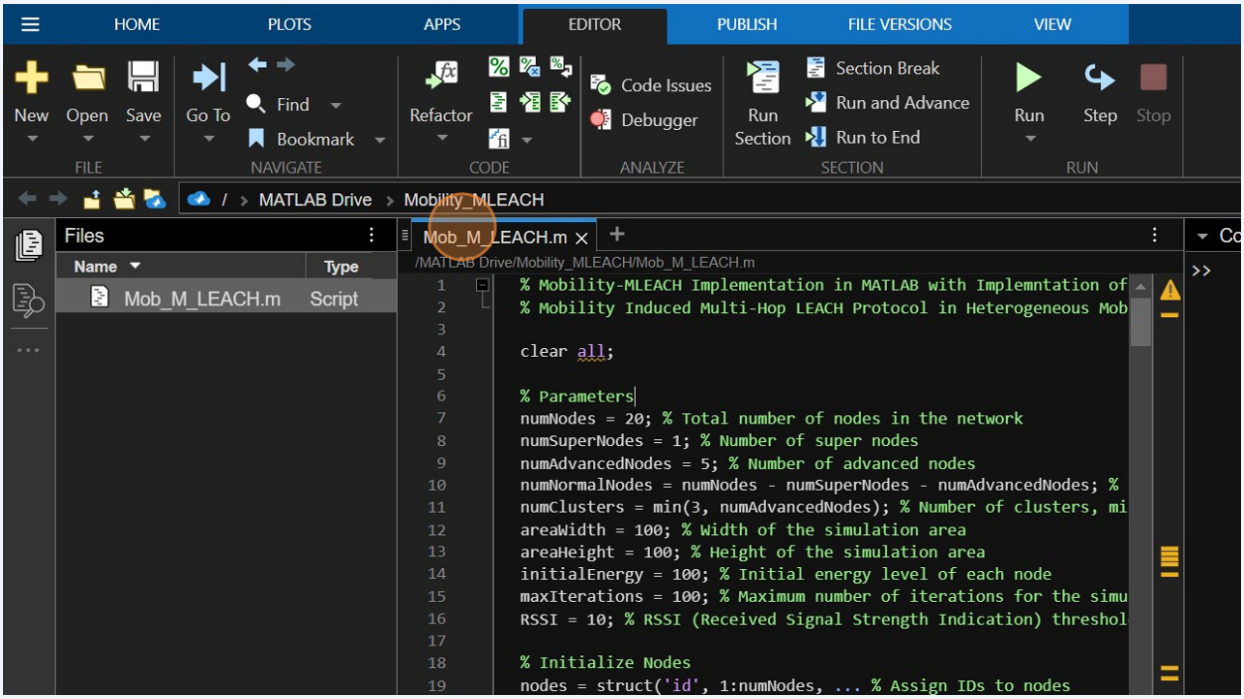
Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network Implementation in MATLAB (with Code) By Laraib Azmat (21-SE-23) and Zain Ejaz (21-SE-71)

1 Navigate to https://matlab.mathworks.com/

2 **Mobility-MLEACH (Mobility-Multi-level Energy Adaptive Clustering Hierarchy) is an advanced protocol designed for wireless sensor networks, integrating mobility-awareness with energy-efficient clustering. This protocol combines the principles of MLEACH, a variant of the well-known LEACH (Low Energy Adaptive Clustering Hierarchy), with mobility models to enhance network performance in dynamic environments.**



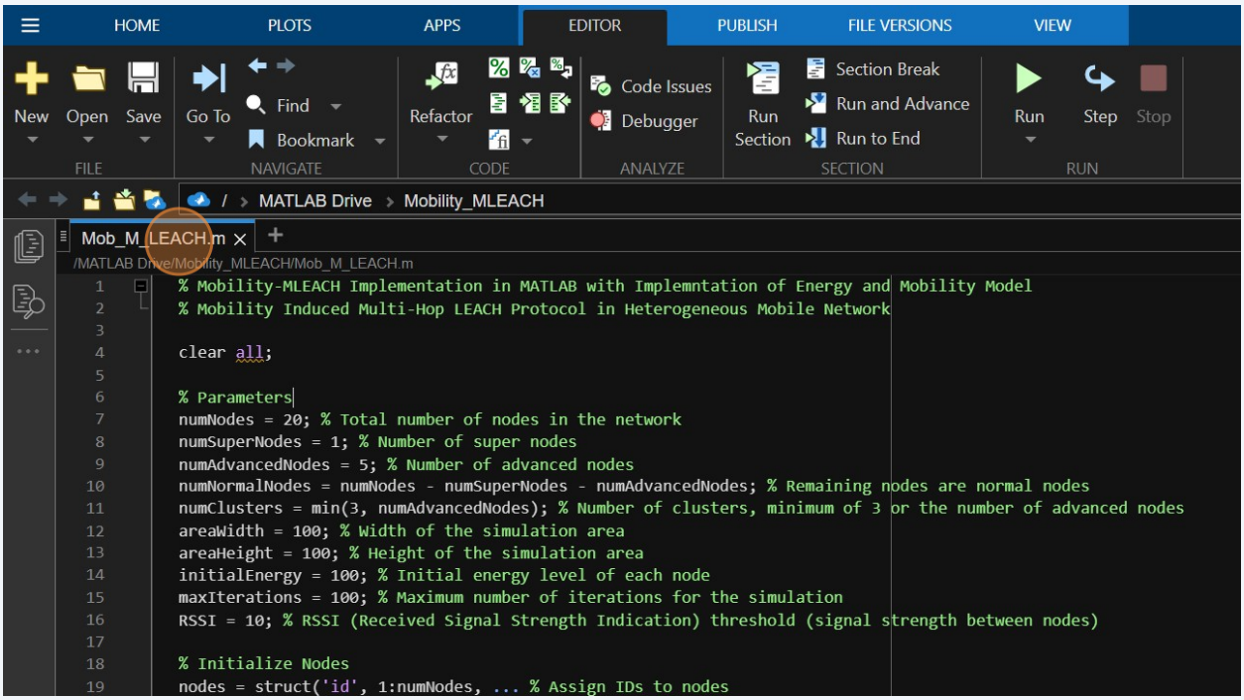
3 The key features and components of Mobility-MLEACH include:

1. **Mobility Awareness**: Unlike traditional clustering protocols that assume static network topologies, Mobility-MLEACH acknowledges the dynamic nature of wireless sensor networks where nodes may move unpredictably due to factors like user mobility, environmental changes, or node failures. It incorporates mobility models to predict and adapt to node movements effectively.

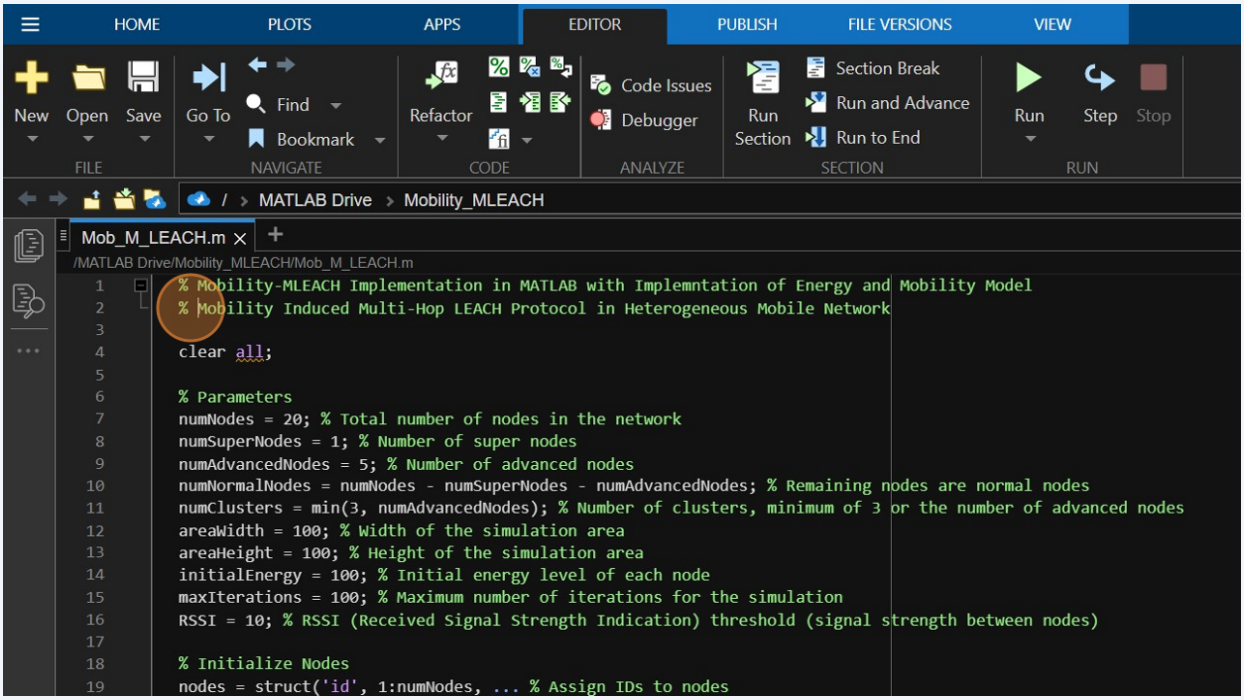
2. **Multi-level Clustering**: Similar to MLEACH, Mobility-MLEACH employs a multi-level clustering approach to organize nodes into clusters based on their energy levels. This hierarchical structure helps in reducing energy consumption and extending network lifetime by evenly distributing the energy load among nodes.

3. **Energy Efficiency**: Mobility-MLEACH focuses on energy efficiency as a primary objective. By dynamically adjusting cluster heads based on both residual energy and node mobility patterns, it aims to prolong network lifetime while maintaining network connectivity and data delivery.

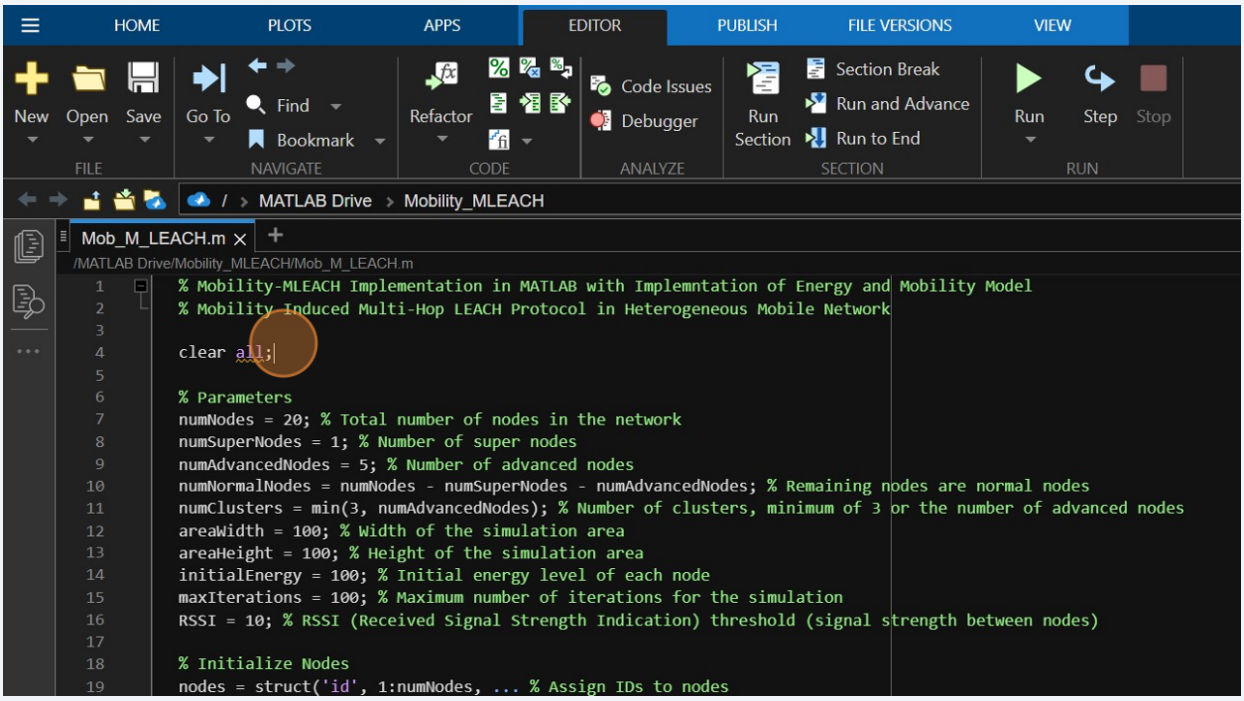
4. **Adaptability**: The protocol is designed to adapt to changing network conditions, including variations in node energy levels, mobility patterns, and environmental factors. It dynamically reconfigures cluster heads and cluster memberships to optimize energy consumption and network performance



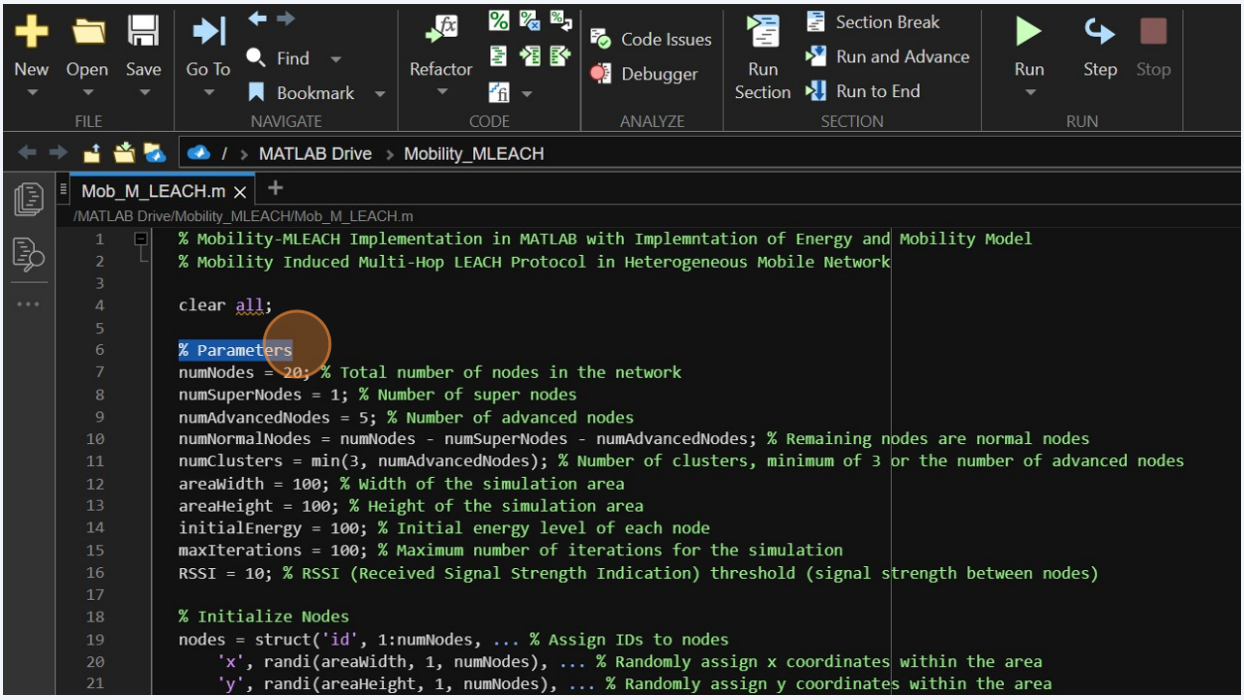
4 Mobility induced means there is mobility present(induced) in the nodes and multi hops means data transmission is divided into multiple hops and staes that will lead to efficient use of energy and send data for longer distances Heterogeneous means that it will contain multiple types of nodes (advanced, super and normal)



5 **The** clear all **statement at the beginning of the MATLAB script is used to clear all variables from the workspace. This means it removes all variables that are currently stored in memory.**



6 Firstly Lets start to initialize some perimeters to use in the code below



7

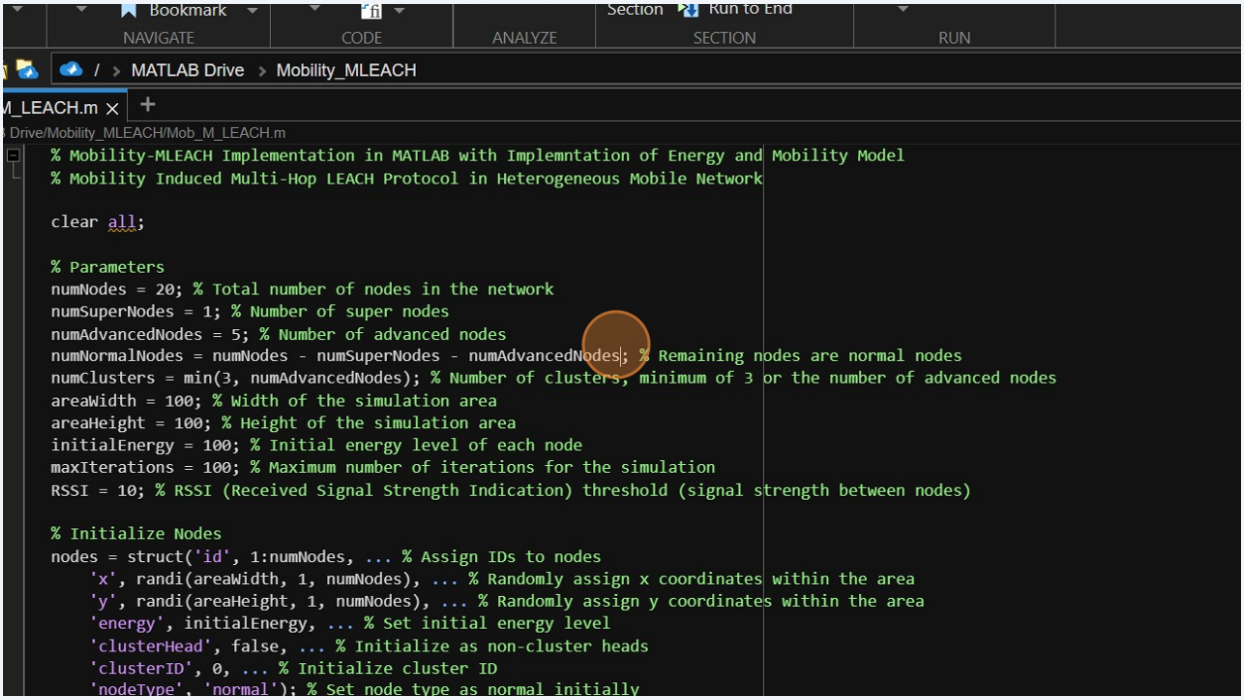
• **Total Number of Nodes (**numNodes**)**: This parameter defines the total number of nodes present in the network. It includes all types of nodes: super nodes, advanced nodes, and normal nodes.

• **Number of Super Nodes (**numSuperNodes**)**: Super nodes are a special type of node that may have higher capabilities or functionalities compared to other nodes. They might have more energy, higher processing power, or longer communication range.

• **Number of Advanced Nodes (**numAdvancedNodes**)**: Advanced nodes are another special type of node in the network. They might have slightly more capabilities than normal nodes but less than super nodes. They are typically used in clustering or routing algorithms for more efficient network management.

• **Number of Normal Nodes (**numNormalNodes**)**: These are the regular nodes in the network that do not possess any special characteristics like super nodes or advanced nodes. They are usually the majority of the nodes in the network.

• **Number of Clusters (**numClusters**)**: This parameter determines how many clusters will be formed in the network. Clustering is a technique used in many wireless sensor network protocols for better organization and management. The number of clusters may depend on various factors such as network size, node density, and energy constraints. In this case, it is set to be the minimum of 3 or the number of advanced nodes, ensuring that there are enough advanced nodes to act as cluster heads in the network.



8 These parameters define various aspects of the simulation environment and the nodes within it:

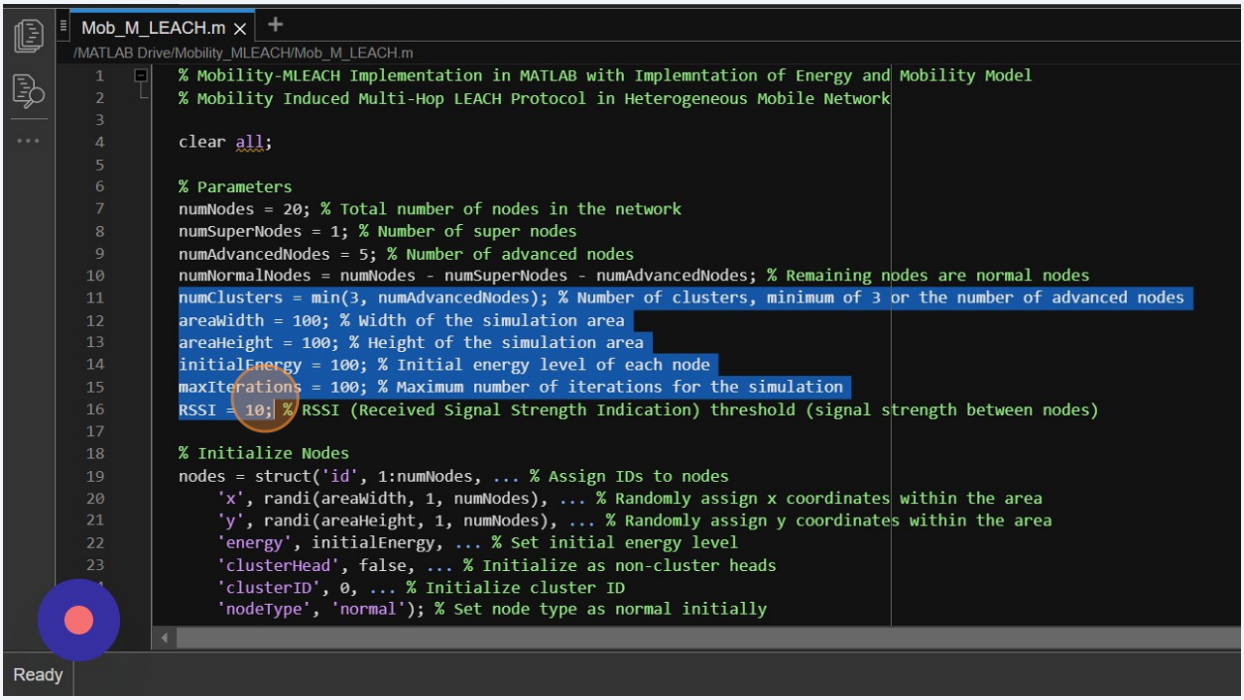
• areaWidth: This parameter specifies the width of the simulation area, usually measured in meters or some other distance unit. It determines the extent of the area in which nodes can move and communicate.

• areaHeight: Similar to areaWidth, this parameter specifies the height of the simulation area, defining the vertical extent of the simulation space.

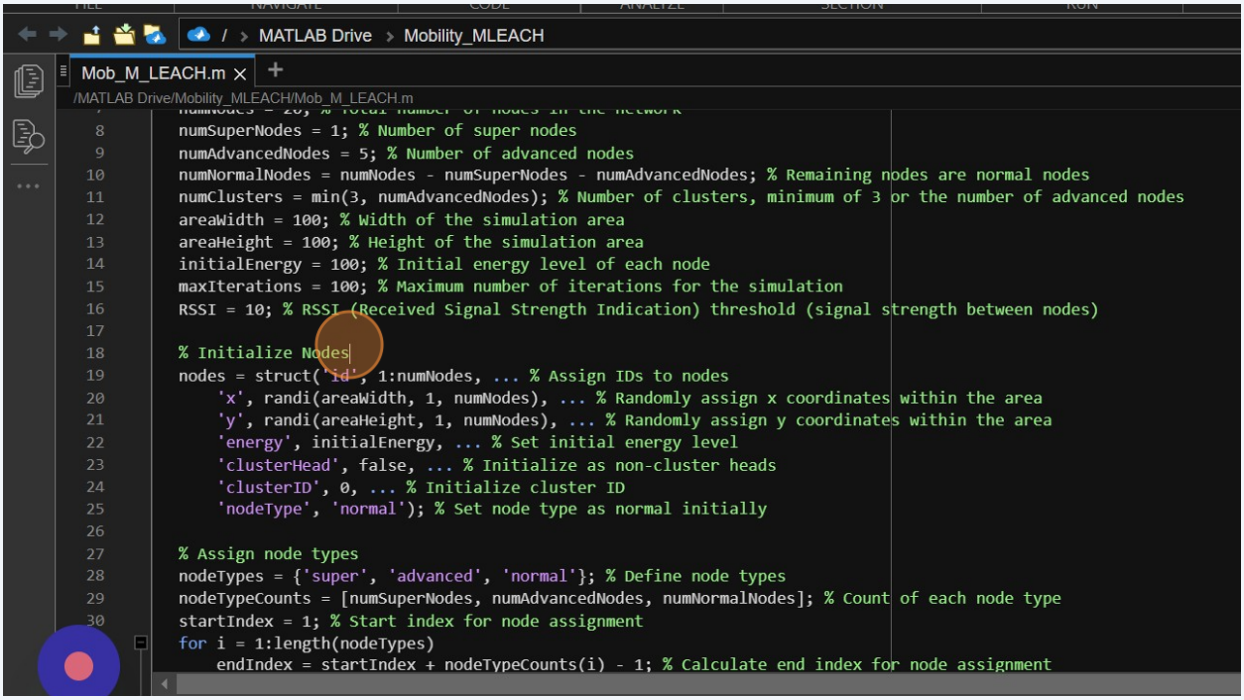
• initialEnergy: This parameter sets the initial energy level for each node in the network. The energy level represents the amount of energy available for the node to perform tasks such as data transmission, reception, and processing. It's a crucial parameter in energy-constrained networks, as nodes need to manage their energy efficiently to prolong network lifetime.

• maxIterations: This parameter defines the maximum number of iterations or time steps for which the simulation will run. Each iteration typically represents a discrete time step in the simulation, during which various actions such as node movement, energy consumption, and data transmission take place.

