

lecture03

October 2, 2026

1 Лекция 3. Дискретное преобразование Фурье (ДПФ)

Курс: Обработка сигналов · 4 курс · 01.03.02 «Прикладная математика и информатика»

Направленность: искусственный интеллект и системная аналитика

На лекции 2 мы поняли, что любой сигнал собирается из синусоид, а спектр — «список деталей». Но спектр мы рисовали, а не считали: непрерывное преобразование Фурье требует интеграла, а у компьютера есть только **N дискретных отсчётов**. Сегодня строим мост — **дискретное преобразование Фурье (ДПФ)**: точный спектр в конечном числе точек. Как и просили, даём **две стороны**: инженерную (что это значит на практике) и **математическую** (строгие определения, свойства, матричная форма).

1.1 Цели лекции

К концу лекции вы сможете:

1. Записать **прямое и обратное ДПФ** формулами и объяснить их смысл.
2. Связать ДПФ с непрерывным спектром: частоты $f_k = k \cdot f_d / N$, периодичность.
3. Сформулировать **свойства ДПФ**: линейность, симметрия, сдвиг, круговую свёртку, Парсеваль.
4. Записать ДПФ в **матричной форме** $X = W x$.
5. Считать ДПФ на Python и правильно строить ось частот.
6. Понять ограничение $O(N^2)$ и мотив к БПФ.

1.2 1. Зачем нужно ещё одно преобразование

- Непрерывный спектр (лекция 2) — это **интеграл**; компьютер его не берёт.
- На практике мы имеем **N отсчётов** $x[0], \dots, x[N-1]$.
- **ДПФ** — линейное преобразование, которое из N чисел делает N чисел: спектр, посчитанный точно в N точках.
- Всё остальное (БПФ, окна, спектрограммы) — надстройки над ДПФ.

1.3 2. Математическое представление

Соглашение: индексы $n = 0 \dots N-1$, экспонента $w_N = e^{-j 2\pi / N}$. Здесь $j = \sqrt{-1}$ — **инженерное обозначение мнимой единицы**: в электротехнике буква i занята током, поэтому мнимую единицу пишут j .

Прямое ДПФ (сигнал → спектр):

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, \quad k = 0, \dots, N-1.$$

Обратное ДПФ (спектр → сигнал):

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}.$$

Смысл: ДПФ раскладывает вектор x по N комплексным экспонентам — **ортogonalьному базису**. Коэффициент $X[k]$ показывает, сколько в сигнале составляющей с частотой $f_k = k \cdot f_{\text{д}}/N$.

Матричная форма:

$$\mathbf{X} = \mathbf{W} \mathbf{x}, \quad W_{kn} = W_N^{kn}, \quad \mathbf{x} = \frac{1}{N} \mathbf{W}^H \mathbf{X}.$$

Обратите внимание: $\mathbf{W}$ — симметричная матрица Вандермонда из W_N .

```
[1]: %matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path

FIG = Path("figures")
FIG.mkdir(exist_ok=True)

plt.rcParams.update({
    "figure.figsize": (11, 3.6),
    "figure.dpi": 110,
    "font.size": 11,
    "axes.grid": True,
    "grid.alpha": 0.3,
})

def save(fig, name):
    fig.savefig(FIG / name, bbox_inches="tight")
    print("сохранено:", name)

print("numpy", np.__version__)
```

numpy 2.5.3

1.3.1 Что такое $e^{-j\theta}$: единичная окружность

Формула Эйлера: $e^{j\theta} = \cos \theta + j \sin \theta$ — это комплексное число. На комплексной плоскости оно даёт **единичный вектор, повёрнутый на угол θ** : его проекция на ось Re равна $\cos \theta$, на ось Im — $\sin \theta$. Умножение на $e^{j\theta}$ — поворот против часовой стрелки на θ ; на $e^{-j\theta}$ — по часовой.

Поэтому $W_N^k = e^{-j2\pi k/N}$ — это N равномерно расставленных точек («стрелок») на окружности: базис ДПФ.

```
[2]: fig, ax = plt.subplots(1, 2, figsize=(13, 4.4))

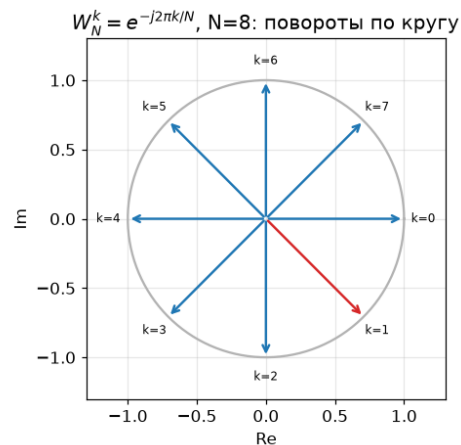
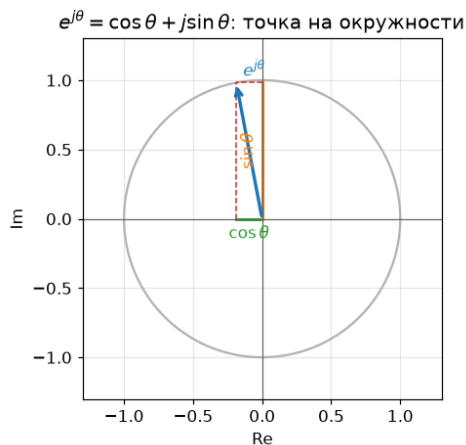
th = np.linspace(0, 2 * np.pi, 400)

t0 = 2 * np.pi * 0.28
c, s = np.cos(t0), np.sin(t0)
ax[0].plot(np.cos(th), np.sin(th), color="0.7")
ax[0].plot([0, c], [0, 0], color="C2", lw=2)
ax[0].plot([0, 0], [0, s], color="C1", lw=2)
ax[0].plot([c, c], [0, s], "C3--", lw=1)
ax[0].plot([0, c], [s, s], "C3--", lw=1)
ax[0].annotate("", xy=(c, s), xytext=(0, 0),
               arrowprops=dict(arrowstyle="->", color="C0", lw=2.2))
ax[0].text(c / 2, -0.14, r"$\cos\theta$", color="C2", ha="center")
ax[0].text(-0.16, s / 2, r"$\sin\theta$", color="C1", va="center", rotation=90)
ax[0].text(c + 0.05, s + 0.05, r"$e^{j\theta}$", color="C0")
ax[0].axhline(0, color="0.4", lw=0.8)
ax[0].axvline(0, color="0.4", lw=0.8)
ax[0].set_aspect("equal")
ax[0].set_xlim(-1.3, 1.3)
ax[0].set_ylim(-1.3, 1.3)
ax[0].set_xlabel("Re")
ax[0].set_ylabel("Im")
ax[0].set_title(r"$e^{j\theta}=\cos\theta+j\sin\theta$: точка на окружности")

N = 8
ax[1].plot(np.cos(th), np.sin(th), color="0.7")
for k in range(N):
    ang = -2 * np.pi * k / N
    x, y = np.cos(ang), np.sin(ang)
    col = "C3" if k == 1 else "C0"
    ax[1].annotate("", xy=(x, y), xytext=(0, 0),
                   arrowprops=dict(arrowstyle="->", color=col, lw=1.6))
    ax[1].text(1.14 * x, 1.14 * y, f"k={k}", ha="center", va="center",
               ↵fontsize=8)
ax[1].set_aspect("equal")
ax[1].set_xlim(-1.3, 1.3)
ax[1].set_ylim(-1.3, 1.3)
```

```
ax[1].set_xlabel("Re")
ax[1].set_ylabel("Im")
ax[1].set_title(r"$W_N^k=e^{-j2\pi k/N}$, N=8: повороты по кругу")
fig.tight_layout()
save(fig, "fig_euler.png")
plt.show()
```

