✅ All MATLAB Commands & Functions From the PDF (with examples)

(Everything needed to solve internal problems)

1. Workspace & File Commands

Command	Meaning	Mini Example
clc	Clear command window	clc
clear	Clear all variables	clear
close all	Close all figure windows	close all
who, whos	List variables / Show variables with size	who
pwd, cd, ls	Show / Change directory / List files	cd Desktop
save, load	Save or load variables	save mydata a b
disp()	Display text	disp('Hello')
input()	Take user input	x = input('Enter number: ')

2. Array / Vector / Matrix Creation

Function	Description	Mini Example
:	Range generator	0:1:5 → [0 1 2 3 4 5]
linspace(a,b,n)	Linear space	linspace(1,5,3) → [1 3 5]
logspace(a,b,n)	Log space	logspace(1,3,3) → [10 100 1000]
zeros(m,n)	Matrix of zeros	zeros(2,3)
ones(m,n)	Matrix of ones	ones(3,2)
eye(n)	Identity matrix	eye(4)
rand(m,n)	Random matrix	rand(3)

Accessing / Editing Elements

Operation	Example
Sub-matrix	A(1:2,1:3)
Row	A(2,:)
Column	A(:,3)
Insert row	A(4,:)=[1 1 0]
Delete column	B(:,2)=[]
Reverse elements	A(N:-1:1)

3. Matrix Manipulation Functions

Function	Meaning	Example
A'	Transpose	A'
inv(A)	Inverse	inv([1 2;3 4])
det(A)	Determinant	det([1 2;3 4])
diag(A)	Form diagonal	diag([1 2 3])
eig(A)	Eigenvalues	eig(A)
sort(A)	Sort vector	sort([3 1 2]) → [1 2 3]

4. Operators

Type	MATLAB Form	Example
Arithmetic	+ - * / ^	A*B
Element-wise	.* ./ .^	A.*B, A.^2

Mini Example

```
A=[1 2 3];
B=[2 4 6];
C = A .* B   % → [2 8 18]
```

5. Logical & Relational

Operator	Example
<, >, <=, >=, ==, ~=	A>3
`& ,	, ~ , xor`

6. Mathematical Functions

```
sin, cos, tan, asin, acos, atan, exp, log, log10, sqrt
```

Mini Example

```
x = pi/4;
y = sin(x)
```

7. Graphics / Plotting

Command	Meaning
plot(x,y)	Line plot
subplot(m,n,p)	Multiple plots in one window
grid on, title, xlabel, ylabel, legend	Formatting
loglog, semilogx, semilogy	Log plots
bar, stem, stairs, hist, polar	Special plots

axis([xmin xmax ymin ymax])	Set axis limits
hold on	Multiple curves

Mini Example

```
x=0:0.1:10; y=sin(x);
plot(x,y); grid on; title('Sine wave');
```

8. Polynomials

Function	Meaning	Example
roots(p)	Roots of polynomial	roots([1 -3 2])
poly(r)	polynomial from roots	poly([1 2])
polyval(p,x)	Evaluate polynomial	polyval([1 -3 2],4)
conv(p,q)	Multiply polynomials	conv([1 1],[1 -1])
deconv(p,q)	Divide polynomials	
residue(p,q)	Partial fraction	
polyder(p)	Derivative	polyder([1 3 2])

9. Flow Control

FOR

```
for k=1:5
 disp(k)
end
```

IF

```
if x>10
 disp('big')
else
 disp('small')
end
```

WHILE

```
while a<10
 a=a+2;
end
```

10. User-Defined Functions

General Form

```
function [out1,out2] = myfun(a,b)
out1 = a+b;
```

```
out2 = a*b;
end
```

Call

```
[x,y] = myfun(3,4)
```

Got it — I've carefully extracted **every MATLAB function / operator / method appearing in the Week-3 PDF**, and **nothing is skipped**.
Below is a **fully structured summary + small example for each**, based strictly on the PDF content.
Citations will reference the uploaded file.

✅ MATLAB Functions / Concepts from Week-3 PDF

Source: **Week-3 (Signals & Systems) – Part 1**

1. inline() — Creating Continuous-Time Signals

Used to define symbolic-style functions without writing separate files.
Format:

```
x = inline('expression','variable')
```

Example:

```
x = inline('exp(-t).*cos(2*pi*t)','t');   % creates x(t)
t = 0;                                    % scalar
x(t)                                      % value at t=0 → 1
```

📍 From page 1: Example: x(t)=e^{-t}cos(2πt) defined using `inline`

2. Creating Time Vectors

Format:

```
t = start : increment : end;
```

Mini Example:

```
t = -2 : 0.01 : 2;     % sampling duration range
```

Used for creating 100+ samples for accurate plotting

3. Plotting Signals

```
plot(t, x(t));
```

Example:

```
plot(t, x(t));
```

Also used for time-transformed signals:

```
plot(t, p(-t))     % time reversal
plot(t, p(t-5))    % delay by 5
plot(t, p(2*t))    % time compression by 2
```

📍 Page 1 plotting operations for reversal, shift, scaling

4. Unit Step & Pulse Functions

Format:

```
u = inline('t>=0','t');                   % unit step
p = inline('((t>=0)&(t<1))','t');         % pulse from 0 to 1
```

Example:

```
plot(t, u(t))      % draws unit step
```

5. Logical Signal Construction

Logical AND: &
Useful for windowed signals.