• RSSI: RSSI stands for Received Signal Strength Indication. It's a measure of the power level of the signal received by a node from another node or source. In wireless communication, RSSI is often used as an indicator of the strength of the received signal, which can be used to estimate the distance between nodes or to determine if a signal is strong enough for reliable communication.



9 Now lets initialize the nodes as these are having a complete structure



10 Let's break down what each field of the nodes struct represents:

• id: This field represents the unique identifier for each node in the network. Nodes are assigned IDs from 1 to numNodes.

• x: This field represents the x-coordinate of each node's position within the simulation area. The x-coordinates are randomly assigned within the range [1, areaWidth].

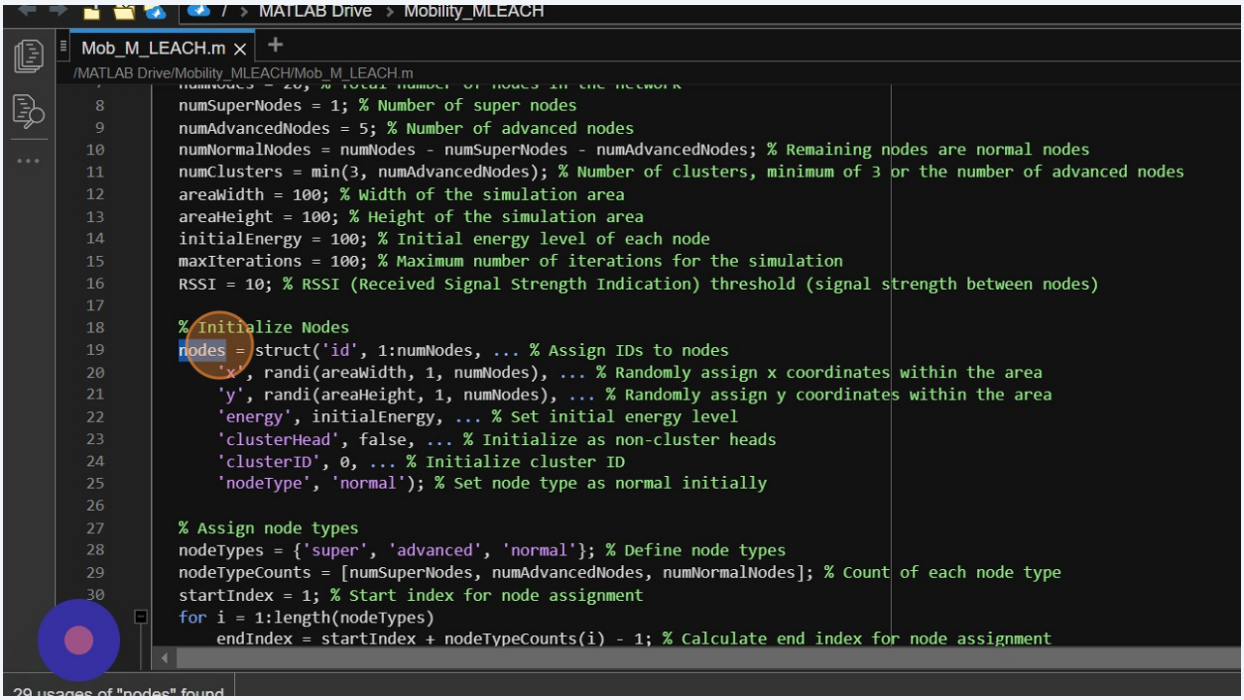
• y: Similar to x, this field represents the y-coordinate of each node's position within the simulation area. The y-coordinates are randomly assigned within the range [1, areaHeight].

• energy: This field represents the current energy level of each node. All nodes start with the same initial energy level (initialEnergy).

• clusterHead: This field indicates whether a node is a cluster head (true) or not (false). Initially, all nodes are initialized as non-cluster heads.

• clusterID: This field represents the cluster ID to which each node belongs. Initially, the cluster ID is set to 0 for all nodes.

• nodeType: This field specifies the type of each node. In this case, all nodes are initialized as "normal" nodes. However, based on the earlier definitions of numSuperNodes, numAdvancedNodes, and numNormalNodes, the node types can be differentiated as super, advanced, or normal.



11

• nodeTypes: This variable is a cell array containing the names of the different node types. In this case, it includes 'super', 'advanced', and 'normal'.

• nodeTypeCounts: This variable is an array containing the counts of each node type. It corresponds to the numbers of super nodes, advanced nodes, and normal nodes specified earlier.

• startIndex: This variable is used to keep track of the start index for assigning node types.

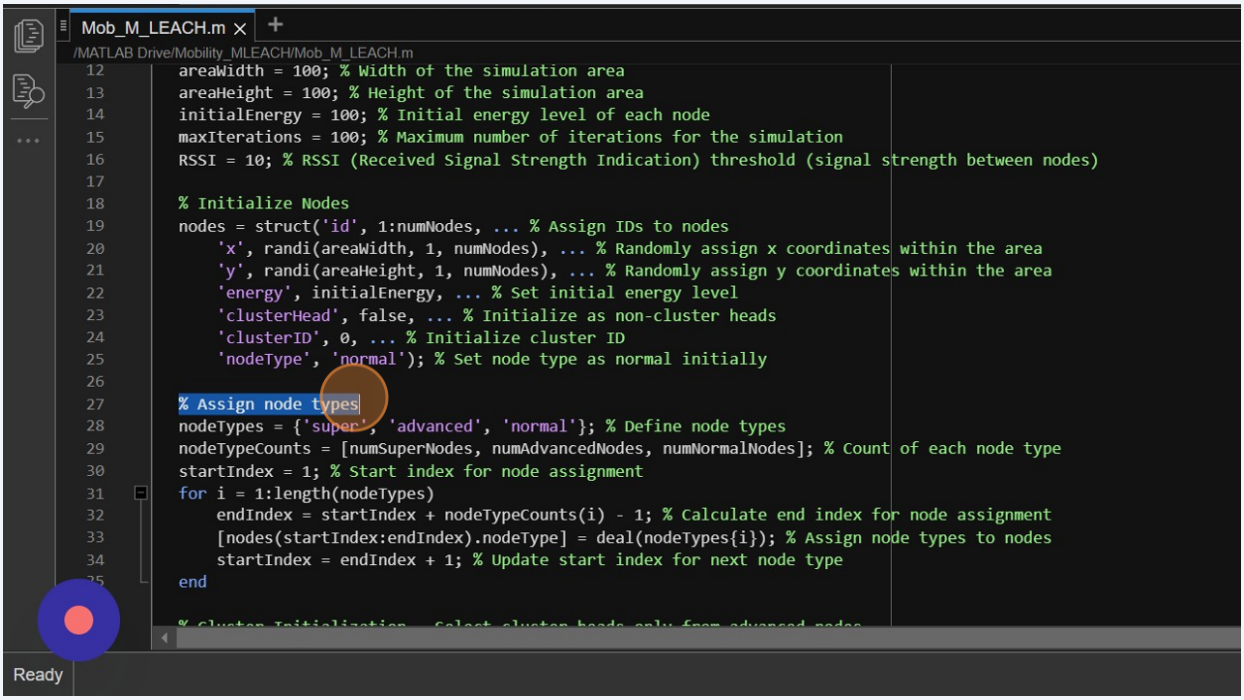
• for **loop**: This loop iterates over each node type.

• endIndex **calculation**: It calculates the end index for node assignment based on the start index and the count of nodes for the current type.

• **Node type assignment**: It assigns the current node type (nodeTypes{i}) to the nodes within the range from startIndex to endIndex.

• **Update** startIndex: It updates the startIndex for the next node type assignment by adding the count of nodes for the current type and 1 (since indices are 1-based in MATLAB).

This loop ensures that nodes are appropriately assigned the specified node types based on the counts provided.



12

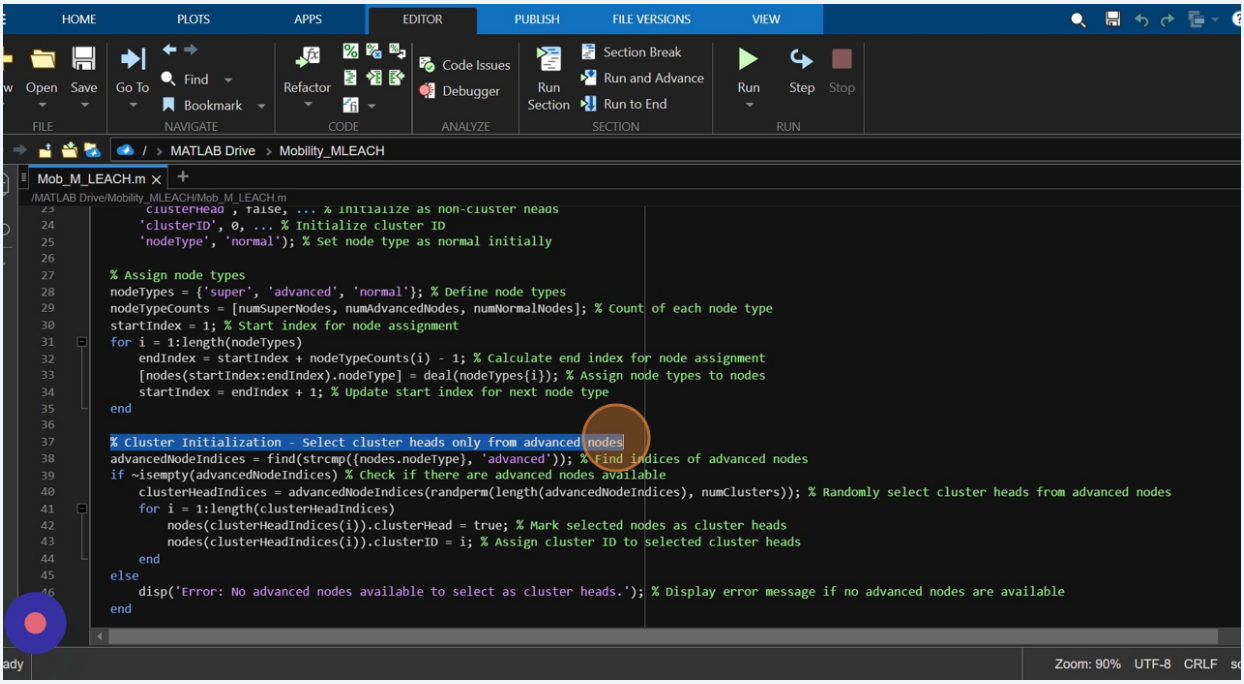
• advancedNodeIndices: This line finds the indices of nodes that have the node type 'advanced'.

• **Check for availability of advanced nodes**: The if ~isempty(advancedNodeIndices) condition checks if there are any advanced nodes available in the network. If there are no advanced nodes (isempty returns true), it displays an error message.

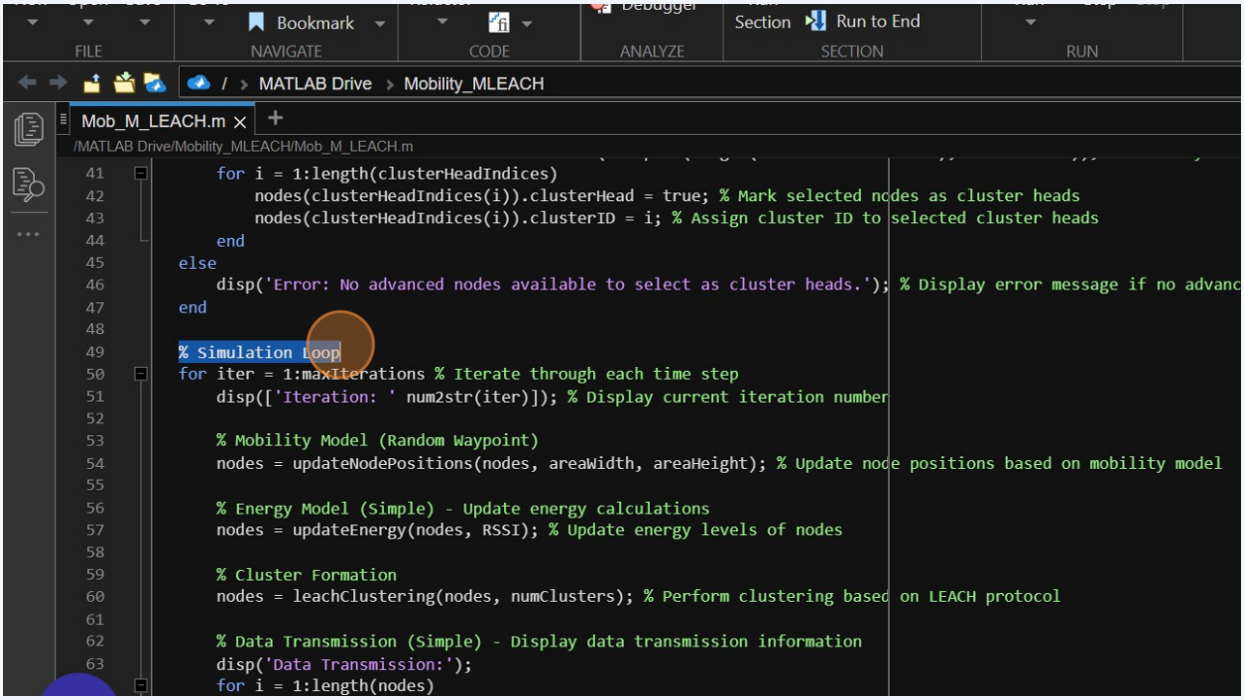
• **Select cluster heads from advanced nodes**: If there are advanced nodes available, the code proceeds to select cluster heads from them. It does so by randomly permuting the indices of advanced nodes (randperm(length(advancedNodeIndices), numClusters)) and selecting numClusters number of them.

• **Mark selected nodes as cluster heads**: For each selected cluster head, it sets the clusterHead field to true and assigns a unique clusterID to it. The clusterID is assigned sequentially starting from 1.

This section ensures that cluster heads are selected only from advanced nodes and assigns them the appropriate properties. If no advanced nodes are available, it notifies the user about the error.



13 Now lets see the simulation loop
This is the loop where all of the simulation starts

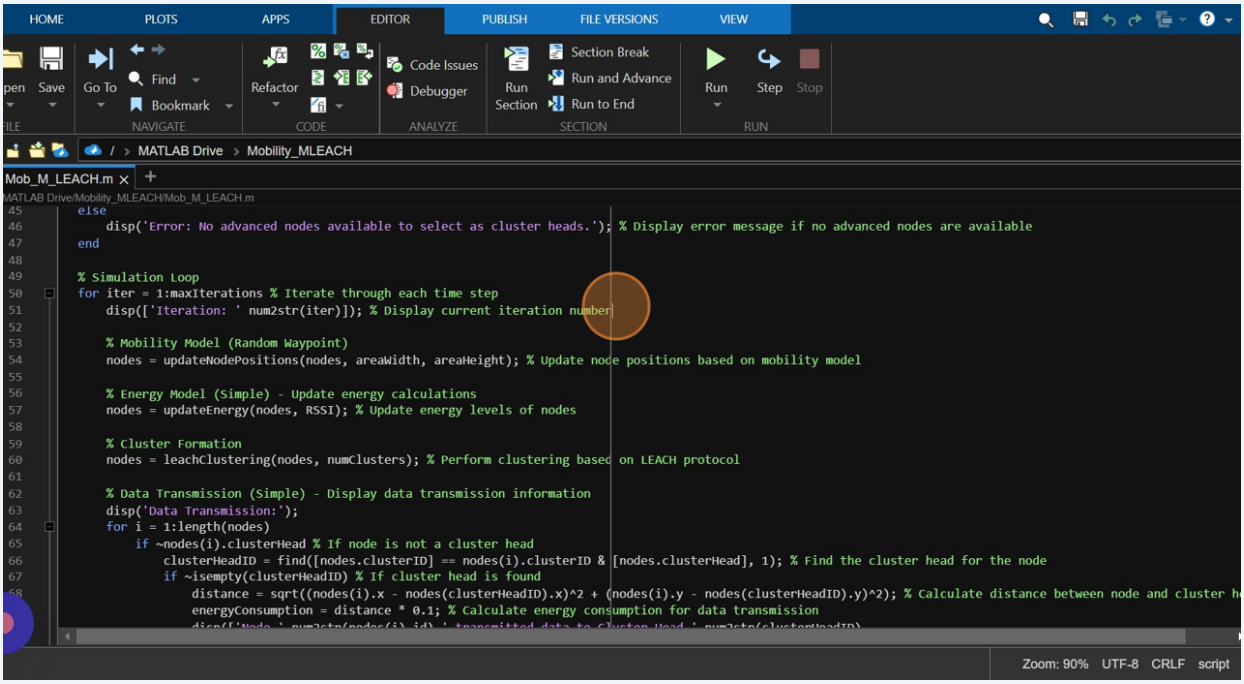


14 Here's an explanation of what happens in this loop:

• for **loop**: Iterates from 1 to maxIterations, which represents the maximum number of iterations for the simulation.

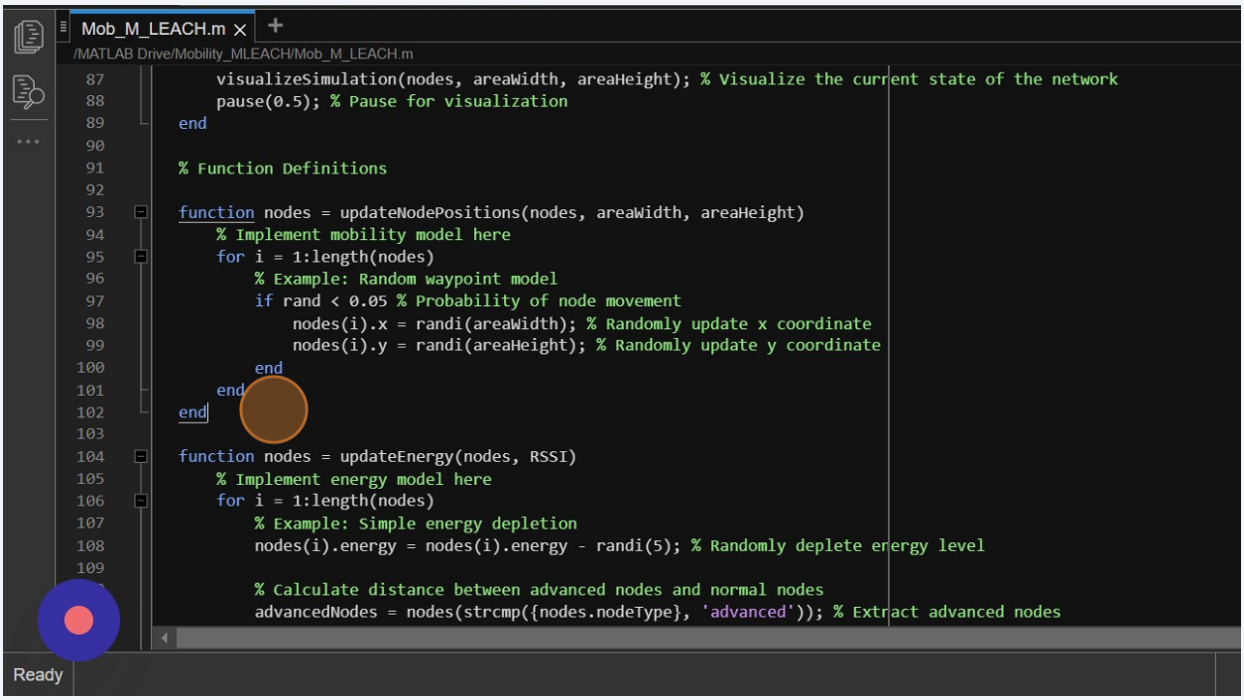
• **Display iteration number**: Each iteration, it displays the current iteration number using disp(['Iteration: ' num2str(iter)]). This helps in monitoring the progress of the simulation.

• **Mobility Model (Random Waypoint)**: It calls the updateNodePositions function to update the positions of nodes based on a mobility model. In this case, the Random Waypoint model is used. This function is responsible for simulating the movement of nodes within the simulation area. The updated node positions are stored back in the nodes structure



15

- **Input arguments**:
  - nodes: A structure array containing information about the nodes in the network.
  - areaWidth: The width of the simulation area.
  - areaHeight: The height of the simulation area.
- **Output**:
  - Updated nodes structure array with new node positions.
- **Function body**:
  - **Mobility model implementation**: This function iterates through each node in the nodes array.
  - **Random waypoint model**: It simulates node movement using a random waypoint model. In each iteration, for each node, it checks if a randomly generated value between 0 and 1 (rand) is less than 0.05, indicating a probability of 5% for node movement (this probability value can be adjusted according to the specific mobility model being simulated).
  - If the condition is met, it updates the node's position (x and y coordinates) by randomly selecting new coordinates within the simulation area (areaWidth and areaHeight) using randi function.



16 • **Input arguments**:

• nodes: The structure array containing information about the nodes in the network.

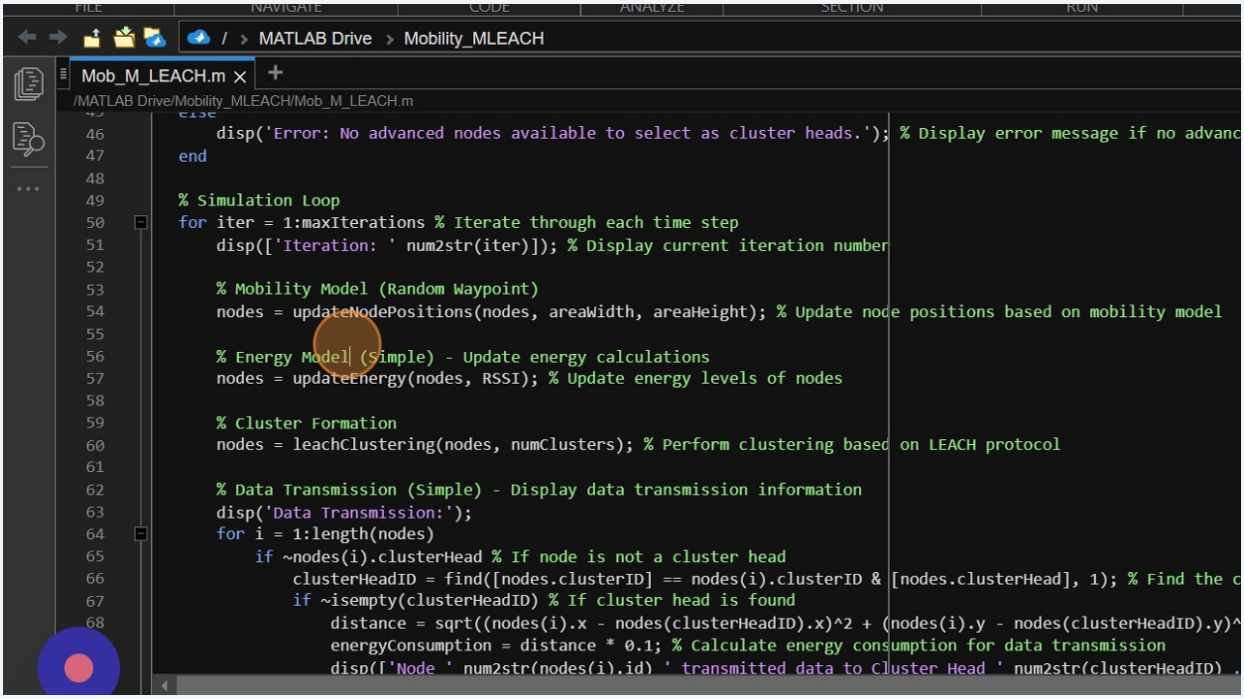
• RSSI: The Received Signal Strength Indication threshold, which is a measure of the signal strength between nodes.

• **Function body**:

• **Energy model implementation**: This function iterates through each node in the nodes array.

• **Simple energy depletion**: It simulates energy depletion by randomly reducing each node's energy level (nodes(i).energy) by a random integer value between 1 and 5 (inclusive) using randi(5). This represents a basic model of energy consumption.

• **Distance calculation and energy model update**: For each normal node, it calculates the distance between it and all advanced nodes. If the distance is less than the RSSI threshold, it calculates several energy-related parameters such as Emob, Ec, Egrp, and Enet. However, these calculations are currently placeholders (0) and need to be implemented based on the specific energy model and routing protocol being simulated.



17 • **Input arguments**:

• nodes: The structure array containing information about the nodes in the network.

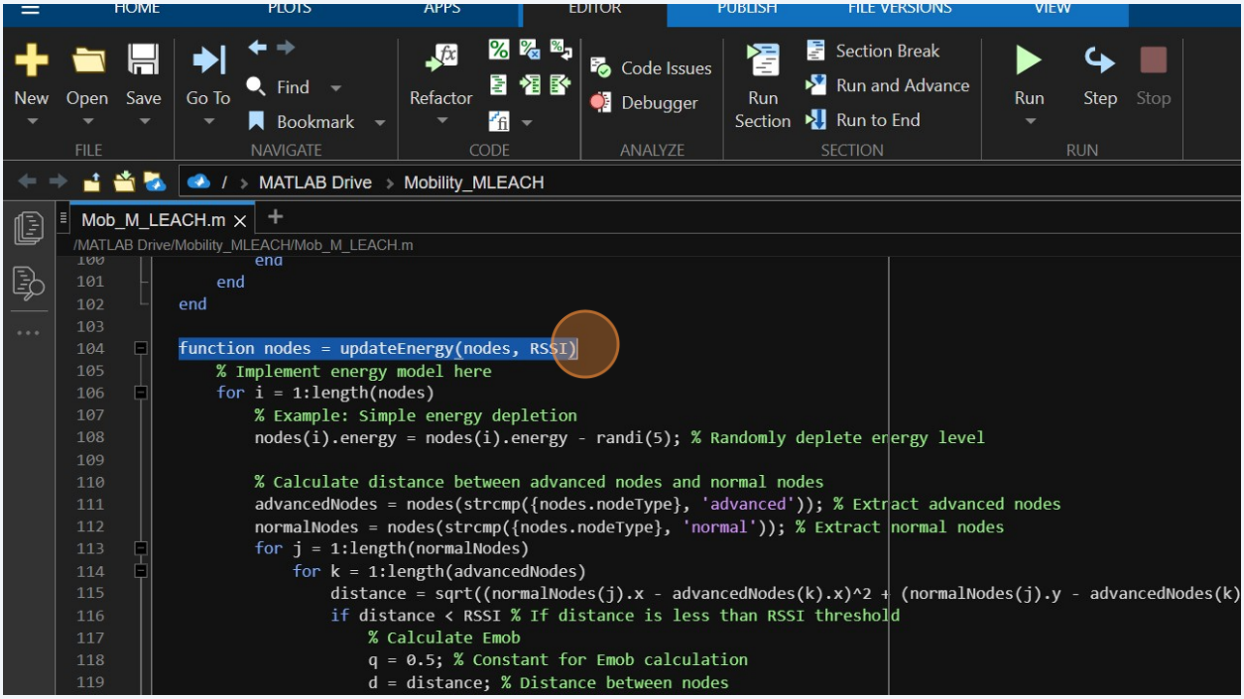
• RSSI: The Received Signal Strength Indication threshold, which is a measure of the signal strength between nodes.