сохранено: fig_euler.png



Проекция = корреляция. ДПФ смотрит, насколько сигнал «совпадает» с каждой поворачивающейся стрелкой: если совпадает — проекции складываются в пик, если нет — сумма обращается в ноль. Отсюда и смысл коэффициента $X[k]$.

1.4 2b. Комплексное представление и примеры спектров

Как представить синус комплексно? По формуле Эйлера косинус — это сумма двух комплексных экспонент:

$$\cos\theta = \frac{e^{j\theta} + e^{-j\theta}}{2}.$$

Значит, реальный синус $x(t)=A\cdot\cos(2\pi f_0 t+\varphi)$ в спектре даёт **две** сопряжённые линии: на $+f_0$ и $-f_0$ (амплитуды $A/2$, фазы $\pm\varphi$). Комплексный (аналитический) сигнал $x_c[n]=A\cdot e^{j\{2\pi f_0 n/f_d+\varphi\}}$ даёт **одну** линию на $+f_0$. Проверим на сигнале 1 кГц: $A=1$, $f_0=1000$ Гц, $\varphi=0$, $f_d=8000$ Гц, $N=8$ — частота ровно в бине ($\Delta f=1000$ Гц, $k=1$).

```
[3]: fs, N = 8000.0, 8
n = np.arange(N)
A, f0, phi = 1.0, 1000.0, 0.0
x = A * np.cos(2 * np.pi * f0 * n / fs + phi)
xc = A * np.exp(1j * (2 * np.pi * f0 * n / fs + phi))
```

```

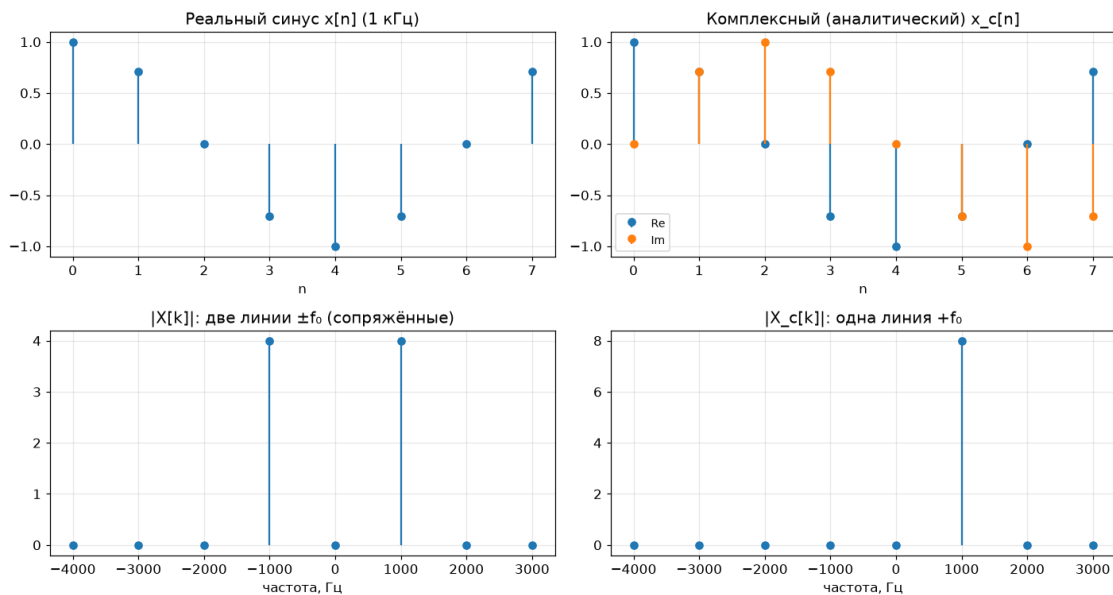
f_axis = np.fft.fftshift(np.fft.fftfreq(N, 1 / fs))
Xs = np.fft.fftshift(np.fft.fft(x))
Xcs = np.fft.fftshift(np.fft.fft(xc))

fig, ax = plt.subplots(2, 2, figsize=(13, 7))
ax[0, 0].stem(n, x, basefmt=" ")
ax[0, 0].set_title("Реальный синус x[n] (1 кГц)")
ax[0, 1].stem(n, xc.real, basefmt=" ", label="Re")
ax[0, 1].stem(n, xc.imag, basefmt=" ", linefmt="C1", markerfmt="C1o", label="Im")
ax[0, 1].set_title("Комплексный (аналитический) x_c[n]")
ax[0, 1].legend(fontsize=9)
ax[1, 0].stem(f_axis, np.abs(Xs), basefmt=" ")
ax[1, 0].set_title("|X[k]|: две линии  $\pm f_0$  (сопряжённые)")
ax[1, 1].stem(f_axis, np.abs(Xcs), basefmt=" ")
ax[1, 1].set_title("|X_c[k]|: одна линия  $+f_0$ ")
for a in ax.flatten()[0:2]:
    a.set_xlabel("n")
for a in ax.flatten()[2:]:
    a.set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_complex_sine.png")
plt.show()

print(f"A = 2 * |X[1]| / N = {2 * np.abs(np.fft.fft(x)[1]) / N:.3f}")
print(f"φ = arg X[1] = {np.angle(np.fft.fft(x)[1]):.3f} рад")

