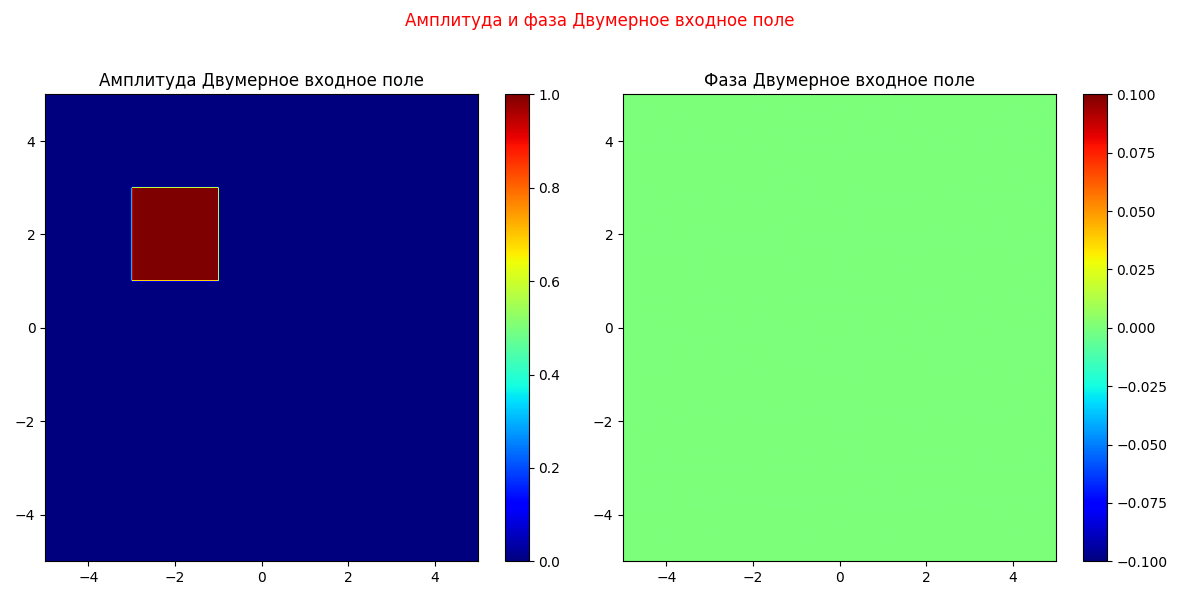


Рисунок 15 – Амплитуда и фаза двумерного входного поля

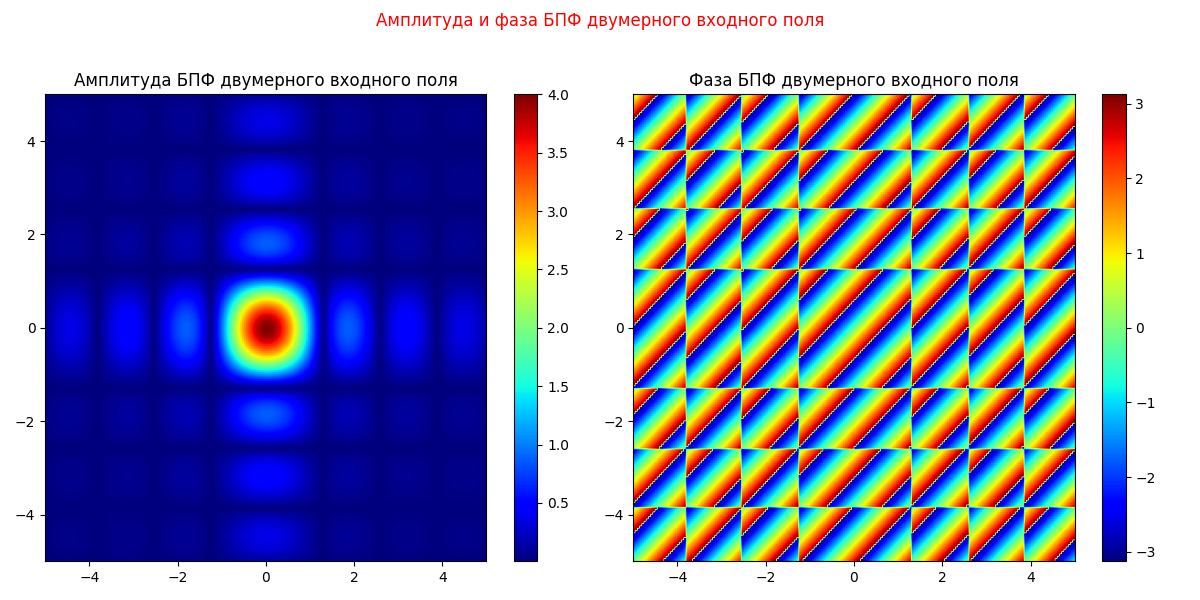


Рисунок 16 – Амплитуда и фаза БПФ двумерного входного поля

Также был рассчитан аналитически результат двумерного преобразования, который представлен на рисунке 17.

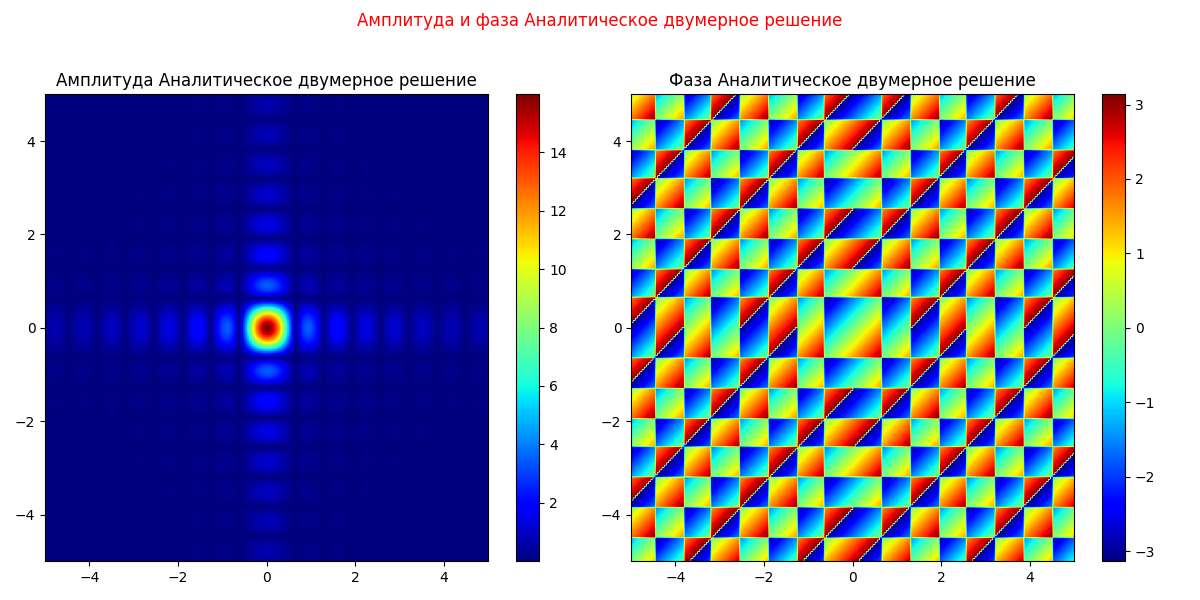


Рисунок 17 – Амплитуда и фаза двумерного входного поля (аналитическое)

1. Код программы на языке Python

from typing import List, Optional  
  
import matplotlib.pyplot as plt  
import numpy as np  
from numpy.typing import NDArray  
  
  
def gauss\_beam(x: NDArray[np.float64]) -> NDArray[np.float64]:  
 *"""Гауссов пучок"""* return np.exp(-x \*\* 2)  
  
  
def plot\_signal(x: NDArray[np.float64], signal: NDArray[np.complex128], title: str, phase: bool = False) -> None:  
 *"""График сигнала"""* plt.figure()  
 data = np.angle(signal) if phase else np.abs(signal)  
 plt.plot(x, data)  
 plt.title(title)  
 plt.grid(True)  
 plt.tight\_layout()  
 plt.show()  
  
  
def fft\_process(signal: NDArray[np.float64], M: int, N: int, hx: float) -> NDArray[np.complex128]:  
 *"""Обработка сигнала через БПФ"""* padded\_signal = np.pad(signal, pad\_width=(M - N) // 2, mode='constant')  
 FFT\_signal = np.fft.fftshift(np.fft.fft(np.fft.fftshift(padded\_signal))) \* hx  
 return FFT\_signal[M // 2 - N // 2:M // 2 + N // 2]  
  
