FINAL PROJECT
WIRELESS COMMUNICATIONS

Prof. Dr. Said E. El-Khamy
Communication & Electronics

Asmaa Gamal Abdel-Halem Mabrouk Nagy
أسماء جمال عبد الحليم مبروك ناجي
15010473

-

Khloud Mousad Mohamed
خلود مسعد محمد الطايفي
18010614

"OFDM Vehicular Car-to-Car Wireless Communication Systems"
With MATLAB Codes & graphs

0. The Contents:

1. **Introduction Of The Signal Design And Coding For High-Bandwidth OFDM In Car-to-Car Communication.**

2. **MOBILE CHANNELS AND OFDM FADING:**
   - **Communication System Description**
   - **Acquisition of channel data**
   - **Doppler spread influence**
   - **Frequency selective OFDM frame fading**
   - **Total SNR**
   - **The Graphical & Mathematical Analysis.**

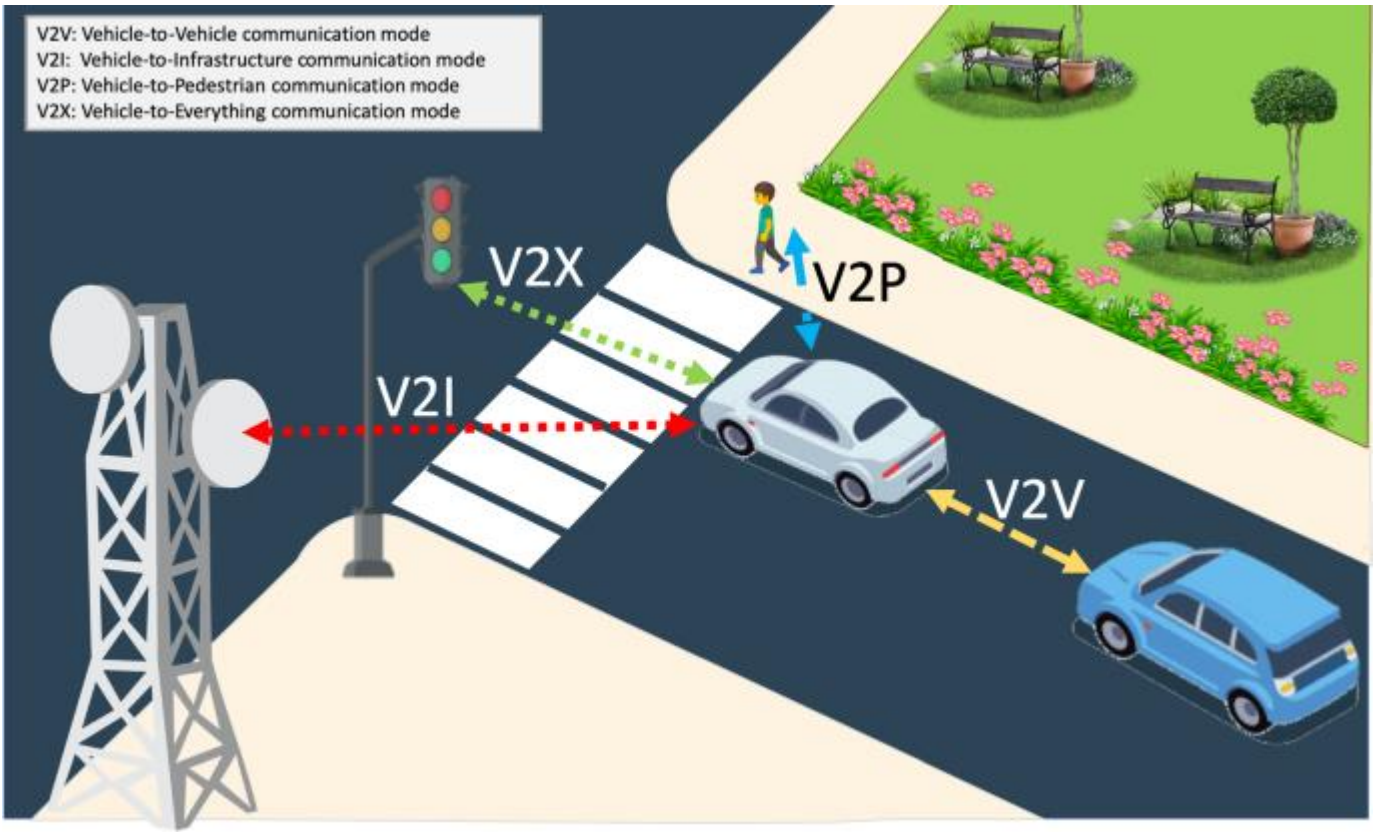
3. **Signal Design Proposal:**
   - **Design Goals.**
   - **Frequency Domain Spreading.**
   - **Channel Coding.**
   - **Results & Conclusion.**

4. **What is V2X and How Does It Work?**
   - **Definition & Introduction.**
   - **How V2X Works?**
   - **DSRC Communication & How It Works.**

5. **MATLAB Codes & Simulations:**
   - **Packet Error Rate Simulation For A Vehicular Channel Using The Protocol Of: IEEE® 802.11p**
   - **Modelling Vehicle-To-Vehicle Communications Using Data Plots for VANET Mobility Simulations & V2V Connectivity MATLAB Codes.**

1. Introduction Of The Signal Design And Coding For High-Bandwidth OFDM In Car-to-Car Communication:

In the domain of traffic safety and intelligent transportation systems, two technologies are playing a lead role: radar systems, which provide additional sensory information about the surroundings including the other traffic, and car-to-car communication (C2C) systems, by which means vehicles can exchange information about traffic events, road safety information or manoeuvre intent. In current research project the authors are analysing the means to merge these technologies. The ultimate goal of this project is to perform radar measurements and inter-vehicle communications with a single hardware and signal design, thus sharing hardware and frequencies between the two systems. The proposed system uses OFDM signals for radar and communications and operates in the 24 GHz ISM band. Sturm et al. describe the basics of the radar measurement.



We present a detailed analysis of the physical OFDM parameter choice. We analyse the radar system from an estimation theoretic point of view and give some answers to the signal-to-noise ratio (SNR) performance of the radar, further restraining the OFDM parameter choice. The choice of the signal bandwidth is a very important aspect in the system design. For the radar to work satisfactorily, the bandwidth must exceed 90 MHz This imposes

problems on the communication system: due to power restrictions1, the receiver must frequently operate with a low power density. Besides, the high bandwidth is not necessary for the communication system itself. As an example, assume one vehicle wishes to relay the information that it is about to perform an emergency brake. The information content is not very high, but it is vital that it gets transferred quickly and reliably. In particular, retransmissions are not desired since they can severely increase the transmission delay. The question arises how a signal with given bandwidth requirements can be designed such that reliability is not affected negatively.

In the proposed system, data is exchanged in form of short frames. Apart from the physical modulation parameters such as sub-carrier distance, we will present the results of a short survey of suitable coding schemes. As an arbitrary limitation, the maximum communication distance for which the high reliability is designed for shall be fixed to 100 m.

2. MOBILE CHANNELS AND OFDM FADING:

• Communication System Description

The analysed communication system uses a high-bandwidth OFDM signal with the parameters denoted in Table I. Every OFDM frame consists of N × M modulation symbols. The goal of the analysis is to first establish which values of SNR can be expected on each of the symbols, and then adapt the frame structure such that reliable data transmission is possible. The identified solution must be able to work if only every U-th sub-carrier is used and the rest are set to zero, in an OFDMA fashion, since this is a possible way this system will be applied. Synchronisation and equalisation issues will not be subject of analysis in this work. We will perform a best-case analysis, assuming that the synchronisation perfectly detects the timing and optimally aligns the centre frequency. In a similar fashion, equalisation at the receiver is assumed to be able to perfectly retrieve the phase. This way, the fading effects can be analysed separately from the issues of imperfect synchronisation or equalisation.

• Acquisition of channel data

The wave propagation channels in the domain of C2C systems are highly complex and are subject to heavy fading. The high number of reflecting objects, in particular other vehicles, leads to multipath propagation, resulting in frequency-selective fading, especially within urban areas. The potentially high velocities of the other participants as well as the continuous changing of the propagation environment lead to a time variance of the channels.

Parameter	Value
Number of carriers N	1024
Number of symbols M	256
OFDM symbol duration T	11 µs
Guard interval fraction G	$1/8$
Total OFDM Symbol duration T_O	$(1+G)T = 12.375$ µs
Total transmit power P	100 mW
Noise Figure NF_{dB}	10 dB
Center frequency f_c	24 GHz
Bits per Modulation Symbol N_b	1 (BPSK)
Number of simultaneous accesses U	8

Table I
OFDM SYSTEM PARAMETERS

Before the signals can be adapted to channels, some information about the channels needs to be acquired. For this work,
we used a RayTracer to create channel data from simulated traffic. This method consists of two steps: first, a traffic scenario is generated, including roadside buildings, vegetation and moving vehicles, among which are the transmitting and receiving vehicle. Every vehicle is assigned a velocity. The position of every vehicle is re-calculated every 10 ms of simulation time, yielding a snapshot. In the second step, the wave propagation between two vehicles is calculated using ray tracing methods. This process returns a list of K propagation paths per snapshot, each with Doppler shift, time delay, attenuation and phase rotation. The received signal r(t) can be calculated from this list according to:

$$r(t) = \sum_{k=0}^{K-1} \underline{a}_k s(t-\tau_k) e^{j2\pi f_{\mathrm{D},k}(t-\tau_k)}$$

where a_k is a complex attenuation factor, τ_k and $f_{D,k}$ are the delay and Doppler shift of the k-th path, respectively, and s(t) is the transmitted signal. A total of 10 different traffic simulations was analysed, chosen such that a wide variety of traffic situations was included. Among these were two highway scenarios, where vehicles move at high velocities and eight urban scenarios with a variety of traffic and building densities. Any snapshots in which the distance between receiver and transmitter exceeded 100 m were discarded. A total of 10567 snapshots was finally used for analysis, yielding the same number of different channels. It must be noted that the distances between the vehicles are not uniformly distributed; Figure 1 depicts their distribution. Distances ranging between 20 and 50 m occur most of the time, while shorter distances are underrepresented. However, data transmission between physically close

vehicles is usually not very challenging to accomplish, and so this data set allows to focus on ranges with stronger fading.