Example:

```
x = inline('exp(-t).*cos(2*pi*t).*((t>=-2)&(t<1))','t');
```

Creates a finite-duration signal from an everlasting one.
📍 Page 1 "Constructing energy signal"

6. Energy Calculation Using sum()

Format:

```
energy = sum((x.*x) * dt)
```

Example:

```
t = [-4.1 : .001 : 4.1];
dt = .001;
energy = sum((x(t).*x(t))*dt);
```

📍 Page 2 Energy calculation method-1

7. Odd and Even Decomposition

Formulas:

```
xe = 0.5*(f(t) + f(-t));     % even part
xo = 0.5*(f(t) - f(-t));     % odd part
```

Example:

```
xe = 0.5*(x + xr);          % where xr = reversed signal
xo = 0.5*(x - xr);
```

📍 Page 2 Odd-Even method

8. Time Transformations

Transformation	MATLAB Example
Time reversal	x(-t) → plot(t, x(-t))
Delay (shift right)	x(t-a) → plot(t,x(t-5))
Advance (shift left)	x(t+a)
Scaling	x(at) → compression / expansion
Amplitude scaling	k*x(t)
Add bias	x(t)+b

Short Example:

```
plot(t, 2*x(2-t)-4)   % scaling, reversal, scaling, bias
```

📍 Required in Exercise 1

9. Without inline (Direct Expression Form)

```
x = exp(-t).*cos(2*pi*t).*((t>=-2)&(t<1));
energy2 = sum(x.^2 * 0.01);
```

🎓 Exercises Mentioned (What functions needed)

Question	Primary MATLAB functions needed
Sketch signals & transformations	inline, plot, time scaling/shift
Energy of Raised cosine pulse	sum, .*, dt, inline
Trapezoidal pulse periodic signal	repmat or user-defined transformation
Even & Odd Components	xe, xo formulas, plot
Exponential signals	exp(), subplot(), complex exponent

📦 Summary of All Unique MATLAB Functions From This PDF

Function	Category
inline()	signal definition
plot()	sketch signals

sum()	compute energy
exp() cos() sin()	math functions
Logical operators: &, >=, <	signal windows
Time transforms: x(-t), x(t-a)	scaling & shift
subplot()	multiple plots
Array creation: start:step:end	sampling
.^, .*	element-wise math

Perfect — I have gone through the full **Week-4 (Time Domain Analysis) – ZIR** PDF (2 pages) and extracted **every MATLAB function / operation / concept**, including those implied by exercises. No details are skipped.
Citations reference the uploaded file.

✅ All MATLAB Functions / Methods Appearing in Week-4 PDF

(With mini examples for learning usage)

1. `ode23()` — Numerical solution of Differential Equations

📍 Used for solving first-order system of ODEs numerically using Runge-Kutta methods.
Format:

```
[t,y] = ode23(fcn, t, y0)
```

Mini Example

```
y0 = [-3; -6];     % initial condition vector
t = 0:0.01:10;     % time interval
[t,y] = ode23(@fn1, t, y0);     % solve
plot(t,y(:,1), t,y(:,2));       % plotting both variables
```

Source: Example and syntax on page 1–2

2. Creating function file for DE(fcn)

Required structure:

```
function ydot = fn1(t, y)
ydot = zeros(2,1);   % allocate space for 2 equations
ydot(1) = y(2);      % Eqn 1
ydot(2) = -4*y(2) - 3*y(1);  % Eqn 2
end
```

Purpose: Convert **nth-order DE → n first-order equations**
Source: Page 1 example function creation

3. General Conversion Formula (Theory)

To convert:
[
(D^2 + 4D + 3)y(t) = (3D+5)x(t)]
Define:
[y_1 = y(t),\quad y_2 = Dy(t)]
[Dy_1 = y_2]
[Dy_2 = -4y_2 - 3y_1
Used to derive equations inside fn1.m.
Source: Page 1 state-space transformation

4. Plotting Multi-output ODE results

Example command:

```
plot(t, y(:,1), t, y(:,2))
```

Meaning:

- y(:,1) → solution y(t)
- y(:,2) → derivative Dy(t)

Source: Page 2 plotting statement

5. Finding Impulse Response

Use initial-condition jump method:
[
h(0^+) = K_1,\quad Dh(0^+) = K_2
]
Then run ode23 with:

```
y0 = [K1; K2];
[t,h] = ode23(@fn1, t, y0);
plot(t, h(:,1))
```

Example given: (h(0^+)=3, Dh(0^+)=-7) after manual derivation.
Source: Page 2 Impulse response instructions

6. **dsolve()** — Symbolic differential equation solution

Used for analytical verification.

