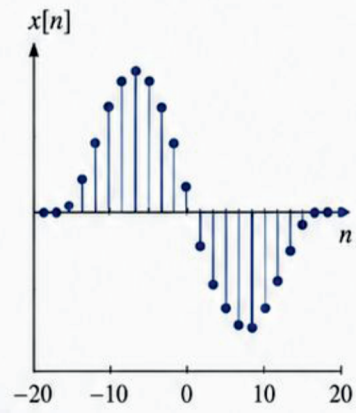
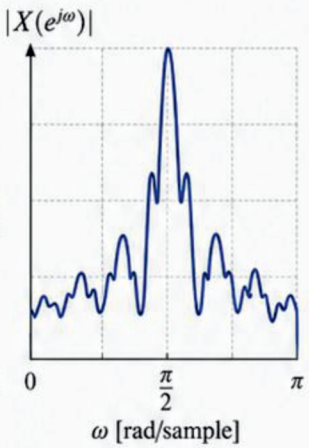
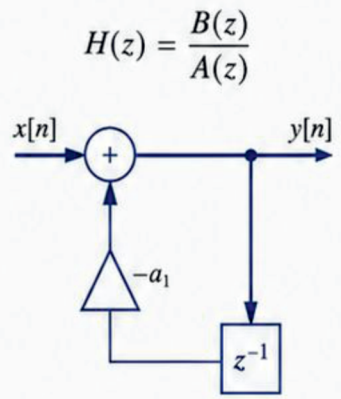
# ЦИФРОВА ОБРАБОТКА НА СИГНАЛИ

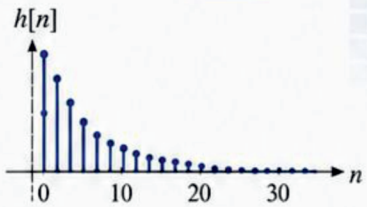
## Ръководство за лабораторни упражнения

ВРЕМЕВА ОБЛАСТ → ЧЕСТОТНА ОБЛАСТ → ЦИФРОВ ФИЛТЪР → ЦИФРОВО ИЗОБРАЖЕНИЕ






Христина Стойчева

Боян Карапенев

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО ГАБРОВО

2026

Христина Стойчева Боян Карапенев

# ЦИФРОВА ОБРАБОТКА НА СИГНАЛИ

**Ръководство за лабораторни упражнения**

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
"ВАСИЛ АПРИЛОВ"
ГАБРОВО, 2026

Рецензент: проф. д-р инж. Красен Киров Ангелов

© гл. ас. д-р инж. Христина Светославова Стойчева
© доц. д-р инж. Боян Димитров Карапенев

© Университетско издателство „Васил Априлов" – Габрово, 2026 г.

**ISBN: 978-954-683-756-1**

# Съдържание

**Съдържание** ..... 3

**Предговор** ..... 4

**Упражнение №1 Генериране и визуализация на дискретни сигнали в MATLAB.** ..... 5

**Упражнение №2 Дискретизация на непрекъснати сигнали** ..... 14

**Упражнение №3 Линейни дискретни инвариантни във времето (LTI) системи – анализ във времевата област** ..... 20

**Упражнение №4 Линейни дискретни инвариантни във времето (LTI) системи – анализ във честотната област** ..... 28

**Упражнение №5 Дискретно преобразуване на Фурие (DFT). Спектрален анализ на дискретни сигнали** ..... 35

**Упражнение №6 Бързо преобразуване на Фурие (FFT). Спектрален анализ на дискретни сигнали** ..... 42

**Упражнение №7 Спектрален анализ и прозоречни функции. Спектрално изтичане** ..... 47

**Упражнение №8 Децимация и интерполация на цифрови сигнали. Промяна на честотата на дискретизация** ..... 59

**Упражнение №9 Филтри с крайна импулсна характеристика - синтез и анализ** ..... 74

**Упражнение №10 Филтри с безкрайна импулсна характеристика - синтез и анализ** ..... 87

**Упражнение №11 Въведение в цифровата обработка на изображения. Представяне и основни операции с изображения в MATLAB** ..... 101

**Упражнение №12 Трансформации на яркостта и подобряване на контраста. Хистограми** ..... 110

**Упражнение №13 Пространствено филтриране на цифрови изображения. Потискане на шум** ..... 123

**Упражнение №14 Обработка на изображения в честотната област** ..... 134

**Упражнение №15 Откриване на контури, морфологична обработка и сегментация на обекти в цифрови изображения** ..... 145

**Използвана литература** ..... 159

**Приложение** ..... 160

# Предговор

Развитието на цифровите технологии и нарастващите възможности на изчислителните системи превръщат цифровата обработка на сигнали в една от основните области на съвременната електроника и комуникационна техника. Преминаването от аналогова към цифрова обработка позволява реализирането на сложни алгоритми с висока точност, гъвкавост и възможност за програмна реализация. Днес методите на цифровата обработка намират широко приложение в комуникационните системи, аудио- и видеообработката, медицинската техника, автоматизацията, измервателните системи, обработката на изображения и редица други области.

Ръководството за лабораторни упражнения е учебно пособие по дисциплината „Цифрова обработка на сигнали“, изучавана от студенти, обучаващи се в бакалавърски курс, по специалност „Комуникационни технологии и киберсигурност“ към катедра Комуникационна техника и технологии (КТТ) при ТУ – Габрово, но може да бъде използвано от всички други студенти и докторанти, които имат интерес към анализа и обработката на цифрови сигнали и изображения, проектиране и прилагане на цифрови филтри, реализация на основни алгоритми за спектрален анализ, филтриране и сегментация в продукта MATLAB.

Ръководството съчетава основни теоретични постановки с практическата им реализация в MATLAB. Изборът на тази програмна среда се обуславя от широките ѝ възможности за моделиране, анализ и визуализация на цифрови сигнали и изображения, както и от наличието на специализирани средства за реализация и изследване на изучаваните алгоритми.

Упражненията са организирани последователно и обхващат генериране и дискретизация на сигнали, теоремата на Найкуист–Шанън и алиасинг, анализ на дискретни LTI системи във времевата и честотната област, DFT и FFT, спектрално изтичане и прозоречни функции. Разглеждат се също децимация и интерполация, както и проектиране и анализ на IIR и FIR цифрови филтри.

Последната част е посветена на цифровата обработка на изображения като двумерни дискретни сигнали. Включени са основни операции с изображения, хистограмна обработка, пространствено и честотно филтриране, потискане на шум, откриване на контури, сегментация и морфологична обработка.

Изпълнението на лабораторните упражнения има за цел да формира практически умения за анализ и обработка на цифрови сигнали и изображения и да подпомогне разбирането на изучаваните процеси чрез тяхното програмно моделиране и визуализация.

*Всички бележки и препоръки за подобряване на ръководството ще бъдат приети с благодарност.*

**Авторите**

**Упражнение №1**

# Генериране и визуализация на дискретни сигнали в MATLAB.

**I Цел на упражнението** – Прилагане на продукта MATLAB за генериране и елементарна обработка на сигнали, изграждане на интуиция за връзката между математическия модел и визуалното представяне.

## II Теоретична част

*2.1. Непрекъснати и дискретни сигнали*

Сигналът представлява физична величина, която пренася информация и се изменя във времето, пространството или друга независима променлива. В зависимост от начина на представяне сигналите се разделят на непрекъснати и дискретни.

- *Непрекъснат сигнал* – дефиниран за всеки момент от времето:
$$x(t), \qquad t \in \mathbb{R}$$
- *Дискретен сигнал* – дефиниран само в определени дискретни моменти:
$$x[n] = x(nT_s), \qquad n \in \mathbb{Z}$$
където $T_s$ е периодът на дискретизация, а $f_s = \frac{1}{T_s}$ – честотата на дискретизация.

В MATLAB дискретните сигнали обикновено се представят чрез *вектори*, а времето – чрез вектор с равномерна стъпка.

*2.2. Хармонични сигнали*

Най-важният клас сигнали в цифровата обработка са хармоничните сигнали, тъй като всеки периодичен сигнал може да бъде представен като сума от постояннотокова съставяща и хармоници.

*Синусоидален сигнал:*
$$x[n] = A\,sin(2\pi f n T_s + \varphi)$$
*Косинусоидален сигнал:*
$$x[n] = A\,cos(2\pi f n T_s + \varphi)$$
където: $A$ – амплитуда, $f$ – честота, $\varphi$ – начална фаза, а $T_s$ – период на дискретизация.

Разликата между синус и косинус е *фазово отместване* от $\frac{\pi}{2}$.

*2.3. Несинусоидални периодични сигнали*

Освен хармоничните сигнали, в практиката се използват и несинусоидални периодични сигнали, като правоъгълен и трионообразен сигнал. Тези сигнали съдържат множество хармонични съставки и се използват в електрониката, комуникациите и управлението. В MATLAB те се генерират чрез готови функции, които създават идеализирани математически модели.

*2.4. Сложни сигнали*

Сложен сигнал представлява сума от няколко елементарни сигнала:
$$x[n] = \sum_{k=1}^{K} A_k\,sin(2\pi f_k n T_s + \varphi_k)$$

Такива сигнали моделират реални физични процеси, позволяват анализ на влиянието на различни честотни компоненти и подготвят основата за Фурие анализа.

*2.5. Шум и зашумени сигнали*

В реални условия сигналите почти винаги са съпроводени с шумове. Най-често използваният математически модел е на адитивен бял гаусов шум (AWGN).

Зашуменият сигнал може да се представи като:

$$y[n] = x[n] + \omega[n],$$

където $\omega[n]$ е случаен процес с нормално разпределение по отношение на времето със средна стойност нула.

Основните характеристики на шума са:

- имат случаен характер;
- широк честотен спектър;
- негативно влияние върху анализа и обработката на сигнали.

**III Подготовка за упражнението**

В настоящото упражнение се използват основни функции на MATLAB за генериране, обработка и визуализация на дискретни сигнали.

*3.1. Функции за генериране на сигнали*

- **`sin()`** – генерира синусоидален сигнал за зададен аргумент.
- **`cos()`** – генерира косинусоидален сигнал за зададен аргумент.
- **`sawtooth()`** – генерира периодичен трионообразен или пилообразен сигнал.
- **`square()`** – генерира периодичен правоъгълен сигнал.
- **`randn()`** – генерира псевдослучайни числа с нормално (гаусово) разпределение.

*3.2. Функции за визуализация*

- **`plot()`** – изчертава графика на сигнал във времевата област, като го представя като непрекъсната линия.
- **`stem()`** – изчертава дискретен сигнал чрез вертикални линии и маркер за всеки стълб.
- **`subplot()`** – позволява изобразяване на няколко графики в един прозорец чрез разделяне на фигурата на подграфики.
- **`figure()`** – отваря нов графичен прозорец за визуализация.

*3.3. Функции за оформление на графики*

- **`xlabel()`** – задава надпис на абсцисната ос.
- **`ylabel()`** – задава надпис на ординатната ос.
- **`title()`** – добавя заглавие към графиката.
- **`legend()`** – добавя легенда за означаване на различните сигнали.
- **`grid on`** – включва координатна мрежа за по-лесно отчитане на стойности.
- **`hold on`** – позволява наслагване на няколко графики в една и съща координатна система.

*3.4. Спомагателни операции*

- **`size()`** – определя размерността на вектор или матрица.
- **`length()`** – връща броя на елементите във вектор.
- **`linspace()`** – генерира равномерно разпределен вектор между две стойности.

## IV Задачи за изпълнение

1. Да се генерира и визуализира синусоидален сигнал с честота 50 Hz. Да се използва линия „тире-точка” в червен цвят, растерна мрежа, заглавие и маркери във вид на квадратчета с размери 3 pt, запълнени със зелен цвят и оконтурени с черни линии в ширина на линията 1 pt.

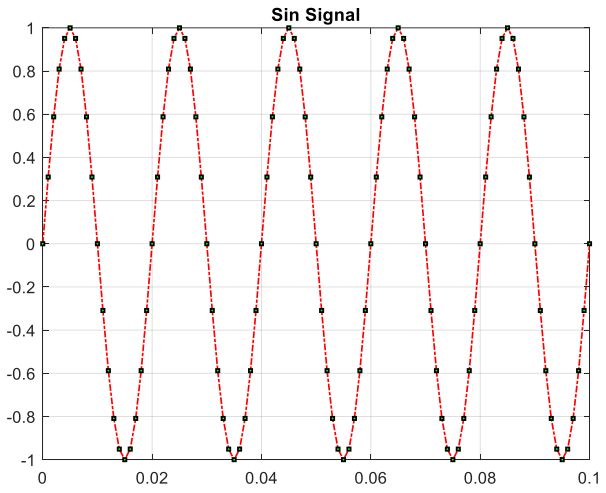
Код в MATLAB:

```
% Generation of Sin Signal
t=0:.001:.1;
y=sin(2*pi*50*t);

plot(t,y,'-.rs','LineWidth',1,'MarkerEdgeColor','k',...
'MarkerFaceColor','g','MarkerSize',3); grid;

title('Sin Signal')
```

```
--- Резултати ---
```




2. Да се генерира и визуализира сигнал, състоящ се от постоянна съставяща, синусоида с честота 50 Hz и бял шум с нормално разпределение. Визуализацията да е с плътна линия в жълт цвят, растерна мрежа, заглавие и величини по координатните оси.

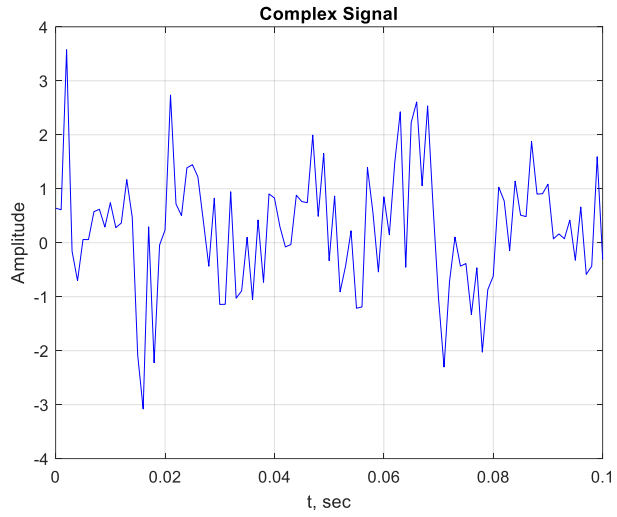
Код в MATLAB:

```
% Complex Signal
t=0:.001:.1;
y=0.2+sin(2*pi*50*t)+randn(1,length(t));

plot(t,y,'color',[0 0 1]); grid

title('Complex Signal');
xlabel('t, sec');
ylabel('Amplitude');
```

```
--- Резултати ---
```




3. Да се генерира трионообразен сигнал с честота 50 Hz и с продължителност 1,5 s с честота на дискретизация 10 kHz. Да се изчертаят първите 0,2 s от продължителността на сигнала.

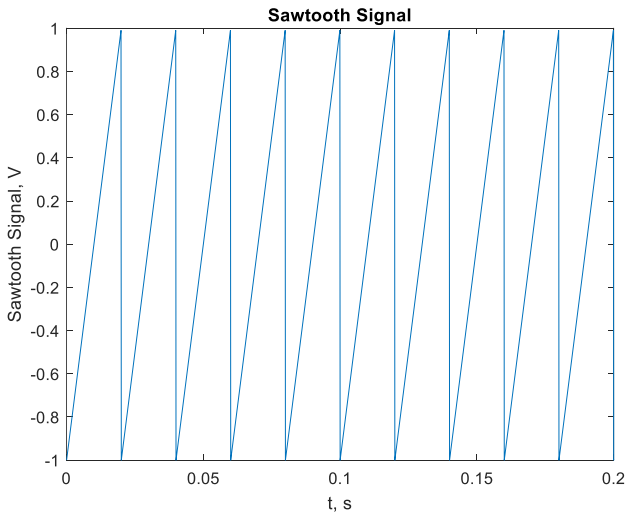
Код в MATLAB:

```
% Sawtooth Signal
t=0:10^(-4):1.5;
s=sawtooth(2*pi*50*t);

plot(t,s);
axis([0 0.2 -1 1]);

title('Sawtooth Signal');
xlabel('t, s');
ylabel('Sawtooth Signal, V');
```

```
--- Резултати ---
```



4. Да се генерира сигнал s(t) с продължителност 1 s, състоящ се от две синусоиди, едната $s_1(t)$ с честота 50 Hz и с единична амплитуда, а другата - $s_2(t)$ - с честота 120 Hz и с два пъти по-малка амплитуда, като честотата на дискретизация е 1 kHz. Да се добави нормално разпределен бял шум към сигнала, като амплитудата му е два пъти по-малка от съставката с честота 50 Hz. Да се изчертаят по отделно първите 50 точки на сигналите $s_1(t)$, $s_2(t)$, s(t) и сигнала, примесен с шума sn(t).

Код в MATLAB:

```
% Generate sin1, sin2, sin + noise

t=0:0.001:1;
s1=sin(2*pi*50*t);
s2=2*sin(2*pi*120*t);
s=s1+s2;
sn=s+0.5*randn(size(t));

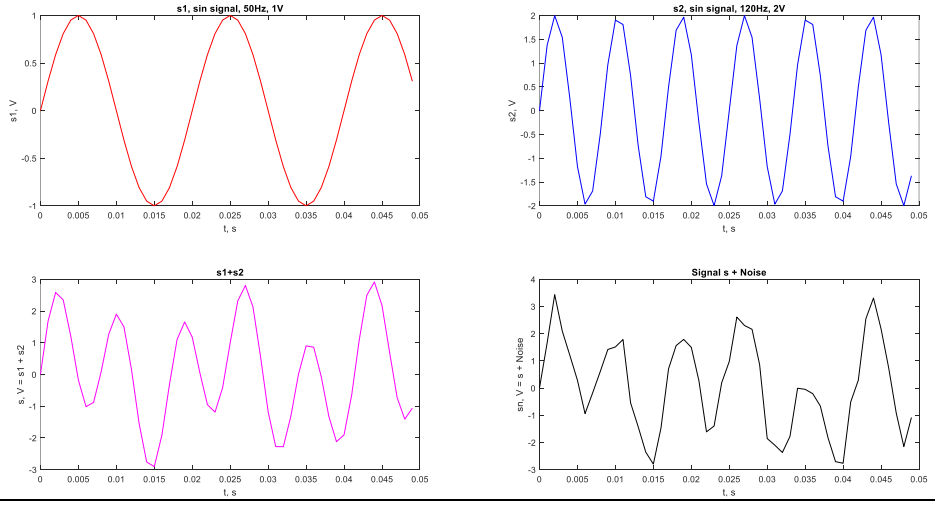
subplot(2,2,1);
plot(t(1:50),s1(1:50),'r');
title('s1, sin signal, 50Hz, 1V');
xlabel('t, s');
ylabel('s1, V');

subplot(2,2,2);
plot(t(1:50),s2(1:50),'b');
title('s2, sin signal, 120Hz, 2V');
xlabel('t, s');
ylabel('s2, V');

subplot(2,2,3);
plot(t(1:50),s(1:50),'m');
title('s1+s2');
xlabel('t, s');
ylabel('s, V = s1 + s2');

subplot(2,2,4);
plot(t(1:50),sn(1:50),'k');
title('Signal s + Noise');
xlabel('t, s');
ylabel('sn, V = s + Noise');
```

```
--- Резултати ---
```



5. Да се генерират сигналите, представени като вектор-стълб със 100 елемента:
   - цифров единичен импулс;

Код в MATLAB:

```
% Unit Impulse
delta = [1; zeros(99,1)];
stem(delta);
axis([0 25 0 1]);

title('Unit Impulse');
xlabel('No sample');
ylabel('Signal, V');
```

   - цифрово единично стъпало;

Код в MATLAB:

```
% Digital Unit Step
u=ones(100,1);
stem(u);
axis([0 25 0 1]);

title(‚Unit Step‘);
xlabel(‚No sample‘);
ylabel(‚Signal, V‘);
```

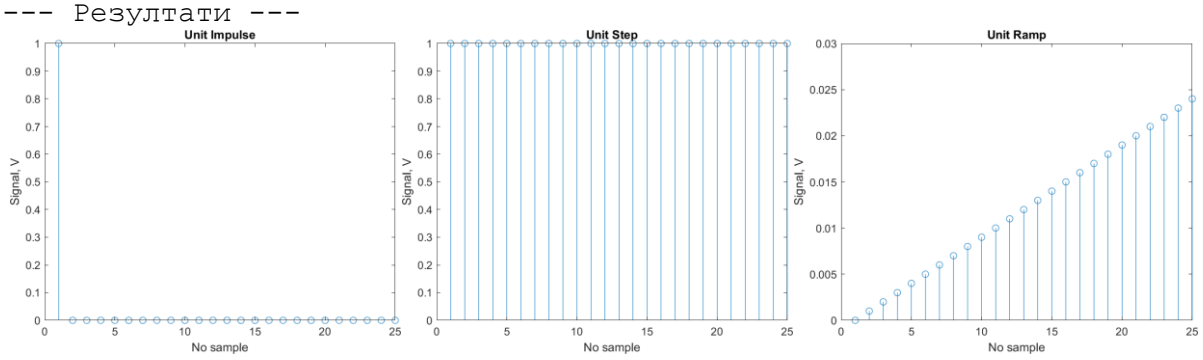
   - цифров единично-изменящ се наклон с време 0 ÷ 1 s със стъпка на изменение 1 ms.

Код в MATLAB:

```
% Unit Ramp
t=(0:0.001:1);
r=t;
stem(r);
axis([0 25 0 0.03]);

title('Unit Ramp');
xlabel('No sample');
ylabel('Signal, V');
```

```
--- Резултати ---
```



## V Допълнителни задачи

1. Дефинирайте честота на дискретизация $f_s = 1000\ Hz$. Създайте времеви вектор с продължителност 1 s. Генерирайте синусоидален и косинусоидален сигнал с честота 5 Hz. Изчертайте получения сигнал като добавите растерна мрежа, заглавие, величини по координатните оси и легенда.

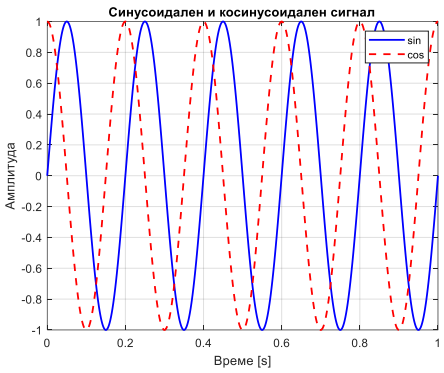
```
fs = 1000;
t = 0:1/fs:1;

x1 = sin(2*pi*5*t);
x2 = cos(2*pi*5*t);

figure;
plot(t, x1, 'b', 'LineWidth', 1.5);
hold on;
plot(t, x2, 'r--', 'LineWidth', 1.5);
grid on;

xlabel('Време [s]');
ylabel('Амплитуда');
title('Синусоидален и косинусоидален сигнал');
legend('sin', 'cos');
```

```
--- Резултати ---
```



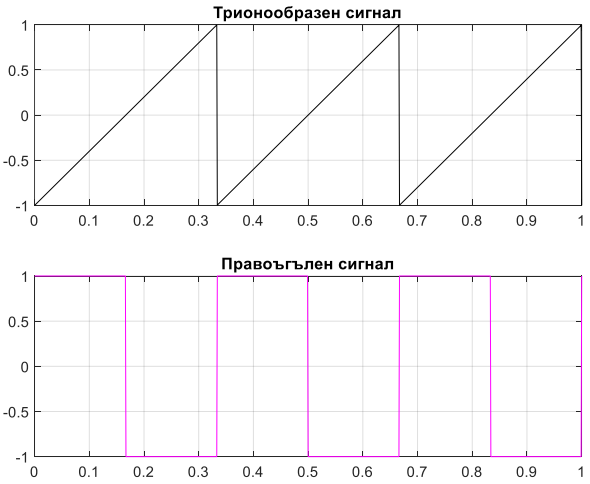

2. Генерирайте трионообразен и правоъгълен сигнал с честота 3 Hz. Визуализацията му да е с плътни линии в различни цветове, растерна мрежа, заглавие и величини по координатните оси.

```
x3 = sawtooth(2*pi*3*t);
x4 = square(2*pi*3*t);

figure;
subplot(2,1,1);
plot(t, x3, 'k');
grid on;
title('Трионообразен сигнал');

subplot(2,1,2);
plot(t, x4, 'm');
grid on;
title('Правоъгълен сигнал');
```

```
--- Резултати ---
```




3. Генерирайте сигнал с три хармонични съставки с различни честоти и амплитуди.
   а) Променете честотата на дискретизация и наблюдавайте резултата;
   б) Използвайте функцията `stem()` вместо `plot()` и сравнете визуализацията.

```
%% Параметри на сигнала
A1 = 1;       % амплитуда на първата хармоника
A2 = 0.6;     % амплитуда на втората хармоника
A3 = 0.3;     % амплитуда на третата хармоника

f1 = 5;       % честота [Hz]
f2 = 15;      % честота [Hz]
f3 = 30;      % честота [Hz]

T = 1;        % продължителност на сигнала [s]

%% Случай 1: Висока честота на дискретизация
fs1 = 1000;               % честота на дискретизация [Hz]
t1 = 0:1/fs1:T;

x1 = A1*sin(2*pi*f1*t1) + ...
     A2*sin(2*pi*f2*t1) + ...
     A3*sin(2*pi*f3*t1);

figure;
plot(t1, x1, 'LineWidth', 1.5);
grid on;
title('Сигнал с три хармонични съставки (fs = 1000 Hz)');
xlabel('Време [s]');
ylabel('Амплитуда');

%% Случай 2: Ниска честота на дискретизация
fs2 = 50;                % честота на дискретизация [Hz]
t2 = 0:1/fs2:T;

x2 = A1*sin(2*pi*f1*t2) + ...
     A2*sin(2*pi*f2*t2) + ...
     A3*sin(2*pi*f3*t2);

figure;
plot(t2, x2, 'o-', 'LineWidth', 1.5);
grid on;
```

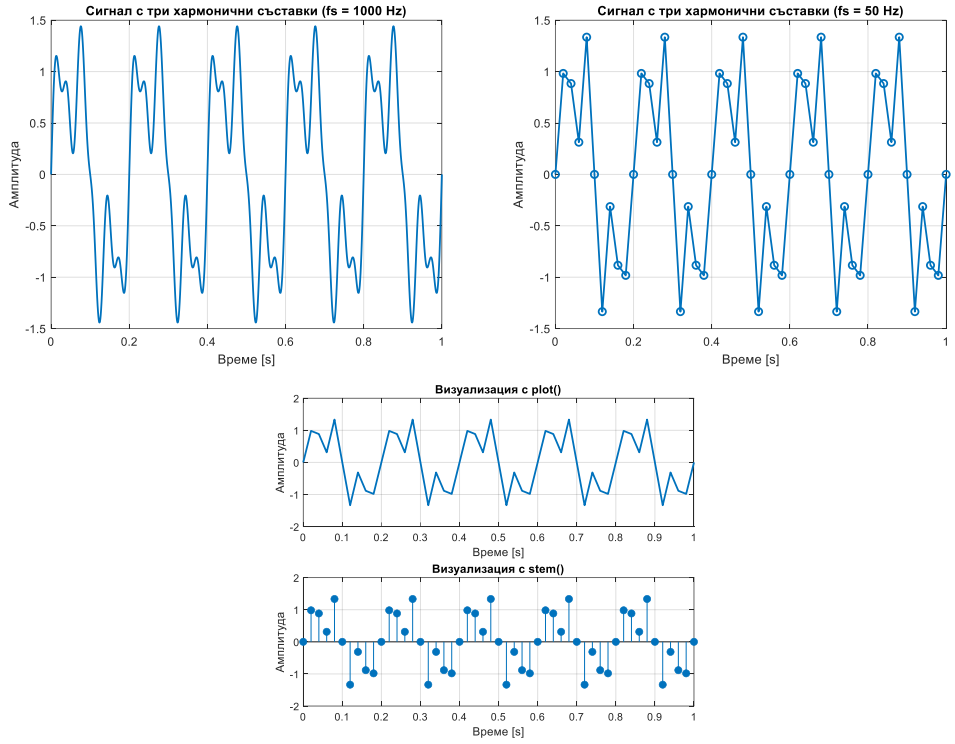
```
title('Сигнал с три хармонични съставки (fs = 50 Hz)');
xlabel('Време [s]');
ylabel('Амплитуда');

%% Сравнение plot() и stem()
figure;

subplot(2,1,1);
plot(t2, x2, 'LineWidth', 1.5);
grid on;
title('Визуализация с plot()');
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(2,1,2);
stem(t2, x2, 'filled');
grid on;
title('Визуализация с stem()');
xlabel('Време [s]');
ylabel('Амплитуда');
```

```
--- Резултати ---
```




## VI Контролни въпроси

1. Какво представлява честотата на дискретизация?
2. Каква е разликата между синусоидален и косинусоидален сигнал?
3. Какво представлява бялият гаусов шум?
4. Каква е ролята на функцията `subplot()`?

**Упражнение №2**

# Дискретизация на непрекъснати сигнали

**I Цел на упражнението** – Приложение на продукта MATLAB за дискретизация на непрекъснати сигнали, значимост на стойността на честотата на дискретизация, анализиране на влиянието на честотата на дискретизация върху формата на сигнала.

## II Теоретична част

*2.1. Непрекъснати и дискретни сигнали*

Непрекъснатият сигнал е дефиниран за всеки момент от времето и се описва с функция $x(t)$. Дискретният сигнал се получава чрез вземане на стойности на непрекъснатия сигнал през равни времеви интервали:

$$x[n] = x(nT_s)$$

където $T_s$ е периодът на дискретизация, а $f_s = \frac{1}{T_s}$ – честотата на дискретизация.

В MATLAB непрекъснатият сигнал се моделира чрез плътно семплиран времеви вектор, а дискретният – чрез по-рядко взети отсчети.

*2.2. Честота на дискретизация*

Честотата на дискретизация определя колко често се вземат стойности от непрекъснатия сигнал. По-висока честота на дискретизация води до по-точно представяне на сигнала, но изисква повече изчислителни ресурси.

*2.3. Теорема на Найкуист–Шенон*

Теоремата за семплиране на Найкуист-Шенон е основен принцип за цифровата обработка на сигнали, свързващ честотния диапазон на сигнала и честотата на семплиране, необходима за избягване на вид изкривяване, наречено алиасинг (назъбване или наричано още „*грешка*[1] *от наслагване на спектрите*“). Теоремата гласи, че честотата на семплиране трябва да бъде поне два пъти по-голяма от честотната лента на сигнала, за да се избегне алиасинг. На практика тя се използва за избор на филтри за ограничаване на честотната лента, за да се поддържа изкривяването под приемливо ниво, когато се семплира аналогов сигнал или когато честотите на семплиране се променят в рамките на функция за цифрова обработка на сигнали.

Теоремата за семплиране на Найкуист-Шанън е теорема в областта на обработката на сигнали, която служи като фундаментален мост между непрекъснати и дискретни сигнали. Тя установява достатъчно условие за честота на семплиране, която позволява дискретна последователност от семплирания да улови цялата информация от непрекъснат сигнал с крайна честотна лента.

Теоремата за семплиране въвежда концепцията за честота на дискретизация, която е достатъчна за перфектна точност за класа функции, които са ограничени по честотна лента до дадена честотна лента, така че да не се губи действителна информация в процеса на семплиране. Тя изразява достатъчната честота на дискретизация по

[1] *Алиасингът не е „грешка“, а математическо следствие от дискретизацията. Ако честотата на дискретизация не е достатъчно висока, различни непрекъснати сигнали стават неразличими след дискретизация.*

отношение на честотната лента за класа функции. Теоремата също така води до формула за перфектно възстановяване на оригиналната функция с непрекъснато време от семплирания.

**Теорема** – Ако функция $x(t)$ **не** съдържа честоти по-високи от B херца, тогава тя може да бъде напълно определена от нейните дискретизирани стойности в поредица от точки, разположени на разстояние по-малко от $\frac{1}{2B}$ секунди една от друга.

За точно възстановяване на честотно ограничен сигнал е необходимо:

$$f_s \geq 2f_{max},$$

където $f_{max}$ е най-високата честотна съставка в сигнала. Честотата $f_N = \frac{f_s}{2}$ се нарича честота на Найкуист.

*2.4. Алиасинг*

Алиасингът е изкривяване, при което реконструиран сигнал от дискрети на оригиналния сигнал съдържа нискочестотни компоненти, които не присъстват в оригиналния. Това води до необратими изкривявания на сигнала и грешна интерпретация на информацията.

Действителните сигнали имат крайна продължителност и честотното им съдържание, както е дефинирано от преобразуването на Фурие, няма горна граница. Винаги се получава известно количество изкривявания, когато се семплират такива непрекъснати във времето функции. Функциите, чието честотно съдържание е ограничено (ограничено по честотна лента), имат безкрайна продължителност във времевата област. Ако се семплират с достатъчно висока честота, определена от честотната лента, оригиналната функция може, на теория, да бъде перфектно реконструирана от безкрайния набор от семплирания.

*2.5. Алиасинг в честотната област*

При дискретизация на непрекъснат сигнал спектърът на дискретния става периодичен по честота с период $\mathrm{T}_s = 1/f_s$. Това означава, че спектралните копия на оригиналния спектър се повтарят около честоти:

$$kf_s, \qquad k = 0, \pm 1, \pm 2, \ldots$$

Ако сигналът съдържа честотни компоненти над честотата на Найкуист $f_N = \frac{f_s}{2}$, тогава спектралните копия се припокриват. Това припокриване се проявява като алиасинг: високочестотна компонента „се сгъва“ (folding) в диапазона $[0, f_N]$ и се наблюдава като по-ниска честота.

За синусоидален сигнал с честота $f$ и честота на дискретизация $f_s$, наблюдаваната (алиас) честота $f_a$ в дискретния сигнал може да се определи като:

$$f_a = |f - kf_s|, \qquad \text{за такова } k, \text{че } f_a \in \left[0, \frac{f_s}{2}\right]$$

*Пример:* ако $f = 15Hz$ и $f_s = 20Hz$, тогава:

$$f_a = |15 - 1 \cdot 20| = 5Hz$$

Тоест сигнал с честота 15 Hz при тази дискретизация ще изглежда като сигнал с честота равна на 5 Hz.

*2.6. Влияние на честотата на дискретизация*

При промяна на честотата на дискретизация:

- при висока $f_s$ сигналът се представя коректно;
- при $f_s \approx 2f_{max}$ се наблюдават гранични ефекти;
- при $f_s < 2f_{max}$ възниква алиасинг.

## III Подготовка за упражнението

Необходими основни функции на MATLAB:

- **`linspace()`** – създава равномерно разпределен времеви вектор.
- **`for ... end`** – изпълнява повтарящи се операции.
- **`length()`** – връща броя на елементите във вектор.
- **`v(k)`** – индексиране - достъпва конкретен елемент от вектор/масив.
- **`num2str()`** – преобразува число в текст, за да се използва в заглавия и надписи.

## IV Задачи за изпълнение

1. Генерирайте непрекъснат синусоидален сигнал с честота 5Hz чрез плътно семплиране. Дискретизирайте сигнала при различни честоти на дискретизация ($f_{s1} = 100Hz$, $f_{s2} = 20Hz$). Визуализирайте и сравнете непрекъснатия и дискретния сигнал ($f_{s1} = 100Hz$).

Код в MATLAB:

```
t_cont = linspace(0, 1, 10000);
x_cont = sin(2*pi*5*t_cont);

figure;
plot(t_cont, x_cont, 'k', 'LineWidth', 1.5);
grid on;
title('Непрекъснат модел на синусоидален сигнал');
xlabel('Време [s]');
ylabel('Амплитуда');

fs1 = 100;
fs2 = 20;

t1 = 0:1/fs1:1;
t2 = 0:1/fs2:1;

x1 = sin(2*pi*5*t1);
x2 = sin(2*pi*5*t2);

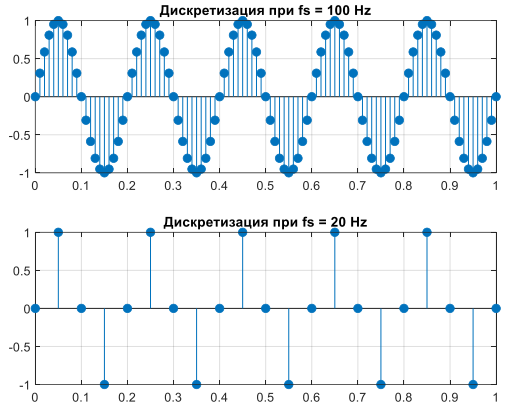
figure;
subplot(2,1,1);
stem(t1, x1, 'filled');
grid on;
title('Дискретизация при fs = 100 Hz');

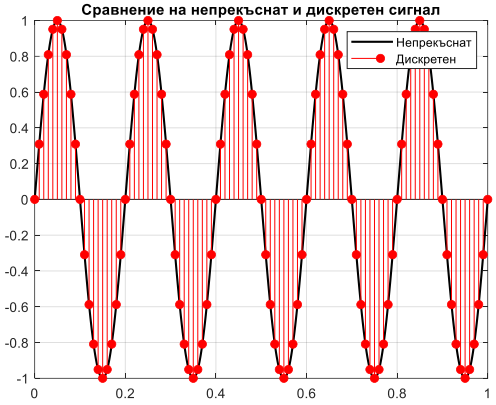
subplot(2,1,2);
stem(t2, x2, 'filled');
grid on;
title('Дискретизация при fs = 20 Hz');

figure;
plot(t_cont, x_cont, 'k', 'LineWidth', 1.5);
hold on;
stem(t1, x1, 'r', 'filled');
grid on;

legend('Непрекъснат', 'Дискретен');
title('Сравнение на непрекъснат и дискретен сигнал');
```

```
--- Резултати ---
```




2. Дискретизирайте сигнал с честота 15Hz при честота на дискретизация - 20Hz.

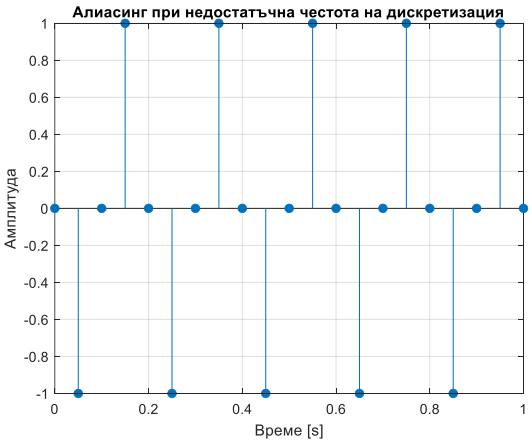
Код в MATLAB:

```
fs = 20;
t = 0:1/fs:1;

x_alias = sin(2*pi*15*t);

figure;
stem(t, x_alias, 'filled');
grid on;
title('Алиасинг при недостатъчна честота на дискретизация');
xlabel('Време [s]');
ylabel('Амплитуда');
```

```
--- Резултати ---
```




3. Визуализирайте дискретизация на един и същ сигнал с честота 12Hz с четири постепенно намалящи честоти на дискретизация, примерно 100Hz, 30Hz, 20Hz, 15Hz. Наблюдавайте кога формата на сигнала започва да се изкривява и кога сигналът изглежда с по-ниска честота.

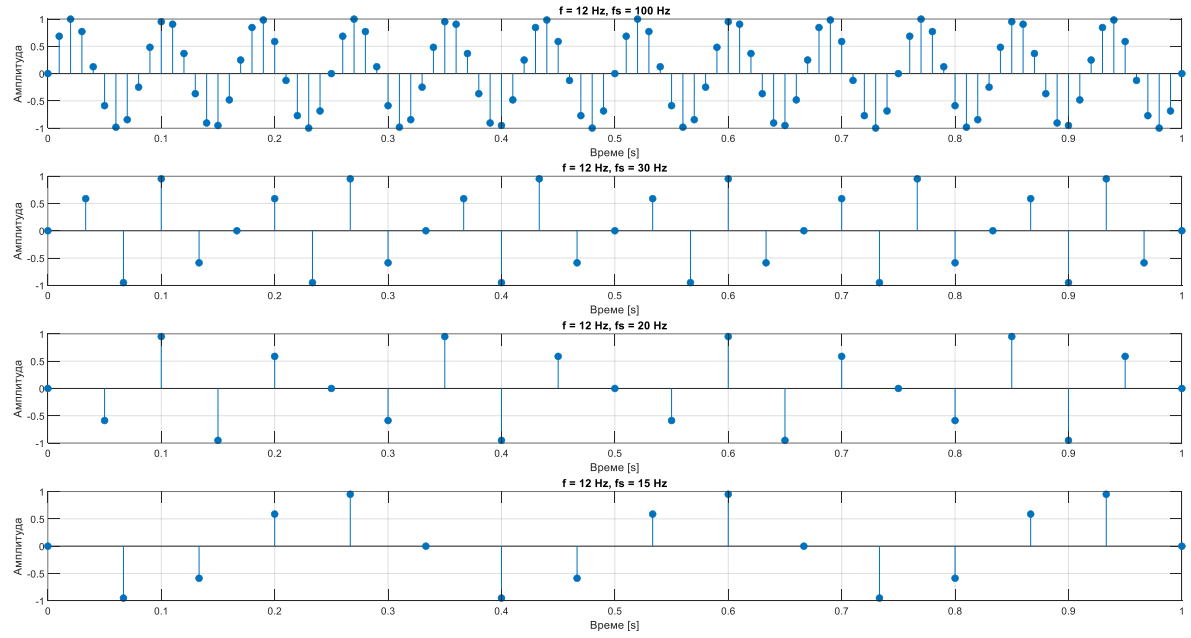
Код в MATLAB:

```
f_sig = 12;        % честота на сигнала
fs_values = [100, 30, 20, 15];    % различни fs
T = 1;

figure;
for k = 1:length(fs_values)
    fs = fs_values(k);
    t = 0:1/fs:T;
    x = sin(2*pi*f_sig*t);

    subplot(length(fs_values),1,k);
    stem(t, x, 'filled');
    grid on;
    title(['f = 12 Hz, fs = ', num2str(fs), ' Hz']);
    xlabel('Време [s]');
    ylabel('Амплитуда');
end
```

```
--- Резултати ---
```




4. Да се визуализира „сгъването“ около честотата на Найкуист.