• **Function body**:

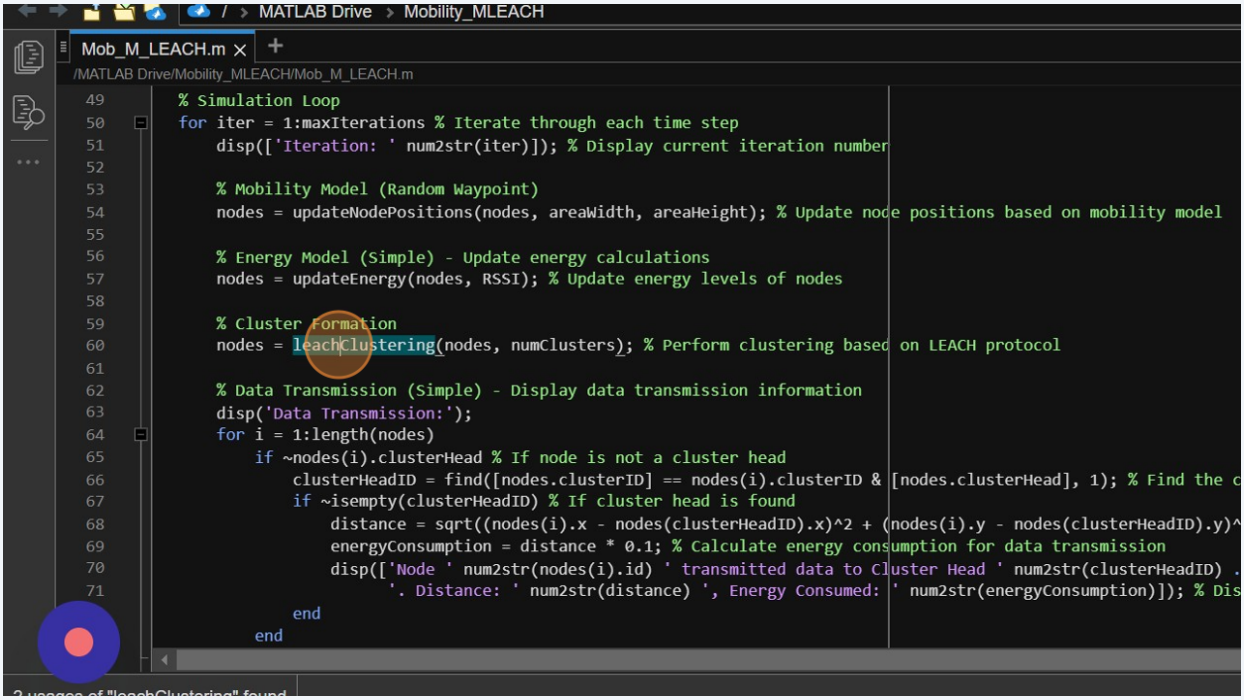
• **Energy model implementation**: This function iterates through each node in the nodes array.

• **Simple energy depletion**: It simulates energy depletion by randomly reducing each node's energy level (nodes(i).energy) by a random integer value between 1 and 5 (inclusive) using randi(5). This represents a basic model of energy consumption.

• **Distance calculation and energy model update**: For each normal node, it calculates the distance between it and all advanced nodes. If the distance is less than the RSSI threshold, it calculates several energy-related parameters such as Emob, Ec, Egrp, and Enet. However, these calculations are currently placeholders (0) and need to be implemented based on the specific energy model and routing protocol being simulated.



18 This function is to create clusters



19

• **Input arguments**:

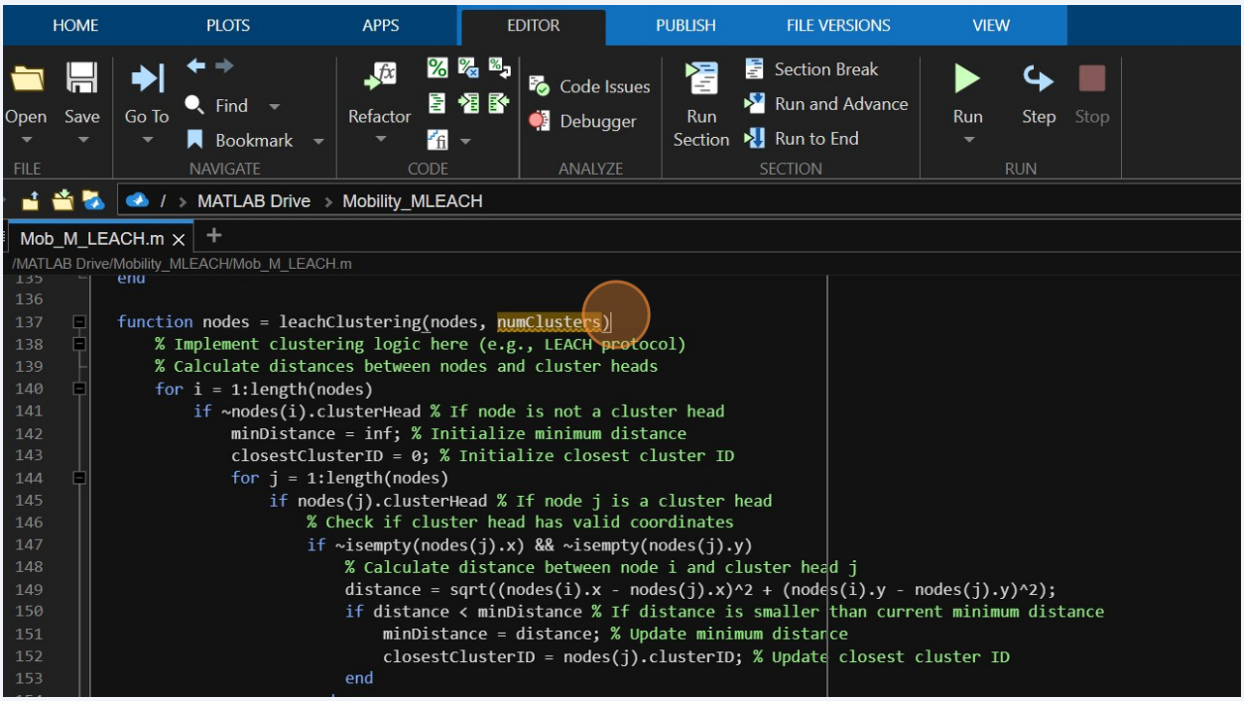
• nodes: The structure array containing information about the nodes in the network.

• numClusters: The number of clusters to be formed.

• **Function body**:

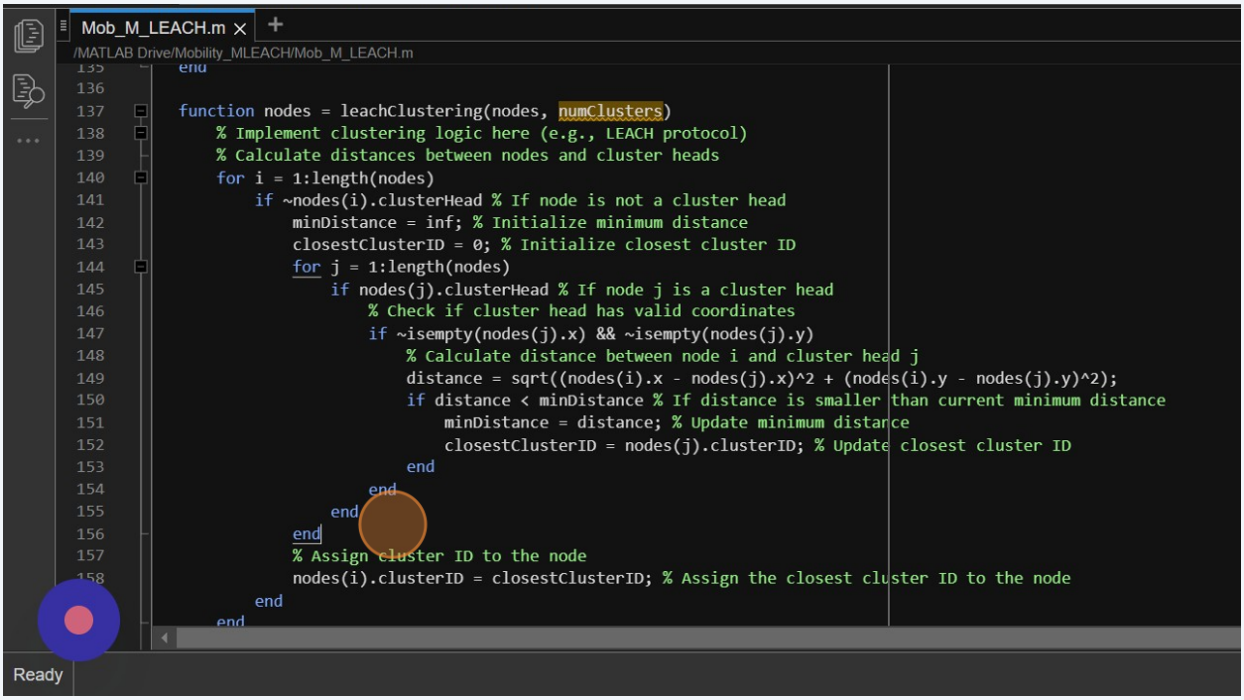
• **Cluster formation logic**: This function implements the clustering logic, specifically based on the LEACH protocol or a similar clustering algorithm.

• **Calculate distances between nodes and cluster heads**: It iterates through each node in the nodes array. For each node that is not already a cluster head (~nodes(i).clusterHead), it calculates the distance to each cluster head and assigns the node to the cluster whose head is closest.



20

• **Assign cluster ID**: It assigns the ID of the closest cluster to the node (nodes(i).clusterID). This ID will be used later for data transmission and other cluster-related operations.



21

• **Data transmission information display**:

• It first displays a message indicating data transmission is taking place within the network (disp('Data Transmission:')).

• It iterates through each node in the nodes array.

• For each non-cluster head node (~nodes(i).clusterHead), it finds the cluster head corresponding to the node's cluster (clusterHeadID) using find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1).

• If a cluster head is found (~isempty(clusterHeadID)), it calculates the distance between the node and the cluster head using the Euclidean distance formula.

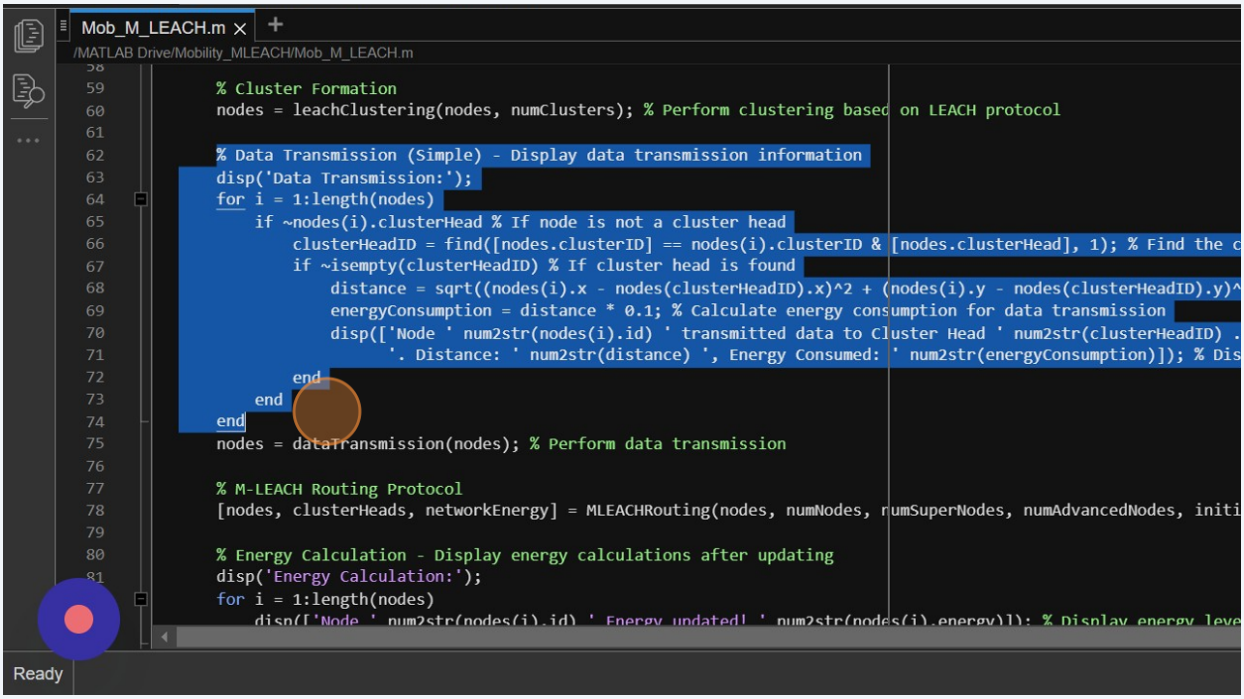
• It then calculates the energy consumption for data transmission based on the distance (energyConsumption = distance * 0.1), assuming a simple energy model.

• Finally, it displays the data transmission information including the node ID, cluster head ID, distance, and energy consumed.

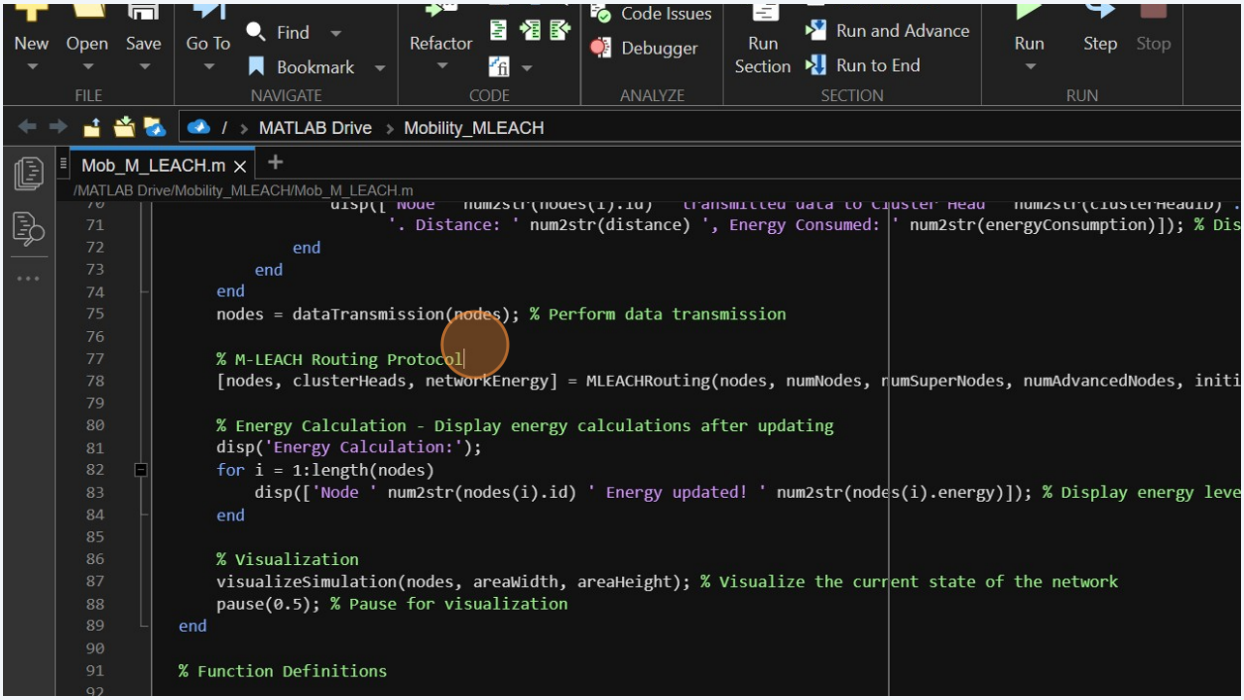
• **Data transmission function call**:

• After displaying the data transmission information, it calls the dataTransmission function to perform data transmission operations, which may involve updating energy levels or other relevant parameters for nodes in the network.

Overall, this code segment provides a simple display of data transmission information and may perform further data transmission operations through the dataTransmission function call.

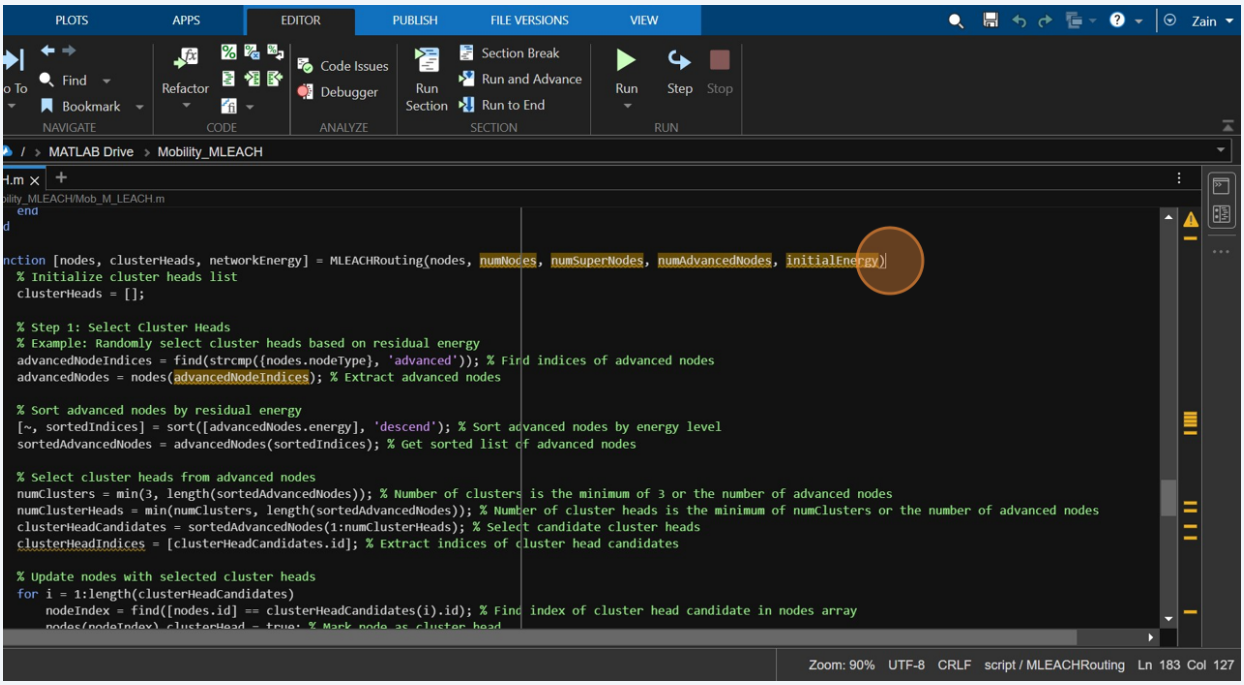


22 Here MLEACH Routing is called and passed some perimeters



23 Explanation:

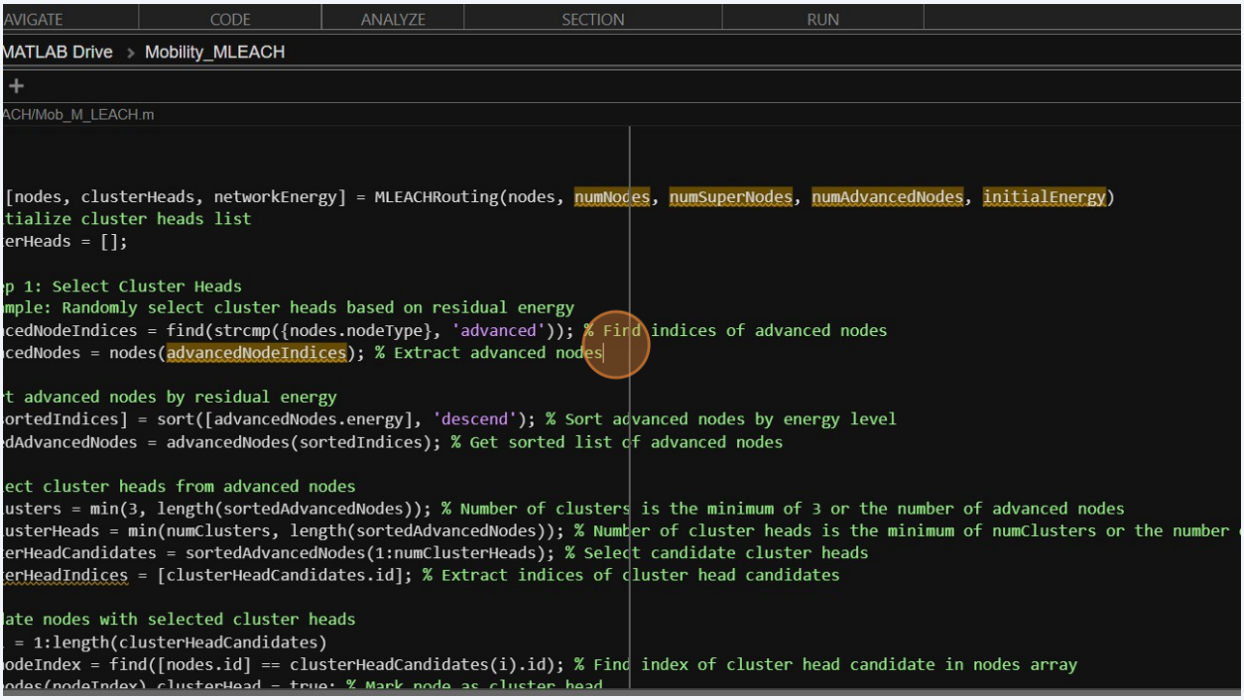
- **Input arguments**:
  - nodes: The structure array containing information about the nodes in the network.
  - numNodes: Total number of nodes in the network.
  - numSuperNodes: Number of super nodes in the network.
  - numAdvancedNodes: Number of advanced nodes in the network.
  - initialEnergy: Initial energy level of each node.
- **Output arguments**:
  - nodes: Updated structure array containing information about the nodes in the network.
  - clusterHeads: List of cluster heads selected by the M-LEACH routing protocol.
  - networkEnergy: Total network energy after energy consumption due to mobility and data transmission.
-



24 • **Function body**:

• **Step 1: Select Cluster Heads**:

• It selects cluster heads based on residual energy. In this example, it randomly selects cluster heads from the advanced nodes based on their energy levels.