```

сохранено: fig_complex_sine.png



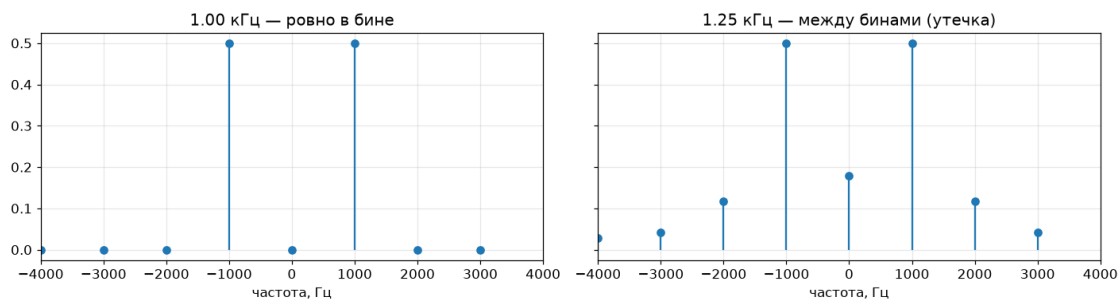
```
A = 2 * |X[1]| / N = 1.000
φ = arg X[1] = -0.000 рад
```

Утечка как контраст. Если частота **не** совпадает с бином (например, 1.25 кГц при $\Delta f = 1000$ Гц), те же две линии «размазываются» по соседним бинам — это и есть утечка из лекции 2.

```
[4]: f0b = 1250.0
xb = A * np.cos(2 * np.pi * f0b * n / fs)

fig, ax = plt.subplots(1, 2, figsize=(13, 3.6), sharey=True)
for a, sig, title in [(ax[0], x, "1.00 кГц — ровно в бине"),
                      (ax[1], xb, "1.25 кГц — между бинами (утечка)"]):
    Xs_ = np.fft.fftshift(np.fft.fft(sig))
    a.stem(f_axis, np.abs(Xs_) / N, basefmt=" ")
    a.set_xlim(-4000, 4000)
    a.set_title(title)
    a.set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_offbin_compare.png")
plt.show()
```

сохранено: fig_offbin_compare.png



Периодические сигналы кроме синуса. Прямоугольный (меандр) и треугольный сигналы содержат только **нечётные** гармоники: у меандра амплитуды падают как $1/k$, у треугольника — как $1/k^2$ (поэтому треугольник «мягче» по спектру).

```
[5]: fs, N = 8000.0, 16
n = np.arange(N)
f0 = 500.0
P = int(round(fs / f0))
sq = np.where((n % P) < P // 2, 1.0, -1.0)
tri = 2 / np.pi * np.arcsin(np.sin(2 * np.pi * f0 * n / fs))
f = np.fft.rfftfreq(N, 1 / fs)

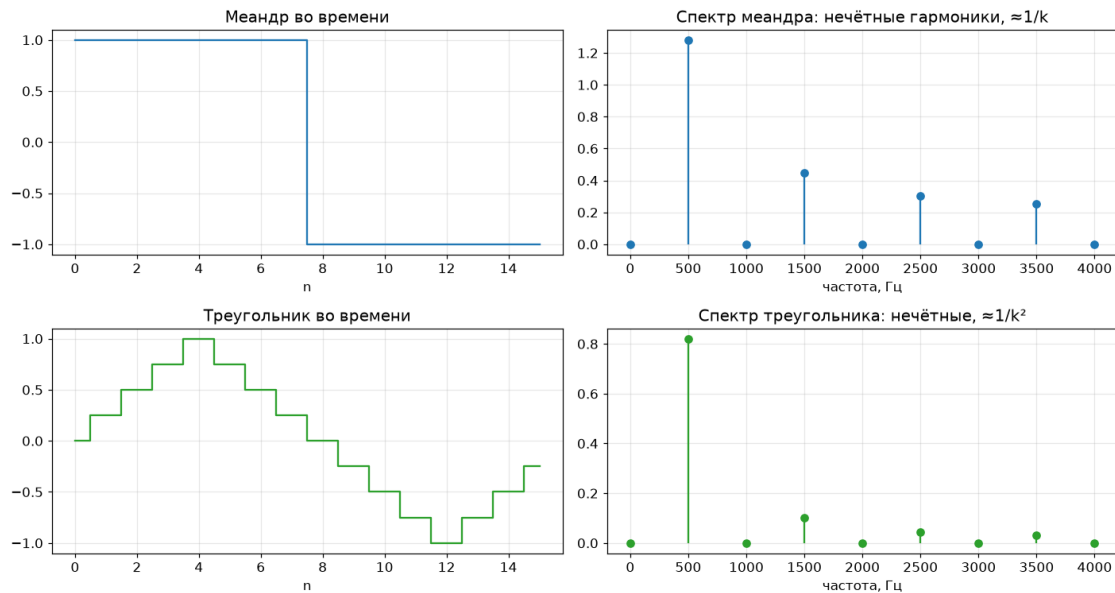
fig, ax = plt.subplots(2, 2, figsize=(13, 7))
ax[0, 0].step(n, sq, where="mid")
```

```

ax[0, 0].set_title("Меандр во времени")
ax[0, 1].stem(f, np.abs(np.fft.rfft(sq)) / N * 2, basefmt=" ")
ax[0, 1].set_title("Спектр меандра: нечётные гармоники,  $\approx 1/k$ ")
ax[1, 0].step(n, tri, where="mid", color="C2")
ax[1, 0].set_title("Треугольник во времени")
ax[1, 1].stem(f, np.abs(np.fft.rfft(tri)) / N * 2, basefmt=" ",
               linefmt="C2", markerfmt="C2o")
ax[1, 1].set_title("Спектр треугольника: нечётные,  $\approx 1/k^2$ ")
for a in ax.flatten()[::2]:
    a.set_xlabel("n")
for a in ax.flatten()[1::2]:
    a.set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_meander_triangle.png")
plt.show()

