Volume: 13 Issue: VIII Month of publication: August 2025
DOI: https://doi.org/10.22214/ijraset.2025.73829

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VIII Aug 2025- Available at www.ijraset.com

Applications of MATLAB in Engineering

Seraz Ahmad
Assistant Professor, Electrical Engineering Department, R.R Institute Of Modern Technology , Lucknow , Uttar Pradesh , India

Abstract: MATLAB has evolved from a numerical computing environment into a comprehensive engineering platform with rich toolboxes, modeling languages (Simulink, Stateflow), and code-generation capabilities. This paper surveys core applications of MATLAB across major engineering disciplines—electrical, electronics & communication, mechanical, civil, and computer engineering—emphasizing signal processing, control systems, power systems, image processing, communication systems, optimization, and machine learning. We present compact case studies, reproducible code snippets, and a discussion on performance, validation, and deployment (including embedded code generation). Limitations and best practices are articulated to guide students and practitioners in selecting MATLAB for research and industrial workflows.
Keywords: MATLAB, Simulink, Signal Processing, Electrical Engineering, Control Systems, Power Systems, Image Processing, Communications, Simulation. Optimization, Machine Learning, Code Generation, Digital Twin

I. INTRODUCTION

MATLAB (MATrix LABoratory) is a high-level programming environment designed for matrix-oriented numerical computation, visualization, and algorithm development.
Beyond its core language, MATLAB’s domain-specific toolboxes and Simulink enable model-based design, rapid prototyping, and hardware-in-the-loop (HIL) testing. Its widespread adoption in academia and industry makes it a pivotal tool for engineering research, teaching, and product development.

A. *Objectives of this Paper*

Summarize cross-disciplinary applications of MATLAB.
Provide compact, reproducible examples demonstrating typical research workflows.
Discuss validation, scalability, and deployment considerations.
Contributions: A structured survey with code-oriented exemplars, guidance on best practices, and a checklist for reproducible MATLAB research.

II. MATLAB ECOSYSTEM OVERVIEW

1) *Core Language:* Array-first paradigm, vectorization, graphics,app building (App Designer). Toolboxes- Signal Processing, Control System, Optimization, Statistics & Machine Learning (SMLT), Image Processing & Computer Vision, Communications, Power Systems (Simscape Electrical), Reinforcement Learning, Deep Learning, etc.
2) *Simulink:* Graphical environment for multi-domain dynamic systems; supports Stateflow (state machines), Simscape (physical modeling), and automatic code generation (Embedded Coder).
3) *Integration:* Python/MEX interfaces, ROS, FPGA/SoC support, and cloud execution.

III. REPRESENTATIVE APPLICATIONS BY DISCIPLINE

A. *Electrical & Electronics Engineering*

Signal Processing: Filtering, spectral analysis, time–frequency transforms. Control Systems: Classical/modern control design, robust control, MPC. Power Systems: Load flow, dynamics, protection studies with Simscape Electrical. Communications: Modulation, channel coding, 5G/NR link-level simulation.

1) *Circuit Analysis and Electronics*

MATLAB excels in solving the systems of linear equations that naturally arise in circuit analysis (e.g., using Node Voltage or Mesh Current methods). Engineers can define complex circuits mathematically and solve for voltages, currents, and transfer functions with minimal code.

2) Transient and AC Analysis

While MATLAB can perform basic analysis, its companion environment, Simulink, and the Simscape Electrical (formerly SimPowerSystems and SimElectronics) toolbox provide a graphical, physical modeling environment. Engineers can drag-and-drop components (resistors, capacitors, transistors, op-amps, motors) to build a circuit schematic. Simulink then automatically formulates and solves the underlying differential equations, enabling detailed time-domain (transient) and frequency-domain (AC sweep) analysis.

Example: Simulating the startup transient of a DC-Do-DC buck converter, analyzing harmonic distortion in an amplifier, or designing active filter networks.

3) Signal Processing and Communications

The Signal Processing Toolbox and Communications Toolbox are industry standards for algorithm development in these fields.

Algorithm Development

MATLAB provides a comprehensive library of functions for signal analysis (FFT, spectrograms), digital filter design (FIR, IIR), and statistical signal processing. Engineers can algorithmically design a filter, analyze its frequency response, and apply it to a signal in a few lines of code.

Communications Systems: The Communications Toolbox allows for the end-to-end simulation of communication systems, including source coding, channel coding (e.g., Reed-Solomon, LDPC), modulation (QAM, PSK, OFDM), channel models (AWGN, fading), and reception. This is vital for designing and testing modern wireless standards like 5G and WiFi.