  
def fourier\_integral(x: NDArray[np.float64], u: NDArray[np.float64], signal: NDArray[np.float64], hx: float) -> NDArray[  
 np.complex128]:  
 *"""Преобразование Фурье через интеграл"""* X, U = np.meshgrid(x, u)  
 Kernel = np.exp(-2j \* np.pi \* X \* U)  
 return Kernel @ signal \* hx  
  
  
def finite\_fourier\_integral(f\_x, x, u, hx):  
 F\_u = np.zeros(len(u), dtype=complex)  
 for i in range(len(u)):  
 F\_u[i] = np.sum(f\_x \* np.exp(-2j \* np.pi \* u[i] \* x)) \* hx # Метод прямоугольников  
 return F\_u  
  
  
def input\_field(x: NDArray[np.float64]) -> NDArray[np.float64]:  
 *"""Входное поле rect((x + 2) / 2)"""* return np.where(np.abs(x + 2) < 1, 1, 0)  
  
  
def analytical\_solution(u: NDArray[np.float64]) -> NDArray[np.float64]:  
 *"""Аналитическое решение"""* return 4 \* np.exp(2 \* 1j \* np.pi \* u) \* np.sinc(4 \* u)  
  
  
def fft\_2d\_process(field: NDArray[np.float64], M: int, N: int, hx: float) -> NDArray[np.complex128]:  
 *"""Обработка двумерного поля через БПФ"""* field = np.apply\_along\_axis(fft\_process, axis=0, arr=field, M=M, N=N, hx=hx)  
 field = np.apply\_along\_axis(fft\_process, axis=1, arr=field, M=M, N=N, hx=hx)  
 return field  
  
  
def plot\_2d\_results(field: NDArray[np.complex128], title: str, extent: Optional[List[float]] = None) -> None:  
 *"""Построение двумерных графиков амплитуды и фазы"""* extent = extent if extent else (-5, 5, -5, 5)  
 fig, axes = plt.subplots(1, 2, figsize=(12, 6))  
  
 axes[0].imshow(np.abs(field), extent=extent, cmap='jet', aspect='auto')  
 axes[0].set\_title(f'Амплитуда {title}')  
 axes[0].figure.colorbar(axes[0].images[0], ax=axes[0])  
  
 axes[1].imshow(np.angle(field), extent=extent, cmap='jet', aspect='auto')  
 axes[1].set\_title(f'Фаза {title}')  
 axes[1].figure.colorbar(axes[1].images[0], ax=axes[1])  
  
 fig.suptitle(f'Амплитуда и фаза {title}', color='r')  
 plt.tight\_layout(rect=(0, 0, 1, 0.96))  
 plt.show()  
  
  
def plot\_results(x: np.ndarray, signal: np.ndarray, title: str) -> None:  
 *"""Построение графиков амплитуды и фазы"""* fig, axes = plt.subplots(1, 2, figsize=(12, 6))  
  
 axes[0].plot(x, np.abs(signal), color='b')  
 axes[0].set\_title(f'Амплитуда {title}')  
 axes[0].grid(True)  
  
 axes[1].plot(x, np.angle(signal), color='b')  
 axes[1].set\_title(f'Фаза {title}')  
 axes[1].grid(True)  
  
 fig.suptitle(f'Амплитуда и фаза {title}', color='r')  
 plt.tight\_layout(rect=(0, 0, 1, 0.96))  
 plt.show()  
  
  
def plot\_results\_multiple(x: np.ndarray, signals: List[NDArray[np.complex128]], titles: List[str]) -> None:  
 *"""Построение графиков амплитуды и фазы для нескольких сигналов"""* fig, axes = plt.subplots(1, 2, figsize=(12, 6))  
  
 for signal, label in zip(signals, titles):  
 axes[0].plot(x, np.abs(signal), label=label)  
 axes[1].plot(x, np.angle(signal), label=label)  
  
 axes[0].set\_title('Амплитуда')  
 axes[1].set\_title('Фаза')  
  
 for ax in axes:  
 ax.grid(True)  
 ax.legend()  
  
 fig.suptitle('Сравнение амплитуд и фаз', color='r')  
 plt.tight\_layout(rect=(0, 0, 1, 0.96))  
 plt.show()  
  
  
def experiment\_change\_N\_M(signal\_func, a: float) -> None:  
 N\_values = np.array([128, 256, 512])  
 M\_values = [512, 1024, 2048]  
  
 for N in N\_values:  
 b = N / (4 \* a)  
 hx = (2 \* a) / N  
 hu = (2 \* b) / N  
 x = np.linspace(-a, a - hx / 2, N)  
 u = np.linspace(-b, b - hu / 2, N)  
  
 signal = signal\_func(x)  
 FFT\_signal = fft\_process(signal, N, N, hx)  
  
 plot\_fft\_results(u, FFT\_signal, N, N)  
  