```

сохранено: fig_meander_triangle.png



1.5 3. Что означает ДПФ (инженерная сторона)

- $X[k]$ — комплексная амплитуда k -й частоты $f_k = k \cdot f_d / N$.
- $|X[k]|$ — амплитуда, $\arg X[k]$ — фаза составляющей.
- Ось частот задаётся через `np.fft.fftfreq` (двусторонняя) или `rfftfreq` (односторонняя).
- Амплитуду нормируют: $|X|/N$ (двусторонняя) или $2|X|/N$ (односторонняя).

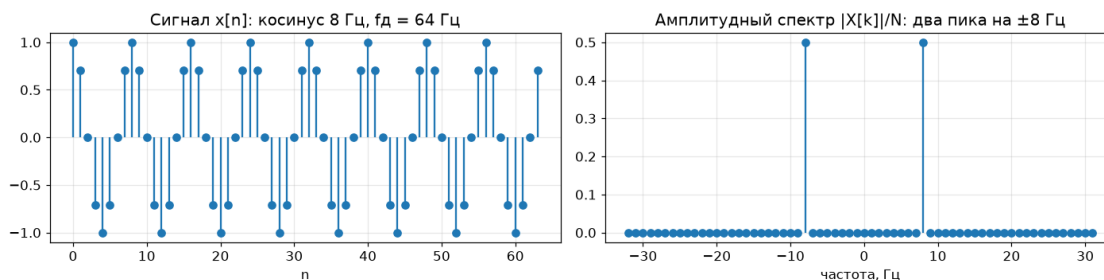
Простейший пример: косинус, частота которого **точно попадает в бин**.

```
[6]: fs, N = 64.0, 64
n = np.arange(N)
f0 = 8.0
x = np.cos(2 * np.pi * f0 * n / fs)

X = np.fft.fft(x)
f = np.fft.fftshift(np.fft.fftfreq(N, 1 / fs))
Xs = np.fft.fftshift(X)

fig, ax = plt.subplots(1, 2, figsize=(13, 3.4))
ax[0].stem(n, x, basefmt=" ")
ax[0].set_title("Сигнал x[n]: косинус {f0:.0f} Гц, fd = {fs:.0f} Гц")
ax[0].set_xlabel("n")
ax[1].stem(f, np.abs(Xs) / N, basefmt=" ")
ax[1].set_title("Амплитудный спектр |X[k]|/N: два пика на ±8 Гц")
ax[1].set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_dft_tone.png")
plt.show()
```

сохранено: fig_dft_tone.png



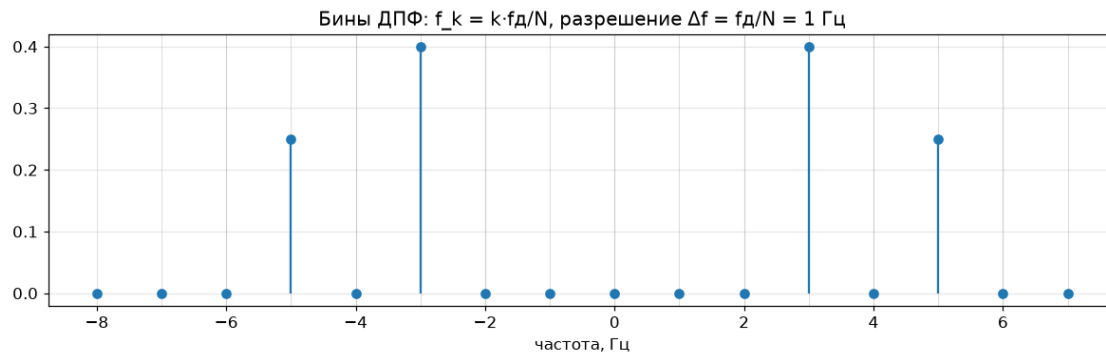
Дискретные частоты (бины). ДПФ знает сигнал только на частотах $f_k = k \cdot f_d / N$. Шаг по частоте — **разрешение** $\Delta f = f_d / N$: оно тем меньше, чем длиннее запись ($T = N / f_d$, при этом $\Delta f = 1/T$).

```
[7]: fs, N = 16.0, 16
n = np.arange(N)
x = 0.8 * np.cos(2 * np.pi * 3 * n / fs) + 0.5 * np.cos(2 * np.pi * 5 * n / fs)
f = np.fft.fftshift(np.fft.fftfreq(N, 1 / fs))
Xs = np.fft.fftshift(np.fft.fft(x))

fig, ax = plt.subplots(figsize=(11, 3.6))
ax.stem(f, np.abs(Xs) / N, basefmt=" ")
for k in range(-int(N / 2) + 1, int(N / 2)):
    ax.axvline(k * fs / N, color="0.85", lw=0.6, zorder=0)
ax.set_title("Бины ДПФ: f_k = k·fd/N, разрешение Δf = fd/N = {fs / N:.0f} Гц")
```

```
ax.set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_dft_bins.png")
plt.show()
```