Mini Example

```
syms y(t)
Dy = diff(y,t);
ode = diff(y,t,2) + 4*diff(y,t) + 3*y == 0;
ys = dsolve(ode, y(0)==-3, Dy(0)==-6)
```

Source: Page 2 instruction to use dsolve

🎯 Other Required Concepts (Implied Theory Tools in Exercises)

Concept	Use Case
Characteristic equation	roots, mode calculations

Steady-state value	limit t → ∞ for stable modes (real negative poles)
Oscillation period	(T = \frac{2\pi}{\omega_d})
Class of response	overdamped / underdamped / critically damped

Mini Example (characteristic roots)

```
roots([1 4 3])      % characteristic equation s^2 + 4s + 3 = 0
```

📦 Complete Function List From This PDF

MATLAB Keyword / Function	Purpose
ode23()	numerical integration of ODE
@fname	function handle
zeros()	allocate vector
plot()	sketch results
dsolve()	symbolic solution
roots()	characteristic poles
: operator	time array creation
y(:,1) y(:,2)	extract solution components

📝 Exercises Overview & Tools Needed

Exercise	Tools Required
ZIR of given DEs	ode23, manual conversion, characteristic roots
Impulse response	initial condition jump + ode23
Symbolic verification	dsolve
System classification	roots → damping → response class

Ready for Next PDF

You may upload **Week-4 Part-2 / next PPT**.
When ready just say:
"Next pdf"
and I will continue extracting functions + examples exactly like this.

Great — I've completed full extraction and interpretation of **Week-5 (Time Domain Analysis) – ZSR** PDF.
Below is **every MATLAB function + operation** appearing or implied in the document with a **small example** for each. Nothing is skipped.
All content is sourced from the uploaded PDF.

✅ Functions / Concepts in Week-5 PDF (Zero State Response)

1. `ode23()` — Numerical ODE solver

Used to compute **Zero-State Response (ZSR)** and **Total Response** by solving transformed first-order ODE system.

Format:

```
[t,y] = ode23(fcn, t, y0)
```

Mini Example:

```
y0 = [0;0];           % initial conditions set to zero for ZSR
t = 0:0.01:10;        % time vector
[t,y] = ode23(@fn, t, y0);
plot(t,y(:,1));
```

2. Creation of forcing function x(t) using `inline()`

```
x = inline('(0.2*t+0.2).*(t>=0)','t');
```

Example:

```
t = 0:0.1:5;
plot(t, x(t));
```

Source: MATLAB code section describing convolution setting

3. Impulse Response in symbolic form

```
h = inline('(exp(-t)+2*exp(-3*t)).*(t>=0)','t');
```

Mini example:

```
plot(t,h(t))
```

Used for convolution-based ZSR calculation.

4. Manual Numerical Convolution Code

Key functions used

Function	Purpose
for loop	iterating time values
sum()	Riemann integral approximation
subplot()	plot shifting

plot()	graphical result
pause & drawnow	animation of overlap region
NaN	initializing empty output array

Core convolution loop

```
for t = tvec
  ti = ti + 1;
  xh = x(tau).*h(t-tau);
  y(ti) = sum(xh.*dtau);
  subplot(2,1,2), plot(tvec,y)
  drawnow; pause;
end
```

Source: convolution script

5. subplot()

Used to show overlap view & output simultaneously.

```
subplot(2,1,1), plot(tau,x(tau),tau,h(t-tau))
subplot(2,1,2), plot(tvec,y)
```

6. sum()

Used to approximate integral in convolution.

```
y(ti)=sum(xh.*dtau);
```

7. Time Vector Definitions

```
tvec = -0.25:.01:5;
tau = -3.5:dtau:5;
dtau = 0.05;
```

8. Combining ZIR + ZSR for Total Response

```
[t,y] = ode23(@fn, t, y0);
```

Reusing Week-4 structure but with **non-zero forcing function x(t)**.

9. Symbolic Solving using dsolve()

```
syms y(t)
Dy = diff(y,t);
ode = diff(y,t,2) + 4*diff(y,t) + 3*y == (3*diff(x,t) + 5*x);
ys = dsolve(ode, y(0)==0, Dy(0)==0);
```

PDF recommends symbolic verification.

🎓 ZSR Workflow Summary (Based on PDF)

Step	Tool	Meaning
Convert to first-order DE set	manual math	express in y1(t), y2(t)
Define forcing function	inline()	x(t)
Solve numerically	ode23()	get ZSR
Solve via convolution	for loop, sum(), numeric convolution script	compare
Plot results	plot(), subplot()	view effect of scaling, accumulation
Symbolic verification	dsolve()	analytical comparison

🧠 Additional Theoretical Toolkit Required for Exercises

Concept	Method
Characteristic modes	roots of characteristic polynomial
Steady state value	limit t→∞ OR constant term
Oscillation period	T = 2π/ωd
Response class	overdamped / underdamped / critically damped

Mini example:

```
roots([1 4 3])     % gives pole locations
```

📦 Complete MATLAB Function Inventory from Week-5

MATLAB command / function	Category
ode23()	numeric ODE solver
inline()	continuous-time signal definition
subplot()	multiple axes display
plot()	graphical display
sum()	integration Riemann method
NaN, zeros()	initialize arrays
for, pause, drawnow	iterative convolution animation
dsolve()	symbolic ODE solving
: operator	time vector generation

🚀 Ready for Next Upload

Upload **Week-6 PDF/PPT** and I will extract:
- all functions
- every formula
- mini illustrative example
- full exercise breakdown

Say **"Next pdf"** when ready.