Код в MATLAB:

```
fs = 40;
fN = fs/2;

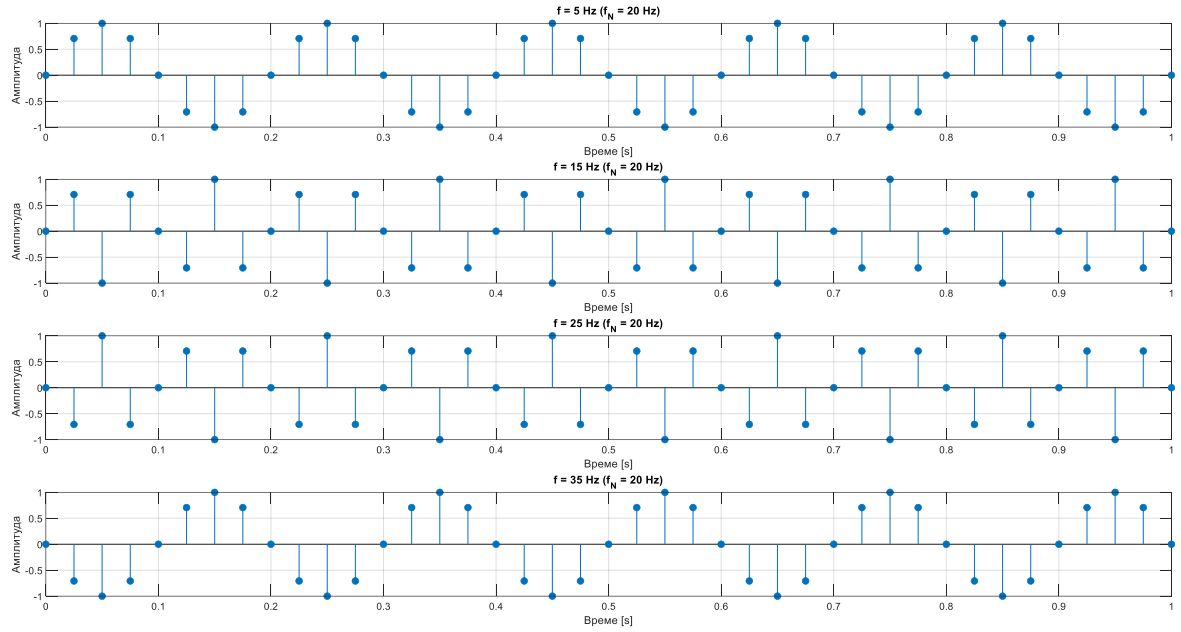
f_vals = [5, 15, 25, 35];    % честоти под и над fN
T = 1;

figure;
for k = 1:length(f_vals)
    t = 0:1/fs:T;
    x = sin(2*pi*f_vals(k)*t);

    subplot(length(f_vals),1,k);
    stem(t, x, 'filled');
    grid on;
```

```
    title(['f = ', num2str(f_vals(k)), ' Hz (f_N = ', num2str(fN), '
Hz)']);
    xlabel('Време [s]');
    ylabel('Амплитуда');
end
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Увеличете честотата на сигнала и повторете експериментите.
2. Изследвайте граничния случай $f_s = 2f$.
3. Добавете шум към сигналите и анализирайте резултатите.

## VI Контролни въпроси

1. Каква е разликата между непрекъснат и дискретен сигнал?
2. Какво представлява честотата на Найкуист?
3. Каква е същността на алиасинга и защо е нежелано изкривяване?
4. Как влияе честотата на дискретизация върху точността на представяне?

**Упражнение №3**

# Линейни дискретни инвариантни във времето (LTI) системи – анализ във времевата област

**I Цел на упражнението** – Дефиниране на линейна дискретна инвариантна във времето (LTI) система. Определяне на импулсната характеристика на дискретна система и реализиране на LTI система чрез диференчно уравнение. Прилагане на продукта MATLAB за анализиране на поведението на система спрямо различни входни сигнали и прилагане на операцията `convolution`.

## II Теоретична част

*2.1. Дискретни системи*

Дискретната система преобразува входен дискретен сигнал $x[n]$ в изходен сигнал $y[n]$:

$$y[n] = T\{x[n]\}$$

Зависимостите между изходните и входните сигнали на една цифрова система с един вход, където x(n) е входният сигнал, и един изход - y(n) изходният сигнал, определят следните им свойства и характеристики:




Фиг. 1. Цифрова система с един вход и един изход

*2.2. Линейност и инвариантност във времето*

- *Линейност*

Дефинира се при нулеви начални условия. Ако при входен сигнал $x_1(n)$ изходният сигнал на цифровата система е $y_1(n)$, а при входен сигнал $x_2(n)$ се получава изходен $y_2(n)$, то цифровата система е линейна. При подаване на входен сигнал от вида

$$x(n) = a_1 \cdot x_1(n) + a_2 \cdot x_2(n),$$

изходният ще бъде

$$y(n) = a_1 \cdot y_1(n) + a_2 \cdot y_2(n).$$

- *Инвариантност във времето*

Цифровата система е инвариантна във времето, ако при входен сигнал $x(n)$ се получава изходен сигнал $y(n)$, а при входен сигнал $x(n-k)$ изходният е $y(n-k)$, т.е. изместването на входния сигнал води до същото такова изместване на изходния й сигнал. Или, системата е инвариантна във времето, ако начинът на реагиране при подаване на определен входен сигнал не зависи от момента, в който сигналът се подава (характеризира се с постоянство на параметрите).

*2.3. Импулсната характеристика*

Импулсната характеристика (ИХ) на цифрова, линейна и инвариантна във времето система е изходният й сигнал $y(n)$ при подаден входен сигнал цифров единичен импулс (функция на Кронекер) и нулеви начални условия.

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \backslash \neq 0 \end{cases}$$

Между изходния сигнал на системата $y(n)$, входния й сигнал $x(n)$ и импулсната й характеристика $h(n)$ съществува следната зависимост:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k).h(n-k) = \sum_{k=-\infty}^{\infty} x(n-k).h(k) = x(n).h(n)\,,$$

Следователно изходният сигнал $y(n)$ на системата се получава чрез операцията конволюция (свиване) на редиците $x(n)$ и $h(n)$, която представлява сума от произведенията на двете изместени във времето редици.

*2.4. Устойчивост и физическа реализуемост*

- *Устойчивост*

Устойчивостта на една система е свързана с нейната реакция при подаване на ограничено входно въздействие. Една система е устойчива, ако след прекратяване на входното въздействие се стреми да се установи в първоначалното си състояние.

- *Физическа реализуемост*

Една ЛДИВС е физически реализуема, ако нейната импулсна характеристика удовлетворява зависимостта: h(n)=0, n<0. При физически реализуемата система измененията на изходния сигнал y(n) не могат да предшестват измененията на входния сигнал x(n).

*2.5. Дискретни във времето системи във времевата област*

Дискретните във времето системи се описват във времевата област с помощта на диференчни уравнения (ДУ), чиито променливи са входните и изходните сигнали на системата, представени чрез числови редици.

Връзката между изходния и входния сигнал на ЛДИВС е:

$$y(n) = f\,[x(n), x(n-1), x(n-2), \ldots, x(n-M), y(n-1), y(n-2), \ldots, y(n-N)]$$

или, изходният сигнал в момента $n$ зависи от входното въздействие в този момент $x(n)$, от предходни негови стойности, а също така и от предходни стойности на самия изходен сигнал $y(n-1), y(n-2), \ldots, y(n-N)$. При ЛДИВС изходният сигнал е линейна комбинация от указаните съставки, като коефициентите, които ги определят, са постоянни:

$$y(n) = \sum_{i=0}^{M} b_i.x(n-i) - \sum_{i=1}^{N} a_i.y(n-i)\,, \quad \text{n} \geq 0,$$

където $a_i$ и $b_i$ са постоянни коефициенти. Първата сума определя нерекурсивната част на ДУ, а втората - рекурсивната. Редът на ДУ се определя от по-голямото от числата M и N. Ако системата е нерекурсивна, то ИХ е крайна. Наличието на рекурсивна част определя рекурсивна система, която в общия случай е с безкрайна ИХ. Наличието на нерекурсивна част може да доведе до неустойчивост на системата.

За изчисляването на $y(0)$ са необходими началните условия, т.е. $x(-i), i = 1, \ldots M$ и $y(-i), i = 1, \ldots N$. Обикновено те се приемат за нулеви.

От теоретична гледна точка недостатък на определянето на изходния сигнал чрез ДУ е необходимостта от последователно изчисляване на реакцията, започвайки от начално състояние до достигане на желаните моменти, което може да изисква голям брой изчисления.

По ДУ могат да се построят и структурни схеми на ЛДИВС, и обратното. Използваните схемни означения за представяне на структурните схеми са: суматори, умножители и клетки памет ($z^{-1}$).

## III Подготовка за упражнението

- **`conv()`** – извършва дискретна конволюция между два сигнала.
- **`stem()`** – визуализира дискретни сигнали.
- **`plot()`** – изчертава сигнал като линия.
- **`zeros()`** – създава вектор от нули.
- **`length()`** – определя дължината на вектор.
- **`filter()`** – реализира LTI система по диференчно уравнение.

## IV Задачи за изпълнение

1. Да се синтезира линейна дискретна инвариантна във времето (LTI) система, чиято импулсна характеристика е експоненциално затихваща функция. Да се дефинира дискретна времева ос $n = 0,1,2, \ldots ,20$. Да се създаде единичен импулс $\delta[n]$, дефиниран за $n = 0$. Да се генерира импулсната характеристика на системата от вида $h[n] = a^n,\ 0 < a < 1$, като се използва стойност $a = 0.8$.

Код в MATLAB:

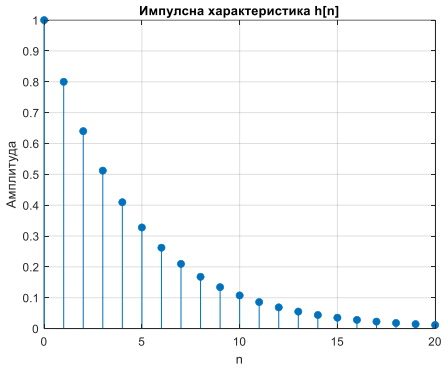
```
n = 0:20;

delta = zeros(size(n));
delta(1) = 1;      % δ[0]

h = (0.8).^n;      % експоненциално затихваща импулсна характеристика

figure;
stem(n, h, 'filled');
grid on;
title('Импулсна характеристика h[n]');
xlabel('n');
ylabel('Амплитуда');
```

```
--- Резултати ---
```




2. Да се определи изходният сигнал на LTI система чрез операцията конволюция. Да се създаде входен сигнал $x[n]$, представляващ единична стъпкова функция. Да се използва импулсната характеристика $h[n]$, дефинирана в Задача 1. Да се изчисли изходният сигнал $y[n]$ чрез дискретна свивка: $y[n] = x[n] * h[n]$. Да се определят индексите на изходния сигнал, като се вземе предвид увеличената дължина след конволюцията. Да се визуализират входният сигнал, импулсната характеристика и изходният сигнал в един графичен прозорец.

Код в MATLAB:

```
x = ones(size(n));      % стъпкова функция

y = conv(x, h);

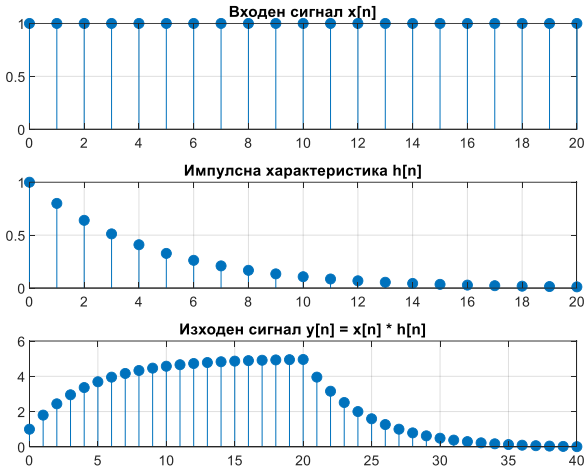
ny = 0:length(y)-1;

figure;
subplot(3,1,1);
stem(n, x, 'filled');
grid on;
title('Входен сигнал x[n]');

subplot(3,1,2);
stem(n, h, 'filled');
grid on;
title('Импулсна характеристика h[n]');

subplot(3,1,3);
stem(ny, y, 'filled');
grid on;
title('Изходен сигнал y[n] = x[n] * h[n]');
```

```
--- Резултати ---
```




3. Да се реализира същата LTI система, използвайки нейното диференчно уравнение. Да се запише ДУ на системата във вида:

$$y[n] = x[n] + 0.8\,y[n-1]$$

Да се реализира системата в MATLAB чрез функцията `filter()`. Да се изчисли изходният сигнал при същия вход $x[n]$, използван в Задача 2. Сравнете резултатите, получени чрез `conv()` и `filter()`.

Код в MATLAB:

```
x = ones(size(n));      % стъпкова функция

y = conv(x, h);
```

```
ny = 0:length(y)-1;

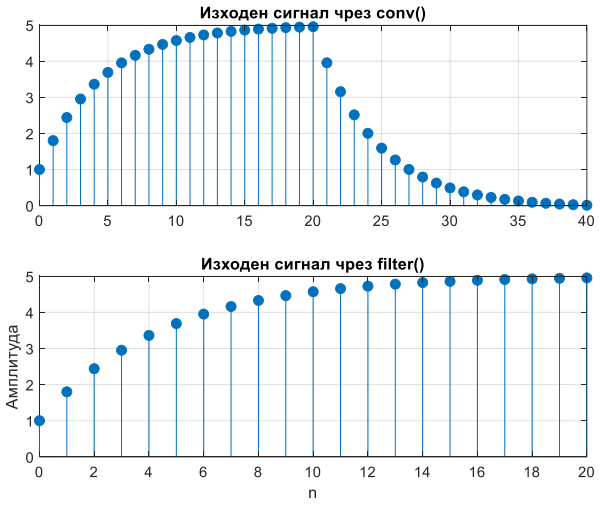
b = 1;        % коефициенти на входа
a = [1 -0.8];

y_filt = filter(b, a, x);


figure;
subplot(2,1,1);
stem(ny, y, 'filled');
grid on;
title('Изходен сигнал чрез conv()');

subplot(2,1,2);
stem(n, y_filt, 'filled');
grid on;
title('Изходен сигнал чрез filter()');
xlabel('n');
ylabel('Амплитуда');
```

```
--- Резултати ---
```




4. Да се изследва поведението на една и съща LTI система при различни входни въздействия. Използвайте системата, реализирана в предходните задания. Генерирайте следните входни сигнали:
   a. единичен входен импулс;
   b. единично входно стъпало;
   c. синусоидален сигнал с дискретна честота $0.3\pi$ rad/sample.

   За всеки входен сигнал да се определи съответният изходен чрез функцията `filter()`. Да се визуализира входът и изходът на системата. Анализирайте влиянието на вида на входния сигнал върху поведението на системата.

Код в MATLAB:

```
%% 1) Параметри
N = 40;              % брой отсчети (n = 0...N)
n = 0:N;
```

```
a1 = 0.8;          % коефициент за обратна връзка

% Коефициенти за filter(): y = (b/a) * x
% y[n] - 0.8 y[n-1] = x[n]  =>  a = [1, -0.8], b = [1]
b = 1;
a = [1, -a1];

%% 2) Дефиниране на входни сигнали

% (a) Единичен импулс δ[n]
x_imp = zeros(size(n));
x_imp(1) = 1;

% (b) Единична стъпка u[n]
x_step = ones(size(n));

% (c) Синусоидален сигнал: x[n] = sin(0.3*pi*n)
w0 = 0.3*pi;              % дискретна ъглова честота [rad/sample]
x_sin = sin(w0*n);

%% 3) Изчисляване на изходите чрез filter()
y_imp  = filter(b, a, x_imp);
y_step = filter(b, a, x_step);
y_sin  = filter(b, a, x_sin);

%% 4) Визуализация

% (a) Импулсен вход
figure;
stem(n, x_imp, 'filled'); hold on;
stem(n, y_imp, 'r', 'filled');
grid on;
title('Реакция на LTI система при импулсен вход');
xlabel('n'); ylabel('Амплитуда');
legend('x_{imp}[n] = \delta[n]', 'y_{imp}[n]');

% (b) Стъпков вход
figure;
stem(n, x_step, 'filled'); hold on;
stem(n, y_step, 'r', 'filled');
grid on;
title('Реакция на LTI система при стъпков вход');
xlabel('n'); ylabel('Амплитуда');
legend('x_{step}[n] = u[n]', 'y_{step}[n]');

% (c) Синусоидален вход
figure;
stem(n, x_sin, 'filled'); hold on;
stem(n, y_sin, 'r', 'filled');
grid on;
title('Реакция на LTI система при синусоидален вход');
xlabel('n'); ylabel('Амплитуда');
legend('x_{sin}[n] = sin(0.3\pi n)', 'y_{sin}[n]');

%% 5) Сравнение в един прозорец
figure;

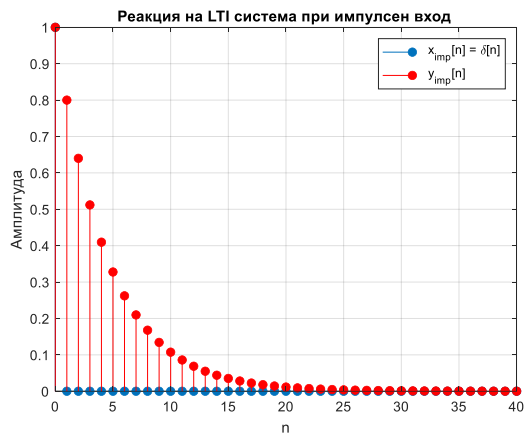
subplot(3,1,1);
stem(n, x_imp, 'filled'); hold on;
stem(n, y_imp, 'r', 'filled');
grid on;
```

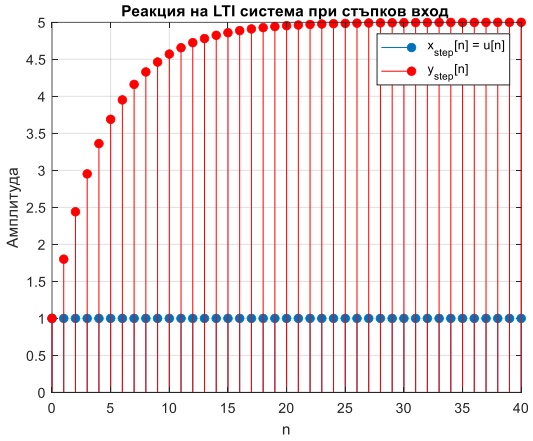
```
title('Импулсен вход');
xlabel('n'); ylabel('Амплитуда');
legend('x[n]', 'y[n]');

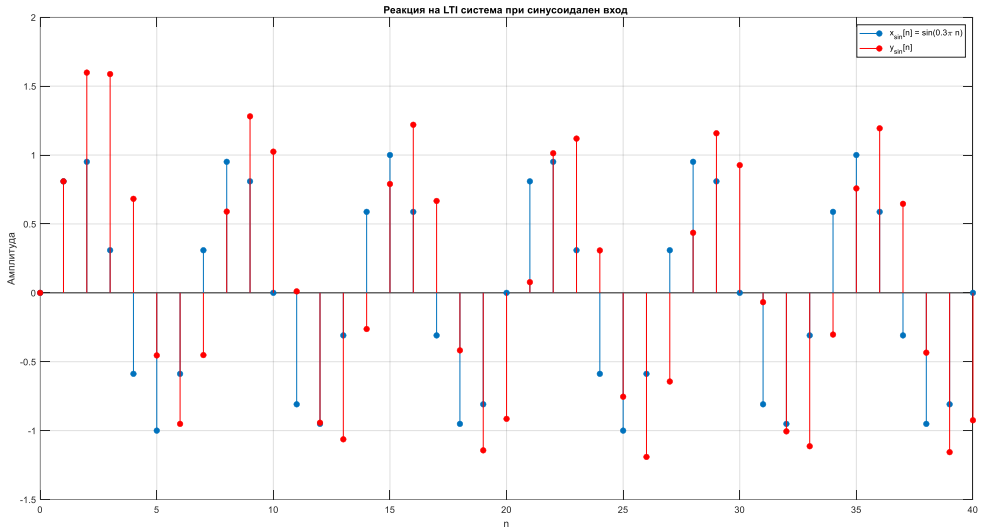
subplot(3,1,2);
stem(n, x_step, 'filled'); hold on;
stem(n, y_step, 'r', 'filled');
grid on;
title('Стъпков вход');
xlabel('n'); ylabel('Амплитуда');
legend('x[n]', 'y[n]');

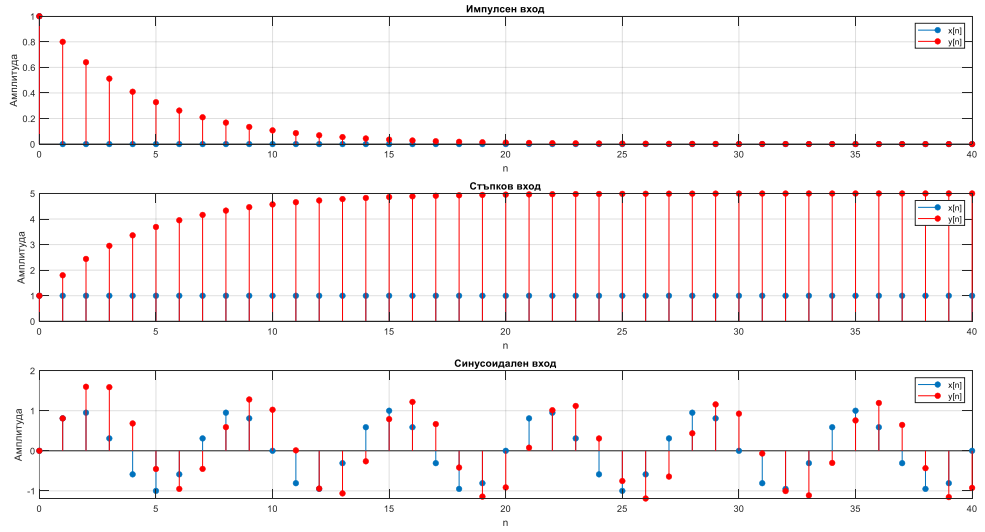
subplot(3,1,3);
stem(n, x_sin, 'filled'); hold on;
stem(n, y_sin, 'r', 'filled');
grid on;
title('Синусоидален вход');
xlabel('n'); ylabel('Амплитуда');
legend('x[n]', 'y[n]');
```

```
--- Резултати ---
```






## V Допълнителни задачи

1. В задача 3 променете коефициента 0.8 и наблюдавайте устойчивостта.
2. В задачи 3 и 4 използвайте правоъгълен импулс като входен.
3. Сравнете резултатите при различна дължина на сигнала за задачи 3 и 4.

## VI Контролни въпроси

1. Каква е същността на импулсната характеристика?
2. Какво определят линейността и инвариантността във времето на LTI?
3. Каква е ролята на функцията конволюция?
4. Какво описва диференчното уравнение?

**Упражнение №4**

# Линейни дискретни инвариантни във времето (LTI) системи – анализ във честотната област

**I Цел на упражнението** – Определяне на връзката между диференчно уравнение и предавателна функция $H(z)$. Намиране и интерпретиране на честотна характеристика $H(e^{j\omega})$, анализиране на амплитудна и фазова характеристика на LTI система. Прилагане на продукта MATLAB за експериментален анализ на честотната характеристика чрез синусоидални входни сигнали и сравняване на поведението на системата във времевата и в честотната област.

**II Теоретична част**

*2.1. Предавателна функция*

Дискретна LTI система често се задава със следното уравнение:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$$

След $Z$ - преобразуване се получава предавателната функция:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}}$$

*2.2. Честотна характеристика*

Честотната характеристика се получава при $z = e^{j\omega}$

$$H\left(e^{j\omega}\right) = H(z)|_{z=e^{j\omega}}$$

Тя е комплексна функция и описва:

- амплитудно усилване/затихване:

$$\left|H(e^{j\omega})\right|$$

- фазово отместване**:**

$$\angle H(e^{j\omega})$$

*2.3. Физически смисъл*

За синусоидален вход $x[n] = A\sin\omega n$ (или $Ae^{j\omega n}$) изходът на LTI система в установен режим е синусоида със същата честота, но с променена амплитуда и фаза:

$$y[n] \approx A\left|H\left(e^{j\omega}\right)\right| \sin\left(\omega n + \angle H\left(e^{j\omega}\right)\right).$$

Това позволява системата да се разглежда като честотен филтър, който потиска или пропуска определени честоти.

*2.4. Примерна система*

Използва се представената в упражнение 3 система:

$$y[n] = x[n] + 0.8y[n-1],$$

която може да се запише като:

$$y[n] - 0.8y[n-1] = x[n].$$

Тогава

$$H(z) = \frac{1}{1 - 0.8z^{-1}}$$

и

$$H\left(e^{j\omega}\right) = \frac{1}{1 - 0.8e^{-j\omega}}$$

## III Подготовка за упражнението

*3.1 Функции за анализ:*

- `freqz(b,a,N)` – изчислява и визуализира честотната характеристика $H\left(e^{j\omega}\right)$ на система с коефициенти $b, a$.
- `impz(b,a,N)` – изчислява импулсната характеристика $h[n]$ за зададен брой дискрети.
- `filter(b,a,x)` – пресмята изхода на LTI система за вход $x[n]$, описана чрез коефициенти $b, a$.
- `grpdelay(b,a,N)` – изчислява групово закъснение.

*3.2 Функции за работа с комплексни числа:*

- `abs()` – връща модула на комплексно число/вектор.
- `angle()` – връща фазовия ъгъл на комплексно число/вектор (в радиани).
- `unwrap()` – премахва фазови скокове $\pm\pi$ за по-гладка фазова характеристика.

*3.3 Визуализация и помощни функции:*

- `plot(), stem(), subplot(), grid on, hold on, legend(), title(), xlabel(), ylabel()` – визуализация и оформление.
- `zeros(), ones(), length(), num2str()` – помощни функции за вектори и текст.

## IV Задачи за изпълнение

1. За системата $y[n] = x[n] + 0.8y[n-1]$ определете коефициентите $b$ и $a$ и изчислете импулсната характеристика $h[n]$ за $n = 0 \ldots 30$. Визуализирайте $h[n]$.

Код в MATLAB:

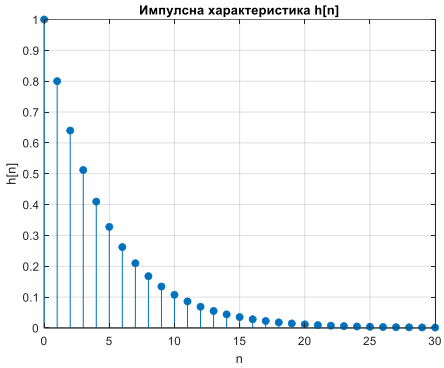
```
N = 30;
n = 0:N;

a1 = 0.8;
b = 1;
a = [1, -a1];

[h, nh] = impz(b, a, N+1);

figure;
stem(nh, h, 'filled');
grid on;
title('Импулсна характеристика h[n]');
xlabel('n'); ylabel('h[n]');
```

```
--- Резултати ---
```




2. Изчислете и визуализирайте амплитудната $\left|H(e^{j\omega})\right|$ и фазовата $\angle H(e^{j\omega})$ характеристика на системата от задача 1.

Код в MATLAB:

```
a1 = 0.8;
b = 1;
a = [1, -a1];

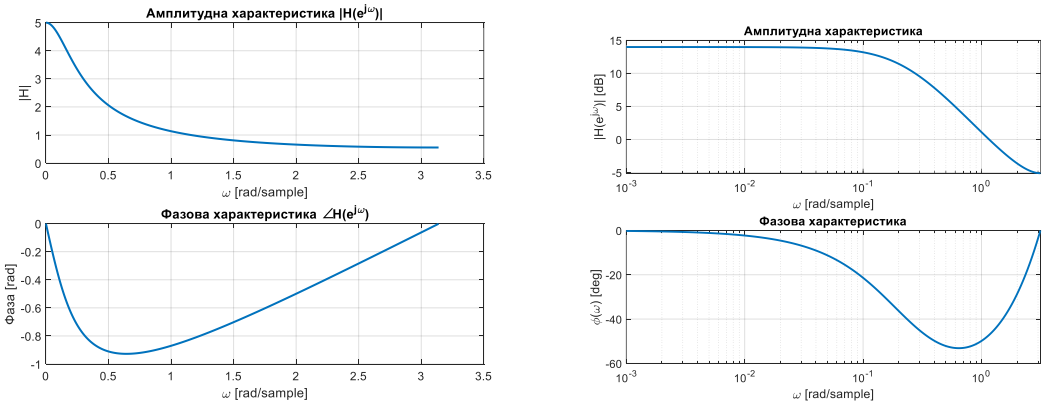
Nw = 1024;                    % брой честотни точки
[H, w] = freqz(b, a, Nw);     % w е в rad/sample (0..pi)

magH = abs(H);
phH  = unwrap(angle(H));

figure;
subplot(2,1,1);
plot(w, magH, 'LineWidth', 1.5);
grid on;
title('Амплитудна характеристика |H(e^{j\omega})|');
xlabel('\omega [rad/sample]'); ylabel('|H|');

subplot(2,1,2);
plot(w, phH, 'LineWidth', 1.5);
grid on;
title('Фазова характеристика \angleH(e^{j\omega})');
xlabel('\omega [rad/sample]'); ylabel('Фаза [rad]');
```

```
--- Резултати ---
```



3. Подайте на системата синусоидален входен сигнал $x[n] = \sin(\omega n)$ за две различни честоти $\omega 1 = 0.2$ и $\omega 2 = 0.7$. След преминаване на преходния процес, определете:
   - отношението на амплитудите $A_{out}/A_{in}$;
   - фазовото отместване.

Код в MATLAB:

```
clc; clear; close all;

a1 = 0.8;
b = 1;
a = [1, -a1];

N = 400;
n = 0:N;

w_test = [0.2*pi, 0.7*pi];    % две честоти за тест
settle = 150;                 % пропускаме началните Nsettle проби (преходен
режим)

for k = 1:length(w_test)
    w0 = w_test(k);

    x = sin(w0*n);
    y = filter(b, a, x);

    % Изрязваме установен режим
    n_ss = n(settle:end);
    x_ss = x(settle:end);
    y_ss = y(settle:end);

    % Амплитуди (оценка чрез пик-пик/2)
    Ain  = (max(x_ss) - min(x_ss))/2;
    Aout = (max(y_ss) - min(y_ss))/2;

    % Фаза: намираме лаг (изместване) чрез максимална корелация в установен
режим
    [c, lags] = xcorr(y_ss, x_ss);
    [~, idx] = max(c);
    lag = lags(idx);  % положителен лаг означава, че y изостава спрямо x

    % Приблизително фазово отместване: phi ≈ -lag*w0
    phi_est = -lag*w0;

    % Теоретични стойности от freqz за същата честота
    H0 = 1 / (1 - a1*exp(-1j*w0));
    mag_theory = abs(H0);
    phi_theory = angle(H0);

    fprintf('\n--- Тест за w = %.2f*pi rad/sample ---\n', w0/pi);
    fprintf('Оценено усилване Aout/Ain = %.4f | Теоретично |H| = %.4f\n',
Aout/Ain, mag_theory);
    fprintf('Оценена фаза (прибл.)     = %.4f rad | Теоретично angle(H) =
%.4f rad\n', phi_est, phi_theory);

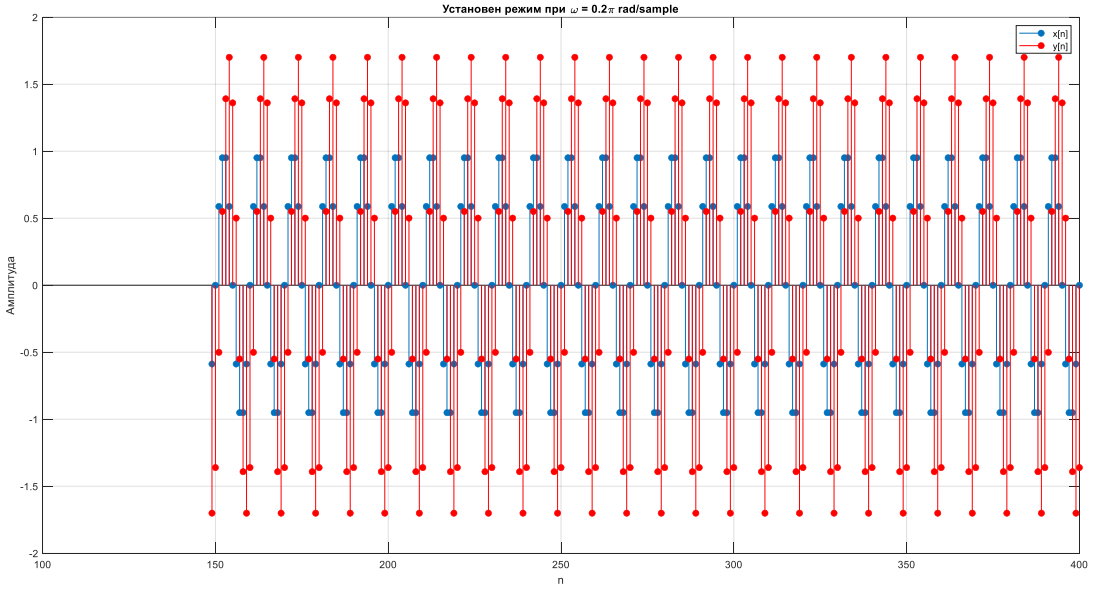
    % Визуализация (установен режим)
    figure;
    stem(n_ss, x_ss, 'filled'); hold on;
    stem(n_ss, y_ss, 'r', 'filled');
```

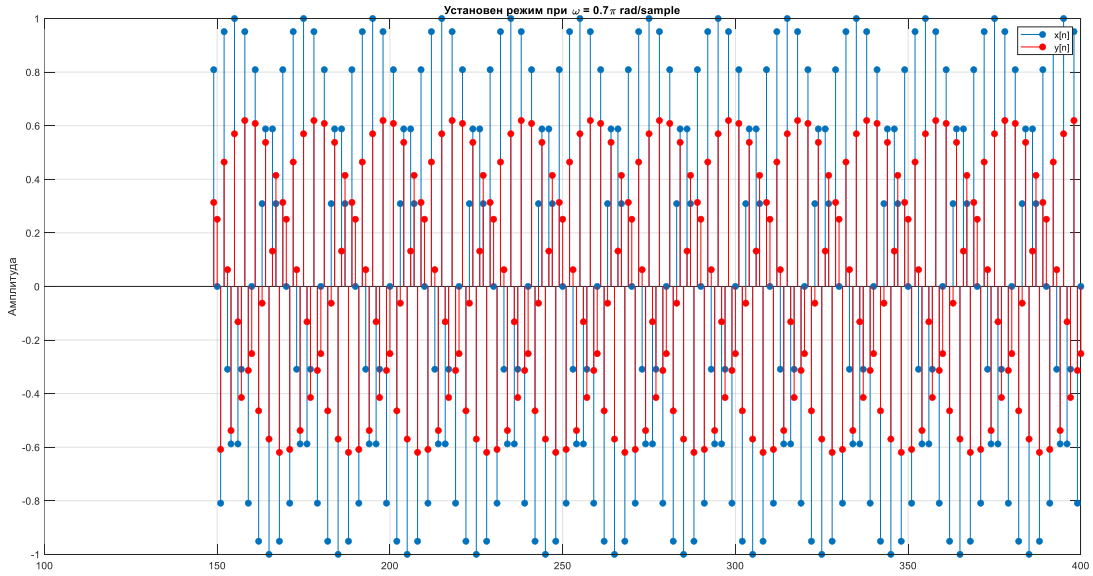
```
    grid on;
    legend('x[n]', 'y[n]');
    title(['Установен режим при \omega = ', num2str(w0/pi), '\pi
rad/sample']);
    xlabel('n'); ylabel('Амплитуда');
end
```

```
--- Резултати ---

--- Тест за w = 0.20*pi rad/sample ---
Оценено усилване Aout/Ain = 1.7884 | Теоретично |H| = 1.7011
Оценена фаза (прибл.)     = -0.6283 rad | Теоретично angle(H) = -0.9271 rad

--- Тест за w = 0.70*pi rad/sample ---
Оценено усилване Aout/Ain = 0.6194 | Теоретично |H| = 0.6225
Оценена фаза (прибл.)     = -6.5973 rad | Теоретично angle(H) = -0.4147 rad
```



4. Променете параметъра $a_1 = [0.2\ 0.5\ 0.8\ 0.95]$ и сравнете амплитудните и фазовите честотни характеристики на системата при всяка от зададените стойности. Представете амплитудните характеристики в dB, а фазовите – в градуси, като използвате логаритмичен мащаб по честотната ос, аналогично на Bode диаграмите. Анализирайте влиянието на параметъра $a_1$ върху честотната селективност и фазовите свойства на системата.

Код в MATLAB:

```
clc;
clear;
close all;

a_vals = [0.2, 0.5, 0.8, 0.95];
Nw = 1024;

figure;
%% Амплитудни характеристики
subplot(2,1,1);
hold on;
grid on;
for k = 1:length(a_vals)

    a1 = a_vals(k);

    b = 1;
    a = [1, -a1];

    [H, w] = freqz(b, a, Nw);

    % Премахване на w = 0
    w_plot = w(2:end);
    H_plot = H(2:end);

    % Амплитуда в dB
    magH_dB = 20*log10(abs(H_plot));

    semilogx(w_plot, magH_dB, 'LineWidth', 1.5);

end
title('Амплитудни характеристики');
xlabel('\omega [rad/sample]');
ylabel('|H(e^{j\omega})| [dB]');
legend('a_1=0.2', 'a_1=0.5', 'a_1=0.8', 'a_1=0.95');
hold off;

%% Фазови характеристики
subplot(2,1,2);
hold on;
grid on;
for k = 1:length(a_vals)

    a1 = a_vals(k);

    b = 1;
    a = [1, -a1];

    [H, w] = freqz(b, a, Nw);
```

```
    % Премахване на w = 0
    w_plot = w(2:end);
    H_plot = H(2:end);

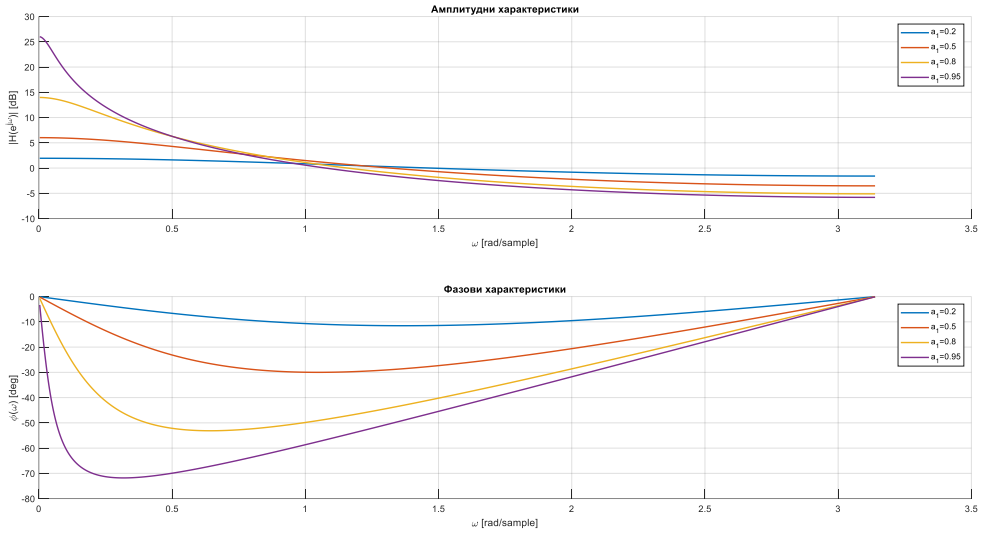
    % Фаза в градуси
    phH_deg = unwrap(angle(H_plot))*180/pi;

    semilogx(w_plot, phH_deg, 'LineWidth', 1.5);

end
title('Фазови характеристики');
xlabel('\omega [rad/sample]');
ylabel('\phi(\omega) [deg]');
legend('a_1=0.2', 'a_1=0.5', 'a_1=0.8', 'a_1=0.95');

hold off;
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Повторете задания 1–2 за следните системи (ДУ):
   - $y(n) = x(n) - 0.5\,x(n-1) - 0.8\,x(n-2) - 0.25\,y(n-2)$
   - $y(n) = x(n) - 0.8\,x(n-1) - 0.3\,x(n-2) - 0.5y(n-1)$
   - $y(n) = x(n) - 0.4\,x(n-1) + 0.1y(n-1) - 0.2\,x(n-2)$

## VI Контролни въпроси

1. Какво представлява предавателната функция $H(e^{j\omega})$ и защо се разглежда при $z = e^{j\omega}$?
2. Какъв е физическият смисъл на $|H(e^{j\omega})|$ и $\angle H(e^{j\omega})$?
3. Как може експериментално да се получи честотната характеристика на система?

**Упражнение №5**

# Дискретно преобразуване на Фурие (DFT). Спектрален анализ на дискретни сигнали

**I Цел на упражнението** – Приложение на продукта MATLAB за разлагане на сложен дискретен сигнал с дискретно преобразуване на Фурие. Интерпретиране на амплитудния и фазовия спектър, анализиране на спектъра на периодични и сложни сигнали, изследване на влиянието на броя дискрети върху спектралната резолюция.

## II Теоретична част

В предходните лабораторни упражнения сигналите и системите бяха разглеждани основно във времевата област, т.е. като функции на времето или дискретния индекс $n$. В много практически приложения (комуникации, аудио, вибрации, изображения) обаче е по-важно да се знае какви честоти съдържа сигналът, отколкото как изглежда той във времето.

Основната идея на преобразуването на Фурие е, че всеки сигнал може да бъде представен като сума от постояннотокова съставяща и хармонични (синусоидални) компоненти. За непрекъснати сигнали това се реализира чрез непрекъснатото преобразуване на Фурие, а за дискретни и крайни по дължина сигнали — чрез дискретното преобразуване на Фурие (DFT).

DFT представя сигнала като линейна комбинация от комплексни експоненти с дискретен набор от честоти.

*2.1. Дискретно преобразуване на Фурие (DFT)*

Нека е даден дискретен сигнал:

$$x[n], \qquad n = 0,1,\dots,N-1$$

DFT на този сигнал се дефинира като:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{j\frac{2\pi}{N}kn}, \qquad k = 0,1,\dots,N-1$$

където: $N$ е броят отсчети на сигнала, $k$ е честотният индекс, а $X[k]$ е комплексната стойност на спектъра при честота $k$.

DFT може да се разглежда като проекция на сигнала върху набор от ортогонални комплексни експоненти.

Правото дискретно преобразуване на Фурие (DFT) се дефинира като:

$$X(k) = \sum_{k=0}^{N-1} x(n).e^{-j\frac{2\pi}{N}nk}, 0 \le k \le N.$$

където $k$ е номера на хармоничната съставяща.

Обратното дискретно преобразуване на Фурие (IDFT) се дефинира чрез:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).e^{j\frac{2\pi}{N}nk}, 0 \le n < N.$$

Изразите за правото и обратното дискретно преобразуване на Фурие могат да се представят чрез комплексния фазов фактор

$$W_N = e^{-j\frac{2\pi}{N}} = \cos\left(\frac{2\pi}{N}\right) - j.\sin\left(\frac{2\pi}{N}\right),$$

при което дискретно преобразуване на Фурие придобива вида:

$$X(k) = \sum_{k=0}^{N-1} x(n).W_N^{nk},\ 0 \le k < N\ ,$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).W_N^{-nk},\ 0 \le n < N\ .$$

Последните две зависимости за правото и обратното дискретно преобразуване на Фурие се различават само по нормиращия коефициент $1/N$ и знака в степента на $W_N$.

*2.2. Комплексна интерпретация на DFT*

Всеки коефициент $X[k]$ е комплексно число и съдържа две важни части:

- амплитуда (модул):

$$|X[k]|$$

- фаза:

$$\angle|X[k]|$$

Тези две величини определят приноса на съответната честотна компонента към сигнала.

За реални сигнали спектърът е комплексно симетричен

$$X[k] = X*[N-k],$$

поради това често е достатъчно да се анализира само половината от спектъра.

*2.3. Честотна ос и физически смисъл на индекса* $\boldsymbol{k}$

DFT не дава честотите директно в Hz, а чрез индекс $k$. Връзката между индекса и реалната честота е:

$$f_k = \frac{k}{N} f_S,$$

където $f_S$ е честотата на дискретизация, а $f_k$ е честотата, съответстваща на спектралния коефициент $X[k]$.

Минималната честотна стъпка (резолюция) е:

$$\Delta f = \frac{f_s}{N}$$

*2.4. Честотна резолюция и влияние на броя дискрети*

Честотната резолюция определя колко близки честоти могат да бъдат разграничени в спектъра.

- Малък $N \rightarrow$ груб спектър, малко честотни линии;
- Голям $N \rightarrow$ по-фина честотна стъпка и по-детайлен спектър.

Важно е да се отбележи, че увеличаването на $N$ не добавя нова информация, а подобрява честотната дискретизация.

*2.5. Периодичност, интерпретация и изчислителна сложност на DFT*

DFT приема, че сигналът $x[n]$ е един период на периодичен сигнал с период $N$. Това означава, че рязко прекъсване на сигнала може да доведе до изкривявания в спектъра.

Директното изчисляване на DFT чрез дефиницията изисква $\mathcal{O}(N^2)$ операции, което е бавно за големи $N$. Това е основната причина в практиката да се използва FFT (Fast Fourier Transform) — бърз алгоритъм за изчисляване на DFT, който ще бъде разгледан в следващото упражнение.

DFT е фундаментален инструмент, използван за спектрален анализ на сигнали, анализ на шум, проектиране и анализ на филтри, компресия на данни, обработка на изображения и аудио. Разбирането на DFT е ключово за правилна интерпретация на резултатите от FFT и всички последващи методи.

## III Подготовка за упражнението

*Функции за анализ:*

- `exp()` – изчислява комплексна експонента.
- `abs()` – изчислява модул на комплексно число.
- `angle()` – изчислява фазов ъгъл.
- `stem()` – визуализира дискретни спектри.
- `zeros()`, `length()` – работа с вектори.

## IV Задачи за изпълнение

1. Да се изчисли дискретното преобразуване на Фурие на синусоидален сигнал чрез директно прилагане на дефиницията за DFT посредством следните действия:
   - Да се генерира синусоидален сигнал с честота $f = 5$ Hz.
   - Да се зададе честота на дискретизация $f_s = 64$ Hz и $N = 64$.
   - Да се изчисли DFT чрез вложени цикли.
   - Да се визуализира амплитудният спектър.

Код в MATLAB:

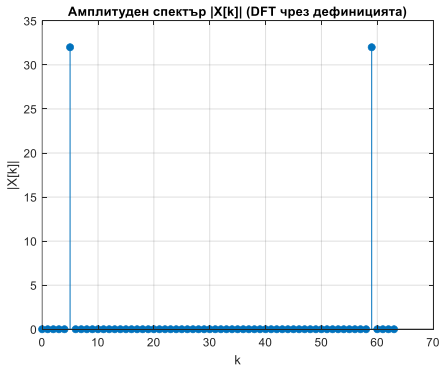
```
fs = 64; N = 64; n = 0:N-1; f = 5;

x = sin(2*pi*f*n/fs);
X = zeros(1, N);

for k = 0:N-1
    for nn = 0:N-1
        X(k+1) = X(k+1) + x(nn+1)*exp(-1j*2*pi*k*nn/N);
    end
end

k = 0:N-1;
figure;
stem(k, abs(X), 'filled');
grid on;
title('Амплитуден спектър |X[k]| (DFT чрез дефиницията)');
xlabel('k'); ylabel('|X[k]|');
```

```
--- Резултати ---
```



2. Да се изследва спектърът на сигнал, съставен от сума на две синусоиди:

$$x[n] = \sin(2\pi f_1 n/f_s) + 0.5\sin(2\pi f_2 n/f_s)$$

Да се изчисли DFT и да се идентифицират спектралните отсчети.

Код в MATLAB:

```
fs = 64;
N = 64;
n = 0:N-1;

f1 = 5;
f2 = 12;

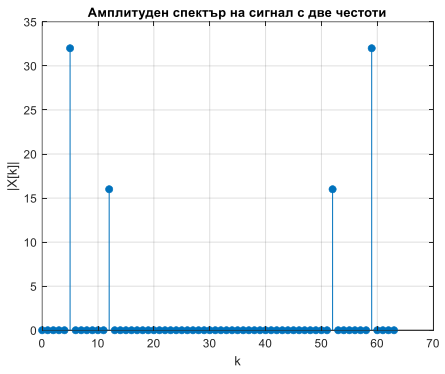
x = sin(2*pi*f1*n/fs) + 0.5*sin(2*pi*f2*n/fs);

X = zeros(1, N);
for k = 0:N-1
    for nn = 0:N-1
        X(k+1) = X(k+1) + x(nn+1)*exp(-1j*2*pi*k*nn/N);
    end
end

k = 0:N-1;

figure;
stem(k, abs(X), 'filled');
grid on;
title('Амплитуден спектър на сигнал с две честоти');
xlabel('k'); ylabel('|X[k]|');
```

```
--- Резултати ---
```




3. Подайте на системата синусоидален входен сигнал $x[n] = \sin(\omega n)$ за две различни честоти $\omega 1 = 0.2rad/s$ и $\omega 2 = 0.7rad/s$. След преминаване на преходния процес, измерете:
   - отношението на амплитудите $A_{out}/A_{in}$;
   - фазовото отместване.

Код в MATLAB:

```
clc; clear; close all;

a1 = 0.8;
b = 1;
a = [1, -a1];

N = 400;
n = 0:N;
```

```
w_test = [0.2*pi, 0.7*pi];   % две честоти за тест
settle = 150;                % пропускаме началните Nsettle проби (преходен
режим)

for k = 1:length(w_test)
    w0 = w_test(k);

    x = sin(w0*n);
    y = filter(b, a, x);

    % Изрязваме установен режим
    n_ss = n(settle:end);
    x_ss = x(settle:end);
    y_ss = y(settle:end);

    % Амплитуди (оценка чрез пик-пик/2)
    Ain  = (max(x_ss) - min(x_ss))/2;
    Aout = (max(y_ss) - min(y_ss))/2;

    % Фаза: намираме лаг (изместване) чрез максимална корелация в установен
режим
    [c, lags] = xcorr(y_ss, x_ss);
    [~, idx] = max(c);
    lag = lags(idx);  % положителен лаг означава, че y изостава спрямо x

    % Приблизително фазово отместване: phi ≈ -lag*w0
    phi_est = -lag*w0;

    % Теоретични стойности от freqz за същата честота
    H0 = 1 / (1 - a1*exp(-1j*w0));
    mag_theory = abs(H0);
    phi_theory = angle(H0);

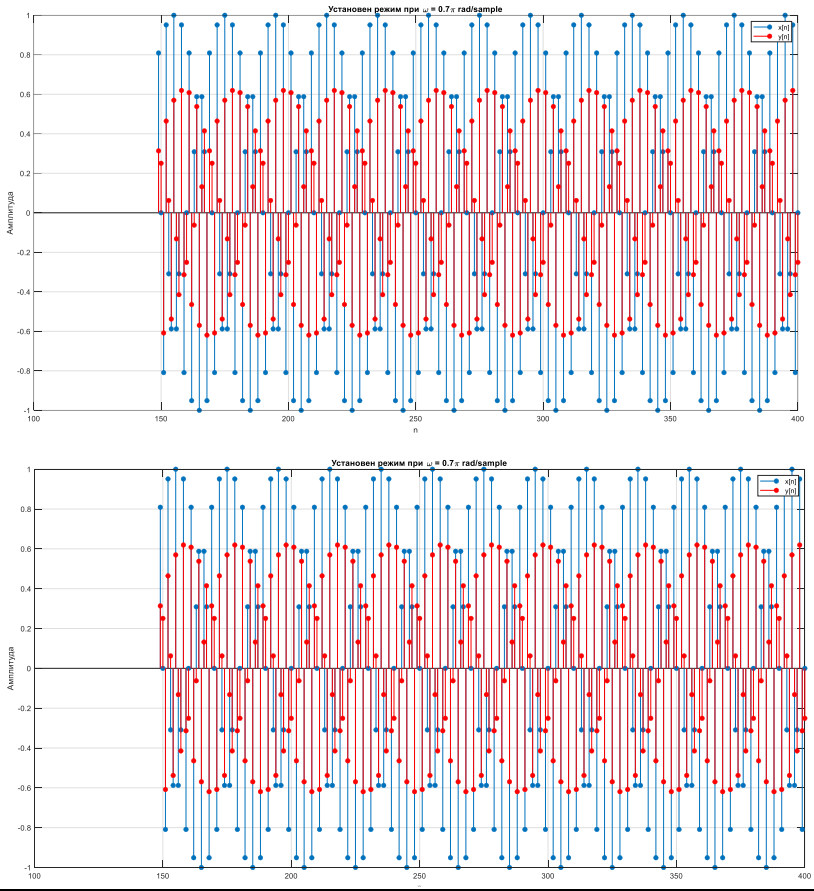
    fprintf('\n--- Тест за w = %.2f*pi rad/sample ---\n', w0/pi);
    fprintf('Оценено усилване Aout/Ain = %.4f | Теоретично |H| = %.4f\n',
Aout/Ain, mag_theory);
    fprintf('Оценена фаза (прибл.)     = %.4f rad | Теоретично angle(H) =
%.4f rad\n', phi_est, phi_theory);

    % Визуализация (установен режим)
    figure;
    stem(n_ss, x_ss, 'filled'); hold on;
    stem(n_ss, y_ss, 'r', 'filled');
    grid on;
    legend('x[n]', 'y[n]');
    title(['Установен режим при \omega = ', num2str(w0/pi), '\pi
rad/sample']);
    xlabel('n'); ylabel('Амплитуда');
end
```

```
--- Резултати ---

--- Тест за w = 0.20*pi rad/sample ---
Оценено усилване Aout/Ain = 1.7884 | Теоретично |H| = 1.7011
Оценена фаза (прибл.)     = -0.6283 rad | Теоретично angle(H) = -0.9271 rad

--- Тест за w = 0.70*pi rad/sample ---
Оценено усилване Aout/Ain = 0.6194 | Теоретично |H| = 0.6225
Оценена фаза (прибл.)     = -6.5973 rad | Теоретично angle(H) = -0.4147 rad
```




3. Да се генерира дискретен синусоидален сигнал с честота $f = 5\,Hz$ и честота на дискретизация $f_s = 64\,Hz$. Да се разгледат три различни стойности за $N \in \{32, 64, 128\}$. За всяка стойност на $N$:
   - да се създаде съответният времеви индекс $n = 0,1, \ldots, N-1$;
   - да се генерира сигналът $x[n] = \sin(2\pi f n / f_s)$
   - да се изчисли дискретното преобразуване на Фурие $X[k]$ чрез директно прилагане на дефиницията за DFT;
   - да се определи честотната резолюция.