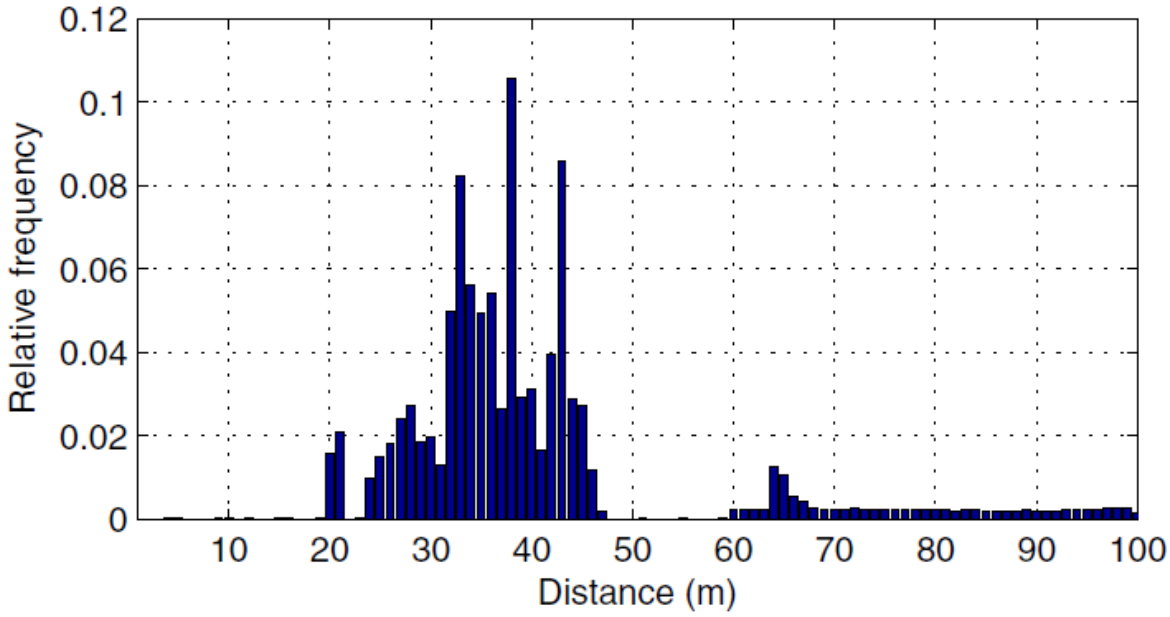


Figure 1. Histogram of vehicle distances in all available snapshots

- **Doppler spread influence**

The first step in calculating SNR at the receiver is to analyse the influence of the Doppler spread. We will largely follow the work of Robertson and Kaiser to get the carrier-to-interference ratio due to Doppler spread and intercarrier interference (ICI) for any given channel.

- **Frequency selective OFDM frame fading**

Unlike the Doppler spread influence, the frequency selective fading must be calculated for every OFDM symbol and subcarrier. The OFDM signal was designed in a manner such that the channel coherence time is larger than one OFDM symbol. As a result, the channel may be assumed time-invariant for the duration of one OFDM symbol and a time-invariant channel impulse response (CIR) is calculated for every OFDM symbol m by fixing the Doppler shifts. The CIR is then sampled at an interval of Ts = T /N, corresponding to the sampling rate of the OFDM system. The time-discrete CIR for symbol m is denoted as

$$h_{l,m} = h(lT_s, m)$$

and further processed by a discrete Fourier transform (DFT) with respect to l, returning the complex attenuation of the m-th OFDM symbol and the n-th sub-carrier:

$$H_{m,n} = \mathrm{DFT}\{h_{l,m}\} \ .$$

The bit energy on the m-th OFDM symbol and the n-th subcarrier is thus:

$$(\mathbf{E}_b)_{m,n} = E_{b,\mathrm{Tx}} |\mathbf{H}_{m,n}|^2.$$

- **Total SNR**

The results from the previous sections enable us to calculate the SNR on every sub-carrier and symbol of an OFDM frame transmitted over a given channel. For brevity, SNR in dB is denoted by the matrix S ∈ $R^{N\times M}$,

$$(\mathbf{S})_{n,m} = 10 \log_{10} \left(\frac{(\mathbf{E}_b)_{n,m}}{(\mathbf{E}_{\mathrm{ICI}})_{n,m} + N_0} \right).$$

- **The Graphical & Mathematical Analysis:**

Figure 2 shows the received SNR for one exemplary OFDM frame. This specific example illustrates the problems that need to be tackled. One can tell SNR varies over a range of several dB for different sub-carriers, from values close to 0 dB, where BER is too high for reliable transmission, up to values greater than 10 dB, where BER is small enough for a reliable communication link. Furthermore, the fading patterns change over time. It is not known a-priori which sub-carriers and which OFDM symbols are reliable, nor can the channel be measured before transmitting due to its rapid variation. The sub-carrier fading does also not necessarily have to vary as much as in the example shown, but can remain fairly constant at a high or low level. To gather some statistics, S was calculated for every generated channel. Figure 3 shows the histogram of all entries of every S out of the 10567 channels.

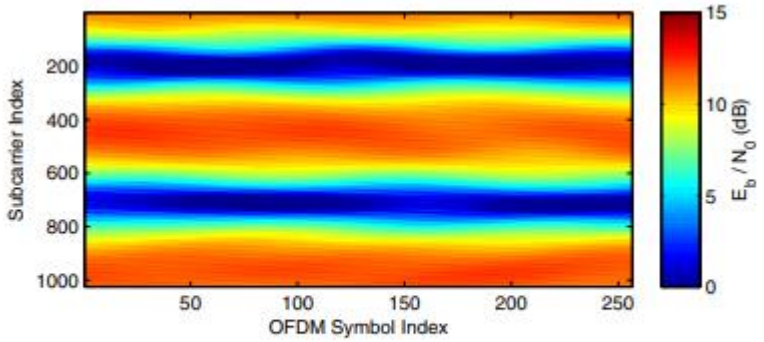


Figure 2. E_b/N_0 distribution over one OFDM frame

In order to understand how the fading fluctuates within one frame, the empirical standard deviation:

$$\hat{\sigma} = \sqrt{\frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} ((\mathbf{S})_{m,n} - \hat{\mu})^2},$$

where ˆμ is the arithmetic mean of all elements of S, was calculated for every snapshot. A cumulative distribution of all standard deviations is shown in Figure 4. Within our set of channels, a standard deviation of 8 dB was never exceeded.

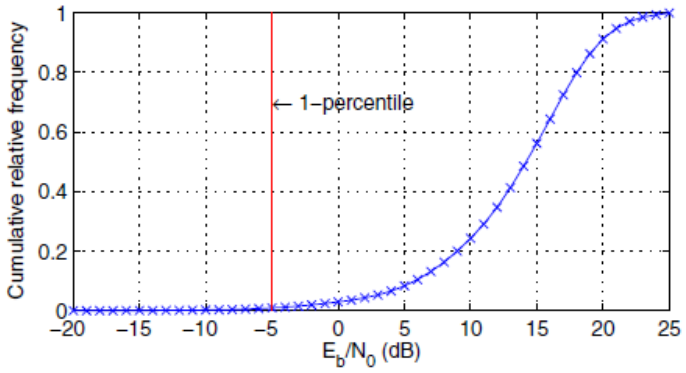
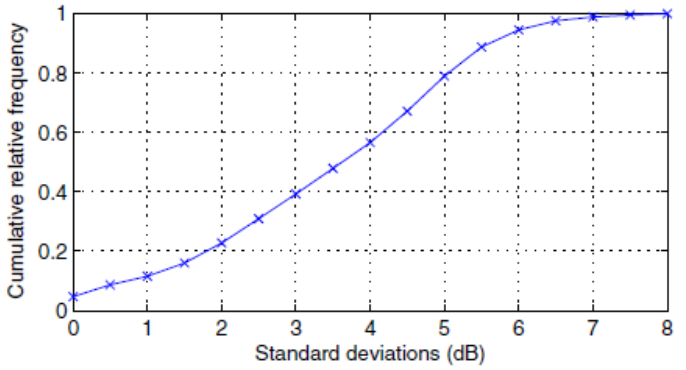


Figure 3. Cumulative distribution of E_b/N_0 values



3. Signal Design Proposal:

• Design Goals:

Given the signal parameters and channel effects discussed in the previous Section, a signal design is to be created. The following guidelines are to be considered in the design process:

1) The maximum considered distance is 100 m. This was already included in the channel analysis, see Section II.

2) Within range, the frame error rate shall not rise above 1%. We assume a frame error as soon as one single bit of the received frame after decoding is wrong.

3) The design must be tolerant towards a wide range of signal-to-noise ratios and fading patterns. Additionally, it must be able to operate without any prior channel sounding.

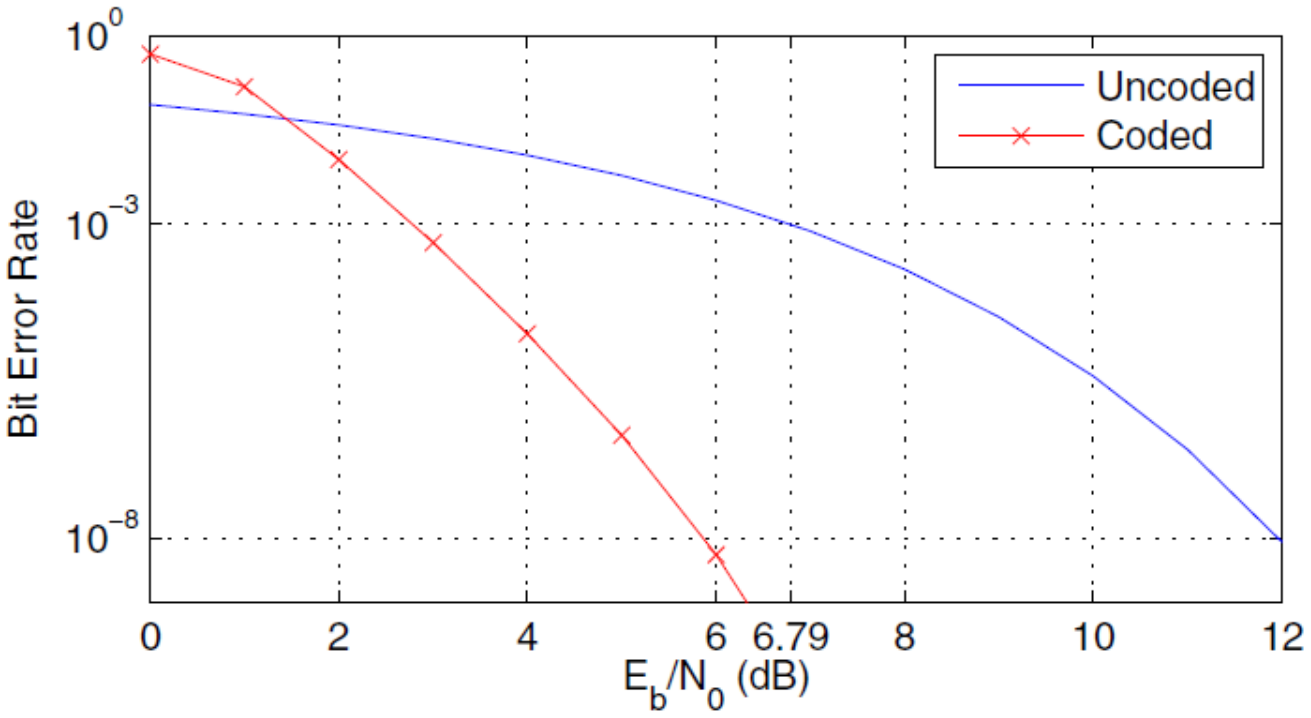


Figure 5. Performance of the convolutional code

In order to achieve the 1% frame error rate it is useful to find the 1-percentile in the cumulative distribution of the SNR values shown in Figure 3. A closer inspection of the empirical data reveals this to be at -5 dB, i.e. 99% of the received symbols have an SNR value of more than -5 dB. In the next sections, two methods to achieve these goals are presented.