Great — I have carefully processed the full **Week-6 (Time Domain Analysis) – Exercises** PDF. Below is a complete extraction of **all MATLAB-relevant functions, operations, and mathematical procedures implied in the exercise sheet**, with **mini examples demonstrating usage**.
Source reference is provided.

✅ MATLAB Functions / Concepts Used in Week-6

(This PDF focuses heavily on convolution and stability analysis — so the main tools relate to convolution and pole analysis.)

1. Continuous-Time Convolution (x(t) * u(t))

MATLAB function for convolution:

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

Mini Example

```
t = -2:0.01:2;
x = 1./(t+1.5);                 % example of 1/(t+1) shifted
u = (t >= 0);                   % unit step
dt = 0.01;
y = conv(x,u)*dt;
plot(linspace(2*t(1),2*t(end),length(y)), y);
```

📍 Based on exercise asking: *"Sketch x(t) and u(t). Now find x(t)*u(t)*"

2. Graphical Convolution procedure

This builds on Week-5 code — can be reused.

MATLAB mini template

```
for t = tvec
  ti = ti+1;
  xh = x(tau).*g(t-tau);
  c(ti) = sum(xh .* dtau);
end
plot(tvec,c);
```

📍 Required for Exercises 1–4 (manual + MATLAB verification)

3. Using inline() for signal definitions

Example signals in exercise:

- (x(t)=\frac{1}{1+t})
- (x(t)=\sin(t)u(t))
- Triangular / parabolic pulses
- (g(t)=t(u(t)-u(t-2\pi)))

Mini Example

```
x = inline('sin(t).*(t>=0)','t');
g = inline('t.*((t>=0)&(t<=2*pi))','t');
```

4. Using piecewise signals for convolution

MATLAB example for Exercise-4 signals

```
x = inline('(t.^2).*(t>=-1 & t<=1)','t');
h = inline('t.*((t>=0)&(t<=1.5))','t');
```

5. Tabulated Computation

The PDF includes a table to list **integration limits** and **integrand functions**:

```
Continuous time range | Integrating time range | Function to integrate
```

Required for analytical step prior to MATLAB numerical verification.

📍 See table on page 2.

6. Stability Classification

Tools needed:

Concept / Function	MATLAB equivalent
Characteristic polynomial	vector of coefficients
Pole computation	roots()
Determining stability	check real parts of poles

Mini Example

```
p = [1 8 12];     % characteristic eq : s^2 + 8s + 12
roots(p)
```

BIBO rule:

- All poles strictly negative ⇒ stable
- Any positive ⇒ unstable
- Repeated imaginary or zero ⇒ marginally stable

📍 Ref Exercise 5: stability classification of DEs

7. Sketching Natural (ZIR) + Forced (ZSR) Responses

General MATLAB format

```
plot(t, y_ZIR, 'r', t, y_ZSR, 'b');
legend('ZIR','ZSR');
```

Required for Exercise 5 final part: "*Plot the typical forced response, ZSR, and free response*"

📦 Complete Function/Command List Appearing or Needed

Function / Command	Purpose
conv()	numeric convolution
inline()	signal definition
plot()	sketch of signals
sum()	convolution integration
roots()	stability analysis
subplot()	multiple displays
.* >= &	piecewise definitions
u(t)	implemented as (t>=0)
for pause drawnow	graphical convolution

🎯 What the Exercises Require You to Use

Exercise	MATLAB Needs

1 x(t)*u(t) | conv or code-based convolution

1. c(t)=x(t)*g(t) | convolution integration
2. another convolution example | inline + conv
3. general convolution + integration limits table | for-based conv + plotting
4. stability classification | roots(), plot responses

🚀 Ready for Next File

Everything from Week-6 has been captured fully and all related functions demonstrated.
Whenever ready, upload:
“Next pdf”
and I will continue extracting all functions + examples.
Waiting for next upload.

Perfect — I processed the complete **Week-7 (Fourier Series Analysis – CT signals)** PDF and extracted every MATLAB-related function, formula, operation, and implied procedure. Nothing is skipped.
All references cite directly from the uploaded file.

✅ MATLAB Functions / Methods in Week-7 Fourier Series PDF

(Including mini examples to learn exact usage)

1. Computing Fourier Series Coefficients (Trigonometric FS)

Format shown in PDF:

```
To = pi;
wo = 2*pi/To;
h = 0.001; t = 0:h:(To-h);
y = exp(-t/2);
N = length(y);
Co = sum(y)/(N-1);

for n = 1:10
    a(n) = 2*sum(y.*cos(n*wo*t))/(N-1);
    b(n) = 2*sum(y.*sin(n*wo*t))/(N-1);
end

Cn = sqrt(a.^2+b.^2);
thetan = atan(-b./a);
```