сохранено: fig_dft_bins.png



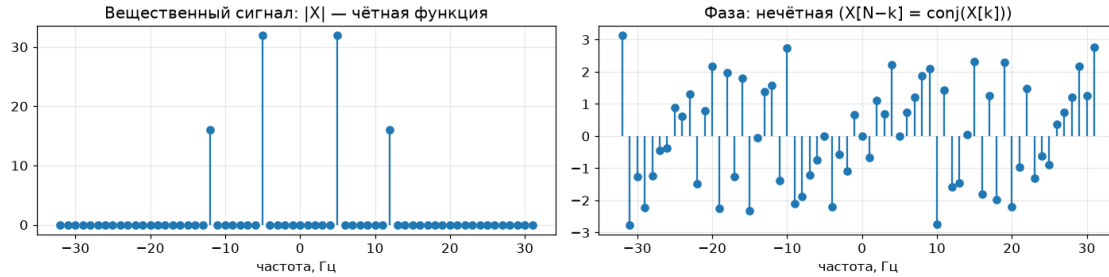
1.6 4. Свойства ДПФ

- **Линейность:** $\text{ДПФ}(a \cdot x + b \cdot y) = a \cdot \text{ДПФ}(x) + b \cdot \text{ДПФ}(y)$.
- **Симметрия вещественного сигнала:** если $x[n]$ веществен, то $x[N-k] = \overline{x[k]}$ (модуль чётный, фаза нечётная).
- **Сдвиг во времени $\leftrightarrow$ фаза:** $x[n-n_0]$ даёт множитель $e^{-j2\pi k n_0 / N}$ (модуль не меняется, фаза линейна по k).
- **Круговая свёртка:** $\text{ДПФ}(x \circledast y) = \text{ДПФ}(x) \cdot \text{ДПФ}(y)$ — свёртка превращается в умножение.
- **Парсеваль:** энергия сохраняется: $\sum |x[n]|^2 = (1/N) \sum |X[k]|^2$.

```
[8]: fs, N = 64.0, 64
n = np.arange(N)
x = 1.0 * np.cos(2 * np.pi * 5 * n / fs) + 0.5 * np.sin(2 * np.pi * 12 * n / fs)
f = np.fft.fftshift(np.fft.fftfreq(N, 1 / fs))
X = np.fft.fftshift(np.fft.fft(x))

fig, ax = plt.subplots(1, 2, figsize=(13, 3.4))
ax[0].stem(f, np.abs(X), basefmt=" ")
ax[0].set_title("Вещественный сигнал: |X| — чётная функция")
ax[0].set_xlabel("частота, Гц")
ax[1].stem(f, np.angle(X), basefmt=" ")
ax[1].set_title("Фаза: нечётная (X[N-k] = conj(X[k]))")
ax[1].set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_dft_real_symmetry.png")
plt.show()
```

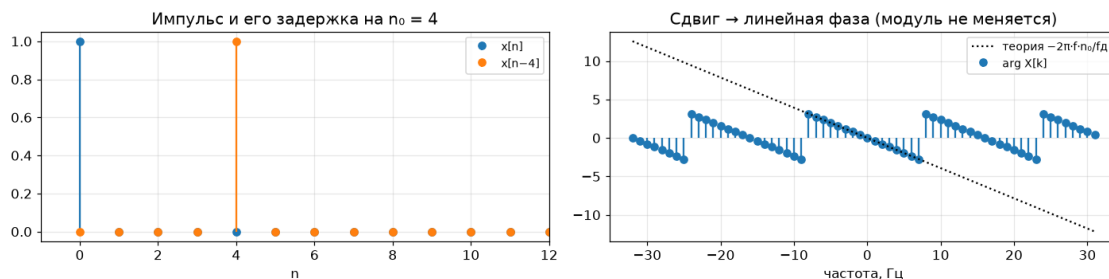
сохранено: fig_dft_real_symmetry.png



```
[9]: fs, N = 64.0, 64
n = np.arange(N)
x = np.zeros(N)
x[0] = 1.0
n0 = 4
xs = np.roll(x, n0)
f = np.fft.fftshift(np.fft.fftfreq(N, 1 / fs))
Xs = np.fft.fftshift(np.fft.fft(xs))

fig, ax = plt.subplots(1, 2, figsize=(13, 3.4))
ax[0].stem(n, x, basefmt=" ", label="x[n]")
ax[0].stem(n, xs, basefmt=" ", linefmt="C1", markerfmt="C1o",
           label="x[n-4]")
ax[0].set_xlim(-1, 12)
ax[0].set_title("Импульс и его задержка на  $n_0 = 4$ ")
ax[0].set_xlabel("n")
ax[0].legend(fontsize=9)
ax[1].stem(f, np.angle(Xs), basefmt=" ", label="arg X[k]")
ax[1].plot(f, -2 * np.pi * f * n0 / fs, "k:", label="теория  $-2\pi \cdot f \cdot n_0 / f_d$ ")
ax[1].set_title("Сдвиг → линейная фаза (модуль не меняется)")
ax[1].set_xlabel("частота, Гц")
ax[1].legend(fontsize=9)
fig.tight_layout()
save(fig, "fig_dft_shift_phase.png")
plt.show()
```