- **Frequency Domain Spreading:**

The first presented approach is to combine frequency domain spreading with a simple convolutional channel code. The information bits are first encoded using the industry standard convolutional code of rate R = 1/2 and the generator polynomials

$$g_1(x) = 1 + x^2 + x^3 + x^5 + x^6,$$
$$g_2(x) = 1 + x + x^2 + x^3 + x^6.$$

Figure 5 shows the performance of the given code when using soft-decision Viterbi decoding. Next, the code bits are modulated onto 16 sub-carriers each, spreading the symbols out in frequency domain. At the receiver, the de-spreading will ideally increase the available SNR by 10 log10(16) = 12.04 dB, while decreasing the maximum possible data rate. These values were chosen according to the following conservative estimates: In order to achieve the desired frame error rate, we target a BER of 10−8 in 99% of the time. The decoder requires an SNR of 6 dB to achieve this BER (see Figure 5). Before de-spreading, SNR must thus be 6 dB−12.04 dB = −6.04 dB,
which is below the 1-percentile. The factor 16 is the smallest integer factor of N which achieves this.

This approach features two main advantages: first, it is very simple and easy to implement. De-spreading can be performed by a simple equal gain combining method, and the featured convolutional code is widely used, e.g. in the IEEE 802.11a standard. Besides, multi user access is also easy to integrate:
instead of spreading onto 16 carriers, a shorter spreading sequence of 32/U can be used to spread the energy onto every U-th carrier. The total bit energy at the receiver after the despreading is not affected by this, and the convolutional coder can be completely ignorant of U.
The achievable data rate of this system is:
r ≈ 2.6 MBit/s
although in a real system, synchronisation and equalisation will require pilot symbols which would further lower the real available throughput.

• Channel Coding:

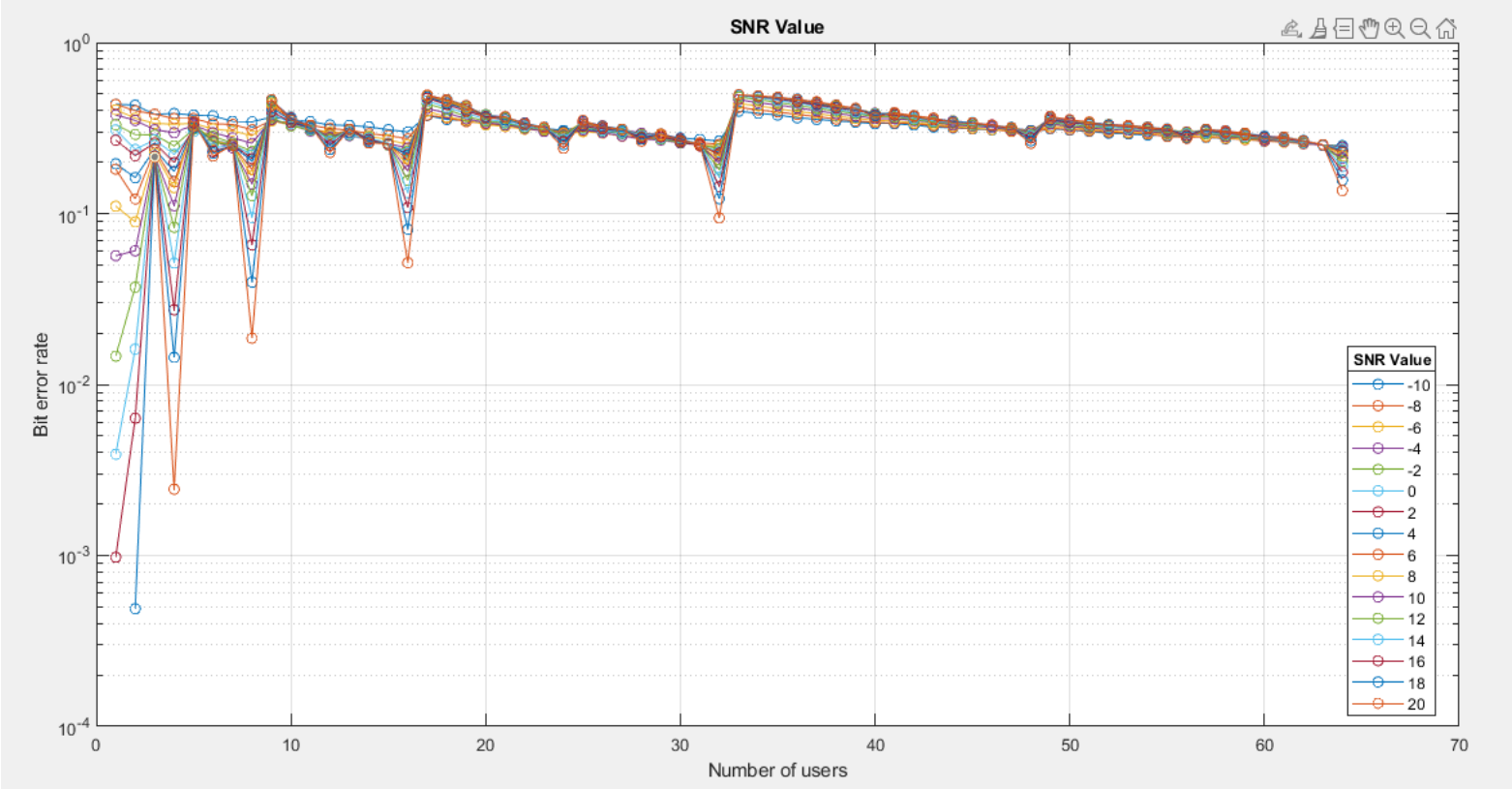
We can use convolutional codes as our forward error correction codes mentioned in the following MATLAB code as an example of using the Convolutional FEC codes in 3G-CDMA2000 with a 1x spreading rate:

The MATLAB code:

```
% Define the number of users to simulate and the signal length
num_users = 64;
signal_length = 1024;

% Generate random data for each user
data = randi([0 1], num_users, signal_length);

% Find the smallest power of 2 greater than or equal to the number of users
pow2_num_users = 2^nextpow2(num_users);

% Generate Walsh codes for the smaller power of 2
walsh_codes_small = hadamard(pow2_num_users);

% Use Kronecker product to generate Walsh codes for the actual number of users
walsh_codes = kron(eye(num_users), walsh_codes_small(1:num_users, 1:num_users));
```

```
% Define the range of number of users to simulate
num_users_range = 1:num_users;

% Define the range of signal-to-noise ratios (SNRs) to simulate
snr_range = -10:2:20;

% Initialize the bit error rate (BER) matrix
ber_matrix = zeros(length(num_users_range), length(snr_range));

% Loop over the number of users and SNRs to calculate the BER for each combination
for i = 1:length(num_users_range)
    for j = 1:length(snr_range)
        % Calculate the index of the current SNR in the snr_range
        snr_index = j;

        % Get the number of users to simulate for the current iteration
        num_users_current = num_users_range(i);

        % Generate the subset of Walsh codes and data for the current number of users
        walsh_codes_current = walsh_codes(1:num_users_current, 1:num_users_current);
        data_current = data(1:num_users_current, :);

        % Spread the data for each user using their assigned Walsh code
        spread_data = walsh_codes_current * data_current;

        % Add Gaussian noise to the spread data to simulate the channel
        noise_power = 10^(-snr_range(snr_index) / 10);
        noise = sqrt(noise_power) * randn(size(spread_data));
        noisy_data = spread_data + noise;

        % Despread the received data using the assigned Walsh codes
        despread_data = walsh_codes_current' * noisy_data;

        % Decode the data for each user and calculate the BER
        decoded_data = (despread_data > 0.5);
        errors = sum(xor(decoded_data, data_current), 2);
        ber = errors / signal_length;

        % Store the BER for the current number of users and SNR
        ber_matrix(i, j) = mean(ber);
    end
end

% Plot the BER vs. the number of users for each SNR
figure
semilogy(num_users_range, ber_matrix, '-o')
xlabel('Number of users')
ylabel('Bit error rate')
title('Network quality vs. number of users')
leg = legend(string(snr_range), 'Location', 'best');
title(leg, 'SNR Value')
grid on

legend_title = title('SNR Value');
set(legend_title, 'String', 'SNR Value');
```

The Result MATLAB Graphs:



Reed-Muller Coding is an alternative approach is to solely rely on one type of channel coding with a very low code rate. The solution proposed here is the use of Reed-Muller-Codes (RM-Codes).

The principal difficulty with OFDM is **the high peak-to-mean envelope power ratio (PMEPR)** of the output signal, which can be as high as the number of carriers in the systems. Popovic shows that the PMEPR of any Golay-sequence is bounded by a value of two (or 3 dB). The connection between Golay-sequences and the second order Reed-Mullercode, which provides efficient encoding and decoding methods, was recognized. Very recently, the application of these codes in an OFDM radar context. The r-th order binary RM-code RM2(r, m) comprises 2k codewords of length 2m, where k is given by

$$k = \sum_{i=0}^{r} \binom{m}{i}.$$

Of particular importance is the code RM2(2,m) comprising $2^{\frac{m(m-1)}{2}}$ cosets of the code RM2(1,m), each containing 2m+1 codewords. By partitioning the codewords of RM2(2,m) into cosets of RM2(1,m), the codewords with large values of PMEPR can be isolated. If the transmitted codewords are

restricted to those belonging to the Golay-cosets, the PMEPR is bounded by 3dB. For implementation convenience only one of the m!/2 Golay-cosets is used, which results in a code rate of $R_c = \frac{m+1}{2^m}$. As the length of the codewords increases, the code rate becomes very low. If only one Golay-coset is used, the minimum Hamming distance of the resulting code is given by $d_{\min} = 2^{m-1}$, whereas the minimum distance is $d_{\min} = 2^{m-2}$ if two or more Golay-cosets are used. Another advantage of using only one Golay-coset is the reduction of the complexity of the decoder. After subtracting the Golay-coset representative of received codeword, the result can be decoded with a basic RM2(1,m) decoder. If a large number of cosets is used, this operation has to be done for every coset representative, which increases the complexity. The binary Reed-Muller-code of first order RM2(1,m) can be maximum-likelihood decoded using the fast Hadamard transform (FHT), which can be implemented in hardware. Furthermore, the decoder can be realised as hard- or soft-decision decoder without extra complexity. Since we only want to use every U-th sub-carrier, we can use the RM-code with m = 7, resulting in one code word of length n = 128 per OFDM symbol, thereby guaranteeing the PMEPR of 3 dB.

	Frequency Domain Spreading	RM-Coding
Urban	0	0
Highway	$1.32 \cdot 10^{-3}$	$2.30 \cdot 10^{-3}$
Total	$1.24 \cdot 10^{-4}$	$2.13 \cdot 10^{-4}$

Table II
FRAME ERROR RATES

The achievable data rate of the system using the parameters of the below Table is:

r ≈ 0.64 MBit/s

although, as we mention before, in reality it will be smaller due to pilot symbols. It must be noted that this data rate is only 25% that of the previous system.

Parameter	Value
Number of carriers N	1024
Number of symbols M	256
OFDM symbol duration T	11 µs
Guard interval fraction G	$1/8$
Total OFDM Symbol duration T_O	$(1 + G)T = 12.375$ µs
Total transmit power P	100 mW
Noise Figure $\mathrm{NF_{dB}}$	10 dB
Center frequency f_c	24 GHz
Bits per Modulation Symbol N_b	1 (BPSK)
Number of simultaneous accesses U	8

Table
OFDM SYSTEM PARAMETERS

- **Results & Conclusion Of OFDM Signal Design in Car-To-Car Communications:**

We performed simulations to test the effectiveness of the methods in real channels. It must be noted that these simulations are, by themselves, not adequate to fully assess the quality of the coding methods, since it is unclear how much they degrade in adverse synchronisation or equalisation.