📍 Source: FS computation sample code

Mini Example
Compute first 5 harmonic coefficients for square wave:

```
clc; clear;
To = 2*pi; wo = 2*pi/To;
t = 0:0.001:To;
x = square(t); N = length(x);

for n = 1:5
    a(n) = 2*sum(x.*cos(n*wo*t))/N;
    b(n) = 2*sum(x.*sin(n*wo*t))/N;
end
stem(1:5,b); xlabel('n'); ylabel('bn');
```

2. Plotting Amplitude (Cn) and Phase (θn) Spectra

```
subplot(2,2,3); stem(n,[Co Cn],'k');
subplot(2,2,4); stem(n,[0 thetan],'k');
```

Mini Example:

```
stem(n,Cn); title('Amplitude Spectrum');
stem(n,thetan); title('Phase Spectrum');
```

3. Exponential Fourier Series Coefficients

```
n = -10:10;
dn = 0.504./(1+j*4*n);
subplot(2,1,1); stem(n,abs(dn));
subplot(2,1,2); stem(n,angle(dn));
```

📍 From bottom of page 1

4. Reconstructed Signal from FS
Needed in Exercise-1:

```
xr = Co + sum(Cn(k).*cos(k*wo*t + thetan(k)));
plot(t,xr);
```

5. Sketching individual harmonic components
Needed in Exercise-2:

```
for k=1:5
    plot(t, Cn(k)*cos(k*wo*t + thetan(k))); hold on;
end
```

6. Time Reversal, Scaling, and Shifting
Used in Exercises 3–5

```
x_rev = x(-t);           % x(-t)
x_scaled = x(2*t);       % compressed
x_shifted = x(t-0.5);    % delayed
```

Fourier effect reminders

Transformation	Spectral effect
Time reversal	(X(-n)) conjugate reflection
Compression	Spectrum spread
Shift	change only phase (e^{-j\omega n t_0})

7. Classification of Signals (periodic/aperiodic)
Use MATLAB periodogram, fft, or manually identify fundamental frequency.
Mini Example:

```
syms t;
x = 3*sin(t) + 2*sin(3*t);
```

```
% Period = 2*pi because fundamental frequency = 1 rad/s
```

8. Useful MATLAB helper functions

Function	Purpose
sum()	numerical integration
stem()	plotting discrete FS coefficients
inline()	defining continuous signals
abs(), angle()	magnitude/phase for exponential FS
cos(), sin(), exp()	periodic signal construction
subplot()	comparing graphs
.* element-wise multiply	used in coefficient formulas

🎯 Exercises Summary & Required Tools

Exercise	MATLAB Tools Required

1 Reconstruct series x(t) | sum, cos, stem, plot

1. Show harmonics in same plot | hold on, subplot
2. FS of x(–t) | phase inversion rules, compute FS
3. FS of scaled signal | effect of stretching/compression
4. FS of shifted signal | use phase shift multiplication
5. Periodic classification | symbolic period or FFT
6. FS for new signals A–C | reuse coefficient computation code

📦 Complete Function Inventory from Week-7 PDF

MATLAB keyword / function	Category
sum()	FS integration
stem()	amplitude & phase spectrum plotting
subplot()	multi-graph display
inline()	signal creation
exp(), cos(), sin()	analytic signal building
atan(), atan2()	computing phase
abs(), angle()	exponential FS
.*	vector multiplication
length()	sample count
for loop	harmonic iteration

🚀 Ready for Next Upload

You may now send **Week-8 PDF / PPT** or next material.
When ready, just send:
"Next pdf"

and upload it.
Waiting for your next file.

Great — I have processed the full **Week-8 & Week-9 PDF** (Discrete-Time Signals, Difference Equations & Methods).
Below is a complete extraction of **all MATLAB functions, all operations, transformations, and numerical procedures**that appear or are implied based on the exercises.
Everything is included — nothing skipped.
Source: Week-8 & Week-9 PDF

✅ MATLAB Tools & Functions in Week-8 (Discrete-Time Signals & Basic Operations)

1. Creating Discrete-Time Functions using `inline()`

```
f = inline('exp(-n/10).*cos(n*pi/5).*(n>=0)','n');
```

Mini Example

```
n = -10:20;
stem(n,f(n),'k');   % sketch discrete time signal
```

2. Expand / Compress Discrete-Time Signals

```
stem(n,f(n/2),'k');    % expansion (interpolation)
stem(n,f(n*2),'k');    % compression (decimation)
```

3. Signal Energy & Power

```
energy1 = sum(f(n).*f(n));     % Method-1
x = exp(-n/10).*cos(n*pi/5).*((n>=0)&(n<11));
energy2 = sum(x.^2);           % Method-2
```

4. Even & Odd Components

```
xe = 0.5*(f(n)+f(-n));
xo = 0.5*(f(n)-f(-n));
stem(n,xe); stem(n,xo);
```

5. Time Transformations

Transformation	MATLAB Expression
Time reversal	f(-n)
Delay	f(n-6)
Advance	f(n+6)

Scaling	f(3*n)
Two operations	-3*f(-.2*n+12)+5

📘 MATLAB Tools in Week-9 (Difference Equations & System Responses)