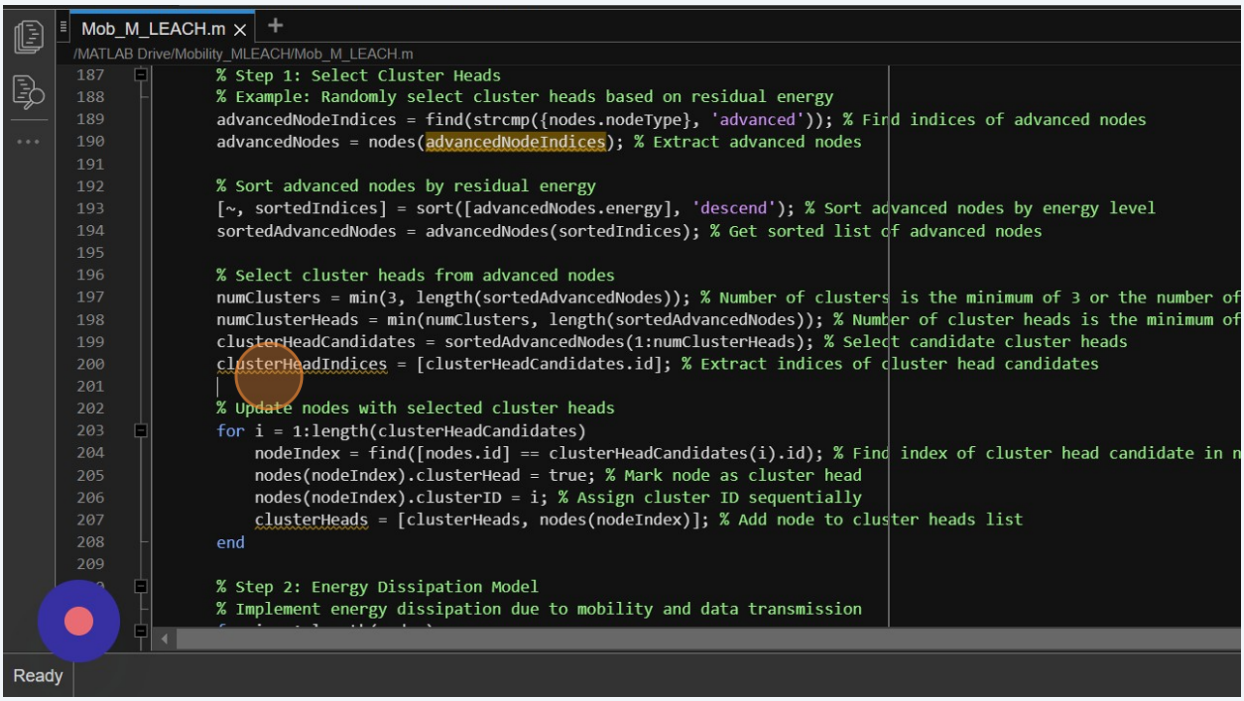
25 Select cluster heads from advanced nodes

Number of clusters is the minimum of 3 or the number of advanced nodes

Number of cluster heads is the minimum of numClusters or the number of advanced nodes

Select candidate cluster heads

Extract indices of cluster head candidates

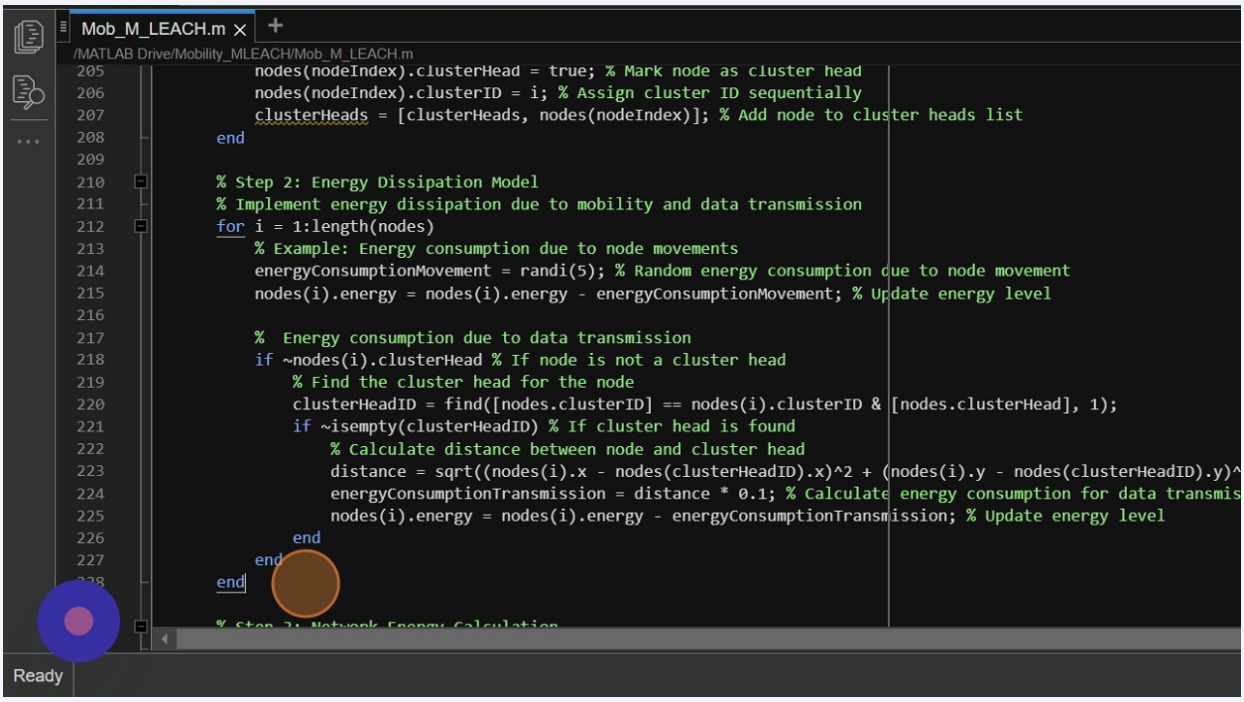


26 • **Step 2: Energy Dissipation Model**:

• It models energy dissipation due to node movements and data transmission.

• Energy consumption due to node movements is simulated by randomly reducing the energy level of each node.

• Energy consumption due to data transmission is calculated for each non-cluster head node by considering the distance to its cluster head and a constant energy consumption rate.



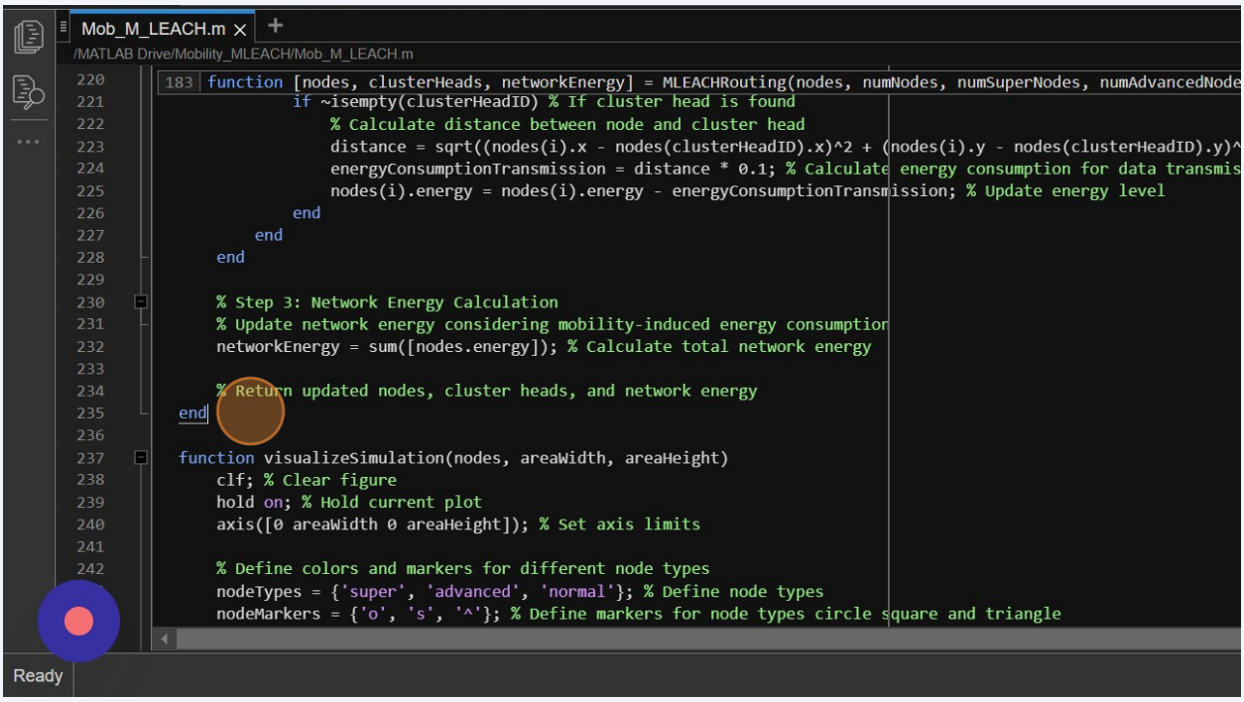
27

• **Step 3: Network Energy Calculation**:

• It calculates the total network energy after considering energy consumption due to mobility and data transmission.

• **Return values**:

• The function returns the updated nodes array, the list of selected cluster heads, and the total network energy.



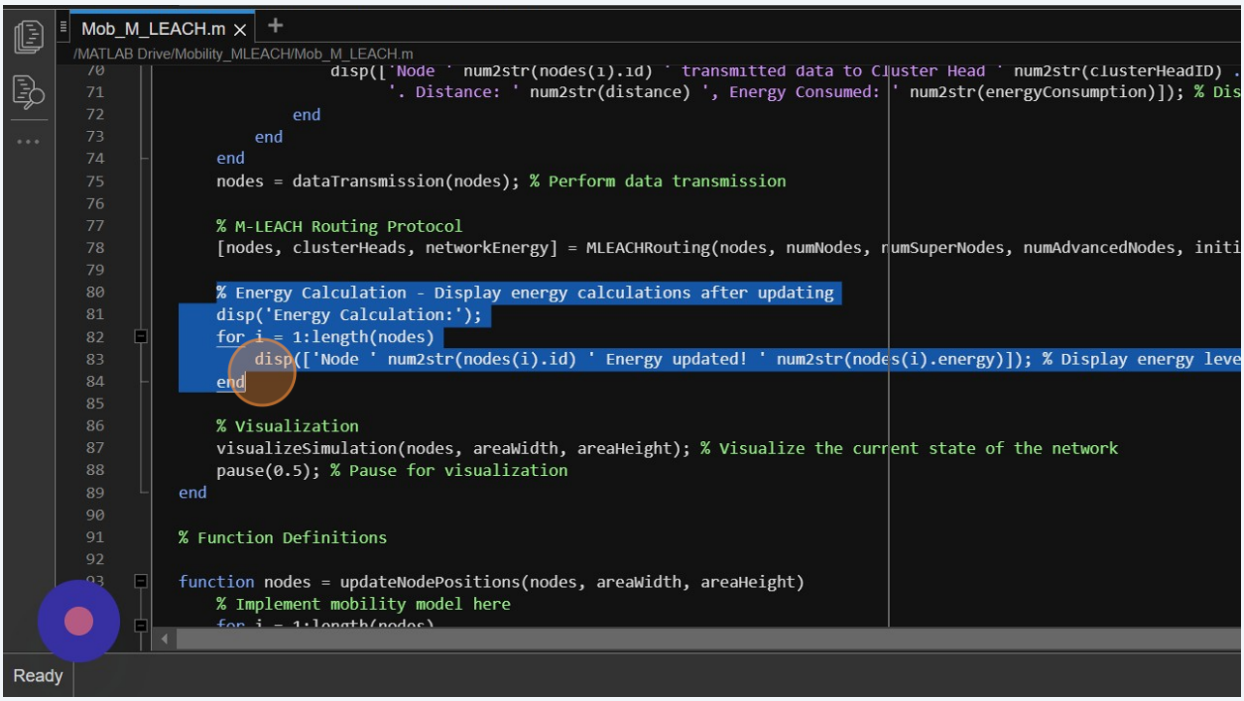

```
function [nodes, clusterHeads, networkEnergy] = MLEACHRouting(nodes, numNodes, numSuperNodes, numAdvancedNode
            if ~isempty(clusterHeadID) % If cluster head is found
                % Calculate distance between node and cluster head
                distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^
                energyConsumptionTransmission = distance * 0.1; % Calculate energy consumption for data transmis
                nodes(i).energy = nodes(i).energy - energyConsumptionTransmission; % Update energy level
            end
        end
    end

    % Step 3: Network Energy Calculation
    % Update network energy considering mobility-induced energy consumption
    networkEnergy = sum([nodes.energy]); % Calculate total network energy

    % Return updated nodes, cluster heads, and network energy
end

function visualizeSimulation(nodes, areaWidth, areaHeight)
    clf; % Clear figure
    hold on; % Hold current plot
    axis([0 areaWidth 0 areaHeight]); % Set axis limits

    % Define colors and markers for different node types
    nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
    nodeMarkers = {'o', 's', '^'}; % Define markers for node types circle square and triangle
```

28 Energy calculation display:

• It displays a message indicating that energy calculations are being shown (disp('Energy Calculation:')).

• It iterates through each node in the nodes array.

• For each node, it displays a message showing the node ID and its updated energy level (disp(['Node ' num2str(nodes(i).id) ' Energy updated! ' num2str(nodes(i).energy)])).



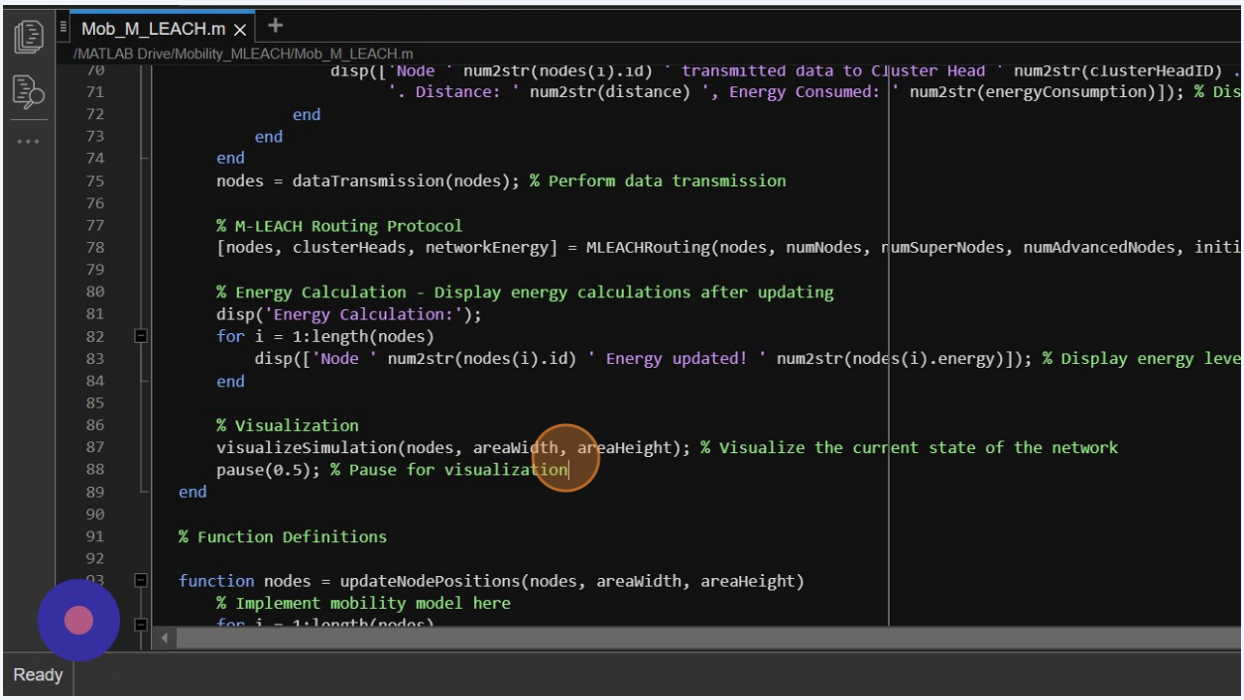
29 Explanation:

- **Visualization**:

  - It calls the visualizeSimulation function to generate a visualization of the current state of the network based on the information stored in the nodes array, as well as the dimensions of the simulation area (areaWidth and areaHeight).

- **Pause for visualization**:

  - After displaying the visualization, it pauses the execution for 0.5 seconds using the pause function. This gives time for the visualization to be observed before proceeding with the next iteration of the simulation.

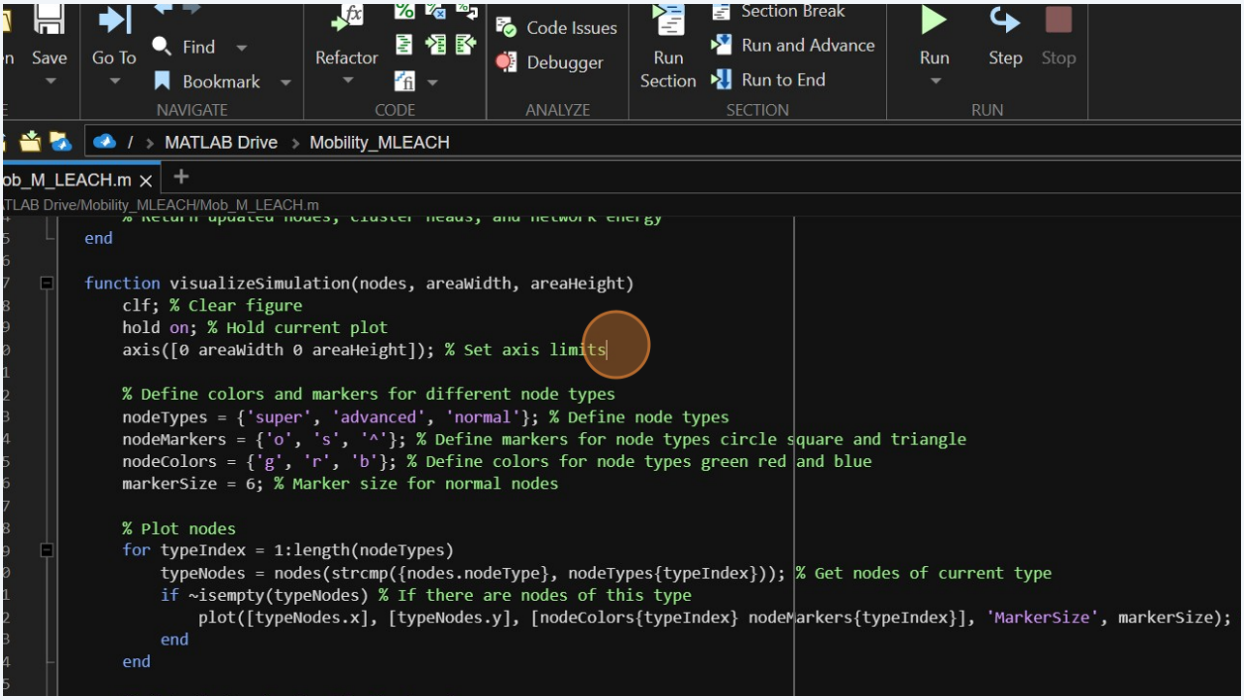


30

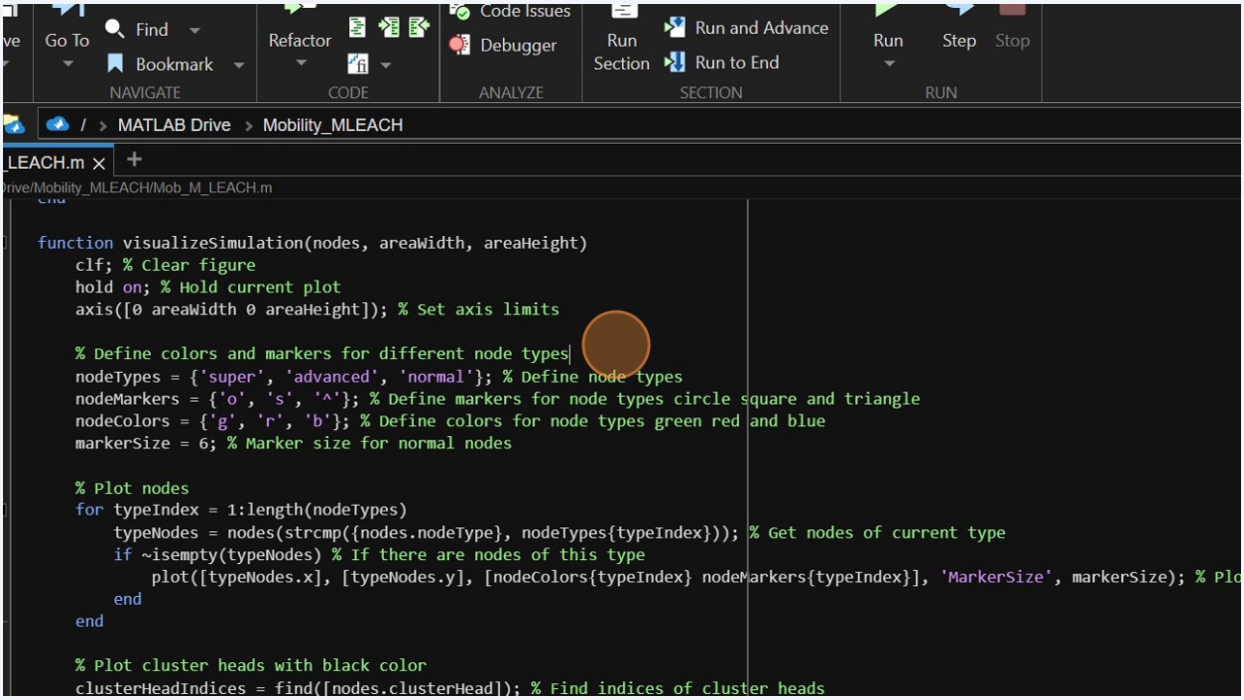
• **Clear figure**: clf clears the current figure, ensuring that each visualization iteration starts with a clean slate.

• **Hold on**: hold on allows subsequent plot commands to be overlaid on the current plot, preserving the existing content.

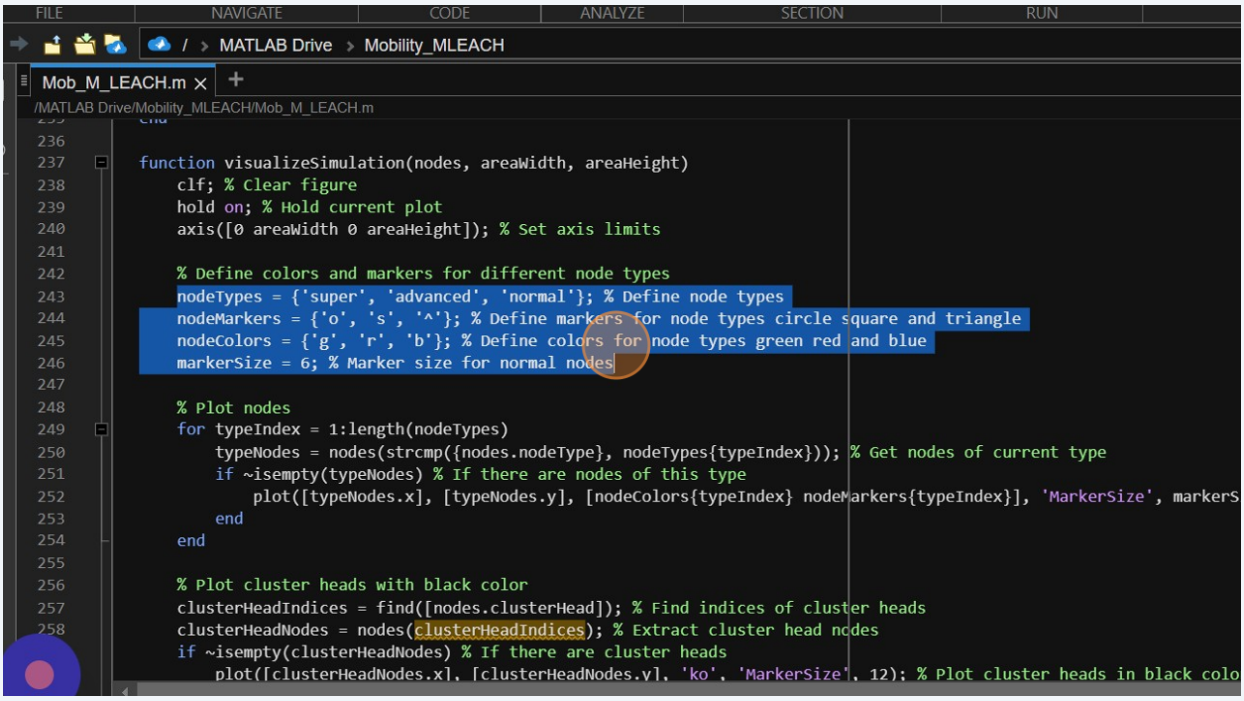
• **Set axis limits**: axis([0 areaWidth 0 areaHeight]) sets the limits for the x and y axes based on the dimensions of the simulation area.



31 **Node types, markers, and colors: Defines different types of nodes and their corresponding markers and colors.**



32 Here markers circle square and triangle are userd and colors red green and blue are used and markersize for nodes is 6



33

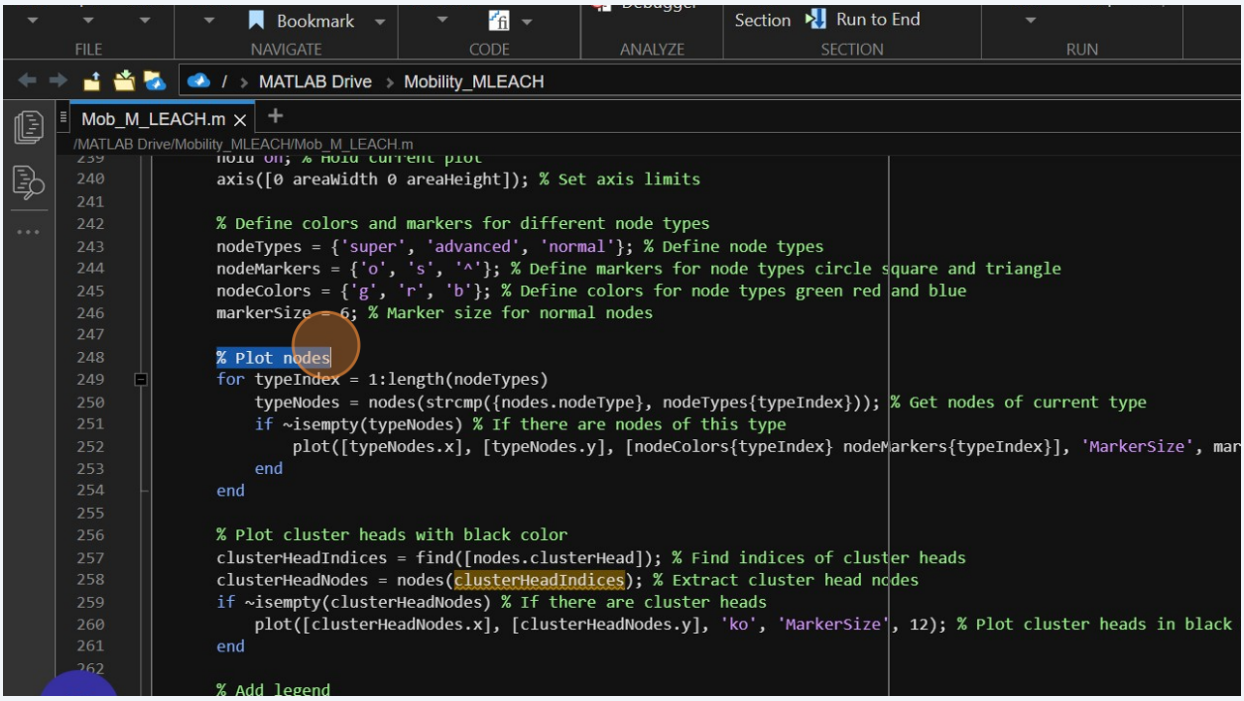
• **Iterating over node types**: The loop iterates over each type of node specified in nodeTypes.

• **Filtering nodes**: For each type of node, it filters out nodes of that type using logical indexing based on the node type stored in the nodes structure.