However, these simulations provide some useful measures if the theoretical analysis of the chosen codes is sensible at all.

The simulations consisted of the following steps:

- For every channel, 16 OFDM frames constructed according to the parameters in Table I were created. The data loaded into the frame was created according to the coding methods described in the previous Section.
- Likewise, the matrix S was calculated.
- For every element of the OFDM frames, white Gaussian noise was added such that the total SNR was consistent with the values in S.
- Every frame was demodulated and decoded according the given method.

Table II shows the results and confirms the applicability of the codes.

	Frequency Domain Spreading	RM-Coding
Urban	0	0
Highway	$1.32 \cdot 10^{-3}$	$2.30 \cdot 10^{-3}$
Total	$1.24 \cdot 10^{-4}$	$2.13 \cdot 10^{-4}$

Table II
FRAME ERROR RATES

The desired frame error rate was achieved by a wide margin. The advantages of each presented method are clear:
The frequency domain spreading approach can achieve higher data rates, while the RM-coding approach has lower requirements towards the RF hardware.

we present two possibilities to package OFDM frames for use in combined OFDM radar and communication in the 24 GHz ISM band. The methods differ in several respects. The combined spreading and convolutional coding yields a higher data rate and the employed coding scheme is very common. However, the channel code makes the RF front-end easier to implement since due to the constant low PMEPR the transmitter linearity requirements are lower. Its decoding scheme is based on the computationally efficient FHT. Both codes have suitable error correction capabilities. Of course these two methods do not exhaust the possibilities, and many other codes could be applied. Our goal was to present two different available paths. Future development of the combined OFDM radar and communications system must clearly define priorities concerning data rate, RF frontend complexity and robustness, thereby making a choice for one of the methods. In a next step, synchronisation and equalisation must be taken into account and full simulations will be performed. For a final and conclusive evaluation of the signal designs, it will be necessary to perform live measurements as well. we have presented a measurement setup which could be used to run live measurements.

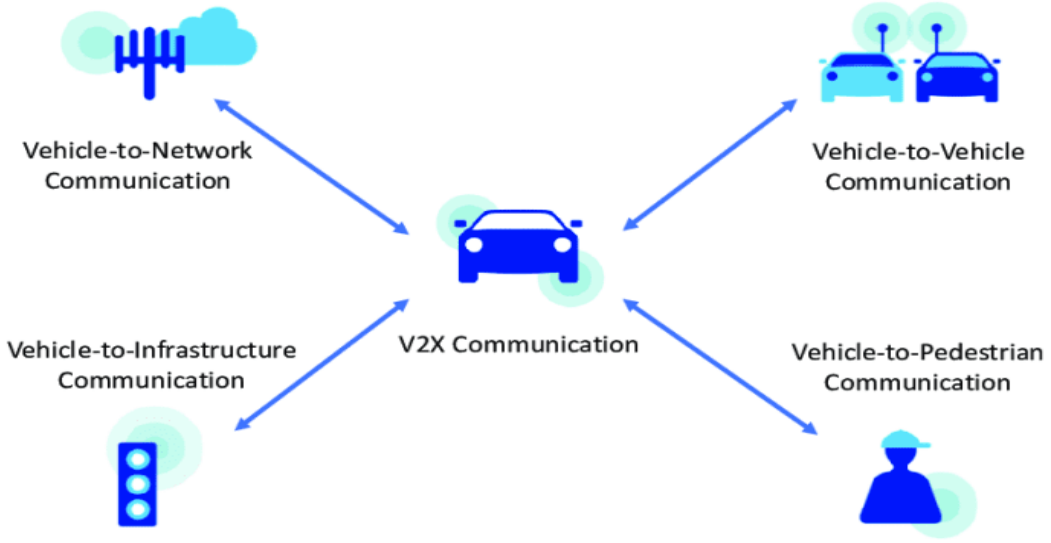
4. What is V2X and How Does It Work?

Definition & Introduction:

V2X, or Vehicle-to-Everything, refers to the communication technology that enables vehicles to communicate with other vehicles (V2V), infrastructure (V2I), pedestrians (V2P), and networks (V2N). This technology is a critical component of smart transportation systems and is aimed at improving road safety, reducing traffic congestion, and enhancing overall transportation efficiency. V2X technology allows vehicles to exchange information such as location, speed, and status, enabling them to make real-time decisions to avoid accidents, optimize traffic flow, and enhance overall driving experience.

How V2X Works?

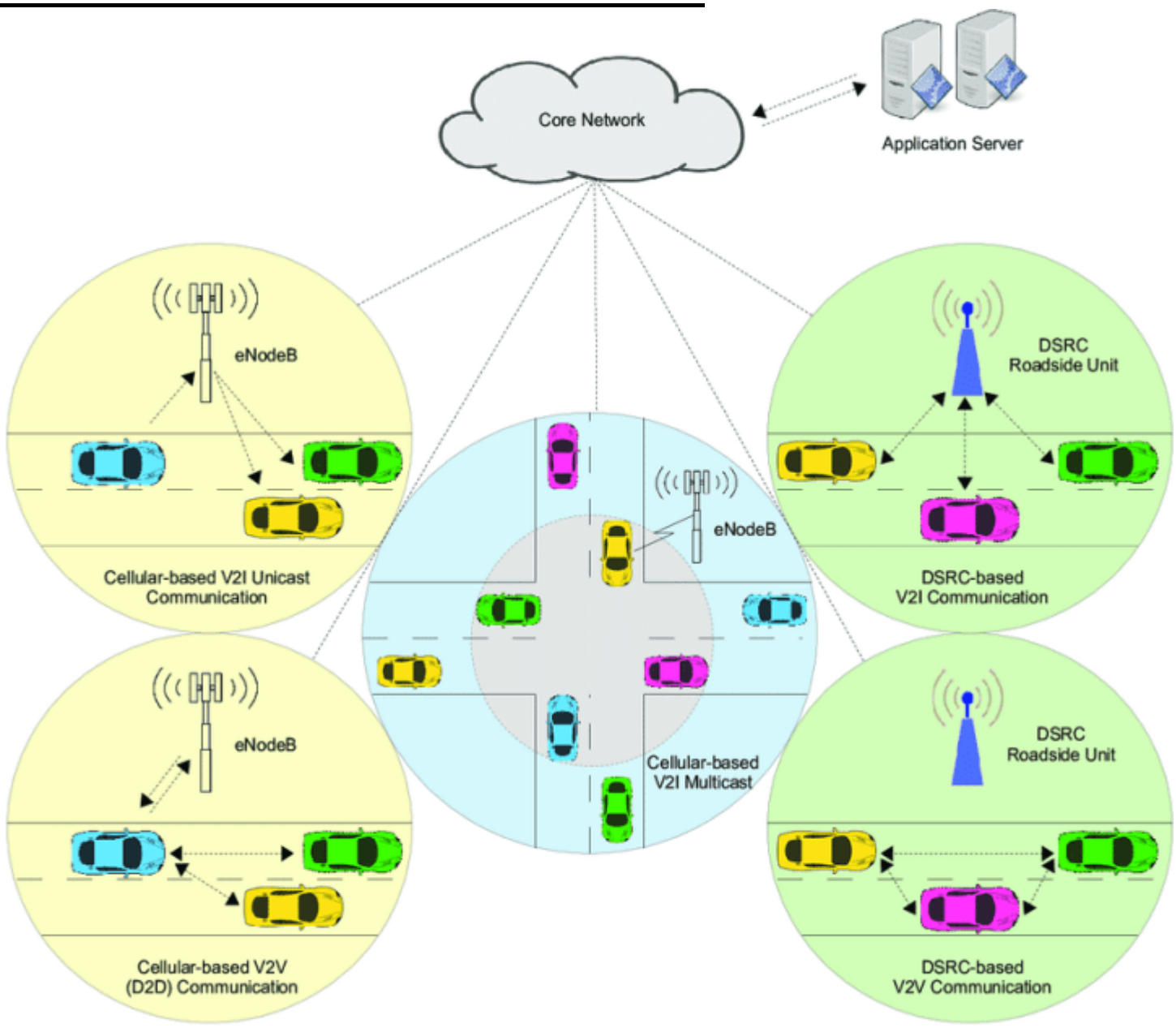
In a V2X system, the data is exchanged through vehicle sensors using high-bandwidth and high-reliability links. Every car has a sensor that sends data to each other as well as infrastructure, such as traffic lights, parking spaces, pedestrians, and more.



- These V2X systems share information related to speed and other entities around the vehicle. This information enhances the driver's awareness of things like road conditions, nearby accidents, approaching of emergency vehicles, road works notices, and activities of other drivers on the same route. It also intimates the drivers about any potential dangers and helps to diminish the likelihood of road accidents and injuries.

- Much like other communication technologies, V2X, in recent years, has split into 2 varied camps based on varied technologies.
    - DSRC, Dedicated Short Range Communication
    - C-V2X, Cellular V2X

DSRC Communication & How It Works:



DSRC is the original V2X standard that enables vehicles to communicate with each other and with the infrastructure via over-the-air messages compliant with the IEEE 802.11P standard. This links two V2X senders that are in close proximity to each other and enables them to communicate without necessitating any specific communication device. Such a capability makes DSRC ideal for less-developed areas.

Moving on, the DSRC standard provides information about toll payments and accident notices. It operates in short-range, i.e. under 1 kilometer, and runs in the unlicensed 5.9GHz band. One of the best things about this standard is that it is totally unaffected by weather conditions. Thus, even in times of rain, fog, or snow, it can easily scan the environment and deliver accurate information.

One of the best benefits of using C-V2X is that it comes with operational modes that the users can choose from. The first mode is a low-latency Cellular-V2X Direct Communications over the PC5 interface, on the unlicensed 5.9GHz band. This mode is specially designed for exchanging active safety messages such as collision warnings and other short-range V2I, V2V, and V2P situations. This mode closely aligns with what is offered by the traditional DSRC standard.

The second mode takes place over the Uu interface on the regular band cellular network that is licensed. This mode is primarily used to manage V2N use cases, such as latency-tolerant safety notifications and infotainment, or traffic conditions. Because this mode does not leverage cellular connectivity, DSRC can only match it by forming ad hoc connections to roadside base stations.

As the V2X market gains momentum, the two contending technologies, i.e. DSRC and C-V2X compete for the attention of leading automakers and regulators. It is worth mentioning that lately, the impetus has swung in favor of C-V2X, with an increasing number of automakers, such as BMW, Ford, Audi, Jaguar, Geely, and Daimler supporting it.

Vehicular mobility is an important challenge that needs to be addressed while designing Vehicular Adhoc NETworks with V2V and V2I connectivity. The first part of this project aims to study the impact of vehicular mobility and traffic volume on V2V connectivity. It successfully implements the freeway mobility model, the car following model, the lane changing model and border effects in Matlab for a 4-lane scenario with 3 entry and 3 exit ramps. The simulations run for 5 trials of 10 minutes duration each, for every traffic density. We have observed linear variation in our graphs with traffic densities and the values seem to be feasible and practical. With the change in communication range, the output also changed.

