```
% Convolutional Encoder and Viterbi Decoder
% Optimized version with better comments and user experience

close all; clear all; clc;

%% Input Parameters Section
fprintf('=== Convolutional Coding with Viterbi Decoding ===\n\n');
```

```
=== Convolutional Coding with Viterbi Decoding ===
```

```

% Get message bits from user
msg = input('Enter the message bits (e.g., [1 0 1 1 0]): ');
if isempty(msg)
    error('Message bits cannot be empty.');
end

% Get shift register length
l = input('Enter the shift register length (constraint length): ');
if isempty(l) || l <= 0
    error('Shift register length must be a positive integer.');
end

% Get number of output polynomials
n = input('Enter the number of output polynomials: ');
if isempty(n) || n <= 0
    error('Number of output polynomials must be positive.');
end

% Get generator polynomials
fprintf('\nEnter generator polynomials in octal format:\n');
```

```
Enter generator polynomials in octal format:
```

```
g = zeros(1, n);
for j = 1:n
    g(j) = input(sprintf('Generator polynomial %d: ', j));
    if isempty(g(j))
        error('Generator polynomial cannot be empty.');
    end
end

%% Trellis Construction and Validation
fprintf('\n--- Trellis Construction ---\n');
```

```
--- Trellis Construction ---
```

```

% Convert polynomials to trellis structure
trellis = poly2trellis(l, g);

% Display trellis information
```

```
fprintf('Trellis Structure:\n');
```

```
Trellis Structure:
```

```
disp(trellis);
```

```
     numInputSymbols: 2
    numOutputSymbols: 4
          numStates: 4
        nextStates: [4×2 double]
           outputs: [4×2 double]
```

```
fprintf('Number of States: %d\n', trellis.numStates);
```

```
Number of States: 4
```

```
fprintf('Number of Input Symbols: %d\n', trellis.numInputSymbols);
```

```
Number of Input Symbols: 2
```

```
fprintf('Number of Output Symbols: %d\n', trellis.numOutputSymbols);
```

```
Number of Output Symbols: 4
```

```
% Validate trellis structure
[isok, status] = istrellis(trellis);
if ~isok
    error('Invalid trellis structure: %s', status);
else
    fprintf('✓ Trellis validation: PASSED\n');
end
```

```
✓ Trellis validation: PASSED
```

```
%% Convolutional Encoding
fprintf('\n--- Encoding Process ---\n');
```

```
--- Encoding Process ---
```

```
fprintf('Original message: ');
```

```
Original message:
```

```
disp(msg);
```

```
     1     0     0     1     1
```

```
% Perform convolutional encoding
code = convenc(msg, trellis);
fprintf('Encoded output: ');
```

```
Encoded output:
```

```
disp(code);
```

```
     1     1     1     0     1     1     1     1     0     1
```

```
fprintf('Code rate: %d/%d\n', length(msg), length(code));
```

```
Code rate: 5/10
```

```
%% Viterbi Decoding
fprintf('\n--- Decoding Process ---\n');
```

```
--- Decoding Process ---
```

```
% Get received code (simulating transmission)
rcode = input('Enter the received code for decoding: ');
if isempty(rcode)
    error('Received code cannot be empty.');
end

% Validate length compatibility
if mod(length(rcode), length(msg)) ~= 0
    warning('Received code length may not be compatible with message length.');
end

% Set traceback length (CORRECTED: ensure it doesn't exceed input symbols)
num_input_symbols = length(rcode) / n;
tblen = min(5 * l, num_input_symbols);  % Fixed: minimum of 5*l and available
symbols
fprintf('Using traceback length: %d\n', tblen);
```

```
Using traceback length: 5
```

```
% Perform Viterbi decoding with hard decision
decoded = vitdec(rcode, trellis, tblen, 'trunc', 'hard');

fprintf('\n--- Results ---\n');
```

```
--- Results ---
```

```
fprintf('Received code:  ');
```

```
Received code:
```

```
disp(rcode);
```

```
     1     1     1     1     1     1     1     1     0     1
```

```
fprintf('Decoded message: ');
```

```
Decoded message:
```

```
disp(decoded);
```

```
     1     0     0     1     1
```

```
fprintf('Original message: ');
```

```
Original message:
```

```
disp(msg);
```

```
     1     0     0     1     1
```

```
%% Performance Analysis
fprintf('\n--- Performance Analysis ---\n');
```

```
--- Performance Analysis ---
```

```
% Calculate bit error rate if original message is available
if length(msg) == length(decoded)
    errors = sum(msg ~= decoded);
    ber = errors / length(msg);
    fprintf('Bit Errors: %d\n', errors);
    fprintf('Bit Error Rate (BER): %.4f\n', ber);

    if errors == 0
        fprintf('✓ Perfect decoding achieved!\n');
    else
        fprintf('✗ Decoding errors detected\n');
    end
else
    fprintf('Note: Message lengths differ, cannot compute BER\n');
end
```

```
Bit Errors: 0
Bit Error Rate (BER): 0.0000
✓ Perfect decoding achieved!
```

```
%% Additional Information
fprintf('\n--- System Information ---\n');
```

```
--- System Information ---
```

```
fprintf('Constraint Length (K): %d\n', l);
```

```
Constraint Length (K): 3
```

```
fprintf('Code Rate: 1/%d\n', n);
```

```
Code Rate: 1/2
```

```
fprintf('Total States: %d\n', trellis.numStates);
```

```
Total States: 4
```

```
fprintf('Traceback Length: %d\n', tblen);
```

```
Traceback Length: 5
```

```


% Display generator polynomials in different formats
fprintf('\nGenerator Polynomials:\n');
```

```
Generator Polynomials:
```

```
fprintf('Octal format: ');
```

```
Octal format:
```

```
disp(g);
```

```
     7     5
```

```
fprintf('Binary format: ');
```

```
Binary format:
```

```
for j = 1:n
    bin_poly = dec2bin(g(j));
    fprintf('%s ', bin_poly);
end
```

```
111 101
```

```
fprintf('\n');
```