Example: Modeling a software-defined radio (SDR), evaluating the bit error rate (BER) performance of a new modulation scheme under noisy conditions, or developing noise-cancellation algorithms for audio devices.

4) Control Systems

The Control System Toolbox provides dedicated functions for analyzing and designing feedback control systems, which are ubiquitous in electronics (power supplies), robotics, and automation.

Modeling and Analysis: Engineers can easily create transfer function or state-space models of dynamic systems. The toolbox provides immediate access to key analytical plots: Bode plots (frequency response), root locus, Nyquist plots, and step response.

Controller Design: It supports techniques for designing PID controllers, lead-lag compensators, and optimal controllers (LQR). Automated tuning algorithms can quickly derive controller parameters to meet desired performance specifications like rise time and overshoot.

Example: Designing a cruise control system for a vehicle, stabilizing the flight of a drone, or regulating the speed of a motor.

5) Power Systems Analysis

The Simscape Electrical toolbox is essential for modeling and simulating generation, transmission, distribution, and consumption of electrical power.

Renewable Integration

Engineers can model large-scale AC and DC grids, integrate renewable energy sources like solar farms and wind turbines, and study their impact on grid stability.

Fault Analysis and Protection

Simulations can be run to analyze system behavior under fault conditions (e.g., line-to-ground fault) to design and test protective relay schemes and circuit breaker coordination.

Example: Simulating the load flow of a small microgrid, analyzing the harmonic injection from power electronic inverters, or designing a maximum power point tracking (MPPT) algorithm for a solar panel.

Beyond Simulation

Code Generation and Hardware Integration

A pivotal advancement is MATLAB's ability to transition from design to implementation.

6) Hardware Support

MATLAB provides built-in support for interacting with common hardware like Arduino, Raspberry Pi, and USB oscilloscopes/data acquisition (DAQ) cards. This allows engineers to acquire real-world data directly into MATLAB for online analysis or to send control signals from MATLAB to hardware.

7) Code Generation

The MATLAB Coder and Simulink Coder toolboxes can automatically convert MATLAB algorithms and Simulink models into readable, efficient C/C++ code. This code can be deployed directly onto embedded processors (like ARM Cortex-M), DSPs, and FPGAs, dramatically reducing the time and effort required for prototyping and production.

B. Mechanical Engineering

Dynamics & Vibration: Modal analysis, frequency response. Control & Mechatronics: Plant modeling, controller synthesis, HIL testing. Thermo-Fluids: CFD coupling, parameter estimation, digital twins.

C. Civil & Structural Engineering

Structural Analysis: Response to loads, damage detection using signal processing/ML. Transportation: Traffic simulation/optimization. Water Resources: Time-series forecasting and optimization for reservoir operation.

D. Computer & Data Science

ML/DL: Classification, regression, clustering, deep networks. Optimization: Convex/nonlinear programming for engineering design. Computer Vision: Image preprocessing, segmentation, feature extraction.

IV. CASE STUDIES WITH REPRODUCIBLE SNIPPETS

The examples are minimal, portable, and designed to run in core **MATLAB** with commonly available toolboxes. Replace synthetic data with domain data for research use.

A. Signal Denoising & Spectral Analysis (Electrical)

```
fs = 1e3; t = (0:1/fs:1-1/fs)';
clean = sin(2*pi*50*t) + 0.5*sin(2*pi*120*t);
noisy = clean + 0.4*randn(size(t));
% Design a bandpass filter around 50–150 Hz
bp = designfilt('bandpassiir','FilterOrder',6,'HalfPowerFrequency1',40,
   'HalfPowerFrequency2',160,'SampleRate',fs);
den = filtfilt(bp,noisy);
% Power spectrum
N = 2^nextpow2(length(den));
Y = fft(den,N)/N; f = fs/2*linspace(0,1,N/2+1);
P2 = 2*abs(Y(1:N/2+1));
figure; plot(f,P2); xlabel('Frequency (Hz)'); ylabel('|P1(f)|');
```

Outcome: Peaks at ~50 Hz and ~120 Hz are recovered after denoising.

B. PID Control via Frequency Design (Control Systems)

```
s = tf('s');
G = 1/(s*(s+2));  % Plant
C = pidtune(G,'PID');
T = feedback(C*G,1);
step(T); grid on; title('Closed-loop Step Response');
```

Outcome: Automatic tuning yields a stable, well-damped response suitable for baseline studies.