5. MATLAB Codes & Simulations:

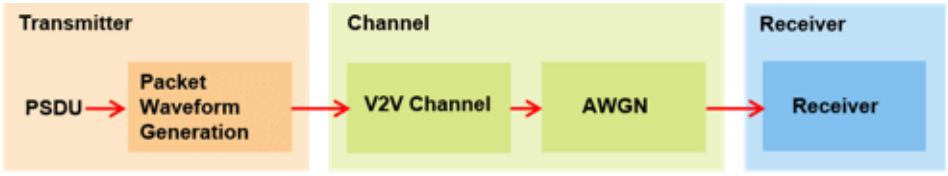
• Packet Error Rate Simulation For A Vehicular Channel Using The Protocol Of: IEEE® 802.11p

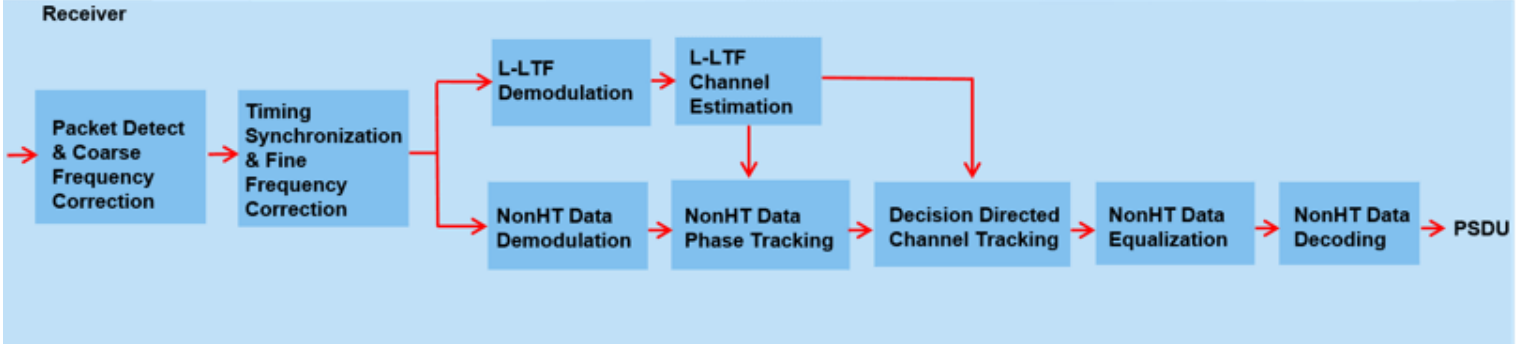
This example shows how to measure the packet error rate (PER) of an IEEE® 802.11p™ link using an end-to-end simulation with a Vehicle-to-Vehicle (V2V) fading channel and additive white Gaussian noise. The PER performance of a receiver with and without channel tracking is compared. In a vehicular environment (high Doppler), a receiver with channel tracking performs better.

Introduction

IEEE 802.11p is an approved amendment to the IEEE 802.11™ standard to enable support for wireless access in vehicular environments (WAVE). Using the half-clocked mode with a 10 MHz channel bandwidth, it operates in 5.85-5.925 GHz bands to support applications for Intelligent Transportation Systems (ITS).

In this example, an end-to-end simulation is used to determine the packet error rate for an 802.11p link with a fading channel at a selection of SNR points with and without channel tracking. For each SNR point, multiple packets are transmitted through a V2V channel, demodulated and the PSDUs are recovered. The PSDUs are compared to those transmitted to determine the number of packet errors. For each packet, packet detection, timing synchronization, carrier frequency offset correction and phase tracking are performed at the receiver. For channel tracking, decision directed channel estimation is used to compensate for the high Doppler spread. The figure below shows the processing chain with channel tracking.





Waveform Configuration

An 802.11p non-HT format transmission is simulated in this example. A non-HT format configuration object contains the format specific configuration of the transmission. This object is created using the `wlanNonHTConfig` function. In this example, the object is configured for a 10 MHz channel bandwidth and QPSK rate 1/2 (MCS 2) operation.

```
% Link parameters
mcs = 2;       % QPSK rate 1/2
psduLen = 500; % PSDU length in bytes

% Create a format configuration object for an 802.11p transmission
cfgNHT = wlanNonHTConfig;
cfgNHT.ChannelBandwidth = 'CBW10';
cfgNHT.PSDULength = psduLen;
cfgNHT.MCS = mcs;
```

Channel Configuration

The V2V radio channel model defines five scenarios to represent fading conditions within a vehicular environment. In this example, 'Urban NLOS' scenario is used. This corresponds to a scenario with two vehicles crossing each other at an urban blind intersection with building and fences present on the corners.

```
% Create and configure the channel
fs = wlanSampleRate(cfgNHT); % Baseband sampling rate for 10 MHz

chan = V2VChannel;
chan.SampleRate = fs;
chan.DelayProfile = 'Urban NLOS';
```

Simulation Parameters

For each SNR (dB) point in the vector snr a number of packets are generated, passed through a channel and demodulated to determine the packet error rate.

```
snr = 15:5:30;
```

The number of packets tested at each SNR point is controlled by two parameters:

1. maxNumErrors is the maximum number of packet errors simulated at each SNR point. When the number of packet errors reaches this limit, the simulation at this SNR point is complete.

2. maxNumPackets is the maximum number of packets simulated at each SNR point. It limits the length of the simulation if the packet error limit is not reached.

The numbers chosen in this example lead to a short simulation. For statistical meaningful results these numbers should be increased.

```
maxNumErrors = 20;   % The maximum number of packet errors at an SNR point
maxNumPackets = 200; % The maximum number of packets at an SNR point

% Set random stream for repeatability of results
s = rng(98);
```

Processing SNR Points

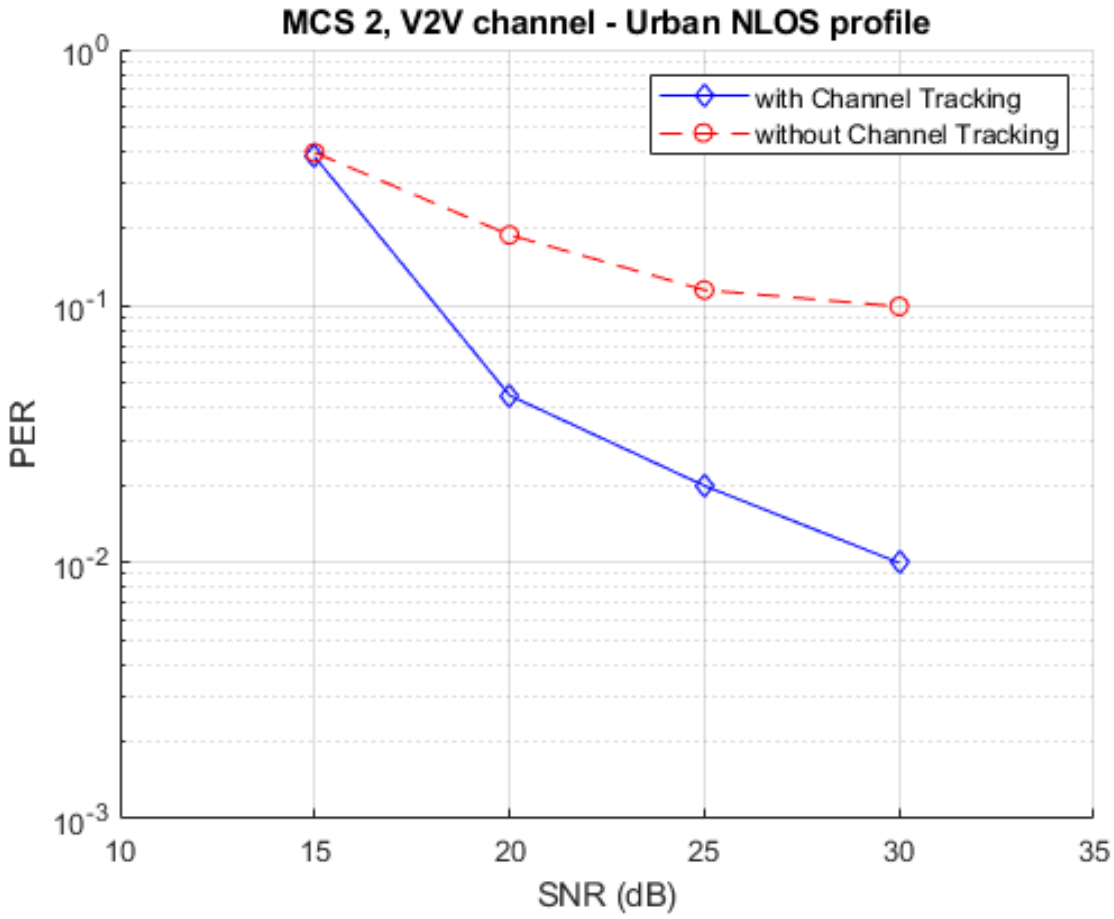
For each SNR point, a number of packets are tested and the packet error rate is calculated. For each packet the following processing steps occur:

1. A PSDU is created and encoded to create a single packet waveform.
2. The waveform is passed through the channel. Different channel realizations are used for each transmitted packet.
3. AWGN is added to the received waveform to create the desired average SNR per active subcarrier after OFDM demodulation.
4. The per-packet processing includes packet detection, coarse carrier frequency offset estimation and correction, symbol timing and fine carrier frequency offset estimation and correction.
5. The L-LTF is extracted from the synchronized received waveform. The L-LTF is OFDM demodulated and initial channel estimates are obtained.
6. Channel tracking can be enabled using the switch enableChanTracking. If enabled, the channel estimates obtained from L-LTF are updated per symbol using decision directed channel tracking as presented in J. A. Fernandez et al in. If disabled, the initial channel estimates from L-LTF are used for the entire packet duration.
7. The non-HT Data field is extracted from the synchronized received waveform. The PSDU is recovered using the extracted data field and the channel estimates and noise power estimate.