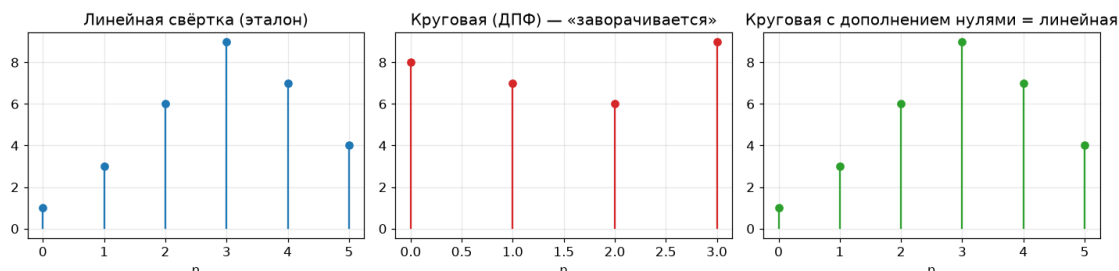
сохранено: fig_dft_shift_phase.png



```
[10]: a = np.array([1.0, 2.0, 3.0, 4.0])
b = np.array([1.0, 1.0, 1.0])
lin = np.convolve(a, b)
Nc = len(a)
circ = np.fft.ifft(np.fft.fft(a, Nc) * np.fft.fft(b, Nc)).real
L = len(a) + len(b) - 1
circ_pad = np.fft.ifft(np.fft.fft(a, L) * np.fft.fft(b, L)).real

fig, ax = plt.subplots(1, 3, figsize=(13, 3.4))
ax[0].stem(lin, basefmt=" ")
ax[0].set_title("Линейная свёртка (эталон)")
ax[1].stem(circ, basefmt=" ", linefmt="C3", markerfmt="C3o")
ax[1].set_title("Круговая (ДПФ) — «заворачивается»")
ax[2].stem(circ_pad, basefmt=" ", linefmt="C2", markerfmt="C2o")
ax[2].set_title("Круговая с дополнением нулями = линейная")
for a_ in ax:
    a_.set_xlabel("n")
fig.tight_layout()
save(fig, "fig_dft_conv.png")
plt.show()
```

сохранено: fig_dft_conv.png



Что такое «заворачивание». В круговой свёртке индексы берутся по модулю N:

$$y[n] = \sum_{m=0}^{N-1} x[m] h[(n - m) \bmod N].$$

Хвост линейной свёртки (индексы $\geq N$) не отбрасывается, а **складывается с началом**:

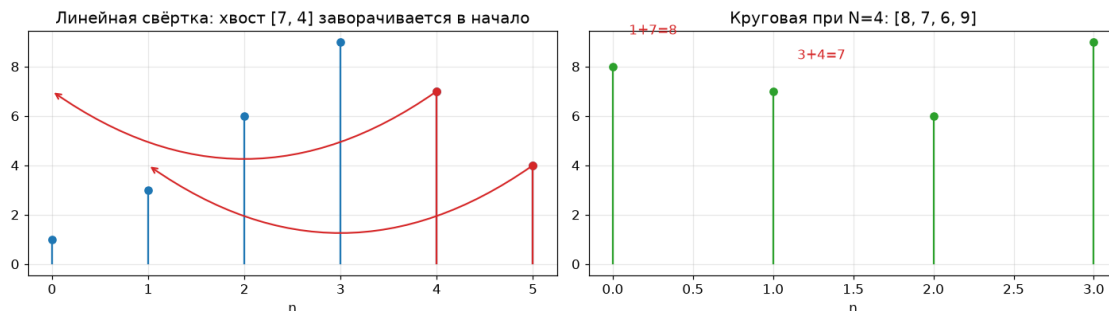
$$y_{\text{circ}}[n] = y_{\text{lin}}[n] + y_{\text{lin}}[n + N] + \dots$$

Пример $[1, 2, 3, 4] \otimes [1, 1, 1]$ при $N=4$: $y_{\text{lin}} = [1, 3, 6, 7, 4]$, хвост $[7, 4]$ заворачивается в начало и даёт $y_{\text{circ}} = [8, 7, 6, 9]$ ($1+7=8, 3+4=7$).

```
[11]: a = np.array([1.0, 2.0, 3.0, 4.0])
h = np.array([1.0, 1.0, 1.0])
lin = np.convolve(a, h)
Nc = len(a)
circ = np.array([sum(lin[i] for i in range(n, len(lin), Nc)) for n in range(Nc)])

fig, ax = plt.subplots(1, 2, figsize=(13, 3.8))
ax[0].stem(np.arange(len(lin)), lin, basefmt=" ")
ax[0].stem(np.arange(4, len(lin)), lin[4:], basefmt=" ",
            linefmt="C3-", markerfmt="C3o")
for src in (4, 5):
    ax[0].annotate("", xy=(src - Nc, lin[src]), xytext=(src, lin[src]),
                  arrowprops=dict(arrowstyle="->", color="C3", lw=1.4,
                                  connectionstyle="arc3,rad=-0.35"))
ax[0].set_title("Линейная свёртка: хвост [7, 4] заворачивается в начало")
ax[0].set_xlabel("n")
ax[1].stem(np.arange(Nc), circ, basefmt=" ", linefmt="C2-", markerfmt="C2o")
ax[1].annotate("1+7=8", xy=(0, circ[0]), xytext=(0.1, circ[0] + 1.3), color="C3")
ax[1].annotate("3+4=7", xy=(1, circ[1]), xytext=(1.15, circ[1] + 1.3),
              color="C3")
ax[1].set_title("Круговая при N=4: [8, 7, 6, 9]")
ax[1].set_xlabel("n")
fig.tight_layout()
save(fig, "fig_circular_ring.png")
plt.show()
print("y_lin =", lin)
print("y_circ =", circ)
```

сохранено: fig_circular_ring.png



```
y_lin = [1. 3. 6. 9. 7. 4.]
y_circ = [8. 7. 6. 9.]
```