• **Plotting nodes**: It plots the nodes of the current type using the plot function. The x-coordinates of the nodes are given by [typeNodes.x], and the y-coordinates by [typeNodes.y]. The color and marker style are determined by nodeColors{typeIndex} and nodeMarkers{typeIndex}, respectively.

• **Marker size**: The marker size for normal nodes is specified by the variable markerSize.

This section ensures that nodes of different types are plotted with distinct markers and colors, allowing for easy visualization and differentiation in the network simulation.

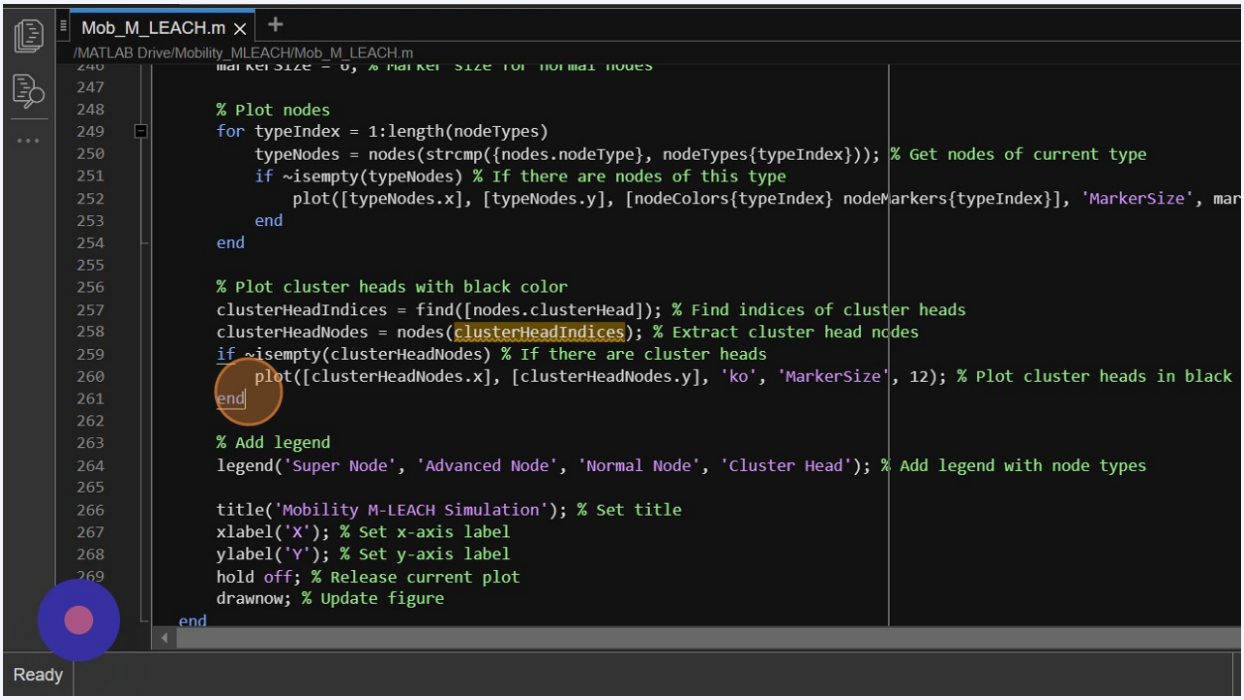


34

• **Find cluster heads**: It finds the indices of cluster heads in the nodes structure by using the logical array [nodes.clusterHead], which contains true for cluster heads and false otherwise. The find function returns the indices of true elements.

• **Extract cluster head nodes**: It extracts the cluster head nodes from the nodes structure based on the indices found in the previous step.

• **Plot cluster heads**: It checks if there are any cluster head nodes. If there are, it plots them using the plot function. The x-coordinates of the cluster head nodes are given by [clusterHeadNodes.x], and the y-coordinates by [clusterHeadNodes.y]. They are plotted as black circles ('ko') with a marker size of 12.



35

• The legend function is used to add a legend to the plot. Each string argument corresponds to a label for the corresponding plot element. In this case, the legend will have labels for 'Super Node', 'Advanced Node', 'Normal Node', and 'Cluster Head'.

title('Mobility M-LEACH Simulation'); % Set title

• The title function sets the title of the plot to 'Mobility M-LEACH Simulation'.

xlabel('X'); % Set x-axis label

• The xlabel function sets the label for the x-axis to 'X'.

ylabel('Y'); % Set y-axis label

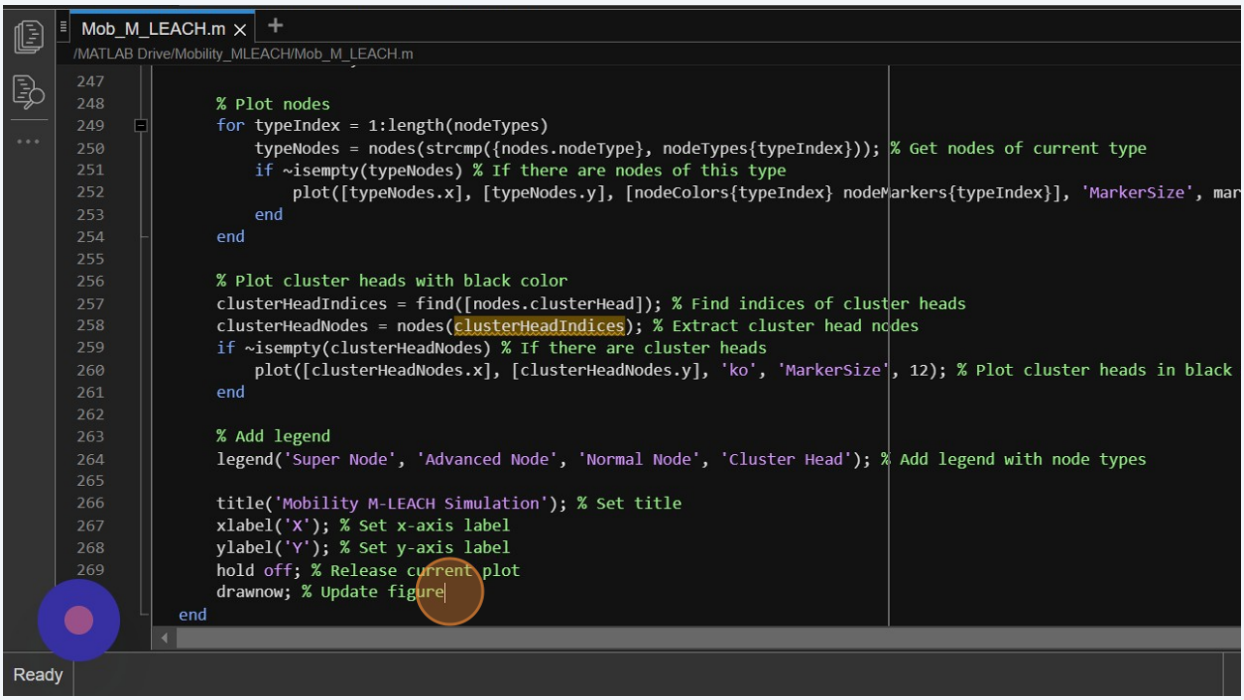
• The ylabel function sets the label for the y-axis to 'Y'.

hold off; % Release current plot

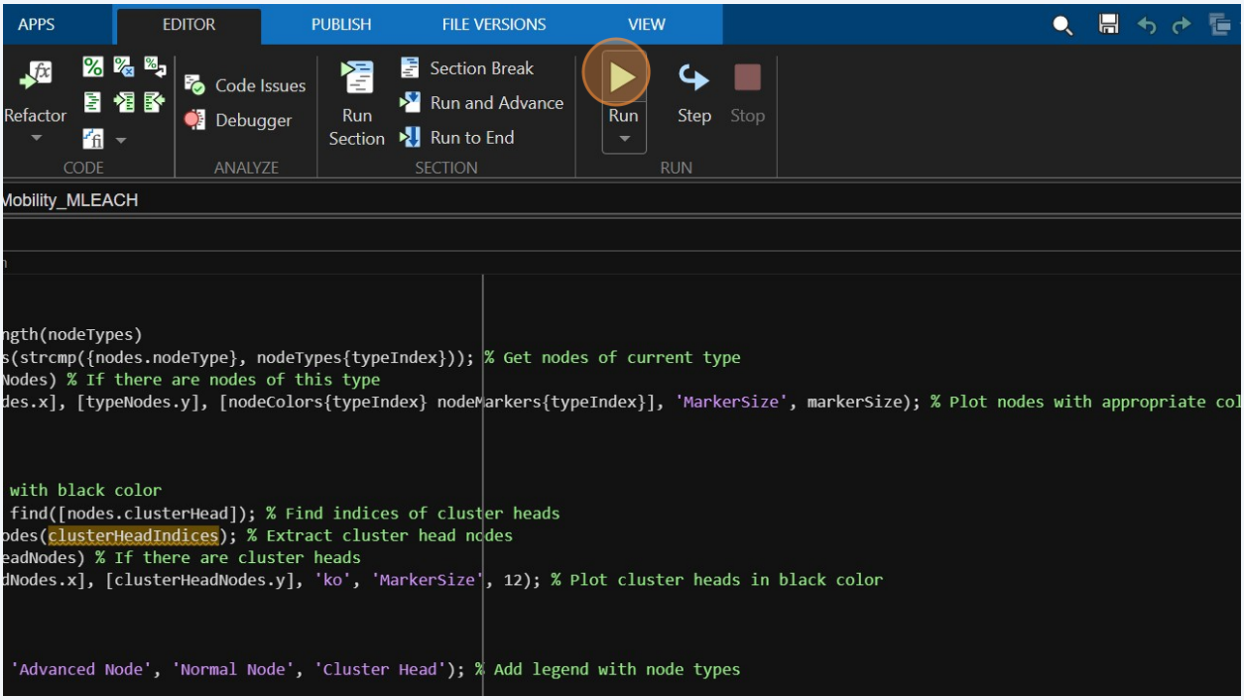
• The hold off command releases the current plot, meaning that subsequent plotting commands will clear the current plot before creating a new one.

drawnow; % Update figure

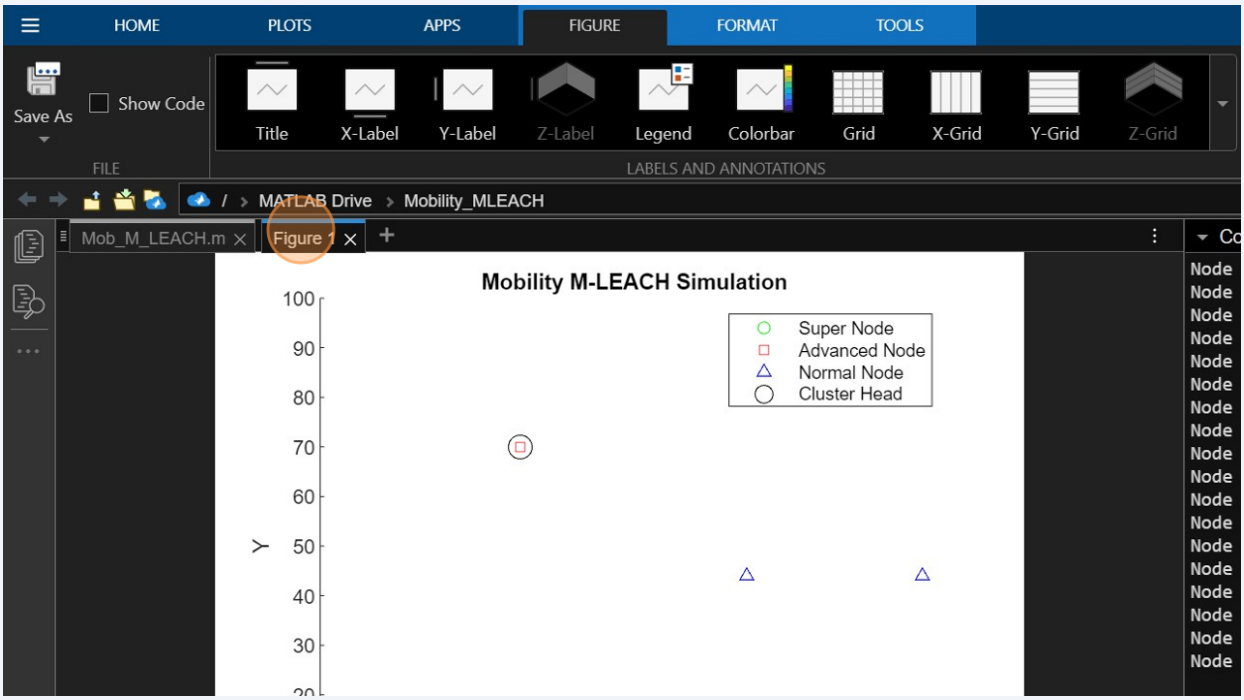
• The drawnow function updates the figure, ensuring that all changes made to the plot are reflected immediately



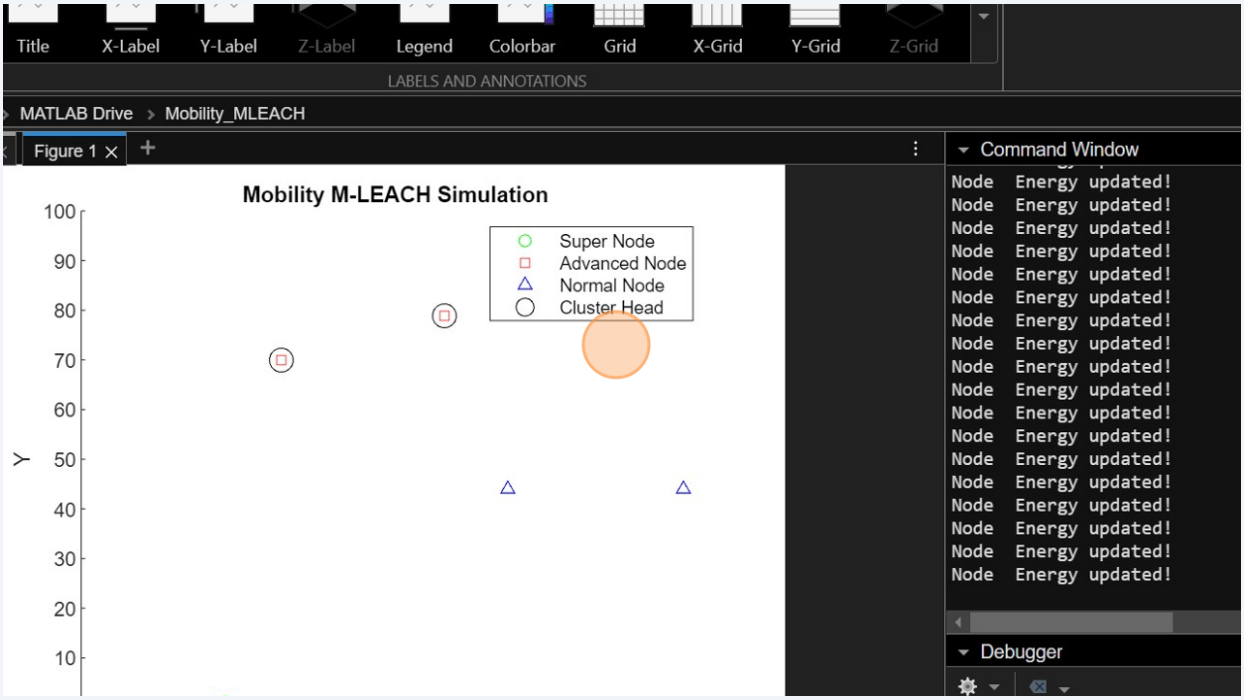
36 Now run it to observe the simulation



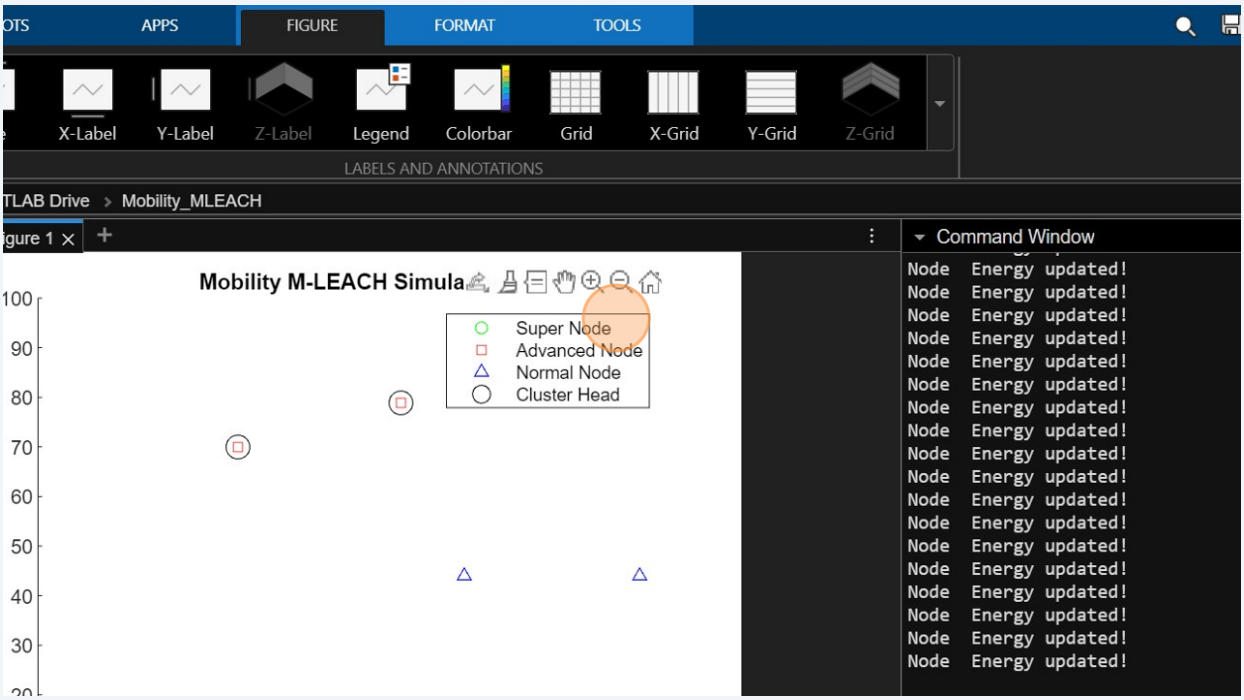
37 Here on "Figure 1" you can see the simulation



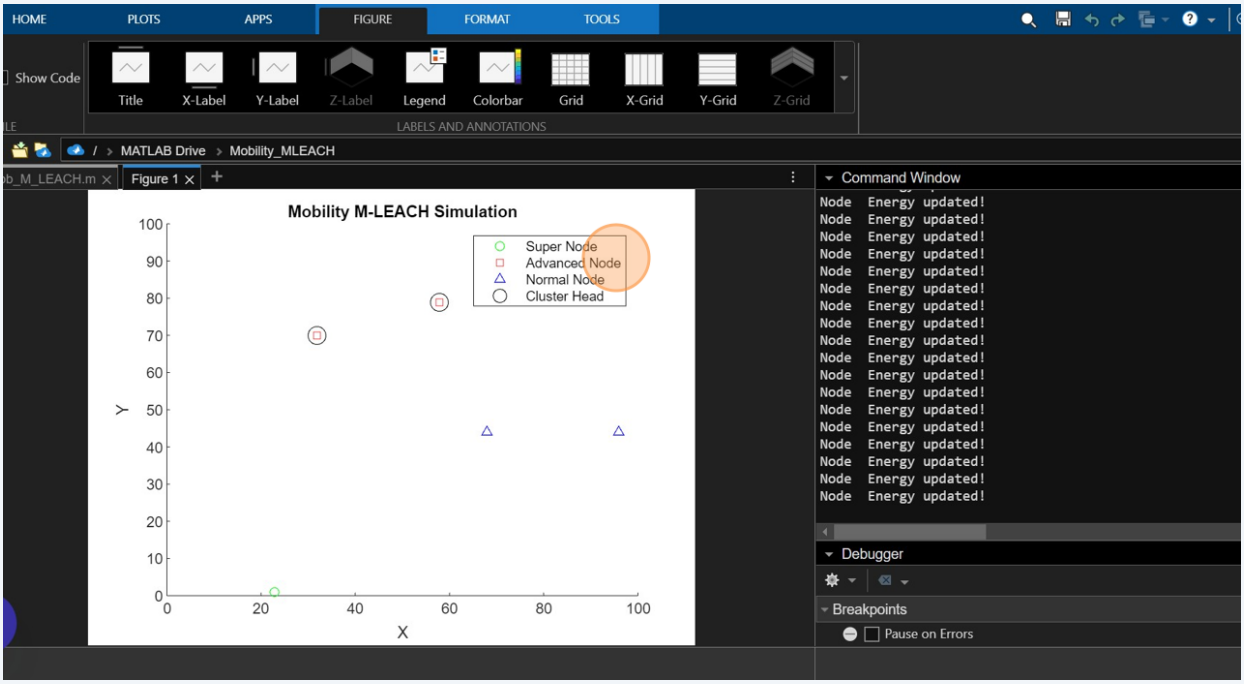
38 Here you can see the legend that has 4 nodes representations and names



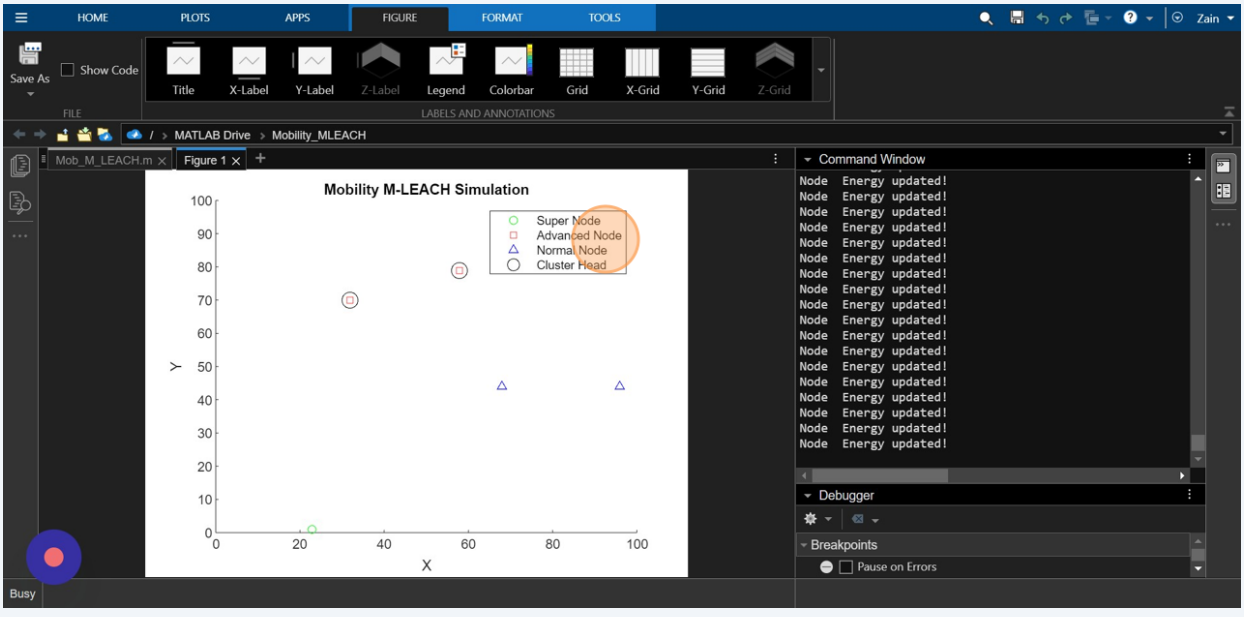
39 As the cluster heads nodes are selected from advance nodes so here the cluster head is drawn over the advance node red square



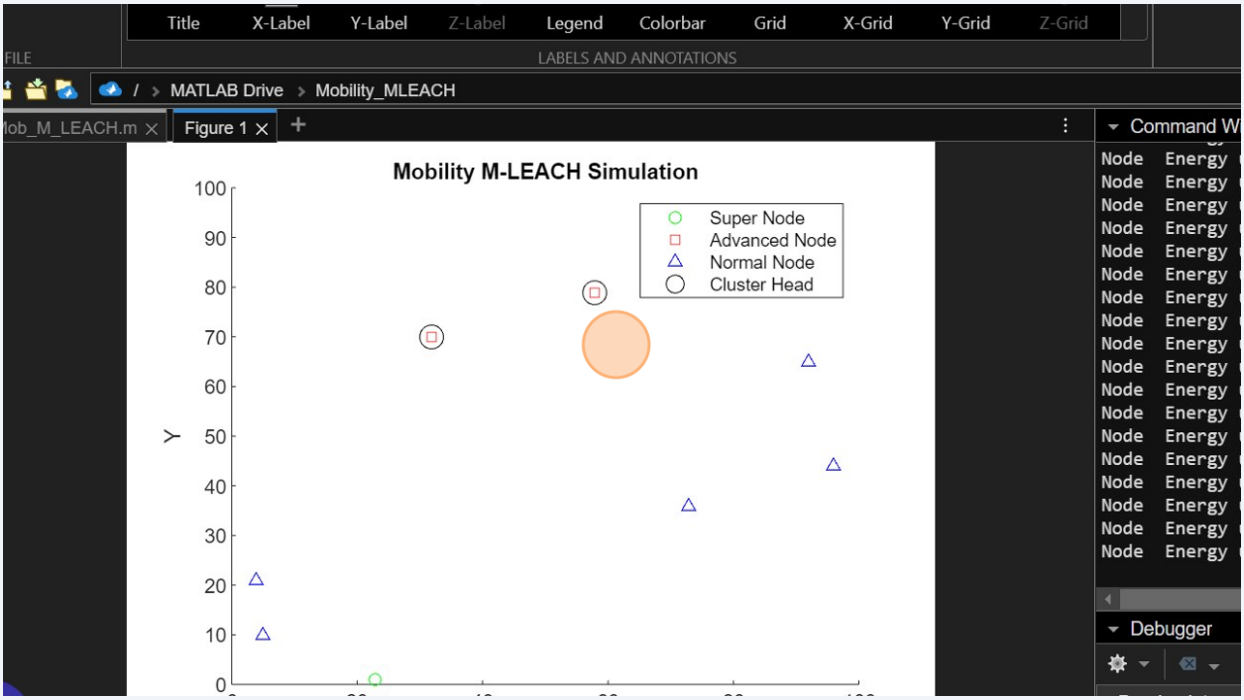
40 On the x axis there is label X and on y axis it is labeled Y
And title of screen is Mobility M-Leach Simulation