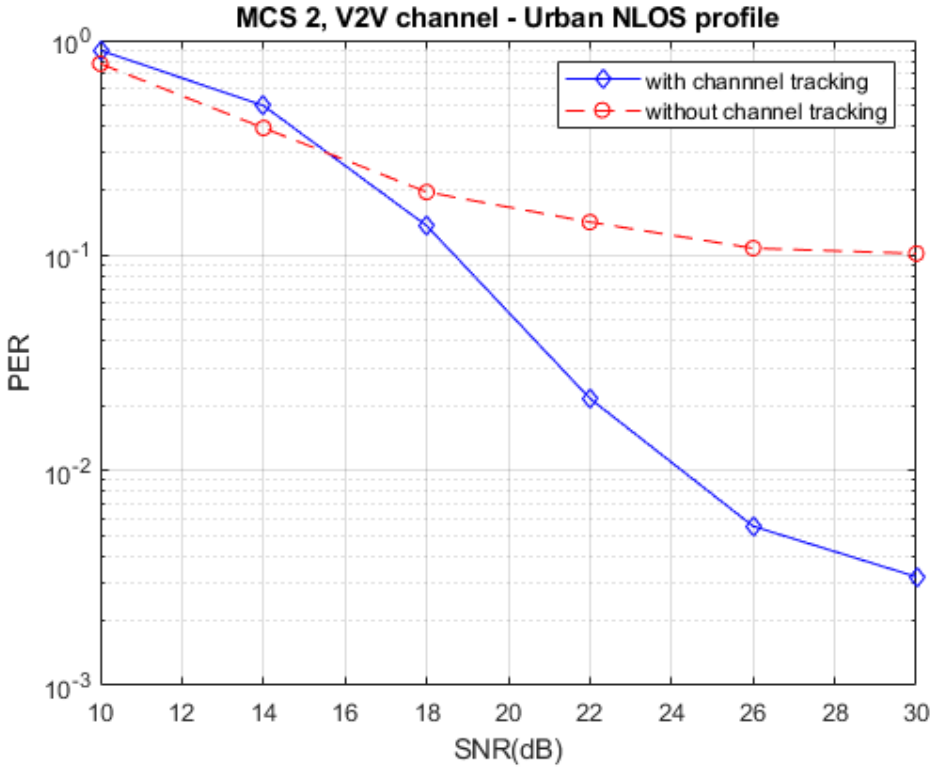
```
% Set up a figure for visualizing PER results
h = figure;
grid on;
hold on;
ax = gca;
ax.YScale = 'log';
```

```
xlim([snr(1), snr(end)]);
ylim([1e-3 1]);
xlabel('SNR (dB)');
ylabel('PER');
h.NumberTitle = 'off';
h.Name = '802.11p ';
title(['MCS ' num2str(mcs) ', V2V channel - ' chan.DelayProfile ' profile']);

% Simulation loop for 802.11p link
S = numel(snr);
per_LS = zeros(S,1);
per_STA = per_LS;
for i = 1:S
    enableChanTracking = true;
    % 802.11p link with channel tracking
    per_STA(i) = v2vPERSimulator(cfgNHT, chan, snr(i), ...
        maxNumErrors, maxNumPackets, enableChanTracking);

    enableChanTracking = false;
    % 802.11p link without channel tracking
    per_LS(i) = v2vPERSimulator(cfgNHT, chan, snr(i), ...
        maxNumErrors, maxNumPackets, enableChanTracking);

    semilogy(snr, per_STA, 'bd-');
    semilogy(snr, per_LS, 'ro--');
    legend('with Channel Tracking','without Channel Tracking')
    drawnow;
end

axis([10 35 1e-3 1])
hold off;

% Restore default stream
rng(s);
```

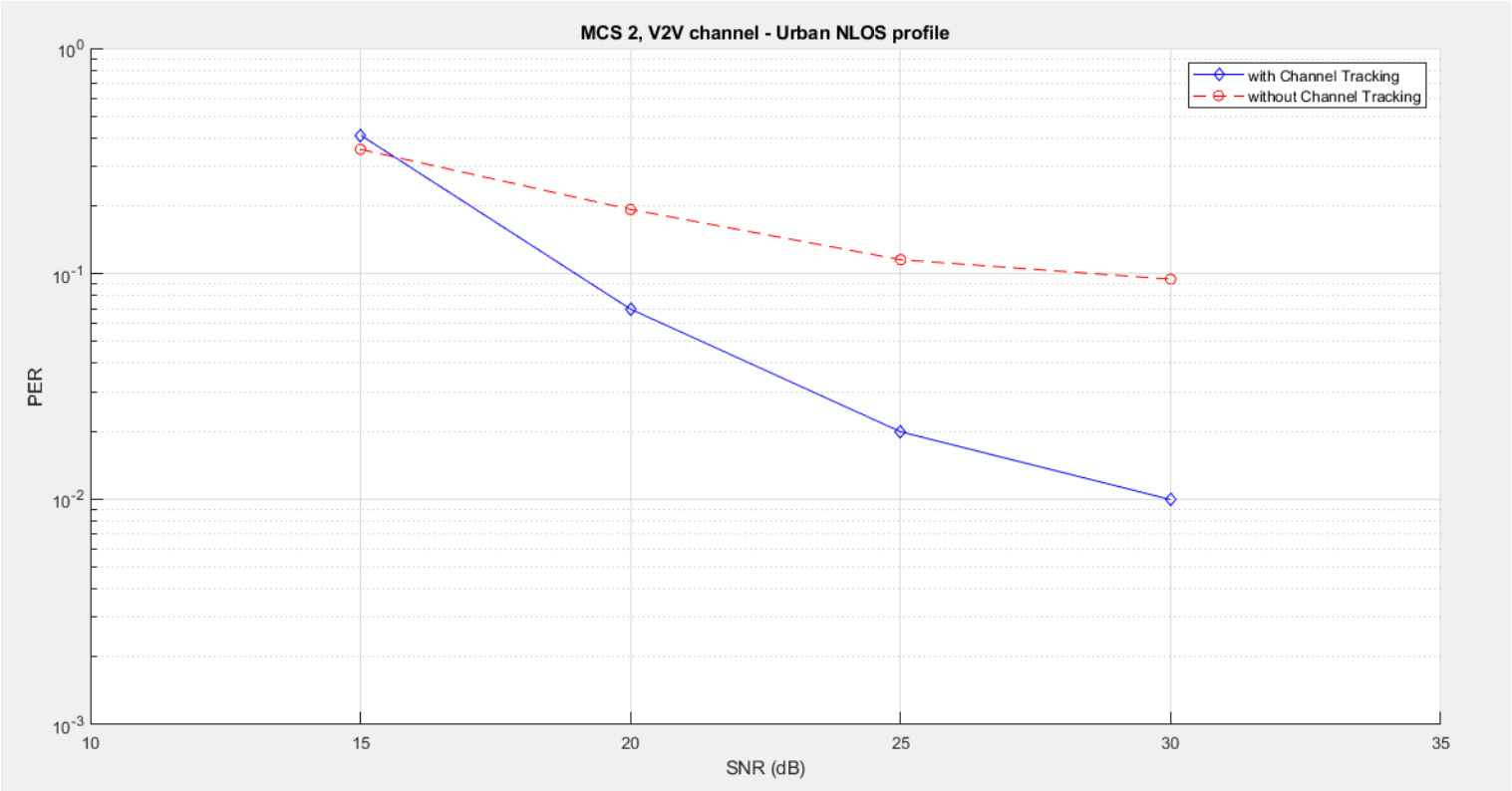
```
>> main_PacketErrorRateSimulationForVehicularChannel
SNR 15 dB with channel tracking completed after 51 packets, PER: 0.41176
SNR 15 dB without channel tracking completed after 59 packets, PER: 0.35593
SNR 20 dB with channel tracking completed after 201 packets, PER: 0.069652
SNR 20 dB without channel tracking completed after 109 packets, PER: 0.19266
SNR 25 dB with channel tracking completed after 201 packets, PER: 0.0199
SNR 25 dB without channel tracking completed after 182 packets, PER: 0.11538
SNR 30 dB with channel tracking completed after 201 packets, PER: 0.0099502
```



For meaningful results maxNumErrors, maxNumPackets should be increased. The below plot provides results for maxNumErrors: 1000 and maxNumPackets: 10000.



Before increasing the maxNumErrors, maxNumPackets, the below plot provides results for our V2Vreciever MATLAB program:



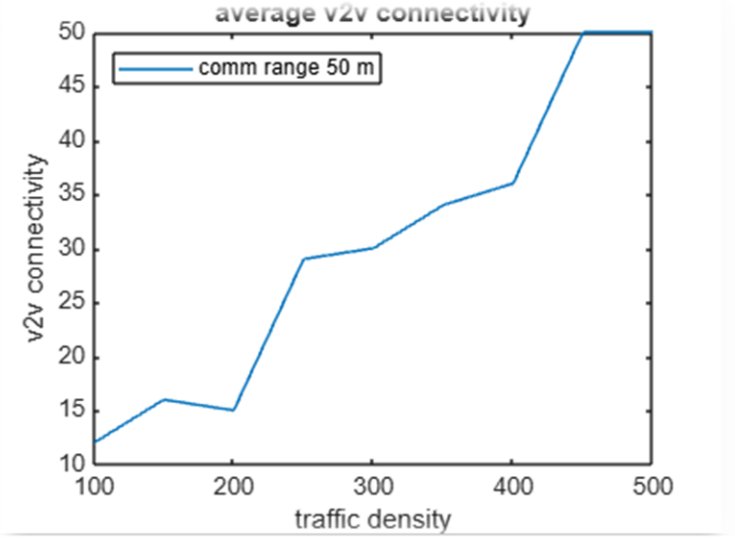
• Modelling Vehicle-To-Vehicle Communications Using Data Plots for VANET Mobility Simulations & V2V Connectivity MATLAB Codes.

VANET Mobility Simulations

VANET stands for Vehicular Ad-Hoc Network. It's a type of network specifically designed for communication among vehicles (V2V), as well as between vehicles and roadside infrastructure (V2I). VANETs use wireless communication technologies, such as DSRC (Dedicated Short-Range Communication) or cellular networks, to enable vehicles to exchange information with each other and with the surrounding infrastructure.

All the plots below are for 5 simulations run for 5 minutes. The communication range is 50 meters.

1. Plot shows the average number of communication nodes within 50 meters from the target.



- The number of neighbors increases as the traffic density increases since the vehicles move closer to each other due to congestion. As a result, the average V2V connectivity increases linearly with traffic density.

- Considering 100 -500 vehicles on each lane of 5km and 50 meters communication range, when uniformly distributed, each node have connectivity with 8-40 vehicles.

2.Plot shows the average duration the target node maintains the same 3 communication neighbors that are within 50 meters.

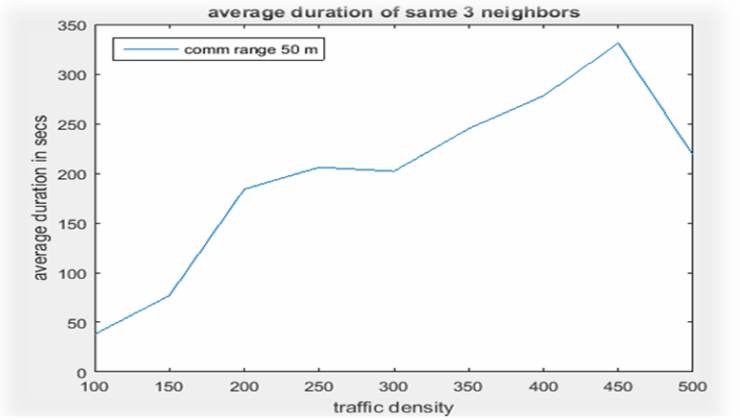


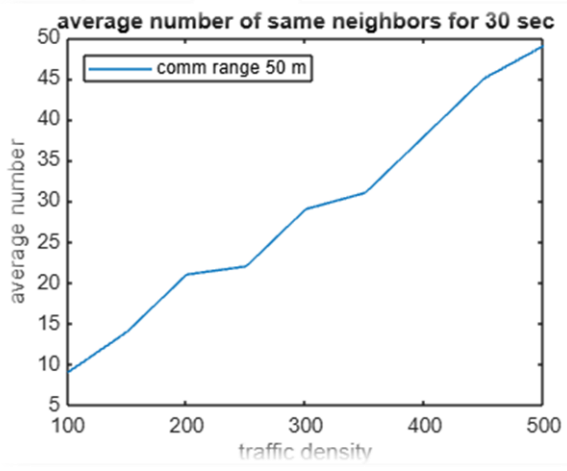
Figure 2

- As expected the average duration of same 3 neighbors also increases linearly with traffic density because the same neighbors tend to stay together longer with increasing traffic. Increasing traffic would lead to congestion and control the way the vehicles move. The vehicles change lanes less often because the possibility of vehicles being beyond safety distance is less likely due to congestion in other lanes as well.