   Да се визуализира амплитудният спектър за всяка стойност на N, като резултатите се представят в един общ графичен прозорец.

Код в MATLAB:

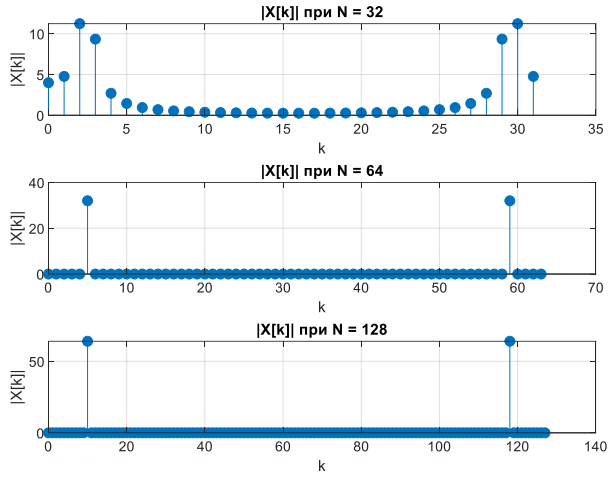
```
Nvals = [32, 64, 128];
f = 5;

figure;
for i = 1:length(Nvals)
    N = Nvals(i);
    n = 0:N-1;
    x = sin(2*pi*f*n/fs);

    X = zeros(1, N);
    for k = 0:N-1
        for nn = 0:N-1
            X(k+1) = X(k+1) + x(nn+1)*exp(-1j*2*pi*k*nn/N);
        end
    end
```

```
    subplot(length(Nvals),1,i);
    stem(0:N-1, abs(X), 'filled');
    grid on;
    title(['|X[k]| при N = ', num2str(N)]);
    xlabel('k'); ylabel('|X[k]|');
end
```

```
Резултати ---
```




## V Допълнителни задачи

1. Изчислете и визуализирайте фазовия спектър $\angle X[k]$.
2. Променете честотата на сигнала така, че тя да бъде различна от честотите, съответстващи на отделните спектрални линии на DFT. Наблюдавайте разпределението на енергията в амплитудния спектър и анализирайте получения резултат.
3. Добавете шум към сигнала и анализирайте спектъра.

## VI Контролни въпроси

1. Какво представлява DFT и каква е разликата с непрекъснатото преобразуване на Фурие?
2. Каква информация съдържа амплитудният спектър?
3. Каква е ролята на N?
4. Защо спектърът на реален сигнал е симетричен?

**Упражнение №6**

# Бързо преобразуване на Фурие (FFT). Спектрален анализ на дискретни сигнали

**I Цел на упражнението** – Определяне на връзката между теоретичното определение на дискретното преобразуване на Фурие (DFT) и реализацията му чрез бързо преобразуване на Фурие (FFT), както и интерпретация на амплитудния и фазовия спектър.

**II Теоретична част**

В предходното упражнение беше разгледано дискретното преобразуване на Фурие (DFT) чрез неговата дефиниция. Макар DFT да дава фундаментална връзка между времевата и честотната област, директното му изчисляване е изчислително скъпо и практически неприложимо при голям брой дискрети.

В реалните системи за цифрова обработка на сигнали (аудио, комуникации, биомедицински сигнали, изображения) спектралният анализ трябва да се извършва бързо и в реално време, което налага използването на по-ефективни алгоритми.

*2.1. Същност на бързото преобразуване на Фурие (FFT)*

Бързото преобразуване на Фурие не представлява ново математическо преобразуване. То е алгоритъм за ефективно изчисляване на DFT, който използва симетрии и периодичности в комплексните експоненти. Това прави възможен спектралния анализ дори за сигнали с хиляди дискрети.

Най-разпространеният клас FFT алгоритми се основава на принципа „разделяй и владей“. Сигналът с дължина $N$ се разделя на по-малки подпоследователности, като DFT на по-малките последователности се изчислява рекурсивно, а резултатите се комбинират чрез $e^{-j2\pi k/N}$.

Алгоритъмът е особено ефективен, когато $N = 2^m$, поради което често на практика се използва *zero-padding* до най-близката степен на 2.

Zero-padding представлява допълване на сигнала с нули до по-голяма дължина $N_{pad}$. Важно е да се подчертае, че zero-padding не добавя нова честотна информация. Той води до по-гъсто дискретизиране на спектъра, при което спектърът изглежда по-гладък и пиковете са по-лесни за локализиране. Това е особено полезно при визуален анализ.

*2.2. FFT в MATLAB*

В MATLAB FFT се реализира чрез функцията `X = fft(x)`, където x е дискретният сигнал, а X е комплексният спектър (DFT коефициентите).

Важно е да се подчертае, че fft() връща нормализиран по индекс, но не по амплитуда спектър, а подредбата на коефициентите съответства на индексите $k = 0,1, \dots, N-1$.

*2.3. Амплитуден и фазов спектър при FFT*

Резултатът от FFT е комплексен вектор. От него се извличат амплитуден спектър - $|X[k]|$ и фазов спектър - $\angle|X[k]|$.

За коректна интерпретация на амплитудите е необходимо мащабиране:

$$|X[k]|_{norm} = \frac{1}{N}|X[k]|$$

При реални сигнали спектърът е симетричен, което позволява използването на едностранен амплитуден спектър. При него честотите над $f_s/2$ не носят нова информация, а амплитудите в положителния честотен диапазон се удвояват. Това води до по-интуитивна визуализация на спектъра и директна връзка с физическите честоти в Hz.

*2.4. Честотна ос и физическа интерпретация*

FFT работи с честотни индекси, но за практическа интерпретация те трябва да се преобразуват в Hz $f_k = \frac{k}{N} f_s$.

Минималната честотна стъпка е $\Delta f = \frac{f_s}{N}$, като тази величина определя честотната резолюция, а не самият FFT алгоритъм.

## III Подготовка за упражнението

*Функции за анализ:*

- `fft()` – изчислява бързо DFT на сигнал.
- `abs()` – изчислява амплитуда на спектъра.
- `angle()` – изчислява фазовия спектър.
- `fftshift()` – пренарежда спектъра около нулевата честота.
- `stem()`, `plot()`, `subplot()` – визуализация.
- `length()`, `linspace()` – помощни функции.

## IV Задачи за изпълнение

1. Да се сравнят резултатите от DFT (изчислено чрез дефиницията) и FFT за един и същ сигнал. За целта да се генерира синусоидален сигнал с честота 5 Hz, да се изчислят DFT чрез дефиницията и FFT чрез функцията `fft()`. Да се сравнят амплитудните спектри.

Код в MATLAB:

```
clc; clear; close all;

fs = 64;
N = 64;
n = 0:N-1;

f = 5;
x = sin(2*pi*f*n/fs);

% DFT чрез дефиницията
X_dft = zeros(1,N);
for k = 0:N-1
    for nn = 0:N-1
        X_dft(k+1) = X_dft(k+1) + x(nn+1)*exp(-1j*2*pi*k*nn/N);
    end
end

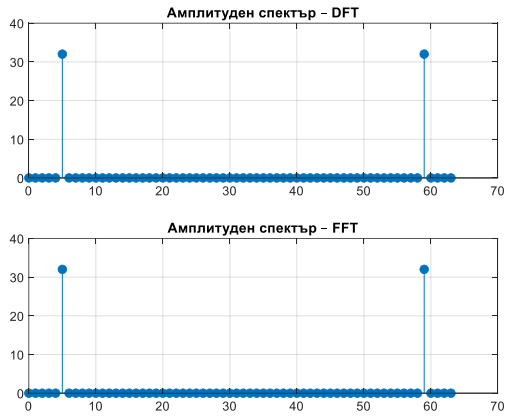
% FFT
X_fft = fft(x);

k = 0:N-1;
```

```
figure;
subplot(2,1,1);
stem(k, abs(X_dft), 'filled');
grid on;
title('Амплитуден спектър – DFT');

subplot(2,1,2);
stem(k, abs(X_fft), 'filled');
grid on;
title('Амплитуден спектър – FFT');
```

```
--- Резултати ---
```




2. За честота на дискретизация $f_s = 200Hz$ и продължителност на сигнала $T = 1s$, да се създаде дискретен времеви вектор и да се генерира сигналът:

$$x(t) = \sin(2\pi \cdot 10t) + 0.5\sin(2\pi \cdot 30t)$$

Да се изчисли FFT на сигнала чрез функцията `fft()` и да се получи комплексният спектър $X[k]$. Да се изчисли амплитудният спектър чрез подходящо нормализиране спрямо броя дискрети $N$. Да се дефинира честотната ос в Hz и да се визуализира амплитудният спектър в обхвата от 0 до 60 Hz.

Код в MATLAB:

```
fs = 200;
T = 1;
t = 0:1/fs:T-1/fs;

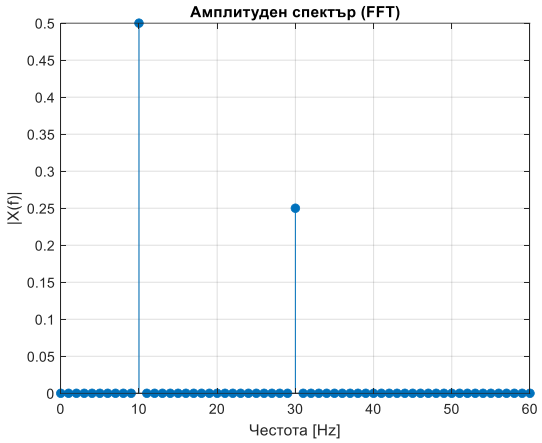
x = sin(2*pi*10*t) + 0.5*sin(2*pi*30*t);

N = length(x);
X = fft(x);

f_axis = (0:N-1)*(fs/N);

figure;
stem(f_axis, abs(X)/N, 'filled');
grid on;
xlim([0 60]);
title('Амплитуден спектър (FFT)');
xlabel('Честота [Hz]');
ylabel('|X(f)|');
```

```
--- Резултати ---
```




3. От получените в Задача 2 сигнал и FFT резултати, изчислете нормализирания амплитуден спектър на сигнала. Да се изведе едностранният спектър, като:
   - се използват само честотните компоненти в диапазона от 0 до $fs/2$;
   - амплитудите на всички честотни компоненти (с изключение на DC и честотата на Найкуист) се умножат по 2.

   Да се дефинира едностранен амплитуден спектър с честотна ос в Hz. Сравнете получения едностранен спектър с двустранния от Задача 2.

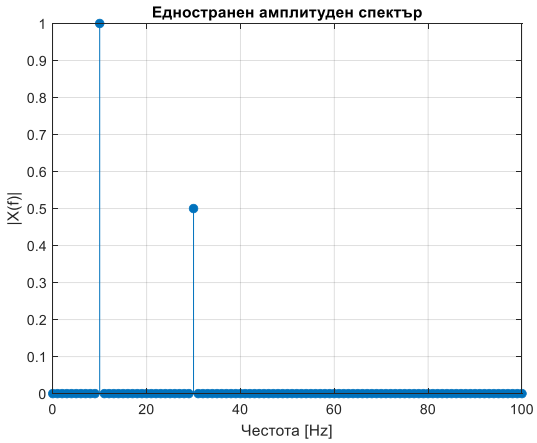
Код в MATLAB:

```
X_mag = abs(X)/N;
X_one = X_mag(1:N/2+1);
X_one(2:end-1) = 2*X_one(2:end-1);

f_one = f_axis(1:N/2+1);

figure;
stem(f_one, X_one, 'filled');
grid on;
title('Едностранен амплитуден спектър');
xlabel('Честота [Hz]');
ylabel('|X(f)|');
```

```
--- Резултати ---
```



4. Използвайки сигналът от Задача 2 и неговата FFT, да се извърши zero-padding, като сигналът се допълни с нули до нова дължина: $N_{pad} = 1024$. Изчислете FFT на допълнения сигнал чрез `fft(x, Npad)`. Да се разшири честотна ос, съответстваща на увеличения брой честотни точки.

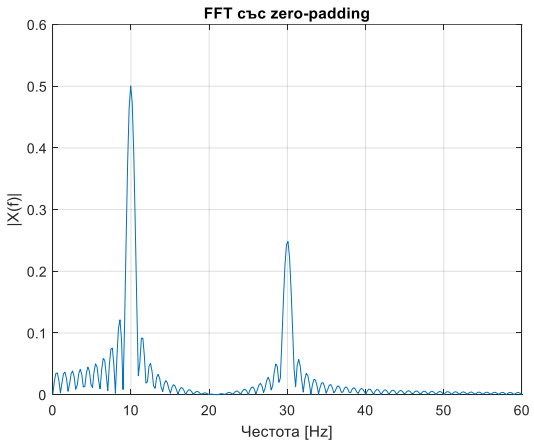
Код в MATLAB:

```
Npad = 1024;
X_pad = fft(x, Npad);

f_pad = (0:Npad-1)*(fs/Npad);

figure;
plot(f_pad, abs(X_pad)/N);
grid on;
xlim([0 60]);
title('FFT със zero-padding');
xlabel('Честота [Hz]');
ylabel('|X(f)|');
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Добавете шум към сигнала и анализирайте спектъра.
2. Променете честотата на една от синусоидите.
3. Сравнете спектрите при различни $f_s$.

## VI Контролни въпроси

1. Каква е разликата между DFT и FFT?
2. Защо FFT е значително по-бързо?
3. Какво представлява zero-padding?
4. Защо спектърът на реален сигнал е симетричен?
5. Какви видове биват получените спектри?

**Упражнение №7**

# Спектрален анализ и прозоречни функции. Спектрално изтичане

**I Цел на упражнението** – Да се изследва влиянието на крайността на наблюдавания сигнал върху спектралния анализ, да се демонстрира спектрално изтичане и да се анализира ролята на прозоречните функции при изчисляване на FFT.

**II Теоретична част**

При реален спектрален анализ сигналът винаги е краен по дължина и е наблюдаван в ограничен времеви интервал. DFT и FFT обаче предполагат, че анализираният сигнал е един период на безкраен периодичен сигнал. Това води до важна последица: ако първата и последната стойност на сигнала не съвпадат, при периодичното продължаване възниква скок (дискретен разрив).

*2.1. Причина за спектралното изтичане (Spectral Leakage)*

Нека дискретен сигнал съдържа синусоида с честота $f$, която не съвпада точно с честотна линия на DFT:

$$f \neq k\frac{f_s}{N}.$$

В този случай енергията на сигнала не се концентрира в една спектрална линия, а се „разлива“ към съседни честоти. Това явление се нарича спектрално изтичане (spectral leakage) и е неизбежно при анализ на крайни сигнали.

*2.2. Интерпретация във времевата област*

Ограничаването на сигнала във времето може да се разглежда като умножение:

$$x_w[n] = x[n] \cdot w[n]$$

където $x[n]$ е оригиналният сигнал, а $w[n]$ е прозоречна функция.

*2.3. Интерпретация във честотната област*

В честотната област умножението във времето съответства на свивка на спектрите:

$$X_w\left(e^{j\omega}\right) = X\left(e^{j\omega}\right) * W\left(e^{j\omega}\right)$$

Спектърът на прозореца $W\left(e^{j\omega}\right)$ определя ширината на главния лоб, както и нивото на страничните лобове. Това води до компромис между честотна резолюция и потискане на страничните лобове.

*2.4. Прозоречни функции*

В обработката на сигнали, прозоречната функция е математическа функция, която е с нулева стойност извън избран интервал. Обикновено тези функции са симетрични около средата на интервала, приближават се до максимум в средата и се стесняват от средата. Математически, когато друга функция (форма на вълната или последователност от данни) се „умножи“ с функция на прозореца, произведението също е с нулева стойност извън интервала: всичко, което остава, е частта, където те се припокриват - „изгледът през прозореца“. На практика, сегментът от данни в рамките на

прозореца първо се изолира и след това само тези данни се умножават по стойностите на функцията на прозореца.

Причините за изследване на сегменти от по-дълга функция включват откриване на преходни събития и осредняване по време на честотните спектри. Продължителността на сегментите се определя във всяко приложение от изисквания като времева и честотна резолюция. Но този метод също така променя честотното съдържание на сигнала чрез ефект, наречен спектрално изтичане. Функциите на прозореца ни позволяват да разпределим спектрално изтичането по различни начини, според нуждите на конкретното приложение.

Функциите на прозореца се използват в спектрални анализ, проектирането на филтри с краен импулсен спектър, обединяването на многомащабни и многомерни набори от данни, както и в проектирането на антени.

- *Спектрален анализ*

Фурие трансформацията на функцията cos(ωt) е нула, освен при честота ±ω. Много други функции и вълнови форми обаче нямат удобни затворени трансформации. Като алтернатива, човек може да се интересува от тяхното спектрално съдържание само през определен период от време.

И в двата случая, Фурие трансформацията (или подобна трансформация) може да се приложи върху един или повече крайни интервали от вълновата форма. Като цяло, трансформацията се прилага към произведението на вълновата форма и функция на прозорец. Всеки прозорец (включително правоъгълен) влияе върху спектралната оценка, изчислена по този метод.

- *Проектиране на филтри*

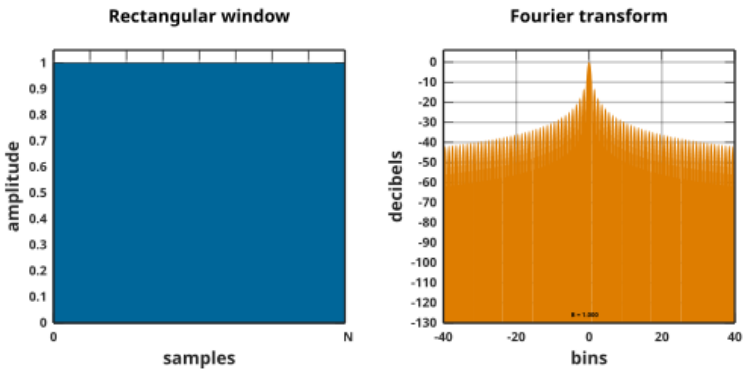
Прозорците понякога се използват при проектирането на цифрови филтри, по-специално за преобразуване на „идеален“ импулсен спектър с безкрайна продължителност, като например sinc функция, в проект на филтър с краен импулсен спектър (FIR). Това се нарича метод на прозореца.

### *2.5. Примери за прозоречни функции*

#### *2.5.1 Правоъгълен прозорец*

Правоъгълният прозорец е най-простият прозорец, еквивалентен на заместване на всички, с изключение на N, последователни стойности на поредица от данни с нули, което принуждава формата на вълната внезапно да се включва и изключва:

$$w[n] = 1,$$
$$n = 0,1,\dots,N-1$$



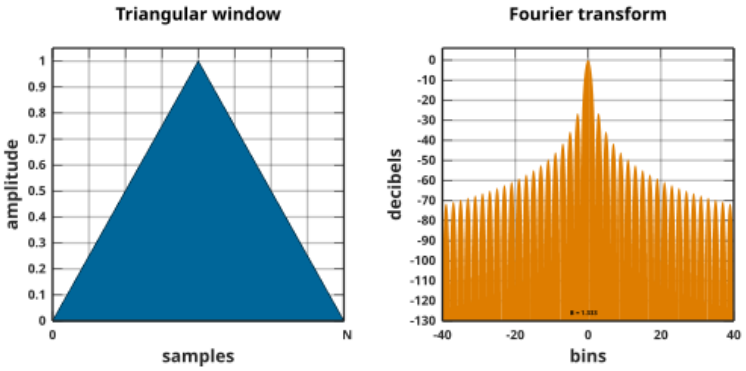

#### *2.5.2 B-spline прозорци*

B-spline прозорците могат да бъдат получени като k-кратни конволюции на правоъгълния прозорец. Те включват самия правоъгълен прозорец (k = 1), Триъгълния прозорец (k = 2) и Прозореца на Парзен (k = 4). B-spline базисна функция от k-ти ред е частично полиномна функция от степен k − 1, която се получава чрез k-кратна самоконволюция на правоъгълната функция.

- Триъгълнен прозорец

Триъгълните прозорци се дефинират като:

$$w[n] = 1 - \left| \frac{n - \frac{N}{2}}{\frac{L}{2}} \right|,$$
$$0 \le n \le N$$



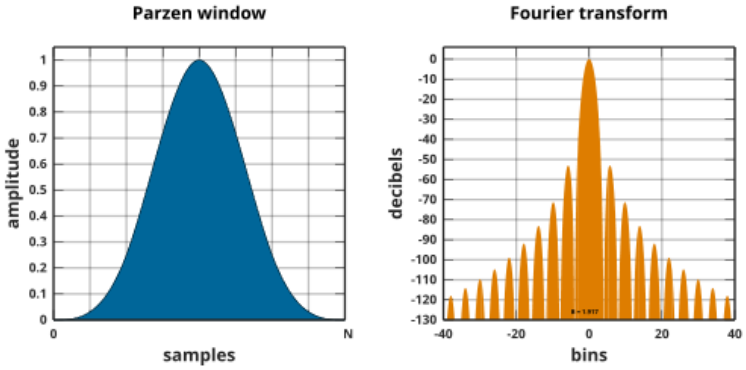

където L може да бъде N, N + 1, или N + 2. Формата L = N може да се разглежда като конволюция на два правоъгълни прозореца с ширина $N / 2$. Фурие трансформацията на резултата е квадратните стойности на трансформацията на правоъгълния прозорец с половин ширина.

- Прозорец на Parzen

Дефинирайки $L \triangleq N + 1$, прозорецът на Parzen е B-spline прозорец от четвърти ред описван с:

$$w_0(n) \triangleq \begin{cases} 1 - 6\left(\frac{n}{\frac{L}{2}}\right)^2 \left(1 - \frac{|n|}{\frac{L}{2}}\right), & 0 \le |n| \le \frac{L}{4} \\ 2\left(1 - \frac{|n|}{\frac{L}{2}}\right)^3, & \frac{L}{4} \le |n| \le \frac{L}{2} \end{cases}$$

$$w[n] = w_0\left(n - \frac{N}{2}\right),$$
$$0 \le n \le N$$

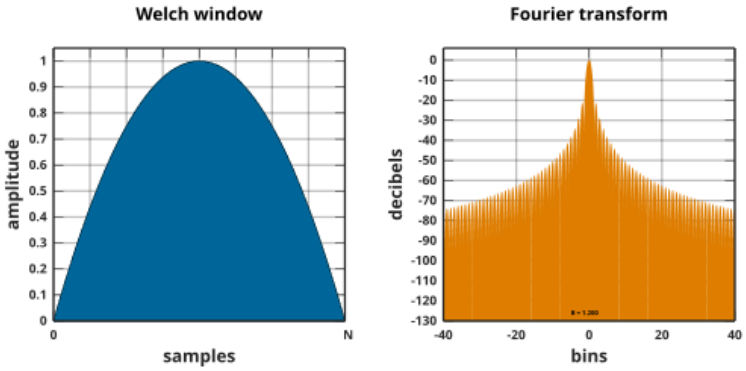



*2.5.3 Други полиномни прозорци*

- Прозорец на Welch

Прозорецът на Welch се състои от едно параболично сечение:

$$w[n] = 1 - \left(\frac{n - \frac{N}{2}}{\frac{N}{2}}\right)^2,$$
$$0 \le n \le N$$



Дефиниращият квадратичен полином достига стойност нула при семплите точно извън обхвата на прозореца.

*2.5.4 Повдигнати косинусоидални прозорци*

Прозорците под формата на косинусна функция, отместена с константа, като например популярните прозорци на Hamming и Hann, понякога се наричат прозорци с повдигнат косинус. Прозорецът на Hann е по-специално подобен на повдигнатото косинусно разпределение, което плавно преминава към нула в краищата си.

Повдигнатите косинусоидални прозорци имат следният вид:

$$w[n] = a_0 - (1 - a_0) \cdot \cos\left(\frac{2\pi n}{N}\right), \qquad 0 \le n \le N$$

- *Прозорец на Hann*

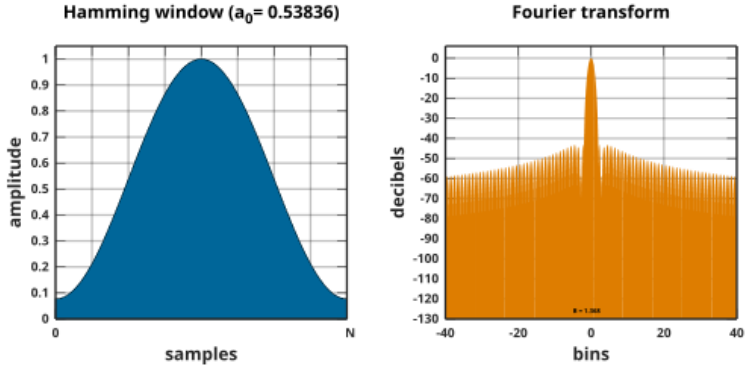
Задавайки $a_0 = 0.5$ и $a_1 = -0.5$ се създава прозорец на Hann:

$$w[n] = \sum_{l=0}^{L-1} A_l \cos\left(\frac{2\pi l n}{N}\right) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N}\right)$$

Тази функция е член както на семействата косинусова сума, така и на степен на синус. За разлика от прозореца на Hamming, крайните точки на прозореца на Hann просто докосват нулата. Получените странични лобове намаляват с около 18 dB на октава.

- *Прозорец на Hamming*

Задаването на $a_0 \approx 0.54$, или по-точно 25/46, създава прозореца на Hamming. Този избор поставя пресичане на нулата при честота 5π/(N − 1), което анулира първия страничен лоб на прозореца на Hann, придавайки му височина от около една пета от тази на прозореца на Hann.




*2.5.5 Косинусоидални суми за прозорци*

Това семейство, което обобщава повдигнатите косинусови прозорци, е известно още като обобщени косинусови прозорци.

$$w[n] = \sum_{k=0}^{K-1} (-1)^k a_k \cos\left(\frac{2\pi k n}{N}\right), \qquad 0 \le n \le N$$

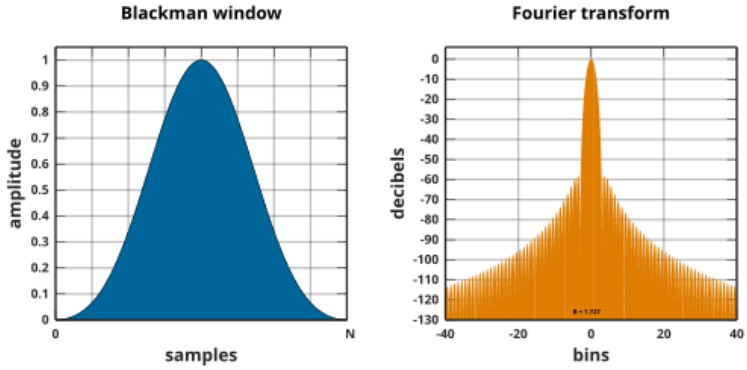
В повечето случаи, включително примерите по-долу, всички коефициенти $a_k \ge 0$. Тези прозорци имат само $2K + 1$ ненулеви N-точкови DFT коефициенти.

- *Прозорец на Blackman*

Прозорците на Blackman се дефинират като

$$w[n] = a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right),$$
$$a_0 = \frac{1-\alpha}{2}; \quad a_1 = \frac{1}{2}; \quad a_2 = \frac{\alpha}{2}$$

Стойностите $\alpha = 0{,}16, a0 = 0{,}42,\ a1 = 0{,}5,\ a2 = 0{,}08,$ са съкратени и поставят нули на третия и четвъртия странични лобове, но водят до прекъсване по краищата и спад от 6 dB/oct.

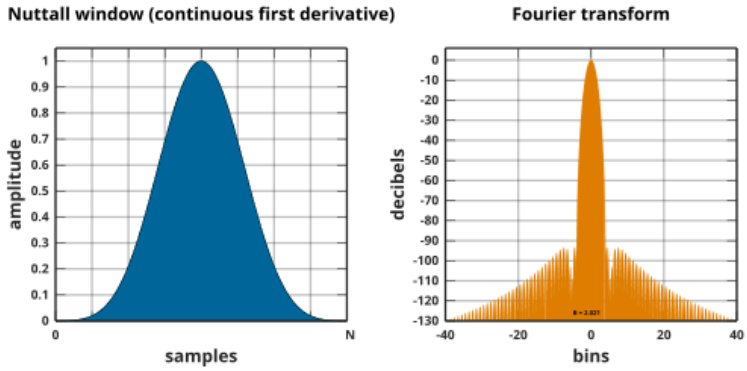



- *Прозорец на Nuttall, непрекъсната първа производна*

Непрекъснатата форма на прозореца на Нътъл $w_0(x)$ и първата му производна са непрекъснати навсякъде, подобно на функцията на Хан. Тоест, функцията клони към 0 при $x = \pm N/2$.

$$w[n] = a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right) - a_3 \cos\left(\frac{6\pi n}{N}\right)$$

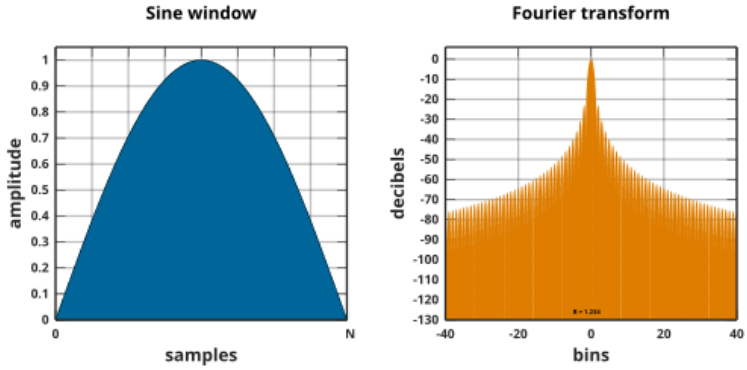
$$a_0 = 0.36;\ \ a_1 = 0.49;\ \ a_2 = 0.14;\ \ a_3 = 0.01$$




*2.5.6 Синусоидални прозоречни функции*

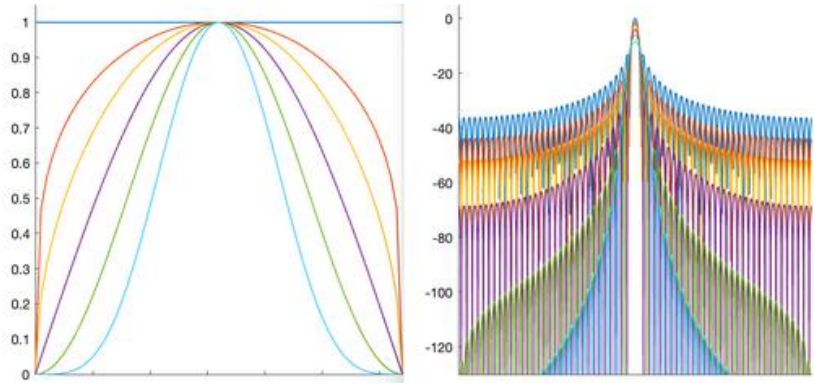
$$w[n] = \sin\left(\frac{\pi n}{N}\right) = \cos\left(\frac{\pi n}{N} - \frac{\pi}{2}\right), \qquad 0 \le n \le N$$

Съответната функция $w_0(n)$ е косинус без фазово отместване $\frac{\pi}{2}$. Така че синусовият прозорец понякога се нарича още косинусов прозорец. Автокорелацията на синусов прозорец създава функция, известна като прозорец на Боман.




- *Прозорци от вида - степен на синус/косинус*

Прозоречни функции на степен на синус (вляво) и техните спектри в dB (вдясно), за степени 0, 0.25, 0.5, 1, 2, 4 са представени на следващата фигура.



Тези прозоречни функции имат вида:

$$w[n] = \sin^{\alpha}\left(\frac{\pi n}{N}\right) = \cos^{\alpha}\left(\frac{\pi n}{N} - \frac{\pi}{2}\right), \qquad 0 \le n \le N$$

Правоъгълният прозорец (α = 0), синусовият прозорец (α = 1) и прозорецът на Хан (α = 2) са членове на това семейство. За четно-целочислови стойности на α тези функции могат да бъдат изразени и в косинусова сума.

## III Подготовка за упражнението

*3.1 Функции за спектрален анализ:*

- `fft()` – изчислява бързото преобразуване на Фурие (FFT) на дискретен сигнал.
- `abs()` – изчислява амплитудата (модула) на комплексния спектър.
- `angle()` – изчислява фазовия спектър (по избор).
- `fftshift()` – пренарежда спектъра около нулевата честота (използва се при двустранни спектри).

*3.2 Прозоречни функции*

- `rectwin()` – генерира правоъгълен прозорец (еквивалентен на липса на прозорец).
- `hann()` – генерира прозорец Hann за намаляване на спектралното изтичане.
- `hamming()` – генерира прозорец Hamming с по-добро потискане на страничните лобове.
- `blackman()` – генерира прозорец Blackman с много силно потискане на изтичането.

*3.3. Функции за работа със сигнали*

- `sin()` – генерира синусоидален сигнал.
- `.*` – извършва поелементно умножение (прилагане на прозорец върху сигнал).
- `length()` – определя броя дискрети на сигнала.
- `zeros()` – създава вектор от нули (използва се при zero-padding).

## IV Задачи за изпълнение

1. Да се генерира синусоидален сигнал с честота $f = 12.5\,Hz$ и честота на дискретизация $f_s = 128\,Hz$ с брой отсчети $N = 128$. Да се изчисли FFT на сигнала без използване на прозоречна функция и да се визуализира едностранният амплитуден спектър.

Код в MATLAB:

```
% Спектрален анализ без прозорец (Rectangular)
clc; clear; close all;

fs = 128;
N  = 128;
n  = 0:N-1;
t  = n/fs;

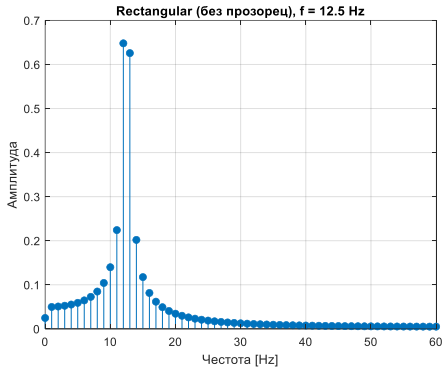
f_sig = 12.5;                  % честота между DFT биновете
x = sin(2*pi*f_sig*t);

X = fft(x);
Xmag = abs(X)/N;

X_one = Xmag(1:N/2+1);
X_one(2:end-1) = 2*X_one(2:end-1);
f = (0:N/2)*(fs/N);

figure;
stem(f, X_one, 'filled'); grid on;
xlim([0 60]);
title('Rectangular (без прозорец), f = 12.5 Hz');
xlabel('Честота [Hz]');
ylabel('Амплитуда');
```

```
--- Резултати ---
```




2. Да се повтори Задание 1 за честота $f = 12\ Hz$ (точно кратна на $\Delta f = fs/N$). Да се сравнят спектрите за 12 Hz и 12.5 Hz. Да се анализира защо при едната честота спектърът е концентриран, а при другата – разлят.

Код в MATLAB:

```
% Сравнение: on-bin и off-bin честота
clc; clear; close all;

fs = 128;
N  = 128;
n  = 0:N-1;
t  = n/fs;

f_on  = 12;      % точно на DFT бин
f_off = 12.5;    % между DFT бинове

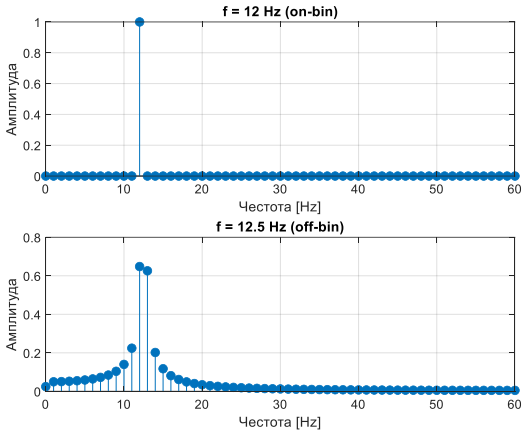
x_on  = sin(2*pi*f_on*t);
x_off = sin(2*pi*f_off*t);
```

```
signals = {x_on, x_off};
titles  = {'f = 12 Hz (on-bin)', 'f = 12.5 Hz (off-bin)'};

figure;
for k = 1:2
    X = fft(signals{k});
    Xmag = abs(X)/N;
    X_one = Xmag(1:N/2+1);
    X_one(2:end-1) = 2*X_one(2:end-1);
    f = (0:N/2)*(fs/N);

    subplot(2,1,k);
    stem(f, X_one, 'filled'); grid on;
    xlim([0 60]);
    title(titles{k});
    xlabel('Честота [Hz]');
    ylabel('Амплитуда');
end
```

```
--- Резултати ---
```




3. Да се приложи прозоречната функция на Hann върху сигнала от Задание 1. Да се изчисли FFT на прозоречния сигнал и да се визуализира амплитудният спектър. Сравнете с резултата без прозорец.

Код в MATLAB:

```
% Влияние на Hann прозорец
fs = 128;
N  = 128;
n  = 0:N-1;
t  = n/fs;

f_sig = 12.5;
x = sin(2*pi*f_sig*t);

w_rect = ones(N,1);
w_hann = hann(N);

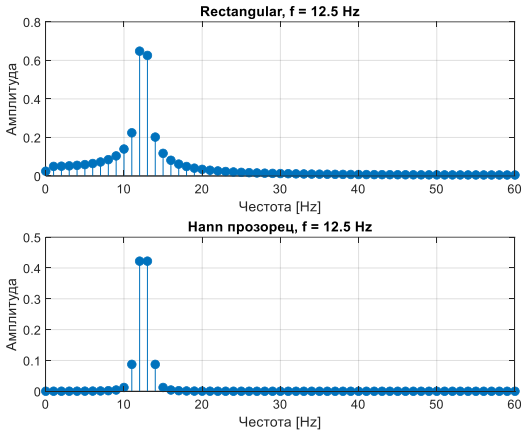
signals = {x .* w_rect', x .* w_hann'};
titles  = {'Rectangular', 'Hann прозорец'};

figure;
for k = 1:2
    X = fft(signals{k});
```

```
    Xmag = abs(X)/N;
    X_one = Xmag(1:N/2+1);
    X_one(2:end-1) = 2*X_one(2:end-1);
    f = (0:N/2)*(fs/N);

    subplot(2,1,k);
    stem(f, X_one, 'filled'); grid on;
    xlim([0 60]);
    title([titles{k}, ', f = 12.5 Hz']);
    xlabel('Честота [Hz]');
    ylabel('Амплитуда');
end
```

```
--- Резултати ---
```




5. Да се анализира сигналът от Задание 1 с различни прозорци:
   - Rectangular
   - Hann
   - Hamming
   - Blackman

За всеки прозорец да се изчисли FFT и да се изобрази амплитудният спектър. Представете резултатите в един общ графичен прозорец.

Код в MATLAB:

```
% Сравнение на прозорци
clc; clear; close all;

fs = 128;
N  = 128;
n  = 0:N-1;
t  = n/fs;

f_sig = 12.5;
x = sin(2*pi*f_sig*t);

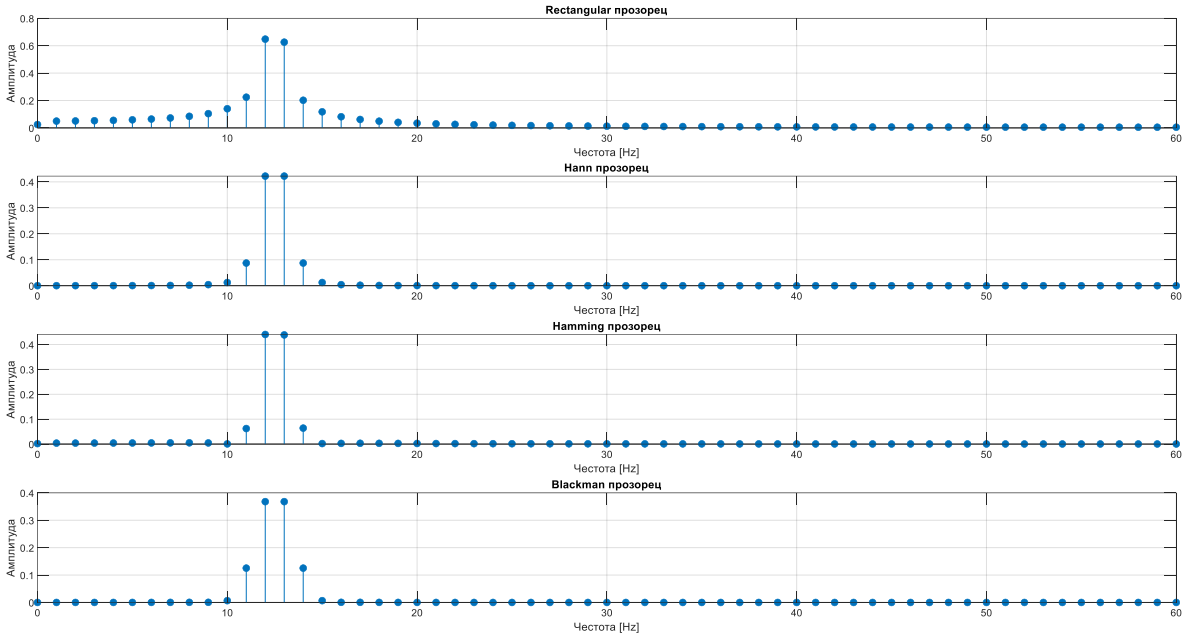
windows = {ones(N,1), hann(N), hamming(N), blackman(N)};
names   = {'Rectangular', 'Hann', 'Hamming', 'Blackman'};

figure;
for k = 1:4
    xw = x .* windows{k}';
    X = fft(xw);
    Xmag = abs(X)/N;
```

```
    X_one = Xmag(1:N/2+1);
    X_one(2:end-1) = 2*X_one(2:end-1);
    f = (0:N/2)*(fs/N);

    subplot(4,1,k);
    stem(f, X_one, 'filled'); grid on;
    xlim([0 60]);
    title([names{k}, ' прозорец']);
    xlabel('Честота [Hz]');
    ylabel('Амплитуда');
end
```

```
--- Резултати ---
```




4. Да се анализира сигналът от Задание 1 с различни прозорци:

Код в MATLAB:

```
% Амплитуда и корекция при различни прозорци
clc; clear; close all;

fs = 128;
N  = 128;
n  = 0:N-1;
t  = n/fs;

f_sig = 12;     % on-bin честота
x = sin(2*pi*f_sig*t);

windows = {ones(N,1), hann(N), hamming(N), blackman(N)};
names   = {'Rectangular', 'Hann', 'Hamming', 'Blackman'};

fprintf('\nПрозорец | Coherent Gain | Амплитуда (raw) | Амплитуда
(corr)\n');
fprintf('-----------------------------------------------------------\n');

figure; hold on; grid on;

for k = 1:4
```

```
    w = windows{k};
    cg = mean(w);

    xw = x .* w';
    X = fft(xw);
    Xmag = abs(X)/N;
    X_one = Xmag(1:N/2+1);
    X_one(2:end-1) = 2*X_one(2:end-1);
    f = (0:N/2)*(fs/N);

    [peak, idx] = max(X_one);
    peak_corr = peak / cg;

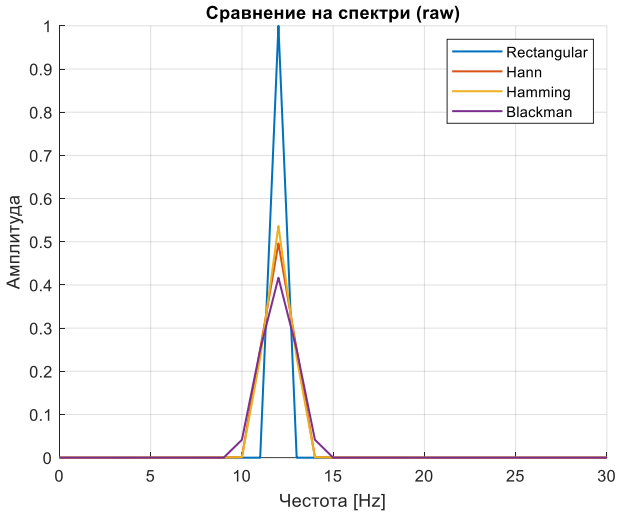
    fprintf('%-12s | %10.4f | %14.4f | %14.4f\n', ...
        names{k}, cg, peak, peak_corr);

    plot(f, X_one, 'LineWidth', 1.2);
end

xlim([0 30]);
title('Сравнение на спектри (raw)');
xlabel('Честота [Hz]');
ylabel('Амплитуда');
legend(names, 'Location','northeast');
```