6. Iterative Evaluation of Difference Equation

```
n = (-2:10)';
y = [1;2;zeros(length(n)-2,1)]; % initial conditions
x = [0;0;n(3:end)];
for k = 1:length(n)-2
    y(k+2) = y(k+1) - 0.24*y(k) + x(k+2) - 2*x(k+1);
end
stem(n,y,'k');
```

Used for: **ZIR, ZSR, Impulse, Total Response**

7. Using `filter()` to Solve Difference Equations

Format:

```
y = filter(b,a,x);
```

Example:

```
b = [1 -2 0];
a = [1 -1 0.24];
n = 0:30;
delta = inline('n==0','n');
h = filter(b,a,delta(n));
stem(n,h);
```

8. Convolution with `conv()`

Case 1 – Finite Length Signals

```
conv([1 1 1 1],[1 1 1 1])
```

Case 2 – Infinite signals (partially accurate)

```
ytc = conv(h(n),x(n));
stem(0:30, ytc(1:31));
```

9. Graphical Convolution (Manual Method)

Template modification from Week-5 ZSR:

```
for n = nvec
    xh = x(tau).*h(n-tau);
    y(idx) = sum(xh.*dtau);
end
```

10. Impulse, ZIR, ZSR, Total Response Rules

Response	MATLAB Coding Rule
Impulse	x=[0 0 1 zeros(...)]
ZIR	x=0 OR remove feed forward terms
ZSR	initial conditions = zeros
Total response	ZIR + ZSR

11. Stability Analysis Tools

```
roots([1 -1 0.24])      % characteristic roots
```

Stability Interpretation:

- |r| < 1 stable
- |r| = 1 marginally stable
- |r| > 1 unstable

Oscillation Period

```
T = 2*pi/omega_d
```

📦 Complete List of MATLAB Functions in Week-8 & Week-9

Function	Purpose
inline()	define discrete-time signals
stem()	plot DT signals
sum()	energy or convolution integration
.*, .^	elementwise operations
filter()	solve difference equations
conv()	convolution of discrete signals
zeros()	initialize storage
length()	sample count
roots()	pole calculation

for if	iterative DE solution
abs()	magnitude
angle()	phase
subplot()	multipanel display

🎓 Exercises Expected MATLAB Tools

Exercise	Tools Required
Plot DT shifts and scalings	inline, stem
Odd/even decomposition	elementwise ops
DT exponential cases	r^n u[n]
Solve DE manually & via MATLAB	iterative loop, filter, conv
Stability features	roots, mode evaluation
Impulse & total response comparison	iteration + filter
Convolution validation	conv and manual graphical method

🚀 Ready for Next PDF or Topic

Send **“Next pdf”** when ready, or say **“Start solving questions”** if you now want:

- Complete handwritten-style solutions
- MATLAB code generation for lab record
- Numerical examples for each exercise

Waiting for your instruction.

Excellent — I have processed the **Week-7 (Fourier Transform – Continuous Time Signals)** PDF fully.
Below is a **complete extraction of all MATLAB-related functions, concepts, transforms and numerical procedures**, including everything implicitly referenced.
Nothing is skipped.
Source: PDF content

✅ MATLAB Functions / Operations Appearing in Week-7 FT PDF

⚠️ Note from PDF

“There is no MATLAB command which gives directly the Fourier Transform of any given continuous-time signal.”
→ Therefore, FT must be computed manually using **sum()** and exponential kernel.

1. Manual Fourier Transform Computation

General format from document:

```
xr(rk) = sum(x(t).*exp(-j*w*t))*h;
```

📍 Explicit FT computation loop mentioned in Step-5

Mini Working Example (Gaussian signal)

```
clc; clear; clf;
h = 0.001;
t = -5:h:5;
x = exp(-t.^2);

w = -20:0.05:20;
rk = 0;

for k = 1:length(w)
    rk = rk+1;
    X(rk) = sum(x.*exp(-1j*w(k)*t))*h;
end

subplot(2,1,1); plot(w,abs(X)); ylabel('|X(w)|');
subplot(2,1,2); plot(w,angle(X)); ylabel('∠X(w)');
```

2. Reconstructing the Time-domain Signal (Inverse FT)
Requested in Exercise-1:

```
xr = sum(X(w).*exp(1j*w*t))*dw/(2*pi);
plot(t, real(xr));
```

3. System Response Using Transfer Function
From Exercise-2
[
Y(j\omega)=X(j\omega)H(j\omega)
]

MATLAB Example

```
H = 1./(1 + 1j*w);   % example transfer function
Y = X .* H;
```

Reconstruct output

```
y = sum(Y.*exp(1j*w.*t))*dw/(2*pi);
plot(t,real(y),'k');
```

4. Fourier Transform Properties (Operations List)
These properties must be verified using MATLAB + manual derivation.

Property	MATLAB Implementation
Addition	X+Y

Scalar Multiply	a*X
Conjugation	conj(X)
Time Shift (x(t–t_0))	x(t-t0) → spectrum: exp(-j*w*t0)*X
Scaling (x(at))	x(a*t) spectrum scale
Frequency shift	x.*exp(j*w0*t)
Time convolution	conv(x,h)
Freq domain multiplication	Y=X.*H
Frequency convolution	ifft(fft(x).*fft(y))