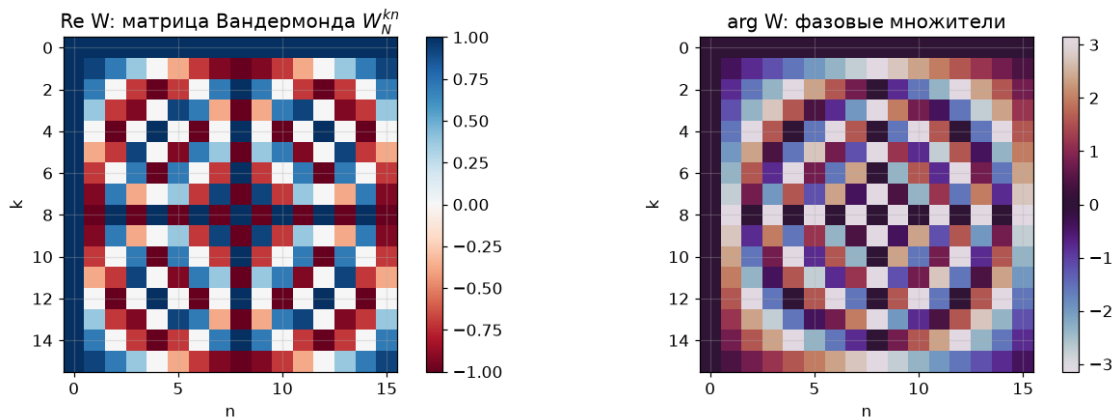
```
[12]: N = 16
k = np.arange(N)
W = np.exp(-2j * np.pi * np.outer(k, k) / N)
```

```

fig, ax = plt.subplots(1, 2, figsize=(12, 4.2))
im0 = ax[0].imshow(W.real, cmap="RdBu", vmin=-1, vmax=1)
ax[0].set_title("Re W: матрица Вандермонда  $W_N^{kn}$ ")
fig.colorbar(im0, ax=ax[0], fraction=0.046)
im1 = ax[1].imshow(np.angle(W), cmap="twilight", vmin=-np.pi, vmax=np.pi)
ax[1].set_title("arg W: фазовые множители")
fig.colorbar(im1, ax=ax[1], fraction=0.046)
for a_ in ax:
    a_.set_xlabel("n")
    a_.set_ylabel("k")
fig.tight_layout()
save(fig, "fig_dft_matrix.png")
plt.show()

```

сохранено: fig_dft_matrix.png



1.7 5. Утечка: когда частота между бинами

Если частота **не совпадает** с сеткой f_k , энергия «растекается» по соседним бинам — возникает **утечка** (см. лекцию 2). ДПФ при этом показывает «холмик» вместо одной линии.

```

[13]: fs, N = 64.0, 64
n = np.arange(N)
f0 = 8.5
x = np.cos(2 * np.pi * f0 * n / fs)
f = np.fft.rfftfreq(N, 1 / fs)
X = np.fft.rfft(x)

fig, ax = plt.subplots(figsize=(11, 3.6))
ax.stem(f, np.abs(X) / N * 2, basefmt=" ")
ax.axvline(f0, color="C3", ls="--", lw=1, label="истинная частота 8.5 Гц")

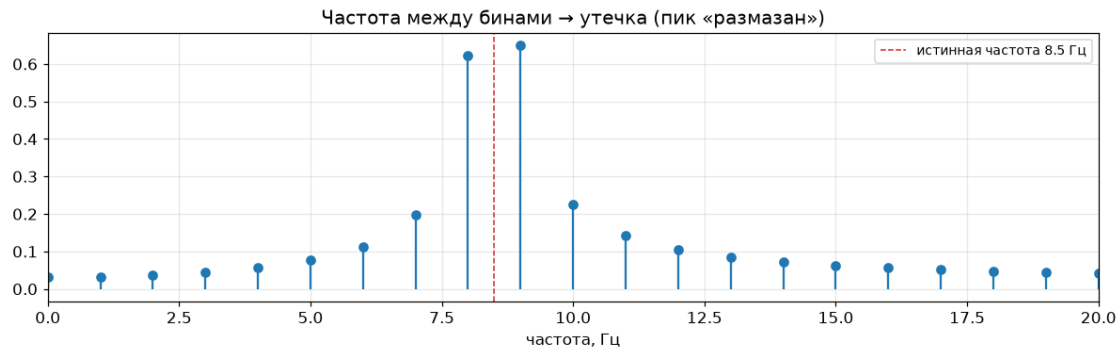
```

```

ax.set_xlim(0, 20)
ax.set_title("Частота между бинами → утечка (пик «размазан»)")
ax.set_xlabel("частота, Гц")
ax.legend(fontsize=9)
fig.tight_layout()
save(fig, "fig_dft_offbin.png")
plt.show()

```

сохранено: fig_dft_offbin.png



1.8 6. ДПФ реального сигнала и ось частот

Для сигнала из проекта возьмём вибрацию: базовая частота, дефект подшипника и шум. Считаем ДПФ и переводим бины в герцы.

```

[14]: fs, N = 1000.0, 1000
t = np.arange(N) / fs
rng = np.random.default_rng(1)
vib = (0.3 * np.sin(2 * np.pi * 50 * t) + 0.8 * np.sin(2 * np.pi * 120 * t)
       + 0.15 * rng.standard_normal(N))

f = np.fft.rfftfreq(N, 1 / fs)
X = np.fft.rfft(vib)

fig, ax = plt.subplots(figsize=(11, 3.6))
mask = f <= 300
ax.stem(f[mask], np.abs(X[mask]) / N * 2, basefmt=" ",
        linefmt="C0-", markerfmt="C0.")
for f0, lab in [(50, "база 50 Гц"), (120, "дефект 120 Гц")]:
    ax.annotate(lab, xy=(f0, np.abs(X[np.argmax(abs(f - f0))]) / N * 2),
               xytext=(f0 + 40, 0.4), color="C3",
               arrowprops=dict(arrowstyle="->", color="C3"))
ax.set_xlim(0, 300)
ax.set_title("Спектр вибрации (ДПФ): дискретные линии  $f_k = k \cdot f_d / N$ ")

```

```
ax.set_xlabel("частота, Гц")
fig.tight_layout()
save(fig, "fig_dft_case.png")
plt.show()
```