C. Power Flow Sketch with Simscape Electrical (Power Systems)

```
% Simple two-bus equivalent using per-unit parameters
V1 = 1.0; X = 0.2; Pload = 0.8; Qload = 0.3;  % p.u.
% Newton–Raphson update for voltage angle at bus 2 (simplified)
V2 = 1.0; delta = 0;
for k=1:20
```

```
   Pcalc = V1*V2/X*sin(delta);
 J = V1*V2/X*cos(delta);
 delta = delta + (Pload - Pcalc)/J;  % update
end
% Report estimated power angle and transfer
Ptransfer = V1*V2/X*sin(delta);
```

Outcome: Illustrates power-angle relation; full-scale research should use Simscape/third-party packages for full AC load flow.

D. Image Edge Detection & Quality Metrics (Computer Vision)

```
I = im2gray(imread('peppers.png'));     % replace with domain image
E = edge(I,'Canny');
psnr_val = psnr(I,imgaussfilt(I,1));
imshowpair(I,E,'montage'); title(sprintf('PSNR ref vs blurred: %.2f dB',psnr_val));
```

Outcome: Reproducible baseline for segmentation and quality analysis.

E. Communications: OFDM Link-Level Skeleton

```
M = 16; N = 64; cp = 16; L = 5; Nsym = 100;  % QAM, subcarriers, CP, channel taps
bits = randi([0 M-1],N*(Nsym),1);
sym = qammod(bits,M,'UnitAveragePower',true);
X = reshape(sym,N,[]);
x_time = ifft(X); x_cp = [x_time(end-cp+1:end,:); x_time];
h = (randn(L,1)+1j*randn(L,1))/sqrt(2*L);
y = filter(h,1,x_cp(:));
Y = fft(reshape(y, N+cp, [])); Y = Y( (cp+1):end, : );
Z = qamdemod(reshape(Y,[],1),M,'UnitAveragePower',true);
ber = mean(bits~=Z);
```

Outcome: Produces a baseline BER for a frequency-selective channel; extend with channel estimation/equalization.

F. Optimization in Engineering Design

```
obj = @(x) (x(1)-1)^2 + (x(2)+2)^2;  % convex bowl
A = []; b = []; Aeq = [1 1]; beq = 1; lb = [-5 -5]; ub = [5 5];
x0 = [0 0];
[x,fval] = fmincon(obj,x0,A,b,Aeq,beq,lb,ub);
```

Outcome: Demonstrates constrained nonlinear optimization typical in design problems.

G. Machine Learning for Fault Classification (SVM)

```
% Synthetic features: [RMS, Kurtosis, CrestFactor]
X = [randn(100,3)+[1 0 0]; randn(100,3)+[3 2 1]];
y = [zeros(100,1); ones(100,1)];
Mdl = fitcsvm(X,y,'KernelFunction','rbf','Standardize',true);
cv = crossval(Mdl); loss = kfoldLoss(cv);
```

Outcome: Illustrates supervised classification with cross-validation; replace with vibration or current-signature features.

V. VALIDATION, VERIFICATION, AND REPRODUCIBILITY

1) Versioning: Record MATLAB and toolbox versions; use ver snapshots.
2) Determinism: Fix RNG with rng(seed) for ML/Monte Carlo.
3) Data Management: Use datastore, timetable, and matfile for large datasets.
4) Model Verification: Compare against analytical solutions or standards; unit tests with matlab.unittest.
5) Performance: Prefer vectorization; profile with profile on; scale via Parallel Computing Toolbox.

VI. DEPLOYMENT & CODE GENERATION

1) Embedded Coder: Generate ANSI-C for MCUs/SoCs from MATLAB/Simulink models. GPU/FPGA Acceleration: Use GPU arrays and HDL Coder for rapid prototyping.
2) Interoperability: Call Python libraries, integrate ROS/ROS2 for robotics, and use Simulink Real-Time for HIL.

VII. MOST ESSENTIAL MATLAB COMMANDS USED IN ENGINEERING

A. Basic Commands

clc → Clears the Command Window.
clear → Removes all variables from workspace.
close all → Closes all figure windows.
who / whos → Displays variables in workspace.
help <command> → Opens help for a command.
doc <command> → Opens detailed documentation.
path → Displays or sets MATLAB search path.
pwd / cd → Current directory / Change directory.

B. Variables & Arrays

a = 5; → Assign value.
zeros(n,m) / ones(n,m) → Create matrices of 0s or 1s.
eye(n) → Identity matrix.
rand(n,m) / randn(n,m) → Random numbers (uniform/normal).
linspace(a,b,n) → Linearly spaced vector.
size(A) / length(A) → Dimensions of matrix/vector.
reshape(A,m,n) → Reshape matrix.
diag(A) → Diagonal elements.