- Duration is varying from 1min(60 seconds) to 6 minutes as traffic density increases. This means we can rely on this model for transmitting non-safety applications also

- Abnormalities in the graphs are not repeatedly occurring for same densities but are randomly occurring. The reason could be the random allocation of speeds and distances which keep changing every 100msecs making the simulation unpredictable even on taking average values.

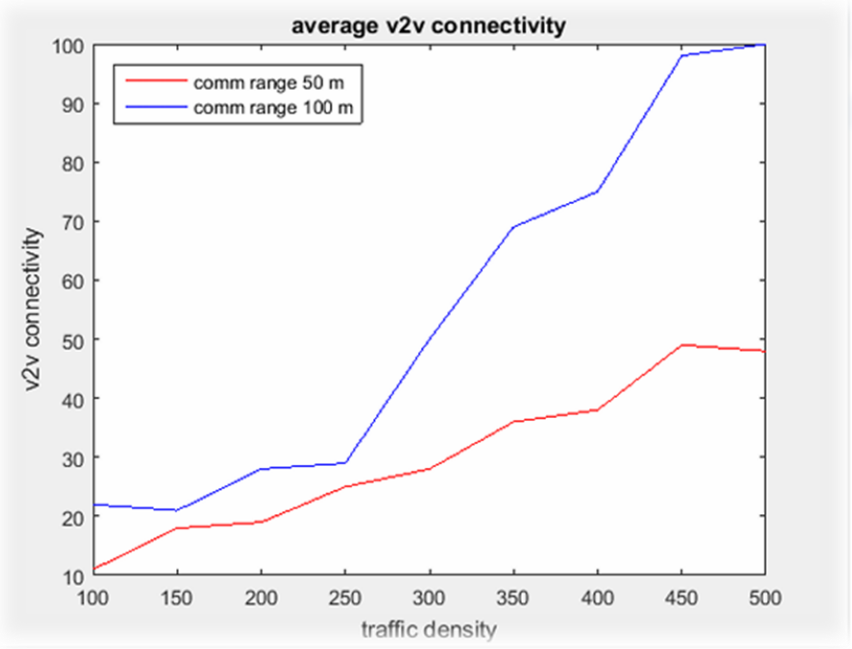
3. Plot shows the average number of same neighbors for 30 seconds



- Again, as expected the average duration of 3 neighbors for a fixed duration also increases linearly with traffic density. Like the explanations for previous observations, Increasing traffic would lead to congestion and reduce the likelihood of lane changing.

- For 30 secs, average number of same neighbours vary increase from 4 to 40.

4. **Plot shows the average number of communication nodes within 50 meters (red) and 100 meters (in blue) from the target.**



5. **Plot shows the average duration the target node maintains the same 3 communication neighbors that are within 50 meters (red) and 100 meters (blue). In 100m and 50 m plots, We can see that the results are almost doubled.**

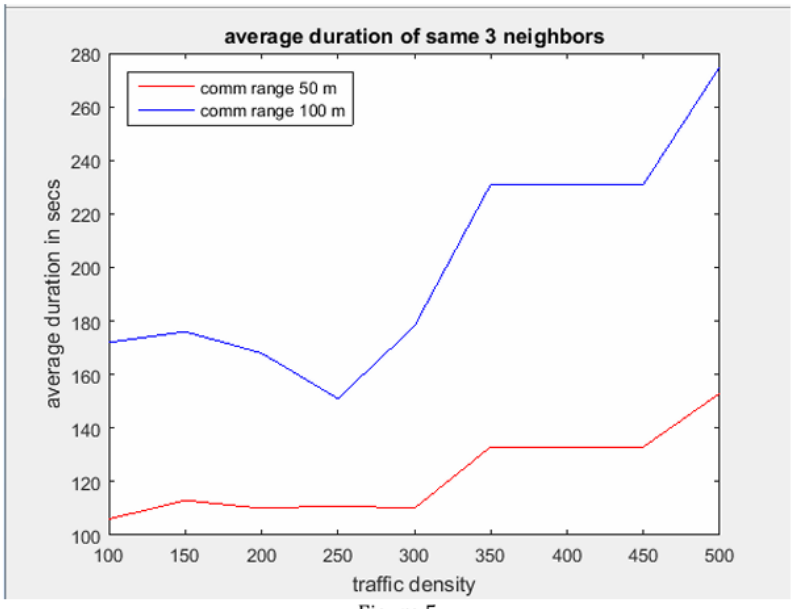
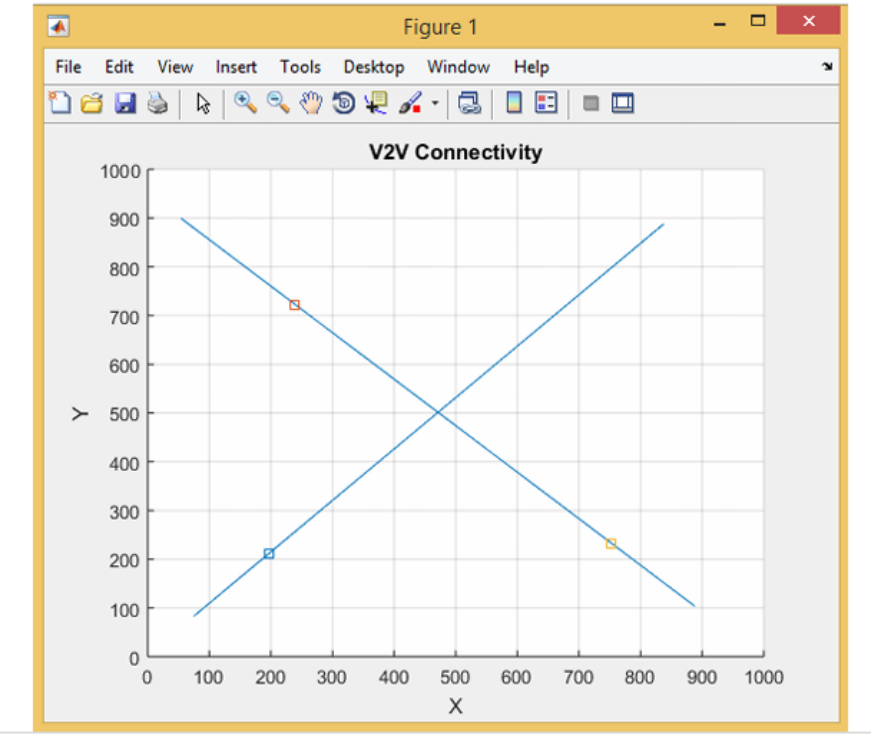


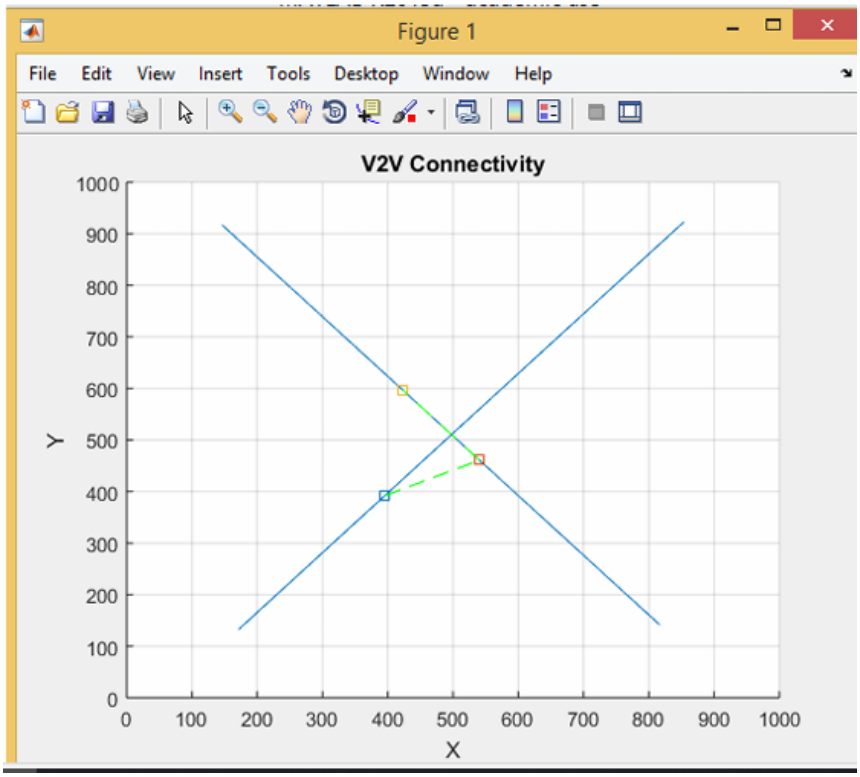
Figure 5

V2V Connectivity

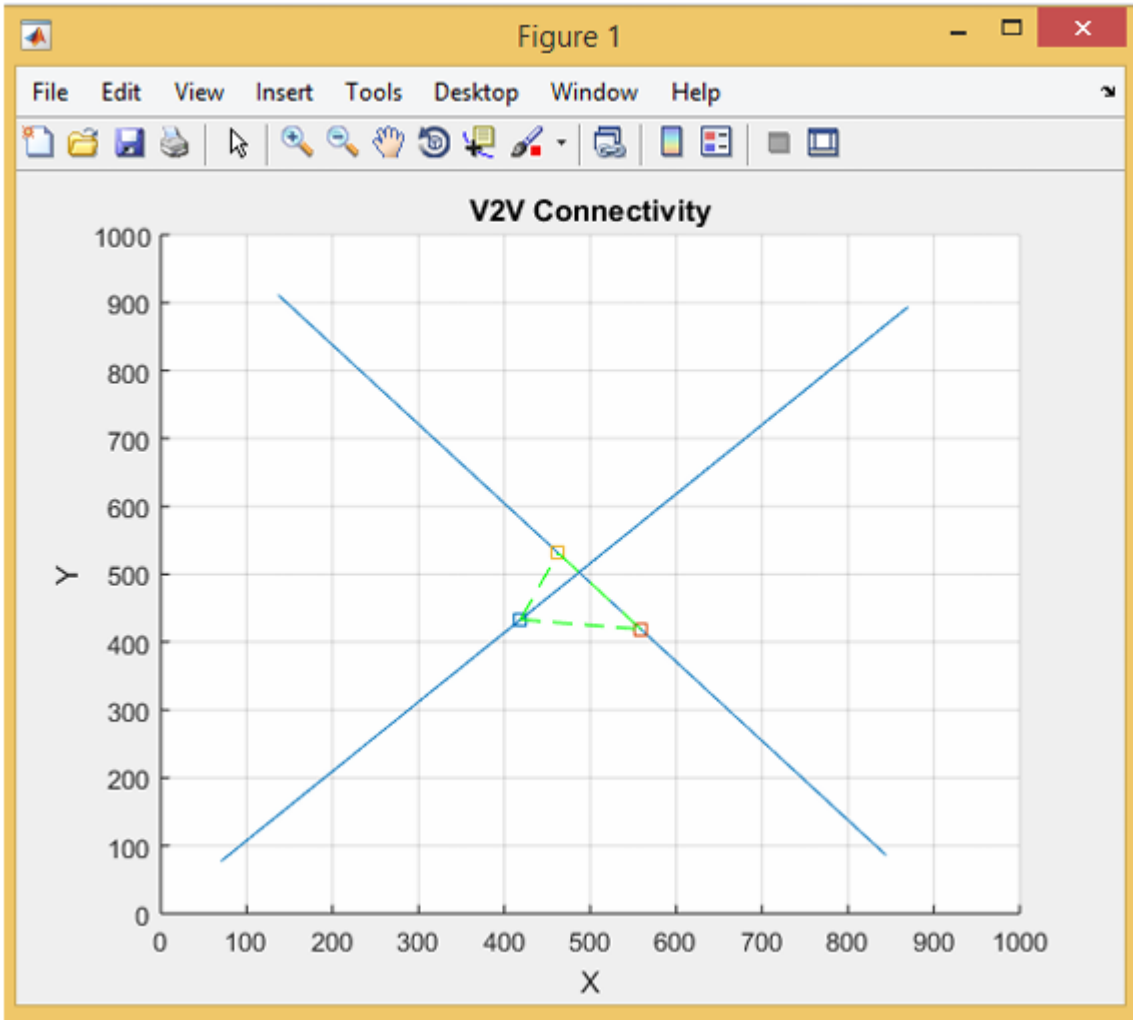
Simulation for 4-Way Road Intersection and V2V Connectivity of Approaching Vehicles 1. The user draws two intersecting roads and three vehicles start moving with random speeds towards the intersection.