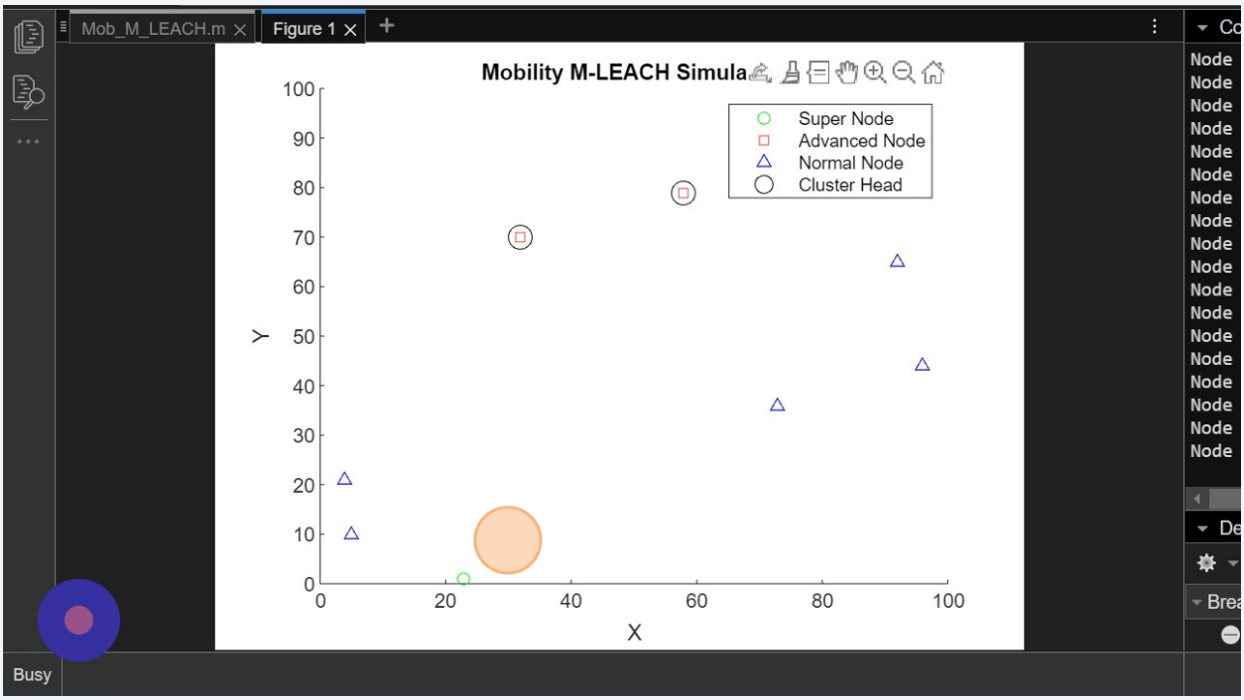
41 The total screen size you can see the width and height both are 100 that we have set in the code perimeters



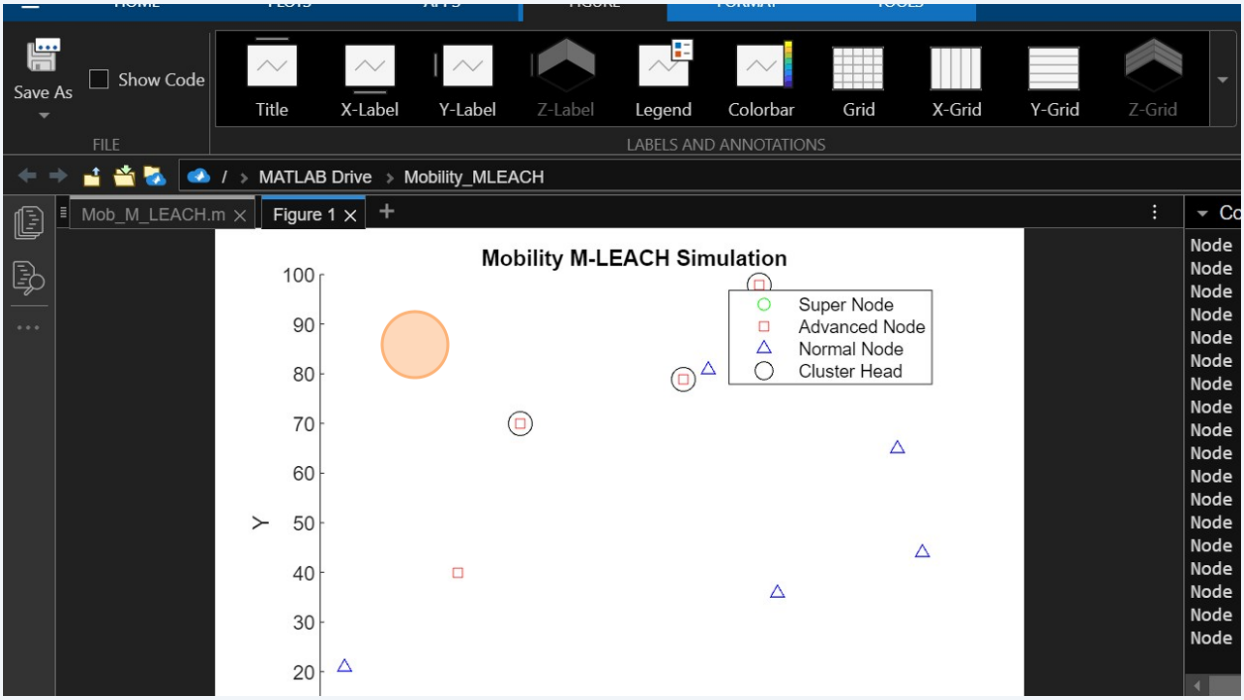
42 Here you can see mobile nodes are moving with each iteration and the number of nodes are according to the perimeters



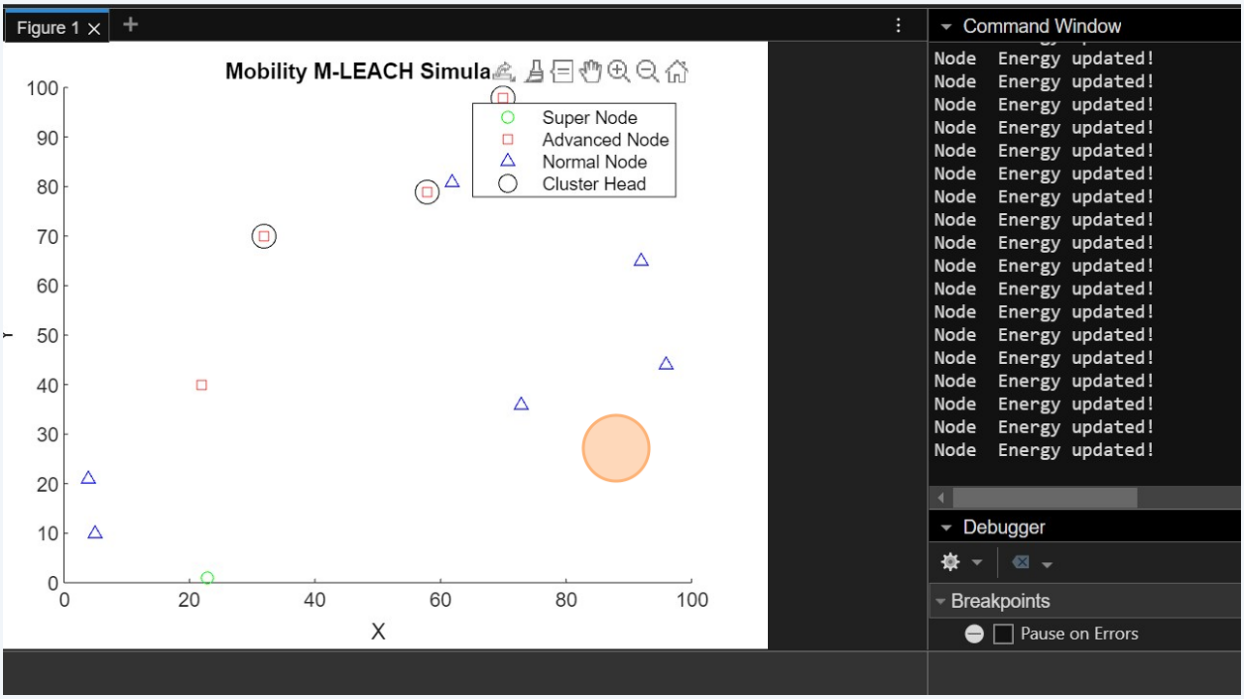
43 Here you can see 1 super node as we have 1 in perimeters



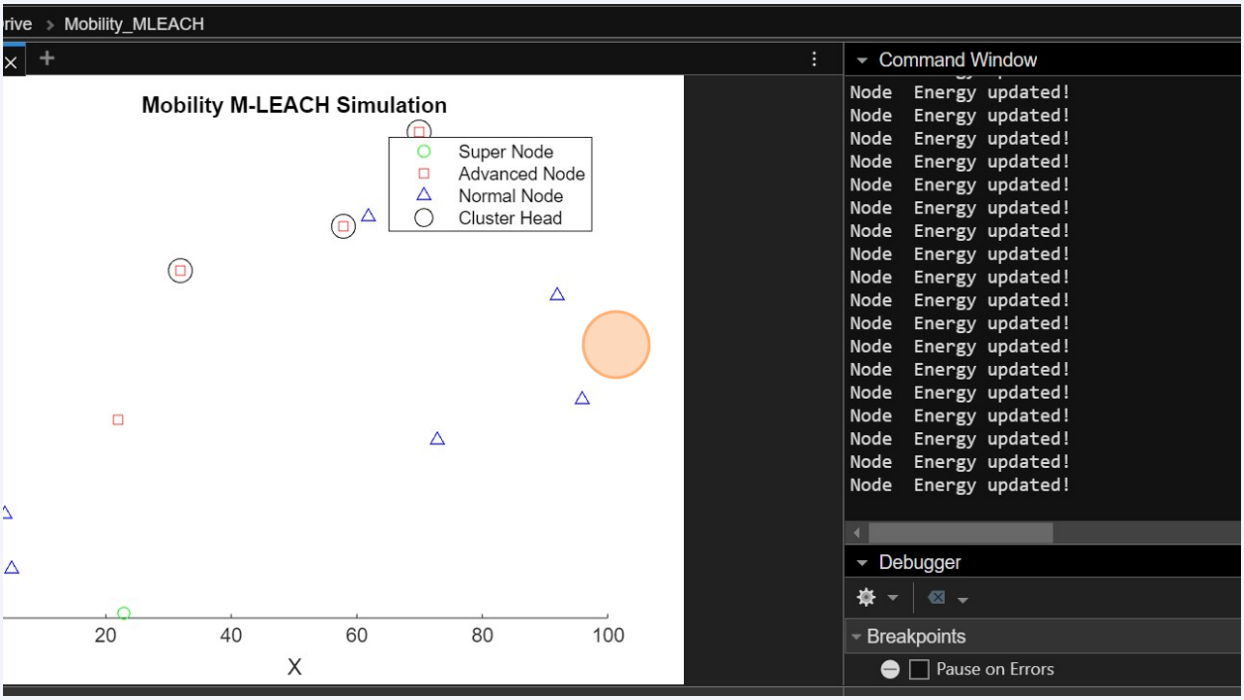
44 Click here.



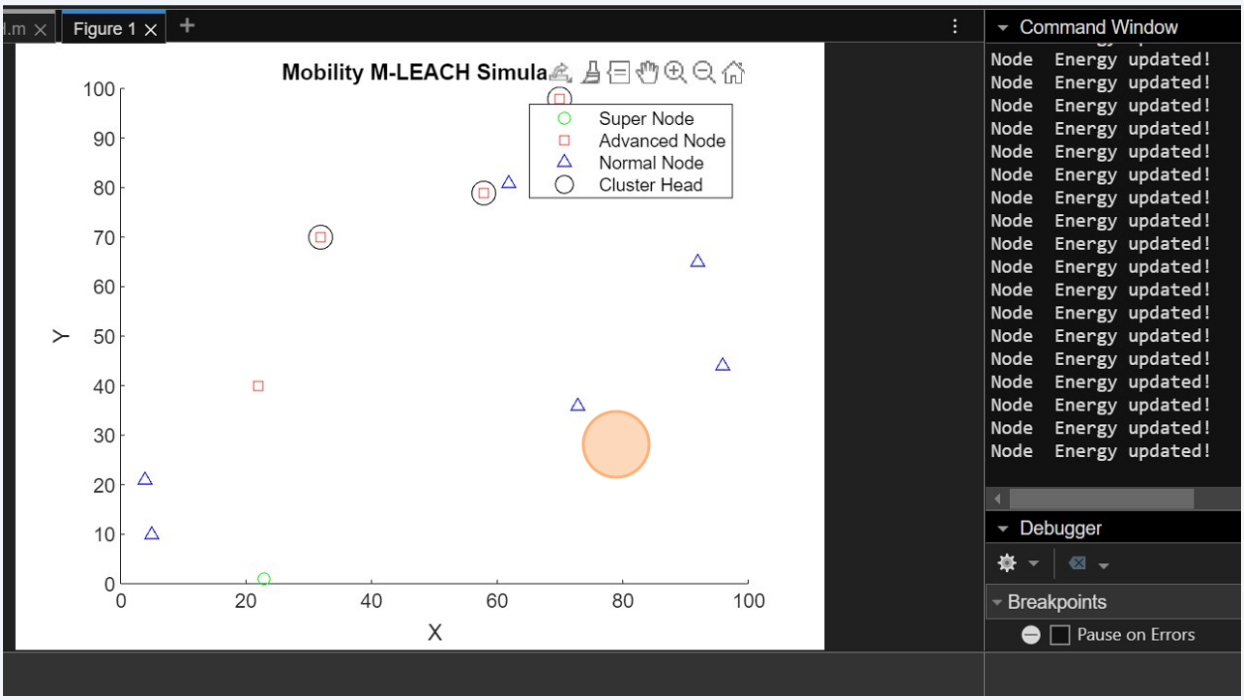
45 Click here.



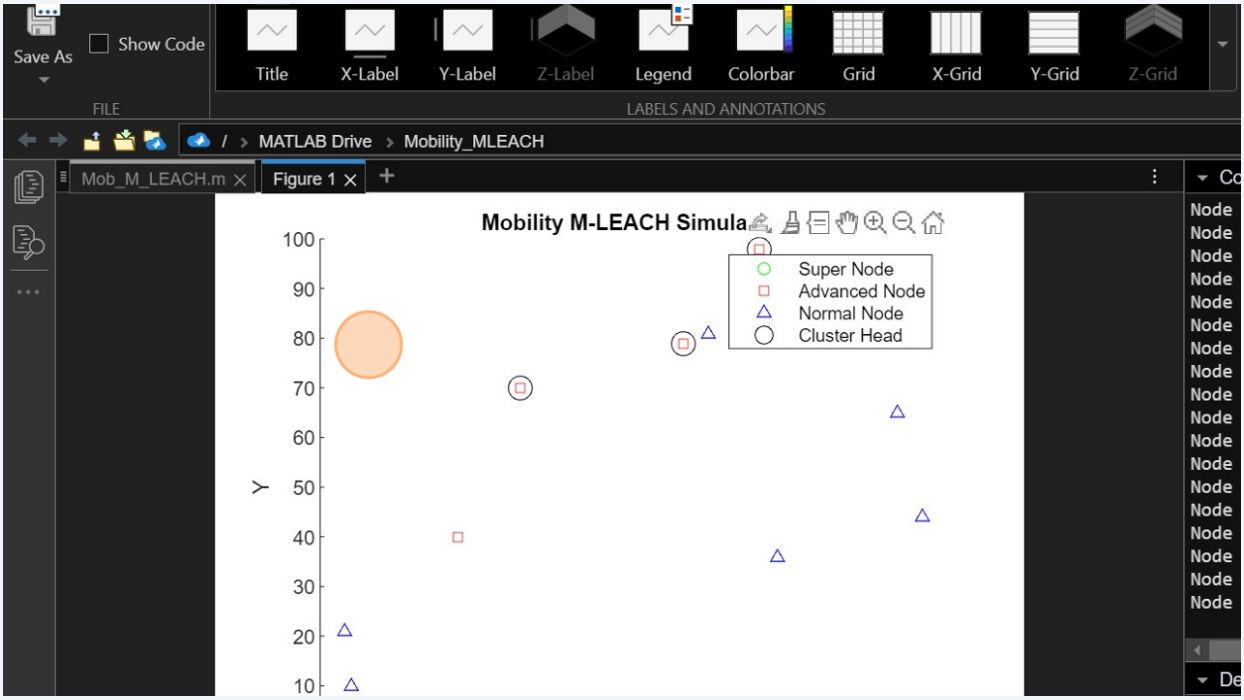
46 Click here.



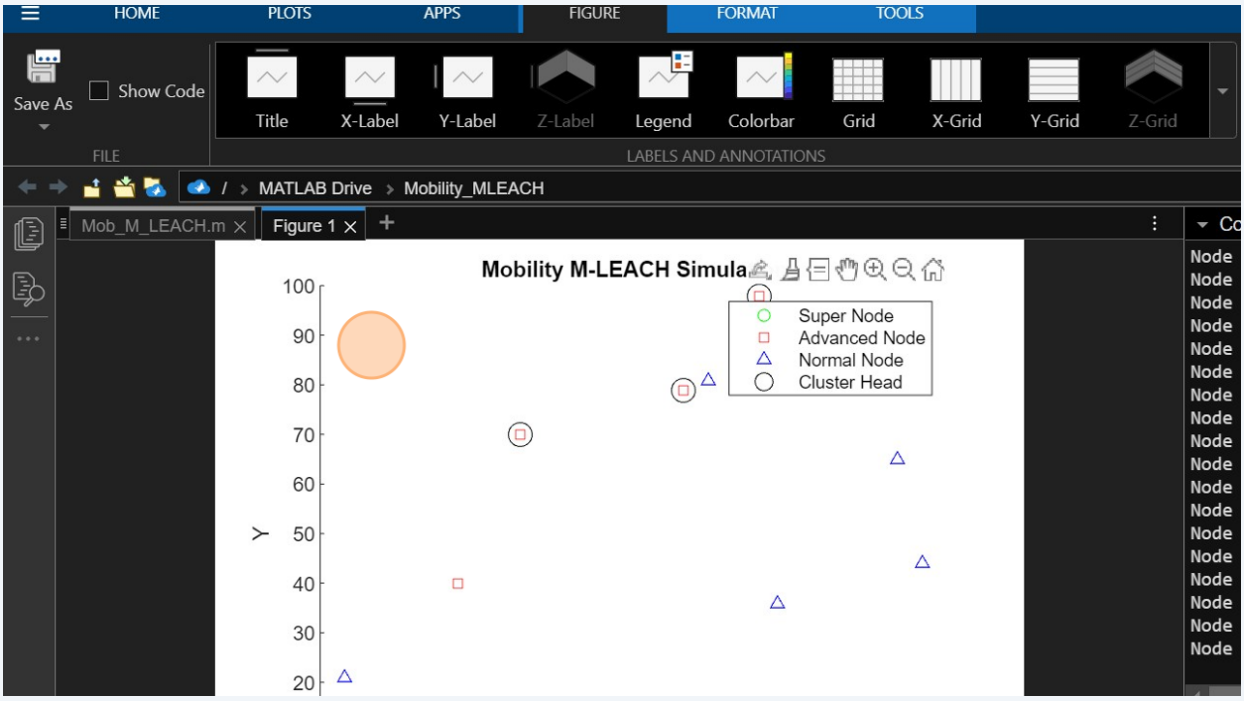
47 Click here.



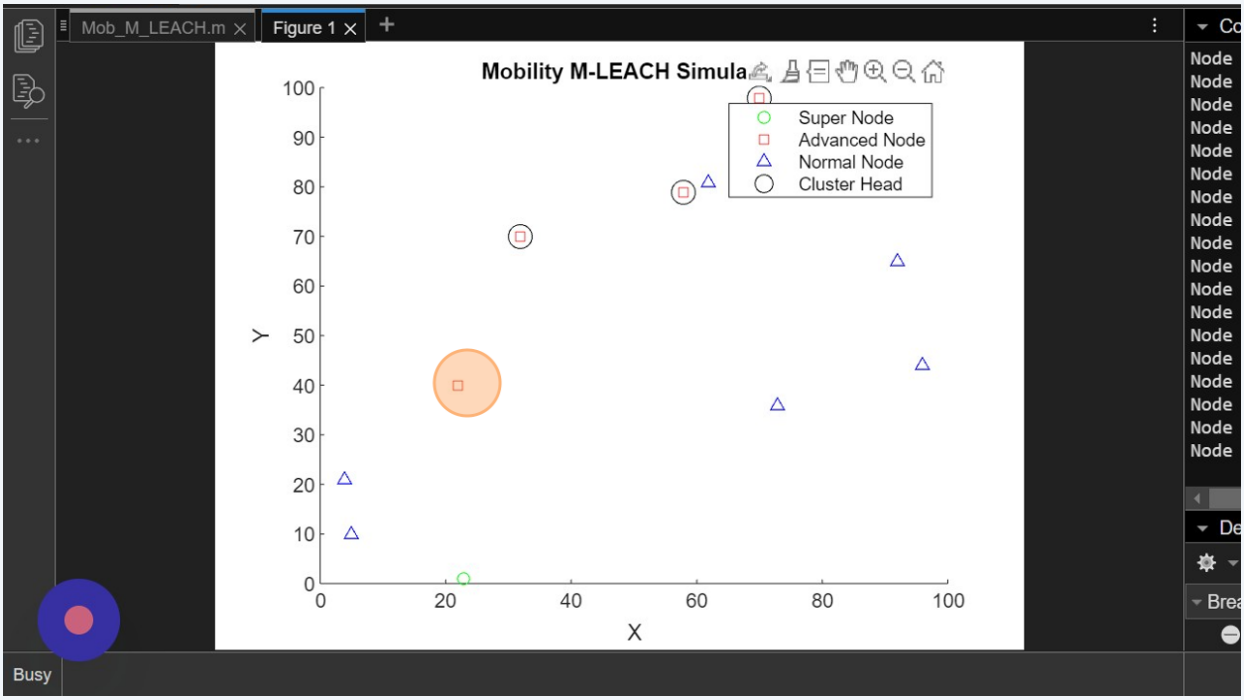
48 Click here.



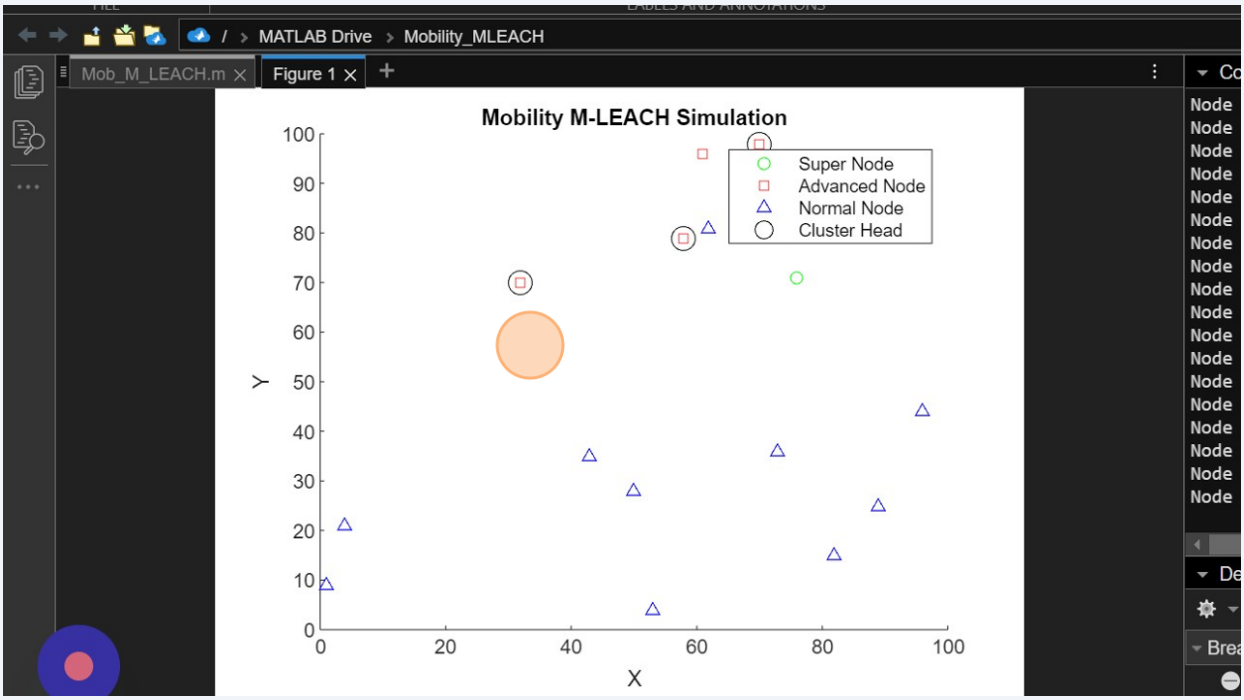
49 Click here.