```
--- Резултати ---

Прозорец | Coherent Gain | Амплитуда (raw) | Амплитуда (corr)
------------------------------------------------------------
Rectangular  |     1.0000 |         1.0000 |         1.0000
Hann         |     0.4961 |         0.4961 |         1.0000
Hamming      |     0.5364 |         0.5364 |         1.0000
Blackman     |     0.4167 |         0.4167 |         1.0000
```



## V Допълнителни задачи

1. Повторете Задание 3, като промените броя отсчети $N = 64, 128, 256$. Използвайте прозореца на Hann.
2. Генерирайте сигнал, състоящ се от две синусоиди. Амплитудата на втората компонента да бъде с 5–10 пъти по-малка спрямо първата, а честотите да са близки. Извършете спектрален анализ без прозорец и с прозорец на Blackman.
3. Приложете zero-padding (например до $N=1024$) към сигнал с прозорец на Hann. Сравнете спектъра със и без zero-padding.

## VI Контролни въпроси

1. Какво представлява спектралното изтичане и защо възниква?
2. Каква е разликата между правоъгълен прозорец и Hann прозорец?
3. Обяснете защо zero-padding подобрява визуализацията, но не премахва спектралното изтичане?

**Упражнение №8**

# Децимация и интерполация на цифрови сигнали. Промяна на честотата на дискретизация

**I Цел на упражнението** – Да се усвоят принципите на промяна на честотата на дискретизация на цифрови сигнали чрез процесите децимация и интерполация. Изследване на влиянието на процесите върху времевото и честотното представяне на сигналите, изясни ролята на изглаждащите филтри, както и връзката между промяната на честотата на дискретизация и спектралните изкривявания.

**II Теоретична част**

Честотата на дискретизация $f_s$ определя броя дискрети на сигнала за единица време.

В много приложения е необходимо намаляване на $f_s$ с цел намаляване на обема данни, докато в други случай е необходимо увеличаване на $f_s$ за съвместимост с други системи. Тези операции се реализират чрез дискретна промяна на времевата ос, а не чрез промяна на самия сигнал.

*2.1. Децимация (Downsampling)*

*Цифровата децимация* е процес на дискретизация на дискретизирани сигнали. Тя се основава на обработване на всяка M-та дискретна стойност на входната поредица. Така се премахва излишната информация, а входният сигнал се описва по-компактно.

Децимацията представлява намаляване на честотата на дискретизация с цял коефициент $M$:

$$y[n] = x[nM].$$

При децимация новата честота на дискретизация ще бъде:

$$f'_s = \frac{f_s}{M},$$

а спектърът на сигнала се „сгъва“ около новата честота на Найкуист.

Преди децимация е необходимо прилагане на нискочестотен филтър, който да потиска честоти над $\frac{f_s}{M}$ и да предотвратява спектрално припокриване.

*2.2. Интерполация (Upsampling)*

*Цифровата интерполация* е процес на възстановяване на дискретен сигнал от друг дискретен сигнал. Интерполацията представлява увеличаване на честотата на дискретизация с цял коефициент $L$. Процесът се състои от две стъпки – първо се вмъкват $L-1$ нулеви дискрета между съществуващите:

$$x_u[n] = \begin{cases} x\left[\frac{n}{L}\right], & n = kL \\ 0, & n \neq kL \end{cases}.$$

След това се прилага изглаждащ (anti-imaging) филтър. Новата честота на дискретизация е:

$$f'_s = Lf_s$$

След интерполация спектърът съдържа повтарящи се копия (образи) около кратни на оригиналната честота на Найкуист. Изглаждащият филтър премахва тези образи и възстановява гладък сигнал.

*2.3. Комбинирана промяна на честотата*

Общата промяна на честотата на дискретизация с коефициент $\frac{L}{M}$ се реализира чрез интерполация с $L$ и последваща децимация с $M$. Този подход се използва в реални системи за честотно преобразуване.

## III Подготовка за упражнението

Използвани функции в MATLAB:

- `downsample()` – децимира сигнал чрез вземане на всеки M-ти отсчет
- `upsample()` – интерполира сигнал чрез вмъкване на нули
- `decimate()` – децимация с вграден антиалиасингов филтър
- `interp()` – интерполация с вграден изглаждащ филтър
- `filter()` – прилагане на цифров филтър
- `fft()`, `abs()` – честотен анализ
- `stem()`, `plot()` – визуализация

## IV Задачи за изпълнение

1. Генерирайте сигнал, състоящ се от две синусоидални компоненти:
   - една под получената честота на Найкуист;
   - една над нея.

   Извършете децимация с коефициент $M = 2$ без филтриране. Анализирайте времевото и честотното представяне.

Код в MATLAB:

```
% Децимация без антиалиасингов филтър

%% Параметри
fs = 200;             % честота на дискретизация [Hz]
T  = 1;               % продължителност [s]
t  = 0:1/fs:T-1/fs;

M = 2;                % коефициент на децимация
fs2 = fs/M;           % нова честота на дискретизация

%% Генериране на входния сигнал
% Първата компонента е под новата честота на Найкуист:
% fs2/2 = 50 Hz
f1 = 20;              % [Hz]

% Втората компонента е над новата честота на Найкуист
% и след децимация ще предизвика алиасинг
f2 = 70;              % [Hz]

x = sin(2*pi*f1*t) + 0.7*sin(2*pi*f2*t);

%% Децимация без антиалиасингов филтър
% Взема се всеки M-ти отсчет
y = x(1:M:end);

% Нова времева ос
```

```
t2 = (0:length(y)-1)/fs2;

%% Визуализация във времевата област
figure('Name','Задание 1 - Времева област');

subplot(2,1,1);
plot(t, x, 'LineWidth', 1.2);
grid on;
title('Оригинален сигнал x(t)');
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(2,1,2);
stem(t2, y, 'filled');
grid on;
title(['Децимиран сигнал y[n] без филтър, f_s'' = ',num2str(fs2), ' Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

%% ============= Едностранен спектър на оригиналния сигнал ================
Nx = length(x);

% FFT
X = fft(x);

% Нормализиран двустранен амплитуден спектър
Xmag = abs(X)/Nx;

% Едностранен спектър
if mod(Nx,2) == 0
    X1 = Xmag(1:Nx/2+1);
    X1(2:end-1) = 2*X1(2:end-1);

    fX = (0:Nx/2)*(fs/Nx);
else
    X1 = Xmag(1:(Nx+1)/2);
    X1(2:end) = 2*X1(2:end);

    fX = (0:(Nx-1)/2)*(fs/Nx);
end

%% ============= Едностранен спектър на децимирания сигнал ================
Ny = length(y);

% FFT
Y = fft(y);

% Нормализиран двустранен амплитуден спектър
Ymag = abs(Y)/Ny;

% Едностранен спектър
if mod(Ny,2) == 0
    Y1 = Ymag(1:Ny/2+1);
    Y1(2:end-1) = 2*Y1(2:end-1);

    fY = (0:Ny/2)*(fs2/Ny);
else
    Y1 = Ymag(1:(Ny+1)/2);
    Y1(2:end) = 2*Y1(2:end);

    fY = (0:(Ny-1)/2)*(fs2/Ny);
```

```
end

%% Визуализация в честотната област
figure('Name','Задание 1 - Честотна област');

subplot(2,1,1);
plot(fX, X1, 'LineWidth', 1.2);
grid on;
xlim([0 fs/2]);
title('Спектър на оригиналния сигнал');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

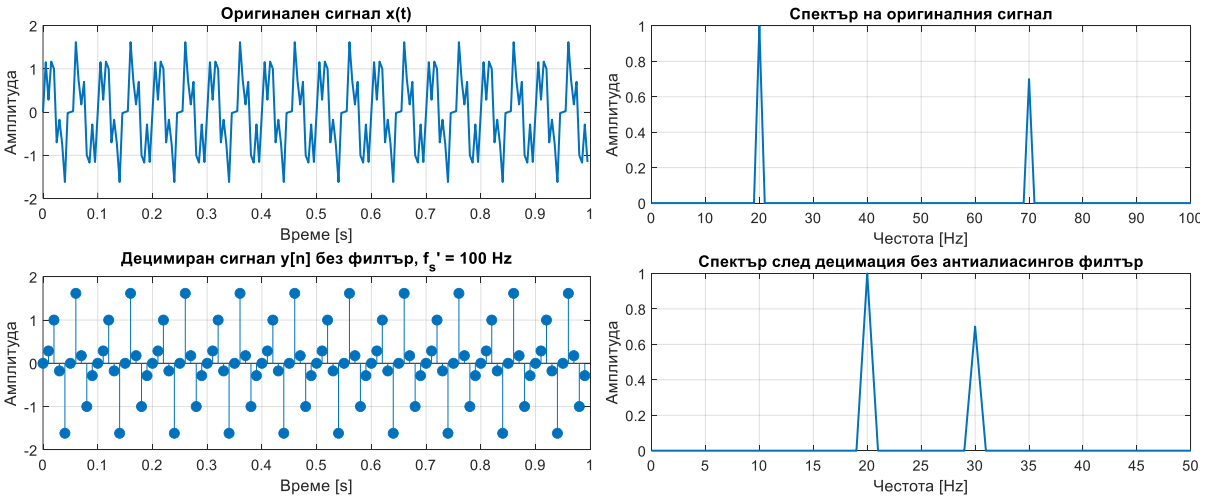
subplot(2,1,2);
plot(fY, Y1, 'LineWidth', 1.2);
grid on;
xlim([0 fs2/2]);
title('Спектър след децимация без антиалиасингов филтър');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

%% Аналитична интерпретация на алиасинга

f_alias = abs(f2 - fs2);

disp(' ');
disp('--- Резултати ---');
disp(['Нова честота на дискретизация: ', num2str(fs2), ' Hz']);
disp(['Нова честота на Найкуист: ', num2str(fs2/2), ' Hz']);
disp(['Оригинална високочестотна компонента: ', num2str(f2), ' Hz']);
disp(['Наблюдавана алиас честота: ', num2str(f_alias), ' Hz']);
```

```
--- Резултати ---
Получена честота на дискретизация: 100 Hz
Получена честота на Найкуист: 50 Hz
Оригинална високочестотна компонента: 70 Hz
Наблюдавана алиас честота: 30 Hz
```



3. Повторете Задание 1, но използвайте `decimate()`. Сравнете резултатите със и без филтър.

Код в MATLAB:

```
% Децимация с антиалиасингов филтър

%% Параметри
fs = 200;             % честота на дискретизация [Hz]
T  = 1;               % продължителност [s]
t  = 0:1/fs:T-1/fs;

M = 2;                % коефициент на децимация
fs2 = fs/M;           % нова честота на дискретизация

%% Генериране на входния сигнал
% Компонента под новата честота на Найкуист
f1 = 20;              % [Hz]

% Компонента над новата честота на Найкуист
f2 = 70;              % [Hz]

x = sin(2*pi*f1*t) + 0.7*sin(2*pi*f2*t);

%% Децимация с антиалиасингов филтър
% Функцията decimate() извършва нискочестотно филтриране
% и последващо намаляване на честотата на дискретизация.
y_dec = decimate(x, M);

% Времева ос на децимирания сигнал
t2 = (0:length(y_dec)-1)/fs2;

%% Визуализация във времевата област
figure('Name','Задание 2 - Времева област');

subplot(2,1,1);
plot(t, x, 'LineWidth', 1.2);
grid on;
title('Оригинален сигнал x(t)');
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(2,1,2);
stem(t2, y_dec, 'filled');
grid on;
title(['Децимиран сигнал чрез decimate(), f_s'' = ', num2str(fs2), ' Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

%% ============= Едностранен спектър на оригиналния сигнал ===============
Nx = length(x);

X = fft(x);
Xmag = abs(X)/Nx;

if mod(Nx,2) == 0
    X1 = Xmag(1:Nx/2+1);
    X1(2:end-1) = 2*X1(2:end-1);

    fX = (0:Nx/2)*(fs/Nx);
else
```

```
    X1 = Xmag(1:(Nx+1)/2);
    X1(2:end) = 2*X1(2:end);

    fX = (0:(Nx-1)/2)*(fs/Nx);
end

%% ============= Едностранен спектър на децимирания сигнал ================
Ny = length(y_dec);

Y = fft(y_dec);
Ymag = abs(Y)/Ny;

if mod(Ny,2) == 0
    Y1 = Ymag(1:Ny/2+1);
    Y1(2:end-1) = 2*Y1(2:end-1);

    fY = (0:Ny/2)*(fs2/Ny);
else
    Y1 = Ymag(1:(Ny+1)/2);
    Y1(2:end) = 2*Y1(2:end);

    fY = (0:(Ny-1)/2)*(fs2/Ny);
end

%% Визуализация в честотната област
figure('Name','Задание 2 - Честотна област');

subplot(2,1,1);
plot(fX, X1, 'LineWidth', 1.2);
grid on;
xlim([0 fs/2]);
title('Спектър на оригиналния сигнал');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

subplot(2,1,2);
plot(fY, Y1, 'LineWidth', 1.2);
grid on;
xlim([0 fs2/2]);
title('Спектър след децимация с антиалиасингов филтър');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

%% Извеждане на основните параметри
disp(' ');
disp('--- Резултати ---');
disp(['Първоначална честота на дискретизация: ', num2str(fs), ' Hz']);

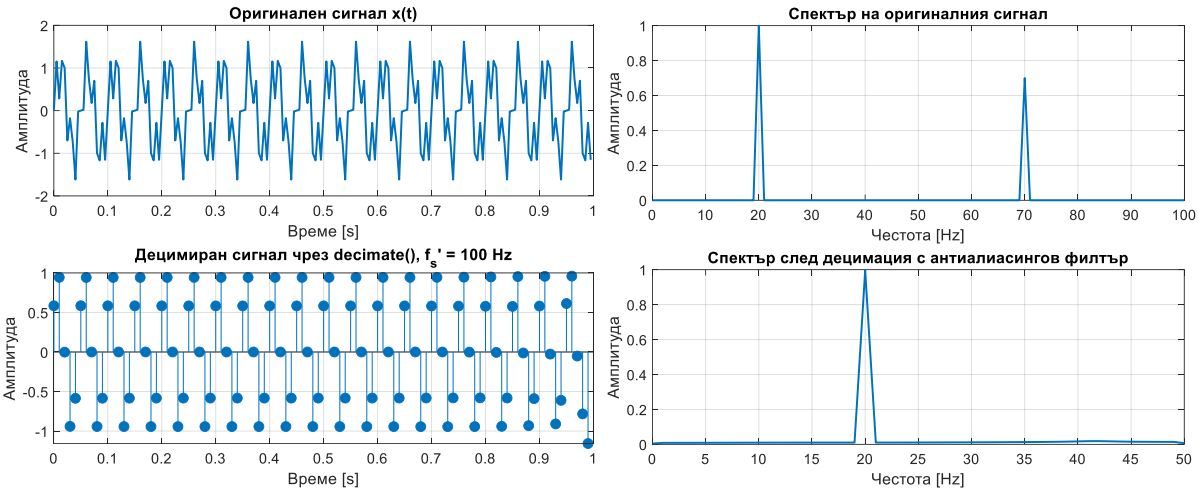
disp(['Нова честота на дискретизация: ', num2str(fs2), ' Hz']);

disp(['Нова честота на Найкуист: ', num2str(fs2/2), ' Hz']);

disp('Компонентата 70 Hz се потиска от антиалиасинговия филтър');
disp('преди намаляване на честотата на дискретизация.');
```

```
--- Резултати ---
Първоначална честота на дискретизация: 200 Hz
Нова честота на дискретизация: 100 Hz
Нова честота на Найкуист: 50 Hz
```

Компонентата 70 Hz се потиска от антиалиасинговия филтър преди намаляване на честотата на дискретизация.




4. Генерирайте нискочестотен сигнал. Извършете upsampling с коефициент $L = 2$ чрез `upsample()`. Анализирайте появата на спектрални образи.

Код в MATLAB:

```
% Интерполация без изглаждащ филтър чрез upsample()

%% Параметри
fs = 100;               % честота на дискретизация [Hz]
T  = 1;                 % продължителност [s]
t  = 0:1/fs:T-1/fs;

L = 2;                  % коефициент на интерполация
fs2 = fs*L;             % нова честота на дискретизация

%% Генериране на нискочестотен сигнал
f1 = 10;                % честота на сигнала [Hz]

x = sin(2*pi*f1*t);

%% Интерполация чрез вмъкване на нули
% upsample() вмъква L-1 нули между всеки два съществуващи отсчета.
x_up = upsample(x, L);

% Времева ос на интерполирания сигнал
t2 = (0:length(x_up)-1)/fs2;

%% Определяне на броя отсчети за визуализация
% Използва се min(), за да не се надхвърли дължината на векторите.
K  = min(200, length(x));
K2 = min(200*L, length(x_up));

%% Визуализация във времевата област
figure('Name','Задание 3 - Времева област');

subplot(2,1,1);
plot(t(1:K), x(1:K), 'LineWidth', 1.2);
grid on;
title('Оригинален сигнал x(t)');
```

```
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(2,1,2);
stem(t2(1:K2), x_up(1:K2), 'filled');
grid on;
title(['Сигнал след upsample(), f_s'' = ', num2str(fs2), ' Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

%% ============= Едностранен спектър на оригиналния сигнал ================
Nx = length(x);

X = fft(x);
Xmag = abs(X)/Nx;

if mod(Nx,2) == 0
    X1 = Xmag(1:Nx/2+1);
    X1(2:end-1) = 2*X1(2:end-1);

    fX = (0:Nx/2)*(fs/Nx);
else
    X1 = Xmag(1:(Nx+1)/2);
    X1(2:end) = 2*X1(2:end);

    fX = (0:(Nx-1)/2)*(fs/Nx);
end

%% ============= Едностранен спектър на децимирания сигнал ================
Nu = length(x_up);

U = fft(x_up);
Umag = abs(U)/Nu;

if mod(Nu,2) == 0
    U1 = Umag(1:Nu/2+1);
    U1(2:end-1) = 2*U1(2:end-1);

    fU = (0:Nu/2)*(fs2/Nu);
else
    U1 = Umag(1:(Nu+1)/2);
    U1(2:end) = 2*U1(2:end);

    fU = (0:(Nu-1)/2)*(fs2/Nu);
end

%% Визуализация в честотната област
figure('Name','Задание 3 - Честотна област');

subplot(2,1,1);
plot(fX, X1, 'LineWidth', 1.2);
grid on;
xlim([0 fs/2]);
title('Спектър на оригиналния сигнал');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

subplot(2,1,2);
plot(fU, U1, 'LineWidth', 1.2);
grid on;
xlim([0 fs2/2]);
```

```
title('Спектър след upsample() - спектрални образи');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

%% Извеждане на основни параметри
disp(' ');
disp('--- Резултати ---');
disp(['Първоначална честота на дискретизация: ', num2str(fs), ' Hz']);

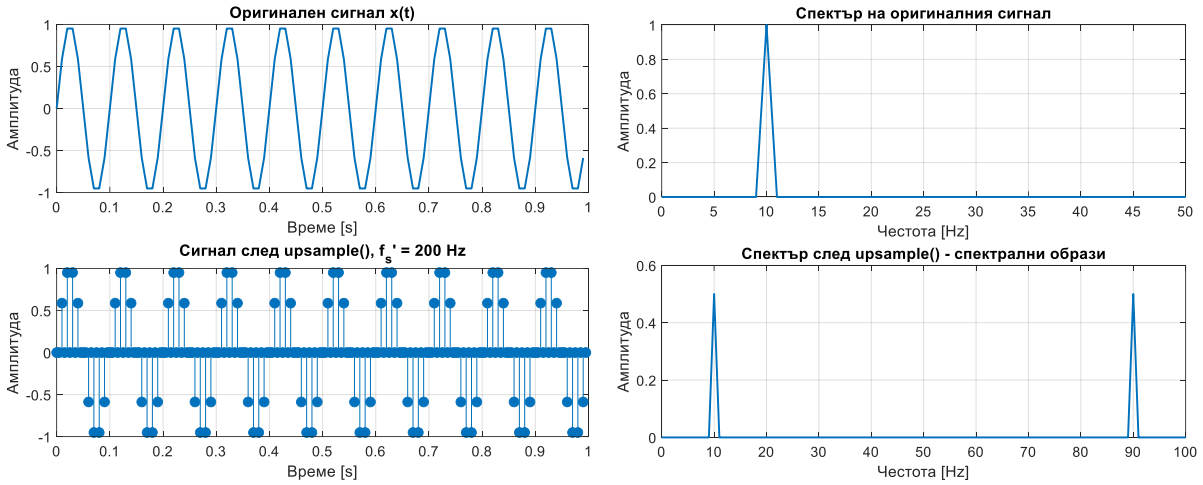
disp(['Коефициент на интерполация L = ', num2str(L)]);

disp(['Нова честота на дискретизация: ', num2str(fs2), ' Hz']);

disp(['Брой отсчети преди интерполация: ', num2str(length(x))]);

disp(['Брой отсчети след upsample(): ', num2str(length(x_up))]);
```

```
--- Резултати ---
Първоначална честота на дискретизация: 100 Hz
Коефициент на интерполация L = 2
Нова честота на дискретизация: 200 Hz
Брой отсчети преди интерполация: 100
Брой отсчети след upsample(): 200
```




6. Повторете Задание 3, използвайки `interp()`. Сравнете резултатите във времевата и честотната област. Обяснете ролята на anti-imaging филтъра.

Код в MATLAB:

```
% Интерполация с изглаждащ филтър. Сравнение между upsample() и interp()

%% Параметри
fs = 100;               % начална честота на дискретизация [Hz]
T  = 1;                 % продължителност на сигнала [s]
t  = 0:1/fs:T-1/fs;

L = 2;                  % коефициент на интерполация
fs2 = fs*L;             % нова честота на дискретизация

%% Генериране на нискочестотен сигнал
f1 = 10;                % честота на сигнала [Hz]
```

```
x = sin(2*pi*f1*t);

%% Интерполация

% Upsampling без изглаждащ филтър:
% между съществуващите отсчети се вмъкват L-1 нули
x_up = upsample(x, L);

% Интерполация с изглаждащ anti-imaging филтър
x_int = interp(x, L);

% Времеви оси
t_up  = (0:length(x_up)-1)/fs2;
t_int = (0:length(x_int)-1)/fs2;

%% Определяне на броя отсчети за визуализация
% Използва се min(), за да не се надвиши дължината
% на съответния сигнал.

K    = min(200, length(x));
K_up = min(200*L, length(x_up));
K_int = min(200*L, length(x_int));

%% Визуализация във времевата област
figure('Name','Задание 4 - Времева област');

subplot(3,1,1);
plot(t(1:K), x(1:K), 'LineWidth', 1.2);
grid on;
title('Оригинален сигнал x(t)');
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(3,1,2);
stem(t_up(1:K_up), x_up(1:K_up), 'filled');
grid on;
title(['След upsample(), f_s'' = ', num2str(fs2), ' Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(3,1,3);
plot(t_int(1:K_int), x_int(1:K_int), 'LineWidth', 1.2);
grid on;
title(['След interp(), f_s'' = ', num2str(fs2), ' Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

%% ============== Едностранен спектър след upsample() ================
Nu = length(x_up);

U = fft(x_up);
Umag = abs(U)/Nu;

if mod(Nu,2) == 0
    U1 = Umag(1:Nu/2+1);
    U1(2:end-1) = 2*U1(2:end-1);

    fU = (0:Nu/2)*(fs2/Nu);
else
    U1 = Umag(1:(Nu+1)/2);
    U1(2:end) = 2*U1(2:end);
```

```
    fU = (0:(Nu-1)/2)*(fs2/Nu);
end

%% ============= Едностранен спектър след interp()================
Ni = length(x_int);

I = fft(x_int);
Imag = abs(I)/Ni;

if mod(Ni,2) == 0
    I1 = Imag(1:Ni/2+1);
    I1(2:end-1) = 2*I1(2:end-1);

    fI = (0:Ni/2)*(fs2/Ni);
else
    I1 = Imag(1:(Ni+1)/2);
    I1(2:end) = 2*I1(2:end);

    fI = (0:(Ni-1)/2)*(fs2/Ni);
end

%% Визуализация в честотната област
figure('Name','Задание 4 - Честотна област');

subplot(2,1,1);
plot(fU, U1, 'LineWidth', 1.2);
grid on;
xlim([0 fs2/2]);
title('Спектър след upsample() - наличие на spectral imaging');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

subplot(2,1,2);
plot(fI, I1, 'LineWidth', 1.2);
grid on;
xlim([0 fs2/2]);
title('Спектър след interp() - spectral imaging е потиснат');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

%% Извеждане на основните параметри
disp(' ');
disp('--- Резултати ---');

disp(['Начална честота на дискретизация: ', num2str(fs), ' Hz']);

disp(['Коефициент на интерполация: L = ', num2str(L)]);

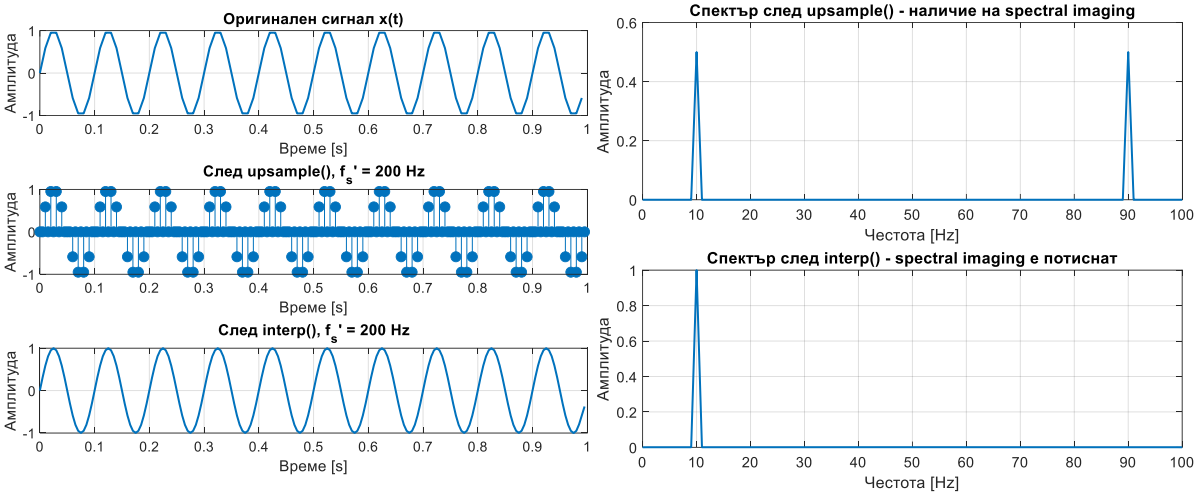
disp(['Нова честота на дискретизация: ', num2str(fs2), ' Hz']);

disp(['Брой отсчети на оригиналния сигнал: ', num2str(length(x))]);

disp(['Брой отсчети след upsample(): ', num2str(length(x_up))]);

disp(['Брой отсчети след interp(): ', num2str(length(x_int))]);
```

```
--- Резултати ---
Начална честота на дискретизация: 100 Hz
Коефициент на интерполация: L = 2
Нова честота на дискретизация: 200 Hz
Брой отсчети на оригиналния сигнал: 100
Брой отсчети след upsample(): 200
Брой отсчети след interp(): 200
```




7. Реализирайте промяна на честотата с коефициент $\frac{3}{2}$. Използвайте интерполация с $L = 3$ и децимация с $M = 2$. Анализирайте спектъра след всяка стъпка.

Код в MATLAB:

```
% Промяна на честотата на дискретизация с коефициент L/M = 3/2
% Интерполация с L = 3 и последваща децимация с M = 2

%% Параметри
fs = 120;                % начална честота на дискретизация [Hz]
T  = 1;                  % продължителност на сигнала [s]
t  = 0:1/fs:T-1/fs;

L = 3;                   % коефициент на интерполация
M = 2;                   % коефициент на децимация

fs_L   = fs*L;           % честота след интерполация
fs_out = fs*L/M;         % крайна честота на дискретизация

%% Генериране на входния сигнал
% Две честотни компоненти
f1 = 10;                 % [Hz]
f2 = 25;                 % [Hz]

x = sin(2*pi*f1*t) + 0.6*sin(2*pi*f2*t);

%% 1. Интерполация с коефициент L
% interp() вмъква допълнителни отсчети и прилага изглаждащ филтър.
xL = interp(x, L);

% Времева ос след интерполация
tL = (0:length(xL)-1)/fs_L;
```

```
%% 2. Децимация с коефициент M
% decimate() прилага антиалиасингов филтър и след това
% намалява броя на отсчетите.
y = decimate(xL, M);

% Времева ос на крайния сигнал
tY = (0:length(y)-1)/fs_out;

%% Визуализация във времевата област
figure('Name','Задание 5 - Времева област');

subplot(3,1,1);
plot(t, x, 'LineWidth', 1.2);
grid on;
title(['Оригинален сигнал, f_s = ', num2str(fs), ' Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(3,1,2);
plot(tL, xL, 'LineWidth', 1.2);
grid on;
title(['След интерполация с L = ', num2str(L), ', f_s = ', num2str(fs_L), '
Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

subplot(3,1,3);
stem(tY, y, 'filled');
grid on;
title(['След децимация с M = ', num2str(M), ', f_s = ', num2str(fs_out), '
Hz']);
xlabel('Време [s]');
ylabel('Амплитуда');

%% ================ Едностранен спектър на входния сигнал =================
Nx = length(x);
X = fft(x);
Xmag = abs(X)/Nx;

if mod(Nx,2) == 0
    X1 = Xmag(1:Nx/2+1);
    X1(2:end-1) = 2*X1(2:end-1);
    fX = (0:Nx/2)*(fs/Nx);
else
    X1 = Xmag(1:(Nx+1)/2);
    X1(2:end) = 2*X1(2:end);
    fX = (0:(Nx-1)/2)*(fs/Nx);
end

%% ================ Едностранен спектър след интерполация ================
NL = length(xL);

XL = fft(xL);
XLmag = abs(XL)/NL;

if mod(NL,2) == 0
    L1 = XLmag(1:NL/2+1);
    L1(2:end-1) = 2*L1(2:end-1);
    fL = (0:NL/2)*(fs_L/NL);
else
    L1 = XLmag(1:(NL+1)/2);
```

```
    L1(2:end) = 2*L1(2:end);
    fL = (0:(NL-1)/2)*(fs_L/NL);
end

%% ================ Едностранен спектър на крайния сигнал =================
Ny = length(y);

Y = fft(y);
Ymag = abs(Y)/Ny;

if mod(Ny,2) == 0
    Y1 = Ymag(1:Ny/2+1);
    Y1(2:end-1) = 2*Y1(2:end-1);
    fY = (0:Ny/2)*(fs_out/Ny);
else
    Y1 = Ymag(1:(Ny+1)/2);
    Y1(2:end) = 2*Y1(2:end);
    fY = (0:(Ny-1)/2)*(fs_out/Ny);
end

%% Визуализация в честотната област
figure('Name','Задание 5 - Честотна област');

subplot(3,1,1);
plot(fX, X1, 'LineWidth', 1.2);
grid on;
xlim([0 fs/2]);
title('Спектър на входния сигнал');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

subplot(3,1,2);
plot(fL, L1, 'LineWidth', 1.2);
grid on;
xlim([0 fs_L/2]);
title('Спектър след интерполация');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

subplot(3,1,3);
plot(fY, Y1, 'LineWidth', 1.2);
grid on;
xlim([0 fs_out/2]);
title('Спектър на крайния сигнал');
xlabel('Честота [Hz]');
ylabel('Амплитуда');

%% Извеждане на резултатите
disp(' ');
disp('--- Резултати ---');

disp(['Начална честота на дискретизация: ', num2str(fs), ' Hz']);

disp(['Честота след интерполация: ', num2str(fs_L), ' Hz']);

disp(['Крайна честота на дискретизация: ', num2str(fs_out), ' Hz']);

disp(['Общ коефициент на преобразуване L/M = ', num2str(L/M)]);

disp(['Брой отсчети на входния сигнал: ', num2str(length(x))]);
```

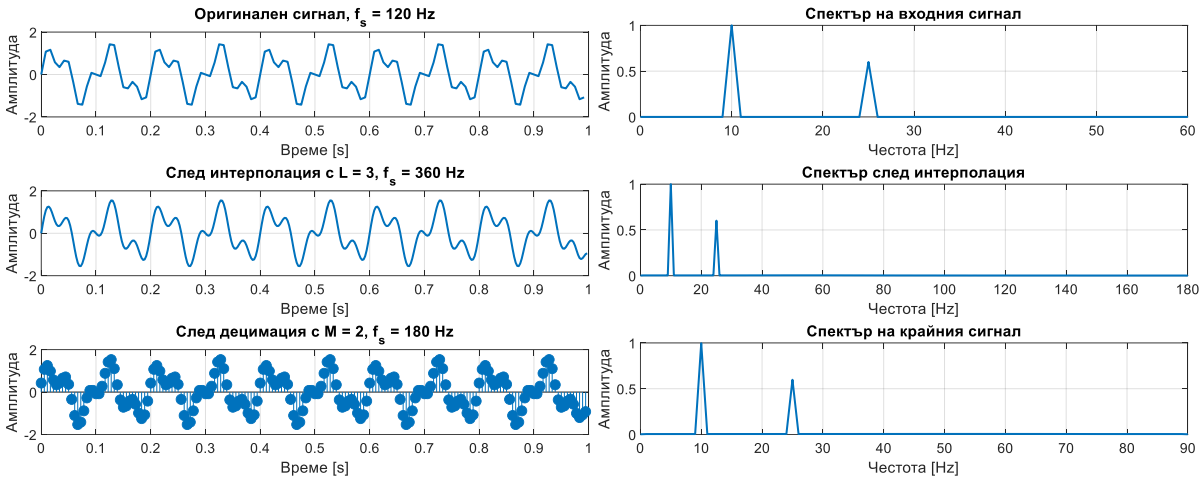
```
disp(['Брой отсчети след интерполация: ', num2str(length(xL))]);

disp(['Брой отсчети на крайния сигнал: ', num2str(length(y))]);

fprintf('\nfs_out / fs = %.3f\n', fs_out/fs);
```

```
--- Резултати ---
Начална честота на дискретизация: 120 Hz
Честота след интерполация: 360 Hz
Крайна честота на дискретизация: 180 Hz
Общ коефициент на преобразуване L/M = 1.5
Брой отсчети на входния сигнал: 120
Брой отсчети след интерполация: 360
Брой отсчети на крайния сигнал: 180

fs_out / fs = 1.500
```




## V Допълнителни задачи

1. Да се децимира един и същ сигнал с коефициенти M=2, M=3 и M=4. Сравнете получените сигнали и техните амплитудни спектри. Анализирайте влиянието на коефициента M.
2. Да се извърши интерполира един и същ сигнал с коефициенти L=2, L=3 и L=4. Определете новата честота на дискретизация и сравнете получените амплитудни спектри.
3. Да се промени честотата на дискретизация с отношение $\frac{L}{M} = \frac{3}{2}$, като използвате функцията resample(). Определете новата честота на дискретизация и сравнете входния и изходния сигнал във времевата и честотната област.

## VI Контролни въпроси

1. Какво представлява процесът децимация и как променя честотата на дискретизация?
2. Какви спектрални ефекти възникват при интерполация?
3. Каква е разликата между upsample() и interp()?

**Упражнение №9**

# Филтри с крайна импулсна характеристика - синтез и анализ

**I Цел на упражнението** – Синтез и анализ на FIR (Finite Impulse Response) филтри, основни предимства, включително линейна ФЧХ и безусловна устойчивост. Проектиране на FIR филтри чрез прозоречния метод и чрез оптимизационни методи (`fir1`, `firpm`), анализират се честотните характеристики, импулсната реакция и груповото закъснение, както и влиянието на реда и избора на прозорец върху характеристиките на филтъра.

**II Теоретична част**

Филтрите с крайна импулсна характеристика (Finite Impulse Response, FIR) са цифрови филтри, чиято импулсна характеристика има крайна продължителност. Те представляват един от основните класове цифрови филтри и се характеризират с това, че стойността на изходния сигнал в даден момент от дискретното време зависи от текущата и определен брой предходни стойности на входния сигнал, но не зависи от предходни стойности на изхода. Поради отсъствието на обратна връзка FIR филтрите се наричат още нерекурсивни филтри.

За FIR филтър от ред (N) входно-изходната зависимост се описва чрез разностното уравнение:

$$y[n] = \sum_{k=0}^{N} b_k \cdot x[n-k]$$

където $b_k$ са коефициентите на филтъра. Съответната предавателна функция в z-областта има вида:

$$H(z) = \sum_{k=0}^{N} b_k \cdot z^{-k}.$$

FIR филтрите притежават няколко важни предимства. Те сравнително лесно се реализират програмно, а при каузална реализация импулсната им характеристика е нулева за отрицателни стойности на дискретното време. Поради крайната дължина на импулсната характеристика условието за BIBO устойчивост

$$\sum_{-\infty}^{\infty} |h[n]| < \infty$$

винаги е изпълнено при крайни коефициенти, поради което FIR филтрите са безусловно устойчиви.

Съществено предимство на FIR филтрите е възможността да бъдат реализирани с точно линейна фазово-честотна характеристика. При подходяща симетрия или антисиметрия на коефициентите, фазата може да се представи във вида

$$\angle H(e^{j\omega}) = -\omega\tau + \varphi_0,$$

където $\tau$ е постоянно групово закъснение. За симетричен FIR филтър с ред (N) то е $\tau = \frac{N}{2}$ дискретни отсчета. Постоянното групово закъснение означава, че честотните

компоненти в пропускателната област се забавят еднакво, което позволява формата на сигнала да се запази без фазови изкривявания.

Основният недостатък на FIR филтрите е, че за изпълнение на една и съща амплитудно-честотна спецификация обикновено е необходим по-висок ред в сравнение с IIR филтрите. Това води до по-голям брой аритметични операции, по-високи изисквания към изчислителните ресурси и по-голямо групово закъснение.

*2.1. Описание на филтрите и изисквания към тях*

При проектирането на цифрови филтри е необходимо ясно да се формулират изискванията към тяхната честотна характеристика, тъй като именно те определят поведението на филтъра в различните честотни области. Тези изисквания служат като основа за избор на тип филтър, ред и метод на синтез и определят доколко полученият филтър ще изпълнява поставената задача.

В зависимост от честотния диапазон, който пропускат, цифровите филтри се разделят на няколко основни типа. Нискочестотните филтри (Low-pass) пропускат ниските честоти и потискат високите, докато високочестотните филтри (High-pass) пропускат високите и потискат ниските честоти. Лентовопропускащите филтри (Band-pass) пропускат честоти в определен диапазон, а лентовозадържащите или режекторните филтри (Band-stop) потискат честотите в зададен интервал.

Честотната характеристика на реалния филтър обикновено се разделя на пропускателна, преходна и спирачна област. В пропускателната област сигналите трябва да преминават с минимално изменение на амплитудата, докато в спирачната област нежеланите честотни компоненти трябва да бъдат достатъчно силно потиснати. Между тези две области се намира преходната зона, в която амплитудната характеристика преминава от пропускане към затихване. Ширината на тази зона е един от основните показатели за селективността на филтъра.

За количественото описание на филтъра се използват граничните честоти на пропускателната и спирачната област, допустимите пулсации в пропускателната област и минималното затихване в спирачната област. Граничните честоти могат да бъдат зададени в Hz или в нормализиран вид спрямо честотата на Найкуист. Ширината на преходната зона оказва пряко влияние върху необходимия ред на филтъра – колкото по-тясна е тя, толкова по-висок ред обикновено е необходим.

Идеалният цифров филтър би имал рязък преход между пропускателната и спирачната област, нулеви пулсации и пълно потискане на нежеланите честоти. Такъв филтър обаче не може да бъде реализиран с крайна импулсна характеристика. Следователно практическото проектиране винаги представлява компромис между ширината на преходната зона, допустимите пулсации, затихването в спирачната област, реда на филтъра и необходимите изчислителни ресурси.

*2.2. Основни параметри при синтез*

Синтезът на FIR филтър започва с определяне на неговия тип и на изискванията към желаната амплитудно-честотна характеристика. В зависимост от приложението филтърът може да бъде нискочестотен, високочестотен, лентов или режекторен. След избора на типа се задават честотата на дискретизация и граничните честоти на съответните честотни области.

$$W = \frac{f}{f_s/2}$$

където $F_s$ е честотата на дискретизация, а $f$ е съответната гранична честота. При тази нормализация стойността $W = 1$ съответства на честотата на Найкуист - $\frac{F_s}{2}$.

Важен параметър е допустимото отклонение на амплитудната характеристика в пропускателната област. То определя доколко амплитудите на полезните честотни компоненти могат да се различават от желаните стойности. В зависимост от използвания метод това изискване може да се задава като пулсация $R_p$ в dB или като линейно допустимо отклонение.

В спирачната област се задава необходимото минимално затихване $R_s$, обикновено в dB. По-голямото затихване осигурява по-ефективно потискане на нежеланите честотни компоненти, но обикновено изисква по-висок ред на филтъра.

Особено значение има ширината на преходната зона. За нискочестотен филтър тя може да бъде определена като

$$\Delta \mathrm{f} = \mathrm{f}_{stop} - \mathrm{f}_{\mathrm{pass}},$$

където $\mathrm{f}_{\mathrm{pass}}$ е граничната честота на пропускателната област, а $\mathrm{f}_{stop}$ – началната честота на областта на задържане. При лентовите филтри съществуват две преходни области, които могат да имат различна ширина. Намаляването на преходната зона увеличава селективността, но води до необходимост от по-висок ред.

Редът N определя броя на закъснителните елементи и е пряко свързан с броя на коефициентите на FIR филтъра, който е N+1. Увеличаването на реда позволява по-добро приближаване до желаната честотна характеристика, но увеличава изчислителната сложност и груповото закъснение. При линейнофазов FIR филтър груповото закъснение е N/2 отсчета.

Следователно параметрите на FIR филтъра са взаимно свързани. Стесняването на преходната зона, намаляването на допустимите пулсации или увеличаването на затихването в спирачната област обикновено изискват увеличаване на реда. Поради това синтезът представлява компромис между точността на апроксимацията, селективността, изчислителната сложност и допустимото закъснение.

*2.3. Основни методи за синтез на FIR филтри*

FIR филтрите могат да бъдат проектирани чрез различни методи, които се отличават по начина, по който апроксимират желаната честотна характеристика. Сред най-често използваните са прозоречният метод, методът на Паркс–МакКлелан и методът на най-малките квадрати.

При прозоречния метод се започва от идеална импулсна характеристика, която по принцип е безкрайна по продължителност. За получаване на реално реализуем FIR филтър тя се ограничава до краен брой коефициенти чрез умножение с прозоречна функция. Така крайният филтър може да се представи като

$$\mathrm{h}[\mathrm{n}] = \mathrm{h}_\mathrm{d}[\mathrm{n}].\,\mathrm{w}[\mathrm{n}],$$

където $\mathrm{h}_\mathrm{d}[\mathrm{n}]$ е идеалната импулсна характеристика, а $\mathrm{w}[\mathrm{n}]$ е избраният прозорец. Видът на прозореца определя компромиса между ширината на преходната зона и нивото на страничните лобове. Често се използват прозорците Hann, Hamming и Blackman. В MATLAB този подход се реализира чрез функцията `fir1()`.

Методът на Паркс–МакКлелан използва *minmax* апроксимация и търси филтър, за който максималната претеглена грешка между желаната и реализираната честотна характеристика е минимална. Получената характеристика има приблизително равномерни пулсации (equiripple) в зададените честотни области. Методът позволява ефективно използване на зададения ред и често осигурява по-тясна преходна зона в сравнение с прозоречния метод. В MATLAB той се реализира чрез функцията `firpm()`.

При метода на най-малките квадрати коефициентите на FIR филтъра се определят така, че интегралната или средноквадратичната грешка между желаната и реализираната честотна характеристика да бъде минимална. За разлика от метода на Паркс–МакКлелан, тук не се минимизира максималното отклонение в отделна точка, а общата

грешка в зададения честотен диапазон. В резултат обикновено се получава гладка характеристика, но максималната локална грешка може да бъде по-голяма. В MATLAB методът се реализира чрез функцията `firls()`.

Изборът на метод за синтез зависи от конкретните изисквания. Прозоречният метод е подходящ, когато се търсят простота и ясна връзка между параметрите на прозореца и характеристиката на филтъра. Методът на Паркс–МакКлелан е подходящ, когато е необходимо ефективно използване на реда и строг контрол върху максималната грешка, докато методът на най-малките квадрати е предпочитан, когато основната цел е минимизиране на общата грешка в честотния диапазон.

*2.4. Влияние на прозоречната функция при синтеза на FIR филтри*

При прозоречния метод изборът на прозоречна функция оказва пряко влияние върху характеристиките на получения FIR филтър. Правоъгълният прозорец осигурява сравнително тясна преходна зона, но се характеризира с високи странични лобове. Прозорците Hann и Hamming намаляват страничните лобове за сметка на известно разширяване на преходната зона, докато Blackman осигурява още по-силно потискане на страничните лобове, но води до по-широк преход.

Следователно изборът на прозорец също представлява компромис между селективност и затихване. Не съществува прозоречна функция, която едновременно да осигурява минимална ширина на преходната зона и максимално потискане в спирачната област.

**III Подготовка за упражнението**

Използвани функции в MATLAB:

- `fir1()` – проектира FIR филтър чрез прозоречния метод. Позволява синтез на нискочестотни, високочестотни, лентовопропускащи и лентовозадържащи филтри.
- `firpm()` – проектира оптимален FIR филтър по метода на Паркс–МакКлелан с равномерни пулсации (equiripple) в зададените честотни области.
- `firls()` – проектира FIR филтър по метода на най-малките квадрати, като минимизира средноквадратичната грешка между желаната и получената честотна характеристика.
- `hamming()` – генерира прозоречна функция на Hamming с определен брой отсчети. Използва се за ограничаване на идеалната импулсна характеристика при прозоречния метод.
- `hann()` – генерира прозоречна функция на Hann. Използва се при прозоречния метод за контрол на преходната зона и страничните лобове.
- `blackman()` – генерира прозоречна функция на Blackman, която осигурява по-силно потискане на страничните лобове за сметка на по-широка преходна зона.
- `unwrap()` – премахва скоковете от $2\pi$ във фазовата характеристика и позволява по-ясното представяне на линейната зависимост на фазата от честотата.
- `fft()` и `conj()` – могат да се използват при ръчно изчисляване и анализ на амплитудната или мощностната честотна характеристика.

**IV Задачи за изпълнение**

1. Да се проектира нискочестотен FIR филтър чрез прозоречния метод с помощта на функцията `fir1()`.

   За целта първо задайте честота на дискретизация $f_s = 200\,Hz$, гранична честота на филтъра $f_c = 30\,Hz$ и ред на филтъра $N = 40$. Изчислете

нормализираната гранична честота спрямо честотата на Найкуист и използвайте прозорец на Hamming за синтез на филтъра. След проектирането анализирайте характеристиките на получения филтър, като изобразите неговата амплитудна и фазова характеристика, импулсната му реакция и груповото закъснение.

Код в MATLAB:

```
% FIR нискочестотен филтър чрез прозоречен метод

% Параметри
fs = 200;          % честота на дискретизация [Hz]
fc = 30;           % гранична честота [Hz]
N  = 40;           % ред на филтъра

% Нормализирана честота
Wn = fc/(fs/2);

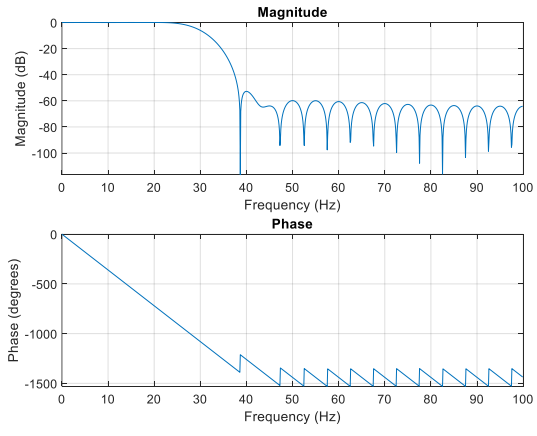
% Синтез на FIR филтър с Hamming прозорец
b = fir1(N, Wn, 'low', hamming(N+1));

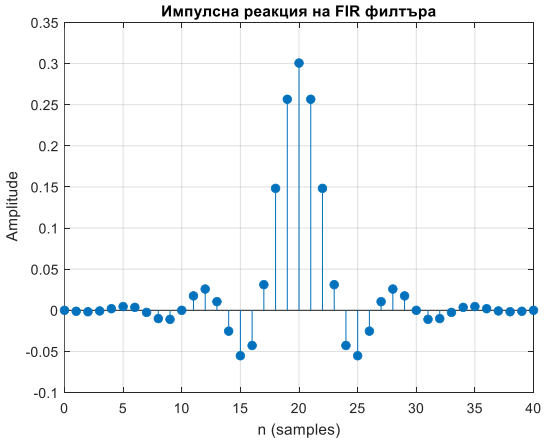
% Амплитудна и фазова характеристика
figure('Name','Задание 1 - Честотна характеристика');
freqz(b, 1, 1024, fs);

% Импулсна реакция
figure('Name','Задание 1 - Импулсна реакция');
impz(b, 1);
grid on;
title('Импулсна реакция на FIR филтъра');

% Групово закъснение
figure('Name','Задание 1 - Групово закъснение');
grpdelay(b, 1, 1024, fs);
grid on;
title('Групово закъснение на FIR филтъра');
```