 N = 256  
 for M in M\_values:  
 b = (N \*\* 2) / (4 \* a \* M)  
 hx = (2 \* a) / N  
 hu = (2 \* b) / N  
 x = np.linspace(-a, a - hx / 2, N)  
 u = np.linspace(-b, b - hu / 2, N)  
  
 signal = signal\_func(x)  
 FFT\_signal = fft\_process(signal, M, N, hx)  
  
 plot\_fft\_results(u, FFT\_signal, N, M)  
  
  
def plot\_fft\_results(u, FFT\_signal, N, M):  
 plt.figure(figsize=(12, 6))  
  
 plt.subplot(1, 2, 1)  
 plt.title(f'Амплитуда N={N}, M={M}')  
 plt.plot(u, np.abs(FFT\_signal), color='b')  
 plt.grid(True)  
  
 plt.subplot(1, 2, 2)  
 plt.title(f'Фаза N={N}, M={M}')  
 plt.plot(u, np.angle(FFT\_signal), color='b')  
 plt.grid(True)  
  
 plt.suptitle(f'Исследование влияния N={N}, M={M}', color='r')  
 plt.tight\_layout(rect=(0, 0, 1, 0.96))  
 plt.show()  
  
  
def main() -> None:  
 M, N = 1024, 200  
 a = 5  
 b = (N \*\* 2) / (4 \* a \* M)  
 hx = (2 \* a) / N  
 hu = (2 \* b) / N  
 x = np.linspace(-a, a - hx / 2, N)  
 u = np.linspace(-b, b - hu / 2, N)  
  
 # Гауссов пучок  
 gauss = gauss\_beam(x)  
 plot\_results(x, gauss, 'Гауссов пучок')  
  
 FFT\_gauss = fft\_process(gauss, M, N, hx)  
 plot\_results(u, FFT\_gauss, 'БПФ Гауссова пучка')  
  
 G = fourier\_integral(x, u, gauss, hx)  
 plot\_results(u, G, 'ПФ Гауссова пучка')  
  
 # Входное поле  
 f = input\_field(x)  
 plot\_results(x, f, 'Входное поле')  
  
 # БПФ входного поля  
 FFT2 = fft\_process(f, M, N, hx)  
 plot\_results(u, FFT2, 'БПФ входного поля')  
  
 # Входное поле - аналитическое решение  
 FA = analytical\_solution(u)  
 # plot\_results\_multiple(u, [FA, FFT2], ['Аналитическое решение', 'БПФ входного поля'])  
  
 # Двумерный гауссов пучок  
 X, Y = np.meshgrid(x, x)  
 gauss2 = np.exp(-(X \*\* 2 + Y \*\* 2))  
 plot\_2d\_results(gauss2, 'Двумерный Гауссов пучок')  
  
 # БПФ двумерного гауссова пучка  
 gauss2 = fft\_2d\_process(gauss2, M, N, hx)  
 plot\_2d\_results(gauss2, 'БПФ двумерного Гауссова пучка')  
  
 # Двумерное входное поле  
 f2 = np.zeros\_like(X)  
 f2[(np.abs(X + 2) < 1) & (np.abs(Y + 2) < 1)] = 1  
 plot\_2d\_results(f2, 'Двумерное входное поле')  
  
 # БПФ двумерного входного поля  
 f2 = fft\_2d\_process(f2, M, N, hx)  
 plot\_2d\_results(f2, 'БПФ двумерного входного поля')  
  
 # Аналитическое двумерное решение  
 FA2 = np.outer(FA, FA)  
 plot\_2d\_results(FA2, 'Аналитическое двумерное решение')  
  
 experiment\_change\_N\_M(gauss\_beam, a)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()