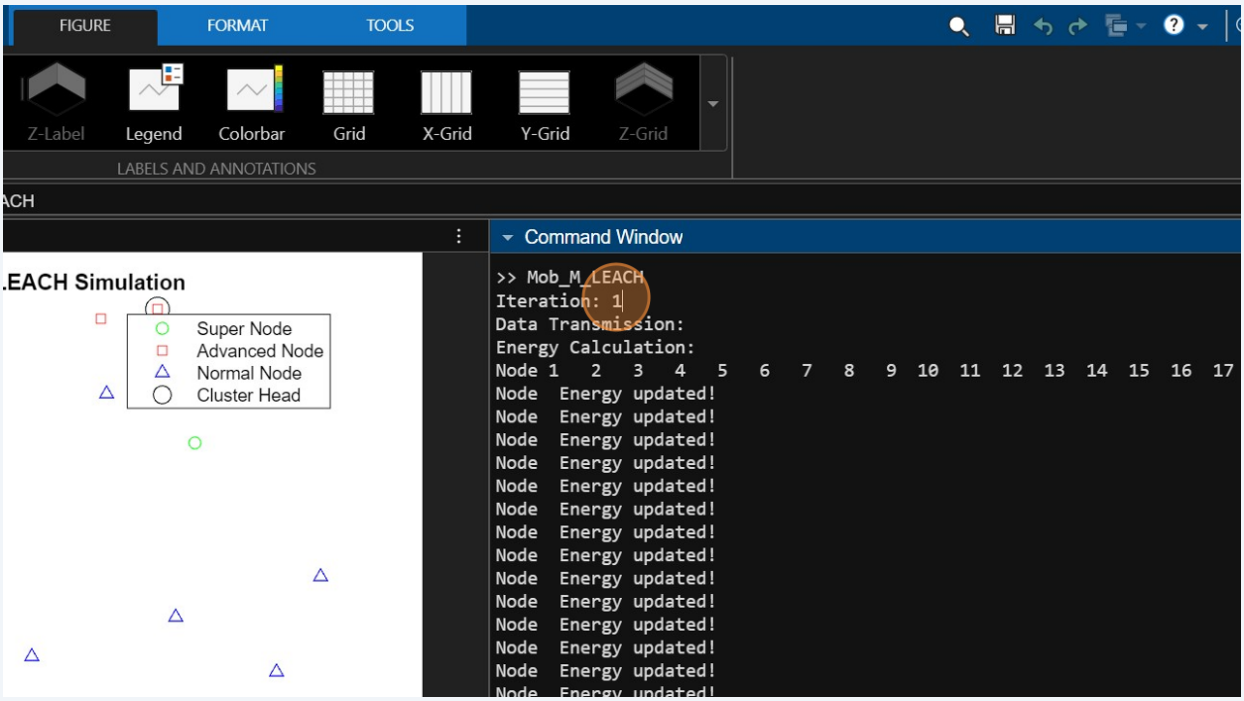
50 Here you can see some advance nodes are not cluster head
Only the selected CH are Encircled Black and denoted as cluster head



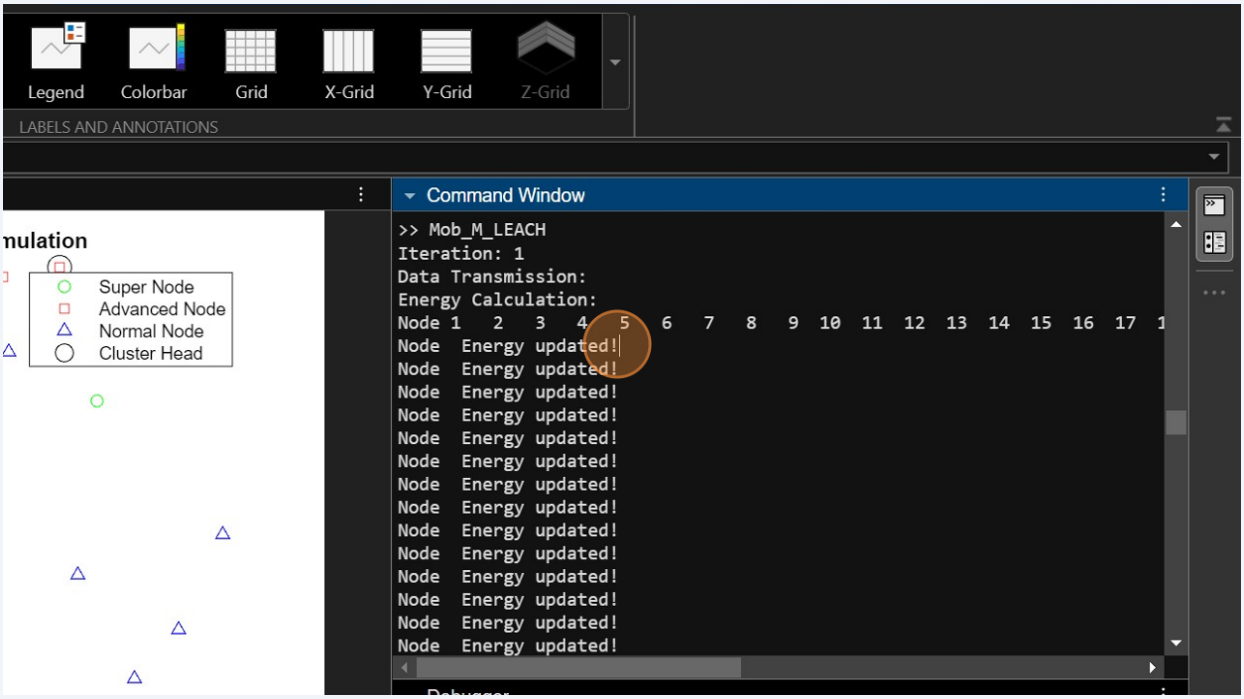
51 Sample output at 100 iterations



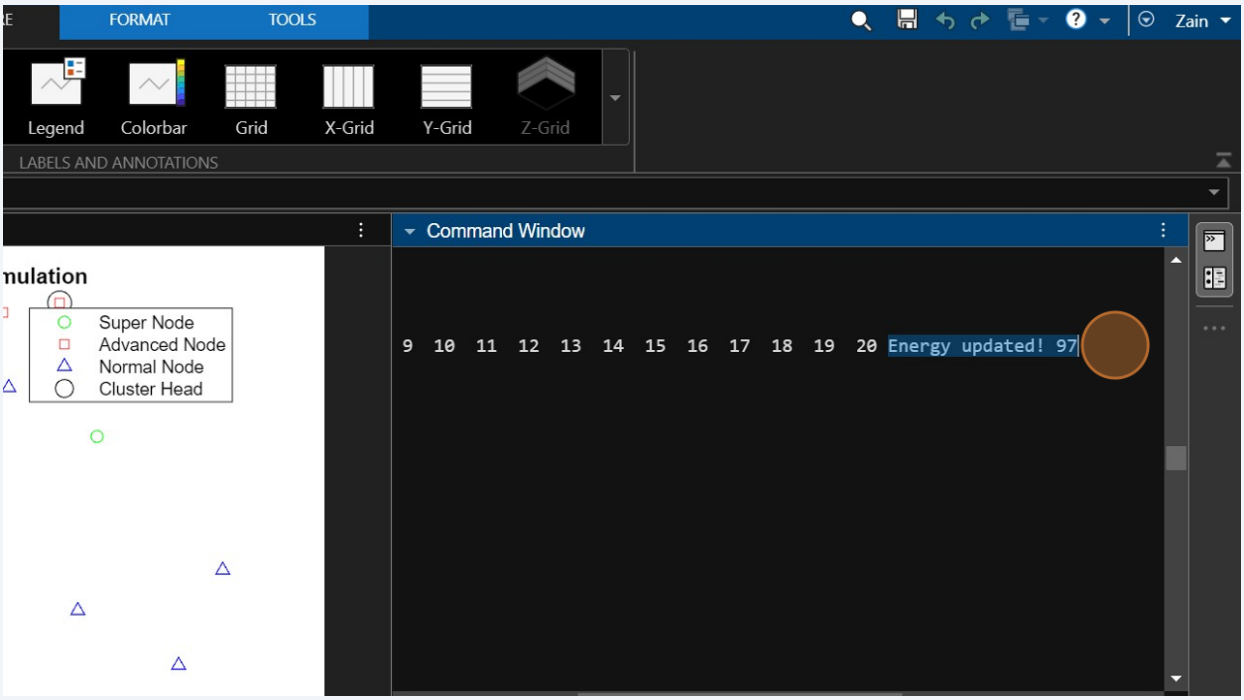
52 Here on Iteration 01:
Data Transmission:
Energy Calculation:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 Energy up..."



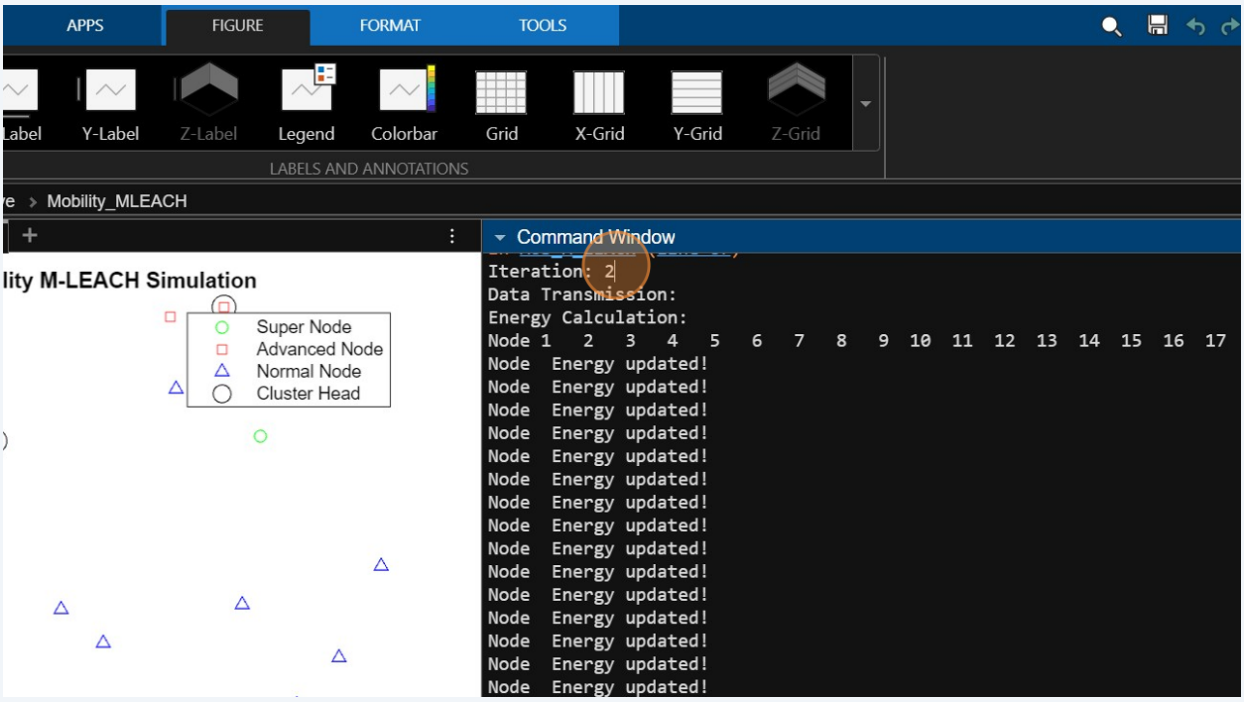
53 Here you can observe the energy updation of each node



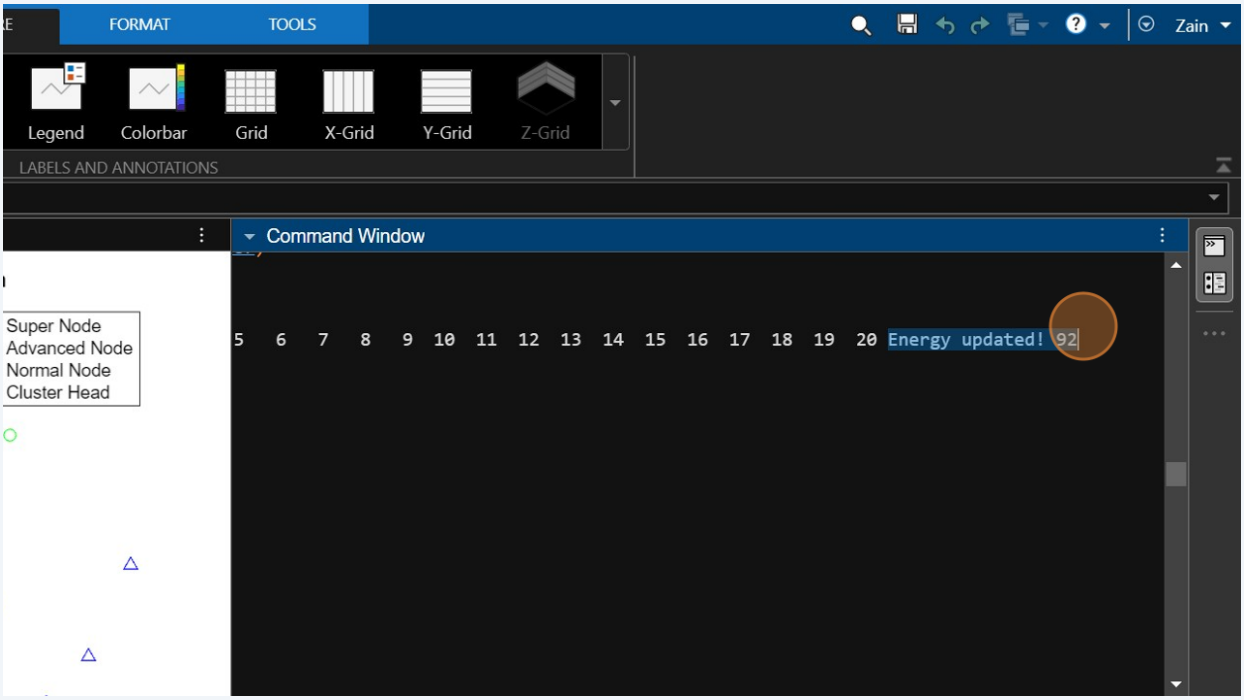
54 Due to mobility you can see the energy dissipation here
as i started with 100 as inital Energy but now it started decreasing with number of iterations



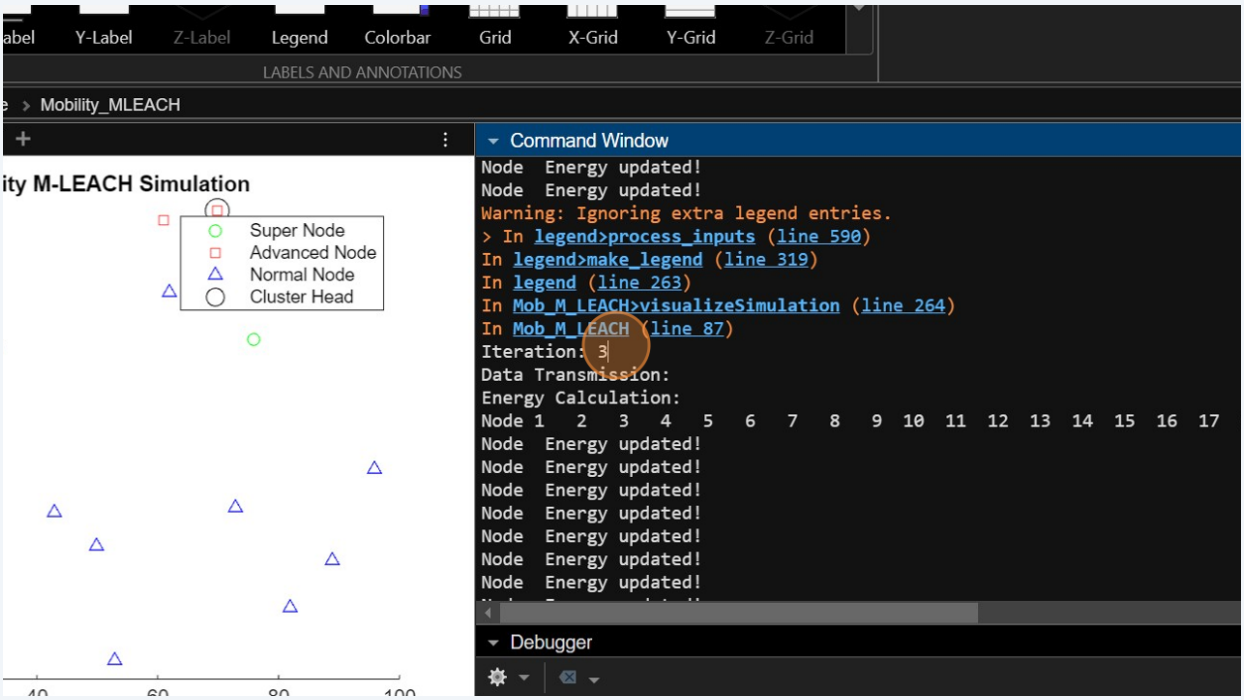
55 Iteration 2



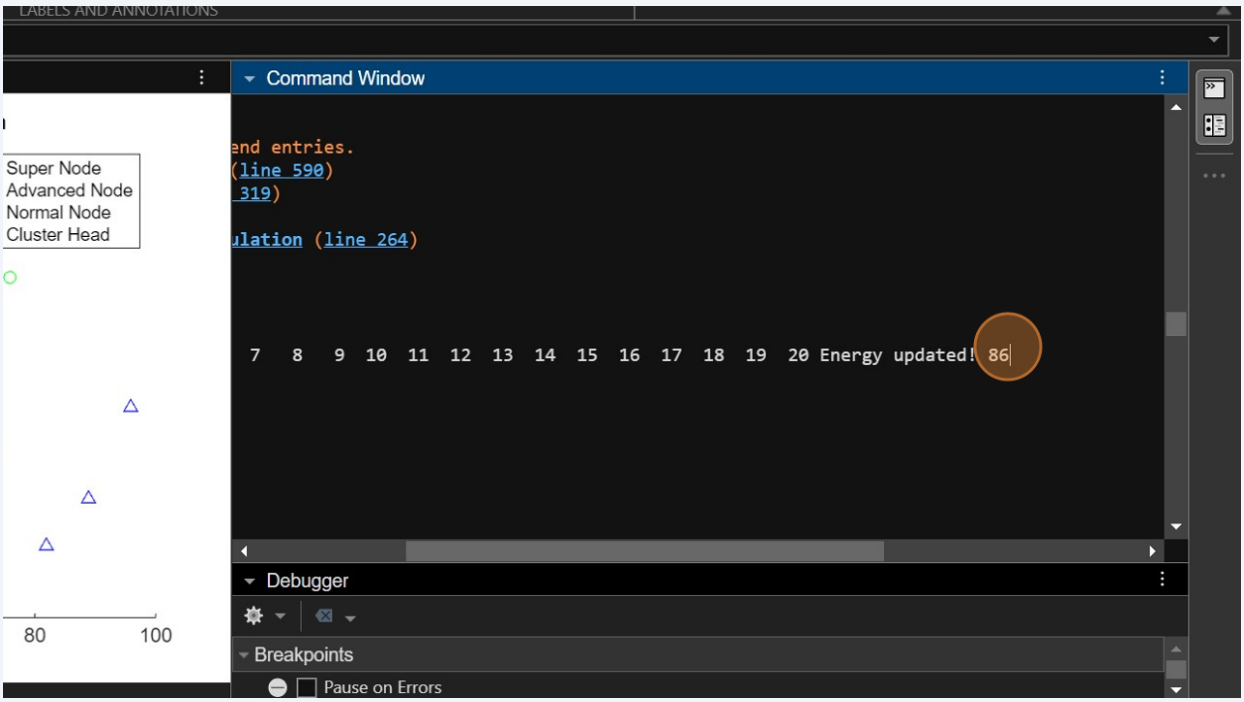
56 Updated energy



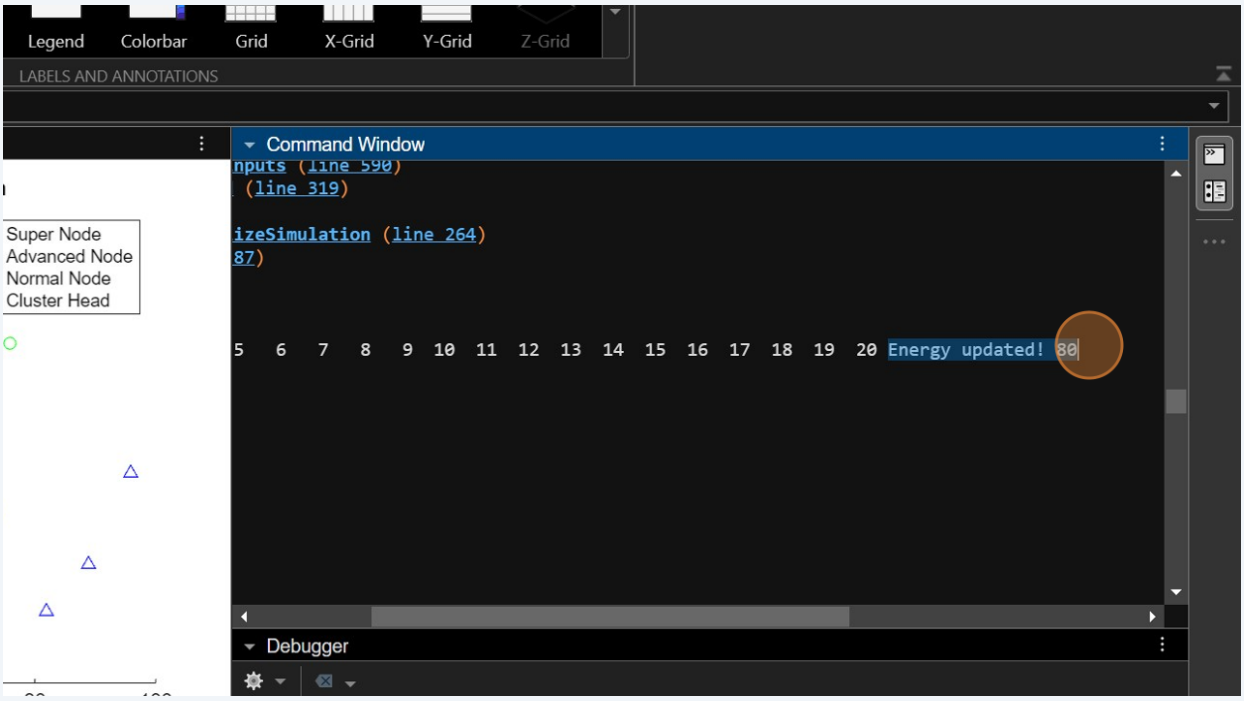
57 Iteration 3



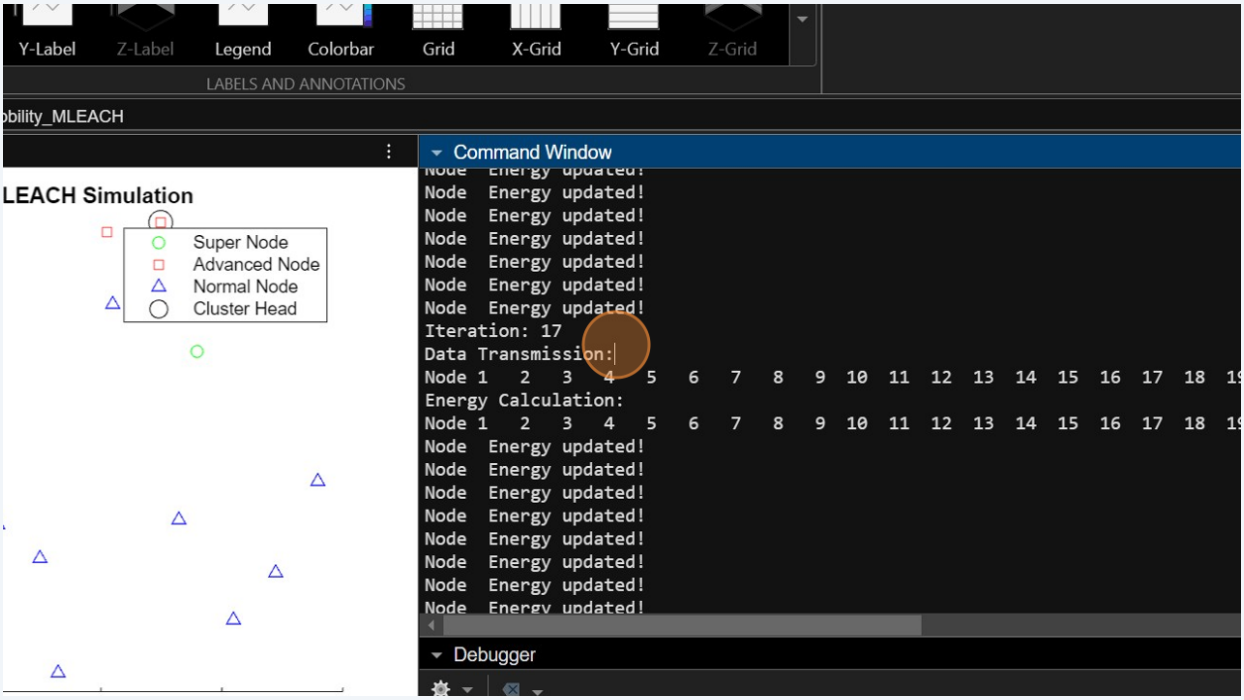
58 Energy updated



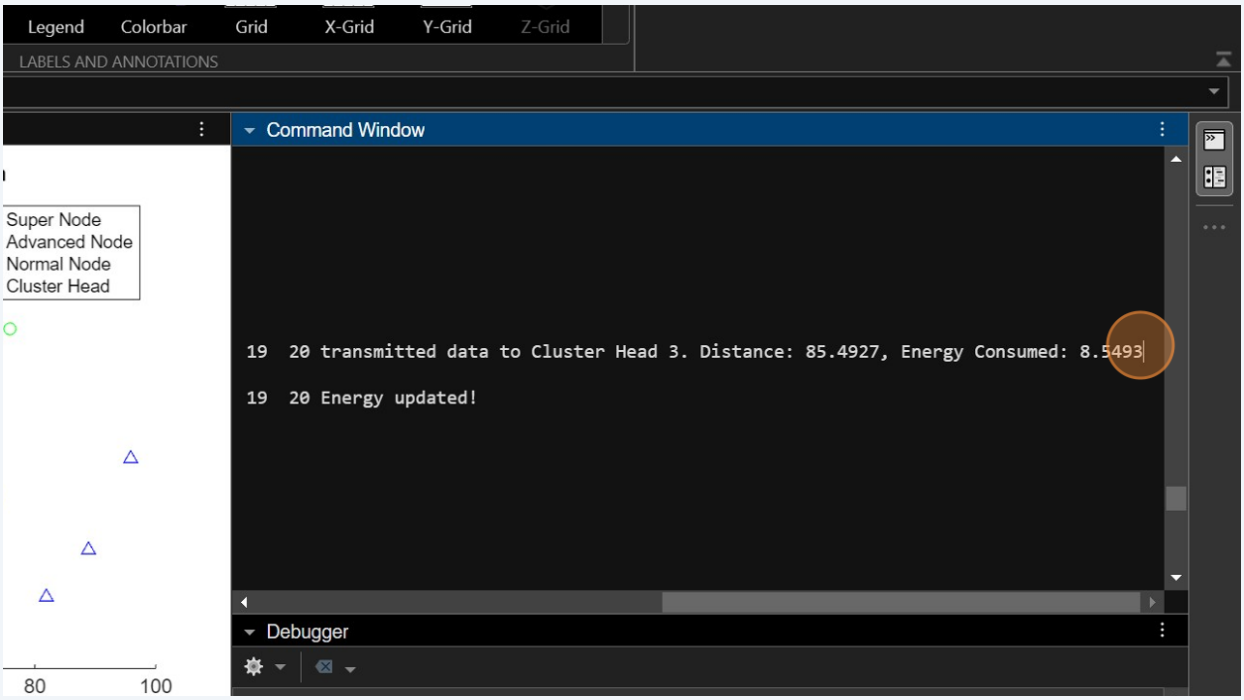
59 Energy keeps on decreasing with the mobility and transmission