```
--- Резултати ---
```




2. Да се апроксимират и визуализират АЧХ на НЧ аналогов филтър-прототип на Бътъруърд от 5, 7 и 9 ред.

Код в MATLAB:

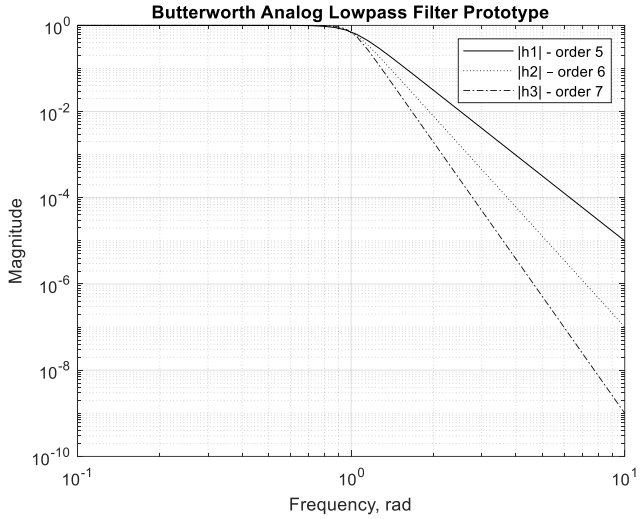
```
[z1,p1,k1] = buttap(5);
[z2,p2,k2] = buttap(7);
[z3,p3,k3] = buttap(9);
w = logspace(-1,1);

h1=freqs(k1*poly(z1), poly(p1),w);
h2=freqs(k2*poly(z2), poly(p2),w);
h3=freqs(k3*poly(z3), poly(p3),w);

loglog(w,abs(h1),'k',w,abs(h2),'k:',w,abs(h3),'k-.'); grid
h=legend('|h1| - order 5','|h2| – order 6','|h3| - order 7');

title('Butterworth Analog Lowpass Filter Prototype');
xlabel('Frequency, rad');
ylabel('Magnitude');
```

```
--- Резултати ---
```




3. Да се изследва влиянието на различни прозоречни функции върху характеристиките на FIR филтър. Използвайте същите стойности за честота на дискретизация, гранична честота и ред на филтъра, както в предходната задача. Проектирайте три нискочестотни FIR филтъра чрез fir1(), като последователно използвате прозорците на Hamming, Hann и Blackman.

   След синтеза определете честотните характеристики на трите филтъра и ги представете в общ графичен прозорец. Сравнете получените резултати по отношение на широчината на преходната област, нивото на страничните лобове и степента на потискане в лентата на задържане.

Код в MATLAB:

```
% Сравнение на FIR филтри при различни прозорци

% Параметри
fs = 200;
fc = 30;
N  = 40;
Wn = fc/(fs/2);
```

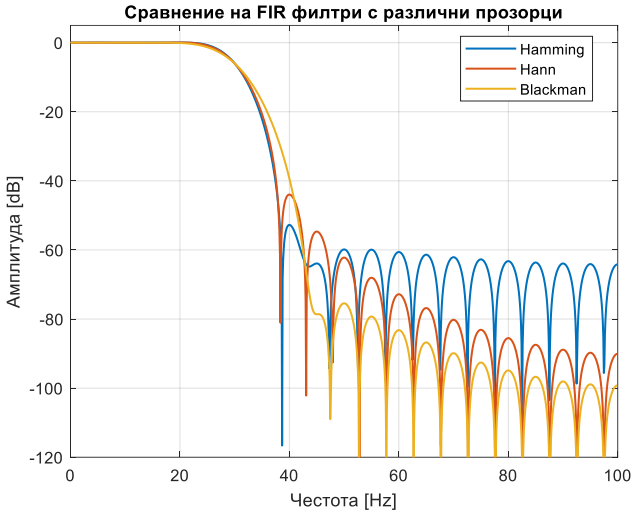
```
% FIR филтри с различни прозорци
b_hamming  = fir1(N, Wn, 'low', hamming(N+1));
b_hann     = fir1(N, Wn, 'low', hann(N+1));
b_blackman = fir1(N, Wn, 'low', blackman(N+1));

% Честотни характеристики
[H1, f] = freqz(b_hamming,  1, 1024, fs);
H2 = freqz(b_hann,     1, 1024, fs);
H3 = freqz(b_blackman, 1, 1024, fs);

figure('Name','Задание 2 - Сравнение на прозорци');
plot(f, 20*log10(abs(H1)+eps), 'LineWidth', 1.2); hold on; grid on;
plot(f, 20*log10(abs(H2)+eps), 'LineWidth', 1.2);
plot(f, 20*log10(abs(H3)+eps), 'LineWidth', 1.2);

xlabel('Честота [Hz]');
ylabel('Амплитуда [dB]');
title('Сравнение на FIR филтри с различни прозорци');
legend('Hamming', 'Hann', 'Blackman', 'Location', 'best');
xlim([0 100]);
ylim([-120 5]);
```

```
--- Резултати ---
```




5. Да се проектира FIR филтър чрез метода на Паркс–МакКлелан, реализиран в MATLAB чрез функцията firpm(). Задайте ред на филтъра $N$=40 и определете желана нискочестотна характеристика чрез нормализирани честоти и съответните желани амплитудни стойности в лентата на пропускане и в лентата на задържане. Синтезирайте филтъра и представете неговата честотна характеристика, импулсна реакция и групово закъснение.

Код в MATLAB:

```
% FIR филтър чрез Parks-McClellan
clc; clear; close all;

% Параметри
fs = 200;
N  = 40;
```

```
% Нормализирани честоти (спрямо Nyquist)
f = [0 0.3 0.4 1];     % 0...1 съответства на 0...(fs/2)
a = [1 1 0 0];         % желана характеристика

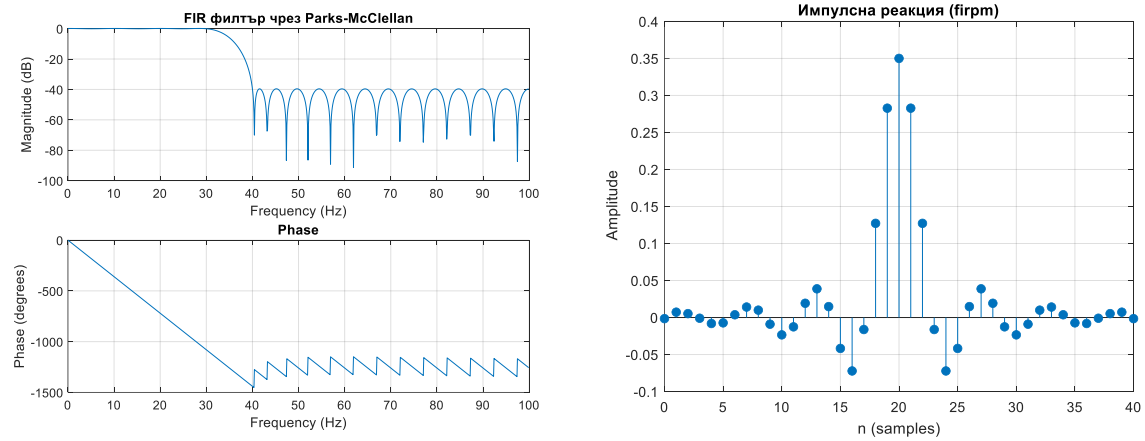
% Синтез чрез firpm
b_pm = firpm(N, f, a);

% Анализ
figure('Name','Задание 3 - Честотна характеристика');
freqz(b_pm, 1, 1024, fs);
title('FIR филтър чрез Parks-McClellan');

figure('Name','Задание 3 - Импулсна реакция');
impz(b_pm, 1);
grid on;
title('Импулсна реакция (firpm)');

figure('Name','Задание 3 - Групово закъснение');
grpdelay(b_pm, 1, 1024, fs);
grid on;
title('Групово закъснение (firpm)');
```

```
--- Резултати ---
```




6. Да се демонстрира приложението на FIR филтър за обработка на реален сигнал. Генерирайте тестов сигнал, състоящ се от полезна нискочестотна синусоидална компонента и добавен шум. Използвайте FIR нискочестотния филтър, синтезиран в Задание 1, за да филтрирате сигнала с помощта на функцията `filter()`.

   След филтрирането представете входния и изходния сигнал във времевата област и сравнете техния вид. След това изчислете едностранния амплитуден спектър на входния и изходния сигнал и анализирайте кои честотни компоненти са били потиснати.

Код в MATLAB:

```
% Филтриране на зашумен сигнал с FIR филтър

%% Параметри
fs = 200;              % честота на дискретизация [Hz]
T  = 2;                % продължителност на сигнала [s]
```

```
t  = 0:1/fs:T-1/fs;

fc = 30;              % гранична честота на FIR филтъра [Hz]
N  = 40;              % ред на филтъра

% Нормализирана гранична честота
Wn = fc/(fs/2);

%% Синтез на FIR нискочестотен филтър
% Използва се прозорец Hamming
b = fir1(N, Wn, 'low', hamming(N+1));

%% Генериране на тестов сигнал
% Полезна нискочестотна синусоидална компонента
f_signal = 10;        % честота на полезния сигнал [Hz]

% Бял гаусов шум
noise = 0.5*randn(size(t));

% Входен зашумен сигнал
x = sin(2*pi*f_signal*t) + noise;

%% Филтриране
y = filter(b, 1, x);

%% Визуализация във времевата област
figure('Name','Задание 6 - Времева област');

subplot(2,1,1);
plot(t, x, 'LineWidth', 1.0);
grid on;
xlabel('Време [s]');
ylabel('Амплитуда');
title('Входен зашумен сигнал');

subplot(2,1,2);
plot(t, y, 'LineWidth', 1.2);
grid on;
xlabel('Време [s]');
ylabel('Амплитуда');
title('Изходен сигнал след FIR филтриране');

%% ============================================================
% Едностранен амплитуден спектър на входния сигнал
%% ============================================================

Nx = length(x);

% FFT на входния сигнал
X = fft(x);

% Нормализиран двустранен амплитуден спектър
Xmag = abs(X)/Nx;

% Получаване на едностранен спектър
if mod(Nx,2) == 0

    X1 = Xmag(1:Nx/2+1);
    X1(2:end-1) = 2*X1(2:end-1);
```

```
    fX = (0:Nx/2)*(fs/Nx);

else

    X1 = Xmag(1:(Nx+1)/2);
    X1(2:end) = 2*X1(2:end);

    fX = (0:(Nx-1)/2)*(fs/Nx);

end

%% ============================================================
% Едностранен амплитуден спектър на изходния сигнал
%% ============================================================

Ny = length(y);

% FFT на филтрирания сигнал
Y = fft(y);

% Нормализиран двустранен амплитуден спектър
Ymag = abs(Y)/Ny;

% Получаване на едностранен спектър
if mod(Ny,2) == 0

    Y1 = Ymag(1:Ny/2+1);
    Y1(2:end-1) = 2*Y1(2:end-1);

    fY = (0:Ny/2)*(fs/Ny);

else

    Y1 = Ymag(1:(Ny+1)/2);
    Y1(2:end) = 2*Y1(2:end);

    fY = (0:(Ny-1)/2)*(fs/Ny);

end

%% Визуализация в честотната област
figure('Name','Задание 6 - Честотна област');

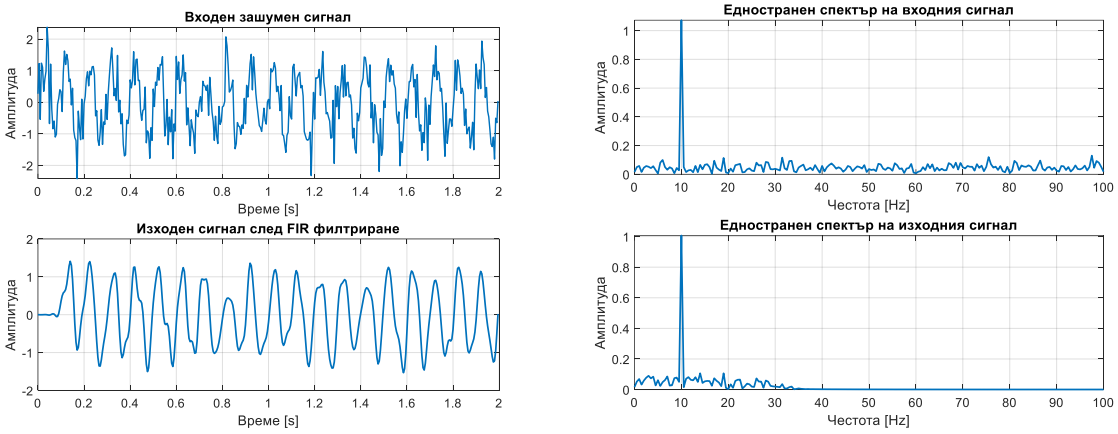
subplot(2,1,1);
plot(fX, X1, 'LineWidth', 1.2);
grid on;
xlabel('Честота [Hz]');
ylabel('Амплитуда');
title('Едностранен спектър на входния сигнал');
xlim([0 fs/2]);

subplot(2,1,2);
plot(fY, Y1, 'LineWidth', 1.2);
grid on;
xlabel('Честота [Hz]');
ylabel('Амплитуда');
title('Едностранен спектър на изходния сигнал');
xlim([0 fs/2]);
```

```
%% Извеждане на основните параметри
disp(' ');
disp('--- Параметри на филтъра ---');

disp(['Честота на дискретизация: ', num2str(fs), ' Hz']);
disp(['Гранична честота: ', num2str(fc), ' Hz']);
disp(['Ред на FIR филтъра: ', num2str(N)]);
disp(['Теоретично групово закъснение: ', num2str(N/2), ' отсчета']);
```

```
--- Резултати ---
--- Параметри на филтъра ---
Честота на дискретизация: 200 Hz
Гранична честота: 30 Hz
Ред на FIR филтъра: 40
Теоретично групово закъснение: 20 отсчета
```




## V Допълнителни задачи

1. Увеличете реда на филтъра и анализирайте ефекта.
2. Проектирайте highpass филтър.
3. Използвайте `fvtool()` за сравнение на характеристиките.
4. Сравнете функциите `fir1` и `firpm`

Код в MATLAB:

```
% Сравнение между fir1 и firpm
clc; clear; close all;

fs = 200;
fc = 30;
N  = 40;

Wn = fc/(fs/2);

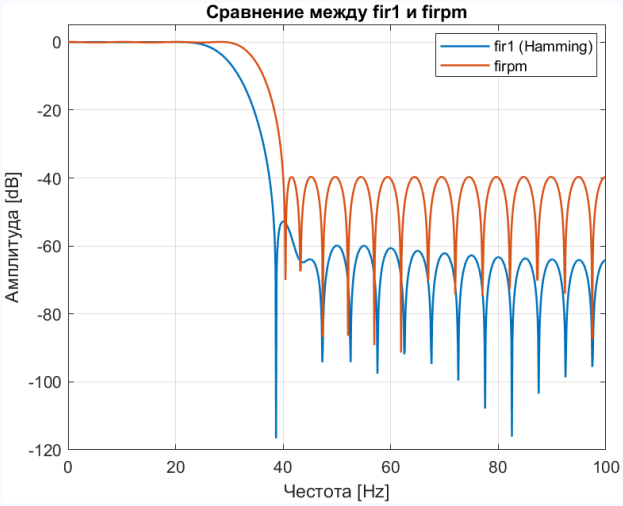
% FIR чрез прозоречен метод
b_fir1 = fir1(N, Wn, 'low', hamming(N+1));

% FIR чрез Parks-McClellan
f = [0 0.3 0.4 1];
a = [1 1 0 0];
b_firpm = firpm(N, f, a);
```

```
% Честотни характеристики
[H1, freq] = freqz(b_fir1, 1, 1024, fs);
H2 = freqz(b_firpm, 1, 1024, fs);

figure('Name','Сравнение fir1 vs firpm');
plot(freq, 20*log10(abs(H1)+eps), 'LineWidth', 1.2); hold on; grid on;
plot(freq, 20*log10(abs(H2)+eps), 'LineWidth', 1.2);

xlabel('Честота [Hz]');
ylabel('Амплитуда [dB]');
title('Сравнение между fir1 и firpm');
legend('fir1 (Hamming)', 'firpm', 'Location', 'best');
xlim([0 100]);
ylim([-120 5]);
```




5. Да се анализира линейнофазовото свойство на FIR филтър и да се изследва неговото групово закъснение. Използвайте FIR филтъра, синтезиран в Задание 1. Изчислете фазовата характеристика и груповото закъснение на филтъра. Представете графично фазата и груповото закъснение и проверете дали в лентата на пропускане груповото закъснение е приблизително постоянно.

Код в MATLAB:

```
% Анализ на линейна фаза и групово закъснение
clc; clear; close all;

% Параметри
fs = 200;
fc = 30;
N  = 40;
Wn = fc/(fs/2);

% FIR филтър
b = fir1(N, Wn, 'low', hamming(N+1));

% Теоретично групово закъснение
tau_theory = N/2;
disp(['Теоретично групово закъснение N/2 = ', num2str(tau_theory), ' проби']);

% Фазова характеристика
```

```
[H, f] = freqz(b, 1, 1024, fs);
phaseH = unwrap(angle(H));

figure('Name','Задание 4 - Фазова характеристика');
plot(f, phaseH, 'LineWidth', 1.2); grid on;
xlabel('Честота [Hz]');
ylabel('Фаза [rad]');
title('Фазова характеристика на FIR филтъра');

% Групово закъснение
[gd, fg] = grpdelay(b, 1, 1024, fs);

figure('Name','Задание 4 - Групово закъснение');
plot(fg, gd, 'LineWidth', 1.2); grid on;
xlabel('Честота [Hz]');
ylabel('Групово закъснение [проби]');
title('Групово закъснение на FIR филтъра');
ylim([0 N]);

% Показване на средна стойност в пропускателната област
passband_idx = fg <= fc;
gd_mean = mean(gd(passband_idx));
disp(['Средно групово закъснение в пропускателната област: ', num2str(gd_mean)]);
```

## VI Контролни въпроси

1. Какво означава линейна ФЧХ?
2. Защо FIR филтрите винаги са устойчиви?
3. Как влияе използваният прозорец върху спектъра на филтъра?
4. Каква е разликата между функциите fir1 и firpm?
5. Какво е значението на груповото закъснение?

**Упражнение №10**

# Филтри с безкрайна импулсна характеристика - синтез и анализ

**I Цел на упражнението** – Синтез, реализация и анализ на цифрови филтри с безкрайна импулсна харатеристика с помощта на Signal Processing Toolbox. Проектиране на нискочестотни, високочестотни, лентови и режекторни филтри по класически апроксимации (Butterworth, Chebyshev I/II, Elliptic), анализиране на амплитудно-фазовите характеристики, груповото закъснение, импулсната и стъпковата реакция, както и ефекта от реда на филтъра и параметрите на рипъл/затихване.

**II Теоретична част**

Филтрите с безкрайна импулсна характеристика (Infinite Impulse Response, IIR) са основен клас цифрови филтри, при които стойността на изходната реакция в определен момент от дискретното време зависи от стойностите на входния сигнал в текущия и предходни моменти, както и от стойностите на изходния сигнал в предходни моменти от време. Наричат се още рекурсивни или филтри с обратна връзка.

Най същественото предимство на IIR филтрите в сравнение с филтрите с крайна импулсна характеристика (Finite Impulse Respond, FIR) е, че те обикновено удовлетворяват зададена съвкупност от изисквания при много по-нисък ред на филтъра от съответния FIR филтър. При избор на IIR филтър от по-висок ред обаче, той може да се окаже нестабилен.

IIR (Infinite Impulse Response) филтър е цифров филтър с обратна връзка, т.е. изходът зависи и от предишни изходни стойности. Общата форма на разностното уравнение е:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$$

или еквивалентно - предавателната функция:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}.$$

Честотната характеристика се получава при $z = e^{j\omega}$ и описва амплитудно усилване/затихване $\left|H(e^{j\omega})\right|$ и фазово отместване $\angle H\left(e^{j\omega}\right)$.

*2.1. Описание на филтрите и изисквания към тях*

Signal Processing Toolbox съдържа функции, които реализират различни алгоритми за обработка на сигнали. С наличните функции може да се извършва анализ, синтез и реализация на цифрови филтри с крайна и безкрайна ИХ в честотната и времевата област (преобразуване на Фурие, интерполация, двумерна обработка на сигнали и др.).

Нормализация на честотата в SPT – всички функции за проектиране на филтри работят с нормализирани честоти, така че те не изискват честотата на дискретизация като допълнителен входен аргумент.

Функциите на инструментариума за избор на реда на филтъра оценяват минималния ред на филтъра, който удовлетворява дадена съвкупност от изисквания.

Основната техника за проектиране на IIR цифров филтър, която предоставя SPT, е базирана на преобразуването на класическите НЧ аналогови филтри към техните цифрови еквиваленти.

*2.2. Основни параметри при синтез*

Проектирането на IIR филтър обикновено се задава чрез:

- тип: Ниско честотен (Low Pass), Високочестотен (High Pass), Лентов (Band Pass), Режекторен (Band Stop);
- гранични честоти (BP, BS);
- пулсации в пропускателната област $R_p\ [dB]$;
- затихване $R_s[dB]$;
- ред $N$ или автоматично определяне на минимален ред.

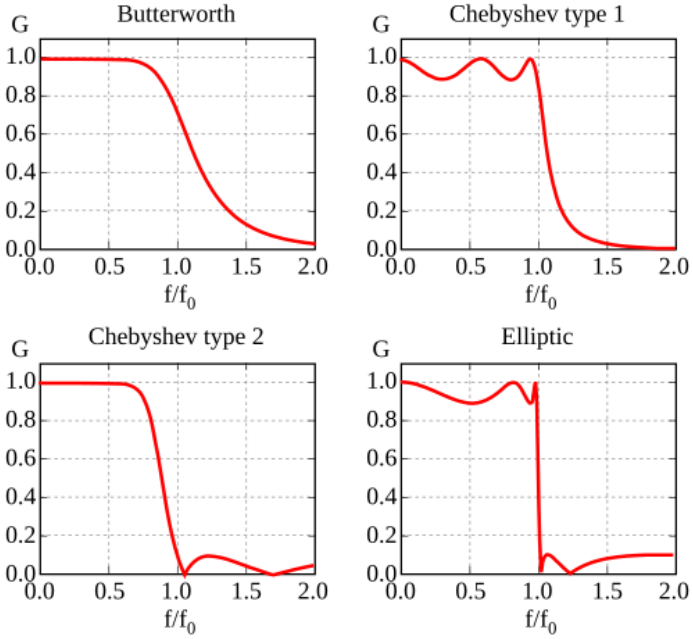
В MATLAB честотите често са нормализирани:

$$W_n = \frac{f_c}{f_s/2} \in (0,1),$$

където $f_s/2$ е честотата на Найкуист.

*2.3. Класически апроксимации (семейства IIR)*

Представено е изображение, сравняващо филтрите на Бътъруърт, Чебишев и елиптичните филтри. Филтрите на тази илюстрация са нискочестотни филтри от пети ред. Конкретната имплементация – аналогова или цифрова, пасивна или активна – няма значение; изходният им сигнал би бил един и същ. Както е ясно от изображението, елиптичните филтри са с най-рязък преход, но показват пулсации по цялата честотна лента.




*2.3.1 Butterworth* - Честотната характеристика на филтъра на Бътъруърт е максимално плоска в лентата на пропускане и намалява към нула в лентата на спиране. Характеристиката на филтър от първи ред намалява с −6 dB на октава, филтър от втори ред намалява с −12 dB на октава, филтър от трети ред с −18 dB и т.н. Филтрите на Бътъруърт имат монотонно променяща се функция на величината с ω, за разлика от други типове филтри, които имат немонотонни пулсации в лентата на пропускане и/или лентата на спиране.

*2.3.2 Chebyshev* - Филтрите на Чебишев са аналогови или цифрови филтри, които имат по-стръмен спад от филтрите на Бътъруърт и имат или пулсации в лентата на пропускане (тип I), или пулсации в лентата на спиране (тип II). Филтрите на Чебишев имат свойството да минимизират грешката между идеализираната и действителната характеристика на филтъра в работния честотен диапазон на филтъра, но постигат това с пулсации в честотната характеристика.

*2.3.3 Elliptic* - Елиптичен филтър е филтър за обработка на сигнали с изравнено поведение на пулсации както в лентата на пропускане, така и в лентата на спиране. Количеството пулсации във всяка лента е независимо регулируемо и никой друг филтър от същия порядък не може да има по-бърз преход в усилването между лентата на пропускане и лентата на спиране за дадените стойности на пулсациите (независимо дали пулсациите са изравнени или не).

*2.4. Сравнение между FIR и IIR филтри*

FIR и IIR филтрите се различават както по структура, така и по своите характеристики. FIR филтрите са нерекурсивни и при крайни коефициенти са винаги устойчиви. Те могат да бъдат синтезирани с точно линейна фаза, което е важно при приложения, в които трябва да се запази формата на сигнала. Недостатъкът им е необходимостта от по-висок ред и съответно по-голям брой изчисления.

IIR филтрите използват обратна връзка и обикновено постигат дадена амплитудна спецификация с по-нисък ред. Това ги прави по-ефективни от изчислителна гледна точка, но устойчивостта им трябва да бъде проверявана чрез разположението на полюсите. Освен това фазовата им характеристика по принцип е нелинейна.

| Характеристика | FIR | IIR |
|---|---|---|
| Импулсна характеристика | Крайна | Безкрайна |
| Структура | Нерекурсивна | Обикновено рекурсивна |
| Устойчивост | Гарантирана при крайни коефициенти | Зависи от положението на полюсите |
| Линейна фаза | Може да бъде реализирана точно | Обикновено не |
| Необходим ред | По-висок | По-нисък |
| Изчислителна сложност | По-голяма | По-малка |
| Групово закъснение | Често по-голямо и постоянно при линейна фаза | Честотно зависимо |

При избора между FIR и IIR трябва да се отчита конкретното приложение. Когато запазването на формата и фазовите съотношения между честотните компоненти е критично, FIR филтрите често са предпочитани. Когато основните изисквания са ниска изчислителна сложност и висока селективност при малък ред, IIR филтрите могат да бъдат по-подходящи.

## III Подготовка за упражнението

Използвани функции в MATLAB:

- `butter()` – проектира цифров или аналогов Butterworth филтър, характеризиращ се с максимално гладка амплитудна характеристика в пропускателната област.

- `cheby1()` – проектира Chebyshev Type I филтър, който допуска зададен рипъл в пропускателната област с цел постигане на по-стръмен преход.
- `cheby2()` – проектира Chebyshev Type II филтър**,** който има гладка пропускателна област и зададено минимално затихване с рипъл в спирачната област.
- `ellip()` – проектира елиптичен филтър, който допуска рипъл както в пропускателната, така и в спирачната област и осигурява висока селективност при сравнително нисък ред.
- `freqz()` – изчислява и визуализира честотната характеристика на цифров филтър, включително амплитудната и фазовата характеристика.
- `impz()` – изчислява и визуализира импулсната реакция на цифров филтър.
- `grpdelay()` – изчислява и визуализира груповото закъснение на цифров филтър като функция на честотата.
- `filter()` – извършва цифрово филтриране на входен сигнал чрез зададените коефициенти на числителя и знаменателя на предавателната функция.
- `filtfilt()` – извършва zero-phase цифрово филтриране, като обработва сигнала последователно в права и обратна посока и по този начин компенсира фазовото отместване.

**IV Задачи за изпълнение**

1. Да се проектира нискочестотен цифров IIR филтър на Butterworth с ред N=4, честота на дискретизация fs=200 Hz и гранична честота fc=30 Hz. Преди синтеза граничната честота да се нормализира спрямо честотата на Найкуист.
    - Филтърът да се синтезира чрез функцията `butter()`;
    - Да се определят коефициентите на числителя и знаменателя на предавателната функция и да се изобразят амплитудно-честотната и фазово-честотната характеристика;
    - Допълнително да се изследват импулсната реакция и разположението на нулите и полюсите;
    - По положението на полюсите да се определи устойчивостта на филтъра.

Код в MATLAB:

```
% Синтез и анализ на нискочестотен Butterworth IIR филтър

%% Параметри
fs = 200;              % честота на дискретизация [Hz]
fc = 30;               % гранична честота [Hz]
N  = 4;                % ред на филтъра

%% Нормализирана гранична честота
Wn = fc/(fs/2);

%% Синтез на Butterworth филтър
[b,a] = butter(N, Wn, 'low');

%% Извеждане на коефициентите
disp('Коефициенти на числителя b:');
disp(b);

disp('Коефициенти на знаменателя a:');
disp(a);

%% Честотна характеристика
[H,f] = freqz(b,a,1024,fs);
```

```
figure('Name','Задание 1 - Амплитудна характеристика');
plot(f,20*log10(abs(H)+eps),'LineWidth',1.2);
grid on;
xlabel('Честота [Hz]');
ylabel('Амплитуда [dB]');
title('Амплитудно-честотна характеристика');
xlim([0 fs/2]);

%% Фазова характеристика
figure('Name','Задание 1 - Фазова характеристика');
plot(f,unwrap(angle(H)),'LineWidth',1.2);
grid on;
xlabel('Честота [Hz]');
ylabel('Фаза [rad]');
title('Фазово-честотна характеристика');

%% Импулсна реакция
figure('Name','Задание 1 - Импулсна реакция');
impz(b,a,50);
grid on;
title('Импулсна реакция на Butterworth IIR филтъра');

%% Нули и полюси
figure('Name','Задание 1 - Нули и полюси');
zplane(b,a);
grid on;
title('Нули и полюси');

%% Проверка за устойчивост
p = roots(a);

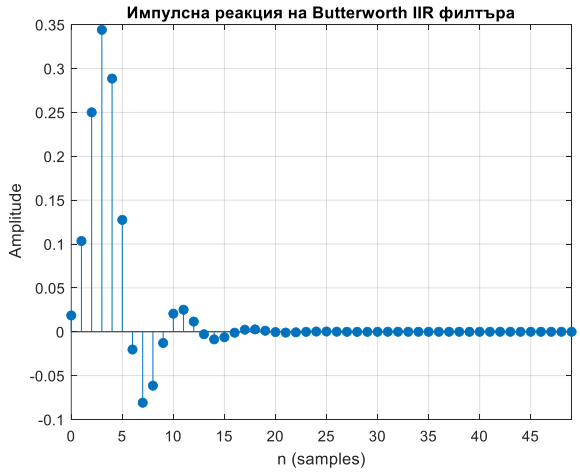
disp('Полюси на филтъра:');
disp(p);

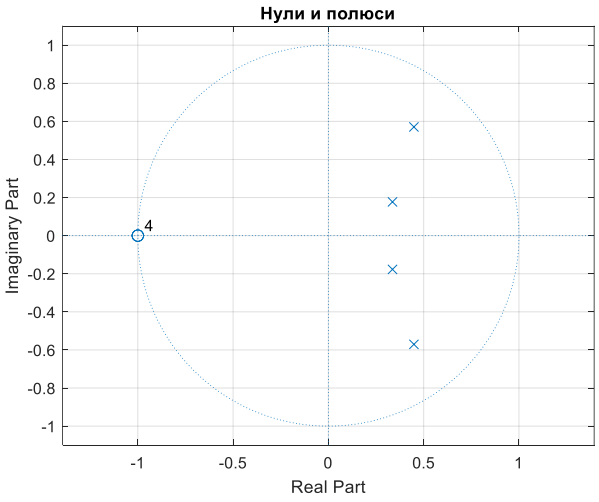
disp('Модули на полюсите:');
disp(abs(p));

if all(abs(p) < 1)
    disp('Филтърът е устойчив.');
else
    disp('Филтърът НЕ е устойчив.');
end
```

```
--- Резултати ---
Коефициенти на числителя b:
  0.0186    0.0743    0.1114    0.0743    0.0186
Коефициенти на знаменателя a:
  1.0000   -1.5704    1.2756   -0.4844    0.0762
Полюси на филтъра:
   0.4488 + 0.5707i
   0.4488 - 0.5707i
   0.3364 + 0.1772i
   0.3364 - 0.1772i
Модули на полюсите:
    0.7261
    0.7261
    0.3802
```

```
    0.3802
Филтърът е устойчив.
```




4. Да се изследва влиянието на реда върху честотната характеристика на нискочестотен Butterworth филтър. При fs=200 Hz и fc=30 Hz да се синтезират три филтъра с редове: N1=2, N2=4, N3=8. Амплитудно-честотните характеристики да се представят върху една координатна система в dB. Да се сравнят ширината и стръмността на преходната област. Да се анализира как увеличаването на реда на филтъра влияе върху неговата селективност.

Код в MATLAB:

```
% Влияние на реда върху Butterworth IIR филтър

%% Параметри
fs = 200; fc = 30;

N1 = 2; N2 = 4; N3 = 8;

Wn = fc/(fs/2);

%% Синтез на филтрите
[b1,a1] = butter(N1,Wn,'low');
[b2,a2] = butter(N2,Wn,'low');
[b3,a3] = butter(N3,Wn,'low');

%% Честотни характеристики
[H1,f] = freqz(b1,a1,1024,fs);
H2 = freqz(b2,a2,1024,fs);
H3 = freqz(b3,a3,1024,fs);

%% Визуализация
figure('Name','Задание 2 - Влияние на реда');

plot(f,20*log10(abs(H1)+eps),'LineWidth',1.2);
hold on;

plot(f,20*log10(abs(H2)+eps),'LineWidth',1.2);
plot(f,20*log10(abs(H3)+eps),'LineWidth',1.2);

grid on;

xlabel('Честота [Hz]');
```

```
ylabel('Амплитуда [dB]');
title('Влияние на реда на Butterworth филтъра');

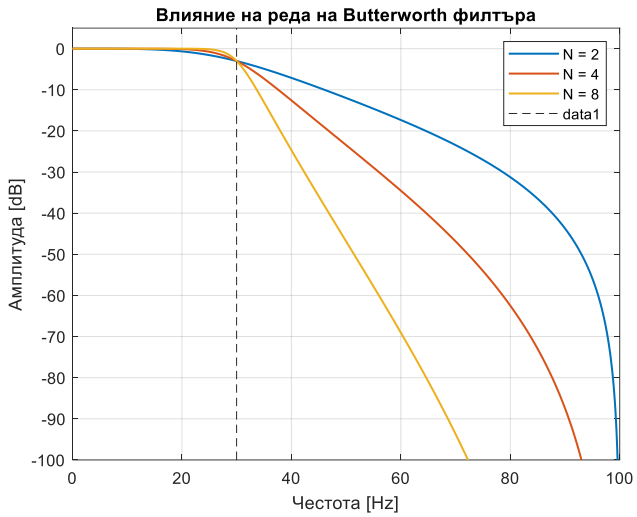
legend('N = 2','N = 4','N = 8');

xlim([0 100]);
ylim([-100 5]);

% Маркиране на граничната честота
plot([fc fc],[-100 5],'--k');

hold off;
```

```
--- Резултати ---
```




3. Да се сравнят четири класически семейства IIR филтри: Butterworth, Chebyshev Type I, Chebyshev Type II и Elliptic. За всички филтри да се използват еднакви:
   - честота на дискретизация fs=200 Hz;
   - ред N=4;
   - гранична честота fc=30 Hz.

   За Chebyshev Type I да се зададе рипъл в пропускателната област Rp=1 dB, а за Chebyshev Type II – минимално затихване Rs=40 dB. За Elliptic филтъра да се използват Rp=1 dB и Rs=40 dB.

   Да се представят амплитудно-честотните характеристики върху обща графика и да се сравнят по отношение на рипъл, стръмност на прехода и затихване.

Код в MATLAB:

```
% Сравнение на класически IIR апроксимации

%% Параметри
fs = 200; fc = 30; N  = 4;
Rp = 1;                % рипъл в пропускателната област [dB]
Rs = 40;               % затихване в спирачната област [dB]

Wn = fc/(fs/2);

%% Синтез
% Butterworth
```

```
[bB,aB] = butter(N,Wn,'low');

% Chebyshev Type I
[bC1,aC1] = cheby1(N,Rp,Wn,'low');

% Chebyshev Type II
[bC2,aC2] = cheby2(N,Rs,Wn,'low');

% Elliptic
[bE,aE] = ellip(N,Rp,Rs,Wn,'low');

%% Честотни характеристики
[HB,f] = freqz(bB,aB,2048,fs);
HC1 = freqz(bC1,aC1,2048,fs);
HC2 = freqz(bC2,aC2,2048,fs);
HE  = freqz(bE,aE,2048,fs);

%% Визуализация
figure('Name','Задание 3 - Сравнение на IIR филтри');

plot(f,20*log10(abs(HB)+eps),'LineWidth',1.2);
hold on;

plot(f,20*log10(abs(HC1)+eps),'LineWidth',1.2);
plot(f,20*log10(abs(HC2)+eps),'LineWidth',1.2);
plot(f,20*log10(abs(HE)+eps),'LineWidth',1.2);

grid on;

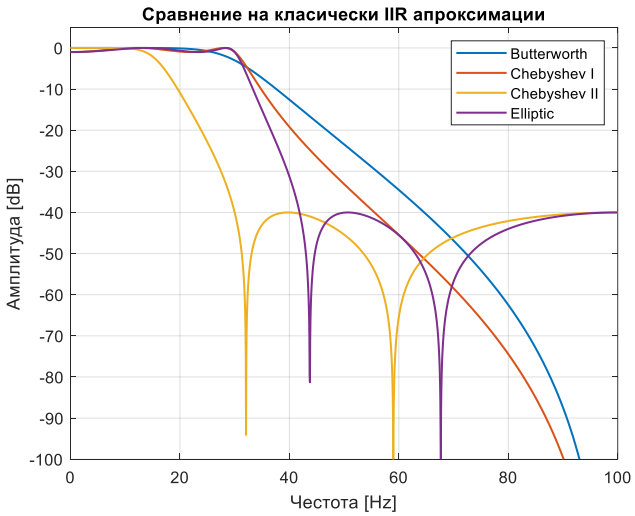
xlabel('Честота [Hz]');
ylabel('Амплитуда [dB]');
title('Сравнение на класически IIR апроксимации');

legend('Butterworth','Chebyshev I','Chebyshev II','Elliptic');

xlim([0 100]);
ylim([-100 5]);

hold off;
```

```
--- Резултати ---
```



4. Да се проектира нискочестотен цифров филтър по предварително зададени изисквания: fs=200 Hz, fp=25 Hz, fstop=40 Hz, Rp=1 dB, Rs=40 dB.
   Да се определят минималните необходими редове за Butterworth, Chebyshev Type I, Chebyshev Type II и Elliptic филтър чрез съответните функции за определяне на реда. След това да се синтезират четирите филтъра и да се сравнят честотните им характеристики.

Код в MATLAB:

```
% Определяне на минимален ред на IIR филтри

%% Спецификация
fs = 200;

fp   = 25;            % край на пропускателната област [Hz]
fstop = 40;           % начало на спирачната област [Hz]

Rp = 1;               % максимален рипъл [dB]
Rs = 40;              % минимално затихване [dB]

%% Нормализирани честоти
Wp = fp/(fs/2);
Ws = fstop/(fs/2);

%% Butterworth
[NB,WnB] = buttord(Wp,Ws,Rp,Rs);
[bB,aB] = butter(NB,WnB,'low');

%% Chebyshev Type I
[NC1,WnC1] = cheb1ord(Wp,Ws,Rp,Rs);
[bC1,aC1] = cheby1(NC1,Rp,WnC1,'low');

%% Chebyshev Type II
[NC2,WnC2] = cheb2ord(Wp,Ws,Rp,Rs);
[bC2,aC2] = cheby2(NC2,Rs,WnC2,'low');

%% Elliptic
[NE,WnE] = ellipord(Wp,Ws,Rp,Rs);
[bE,aE] = ellip(NE,Rp,Rs,WnE,'low');

%% Извеждане на получените редове
disp(' ');
disp('--- Минимални редове ---');

disp(['Butterworth:       N = ',num2str(NB)]);
disp(['Chebyshev Type I:  N = ',num2str(NC1)]);
disp(['Chebyshev Type II: N = ',num2str(NC2)]);
disp(['Elliptic:          N = ',num2str(NE)]);

%% Честотни характеристики
[HB,f] = freqz(bB,aB,2048,fs);
HC1 = freqz(bC1,aC1,2048,fs);
HC2 = freqz(bC2,aC2,2048,fs);
HE  = freqz(bE,aE,2048,fs);

%% Визуализация
figure('Name','Задание 4 - Минимален ред');

plot(f,20*log10(abs(HB)+eps),'LineWidth',1.2);
hold on;
```

```
plot(f,20*log10(abs(HC1)+eps),'LineWidth',1.2);
plot(f,20*log10(abs(HC2)+eps),'LineWidth',1.2);
plot(f,20*log10(abs(HE)+eps),'LineWidth',1.2);

grid on;

xlabel('Честота [Hz]');
ylabel('Амплитуда [dB]');
title('IIR филтри с минимален ред');

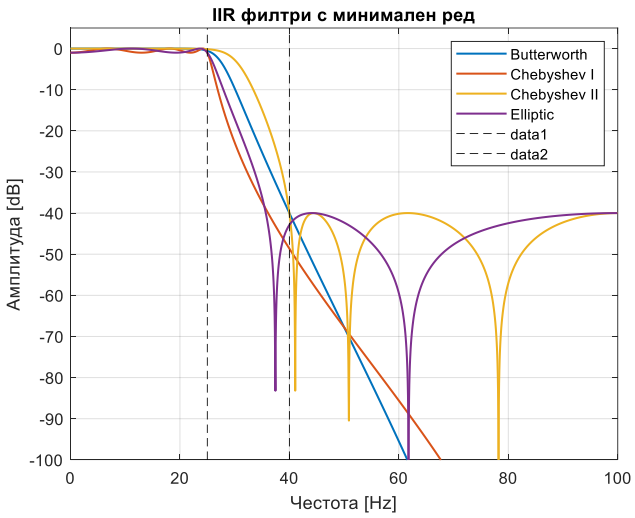
legend('Butterworth','Chebyshev I','Chebyshev II','Elliptic');

xlim([0 100]);
ylim([-100 5]);

% Гранични честоти
plot([fp fp],[-100 5],'--k');
plot([fstop fstop],[-100 5],'--k');

hold off;
```

```
--- Резултати ---
--- Минимални редове ---
Butterworth:       N = 10
Chebyshev Type I:  N = 6
Chebyshev Type II: N = 6
Elliptic:          N = 4
```




5. Да се изследва действието на нискочестотен IIR филтър върху сигнал, съдържащ полезна и нежелана честотна компонента. При честота на дискретизация fs =200 Hz да се генерира сигнал: $x(t)=\sin(2\pi 10t)+0.6\sin(2\pi 60t)$.
   - Компонентата с честота 10 Hz да се разглежда като полезен сигнал, а компонентата 60 Hz – като високочестотно смущение.
   - Да се проектира Butterworth нискочестотен филтър от четвърти ред с гранична честота 30 Hz. Сигналът да се филтрира чрез `filter()`.

- Да се представят входният и изходният сигнал във времевата област и да се изчислят техните едностранни амплитудни спектри. Да се установи дали компонентата 60 Hz е потисната.

Код в MATLAB:

```
% Филтриране на съставен сигнал с IIR филтър

%% Параметри
fs = 200;
T = 2;

t = 0:1/fs:T-1/fs;

%% Генериране на сигнал
f1 = 10;              % полезна компонента [Hz]
f2 = 60;              % нежелана компонента [Hz]

x = sin(2*pi*f1*t) + 0.6*sin(2*pi*f2*t);

%% Синтез на IIR филтър
N = 4;
fc = 30;

Wn = fc/(fs/2);

[b,a] = butter(N,Wn,'low');

%% Филтриране
y = filter(b,a,x);

%% Времева област
figure('Name','Задание 5 - Времева област');

subplot(2,1,1);
plot(t,x,'LineWidth',1.2);
grid on;
xlabel('Време [s]');
ylabel('Амплитуда');
title('Входен сигнал');

subplot(2,1,2);
plot(t,y,'LineWidth',1.2);
grid on;
xlabel('Време [s]');
ylabel('Амплитуда');
title('Сигнал след IIR филтриране');

%% ============== Едностранен спектър на входния сигнал ================
Nx = length(x);

X = fft(x);
Xmag = abs(X)/Nx;

if mod(Nx,2) == 0

    X1 = Xmag(1:Nx/2+1);
    X1(2:end-1) = 2*X1(2:end-1);

    fX = (0:Nx/2)*(fs/Nx);
```

```
else

    X1 = Xmag(1:(Nx+1)/2);
    X1(2:end) = 2*X1(2:end);

    fX = (0:(Nx-1)/2)*(fs/Nx);

end

%% ============= Едностранен спектър на филтрирания сигнал =============
Ny = length(y);

Y = fft(y);
Ymag = abs(Y)/Ny;

if mod(Ny,2) == 0

    Y1 = Ymag(1:Ny/2+1);
    Y1(2:end-1) = 2*Y1(2:end-1);

    fY = (0:Ny/2)*(fs/Ny);

else

    Y1 = Ymag(1:(Ny+1)/2);
    Y1(2:end) = 2*Y1(2:end);

    fY = (0:(Ny-1)/2)*(fs/Ny);

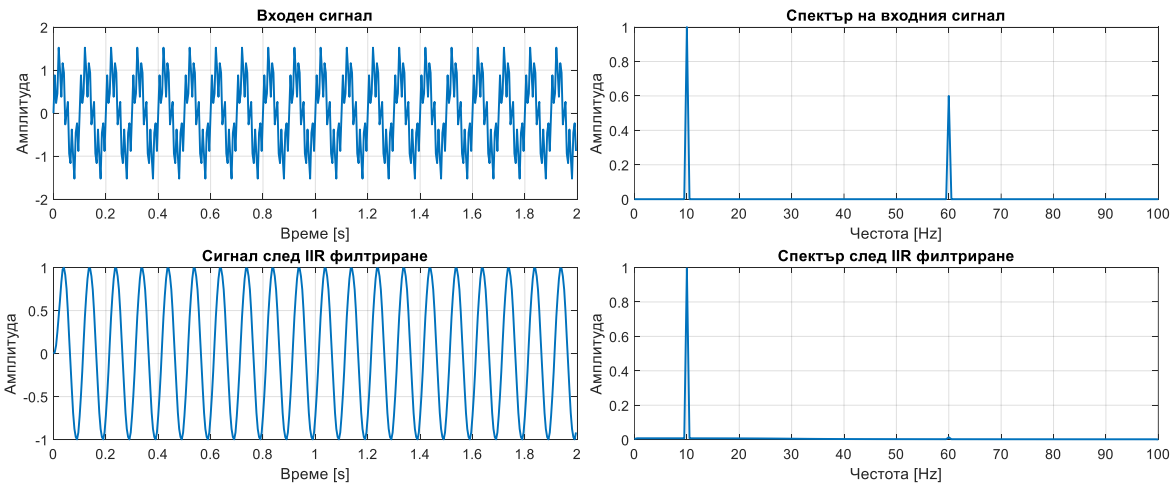
end

%% Честотна област
figure('Name','Задание 5 - Честотна област');

subplot(2,1,1);
plot(fX,X1,'LineWidth',1.2);
grid on;
xlabel('Честота [Hz]');
ylabel('Амплитуда');
title('Спектър на входния сигнал');
xlim([0 fs/2]);

subplot(2,1,2);
plot(fY,Y1,'LineWidth',1.2);
grid on;
xlabel('Честота [Hz]');
ylabel('Амплитуда');
title('Спектър след IIR филтриране');
xlim([0 fs/2]);
```

```
--- Резултати ---
```




6. Използвайте филтъра от Задание 5 и обработете същия сигнал веднъж с `filter()` и веднъж с `filtfilt()`. Представете резултатите върху обща графична зависимост и анализирайте фазовото отместване.

Код в MATLAB:

```
% Сравнение между filter() и filtfilt()

fs = 200;
T = 1;

t = 0:1/fs:T-1/fs;

%% Сигнал
x = sin(2*pi*10*t);

%% Butterworth филтър
N = 4;
fc = 30;

Wn = fc/(fs/2);

[b,a] = butter(N,Wn,'low');

%% Филтриране
y1 = filter(b,a,x);
y2 = filtfilt(b,a,x);

%% Визуализация
figure('Name','filter() и filtfilt()');

plot(t,x,'LineWidth',1.2);
hold on;

plot(t,y1,'LineWidth',1.2);
plot(t,y2,'LineWidth',1.2);

grid on;

xlabel('Време [s]');
```

```
ylabel('Амплитуда');

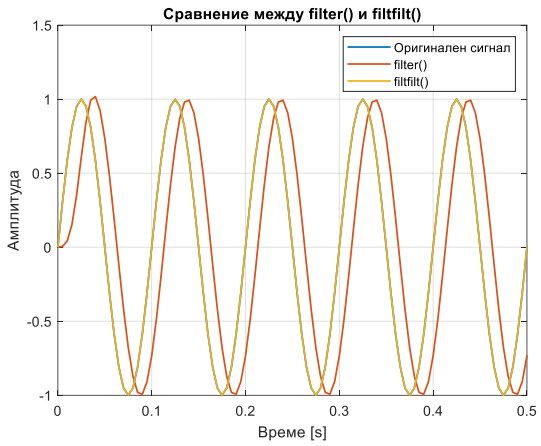
title('Сравнение между filter() и filtfilt()');

legend('Оригинален сигнал','filter()','filtfilt()');

xlim([0 0.5]);

hold off;
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Променете реда N и определете как се променя стръмността и груповото закъснение.
2. Проектирайте highpass вместо lowpass филтър и направете сравнителна оценка.
3. Използвайте `fvtool(b,a)` за визуално сравнение на резултатите от двата филтъра.
4. Проверете устойчивостта на филтрите чрез `roots(a)` и условието $|\pi| < 1$.