сохранено: fig_dft_case.png



1.9 7. Что такое БПФ и почему оно нужно

- **БПФ** (быстрое преобразование Фурье, FFT) — **не новое преобразование**, а быстрый **алгоритм** для того же самого ДПФ.
- Идея «разделяй и властвуй» (Cooley–Tukey): разбить N отсчётов на **чётные** и **нечётные** и свести к двум ДПФ половинной длины; результаты соединяются «бабочкой»:

$$X[k] = E[k] + W_N^k O[k], \quad X[k + N/2] = E[k] - W_N^k O[k].$$

- Для $N = 2^m$ рекурсия даёт $\log_2 N$ уровней по N операций:

$$\text{ДПФ: } O(N^2) \quad \text{против} \quad \text{БПФ: } O(N \log_2 N).$$

Для $N = 10^6$: $N^2 = 10^{12}$, а $N \cdot \log_2 N \approx 2 \cdot 10^7$ — разница в десятки тысяч раз.

```
[15]: from time import perf_counter
```

```
def dft_slow(x):
    N = len(x)
    out = np.zeros(N, dtype=complex)
    for k in range(N):
        s = 0.0
        for n in range(N):
            s += x[n] * np.exp(-2j * np.pi * k * n / N)
        out[k] = s
    return out
```

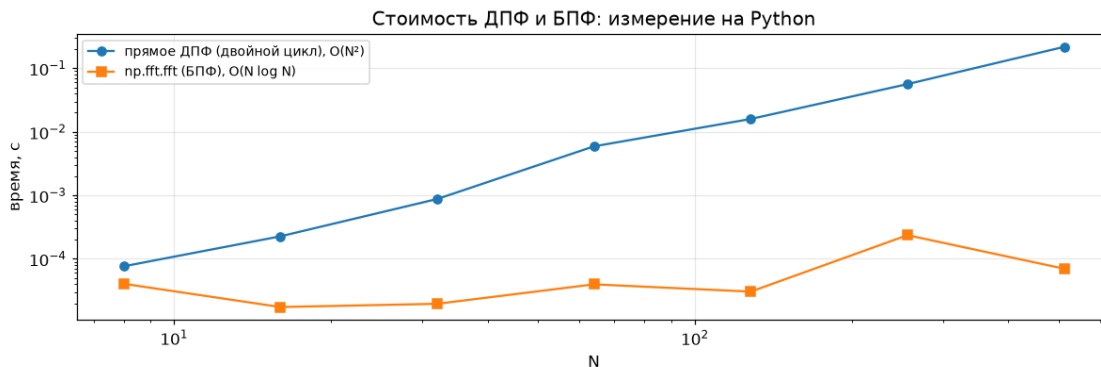
```

Ns = 2 ** np.arange(3, 10)
t_dft, t_fft = [], []
rng = np.random.default_rng(0)
for N in Ns:
    x = rng.standard_normal(N) + 1j * rng.standard_normal(N)
    t0 = perf_counter(); dft_slow(x); t_dft.append(perf_counter() - t0)
    t0 = perf_counter(); np.fft.fft(x); t_fft.append(perf_counter() - t0)

fig, ax = plt.subplots(figsize=(11, 3.8))
ax.loglog(Ns, t_dft, "o-", label="прямое ДПФ (двойной цикл), O(N²)")
ax.loglog(Ns, t_fft, "s-", label="np.fft.fft (БПФ), O(N log N)")
ax.set_xlabel("N")
ax.set_ylabel("время, с")
ax.set_title("Стоимость ДПФ и БПФ: измерение на Python")
ax.legend(fontsize=9)
fig.tight_layout()
save(fig, "fig_fft_cost.png")
plt.show()

```

сохранено: fig_fft_cost.png



На графике видно, как растёт разрыв: при больших N прямое ДПФ квадратично «упирается», а БПФ растёт почти по прямой в лог-масштабе. Поэтому на практике считают **БПФ**, а ДПФ понимают как его математическую основу (лекция 4).

1.10 Сквозной проект: задание №3

Задание после лекции 3 (для вашего проекта):

1. Посчитать **ДПФ** сигнала своего проекта (непрерывный→дискретный спектр).
2. Правильно построить **ось частот** (`rfftfreq`), отметить разрешение $\Delta f = f_d/N$.
3. Проверить **симметрию** для вещественного сигнала.
4. Показать **утечку** при выборе числа отсчётов, не кратного периоду.

5. Сравнить результат с непрерывной теорией (где она известна).

Сдавать: ноутбук с графиками + **один** слайд-вывод.

1.11 Итоги лекции 3

- **ДПФ** переводит N отсчётов в N спектральных коэффициентов: $X[k] = \sum x[n] e^{-j2\pi kn/N}$.
- Частоты дискретны: $f_k = k \cdot f_d/N$, разрешение $\Delta f = f_d/N = 1/T$.
- Свойства: линейность, симметрия вещественного сигнала, сдвиг $\leftrightarrow$ фаза, **круговая свёртка** $\leftrightarrow$ умножение, Парсеваль.
- Матричная форма $X = Wx$; обратное — через W^H/N .
- Прямое ДПФ $O(N^2) \rightarrow$ отсюда переход к **БПФ** (лекция 4).

Дальше: быстрое преобразование Фурье (БПФ) и практический спектральный анализ: окна, разрешение, усреднение.

1.12 Литература

1. Оппенгейм А., Шафер Р. *Цифровая обработка сигналов*. — М.: Техносфера, 2012.
2. Сергиенко А. Б. *Цифровая обработка сигналов*. — СПб.: БХВ-Петербург, 2011.
3. Солонина А. И. и др. *Основы цифровой обработки сигналов*. — СПб.: БХВ-Петербург, 2005.
4. Умняшкин С. В. *Основы теории цифровой обработки сигналов*.
5. Материалы курса и RAG-ассистент по этим книгам — в папке курса.