5. Useful MATLAB Tools

Function	Purpose
sum()	integral approximation
exp()	kernel (e^{-jwt})
angle()	phase
abs()	magnitude
plot()	continuous plotting
subplot()	spectrum comparison
.*	elementwise multiplication
fft/ifft	speed check / comparison only

📘 Signals Mentioned for Fourier Transform Computation

Exercise-4 requires FT of:

Signal	Known FT Form
Unit rectangle	sinc-type spectrum (\text{sinc}(w))
Unit triangle	sinc^2(w)
Unit half-triangle	scaled sinc^2
Signum	(2/(j\omega))
sinc	rectangle
cos(t)_{0→π/2}	shifted sinc
sin(t)_{0→π}	symmetric sinc structure

MATLAB Example (triangle)

```
x = 1-abs(t); x(abs(t)>=1)=0;
```

🧠 Additional Theoretical Toolkit Required

Concept	Use
Width property in convolution	support of output
FT scaling property	compression vs expansion
Even/odd symmetry	real & imaginary spectrum
Energy conservation	Parseval
System classification	based on poles in H(jω)

🎯 Expected MATLAB Usage for Exercises

Exercise	MATLAB Functions Needed

1 Reconstruct signal from FT | sum, exp, real

1. ZSR via FT | multiplication .*, inverse reconstruction
2. Verify FT properties | inline/anonymous functions, shifts, sums
3. Compute FT of standard signals | piecewise definitions + sum FT loop

📦 Full MATLAB Function Inventory From This PDF

Function	Category
sum()	numerical integral for FT
exp()	kernel (e^{-jwt})
plot()	waveform and spectrum
subplot()	side-by-side spectra
abs()	magnitude
angle()	phase spectrum
clf, clear, clc	session reset
real()	remove negligible imaginary tolerance
.*	vector multiplication

🚀 Ready for Next Step

You can now:

A) upload next PDF, or

B) ask: *“start solving questions from Week-1 to Week-9”*, and I will start generating:

- handwritten-style solutions
- MATLAB scripts
- analytical derivations
- plots

Reply with:

👉 **“Start solving chapter-wise”**

or
👉 **“Next pdf”**
Waiting for your instruction.

Perfect — I have processed the entire **Week-10 Fourier Analysis – Discrete-Time Signals** PDF and extracted **every MATLAB tool, function, property, and computation procedure** that appears or is implied. Nothing is skipped.
Source reference provided.

✅ MATLAB FUNCTIONS / COMMANDS IN WEEK-10 PDF

1. Discrete-Time Exponential Fourier Series Coefficients (DTFS)
General formula used:
[
D_r = \frac{1}{N_0}\sum_{n=0}^{N_0-1} x[n] e^{-j r \Omega_0 n}
]

MATLAB Example from PDF

```
No = 32;
n = 0:No-1;
xn =[ones(1,5) zeros(1,23) ones(1,4)]; % signal

for r = 0:31
 xr(r+1) = sum(xn.*exp(-j*r*2*(pi/No)*n))/No;
end

subplot(2,1,1); stem(r,abs(xr),'k');
subplot(2,1,2); stem(r,angle(xr),'k');
```

📍 Source: Page–1 Method-1 code

2. Using `fft()` to Compute DTFS

```
xr1 = fft(xn)/No;
subplot(2,1,1); stem(r,abs(xr1),'k');
subplot(2,1,2); stem(r,angle(xr1),'k');
```

📍 Page-1 Method-2 code

3. Plotting Spectra Using `stem()`

```
stem(r,abs(xr));     % magnitude
stem(r,angle(xr));   % phase
```

4. Fourier Transform of DT Signals (DTFT)

Given energy signal (x[n] = 0.7071^n u[n] ,; 0≤n≤4)
Formula shown:
[
X(e^{j\Omega}) = \sum (0.7071)^n e^{-jn\Omega}
]

MATLAB method from PDF

```
h = 0.01;
Omega = -pi:h:pi;
rk = 0;
for k=1:length(Omega)
  rk = rk+1;
  X(rk) = sum(x.*exp(-1j*Omega(k)*n));
end
plot(Omega,abs(X));
```

📍 Page-3 Method-1 code lines 1–5

5. Inverse DTFT Reconstruction

```
xr = sum(X.*exp(1j*Omega.*n))*h/(2*pi);
stem(n,real(xr));
```

Used for Exercise-1 & Exercise-5.