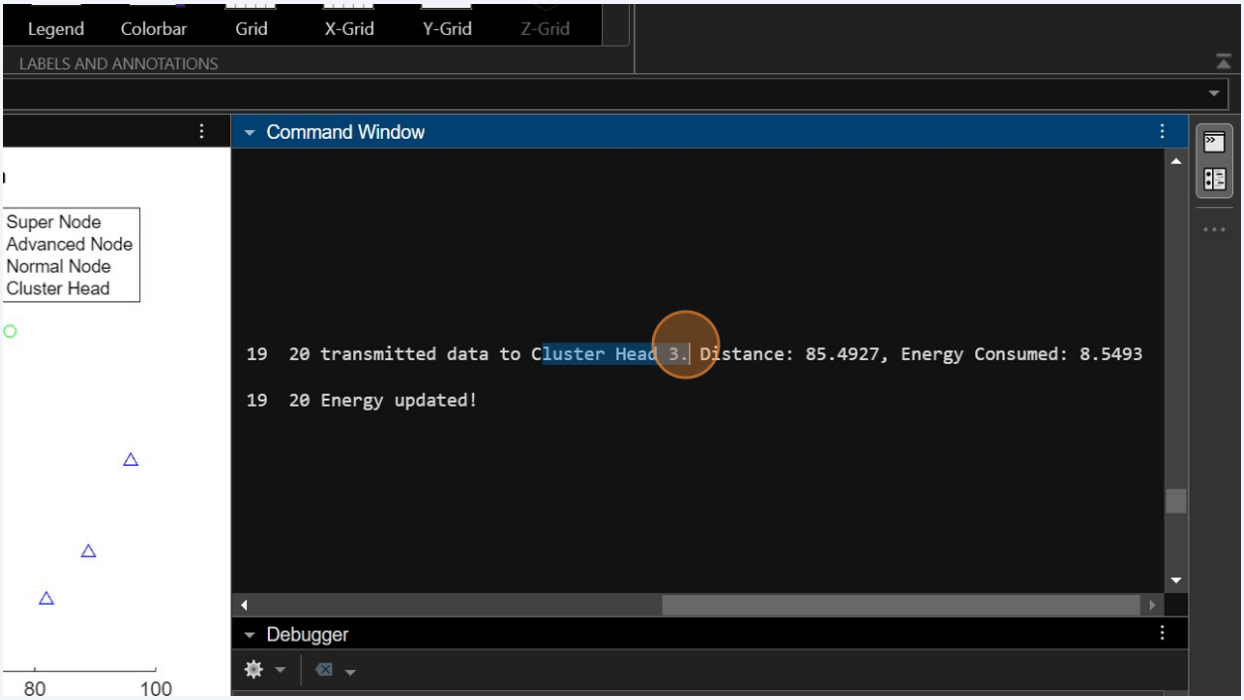
60 Iteration 17



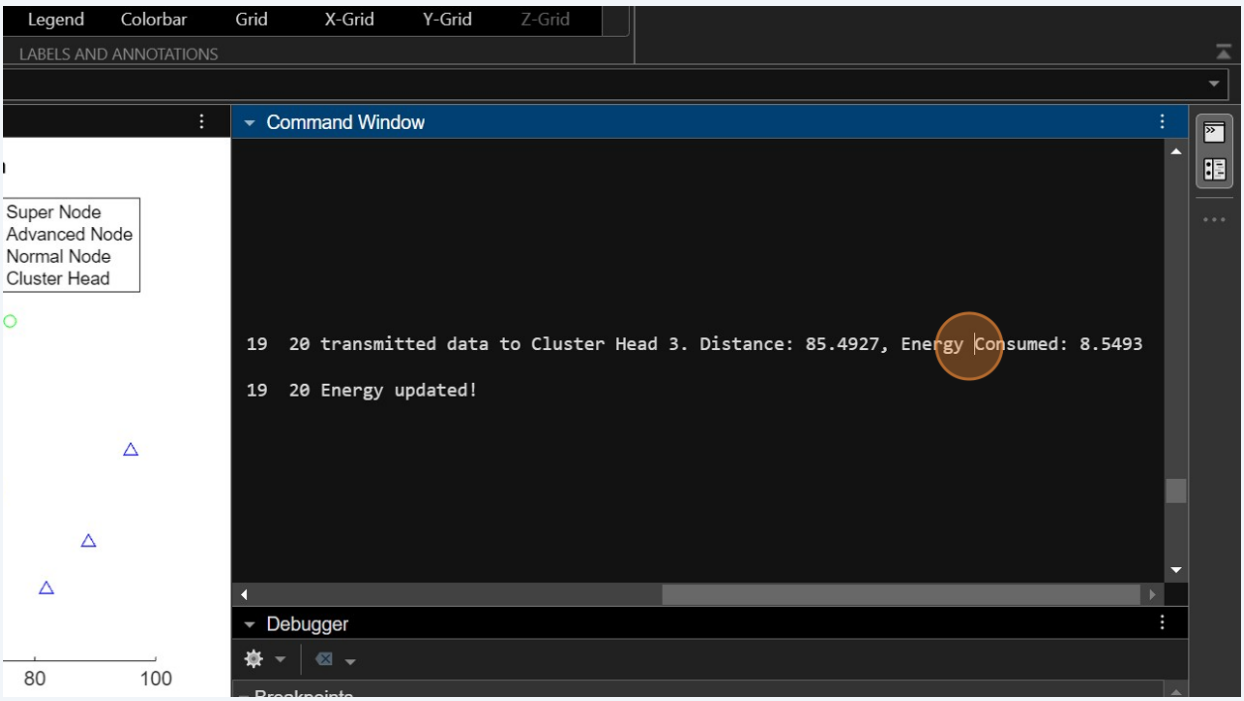
61 Sample output



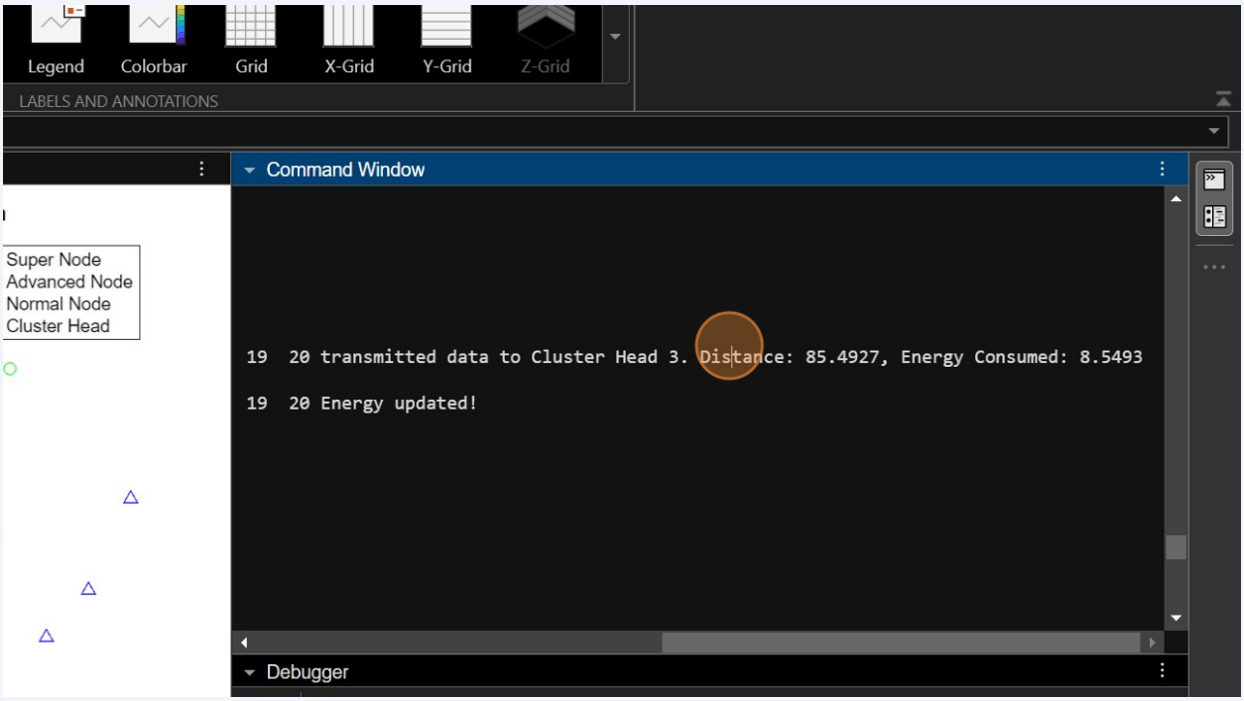
62 The cluster head to whom data is transmitted is also mentioned



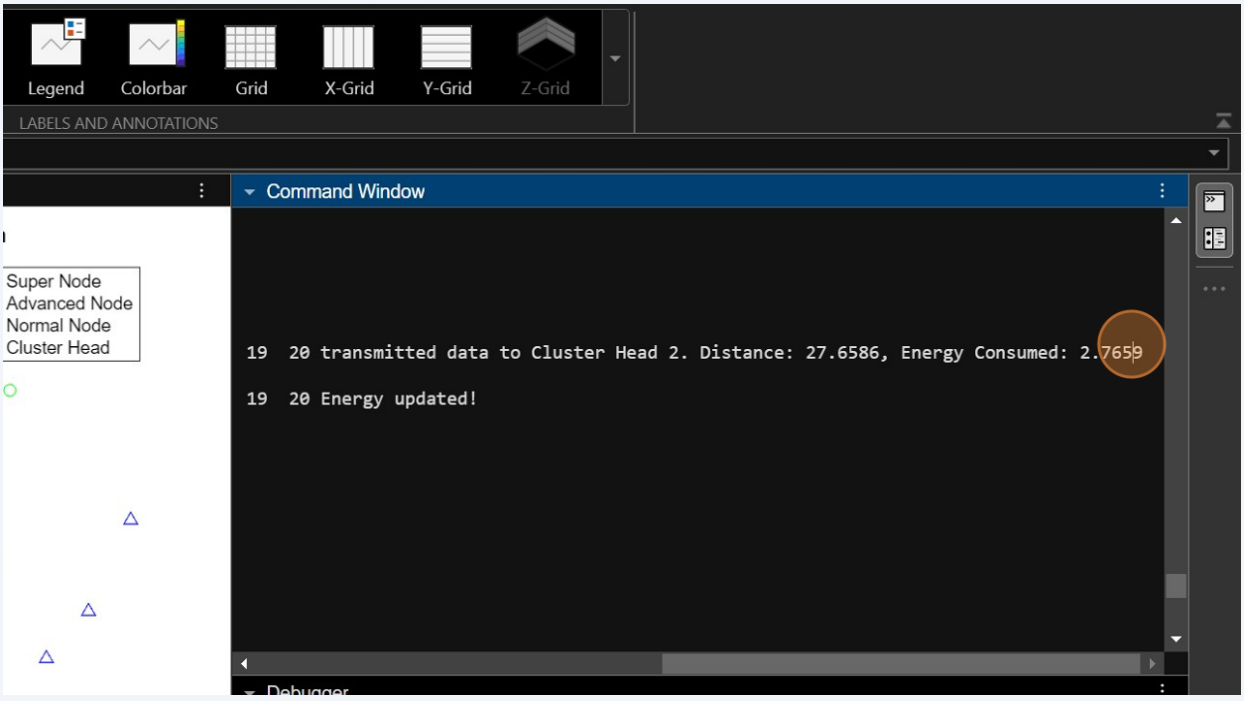
63 On"Iteration: 17
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster ..."



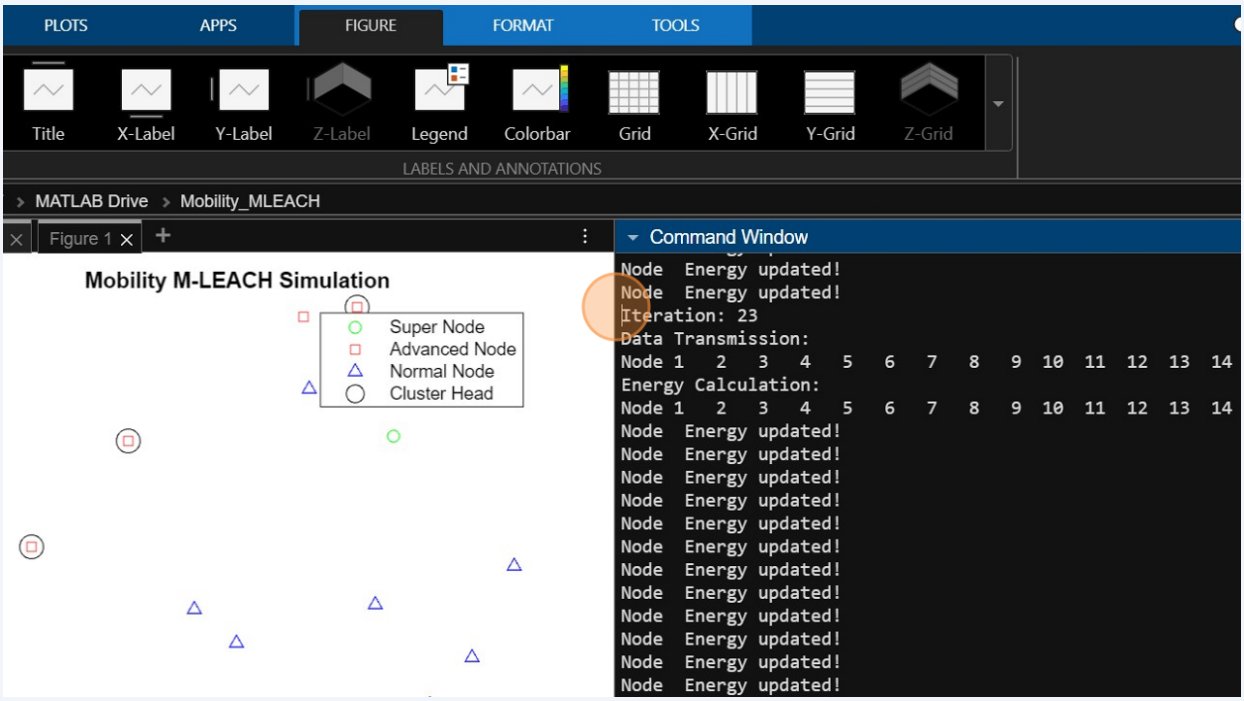
64 On"Iteration: 18
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster ..."



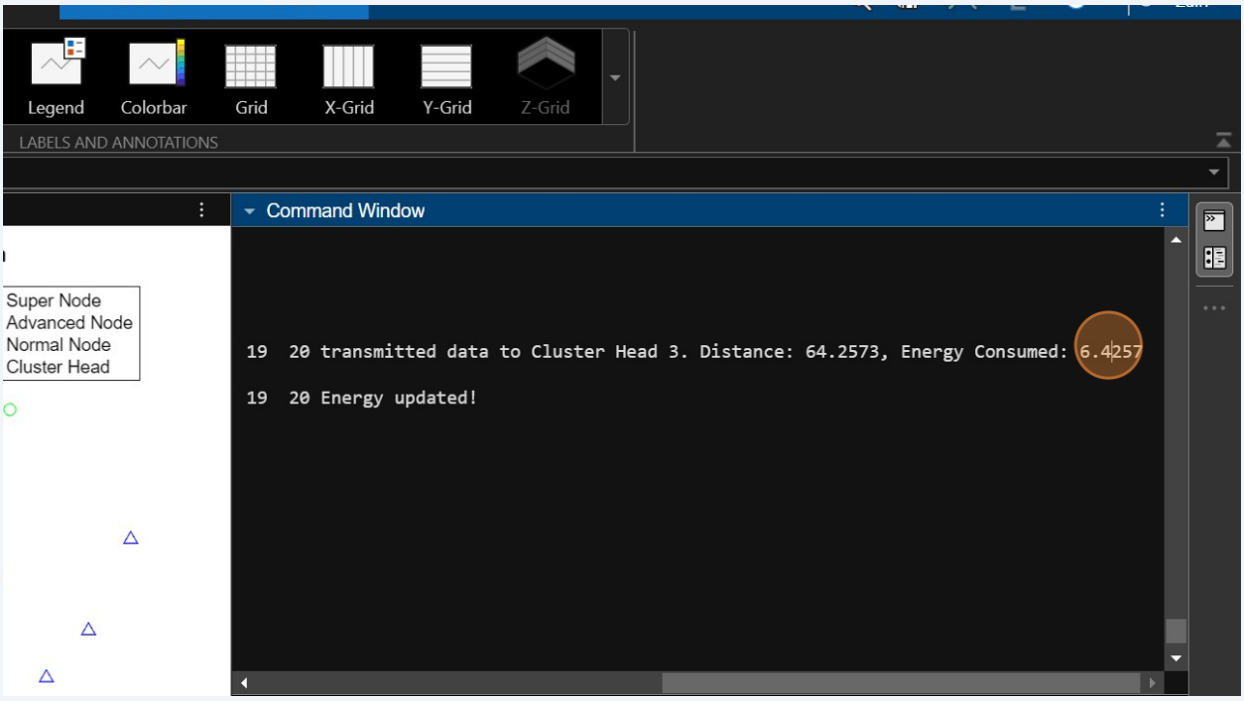
65 On "Iteration: 20
Data Transmission: (Energy COnsumed Distance values calculated



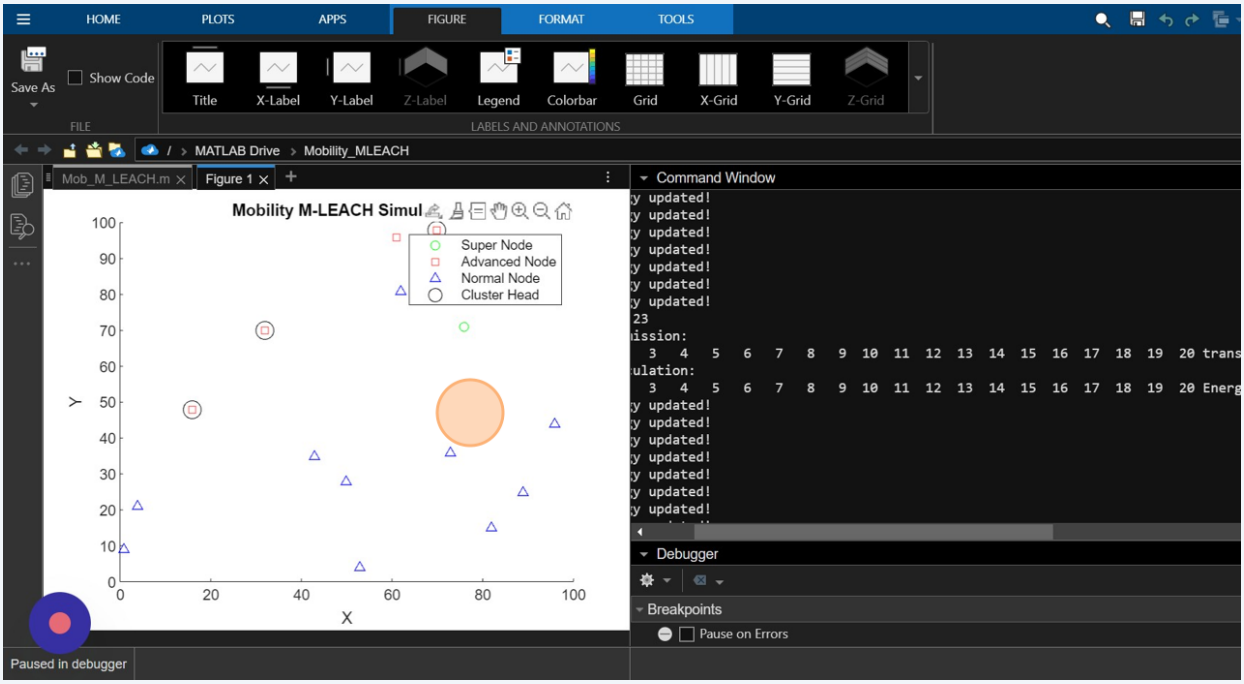
66 Click "Iteration: 23
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster ..."



67 On "Iteration: 23
Data Transmission:
Node 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 transmitted data to Cluster ..."



68 sample output at iteration 23



69

```
% Mobility-MLEACH Implementation in MATLAB with Implemntation of Energy and Mobility Model
% Mobility Induced Multi-Hop LEACH Protocol in Heterogeneous Mobile Network

clear all;

% Parameters
numNodes = 20; % Total number of nodes in the network
numSuperNodes = 1; % Number of super nodes
numAdvancedNodes = 5; % Number of advanced nodes
numNormalNodes = numNodes - numSuperNodes - numAdvancedNodes; % Remaining nodes are normal nodes
numClusters = min(3, numAdvancedNodes); % Number of clusters, minimum of 3 or the number of advanced nodes
areaWidth = 100; % Width of the simulation area
areaHeight = 100; % Height of the simulation area
initialEnergy = 100; % Initial energy level of each node
maxIterations = 100; % Maximum number of iterations for the simulation
RSSI = 10; % RSSI (Received Signal Strength Indication) threshold (signal strength between nodes)

% Initialize Nodes
nodes = struct('id', 1:numNodes, ... % Assign IDs to nodes
'x', randi(areaWidth, 1, numNodes), ... % Randomly assign x coordinates within the area
'y', randi(areaHeight, 1, numNodes), ... % Randomly assign y coordinates within the area
'energy', initialEnergy, ... % Set initial energy level
'clusterHead', false, ... % Initialize as non-cluster heads
'clusterID', 0, ... % Initialize cluster ID
'nodeType', 'normal'); % Set node type as normal initially

% Assign node types
nodeTypes = {'super', 'advanced', 'normal'}; % Define node types
nodeTypeCounts = [numSuperNodes, numAdvancedNodes, numNormalNodes]; % Count of each node type
startIndex = 1; % Start index for node assignment
for i = 1:length(nodeTypes)
endIndex = startIndex + nodeTypeCounts(i) - 1; % Calculate end index for node assignment
[nodes(startIndex:endIndex).nodeType] = deal(nodeTypes{i}); % Assign node types to nodes
startIndex = endIndex + 1; % Update start index for next node type
end
```

```
% Cluster Initialization - Select cluster heads only from advanced nodes
advancedNodeIndices = find(strcmp({nodes.nodeType}, 'advanced')); % Find indices of advanced nodes
if ~isempty(advancedNodeIndices) % Check if there are advanced nodes available
clusterHeadIndices = advancedNodeIndices(randperm(length(advancedNodeIndices), numClusters)); % Randomly select cluster heads from advanced nodes
for i = 1:length(clusterHeadIndices)
nodes(clusterHeadIndices(i)).clusterHead = true; % Mark selected nodes as cluster heads
nodes(clusterHeadIndices(i)).clusterID = i; % Assign cluster ID to selected cluster heads
end
else
disp('Error: No advanced nodes available to select as cluster heads.'); % Display error message if no advanced nodes are available
end

% Simulation Loop
for iter = 1:maxIterations % Iterate through each time step
disp(['Iteration: ' num2str(iter)]); % Display current iteration number

`% Mobility Model (Random Waypoint) nodes = updateNodePositions(nodes, areaWidth, areaHeight); % Update node positions based on mobility model % Energy Model (Simple) - Update energy calculations nodes = updateEnergy(nodes, RSSI); % Update energy levels of nodes % Cluster Formation nodes = leachClustering(nodes, numClusters); % Perform clustering based on LEACH protocol % Data Transmission (Simple) - Display data transmission information disp('Data Transmission:'); for i = 1:length(nodes) if ~nodes(i).clusterHead % If node is not a cluster head clusterHeadID = find([nodes.clusterID] == nodes(i).clusterID & [nodes.clusterHead], 1); % Find the cluster head for the node if ~isempty(clusterHeadID) % If cluster head is found distance = sqrt((nodes(i).x - nodes(clusterHeadID).x)^2 + (nodes(i).y - nodes(clusterHeadID).y)^2); % Calculate distance between node and cluster head energyConsumption = distance * 0.1; % Calculate energy consumption for data transmission disp(['Node ' num2str(nodes(i).id) ' transmitted data to Cluster Head ' num2str(clusterHeadID) ... '. Distance: ' num2str(distance) ', Energy Consumed: ' num2str(energyConsumption)]); % Display data transmission information end end en
```