## VI Контролни въпроси

1. Какво означава и как се определя устойчивостта на IIR филтър?
2. Как рипълът Rp и затихването Rs влияят на реда на филтъра?
3. Кой тип IIR апроксимация дава най-стръмна граница при дадени Rp, Rs?

**Упражнение №11**

# Въведение в цифровата обработка на изображения. Представяне и основни операции с изображения в MATLAB

**I Цел на упражнението** – Запознаване с основните принципи за представяне на цифровите изображения и с възможностите на MATLAB и Image Processing Toolbox за тяхното зареждане, визуализиране и обработване. Изучаване на структурата на цифровото изображение, пространствената и яркостната дискретизация, основните типове изображения и начините за тяхното представяне като матрици. Изследванията включват работа с grayscale, RGB и бинарни изображения, анализ на отделните цветови канали, преобразуване между различни типове изображения, изрязване и промяна на пространствената разделителна способност.

## II Теоретична част

*2.1. Цифрово изображение*

Цифровото изображение може да се разглежда като двумерна дискретна функция

$$f(x, y),$$

където x и y определят пространственото положение на даден елемент от изображението, а стойността на функцията $f(x, y)$ характеризира неговата яркост или цвят.

Най-малкият елемент на цифровото изображение се нарича пиксел (pixel – picture element). Всеки пиксел има определени координати и числова стойност, която носи информация за неговата яркост или цветови компоненти.

Цифровото изображение може да бъде представено като матрица:

$$I = \begin{bmatrix} I(1,1) & I(1,2) & \cdots & I(1,N) \\ I(2,1) & I(2,2) & \dots & I(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ I(M,1) & I(M,2) & \cdots & I(M,N) \end{bmatrix}$$

където M е броят на редовете, а N – броят на колоните.

Следователно изображение с размери 1920×1080 съдържа

$$1920 \cdot 1080 = 2\,073\,600$$

пиксела.

Това матрично представяне е в основата на цифровата обработка на изображения в MATLAB – операциите върху изображението практически представляват операции върху неговите матрични елементи.

*2.2. Пространствена дискретизация*

При преобразуването на непрекъснато изображение в цифров вид пространствените координати трябва да бъдат дискретизирани. В резултат изображението се представя чрез краен брой пиксели.

Броят на пикселите определя *пространствената разделителна способност* на изображението. При еднакъв физически размер по-големият брой пиксели позволява представяне на по-фини детайли.

Например изображение с размери 256×256 съдържа значително по-малко пространствена информация от изображение 2048×2048. При намаляване на пространствената разделителна способност част от фините детайли могат да бъдат загубени.

*2.3. Квантуване на яркостта*

Освен пространствените координати трябва да бъдат дискретизирани и стойностите на яркостта. Този процес се нарича *квантуване*.

Ако за представяне на един пиксел се използват $b$ бита, броят на възможните нива е $L = 2^b$. При стандартно 8-битово изображение:

$$L = 2^8 = 256.$$

Следователно пикселът може да приема стойности от 0 до 255. При grayscale изображение обикновено: 0 → черно, 255 → бяло, а междинните стойности представят различни нива на сивото.

*2.4. Основни типове цифрови изображения*

- Grayscale изображения

Grayscale изображението съдържа информация само за интензитета на пикселите. В MATLAB то се представя чрез двумерна матрица $\boldsymbol{M \times N}$. При 8-битово изображение всеки елемент обикновено приема стойност между 0 и 255.

- RGB изображения

Цветните изображения най-често се представят чрез RGB модела, при който цветът на всеки пиксел се определя от три компоненти: R−Red, G−Green, B−Blue.

Следователно RGB изображението в MATLAB представлява тримерен масив $\boldsymbol{M \times N \times 3}$. Трите двумерни матрици могат да се разглеждат като отделни цветови канали:

$$R = I(:,:,1),$$
$$G = I(:,:,2),$$
$$B = I(:,:,3).$$

Комбинацията на различни стойности на трите компоненти позволява представянето на различни цветове.

Например:

$$(255,0,0) \rightarrow \text{червено},$$
$$(0,255,0) \rightarrow \text{зелено},$$
$$(0,0,255) \rightarrow \text{синьо},$$
$$(255,255,255) \rightarrow \text{бяло}.$$

*2.5. Бинарни изображения*

Бинарното изображение съдържа само две възможни стойности – 0 и 1. Обикновено 0 представя фон, а 1 – обект, въпреки че интерпретацията може да бъде обратна.

В MATLAB бинарните изображения обикновено са от тип `logical`. Те имат особено важно значение при сегментацията, морфологичната обработка и анализа на обекти, които ще бъдат разгледани в следващите упражнения.

*2.6. Типове данни при цифровите изображения*

MATLAB позволява изображенията да бъдат представяни чрез различни типове данни. Сред най-често използваните са `uint8, uint16, double` и `logical`.

При `uint8` стойностите са цели числа:

$$0 \leq I(x,y) \leq 255.$$

При преобразуване чрез `im2double()` стойностите обикновено се представят като числа с плаваща запетая в диапазона

$$0 \leq I(x, y) \leq 1.$$

Типът данни е важен, тъй като оказва влияние върху начина, по който MATLAB интерпретира стойностите на пикселите при визуализация и обработка.

*2.7. Преобразуване на RGB изображение в grayscale*

Цветното RGB изображение може да бъде преобразувано в grayscale. Това не се извършва чрез просто осредняване на трите цветови канала. Използва се претеглена комбинация, която отчита различната чувствителност на човешкото зрение към отделните цветове, която има следният общ вид:

$$Y = w_R R + w_G G + w_B B,$$

където $w_R$, $w_G$ и $w_B$ са тегловни коефициенти.

В MATLAB преобразуването се реализира чрез функцията `rgb2gray()`.

## III Подготовка за упражнението

Използвани функции в MATLAB:

- **`imread()`** – зарежда цифрово изображение от файл в работната среда на MATLAB.
- **`imshow()`** – визуализира изображение.
- **`imwrite()`** – записва изображение във файл.
- **`size()`** – определя размерите на матрицата или изображението.
- **`class()`** – определя типа данни на променливата, например `uint8`, `double` или `logical`.
- **`rgb2gray()`** – преобразува RGB цветно изображение в grayscale изображение.
- **`im2double()`** – преобразува изображение в тип `double` с подходящо мащабиране на стойностите.
- **`imresize()`** – променя пространствените размери на изображение.
- **`imcrop()`** – позволява изрязване на избрана област от изображение.
- **`subplot()`** – разделя графичния прозорец на няколко области за едновременно визуализиране на изображения.
- **`title()`** – добавя заглавие към визуализираното изображение.

## IV Задачи за изпълнение

1. Да се зареди цветно цифрово изображение и да се визуализира в MATLAB. Да се определят размерите на изображението, броят на цветовите канали и типът на използваните данни. Да се определи общият брой пиксели.

Код в MATLAB:

```
% Зареждане и основен анализ на изображение

%% Зареждане на изображението
I = imread('peppers.png');

%% Визуализация
figure;
imshow(I);
title('Оригинално RGB изображение');

%% Размери
[M,N,C] = size(I);
```

```
%% Тип данни
data_type = class(I);

%% Брой пиксели
pixels = M*N;

%% Извеждане на информация
disp('--- Информация за изображението ---');

disp(['Брой редове: ', num2str(M)]);
disp(['Брой колони: ', num2str(N)]);
disp(['Брой цветови канали: ', num2str(C)]);
disp(['Тип данни: ', data_type]);
disp(['Общ брой пиксели: ', num2str(pixels)]);
```

```
--- Резултати ---
--- Информация за изображението ---
Брой редове: 384
Брой колони: 512
Брой цветови канали: 3
Тип данни: uint8
Общ брой пиксели: 196608
```




2. Да се зареди RGB изображение и да се отделят неговите червен, зелен и син цветови канал. Оригиналното изображение и трите канала да се визуализират в един графичен прозорец. Да се сравни разпределението на интензитета в отделните канали.

Код в MATLAB:

```
% Анализ на RGB каналите

clc;
clear;
close all;

I = imread('peppers.png');

%% Разделяне на каналите
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);

%% Визуализация
figure;
```

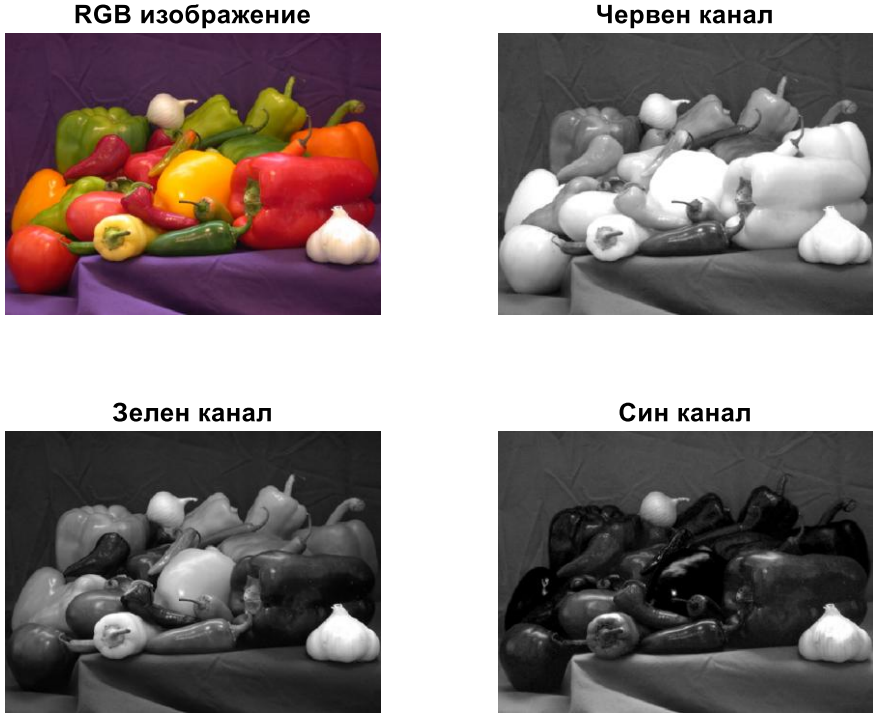
```
subplot(2,2,1);
imshow(I);
title('RGB изображение');

subplot(2,2,2);
imshow(R);
title('Червен канал');

subplot(2,2,3);
imshow(G);
title('Зелен канал');

subplot(2,2,4);
imshow(B);
title('Син канал');
```

```
--- Резултати ---
```




3. Да се преобразува цветното изображение в grayscale чрез функцията rgb2gray(). Да се сравнят оригиналното и полученото изображение и да се определят техните размери.

Код в MATLAB:

```
% RGB към grayscale
clc;
clear;
close all;

I = imread('peppers.png');

%% Преобразуване
Igray = rgb2gray(I);
```

```
%% Визуализация
figure;

subplot(1,2,1);
imshow(I);
title('RGB изображение');

subplot(1,2,2);
imshow(Igray);
title('Grayscale изображение');

%% Размери
disp('Размер на RGB изображението:');
disp(size(I));

disp('Размер на grayscale изображението:');
disp(size(Igray));

disp('Тип на grayscale изображението:');
disp(class(Igray));
```

```
--- Резултати ---
Размер на RGB изображението:   384   512     3

Размер на grayscale изображението:   384   512

Тип на grayscale изображението: uint8
```




4. Да се преобразува grayscale изображението от uint8 към double. Да се сравнят диапазоните на стойностите преди и след преобразуването.

Код в MATLAB:

```
% Преобразуване на типа данни

clc;
clear;
close all;

I = imread('peppers.png');
Igray = rgb2gray(I);

%% Преобразуване към double
Idouble = im2double(Igray);

%% Информация
```

```
disp('Преди преобразуването:');
disp(class(Igray));
disp(['Min = ',num2str(min(Igray(:)))]);
disp(['Max = ',num2str(max(Igray(:)))]);

disp(' ');

disp('След преобразуването:');
disp(class(Idouble));
disp(['Min = ',num2str(min(Idouble(:)))]);
disp(['Max = ',num2str(max(Idouble(:)))]);

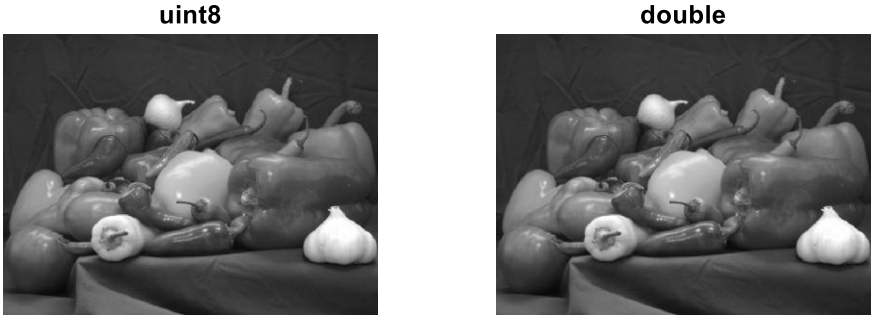
%% Визуализация
figure;

subplot(1,2,1);
imshow(Igray);
title('uint8');

subplot(1,2,2);
imshow(Idouble);
title('double');
```

```
--- Резултати ---
Преди преобразуването:
uint8
Min = 8
Max = 255

След преобразуването:
double
Min = 0.031373
Max = 1
```




5. Да се намалят размерите на изображението последователно до 50% и 25% от оригиналните. Получените изображения да се сравнят визуално с оригинала и да се определят новите им размери.

Код в MATLAB:

```
% Промяна на размерите

clc;
clear;
close all;

I = imread('peppers.png');

%% Промяна на размера
```

```
I50 = imresize(I,0.5);
I25 = imresize(I,0.25);

%% Визуализация
figure;

subplot(1,3,1);
imshow(I);
title('Оригинал');

subplot(1,3,2);
imshow(I50);
title('50%');

subplot(1,3,3);
imshow(I25);
title('25%');

%% Размери
disp('Оригинален размер:');
disp(size(I));

disp('Размер при 50%:');
disp(size(I50));

disp('Размер при 25%:');
disp(size(I25));
```

```
--- Резултати ---
Оригинален размер:
   384   512     3
Размер при 50%:
   192   256     3
Размер при 25%:
    96   128     3
```




6. Да се избере чрез координати правоъгълна област от изображението и да се създаде ново изображение, съдържащо само избраната област. Да се визуализират оригиналното и изрязаното изображение.

Код в MATLAB:

```
% Изрязване на област от изображение

clc;
clear;
close all;

I = imread('peppers.png');
```

```
%% Правоъгълна област
% [x y width height]
rect = [100 50 250 200];

%% Изрязване
Icrop = imcrop(I,rect);

%% Визуализация
figure;

subplot(1,2,1);
imshow(I);
title('Оригинално изображение');

subplot(1,2,2);
imshow(Icrop);
title('Изрязана област');
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Заредете избрано от Вас цветно изображение във формат .jpg, .png или .bmp. Определете неговите размери, типа на данните, броя на цветовите канали и общия брой пиксели. Сравнете получените резултати с тези за изображението peppers.png.
2. Изберете RGB изображение и отделете трите му цветови канала. Определете минималната и максималната стойност на пикселите във всеки канал чрез функциите min() и max(). Определете кой цветови компонент е най-силно представен в избраното изображение.
3. Увеличете изображение с коефициенти 2 и 4. Сравнете увеличените изображения с оригинала. Установете дали увеличаването на броя на пикселите чрез imresize() води до появата на нова информация и нови детайли в изображението.

## VI Контролни въпроси

1. Какво представлява пространствената разделителна способност на цифровото изображение?
2. Как се изменя качеството и детайлността на изображението при намаляване на пространствената му разделителна способност?
3. Какво е предназначението на функциите `imread()`, `imshow()` и `imwrite()`?

**Упражнение №12**

# Трансформации на яркостта и подобряване на контраста. Хистограми

**I Цел на упражнението** – Запознаване с основните методи за точкова обработка на цифрови изображения и тяхното приложение за изменение на яркостта и подобряване на контраста. Разглеждат се връзката между входните и изходните стойности на пикселите, получаването на негативно изображение, линейното изменение на яркостта и контраста, както и използването на хистограмата за анализ на разпределението на нивата на сивото.

## II Теоретична част

*2.1. Точкови преобразувания на цифрови изображения*

Един от най-простите подходи за обработка на цифрови изображения е промяната на стойностите на отделните пиксели. При този тип обработка стойността на всеки пиксел в изходното изображение се определя единствено от стойността на съответния пиксел във входното изображение.

Общият вид на точковото преобразуване може да бъде представен като:

$$s = T(r)$$

където $r$ е стойността на пиксела във входното изображение, $s$ е стойността на съответния пиксел в изходното изображение, а $T$ е функцията на преобразуване.

За разлика от пространственото филтриране, което ще бъде разгледано в следващото упражнение, при точковите операции не се използват стойностите на съседните пиксели. Чрез подходящ избор на функцията $T$(r) могат да бъдат реализирани различни операции – изменение на яркостта, промяна на контраста, получаване на негатив и други преобразувания на интензитета.

*2.2. Яркост и контраст*

Яркостта характеризира общото възприятие за светлина в изображението и е свързана със стойностите на неговите пиксели. При 8-битово grayscale изображение стойностите обикновено са в интервала $0 \leq r \leq 255$, където 0 съответства на черно, а 255 – на бяло.

Една проста операция за изменение на яркостта може да се представи чрез:

$$s = r + \beta$$

където $\beta$ е константа. При положителна стойност на $\beta$ изображението се изсветлява, а при отрицателна – потъмнява.

При реализацията трябва да се отчита допустимият диапазон на пикселните стойности. Ако след операцията бъдат получени стойности извън него, те трябва да бъдат ограничени до допустимия интервал.

Контрастът характеризира разликата между тъмните и светлите области на изображението. При нисък контраст стойностите на пикселите са концентрирани в сравнително тесен диапазон. При висок контраст разликата между тъмните и светлите области е по-голяма.

Едно просто линейно преобразуване може да се запише като:

$$s = \alpha r + \beta,$$

където $\alpha$ управлява основно контраста, а $\beta$ – яркостта.

При $\alpha > 1$ - разликите между стойностите на пикселите се увеличават и контрастът нараства, докато при $0 < \alpha < 1$ - диапазонът на стойностите се свива и контрастът намалява.

*2.3. Негатив на цифрово изображение*

Негативното преобразуване инвертира нивата на яркост. Светлите области стават тъмни, а тъмните – светли. За изображение с L възможни нива на яркост преобразуването е:

$$s = (L - 1) - r$$

При 8-битово изображение $L = 256$ и следователно $s = 255 - r$. Така пиксел със стойност 0 се преобразува в 255, пиксел със стойност 255 – в 0, а междинните стойности се инвертират спрямо средата на динамичния диапазон.

Негативното преобразуване може да бъде полезно при визуализиране на детайли, които са трудно различими в тъмни области, както и при някои специализирани приложения за анализ на изображения.

*2.4. Хистограма на цифрово изображение*

Хистограмата представя разпределението на пикселите според техните стойности на яркост. За grayscale изображение с $L$ възможни нива хистограмата може да бъде описана чрез:

$$h(r_\mathrm{k}) = n_\mathrm{k}$$

където $r_k$ е k-тото ниво на яркост, а $n_k$ е броят на пикселите с тази стойност. При 8-битово grayscale изображение хистограмата обикновено съдържа 256 нива – от 0 до 255.

Хистограмата предоставя полезна информация за изображението. Ако стойностите са концентрирани предимно в лявата ѝ част, изображението съдържа много тъмни пиксели. Ако са концентрирани вдясно, преобладават светлите области.

При изображение с нисък контраст хистограмата обикновено заема ограничена част от възможния динамичен диапазон. При изображение с по-висок контраст стойностите са разпределени в по-широк диапазон.

Важно е да се отбележи, че хистограмата не съдържа информация за *пространственото разположение* на пикселите. Следователно две визуално различни изображения могат да имат еднакви или много близки хистограми.

*2.5. Нормализирана хистограма*

Хистограмата може да бъде нормализирана чрез разделяне на броя пиксели за всяко ниво на яркост на общия брой пиксели:

$$p(r_\mathrm{k}) = \frac{n_k}{M \cdot N},$$

където $M \cdot N$ е общият брой пиксели.

Получената величина може да се интерпретира като относителна честота или оценка на вероятността даден пиксел да има стойност $r_k$.

Сумата на всички нормализирани стойности е $\sum p(r_k) = 1$.

*2.6. Разтягане на контраста*

При изображения с нисък контраст стойностите на пикселите често използват само малка част от възможния динамичен диапазон. Например вместо целия интервал от 0 до 255 те могат да бъдат концентрирани между 80 и 170.

Разтягането на контраста преобразува този ограничен диапазон в по-широк. При линейно преобразуване от интервала $[r_{min}, r_{max}]$ към $[0, L-1]$ може да се използва:

$$s = \left[\frac{r - r_{min}}{r_{max} - r_{min}}\right] \cdot (L - 1)$$

В резултат най-тъмните стойности се приближават до черното, а най-светлите – до бялото. Това увеличава визуалната разлика между различните области.

В MATLAB подобно преобразуване може да бъде реализирано чрез функцията `imadjust()`.

*2.7. Хистограмна еквализация*

Хистограмната еквализация е метод за автоматично подобряване на контраста чрез преобразуване на стойностите на пикселите въз основа на тяхното статистическо разпределение.

За тази цел се използва кумулативното разпределение на нивата на яркост:

$$CDF(r_k) = \sum_{j=0}^{k} p(r_j) .$$

Преобразуването може да бъде представено приблизително като:

$$s_k = (L - 1) \cdot CDF(r_k),$$

като целта е стойностите на пикселите да бъдат разпределени по-ефективно върху наличния динамичен диапазон.

След еквализация хистограмата **не е** напълно равномерна. Обикновено при дискретни изображения това не е възможно. Методът по-скоро преразпределя нивата на яркост така, че динамичният диапазон да се използва по-пълноценно.

В MATLAB хистограмната еквализация се реализира чрез функцията `histeq()`.

## III Подготовка за упражнението

Използвани функции в MATLAB:

- **`imread()`** – зарежда изображение от файл.
- **`imshow()`** – визуализира цифрово изображение.
- **`rgb2gray()`** – преобразува RGB изображение в grayscale.
- **`imhist()`** – изчислява и визуализира хистограмата на grayscale изображение.
- **`imadjust()`** – преобразува диапазона на интензитетите и се използва за корекция на яркостта и контраста.
- **`stretchlim()`** – определя подходящи граници на интензитета за разтягане на контраста.
- **`histeq()`** – извършва хистограмна еквализация.
- **`im2double()`** – преобразува изображение към тип `double` с подходящо мащабиране.
- **`min()`** – определя минималната стойност.
- **`max()`** – определя максималната стойност.
- **`subplot()`** – разделя графичния прозорец на области.
- **`plot()`** – визуализира числови зависимости.
- **`title()`**, **`xlabel()`**, **`ylabel()`** и **`grid on`** – използват се за оформяне на графиките.

## IV Задачи за изпълнение

1. Да се зареди цветно изображение и да се преобразува в grayscale. Да се визуализират полученото изображение и неговата хистограма. Определете минималната и максималната стойност на яркостта. По вида на хистограмата анализирайте дали изображението използва пълния динамичен диапазон и дали преобладават тъмни или светли области.

Код в MATLAB:

```
% Анализ на изображение чрез хистограма
clc;
clear;
close all;

%% Зареждане на изображението
I = imread('pout.tif');

%% Минимална и максимална стойност
Imin = min(I(:));
Imax = max(I(:));

%% Визуализация
figure('Name','Задание 1');

subplot(1,2,1);
imshow(I);
title('Оригинално изображение');

subplot(1,2,2);
imhist(I);
title('Хистограма на изображението');
xlabel('Ниво на яркост');
ylabel('Брой пиксели');

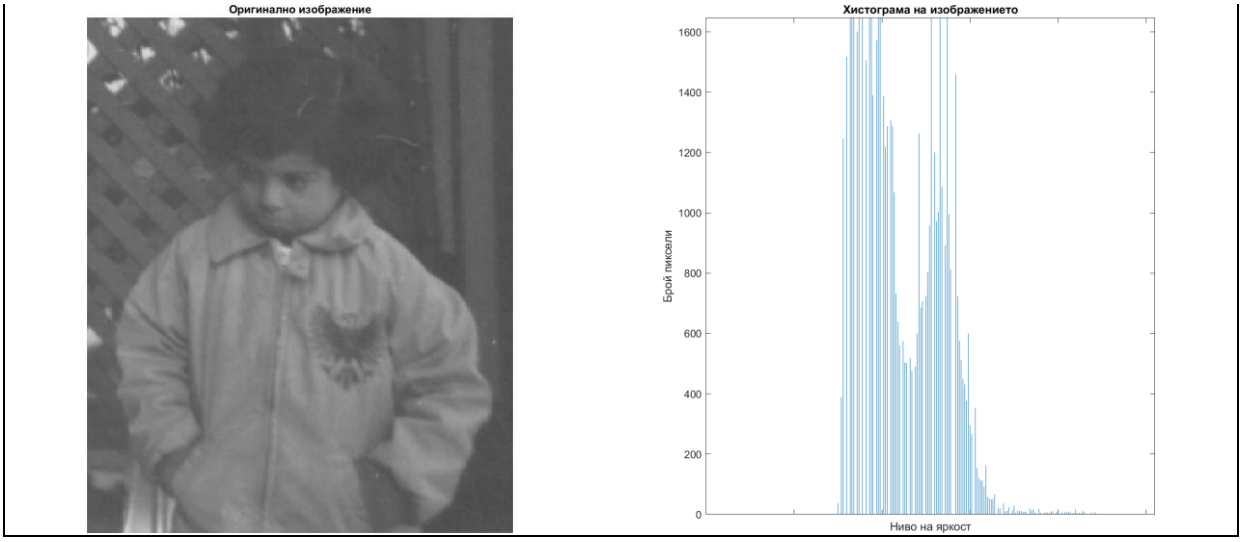
%% Извеждане на резултатите
disp('--- Анализ на изображението ---');

disp(['Минимална стойност: ', num2str(Imin)]);
disp(['Максимална стойност: ', num2str(Imax)]);
disp(['Тип данни: ', class(I)]);

[M,N] = size(I);

disp(['Размер на изображението: ', num2str(M), ' x ', num2str(N)]);
```

```
--- Резултати ---
Минимална стойност: 74
Максимална стойност: 224
Тип данни: uint8
Размер на изображението: 291 x 240
```




2. Да се изследва влиянието на добавянето и изваждането на константа върху яркостта на grayscale изображение. Изображението да се преобразува към double. Да се създадат две нови изображения – едното с увеличена, а другото с намалена яркост. Да се осигури ограничаване на резултатите в допустимия диапазон [0,1]. Сравнете визуално трите изображения и техните хистограми.

Код в MATLAB:

```
% Изменение на яркостта
clc;
clear;
close all;

I = imread('peppers.png');
Igray = rgb2gray(I);

%% Преобразуване към double
Id = im2double(Igray);

%% Изменение на яркостта
beta = 0.20;

Ilight = Id + beta;
Idark  = Id - beta;

%% Ограничаване в интервала [0,1]
Ilight(Ilight > 1) = 1;
Ilight(Ilight < 0) = 0;

Idark(Idark > 1) = 1;
Idark(Idark < 0) = 0;

%% Изображения
figure('Name','Задание 2 - Яркост');

subplot(1,3,1);
imshow(Id);
title('Оригинал');

subplot(1,3,2);
```

```
imshow(Ilight);
title('Увеличена яркост');

subplot(1,3,3);
imshow(Idark);
title('Намалена яркост');

%% Хистограми
figure('Name','Задание 2 - Хистограми');

subplot(3,1,1);
imhist(Id);
title('Оригинал');

subplot(3,1,2);
imhist(Ilight);
title('Увеличена яркост');

subplot(3,1,3);
imhist(Idark);
title('Намалена яркост');
```

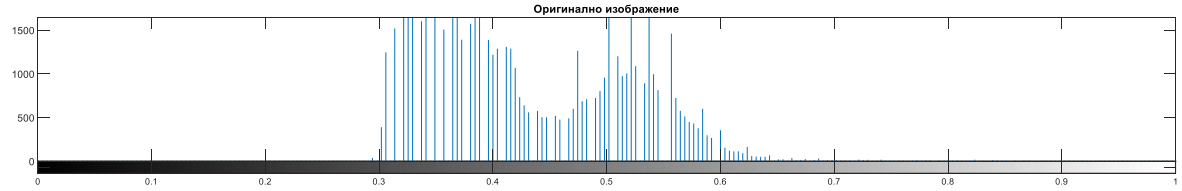
```
--- Резултати ---
```

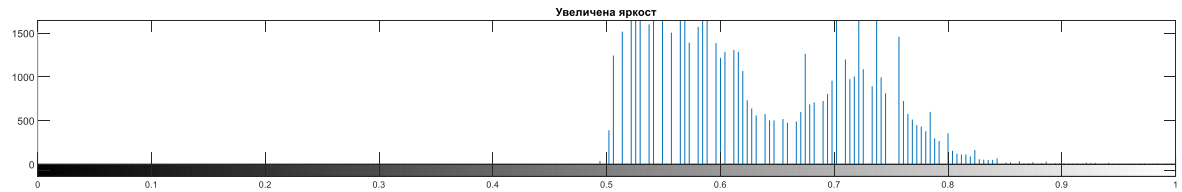


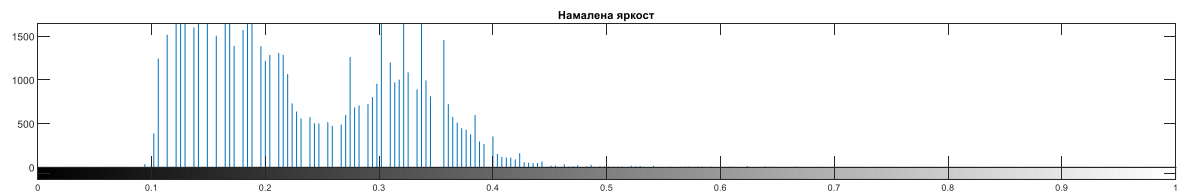






3. Да се създаде негатив на 8-битово grayscale изображение чрез директно прилагане на зависимостта $s = 255 - r$. Да се визуализират оригиналното и негативното изображение и съответните им хистограми. Анализирайте как се изменя хистограмата при негативното преобразуване.

Код в MATLAB:

```
% Получаване на негатив на 8-битово grayscale изображение
clc;
clear;
close all;

%% Зареждане
I = imread('pout.tif');

%% Получаване на негатив
Inegative = 255 - I;

%% Визуализация
figure('Name','Задание 3 - Негатив');

subplot(1,2,1);
imshow(I);
title('Оригинално изображение');

subplot(1,2,2);
imshow(Inegative);
title('Негативно изображение');

%% Хистограми
figure('Name','Задание 3 - Хистограми');

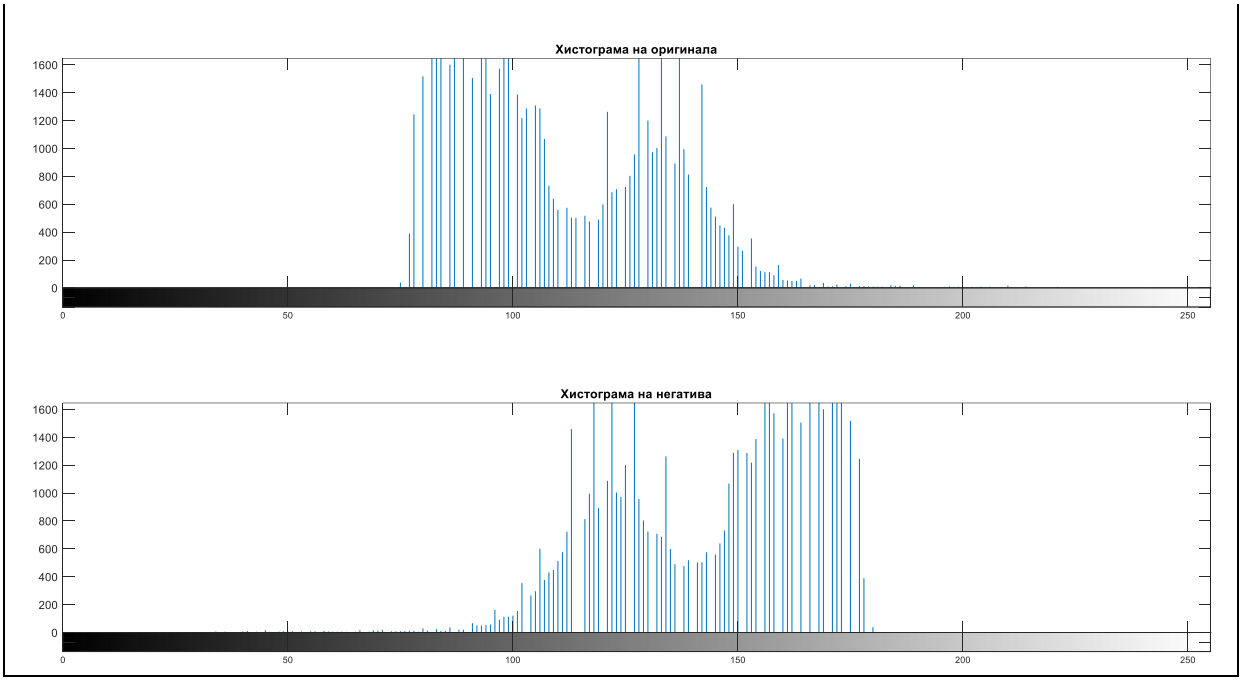
subplot(2,1,1);
imhist(I);
title('Хистограма на оригинала');

subplot(2,1,2);
imhist(Inegative);
title('Хистограма на негатива');
```

```
--- Резултати ---
```






4. Да се изследва подобряването на изображение с ограничен динамичен диапазон чрез разтягане на контраста. Да се създаде изображение с понижен контраст, чрез stretchlim() да се определят подходящи граници и с функцията imadjust() да се извърши разтягане на контраста. Да се сравнят изображенията и техните хистограми преди и след обработката.

Код в MATLAB:

```
% Разтягане на контраста
clc;
clear;
close all;

I = imread('peppers.png');
Igray = rgb2gray(I);

%% Създаване на изображение с понижен контраст
Id = im2double(Igray);

Ilow = 0.4*Id + 0.3;

%% Определяне на граници за разтягане
limits = stretchlim(Ilow);

%% Разтягане на контраста
Istretch = imadjust(Ilow,limits,[0 1]);

%% Визуализация
figure('Name','Задание 4 - Контраст');

subplot(1,3,1);
imshow(Id);
title('Оригинал');

subplot(1,3,2);
```

```
imshow(Ilow);
title('Понижен контраст');

subplot(1,3,3);
imshow(Istretch);
title('Разтегнат контраст');

%% Хистограми
figure('Name','Задание 4 - Хистограми');

subplot(3,1,1);
imhist(Id);
title('Оригинал');

subplot(3,1,2);
imhist(Ilow);
title('Понижен контраст');

subplot(3,1,3);
imhist(Istretch);
title('След разтягане');
```

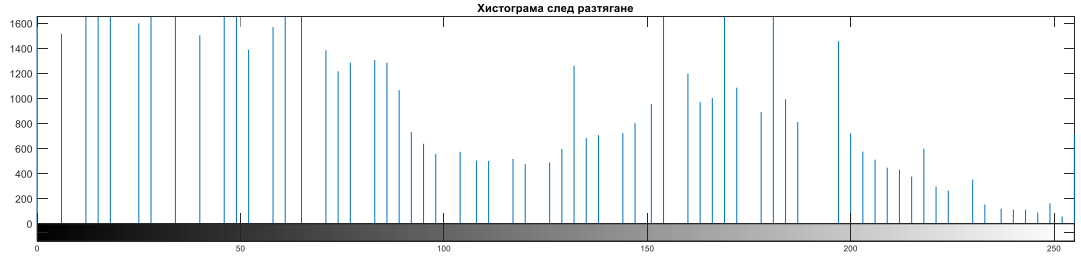
```
--- Резултати ---
```









5. Да се приложи хистограмна еквализация върху grayscale изображение чрез функцията histeq(). Да се представят оригиналното и еквализираното изображение заедно с техните хистограми. Анализирайте как еквализацията променя разпределението на нивата на яркост и визуалния контраст.

Код в MATLAB:

```
% Хистограмна еквализация
clc;
clear;
close all;

%% Зареждане
I = imread('pout.tif');

%% Хистограмна еквализация
Ieq = histeq(I);

%% Визуализация на изображенията
figure('Name','Задание 5 - Еквализация');

subplot(1,2,1);
imshow(I);
title('Оригинално изображение');

subplot(1,2,2);
imshow(Ieq);
title('След хистограмна еквализация');

%% Хистограми
figure('Name','Задание 5 - Хистограми');

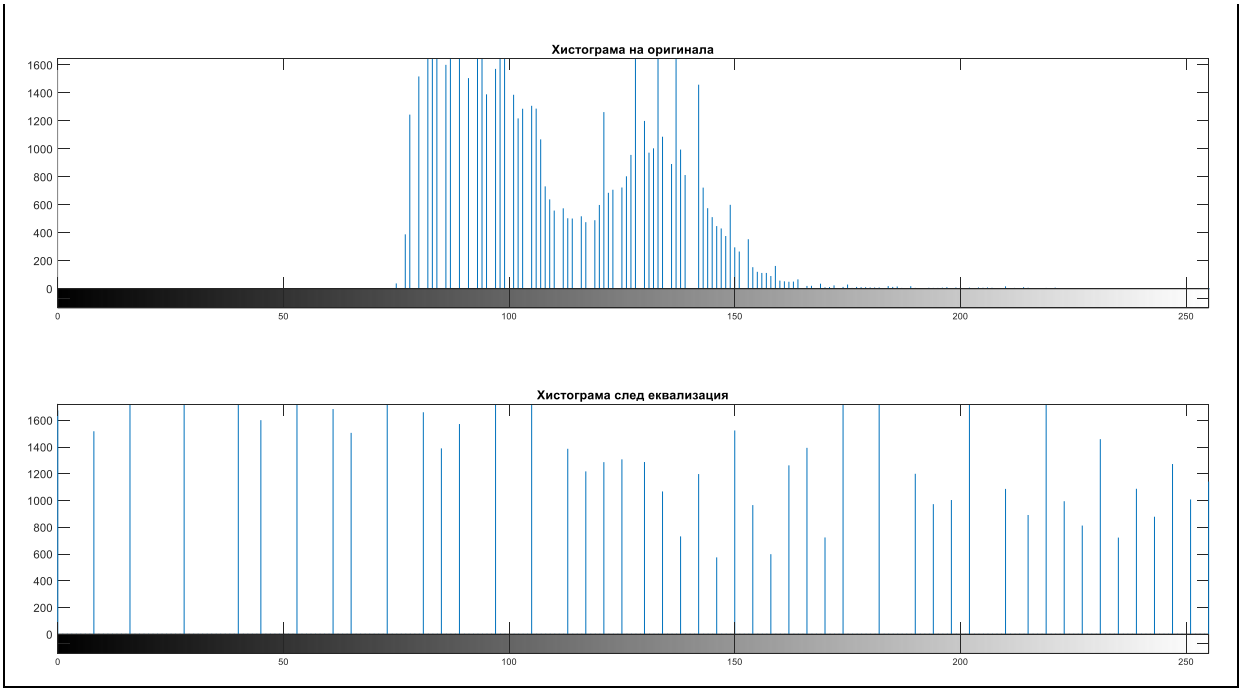
subplot(2,1,1);
imhist(I);
title('Хистограма на оригинала');

subplot(2,1,2);
imhist(Ieq);
title('Хистограма след еквализация');
```

```
--- Резултати ---
```






6. Да се сравнят два метода за подобряване на контраста – разтягане чрез imadjust() и хистограмна еквализация чрез histeq(). Да се използва едно и също входно изображение с понижен контраст. Получените чрез двата метода изображения да се представят едновременно и да се сравнят техните хистограми. Определете кой от методите води до по-съществено преразпределение на нивата на яркост.

Код в MATLAB:

```
% Сравнение на методи за подобряване на контраста
clc;
clear;
close all;

%% Зареждане
I = imread('pout.tif');

%% Метод 1: Разтягане на контраста
limits = stretchlim(I);

Iadjust = imadjust(I, limits, []);

%% Метод 2: Хистограмна еквализация
Ieq = histeq(I);

%% Сравнение на изображенията
figure('Name','Задание 6 - Сравнение');

subplot(1,3,1);
imshow(I);
title('Оригинал');

subplot(1,3,2);
imshow(Iadjust);
title('Разтягане на контраста');
```

```
subplot(1,3,3);
imshow(Ieq);
title('Хистограмна еквализация');

%% Сравнение на хистограмите
figure('Name','Задание 6 - Хистограми');

subplot(3,1,1);
imhist(I);
title('Оригинално изображение');

subplot(3,1,2);
imhist(Iadjust);
title('След imadjust()');

subplot(3,1,3);
imhist(Ieq);
title('След histeq()');
```

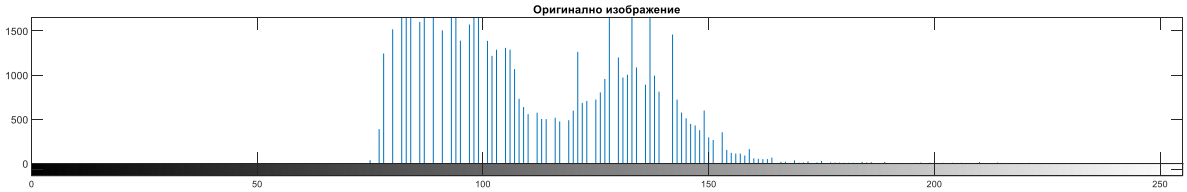
```
--- Резултати ---
```

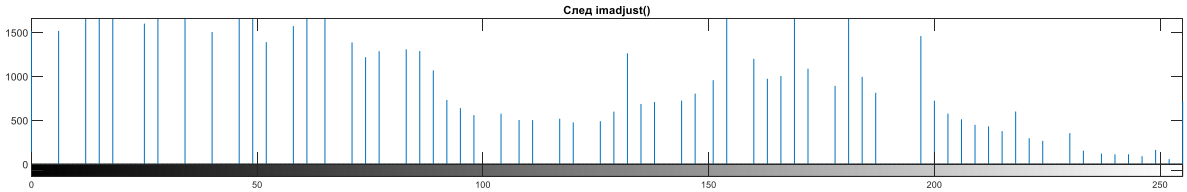


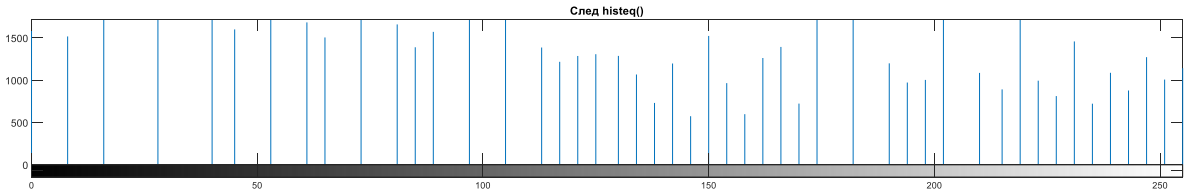






## V Допълнителни задачи

1. Променете стойността на параметъра beta в Задача 2 последователно на 0.1, 0.3 и 0.5. Изследвайте как увеличаването му влияе върху изображението и неговата хистограма.
2. Реализирайте негативното преобразуване върху RGB изображение, без предварително преобразуване в grayscale. Сравнете цветното негативно изображение с оригинала.
3. Създайте изкуствено прекалено светло изображение и изследвайте възможностите за подобряване на контраста му.

## VI Контролни въпроси

1. Какво представлява хистограмата на цифрово изображение и каква информация за яркостта и контраста може да бъде получена от нея?
2. Каква е разликата между изменение на яркостта и изменение на контраста на цифрово изображение и как тези преобразувания влияят върху хистограмата?
3. Каква е разликата между разтягането на контраста и хистограмната еквализация и в какви случаи се използват двата метода?

**Упражнение №13**

# Пространствено филтриране на цифрови изображения. Потискане на шум

**I Цел на упражнението** – Запознаване с основните методи за пространствена обработка на цифрови изображения чрез двумерни филтри. Изучаване на принципа на двумерната дискретна конволюция, използването на филтриращи маски (ядра) и влиянието им върху изображението. Разглеждане на основните изглаждащи филтри за потискане на шум, включително усредняващ, Гаусов и медианен филтър.

## II Теоретична част

*2.1. Пространствено филтриране на цифрови изображения*

При точковите преобразувания, разгледани в предходното упражнение, стойността на всеки изходен пиксел зависи единствено от стойността на съответния входен пиксел. При пространственото филтриране, за разлика от тях, новата стойност на даден пиксел се определя въз основа на стойностите на пикселите в определена негова околност.

Обработката се извършва чрез малка двумерна матрица, наречена филтрираща маска, ядро или *kernel*. Маската последователно се премества върху изображението и за всяко нейно положение се изчислява нова стойност на централния пиксел.

Ако входното изображение е $f(x,y)$, филтриращата маска с $h(m,n)$, а изходното изображение с $g(x,y)$, линейното пространствено филтриране чрез двумерна конволюция може да се представи като:

$$g(x,y) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} h(m,n) f(x-m, y-n),$$

където размерът на филтъра е $(2a+1) \times (2b+1)$.

Тази зависимост е двумерен аналог на дискретната конволюция, използвана при обработката на едномерни сигнали.

*2.2. Филтрираща маска*

Филтриращата маска е матрица с относително малки размери, например 3×3, 5×5 или 7×7. Всеки неин елемент представлява коефициент, определящ влиянието на съответния съседен пиксел върху изходната стойност.

Пример за обща маска с размер 3×3 е:

$$h = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

При линейното филтриране стойностите на пикселите, покрити от маската, се умножават по съответните коефициенти, след което получените произведения се сумират. Изборът на стойностите на коефициентите определя вида на филтъра и неговото действие върху изображението.

*2.3. Изглаждащи пространствени филтри*

Изглаждащите филтри се използват основно за потискане на високочестотни изменения, сред които шумът и фините детайли. В честотен смисъл те имат поведение, подобно на нискочестотните филтри при едномерните сигнали.

Основният недостатък на изглаждането е, че заедно с шума могат да бъдат потиснати и полезни детайли, включително контури и малки обекти.

Степента на изглаждане зависи от вида на филтъра, размера на маската и стойностите на нейните коефициенти.

*2.4. Усредняващ филтър*

Усредняващият филтър заменя стойността на всеки пиксел със средната аритметична стойност на пикселите в неговата околност. За маска с размер 3×3 филтърът може да бъде представен като:

$$h = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}.$$

Ако стойностите в разглежданата околност са $r_1, r_2, \dots, r_9$, новата стойност е:

$$s = \frac{1}{9}\sum_{k=1}^{9} r_k$$

Усредняващият филтър е лесен за реализация и може да намали случаен шум, но води до размазване на контурите. При увеличаване на размера на маската ефектът на изглаждане става по-силен, но загубата на детайли също нараства.

*2.5. Гаусов филтър*

Гаусовият филтър също извършва локално усредняване, но различните пиксели в околността участват с различни тегла. Най-голяма тежест има централният пиксел, а влиянието на останалите намалява с отдалечаването им от центъра.

Двумерната Гаусова функция се определя като:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}}$$

където $\sigma$ определя ширината на Гаусовото разпределение и съответно степента на изглаждане. По-голяма стойност на $\sigma$ води до по-силно изглаждане на изображението.

В сравнение с обикновения усредняващ филтър Гаусовият филтър осигурява по-плавна обработка и често по-добре запазва структурата на изображението.

*2.6. Медианен филтър*

Медианният филтър е нелинеен пространствен филтър. При него стойността на централния пиксел не се получава чрез линейна комбинация на съседите, а чрез определяне на медианата на стойностите в избраната околност.

Например при следната околност:

$$\begin{bmatrix}15 & 16 & 18\\ 17 & 255 & 19\\ 16 & 18 & 17\end{bmatrix}$$

след подреждане на стойностите се получава:

$$15,16,16,17,17,18,18,19,255$$

Медианната стойност е 17. Следователно централната стойност 255, която може да представлява импулсен шум, се заменя с 17.

Медианният филтър е особено ефективен при потискане на импулсен шум, като обикновено запазва контурите по-добре от линейните изглаждащи филтри.

*2.7. Шум в цифровите изображения*

Шумът представлява нежелано случайно изменение на стойностите на пикселите. Той може да възникне при регистриране, цифровизация, предаване или съхранение на изображението.

В практиката се срещат различни модели на шум. В настоящото упражнение ще бъдат разгледани два от най-разпространените – гаусов шум и импулсен шум от тип „salt-and-pepper“.

- Гаусов шум

При гаусовия шум към стойностите на пикселите се добавя случайна величина с нормално разпределение. Моделът може да бъде представен като:

$$g(x,y) = f(x,y) + \eta(x,y)$$

където $\eta(x,y)$ е случайна величина с Гаусово разпределение. Този шум води до множество малки случайни изменения в цялото изображение.

- Salt-and-Pepper шум

При този тип шум определена част от пикселите се заменят със стойности, близки до минималната или максималната допустима яркост. В изображението се появяват случайни черни и бели точки.

Salt-and-pepper шумът е типичен пример за импулсен шум и медианният филтър е особено подходящ за неговото потискане.

*2.8. Влияние на размера на маската*

Размерът на пространствения филтър има съществено влияние върху обработката. При по-малка маска, например 3×3, се извършва умерено изглаждане и се запазват повече детайли.

При увеличаване на маската до 5×5, 7×7 или повече в изчислението участват повече съседни пиксели. Това усилва потискането на шума, но едновременно с това води до по-силно размазване на контурите и загуба на фини структури.

Следователно изборът на размера на маската представлява компромис между **потискане на шума** и **запазване на детайлите**.

## III Подготовка за упражнението

Използвани функции в MATLAB:

- **`imread()`** – зарежда цифрово изображение от файл.
- **`imshow()`** – визуализира изображение.
- **`rgb2gray()`** – преобразува RGB изображение в grayscale.
- **`im2double()`** – преобразува изображението към тип `double` с подходящо нормализиране на стойностите.
- **`imnoise()`** – добавя избран модел на шум към изображение, например Gaussian или salt-and-pepper.
- **`fspecial()`** – създава стандартна двумерна филтрираща маска, например усредняваща или Гаусова.
- **`imfilter()`** – извършва пространствено филтриране на изображение чрез зададена филтрираща маска.
- **`conv2()`** – извършва двумерна дискретна конволюция.
- **`medfilt2()`** – извършва двумерно медианно филтриране.
- **`subplot()`** – разделя графичния прозорец на отделни области.
- **`title()`** – добавя заглавие към визуализираното изображение.
- **`mesh()`** – визуализира стойностите на двумерна маска като тримерна повърхнина.

**IV Задачи за изпълнение**

1. Да се зареди grayscale изображение и да се създаде ръчно усредняваща филтрираща маска с размер 3×3. Да се приложи двумерна конволюция чрез функцията conv2(). Резултатът да се сравни с филтриране чрез imfilter(). Да се анализира влиянието на усредняващия филтър върху контурите и финните детайли.

Код в MATLAB:

```
% Усредняващо пространствено филтриране
clc;
clear;
close all;

%% Зареждане на изображение
I = imread('cameraman.tif');

%% Преобразуване към double
Id = im2double(I);

%% Усредняваща маска 3x3
h = (1/9)*ones(3,3);

%% Филтриране чрез conv2
Iconv = conv2(Id,h,'same');

%% Филтриране чрез imfilter
Ifilter = imfilter(Id,h,'replicate');

%% Визуализация
figure('Name','Задание 1');

subplot(1,3,1);
imshow(Id);
title('Оригинал');

subplot(1,3,2);
imshow(Iconv);
title('Филтриране чрез conv2');

subplot(1,3,3);
imshow(Ifilter);
title('Филтриране чрез imfilter');
```

```
--- Резултати ---
```



2. Да се добави гаусов шум към grayscale изображение чрез imnoise(). Да се използва средна стойност на шума 0 и дисперсия 0.01. След това изображението да се обработи с усредняващ филтър с размер 3×3. Да се сравнят оригиналното, зашуменото и филтрираното изображение.

Код в MATLAB:

```
% Потискане на Gaussian шум чрез усредняващ филтър
clc;
clear;
close all;

%% Зареждане
I = imread('cameraman.tif');

%% Добавяне на Gaussian шум
Inoise = imnoise(I,'gaussian',0,0.01);

%% Усредняващ филтър 3x3
h = fspecial('average',[3 3]);

%% Филтриране
Ifilt = imfilter(Inoise,h,'replicate');

%% Визуализация
figure('Name','Задание 2');

subplot(1,3,1);
imshow(I);
title('Оригинал');

subplot(1,3,2);
imshow(Inoise);
title('Gaussian шум');

subplot(1,3,3);
imshow(Ifilt);
title('След усредняващ филтър');
```

```
--- Резултати ---
```



3. Да се добави гаусов шум към grayscale изображение. Полученото изображение да се обработи с усредняващ и Гаусов филтър със същия размер 5×5. За Гаусовия филтър да се използва стандартно отклонение σ=1. Да се сравни качеството на двата резултата, като се анализират степента на изглаждане и запазването на контурите.

Код в MATLAB:

```
% Усредняващ срещу Gaussian филтър
clc;
clear;
close all;

I = imread('cameraman.tif');

%% Добавяне на Gaussian шум
Inoise = imnoise(I,'gaussian',0,0.01);

%% Усредняващ филтър
h_mean = fspecial('average',[5 5]);

%% Gaussian филтър
h_gauss = fspecial('gaussian',[5 5],1);

%% Филтриране
Imean = imfilter(Inoise,h_mean,'replicate');
Igauss = imfilter(Inoise,h_gauss,'replicate');

%% Визуализация
figure('Name','Задание 3');

subplot(2,2,1);
imshow(I);
title('Оригинал');

subplot(2,2,2);
imshow(Inoise);
title('Gaussian шум');

subplot(2,2,3);
imshow(Imean);
title('Усредняващ филтър');

subplot(2,2,4);
imshow(Igauss);
title('Gaussian филтър');
```

```
--- Резултати ---
```




4. Да се добави salt-and-pepper шум с плътност 0.05 към grayscale изображение. Зашуменото изображение да се обработи първо чрез усредняващ филтър с размер 3×3, а след това чрез медианен филтър със същия размер. Да се сравнят резултатите и да се направи заключение кой от двата метода е по-подходящ за импулсен шум.

Код в MATLAB:

```
% Salt-and-Pepper шум
% Сравнение на усредняващ и медианен филтър
clc;
clear;
close all;

I = imread('cameraman.tif');

%% Добавяне на Salt-and-Pepper шум
Inoise = imnoise(I,'salt & pepper',0.05);

%% Усредняващ филтър
h = fspecial('average',[3 3]);

Imean = imfilter(Inoise,h,'replicate');

%% Медианен филтър
Imedian = medfilt2(Inoise,[3 3]);

%% Визуализация
figure('Name','Задание 4');
```

```
subplot(2,2,1);
imshow(I);
title('Оригинал');

subplot(2,2,2);
imshow(Inoise);
title('Salt-and-Pepper шум');

subplot(2,2,3);
imshow(Imean);
title('Усредняващ филтър');

subplot(2,2,4);
imshow(Imedian);
title('Медианен филтър');
```

```
--- Резултати ---
```




5. Да се изследва влиянието на размера на усредняващия филтър върху потискането на гаусов шум. Да се използват маски с размер: 3×3, 5×5, 7×7. За всяка маска да се обработи едно и също зашумено изображение. Представете резултатите в един графичен прозорец и анализирайте връзката между размера на маската, намаляването на шума и загубата на детайли.

Код в MATLAB:

```
% Влияние на размера на маската
clc;
clear;
close all;

I = imread('cameraman.tif');

%% Добавяне на Gaussian шум
```

```
Inoise = imnoise(I,'gaussian',0,0.01);

%% Филтри
h3 = fspecial('average',[3 3]);
h5 = fspecial('average',[5 5]);
h7 = fspecial('average',[7 7]);

%% Филтриране
I3 = imfilter(Inoise,h3,'replicate');
I5 = imfilter(Inoise,h5,'replicate');
I7 = imfilter(Inoise,h7,'replicate');

%% Визуализация
figure('Name','Задание 5');

subplot(2,2,1);
imshow(Inoise);
title('Зашумено изображение');

subplot(2,2,2);
imshow(I3);
title('Маска 3x3');

subplot(2,2,3);
imshow(I5);
title('Маска 5x5');

subplot(2,2,4);
imshow(I7);
title('Маска 7x7');
```

```
--- Резултати ---
```



6. Да се изследва влиянието на стандартното отклонение $\sigma$ върху действието на Гаусов пространствен филтър. Да се използва маска с размер 7×7 и последователно да се зададат: σ=0.5, σ=1, σ=2. С всеки от получените филтри да се обработи едно и също изображение, съдържащо гаусов шум. Да се анализира как увеличаването на σ влияе върху потискането на шума и съхраняването на детайлите.

Код в MATLAB:

```
% Влияние на sigma при Gaussian филтър
clc;
clear;
close all;

I = imread('cameraman.tif');

%% Gaussian шум
Inoise = imnoise(I,'gaussian',0,0.01);

%% Gaussian маски
h1 = fspecial('gaussian',[7 7],0.5);
h2 = fspecial('gaussian',[7 7],1);
h3 = fspecial('gaussian',[7 7],2);

%% Филтриране
I1 = imfilter(Inoise,h1,'replicate');
I2 = imfilter(Inoise,h2,'replicate');
I3 = imfilter(Inoise,h3,'replicate');

%% Визуализация
figure('Name','Задание 6');

subplot(2,2,1);
imshow(Inoise);
title('Зашумено изображение');

subplot(2,2,2);
imshow(I1);
title('\sigma = 0.5');

subplot(2,2,3);
imshow(I2);
title('\sigma = 1');

subplot(2,2,4);
imshow(I3);
title('\sigma = 2');
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Променете дисперсията на гаусовия шум последователно на 0.005, 0.02 и 0.05. За всяка стойност приложете един и същ Гаусов филтър и сравнете резултатите.
2. Променете плътността на salt-and-pepper шума на 0.01, 0.05 и 0.10. Изследвайте ефективността на медианен филтър с размер 3×3.
3. Сравнете медианни филтри с размери 3×3, 5×5 и 7×7. Анализирайте кога увеличаването на областта започва да води до значителна загуба на детайли.
4. Визуализирайте Гаусова маска чрез mesh() и анализирайте как се променя формата ѝ при различни стойности на σ.

## VI Контролни въпроси

1. Каква е разликата между точковата обработка и пространственото филтриране на цифрово изображение и каква е ролята на филтриращата маска?
2. Каква е разликата между усредняващия, Гаусовия и медианния филтър и за какъв тип шум е най-подходящ всеки от тях?
3. Как размерът на филтриращата маска и параметърът σ при Гаусовия филтър влияят върху степента на изглаждане и запазването на детайлите в изображението?

**Упражнение №14**

# Обработка на изображения в честотната област

**I Цел на упражнението** – Представяне на цифрови изображения в честотната област чрез двумерно дискретно преобразуване на Фурие и неговата ефективна реализация чрез двумерно FFT. Получаване на амплитудния и фазовия спектър на изображението, центриране на нулевата честота и интерпретация на ниско- и високочестотните компоненти. Практически се изследва действието на идеални нискочестотни и високочестотни филтри в честотната област и се сравнява ефектът им върху гладките области, детайлите и контурите на изображението.

## II Теоретична част

*2.1. Честотно представяне на цифрово изображение*

Цифровото изображение може да се разглежда като двумерен дискретен сигнал $f(x,y)$, където $x$ и $y$ са пространствените координати, а $f(x,y)$ е яркостта на съответния пиксел.

Както при едномерните сигнали, изображението може да бъде представено не само в пространствената, но и в честотната област. Това позволява да се анализира каква част от изображението се дължи на бавни пространствени изменения и каква – на бързи изменения, контури и фини детайли.

При изображенията вместо времева честота се използва понятието пространствена честота. Ниските пространствени честоти съответстват на плавни изменения на яркостта и големи еднородни области, докато високите пространствени честоти са свързани с контури, текстури, шум и малки детайли

*2.2. Двумерно дискретно преобразуване на Фурие*

За изображение с размер $M \times N$ двумерното дискретно преобразуване на Фурие се дефинира като:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)},$$

където $f(x,y)$ е изображението в пространствената област, $F(u,v)$ е неговото честотно представяне, $u$ и $v$ са индексите на пространствените честоти. Резултатът $F(u,v)$ е комплексна двумерна матрица.

*2.3. Обратно двумерно преобразуване на Фурие*

Изображението може да бъде възстановено от честотния спектър чрез обратното двумерно преобразуване:

$$F(u,v) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} f(u,v)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)},$$

В MATLAB двумерното FFT и обратното FFT се реализират съответно чрез `F = fft2(I)` и `Irec = ifft2(F)`.

*2.4. Амплитуден и фазов спектър*

Тъй като резултатът от двумерното преобразуване е комплексен, той може да бъде описан чрез модул и фаза. Амплитудният спектър е $|F(u,v)|$, а фазовият спектър - $\varphi(u,v) = \angle F(u,v)$.

Амплитудният спектър показва относителното участие на различните пространствени честоти, а фазата съдържа съществена информация за пространственото разположение на структурите в изображението.

*2.5. Центриране на спектъра*

След прилагане на `fft2()` нулевата честота се намира в горния ляв ъгъл на спектъра. За по-удобна визуализация спектърът обикновено се пренарежда така, че нулевата честота да се намира в центъра. В MATLAB това се извършва чрез `Fc = fftshift(F)`.

След центриране ниските пространствени честоти се намират около центъра, а високите – към периферията.

*2.6. Логаритмично представяне на спектъра*

Амплитудният спектър на реално изображение обикновено има много голям динамичен диапазон. Поради това директното визуализиране на $|F(u,v)|$ често не позволява да се видят слабите честотни компоненти. Затова се използва логаритмично преобразуване:

$$S(u,v) = \log(1 + |F(u,v)|)$$

Добавянето на единица предотвратява изчисляването на логаритъм от нула.

*2.7. Нискочестотно филтриране в честотната област*

Нискочестотният филтър пропуска ниските пространствени честоти и потиска високите. В резултат на което плавните области се запазват, шумът и фините детайли се потискат, изображението се изглажда а контурите могат да се размият.

Идеален нискочестотен филтър може да бъде представен като:

$$H(u,v) = \begin{cases} 1, & D(u,v) \le D_0 \\ 0, & D(u,v) > D_0 \end{cases}$$

където $D(u,v)$ е разстоянието до центъра на спектъра, а $D_0$ е граничният радиус.

*2.8. Високочестотно филтриране*

Високочестотният филтър потиска ниските пространствени честоти и пропуска високите. Идеален високочестотен филтър може да бъде представен като:

$$H(u,v) = \begin{cases} 1, & D(u,v) \le D_0 \\ 0, & D(u,v) > D_0 \end{cases}$$

В резултат се потискат плавните промени в яркостта, а контурите и фините детайли се подчертават.

*2.9. Филтриране в честотната област*

При честотно филтриране спектърът на изображението се умножава по честотната характеристика на филтъра:

$$G(u,v) = F(u,v)H(u,v).$$

След това обработеното изображение се получава чрез обратно двумерно преобразуване:

$$g(x,y) = \mathcal{F}^{-1}\{G(u,v)\}.$$

Практическата последователност е: Изображение → `fft2()` → `fftshift()` → филтриране → `ifftshift()` → `ifft2()` → обработено изображение.

## III Подготовка за упражнението

Използвани функции в MATLAB:

- **`fft2()`** – изчислява двумерното бързо преобразуване на Фурие.
- **`ifft2()`** – изчислява обратното двумерно FFT и възстановява изображението.
- **`fftshift()`** – пренарежда спектъра така, че нулевата честота да бъде в центъра.
- **`ifftshift()`** – възстановява първоначалното подреждане на спектъра преди `ifft2()`.
- **`abs()`** – определя модула на комплексния спектър.
- **`angle()`** – определя фазата на спектъра.
- **`log()`** – използва се за логаритмично представяне на амплитудния спектър.
- **`meshgrid()`** – създава двумерни координатни матрици, използвани при формиране на честотни филтри.
- **`sqrt()`** – използва се при изчисляване на радиалното разстояние от центъра на спектъра.
- **`double()`** и **`im2double()`** – преобразуват изображението към подходящ числов тип.
- **`real()`** – извлича реалната част на резултата след обратно FFT.
- **`imshow()`** – визуализира изображения.
- **`imagesc()`** – визуализира двумерни числови матрици.
- **`subplot()`**, **`title()`**, **`axis image`**, **`colormap()`** и **`colorbar()`** – използват се за оформление и визуализация.

## IV Задачи за изпълнение

1. Да се зареди grayscale изображение и да се изчисли неговото двумерно FFT чрез `fft2()`. Да се визуализира амплитудният спектър преди и след центриране чрез функцията `fftshift()`. За по-добра видимост да се използва логаритмично преобразуване на амплитудния спектър. Да се анализира разпределението на ниските и високите пространствени честоти.

Код в MATLAB:

```
% Двумерно FFT на изображение
clc;
clear;
close all;

%% Зареждане
I = imread('coins.png');

Id = im2double(I);

%% 2D FFT
F = fft2(Id);

%% Амплитуден спектър
S = log(1 + abs(F));

%% Центриране
Fc = fftshift(F);
Sc = log(1 + abs(Fc));

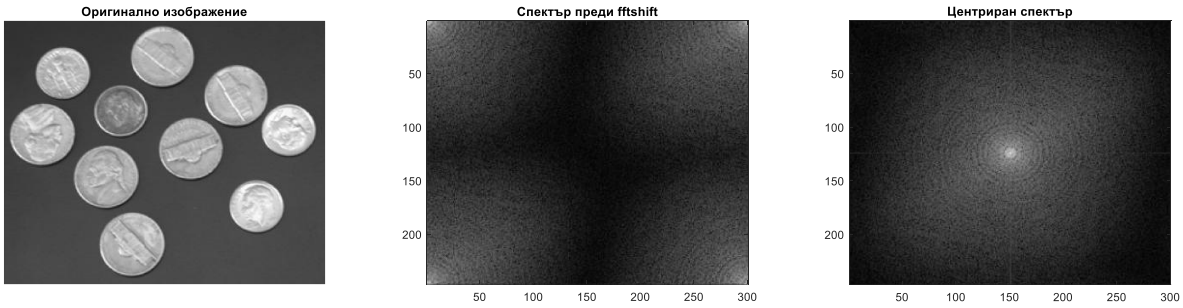
%% Визуализация
figure('Name','Задание 1');
```

```
subplot(1,3,1);
imshow(Id);
title('Оригинално изображение');

subplot(1,3,2);
imagesc(S);
axis image;
colormap gray;
title('Спектър преди fftshift');

subplot(1,3,3);
imagesc(Sc);
axis image;
colormap gray;
title('Центриран спектър');
```

```
--- Резултати ---
```




2. Да се изчислят амплитудният и фазовият спектър на изображението. Амплитудният спектър да се визуализира логаритмично, а фазовият – чрез функцията `angle()`. Да се направи анализира каква информация носят амплитудата и фазата.

Код в MATLAB:

```
% Амплитуден и фазов спектър
clc;
clear;
close all;

I = imread('coins.png');
Id = im2double(I);

%% FFT и центриране
F = fft2(Id);
Fc = fftshift(F);

%% Амплитуда
Magnitude = log(1 + abs(Fc));

%% Фаза
Phase = angle(Fc);

%% Визуализация
figure('Name','Задание 2');

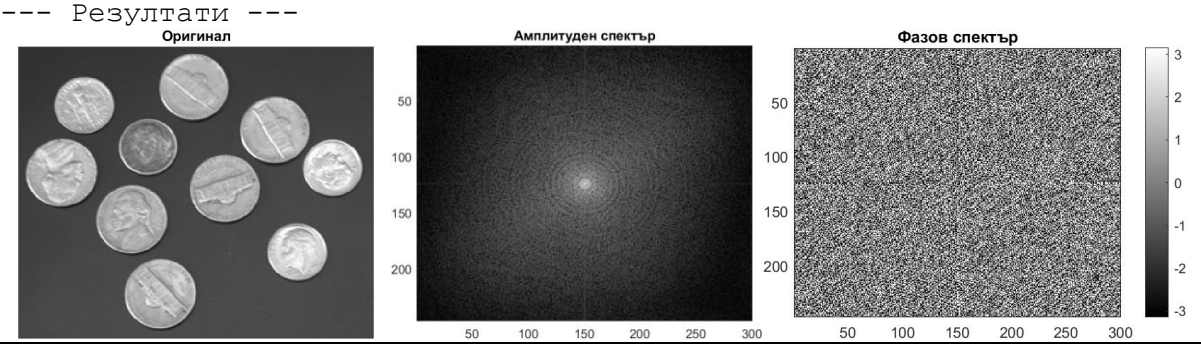
subplot(1,3,1);
imshow(Id);
```

```
title('Оригинал');

subplot(1,3,2);
imagesc(Magnitude);
axis image;
colormap gray;
title('Амплитуден спектър');

subplot(1,3,3);
imagesc(Phase);
axis image;
colormap gray;
title('Фазов спектър');
colorbar;
```

```
--- Резултати ---
```




3. Да се изчисли двумерното FFT на изображение и след това да се извърши обратно преобразуване чрез ifft2(). Да се сравнят оригиналното и възстановеното изображение. Допълнително да се изчисли максималната абсолютна разлика между тях.

Код в MATLAB:

```
% Възстановяване чрез ifft2()
clc;
clear;
close all;

I = imread('coins.png');
Id = im2double(I);

%% FFT
F = fft2(Id);

%% Обратно FFT
Irec = ifft2(F);

%% Премахване на много малка имагинерна част
Irec = real(Irec);

%% Грешка
difference = abs(Id - Irec);
max_error = max(difference(:));

%% Визуализация
figure('Name','Задание 3');
```

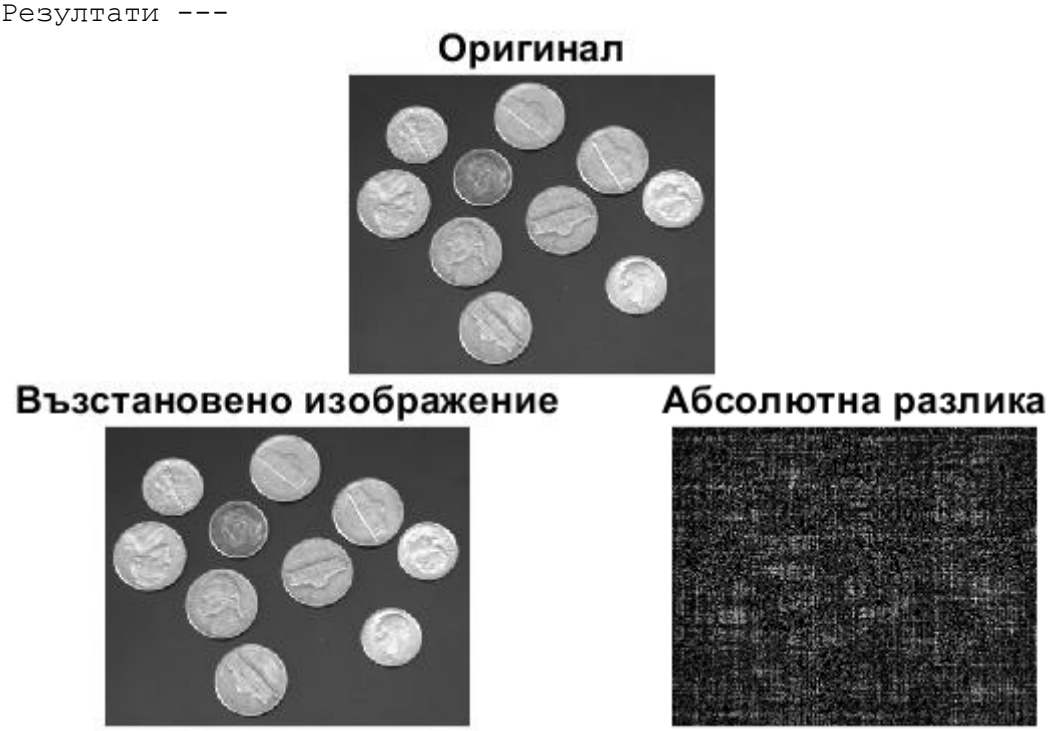
```
subplot(1,3,1);
imshow(Id);
title('Оригинал');

subplot(1,3,2);
imshow(Irec);
title('Възстановено изображение');

subplot(1,3,3);
imshow(difference,[]);
title('Абсолютна разлика');

disp(['Максимална грешка = ',num2str(max_error)]);
```

```
--- Резултати ---
```




4. Да се реализира идеален нискочестотен филтър чрез кръгова честотна маска. Да се използва граничен радиус $D_0 = 30$ пиксела в честотната област. След филтриране изображението да се възстанови чрез обратно FFT. Да се сравнят оригиналното и филтрираното изображение.

Код в MATLAB:

```
% Идеален нискочестотен филтър
clc;
clear;
close all;

I = imread('coins.png');
Id = im2double(I);

[M,N] = size(Id);

%% FFT
```

```
F = fft2(Id);
Fc = fftshift(F);

%% Координати
[u,v] = meshgrid(1:N,1:M);

centerX = floor(N/2) + 1;
centerY = floor(M/2) + 1;

%% Разстояние от центъра
D = sqrt((u-centerX).^2 + (v-centerY).^2);

%% Граничен радиус
D0 = 30;

%% Нискочестотна маска
H = double(D <= D0);

%% Филтриране
G = Fc .* H;

%% Обратно преобразуване
G0 = ifftshift(G);
Ifilt = real(ifft2(G0));

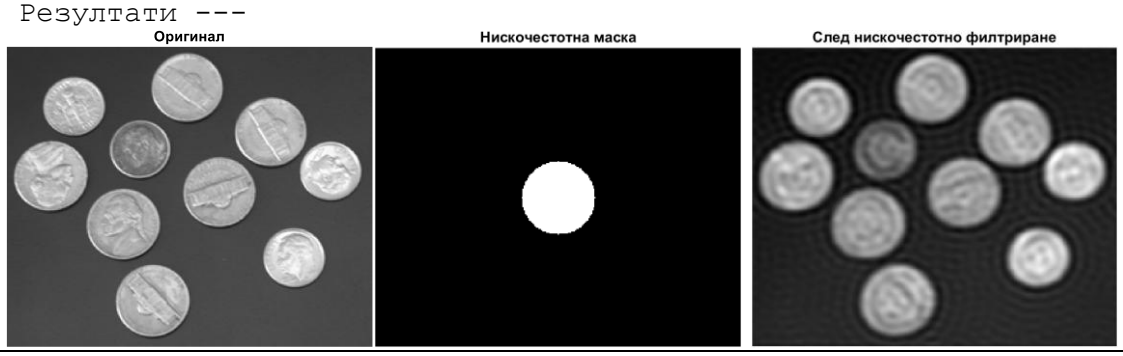
%% Визуализация
figure('Name','Задание 4');

subplot(1,3,1);
imshow(Id);
title('Оригинал');

subplot(1,3,2);
imshow(H);
title('Нискочестотна маска');

subplot(1,3,3);
imshow(Ifilt,[]);
title('След нискочестотно филтриране');
```

```
--- Резултати ---
```




5. Да се изследва влиянието на радиуса $D_0$ върху степента на изглаждане. Да се използват $D_0 = 10$, $D_0 = 30$ и $D_0 = 60$. Да се сравнят получените изображения и да се анализира връзката между радиуса на пропускателната област и количеството запазени детайли.

Код в MATLAB:

```
% Влияние на граничния радиус
I = imread('coins.png');
Id = im2double(I);

[M,N] = size(Id);

F = fft2(Id);
Fc = fftshift(F);

[u,v] = meshgrid(1:N,1:M);

centerX = floor(N/2) + 1;
centerY = floor(M/2) + 1;

D = sqrt((u-centerX).^2 + (v-centerY).^2);

%% Радиуси
D1 = 10;
D2 = 30;
D3 = 60;

%% Маски
H1 = double(D <= D1);
H2 = double(D <= D2);
H3 = double(D <= D3);

%% Филтриране
G1 = Fc .* H1;
G2 = Fc .* H2;
G3 = Fc .* H3;

I1 = real(ifft2(ifftshift(G1)));
I2 = real(ifft2(ifftshift(G2)));
I3 = real(ifft2(ifftshift(G3)));

%% Визуализация
figure('Name','Задание 5');

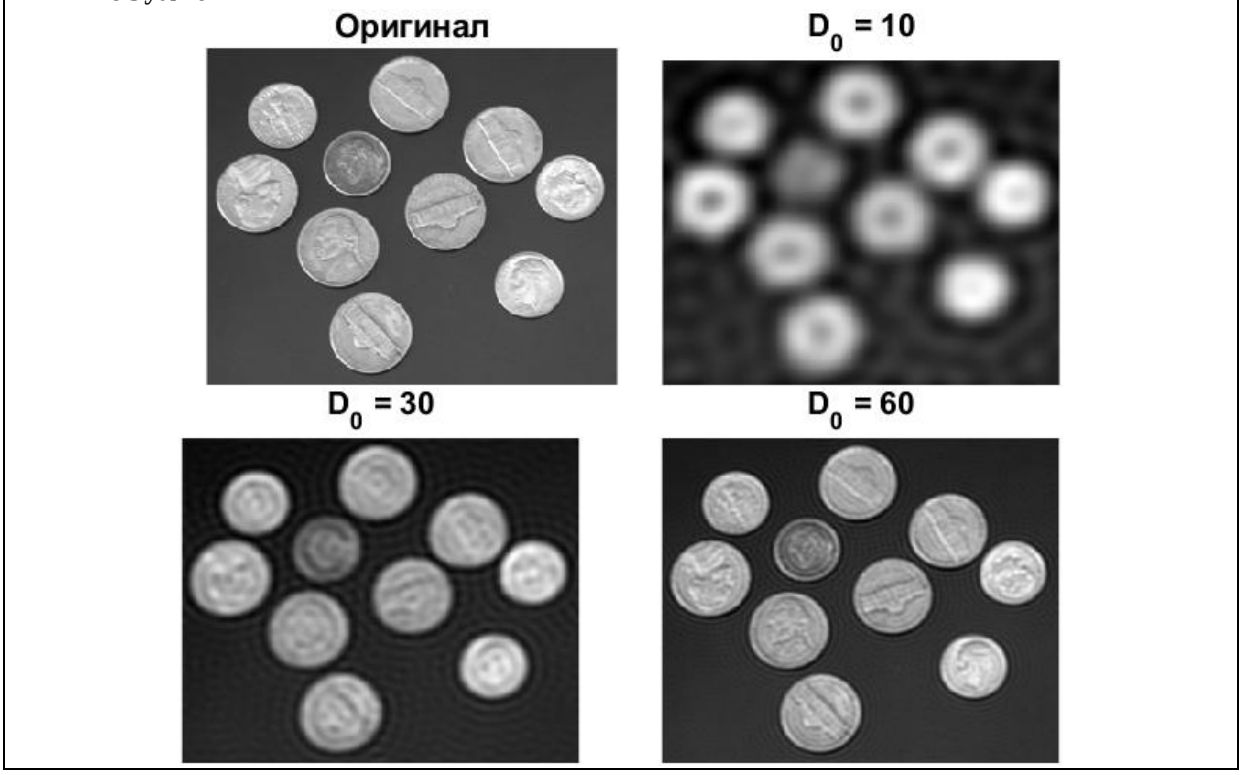
subplot(2,2,1);
imshow(Id);
title('Оригинал');

subplot(2,2,2);
imshow(I1,[]);
title('D_0 = 10');

subplot(2,2,3);
imshow(I2,[]);
title('D_0 = 30');

subplot(2,2,4);
imshow(I3,[]);
title('D_0 = 60');
```

```
--- Резултати ---
```




6. Да се реализира идеален високочестотен филтър с радиус $D_0 = 30$. Да се потиснат ниските пространствени честоти и да се анализира кои структури от изображението остават след филтрирането.

Код в MATLAB:

```
% Идеален високочестотен филтър
clc;
clear;
close all;

I = imread('coins.png');
Id = im2double(I);

[M,N] = size(Id);

%% FFT
F = fft2(Id);
Fc = fftshift(F);

%% Честотни координати
[u,v] = meshgrid(1:N,1:M);

centerX = floor(N/2) + 1;
centerY = floor(M/2) + 1;

D = sqrt((u-centerX).^2 + (v-centerY).^2);

%% Радиус
D0 = 30;

%% Високочестотна маска
H = double(D > D0);
```

```
%% Филтриране
G = Fc .* H;

%% Обратно преобразуване
Ifilt = real(ifft2(ifftshift(G)));

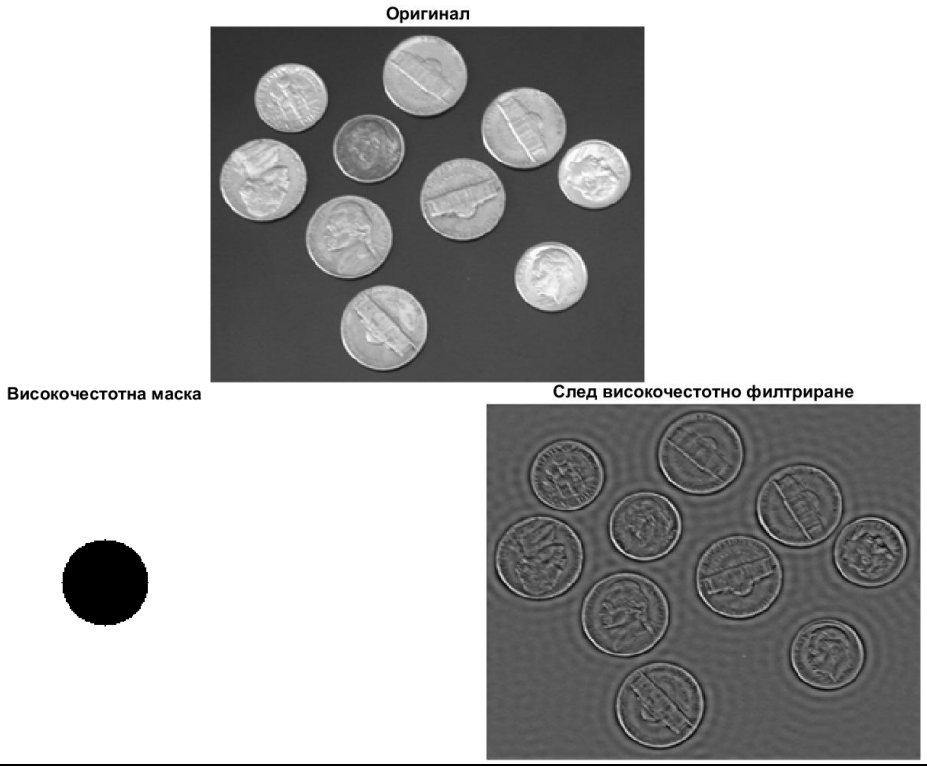
%% Визуализация
figure('Name','Задание 6');

subplot(1,3,1);
imshow(Id);
title('Оригинал');

subplot(1,3,2);
imshow(H);
title('Високочестотна маска');

subplot(1,3,3);
imshow(Ifilt,[]);
title('След високочестотно филтриране');
```

```
--- Резултати ---
```




## V Допълнителни задачи

1. Променете радиуса на идеалния нискочестотен филтър на 5, 20, 50 и 100. Анализирайте как се изменя степента на изглаждане.
2. Повторете високочестотното филтриране за различни стойности на $D_0$ и сравнете количеството запазени детайли.

3. Добавете гаусов шум към изображението и приложете нискочестотно филтриране в честотната област. Сравнете резултата с Гаусовото пространствено филтриране от Упражнение 13.
4. Използвайте RGB изображение, обработете отделно трите му цветови канала в честотната област и след това ги обединете отново.
5. Сравнете амплитудните спектри на оригинално изображение и на негово силно изгладено копие.
6. Сравнете спектрите на изображение с много гладки области и изображение с множество фини детайли и текстури.

## VI Контролни въпроси

1. Какво представлява двумерното преобразуване на Фурие и как се интерпретират ниските и високите пространствени честоти в цифрово изображение?
2. Какво е действието на функцията `fftshift()` и защо спектърът обикновено се визуализира в логаритмичен мащаб?
3. Каква е разликата между нискочестотното и високочестотното филтриране в честотната област и как влияят те върху контурите и детайлите на изображението?

**Упражнение №15**

# Откриване на контури, морфологична обработка и сегментация на обекти в цифрови изображения

**I Цел на упражнението** – Запознаване с основните методи за откриване и анализ на обекти в цифрови изображения. Откриване на контури чрез градиентни оператори, преобразуване на grayscale изображения в бинарни чрез прагова сегментация и основни морфологични операции – ерозия, дилатация, отваряне и затваряне. Отстраняване на нежелани малки области, етикетиране на свързани компоненти, преброяване на обекти и определяне на техните геометрични характеристики като площ, център и ограничителен правоъгълник.

## II Теоретична част

*2.1. Сегментация на цифрови изображения*

Сегментацията представлява процес на разделяне на цифровото изображение на области, съответстващи на различни обекти или структурни компоненти. Основната цел е да се отделят интересуващите ни обекти от фона, така че впоследствие да могат да бъдат анализирани независимо.

Ако grayscale изображението се означи с $f(x,y)$, резултатът от сегментацията често е бинарно изображение:

$$g(x,y) = \begin{cases} 1, & \text{ако пикселът принадлежи на обекта} \\ 0, & \text{ако пикселът принадлежи на фона} \end{cases}$$

Сегментацията е основна стъпка при автоматично разпознаване, преброяване, измерване и проследяване на обекти.

*2.2. Прагова сегментация*

Един от най-простите методи за сегментация е използването на праг T. Стойността на всеки пиксел се сравнява с прага и според резултата се определя принадлежността му към обекта или фона:

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \le T \end{cases}$$

Изборът на подходящ праг е критичен за качеството на сегментацията. При изображения с ясно различаващи се яркости на обектите и фона може да се използва един глобален праг.

В MATLAB автоматична оценка на глобалния праг може да бъде получена чрез `graythresh()`. Функцията използва метода на Otsu, който избира праг чрез статистическото разпределение на пикселните стойности.

*2.3. Неравномерно осветяване и корекция на фона*

При реални изображения фонът невинаги има еднаква яркост. Това може да затрудни използването на един глобален праг.

Подобен проблем присъства и в rice.png, където яркостта на фона се изменя пространствено. Един начин за намаляване на този ефект е първо да се оцени бавно изменящият се фон и след това той да се извади от оригиналното изображение:

$$f_c(x,y) = f(x,y) - b(x,y),$$

където $b(x, y)$ е оценката на фона.

Морфологичното отваряне с достатъчно голям структуриращ елемент може да се използва за получаване на такава оценка.

*2.4. Откриване на контури*

Контурът представлява област, в която яркостта на изображението се изменя рязко. Следователно откриването на контури се основава на анализ на пространствените производни.

За двумерно изображение градиентът може да бъде представен като:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix}$$

където $G_x$ и $G_y$ описват измененията по хоризонталното и вертикалното направление. Модулът на градиента е:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2}$$

Големи стойности на градиента съответстват на вероятни контури.

Сред широко използваните оператори са Sobel, Prewitt, Roberts и Canny. Canny включва повече етапи на обработка и обикновено осигурява по-добра локализация на контурите при наличие на шум.

*2.5. Морфологична обработка*

Математическата морфология представлява набор от операции за обработка на формата и структурата на обектите в бинарни изображения.

Основният елемент при морфологичната обработка е структуриращият елемент. Той представлява малка геометрична форма, която се използва като сонда при изследване на изображението. Структуриращият елемент може да бъде квадратен, правоъгълен, дисковиден, линеен и др.

*2.6. Ерозия*

Ерозията намалява размера на белите обекти в бинарното изображение. Граничните пиксели се премахват в зависимост от формата и размера на структуриращия елемент. Символично:

$$A \ominus B,$$

където A е бинарното множество, а B – структуриращият елемент.

Ерозията може да се използва за премахване на малки обекти, разделяне на слабо свързани структури и намаляване на дебелината на обектите.

*2.7. Дилатация*

Дилатацията увеличава размера на обектите чрез добавяне на пиксели по техните граници. Символично:

$$A \oplus B.$$

Дилатацията може да се използва за свързване на близко разположени области, запълване на малки прекъсвания и увеличаване на дебелината на структурите.

*2.8. Морфологично отваряне*

Отварянето представлява последователно прилагане на ерозия и дилатация:

$$A \circ B = (A \oplus B) \oplus B$$

То се използва основно за премахване на малки обекти и издатини, като в същото време приблизително се запазва размерът на по-големите структури.

*2.9. Морфологично затваряне*

Затварянето представлява последователност от дилатация и ерозия:

$$A \bullet B = (A \oplus B) \oplus B$$

Операцията може да запълва малки прекъсвания, тесни празнини и да свързва близко разположени части на един обект.

*2.10. Свързани компоненти*

След бинаризацията всеки отделен обект може да бъде разглеждан като множество от взаимно свързани пиксели.

При двумерни изображения често се използват:

- 4-свързаност – пикселът се счита свързан с непосредствените си съседи нагоре, надолу, наляво и надясно;
- 8-свързаност – допълнително се включват и четирите диагонални съседи.

Функцията `bwlabel()` може да присвои различен номер на всеки свързан обект в бинарно изображение.

*2.11. Измерване на обекти*

След откриването на свързаните области могат да бъдат определени различни техни геометрични характеристики.

Сред най-полезните са:

- Area – площта на обекта, измерена в пиксели;
- Centroid – координатите на геометричния център;
- BoundingBox – най-малкият правоъгълник, който огражда обекта.

Тези характеристики могат да се получат чрез `regionprops()`. Така цифровата обработка преминава от анализ на отделни пиксели към извличане на информация за реални обекти.

**III Подготовка за упражнението**

Използвани функции в MATLAB:

- **`imread()`** – зарежда цифрово изображение.
- **`imshow()`** – визуализира изображение.
- **`imadjust()`** – подобрява контраста чрез преобразуване на диапазона на яркостите.
- **`graythresh()`** – определя автоматично глобален праг за бинаризация.
- **`im2bw()`** – преобразува grayscale изображение в бинарно изображение при зададен праг.
- **`edge()`** – открива контури чрез методи като Sobel, Prewitt, Roberts и Canny.
- **`strel()`** – създава структуриращ елемент за морфологична обработка.
- **`imerode()`** – извършва морфологична ерозия.
- **`imdilate()`** – извършва морфологична дилатация.
- **`imopen()`** – извършва морфологично отваряне.
- **`imclose()`** – извършва морфологично затваряне.
- **`imsubtract()`** – изважда едно изображение от друго.
- **`bwareaopen()`** – премахва свързани области с площ под зададен брой пиксели.
- **`bwlabel()`** – етикетира свързаните компоненти в бинарно изображение.
- **`label2rgb()`** – преобразува матрица с етикети в цветно изображение за визуализация на отделните обекти.
- **`regionprops()`** – определя геометрични характеристики на сегментираните области.

- **`rectangle()`** – изчертава правоъгълник върху графично изображение.
- **`plot()`** – използва се за означаване на центровете на обектите.

**IV Задачи за изпълнение**

1. Да се зареди grayscale изображението *rice.png* и да се приложат четири метода за откриване на контури – Sobel, Prewitt, Roberts и Canny. Получените бинарни изображения да се представят в един графичен прозорец. Да се сравнят методите по отношение на количеството открити контури, тяхната непрекъснатост и чувствителността към малки изменения на яркостта.

Код в MATLAB:

```
% Сравнение на методи за откриване на контури
clc;
clear;
close all;

%% Зареждане на изображението
I = imread('rice.png');

%% Откриване на контури
BW_sobel   = edge(I,'sobel');
BW_prewitt = edge(I,'prewitt');
BW_roberts = edge(I,'roberts');
BW_canny   = edge(I,'canny');

%% Визуализация
figure('Name','Задание 1 - Откриване на контури');

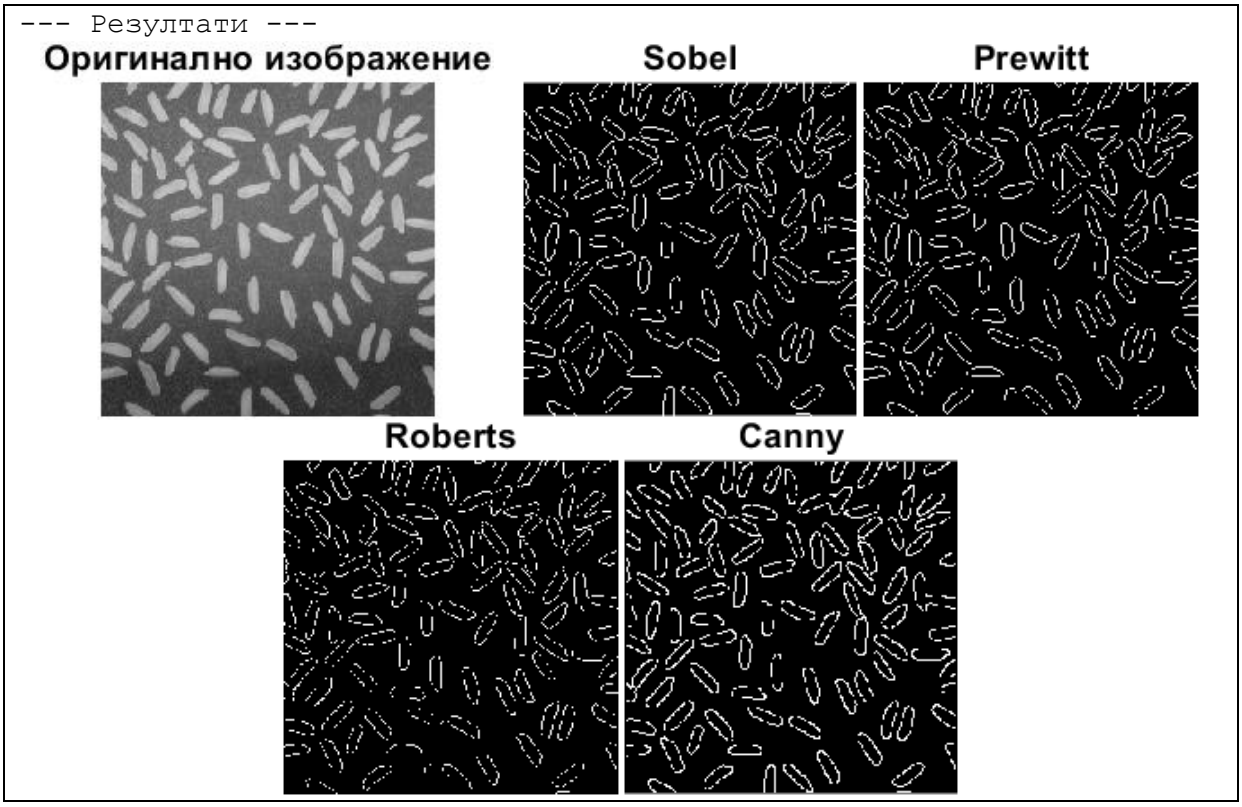
subplot(2,3,1);
imshow(I);
title('Оригинално изображение');

subplot(2,3,2);
imshow(BW_sobel);
title('Sobel');

subplot(2,3,3);
imshow(BW_prewitt);
title('Prewitt');

subplot(2,3,4);
imshow(BW_roberts);
title('Roberts');

subplot(2,3,5);
imshow(BW_canny);
title('Canny');
```




2. Да се сегментира изображението rice.png чрез автоматично определен глобален праг. Приложете graythresh() директно върху оригиналното изображение, оценете неравномерния фон чрез морфологично отваряне с дисковиден структуриращ елемент, извадете получения фон от изображението и повторете бинаризацията. Сравнете резултатите и анализирайте влиянието на неравномерното осветяване върху глобалната прагова сегментация.