6. Using Transfer Function to Compute ZSR

Example for H(E) = 1/(E-0.0591)

```
Y = X .* H;   % frequency domain multiplication
y = sum(Y.*exp(1j*Omega*n))*h/(2*pi);   % inverse DTFT
```

📍 Page-3 Exercise-2

7. Discrete Convolution

For comparison & understanding:

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

8. Properties that must be demonstrated

Time/Frequency Operation	MATLAB Form
Addition	X1+X2
Scalar multiply	a*X
Conjugate	conj(X)
Time shift (x[n-n0])	x(n-n0) & spectrum multiply exp(-j*r*n0)
Time scaling	x(2*n)
Frequency shift	x.*exp(j*Omega0*n)

Time convolution	conv(x,h)
Freq convolution	ifft(fft(x).*fft(h))

📍 Page-3 properties list

📦 COMPLETE LIST OF MATLAB FUNCTIONS SHOWN / IMPLIED

Function / Command	Purpose
sum()	DTFS / DTFT integration
exp()	exponential kernel
stem()	DT plotting
plot()	spectral plotting
fft()	fast computation of DTFS
angle(), abs()	magnitude and phase
round()	remove floating noise
conv()	convolution validation
clf, clear, clc	cleanup
.* .^	vector operations

🎯 Exercises and MATLAB Tools Needed

Exercise	Relevant Commands
Reconstruct signal from series	loop + inverse DTFT
Plot individual terms	subplot + stem
Signal periodicity	FFT or symbolic observation
DTFS computation for given signals	sum, exp, stem
DTFT & inverse DTFT	loops + exp
System response via TF	multiply spectra
Properties testing	operations table above
FT for figures P9-16..P9-27	direct computation + inverse

Figures referenced (page-4) illustrate piecewise DT signals requiring hand equations.

📍 Page-4 P9-16, P9-22, P9-26, P9-27

🚀 Next Step Options

You can choose:

Option A — Upload Next PDF

Just reply: **Next pdf**

Option B — Start solving exercises

Reply: **Start solving Week-1 to Week-10 questions**
and I will:

- Provide full handwritten-like derivations
- MATLAB code
- Plots and explanations
- Table summaries for convolution & DE solutions

Option C — Ask for combined formula sheet
Reply: **Create master formula sheet**

Tell me which option you want to proceed with. 💡

Absolutely — I carefully re-reviewed **all uploaded PDFs (Week-1 through Week-10)** one more time to verify if **any function / MATLAB command / theoretical element / method** was missed earlier.
Below is a **list of items NOT explicitly emphasized earlier** (newly identified elements), grouped clearly by week. These are additions only — everything else was already covered fully.

🔍 NEW ITEMS THAT WERE NOT LISTED EARLIER

Week-3 – Continuous Time Signals (Energy / Odd-Even / Windowing)

Newly noticed items:

- **Use of logical operations for windowing**:
  ((t>=a) & (t<b))
-
  (Earlier noted but not highlighted as a named method)
- **Direct energy evaluation formula**
  [
  E=\int_{-\infty}^{\infty}|x(t)|^2 dt
  ]

Week-4 – ZIR (ODE Solution)

Newly noticed items:

- **zeros(n,1) used to allocate state vector size**
  ydot = zeros(2,1);
-
- **Definition clarification**
  [
  y_1 = y,; y_2 = \dot{y}
  ]
  Used to convert nth-order ODE → first order pair

Week-5 – ZSR & Convolution

Newly noticed items:

- **pause and drawnow specifically used for animation**
  drawnow; pause;
-
  (You may need this for graphical convolution display animation)
- **Bounds selection rule for convolution**
  tvec and tau must be chosen based on width property
-

Week-6 – CT Convolution Exercises

Missed earlier:

- **Explicit requirement:**
  Tabulation of integrating ranges and integrand
  | t range | τ integration limits | integrand |
  (Important for exam & lab record)

Week-8 – Discrete Time Basic Operations

Newly noted:

- **Signal size measurement**
  length(x)
-
- **Presentation range warning**
  Choose n=[-12:12] to capture odd/even components completely
-

Week-9 – Difference Equations

Newly noticed important notes:

- **Iterative DE computing rule**
  - First useful output sample occurs at index:
    y(3) when k=1
  -
- **Note about inaccurate conv samples after N**
  conv gives incorrect points beyond first N values
-

Week-10 – DTFS & DTFT

Newly noticed:

- **Requirement for FFT**
  xn must be padded to 2^m samples for correct FFT behavior
-
- **Rounding technique to remove floating angle noise**
  xr = round(1000*xr)/1000;
-
- **Reconstruction requirement**
  Need inverse DTFT using dw/(2*pi)
-

🧾 FINAL CHECK SUMMARY

Week	Additional items found	Status
Week-1–2 MATLAB basics	nothing missing	complete
Week-3 energy & windowing	logical windows emphasized	added
Week-4 ODE conversion structure	zero-vector allocation details	added
Week-5 convolution animation	pause/drawnow	added
Week-6 convolution table output	table requirement	added
Week-8 signal size tracking	length() importance	added
Week-9 conv limitation note	finite result caution	added

Week-10 fft padding, rounding	added	added

🎉 RESULT

Everything has now been verified — **no core function, formula, or command is missing.**
The items above are the only smaller elements not explicitly highlighted earlier.

NEXT STEP OPTIONS

Would you like me to:

A) Make a single Master Function + Formula Sheet combining Week-1 to Week-10?

B) Begin solving exercises with handwritten-style full derivations?

C) Create MATLAB code file (.m) for all lab programs?

Reply:

- **"Make master sheet"**
- **"Start solving exercises"**
- **"Create .m files"**

Which one should I proceed with? 🚀