C. Matrix & Linear Algebra

A*B → Matrix multiplication.
A.*B → Element-wise multiplication.
inv(A) → Inverse of matrix.
det(A) → Determinant.
eig(A) → Eigenvalues & eigenvectors.
svd(A) → Singular Value Decomposition.
rank(A) → Matrix rank.
pinv(A) → Pseudoinverse.

D. Plotting & Visualization

plot(x,y) → 2D plot.
subplot(m,n,p) → Multiple plots in one figure.
title('text'), xlabel('x'), ylabel('y') → Labels.
legend('data1','data2') → Legend.
grid on / grid off → Toggle grid.
mesh(X,Y,Z) / surf(X,Y,Z) → 3D plots.
imshow(I) → Display image.

E. Programming & Logic

for, while, if, switch → Control statements.
function [out1,out2] = func(in1,in2) → Define function.
disp('text') → Display output.
fprintf('x = %d\\n',x) → Formatted printing.

F. Signal Processing & Math

fft(x) / ifft(x) → Fast Fourier Transform & inverse.
conv(x,h) → Convolution.
filter(b,a,x) → Digital filter.
mean(x), median(x), std(x) → Statistics.
max(x), min(x) → Extremes.
sum(x), prod(x) → Summation, product.
diff(x) → Differences.
trapz(x,y) → Numerical integration.

G. Control Systems / Simulation

tf(num,den) → Transfer function.
zpk(z,p,k) → Zero-pole-gain model.
bode(sys) → Bode plot.
step(sys) → Step response.
sim('model') → Run a Simulink model.

H. Discussion: Strengths, Limitations, and When to Use MATLAB

1) Strengths

Rich domain toolboxes and documentation.
Model-Based Design in Simulink shortens development cycles.
Strong visualization and app-building capabilities.

2) Limitations

Commercial licensing cost.
Runtime speed vs. hand-optimized C/C++ (mitigated by codegen/vectorization).
Ecosystem lock-in; portability considerations vs. open-source stacks.

3) Decision Pointers

Choose MATLAB for rapid prototyping with strong domain toolboxes and codegen needs.
Prefer mixed workflows (MATLAB + Python/C) when leveraging broader OSS ecosystems.

VIII. CONCLUSION

MATLAB remains a cornerstone for engineering computation, simulation, and deployment. Its strength lies in end-to-end workflows—from algorithm design to real-time testing and code generation. With disciplined validation and reproducible practices, MATLAB accelerates research translation to industry-grade solutions across multiple engineering domains.

REFERENCES

[1] MathWorks. MATLAB Documentation and Simulink Documentation.
[2] S. K. Mitra, Digital Signal Processing: A Computer-Based Approach.
[3] K. Ogata, Modern Control Engineering.
[4] R. C. Gonzalez and R. E. Woods, Digital Image Processing.
[5] S. Boyd and L. Vandenberghe, Convex Optimization.
[6] Kundur, P., Power System Stability and Control.
[7] Proakis, J. G., and Salehi, M., Digital Communications.
[8] MathWorks. (2023). Simscape Electrical Documentation. https://www.mathworks.com/help/physmod/sps/index.html
[9] Chapman, S. J. (2022). MATLAB Programming for Engineers. Cengage Learning.
[10] Oppenheim, A. V., & Schafer, R. W. (2009). Discrete-Time Signal Processing. Prentice Hall. (Concepts often implemented in MATLAB).
[11] Glover, J. D., Sarma, M. S., & Overbye, T. J. (2012). Power System Analysis & Design. Cengage Learning.
[12] Palm, W. J. (2020). Introduction to MATLAB for Engineers (4th ed.). McGraw-Hill Education.
[13] Attaway, S. (2022). MATLAB: A Practical Introduction to Programming and Problem Solving (6th ed.).
[14] Krein, P. T. (2018). Elements of Power Electronics (2nd ed.). Oxford University Press.
[15] Often utilizes MATLAB/Simulink and Simscape for power electronics simulation.

[16] Haykin, S., & Van Veen, B. (2018). Signals and Systems (3rd ed.). Wiley.
[17] Kuo, B. C., & Golnaraghi, F. (2013). Automatic Control Systems (10th ed.). Wiley.
[18] Lyshevski, S. E. (2003). Engineering and Scientific Computations Using MATLAB. Wiley.
[19] Singh, M., & Panigrahi, B. K. (2011). MATLAB/Simulink-Based Transient Analysis of Electrical Machines and Drives. International Journal of Electrical Engineering Education, 48(1), 99-113.
[20] Garcia, C., et al. (2020). A MATLAB-Based Framework for the Design and Analysis of Hybrid Renewable Energy Systems. Energies, 13(15), 3891.