Код в MATLAB:

```
% Прагова сегментация и корекция на неравномерен фон
clc;
clear;
close all;

%% Зареждане
I = imread('rice.png');

%% Сегментация без корекция на фона
T1 = graythresh(I);
BW1 = im2bw(I,T1);

%% Оценка на фона
se = strel('disk',15);
background = imopen(I,se);

%% Изваждане на фона
Icorr = imsubtract(I,background);

%% Подобряване на контраста
Icorr = imadjust(Icorr);

%% Нов праг
```

```
T2 = graythresh(Icorr);
BW2 = im2bw(Icorr,T2);

%% Визуализация
figure('Name','Задание 2');

subplot(2,3,1);
imshow(I);
title('Оригинал');

subplot(2,3,2);
imshow(BW1);
title('Директна бинаризация');

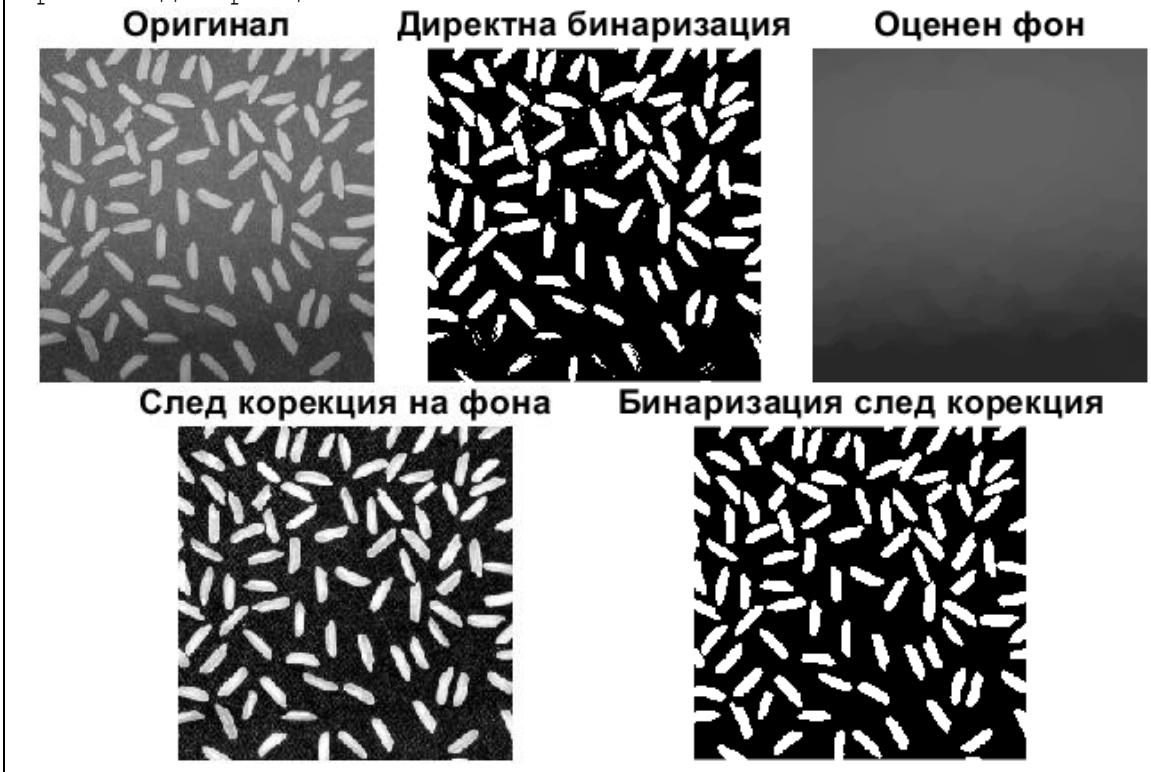
subplot(2,3,3);
imshow(background);
title('Оценен фон');

subplot(2,3,4);
imshow(Icorr);
title('След корекция на фона');

subplot(2,3,5);
imshow(BW2);
title('Бинаризация след корекция');

%% Извеждане на праговете
disp(['Праг преди корекция: ',num2str(T1)]);
disp(['Праг след корекция: ',num2str(T2)]);
```

```
--- Резултати ---
Праг преди корекция: 0.51373
Праг след корекция: 0.48431
```



3. Да се използва бинарното изображение, получено след корекция на фона и прагова сегментация. Създайте дисковиден структуриращ елемент с малък радиус и приложете последователно ерозия, дилатация, отваряне и затваряне. Визуализирайте резултатите и анализирайте как всяка операция изменя размера, границите и взаимната свързаност на сегментираните обекти.

Код в MATLAB:

```
% Морфологична обработка
clc;
clear;
close all;

%% Подготовка на бинарно изображение
I = imread('rice.png');

se_background = strel('disk',15);
background = imopen(I,se_background);

Icorr = imsubtract(I,background);
Icorr = imadjust(Icorr);

T = graythresh(Icorr);
BW = im2bw(Icorr,T);

%% Структуриращ елемент
se = strel('disk',2);

%% Морфологични операции
BW_erode  = imerode(BW,se);
BW_dilate = imdilate(BW,se);
BW_open   = imopen(BW,se);
BW_close  = imclose(BW,se);

%% Визуализация
figure('Name','Задание 3 - Морфология');

subplot(2,3,1);
imshow(BW);
title('Оригинално бинарно');

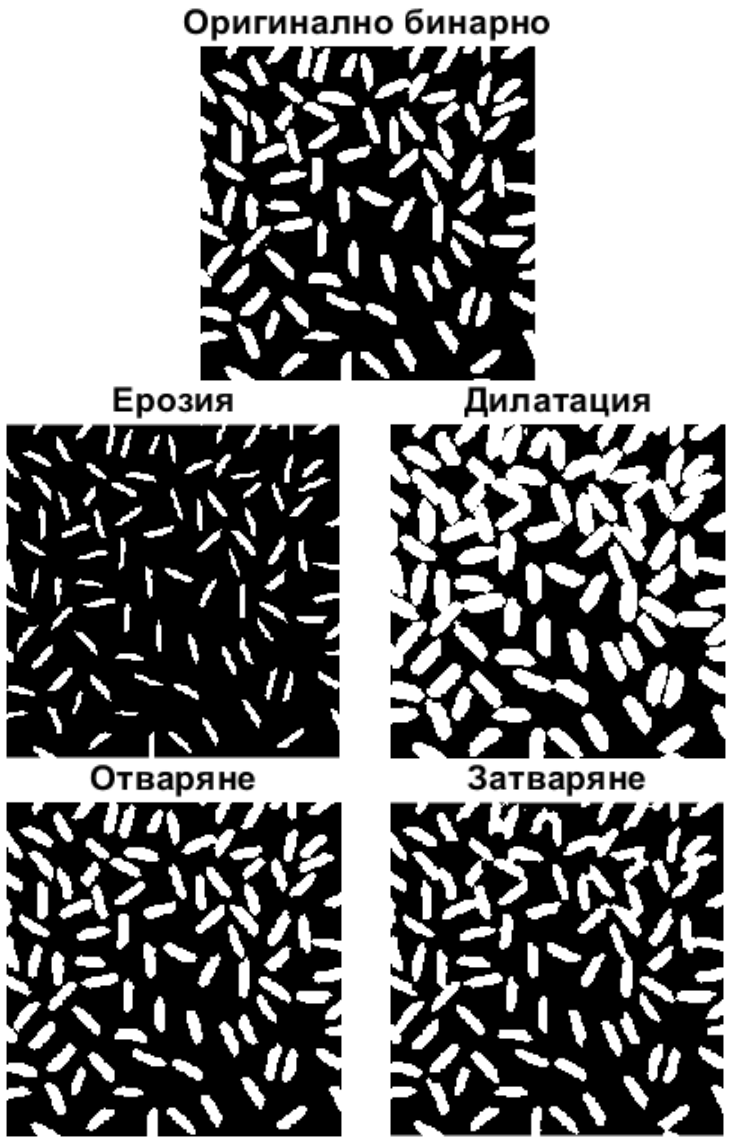
subplot(2,3,2);
imshow(BW_erode);
title('Ерозия');

subplot(2,3,3);
imshow(BW_dilate);
title('Дилатация');

subplot(2,3,4);
imshow(BW_open);
title('Отваряне');

subplot(2,3,5);
imshow(BW_close);
title('Затваряне');
```

```
--- Резултати ---
```




4. Да се премахнат малките нежелани области от сегментираното изображение чрез задаване на минимална допустима площ. Използвайте bwareaopen() с праг 50 пиксела. Етикетирайте оставащите свързани компоненти чрез функцията bwlabel() и визуализирайте всеки обект с различен цвят. Определете броя на откритите обекти.

Код в MATLAB:

```
% Премахване на малки области и етикетиране
clc;
clear;
close all;

%% Подготовка
I = imread('rice.png');

se = strel('disk',15);

background = imopen(I,se);
```

```
Icorr = imsubtract(I,background);
Icorr = imadjust(Icorr);

T = graythresh(Icorr);
BW = im2bw(Icorr,T);

%% Премахване на малки области
BWclean = bwareaopen(BW,50);

%% Етикетиране на обектите
[L,num] = bwlabel(BWclean,8);

%% Цветно представяне
RGBlabels = label2rgb(L);

%% Визуализация
figure('Name','Задание 4');

subplot(1,3,1);
imshow(BW);
title('След бинаризация');

subplot(1,3,2);
imshow(BWclean);
title('След премахване на малки области');

subplot(1,3,3);
imshow(RGBlabels);
title('Етикетирани обекти');

%% Брой обекти
disp(['Брой открити обекти: ',num2str(num)]);
```

```
--- Резултати ---
Брой открити обекти: 93
```






5. Да се определят площта, центърът и ограничителният правоъгълник на всеки сегментиран обект чрез функцията regionprops(). Ограничителните правоъгълници и центровете на обектите да бъдат нанесени върху оригиналното изображение. Да се изведат в Command Window площите на първите няколко открити обекта.

Код в MATLAB:

```
% Геометричен анализ на сегментирани обекти
clc;
clear;
close all;

%% Сегментация
I = imread('rice.png');

se = strel('disk',15);

background = imopen(I,se);
Icorr = imsubtract(I,background);
Icorr = imadjust(Icorr);

T = graythresh(Icorr);
BW = im2bw(Icorr,T);

%% Премахване на малки области
BWclean = bwareaopen(BW,50);

%% Етикетиране
[L,num] = bwlabel(BWclean,8);

%% Геометрични характеристики
stats = regionprops(L,'Area','Centroid','BoundingBox');

%% Визуализация
figure('Name','Задание 5');

imshow(I);
title('Открити обекти');
hold on;

for k = 1:num

    rectangle('Position',stats(k).BoundingBox,'EdgeColor','r','LineWidth',1);

    plot(stats(k).Centroid(1),stats(k).Centroid(2),'b+');
```

```
end

hold off;

%% Извеждане на информация
disp(['Общ брой обекти: ',num2str(num)]);

Nshow = min(10,num);

disp(' ');
disp('Площ на първите обекти:');

for k = 1:Nshow
    disp(['Обект ',num2str(k),': Area = ',num2str(stats(k).Area),' pixels']);
end
```

```
--- Резултати ---
Общ брой обекти: 93

Площ на първите обекти:
Обект 1: Area = 138 pixels
Обект 2: Area = 121 pixels
Обект 3: Area = 68 pixels
Обект 4: Area = 178 pixels
Обект 5: Area = 157 pixels
Обект 6: Area = 286 pixels
Обект 7: Area = 196 pixels
Обект 8: Area = 134 pixels
Обект 9: Area = 176 pixels
Обект 10: Area = 208 pixels
```




6. Да се обединят изучените операции в един последователен алгоритъм за автоматична обработка на изображението rice.png. Алгоритъмът трябва да включва: оценка на фона → корекция → подобряване на контраста → бинаризация → премахване на малки области → етикетиране → измерване → преброяване. Да се визуализира оригиналното изображение и крайният резултат, като всеки открит обект бъде ограден с правоъгълник и центърът му бъде означен.

Код в MATLAB:

```
% Цялостна сегментация, преброяване и анализ на обекти
clc;
```

```
clear;
close all;

%% 1. Зареждане
I = imread('rice.png');

%% 2. Оценяване на неравномерния фон
se_background = strel('disk',15);
background = imopen(I,se_background);

%% 3. Корекция на фона
Icorr = imsubtract(I,background);

%% 4. Подобряване на контраста
Ienhanced = imadjust(Icorr);

%% 5. Автоматично определяне на праг
T = graythresh(Ienhanced);

%% 6. Бинаризация
BW = im2bw(Ienhanced,T);

%% 7. Премахване на малки обекти
BWclean = bwareaopen(BW,50);

%% 8. Етикетиране
[L,num] = bwlabel(BWclean,8);

%% 9. Измерване
stats = regionprops(L,'Area','Centroid','BoundingBox');

%% 10. Визуализация на отделните етапи
figure('Name','Задание 6 - Етапи');

subplot(2,3,1);
imshow(I);
title('1. Оригинал');

subplot(2,3,2);
imshow(background);
title('2. Фон');

subplot(2,3,3);
imshow(Icorr);
title('3. Корекция');

subplot(2,3,4);
imshow(Ienhanced);
title('4. Подобрен контраст');

subplot(2,3,5);
imshow(BW);
title('5. Бинаризация');

subplot(2,3,6);
imshow(BWclean);
title('6. Почистено изображение');

%% 11. Краен резултат
```

```
figure('Name','Задание 6 - Краен резултат');

imshow(I);
title(['Открити обекти: ',num2str(num)]);
hold on;

for k = 1:num

    rectangle('Position',stats(k).BoundingBox,'EdgeColor','r','LineWidth',1);

    plot(stats(k).Centroid(1),stats(k).Centroid(2),'b+');
end

hold off;

%% 12. Резултати
disp(' ');
disp('--- Резултати от анализа ---');

disp(['Използван праг: ',num2str(T)]);
disp(['Брой открити обекти: ',num2str(num)]);

%% Средна площ
if num > 0

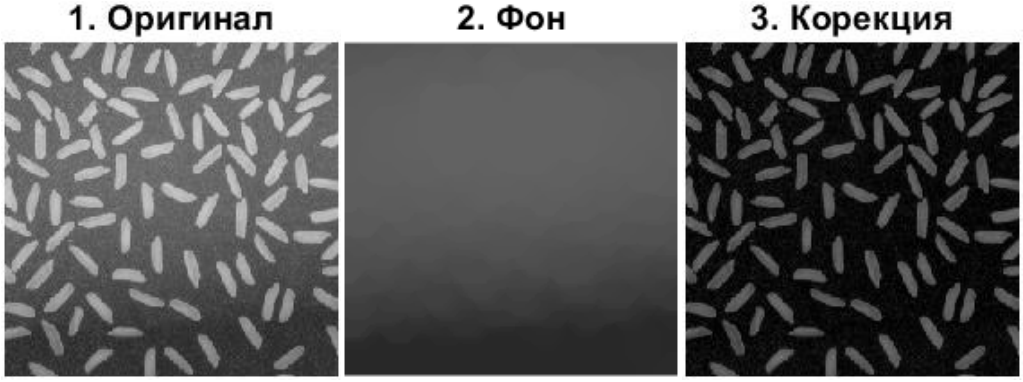
    totalArea = 0;

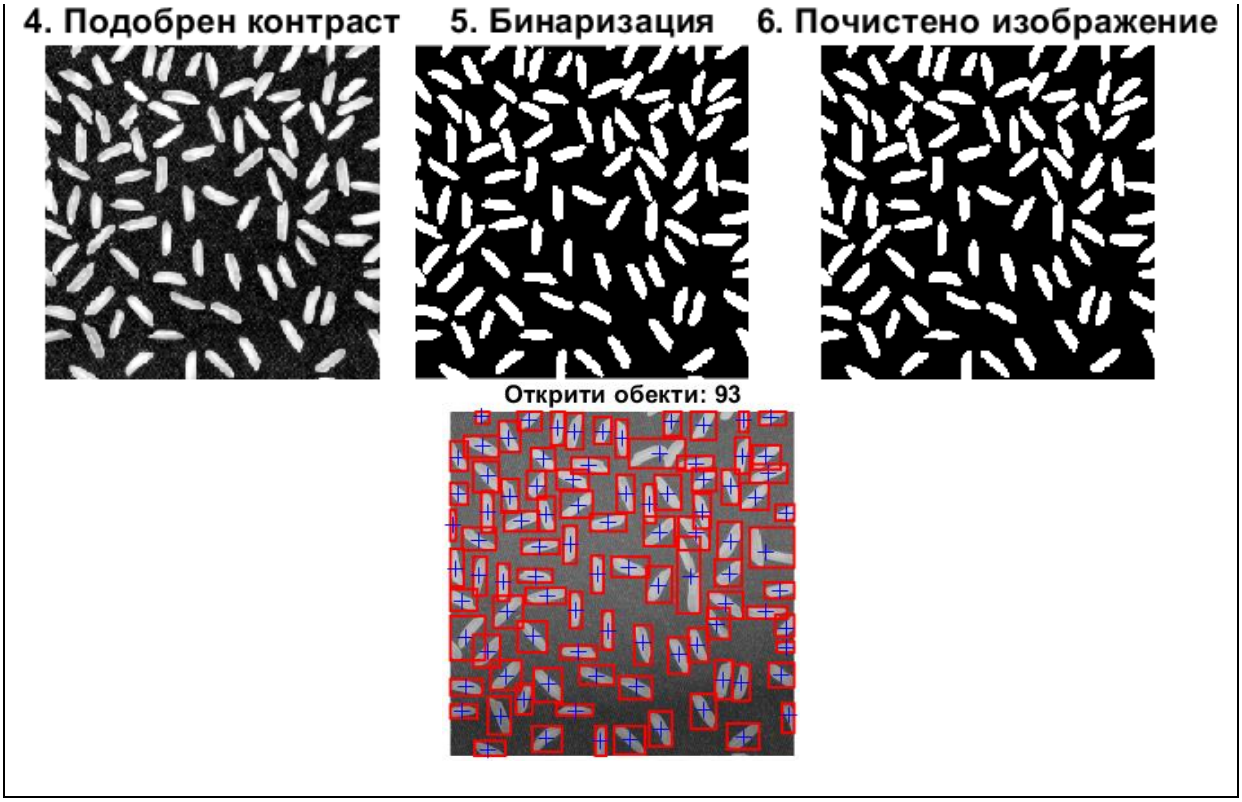
    for k = 1:num
        totalArea = totalArea + stats(k).Area;
    end

    meanArea = totalArea/num;

    disp(['Средна площ на обектите: ',num2str(meanArea),' pixels']);
end
```

```
--- Резултати ---
Използван праг: 0.48431
Брой открити обекти: 93
Средна площ на обектите: 189.0968 pixels
```






## V Допълнителни задачи

1. Променете минималната площ в `bwareaopen()` последователно на 20, 50, 100 и 200 пиксела. Анализирайте как се изменя броят на откритите обекти.
2. Променете радиуса на структуриращия елемент при оценяване на фона и анализирайте влиянието му върху качеството на сегментацията.
3. Сравнете резултатите при 4-свързаност и 8-свързаност в `bwlabel()`.
4. Определете най-малкия и най-големия сегментиран обект според неговата площ.
5. Изчислете средната площ на всички открити обекти и премахнете тези, чиято площ се различава значително от средната.
6. Използвайте собствено изображение с няколко ясно различими обекта и реализирайте цялата последователност за сегментация, преброяване и измерване.

## VI Контролни въпроси

1. Какво представлява сегментацията на цифрово изображение и как изборът на праг влияе върху отделянето на обектите от фона?
2. Как действат ерозията, дилатацията, морфологичното отваряне и затваряне и с каква цел се използват?
3. Какво представляват свързаните компоненти и как чрез `bwlabel()` и `regionprops()` могат да бъдат преброени и измерени отделните обекти?

# Използвана литература


1. Hussain, Zahir M., Amin Z. Sadik, and Peter O'Shea. Digital signal processing: an introduction with MATLAB and applications. Berlin: Springer, 2011.
2. Leis, John W. Digital signal processing using MATLAB for students and researchers. John Wiley & Sons, 2011.
3. Khan, M. N., Hasnain, S. K., Jamil, M., and Imran, A. Electronic Signals and Systems: Analysis, Design and Applications. CRC Press, 2022.
4. Baskar, A., Rajappa, M., Vasudevan, S. K., and Murugesh, T. S. Digital image processing. CRC Press, 2023.
5. Umbaugh, Scott E. Digital image processing and analysis: computer vision and image analysis. CRC Press, 2023.

# Приложение

| Функция | Синтаксис | Описание |
|---|---|---|
| **sin** | `Y = sin(X)` | Връща синуса за всеки елемент от X. Функцията sin работи поелементно с масиви. Функцията приема както реални, така и комплексни входни данни.<br>• За реални стойности на X, sin(X) връща реални стойности в интервала [-1, 1].<br>• За комплексни стойности на X, sin(X) връща комплексни стойности. |
| **cos** | `Y = cos(X)` | Връща косинуса за всеки елемент от X. Функцията cos работи поелементно с масиви. Функцията приема както реални, така и комплексни входни данни.<br>• За реални стойности на X, cos(X) връща реални стойности в интервала [-1, 1].<br>• За комплексни стойности на X, cos(X) връща комплексни стойности. |
| **sawtooth** | `x = sawtooth(t)` | Генерира трионообразна вълна с период 2π за елементите на времевия масив t. `sawtooth` е подобна на синусоидалната функция, но създава трионообразна вълна с пикове –1 и 1. Трионообразната вълна се определя като –1 при стойности кратни на 2π и нараства линейно с времето с наклон 1/π във всички останали моменти. |
| | `x = sawtooth(t,xmax)` | Генерира модифицирана триъгълна вълна с максимално местоположение във всеки период, контролиран от xmax. Задайте xmax на 0.5, за да генерирате стандартна триъгълна вълна. |
| **square** | `x = square(t)` | Генерира правоъгълна вълна с период 2π за елементите на времевия масив t. square е подобна на синусоидалната функция, но създава правоъгълна вълна със стойности –1 и 1. |
| **randn** | `X = randn(n)` | Връща матрица от нормално разпределени случайни числа с размери n x n. |
| **size** | `sz = size(A)` | Връща вектор-ред, чиито елементи са дължините на съответните измерения на A. Например, ако A е матрица 3x4, тогава size(A) връща вектора [3 4]. |
| **length** | `L = length(X)` | Връща дължината на най-голямото измерение на масива в X. За вектори, дължината е просто броят на елементите. За масиви с повече измерения, дължината е `max(size(X))`. Дължината на празен масив е нула. |
| **linspace** | `y = linspace(x1,x2)` | Връща вектор от редове с равномерно разположени точки между x1 и x2. По подразбиране linspace генерира 100 точки. |
| | `y = linspace(x1,x2,n)` | Генерира n точки, разстоянието между които е (x2-x1)/(n-1).<br>`linspace` е подобен на оператора за двоеточие „:“, но дава директен контрол върху броя на точките и винаги включва крайните точки. „Lin“ в името „`linspace`“ се отнася до генериране на линейно разположени стойности, за разлика от функцията `logspace`, която генерира логаритмично разположени стойности. |

| Функция | Синтаксис | Описание |
|---|---|---|
| **plot** | `plot(X,Y)` | Създава двуизмерна линейна графика на данните в Y спрямо съответните стойности в X.<br>За да начертаете набор от координати, свързани с отсечки, задайте X и Y като вектори с еднаква дължина. |
| | `plot(X,Y,LineSpec)` | Създава графиката, използвайки зададения стил на линията, маркер и цвят. |
| | `plot(X1,Y1,...,Xn,Yn)` | Начертава множество двойки x- и y-координати върху един и същ набор от оси. Използвайте този синтаксис като алтернатива на задаването на координати като матрици. |
| **stem** | `stem(Y)` | Изобразява последователността от данни Y като „стълбове“, които се простират от базова линия по оста x. Стойностите на данните са обозначени с кръгове, завършващи всеки стълб.<br>• Ако Y е вектор, тогава скалата по оста x варира от 1 до `length(Y)`.<br>• Ако Y е матрица, тогава `stem` изобразява всички елементи в реда спрямо една и съща стойност x, а скалата по оста x варира от 1 до броя на редовете в Y. |
| | `stem(X,Y)` | Изобразява последователността от данни Y при стойности, зададени от X. Входните данни X и Y трябва да са вектори или матрици с еднакъв размер. Освен това, X може да бъде вектор от ред или колона, а Y трябва да е матрица с length(X) реда. |
| **subplot** | `subplot(m,n,p)` | Разделя текущата фигура на мрежа m x n и създава оси в позицията, зададена от p. MATLAB® номерира позициите на подграфиките по ред. Първата подграфика е първата колона на първия ред, втората подграфика е втората колона на първия ред и така нататък. Ако осите съществуват в зададената позиция, тогава тази команда прави осите текущи оси. |
| **figure** | `figure` | Създава нов прозорец на фигурата, използвайки стойностите на свойствата по подразбиране. Получената фигура е текущата фигура. |
| **for** | `for index = values`<br>`  statements`<br>`end` | Изпълнява група от `statements` в цикъл за определен брой пъти. `values` има една от следните форми:<br>• `initVal:endVal` - Увеличава индексната променлива от `initVal` до `endVal` с 1 и повтаря изпълнението на `statements`, докато `index` стане по-голям от `endVal`.<br>• `initVal:step:endVal` — Увеличава индекса със стойността `step` на всяка итерация или намалява индекса, когато `step` е отрицателен.<br>• `valArray` — Създава вектор-колона, `index`, от следващите колони на масив `valArray` на всяка итерация. Входната `valArray` може да бъде от всеки тип данни на MATLAB®, включително вектор от символи, масив от клетки или структура. |
| **num2str** | `s = num2str(A)` | Преобразува числов масив в символен масив, който представлява числата. Изходният формат зависи от величините на оригиналните стойности. `num2str` е |

| Функция | Синтаксис | Описание |
|---|---|---|
| | | полезен за етикетиране и озаглавяване на графики с числови стойности. |
| **conv** | `w = conv(u,v)` | връща конволюцията на вектори u и v. Ако u и v са вектори на полиномни коефициенти, конволвирането им е еквивалентно на умножаване на двата полинома. |
| | `w = conv(u,v,shape)` | Връща подсекция от конволюцията, както е зададено от shape. Например, `conv(u,v,"same")` връща само централната част на конволюцията, със същия размер като u, а `conv(u,v,"valid")` връща само частта от конволюцията, изчислена без допълнените с нули ръбове. |
| **zeros** | `X = zeros` | Връща скалара 0 |
| | `X = zeros(n)` | Връща матрица от нули n x n. |
| | `X = zeros(sz)` | Връща масив от нули, където векторът на размера sz определя size(X). Например, zeros([2 3]) връща матрица 2x3. |
| **filter** | `tsout = filter(tsin,b,a)` | Прилага филтъра на рационалната предавателна функция $\frac{b(z-1)}{a(z-1)}$ към равномерно разположените данни във времевия ред tsin. Числителят *b* и знаменателят *a* са вектори, съдържащи коефициентите на предавателната функция. |
| **freqz** | `[h,w] = freqz(b,a,n)` | Връща честотната характеристика на зададения цифров филтър. Задайте цифров филтър с коефициенти в числителя *b* и коефициенти в знаменателя *a*. Функцията връща вектора на честотната характеристика с *n* точки в *h* и съответния вектор на ъгловата честота *w*. |
| **impz** | `[h,t] = impz(b,a)` | Връща импулсния спектър на зададения цифров филтър. Задайте цифров филтър с коефициенти в числителя *b* и коефициенти в знаменателя *a*. Функцията избира броя на семплите и връща коефициентите на спектъра в *h* и времената за семплиране в *t*. |
| **grpdelay** | `[gd,w] = grpdelay(b,a,n)` | Връща груповото закъснение на зададения цифров филтър. Задайте цифров филтър с коефициенти в числителя *b* и коефициенти в знаменателя *a*. Функцията връща вектора на груповото закъснение с *n* точки в *gd* и съответния вектор на ъгловата честота *w*. |
| **abs** | `Y = abs(X)` | Връща абсолютната стойност на всеки елемент във входа X. |
| **angle** | `theta = angle(z)` | Връща фазовия ъгъл в интервала [-π,π] за всеки елемент от комплексен масив *z*. Ъглите в *theta* са такива, че $z = abs(z).* exp(i * theta)$. |
| **unwrap** | `Q = unwrap(P)` | Разгъва радианните фазови ъгли във вектор P. Винаги, когато скокът между последователни ъгли е по-голям или равен на π радиана, разгъването измества ъглите, като добавя кратни на ±2π, докато скокът стане по-малък от π. Ако P е матрица, разгъването работи по колони. Ако P е многомерен масив, разгъването работи върху първото измерение, чийто размер е по-голям от 1. |
| **exp** | `Y = exp(X)` | връща експоненциалната стойност ex за всеки елемент в масив X. За комплексни елементи z = x + iy, връща комплексната експонента $e^z = e^x(\cos y + i \sin y)$. Използвайте `expm`, за да изчислите матрична експонента. |

| Функция | Синтаксис | Описание |
|---|---|---|
| **fft** | `Y = fft(X)` | Изчислява дискретното преобразуване на Фурие (DFT) на X, използвайки алгоритъм за бързо преобразуване на Фурие (FFT). Y е със същия размер като X:<br>• Ако X е вектор, тогава `fft(X)` връща преобразуването на Фурие на вектора.<br>• Ако X е матрица, тогава `fft(X)` третира колоните на X като вектори и връща преобразуването на Фурие за всяка колона.<br>• Ако X е многомерен масив, тогава `fft(X)` третира стойностите по първото измерение на масива, чийто размер не е равен на 1, като вектори и връща преобразуването на Фурие за всеки вектор. |
| | `Y = fft(X,n)` | Връща n-точково DFT:<br>• Ако X е вектор и дължината на X е по-малка от n, тогава X се допълва с нули в края до дължина n.<br>• Ако X е вектор и дължината на X е по-голяма от n, тогава X се отрязва до дължина n.<br>• Ако X е матрица, тогава всяка колона се третира както във векторния случай.<br>• Ако X е многомерен масив, тогава първото измерение на масива, чийто размер не е равен на 1, се третира както във векторния случай. |
| **fftshift** | `Y = fftshift(X)` | Пренарежда Фурие трансформация X чрез изместване на компонента с нулева честота към центъра на масива.<br>• Ако X е вектор, тогава `fftshift` разменя лявата и дясната половина на X.<br>• Ако X е матрица, тогава `fftshift` разменя първия квадрант на X с третия, а втория квадрант с четвъртия.<br>• Ако X е многомерен масив, тогава `fftshift` разменя полупространствата на X по всяко измерение. |
| **rectwin** | `w = rectwin(L)` | Връща правоъгълен прозорец с дължина L. |
| **hann** | `w = hann(L)` | Връща L-точков симетричен прозорец на Хан. |
| **hamming** | `w = hamming(L)` | Връща L-точков симетричен прозорец на Хеминг. |
| **blackman** | `w = blackman(L)` | Връща L-точков симетричен прозорец на Блекман. |
| **downsample** | `y = downsample(x,n)` | Намалява честотата на семплиране на x, като запазва първата семпла и след това всяка n-та семпла след първата. Ако x е матрица, функцията третира всяка колона като отделна последователност. |
| **upsample** | `y = upsample(x,n)` | Увеличава честотата на семплиране на x чрез вмъкване на n – 1 нули между семплите. Ако x е матрица, функцията третира всяка колона като отделна последователност. |
| **decimate** | `y = decimate(x,r)` | Намалява честотата на семплиране на входния сигнал *x* с коефициент *r*. Дециматираният сигнал *y* се скъсява с коефициент *r*, така че `length(y) = ceil(length(x)/r)`. По подразбиране `decimate` използва нискочестотен филтър с безкраен импулсен спектър (IIR) тип I на Чебишев от порядък 8. |
| | `y = decimate(x,r,n)` | Използва филтър на Чебишев от порядък n. |

| Функция | Синтаксис | Описание |
|---|---|---|
| **interp** | `y = interp(x,r)` | Увеличава честотата на дискретизация на входния сигнал x с коефициент r. |
| | `y = interp(x,r,n,cutoff)` | Задава две допълнителни стойности:<br>• *n* е половината от броя на оригиналните стойности на извадката, използвани за интерполиране на разширения сигнал.<br>• `cutoff` е нормализираната гранична честота на входния сигнал, зададена като част от честотата на Найкуист. |
| **butter** | `[b,a] = butter(n,Wn)` | Проектира нискочестотен цифров филтър на Бътруърт от n-ти ред с нормализирана гранична честота Wn. Функцията на Бътър връща коефициентите в числителя и знаменателя на предавателната функция на филтъра |
| | `[b,a] = butter(n,Wn,ftype)` | Проектира нискочестотен, високочестотен, лентов или задържащ цифров филтър на Бътруърт, в зависимост от стойността на `ftype` и броя на елементите на *Wn*. Получените проекти на лентов и лентов филтър са от порядъка 2n. |
| **buttap** | `[z,p,k] = buttap(n)` | Връща полюсите и коефициента на усилване на прототип на аналогов нискочестотен филтър от ред n на Бътруърт. |
| **cheby1** | `[b,a] = cheby1(n,Rp,Wp)` | Проектира нискочестотен цифров филтър тип I на Чебишев от n-ти ред с нормализирана честота на ръба на лентата за пропускане *Wp* и *Rp* децибела от пик до пик пулсации на лентата за пропускане. Функцията `cheby1` връща коефициентите в числителя и знаменателя на предавателната функция на филтъра. |
| | `[b,a] = cheby1(n,Rp,Wp,fType)` | Проектира нискочестотен, високочестотен, лентов или задържащ цифров филтър Чебишев тип I, в зависимост от стойността на `ftype` и броя на елементите на *Wp*. Получените лентови и лентови препятстващи проекти са от порядък 2n. |
| **cheby2** | `[b,a] = cheby2(n,Rs,Ws)` | Проектира нискочестотен цифров филтър Чебишев тип II от n-ти ред с нормализирана честота на ръба на лентата на спиране *Ws* и *Rs* децибела затихване в лентата на спиране надолу от пиковата стойност на лентата на пропускане. Функцията `cheby2` връща коефициентите в числителя и знаменателя на предавателната функция на филтъра. |
| | `[b,a] = cheby2(n,Rs,Ws,fType)` | Проектира нискочестотен, високочестотен, лентов или задържащ препятстващ цифров филтър Чебишев тип II, в зависимост от стойността на `ftype` и броя на елементите на *Ws*. Получените лентови и лентови препятстващи проекти са от порядък 2n. |
| **ellip** | `[b,a] = ellip(n,Rp,Rs,Wp)` | Проектира нискочестотен цифров елиптичен филтър от n-ти ред с нормализирана честота на ръба на лентата на пропускане *Wp*. Полученият филтър има *Rp* децибела пулсации от пик до пик в лентата на пропускане и *Rs* децибела затихване в лентата на спиране спрямо пиковата стойност на лентата на пропускане. Функцията `ellip` връща коефициентите в числителя и знаменателя на предавателната функция на филтъра. |
| | `[b,a] = ellip(n,Rp,Rs,Wp,fType)` | Проектира нискочестотен, високочестотен, лентов или задържащ елиптичен филтър, в зависимост от стойността на `ftype` и броя на елементите на *Wp*. |

| Функция | Синтаксис | Описание |
|---|---|---|
| | | Получените лентови и лентови елиптични филтри са от порядъка на 2n. |
| **freqz** | `[h,w] = freqz(b,a,n)` | Връща честотната характеристика на зададения цифров филтър. Задайте цифров филтър с коефициенти в числителя *b* и коефициенти в знаменателя *a*. Функцията връща вектора на честотната характеристика с *n* точки в *h* и съответния вектор на ъгловата честота *w*. |
| **impz** | `[h,t] = impz(b,a)` | Връща импулсния отклик на зададения цифров филтър. Задайте цифров филтър с коефициенти в числителя *b* и коефициенти в знаменателя *a*. Функцията избира броя на семплите и връща коефициентите на отклик в *h* и времената за семплиране в *t*. |
| **grpdelay** | `[gd,w] = grpdelay(b,a,n)` | Връща груповото закъснение на зададения цифров филтър. Задайте цифров филтър с коефициенти в числителя *b* и коефициенти в знаменателя *a*. Функцията връща вектора на груповото закъснение с *n* точки в *gd* и съответния вектор на ъгловата честота *w*. |
| **filter** | `y = filter(b,a,x)` | Филтрира входните данни *x*, използвайки рационална предавателна функция, дефинирана от коефициентите в числителя и знаменателя *b* и *a*.<br>• Ако *a*(1) не е равно на 1, тогава `filter` нормализира коефициентите на филтъра чрез *a*(1). Следователно, *a*(1) трябва да е различно от нула.<br>• Ако *x* е вектор, тогава `filter` връща филтрираните данни като вектор със същия размер като *x*.<br>• Ако *x* е матрица, тогава `filter` действа по първото измерение и връща филтрираните данни за всяка колона.<br>• Ако *x* е многомерен масив, тогава `filter` действа по първото измерение на масива, чийто размер не е равен на 1. |
| **filtfilt** | `y = filtfilt(b,a,x)` | Извършва цифрово филтриране с нулева фаза, като обработва входните данни *x* както в права, така и в обратна посока. След филтриране на данните в права посока, функцията съпоставя началните условия, за да минимизира преходните процеси при стартиране и завършване, обръща филтрираната последователност и пуска обратната последователност обратно през филтъра. Резултатът има следните характеристики:<br>• Нулево фазово изкривяване<br>• Предавателна функция на филтъра, равна на квадрата на величината на оригиналната предавателна функция на филтъра<br>• Ред на филтъра, който е два пъти по-голям от реда на филтъра, зададен от *b* и *a*<br>Не използвайте `filtfilt` с диференциращи и Хилбертови FIR филтри, тъй като работата на тези филтри зависи силно от фазовата им характеристика. |
| **fir1** | `b = fir1(n,Wn)` | Използва прозорец на Хеминг за проектиране на нискочестотен, лентов или многолентов FIR филтър от n-ти ред с линейна фаза. Типът на филтъра зависи от броя на елементите на Wn. |

| Функция | Синтаксис | Описание |
|---|---|---|
| | `b = fir1(n,Wn,ftype)` | Проектира нискочестотен, високочестотен, лентов, режекторен или многолентов филтър, в зависимост от стойността на `ftype` и броя на елементите на Wn. |
| | `b = fir1(___,window)` | проектира филтъра, използвайки вектора, зададен в window, и някой от аргументите от предишните синтаксиси. |
| **fir2** | `b = fir2(n,f,m)` | Връща FIR филтър от n-ти ред с честотно-амплитудни характеристики, зададени във векторите $f$ и $m$. Функцията линейно интерполира желаната честотна характеристика върху плътна мрежа и след това използва обратното преобразуване на Фурие и прозорец на Хеминг, за да получи коефициентите на филтъра. |
| **firpm** | `b = firpm(n,f,a)` | Връща вектор-ред $b$, съдържащ n+1 коефициента на FIR филтър от ред $n$. Честотните и амплитудните характеристики на получения филтър съвпадат с тези, дадени от векторите $f$ и $a$. |
| **firls** | `b = firls(n,f,a)` | Връща вектор-ред $b$, съдържащ $n+1$ коефициента на FIR филтър от ред $n$. Честотните и амплитудните характеристики на получения филтър съвпадат с тези, дадени от векторите $f$ и $a$. |
| | `b = firls(___,ftype)` | Проектира асиметрични (нечетни) филтри, където `ftype` определя филтъра като диференциатор или Хилбертов трансформатор. Можете да използвате `ftype` с всеки от предишните входни синтаксиси. |
| **conj** | `Zc = conj(Z)` | Връща комплексно спрегнатото число на всеки елемент в Z. |
| **logspace** | `y = logspace(a,b)` | Генерира вектор-ред $y$ от 50 логаритмично разположени точки между десетилетията $10^a$ и $10^b$. Функцията `logspace` е особено полезна за създаване на честотни вектори. Функцията е логаритмичен еквивалент на `linspace` и оператора „:“. |
| **poly** | `p = poly(r)` | Връща коефициентите на полинома, чиито корени са елементите на $r$ ($r$ е вектор). |
| **loglog** | `loglog(X,Y)` | Начертава x- и y-координати, използвайки логаритмична скала с основа 10 по оста x и оста y.<br>• За да начертаете набор от координати, свързани с отсечки, задайте X и Y като вектори с еднаква дължина.<br>• За да начертаете множество набори от координати по един и същ набор от оси, задайте поне една от X или Y като матрица. |
| **mod** | `b = mod(a,m)` | Връща остатъка след деление на $a$ с $m$, където $a$ е делимото, а $m$ е делител. Тази функция често се нарича операция по модул, която може да се изрази като $b = a - m.* floor(a./m)$. Функцията `mod` следва конвенцията, че `mod(a,0)` връща $a$. |
| **imread** | `I = imread('filename')` | Зарежда цифрово изображение от графичен файл и го записва като масив в работното пространство. |
| **imshow** | `imshow(I);`<br>`imshow(I,[])` | Визуализира grayscale, RGB или бинарно изображение. При [] диапазонът за визуализация се определя автоматично от данните. |
| **class** | `C = class(I)` | Връща типа на данните, например uint8, double или logical. |

| Функция | Синтаксис | Описание |
|---|---|---|
| **double** | `J = double(I)` | Преобразува числовите стойности към тип double, без автоматично нормализиране на диапазона. |
| **im2double** | `J = im2double(I)` | Преобразува изображението към double; при стандартни integer изображения стойностите се преобразуват в диапазона [0,1]. |
| **rgb2gray** | `J = rgb2gray(RGB)` | Преобразува RGB изображение в grayscale изображение. |
| **min** | `m = min(I(:))` | Определя минималната стойност на пикселите в изображението. |
| **max** | `m = max(I(:))` | Определя максималната стойност на пикселите в изображението. |
| **imresize** | `J = imresize(I,scale)` | Променя пространствения размер на изображението чрез зададен коефициент на мащабиране. |
| **imcrop** | `J = imcrop(I,rect)` | Изрязва правоъгълна област от изображението, зададена чрез [xmin ymin width height]. |
| **imhist()** | `imhist(I); [counts,x] = imhist(I)` | Изчислява и/или визуализира хистограмата на grayscale изображение. |
| **stretchlim** | `limits = stretchlim(I)` | Определя подходящи долна и горна граница на интензитета за разтягане на контраста. |
| **imadjust** | `J = imadjust(I); J = imadjust(I,[low high],[])` | Преобразува стойностите на интензитета и се използва за корекция на яркостта и подобряване на контраста. |
| **histeq** | `J = histeq(I)` | Подобрява глобалния контраст чрез хистограмна еквализация. |
| **imnoise** | `J = imnoise(I,'gaussian',m,v)` | Добавя моделиран шум към изображението. Може да генерира Gaussian, salt-and-pepper и други видове шум. |
| **fspecial** | `h = fspecial('average',[m n])` | Създава предварително дефинирана двумерна филтрираща маска. Използва се например за усредняващи и Гаусови филтри. |
| **conv2** | `C = conv2(A,B,'same')` | Изчислява двумерната дискретна конволюция между две матрици. Опцията 'same' връща резултат с размерите на първия входен масив. |
| **imfilter** | `J = imfilter(I,h,'replicate')` | Филтрира изображението с двумерна маска h. 'replicate' определя начина за обработване на границите. |
| **medfilt2** | `J = medfilt2(I,[m n])` | Извършва двумерно медианно филтриране. Особено ефективна е при импулсен шум. |
| **fft2** | `F = fft2(I)` | Изчислява двумерното бързо преобразуване на Фурие на изображението. |
| **fftshift** | `Fc = fftshift(F)` | Пренарежда спектъра, така че нулевата пространствена честота да бъде разположена в центъра. |
| **ifftshift** | `F = ifftshift(Fc)` | Възстановява подреждането на спектъра преди прилагане на обратното преобразуване. |
| **ifft2** | `I = ifft2(F)` | Изчислява обратното двумерно преобразуване на Фурие.abs |
| **real** | `R = real(Z)` | Извлича реалната част на комплексен масив. |
| **log** | `Y = log(X)` | Изчислява естествения логаритъм. При спектралния анализ се използва за компресиране на динамичния диапазон. |
| **sqrt** | `Y = sqrt(X)` | Изчислява квадратен корен. Използва се например за определяне на радиално разстояние в честотната равнина. |

| Функция | Синтаксис | Описание |
|---|---|---|
| **meshgrid** | `[X,Y] = meshgrid(x,y)` | Създава двумерни координатни матрици. При честотно филтриране се използва за конструиране на двумерни честотни маски. |
| **imagesc** | `imagesc(A)` | Визуализира стойностите на матрица като изображение, като ги мащабира спрямо текущата цветова карта. |
| **colormap** | `colormap(gray)` | Задава цветовата карта на текущата фигура. |
| **colorbar** | `colorbar` | Добавя цветова скала към визуализацията. |
| **axis** | `axis image` | Управлява вида и мащаба на координатните оси. axis image задава еднакъв мащаб по двете пространствени оси. |
| **edge** | `BW = edge(I,'method')` | Открива контури в grayscale или бинарно изображение. Поддържа методи като 'sobel', 'prewitt', 'roberts' и 'canny'. |
| **graythresh** | `T = graythresh(I)` | Определя глобален праг за бинаризация чрез метода на Otsu. Прагът е нормализиран в диапазона [0,1]. |
| **im2bw** | `BW = im2bw(I,T)` | Преобразува изображението в бинарно чрез зададен праг T. |
| **strel** | `SE = strel('disk',r)` | Създава структуриращ елемент за морфологична обработка с определена форма и размер. |
| **imerode** | `J = imerode(I,SE)` | Извършва морфологична ерозия. При бинарни изображения води до свиване на обектите. |
| **imdilate** | `J = imdilate(I,SE)` | Извършва морфологична дилатация и разширява обектите в бинарното изображение. |
| **imopen** | `J = imopen(I,SE)` | Извършва морфологично отваряне – ерозия, последвана от дилатация. Подходящо е за отстраняване на малки структури. |
| **imclose** | `J = imclose(I,SE)` | Извършва морфологично затваряне – дилатация, последвана от ерозия. Подходящо е за затваряне на малки празнини и прекъсвания. |
| **imsubtract** | `J = imsubtract(A,B)` | Извършва поелементно изваждане на две изображения. Може да се използва за отстраняване на оценения фон. |
| **bwareaopen** | `BW2 = bwareaopen(BW,P)` | Премахва от бинарното изображение всички свързани компоненти с площ под P пиксела. |
| **bwlabel** | `[L,num] = bwlabel(BW,8)` | Етикетира свързаните компоненти в двумерно бинарно изображение и определя броя им. |
| **label2rgb** | `RGB = label2rgb(L)` | Преобразува матрица с етикети в RGB изображение, позволяващо визуално разграничаване на отделните обекти. |
| **regionprops** | `S = regionprops(L,'Area', 'Centroid','BoundingBox')` | Измерва геометрични и други характеристики на сегментираните области – площ, център, ограничителен правоъгълник и др. |
| **rectangle** | `rectangle('Position',position)` | Изчертава правоъгълник върху текущите координатни оси. Използва се за ограждане на открити обекти. |