A connection is established if the distance between the vehicles is less than 100m. The connection is shown by the green dashed line.



Connection is established between all the three vehicles when they are close to the intersection.



Connection established between all the three vehicles.

Matlab codes:

1. Vanet mobility Code

```
function VANET_mobility
% Parameters for the simulations
% Safety distance (Ds) = 10 meters
% Vehicle speed Smin = 50 miles/hour = 22.35 meters/second, Smax = 70
% miles/hour = 31.29 meters/second
% Vehicle acceletation = a_min= 0 meters/seconds2, a_max = +(-) 5 meters/seconds2
% Road Traffic Volume = vol = 3000 vehicles/hour/lane
% Vehicle Arrival rate/Departure rate = 0.833
% Road Traffic Density = 100-500 vehicles/lane

minutes=2;            %no of minutes
range=50;            %comm range

global Smax;
global Smin;
global amax;
global amin;
global veh_id;
global ytotal;
```

```
global target;

Smin = 22.35;
Smax = 31.29;
amin = 0;
amax = 5;
traffic_density = 100:50:500;
ytotal = 5000;
arrival_rate = 0.833;
departure_rate = 0.833;

x_parta=[];
x_partb=[];
x_partc=[];
sim_conn=0;
tot_same3=0;
tot_same=0;

Ds = 10;
b=0.75;
r=0.0070104;
sdep=(-b+sqrt(b^2+(4*r*Ds)))/(2*r);    %used in car following model

% increments of 50 traffic density
% Assuming same traffic density in all lanes
% Giving initial coordinates to nodes based on uniform distribution


for m=traffic_density

    %fix your target node using veh_id
    target=randi([1 4*m],1,1)

    %empty lanes for each density
    lane1=[];
    lane2=[];
    lane3=[];
    lane4=[];

    for runs=1:1:5

        % Initializing the lanes
        lane1= initialize(m,ytotal,1);
        lane2= initialize(m,ytotal,2);
        lane3= initialize(m,ytotal,3);
        lane4= initialize(m,ytotal,4);

        % Intitializing number of neighbors
        num_neigh=0;

        % Iterations
        % Iterations with 100ms steps for 10 minutes 10*60 / 0.1 = 6000
        % Outer_loop iterations=600; inner_loop=each 100

        for t=0:1:minutes*60

            % Entry ramp once every sec
            if rand<arrival_rate
                random_entry_ramp=randi([1 3],1,1);     %randomly selects one integer out of 1 to 3
                random_entry_lane=randi([1 4],1,1);
                sp=Smin + (Smax - Smin)*rand(1);      %random speed for new node entering
                acc=amin + (amax - amin)*rand(1);     %random speed for new node entering
                veh_id=veh_id+1;                      %if a new node enters,it gets the next id
                switch random_entry_lane
                    case 1
                        %insertnode takes random speed,acceleration,ramp,lane
                        lane1=insertnode(sp,acc,lane1,random_entry_ramp);
                    case 2
                        lane2=insertnode(sp,acc,lane2,random_entry_ramp);
                    case 3
                        lane3=insertnode(sp,acc,lane3,random_entry_ramp);
                    case 4
                        lane4=insertnode(sp,acc,lane4,random_entry_ramp);
                    end
            end
```

```
%exit ramp once every second
if rand<departure_rate
    random_exit_ramp=randi([1 3],1,1);
    random_exit_lane=randi([1 4],1,1);
    %delete node function takes ramp and lane no as arguments
    switch random_exit_lane
        case 1
            lane1=deletenode(lane1,random_exit_ramp);
        case 2
            lane2=deletenode(lane2,random_exit_ramp);
        case 3
            lane3=deletenode(lane3,random_exit_ramp);
        case 4
            lane4=deletenode(lane4,random_exit_ramp);
    end
end

for time=0:0.1:1

    lane=[];   %a temporary array used in lane changing model

    % Border effect and position update
    lane1=border(lane1);
    lane2=border(lane2);
    lane3=border(lane3);
    lane4=border(lane4);

    % Freeway and car following mobility
    lane1=freeway(lane1, sdep, Ds);
    lane2=freeway(lane2, sdep, Ds);
    lane3=freeway(lane3, sdep, Ds);
    lane4=freeway(lane4, sdep, Ds);

    %lane changing model
    %coulnt use a common function as lane2 and lane3 have two adjacent lanes
    %lane1
    i=1;
    while(i<size(lane1,1))
        if((lane1(i+1,1)-lane1(i,1))<10)      %only change lane if leading node is closer than 10 metres
        x = lane2(:,1)<lane1(i,1);            %select nodes on nxt lane less than considered node
        [val,id]=max(lane2(x));               %select the lowest nearest node on next lane
        if(isempty(id)==1)                    %if appending is to be done at start of next lane
            val=lane2(1,1);
                id=1;                          %node at start of lane
            if((lane2(id,1)-lane1(i,1))>10)       %change lane only if difference is Ds
                lane2=vertcat(lane1(i,:),lane2(id:end,:));   %move the vehicle to nxt lane
                lane1(i,:)=[];
                i=i-1;                            %decrement id whenever node leaves the lane
            end
        elseif(id==size(lane2,1) && ((lane1(i,1)-val)>10))   %if appending is to be done at end of next lane
            lane2=vertcat(lane2(1:end,:),lane1(i,:));        %move the vehicle to nxt lane
            lane1(i,:)=[];
            i=i-1;
        else
            if(((lane1(i,1)-val)>10)&&((lane2(id+1,1)-lane1(i,1))>10))       %check if no vehicle within +/- 10
range
                lane2=vertcat(lane2(1:id,:),lane1(i,:),lane2(id+1:end,:));   %move the vehicle to nxt lane
                lane1(i,:)=[];
                i=i-1;
            end
        end
        i=i+1;                            %increment id to run the loop
        else
            i=i+1;                        %increment id if no changes are required
        end
    end


    %lane2
    i=1;
    while(i<size(lane2,1))
        if((lane2(i+1,1)-lane2(i,1))<10)       %only change lane if leading node is closer than 10 metres
        l_ch=round(rand(1));            % coin flip and choose adj lane
        if l_ch==0
            lane=lane1;                 % used a single variable var lane for code optmisation
        else
```

```
        lane=lane3;
    end
    x = lane(:,1)<lane2(i,1);       %select nodes on nxt lane less than considered node
    [val,id]=max(lane(x));          %select the nearest node on next lane
    if(isempty(id)==1)              %if appending is to be done at start of next lane
        val=lane(1,1);
            id=1;                   %node at start of lane
        if((lane(id,1)-lane2(i,1))>=10)                %change lane only if difference is Ds
            lane=vertcat(lane2(i,:),lane(id:end,:));   %move the vehicle to nxt lane
            lane2(i,:)=[];
            i=i-1;                  %decrement id whenever node leaves the lane
        end
    elseif(id==size(lane,1) && ((lane2(i,1)-val)>10))     %nearest node is at the end of other lane
        lane=vertcat(lane(1:end,:),lane2(i,:));           %move the vehicle to nxt lane
        lane2(i,:)=[];
        i=i-1;
    else
        if(((lane2(i,1)-val)>10)&&((lane(id+1,1)-lane2(i,1))>10))    %check if no vehicle within +/- 10 range
            lane=vertcat(lane(1:id,:),lane2(i,:),lane(id+1:end,:));  %move the vehicle to nxt lane
            lane2(i,:)=[];
            i=i-1;
        end
    end
    i=i+1;                                      %increment id to run the loop
    if l_ch==0                                   %as changes are made in a single variable
        lane1=lane;                              %assigning them back
    else
        lane3=lane;
    end
    else
        i=i+1;                                  %increment id if no changes are required
    end
end


%lane3
i=1;
while(i<size(lane3,1))
    if((lane3(i+1,1)-lane3(i,1))<10)        %only change lane if leading node is closer than 10 metres
    l_ch=round(rand(1));            % coin flip and choose adj lane
    if l_ch==0
        lane=lane2;                 % used a single variable var lane for code optmisation
    else
        lane=lane4;
    end
    x = lane(:,1)<lane3(i,1);            %select nodes on nxt lane less than considered node
    [val,id]=max(lane(x));               %select the nearest node on next lane
    if(isempty(id)==1)                    %if appending is to be done at start of next lane
        val=lane(1,1);
            id=1;                         %node at start of lane
        if((lane(id,1)-lane3(i,1))>10)              %nearest node is at the end of other lane
            lane=vertcat(lane3(i,:),lane(id:end,:));   %move the vehicle to nxt lane
            lane3(i,:)=[];
            i=i-1;                  %decrement id whenever node leaves the lane
        end
    elseif(id==size(lane,1) && ((lane3(i,1)-val)>10))       %nearest node is at the end of other lane
        lane=vertcat(lane(1:end,:),lane3(i,:));   %move the vehicle to nxt lane
        lane3(i,:)=[];
        i=i-1;
    else
        if(((lane3(i,1)-val)>10)&&((lane(id+1,1)-lane3(i,1))>10))   %check if no vehicle within +/- 10 range
            lane=vertcat(lane(1:id,:),lane3(i,:),lane(id+1:end,:));   %move the vehicle to nxt lane
            lane3(i,:)=[];
            i=i-1;
        end
    end
    i=i+1;                                      %increment id to run the loop
    if l_ch==0                                  %as changes are made in a single variable
        lane2=lane;                             %assigning them back
    else
        lane4=lane;
    end
    else
        i=i+1;                                %increment id if no changes are required
    end
end
```

```
%lane4
i=1;
while(i<size(lane4,1))
    if((lane4(i+1,1)-lane4(i,1))<10)       %only change lane if leading node is closer than 10 metres
    x = lane3(:,1)<lane4(i,1);            %select nodes on nxt lane less than considered node
    [val,id]=max(lane3(x));               %select the nearest node on next lane
    if(isempty(id)==1)                    %if appending is to be done at start of next lane
        val=lane3(1,1);
            id=1;
        if((lane3(id,1)-lane4(i,1))>10)
            lane3=vertcat(lane4(i,:),lane3(id:end,:));   %move the vehicle to nxt lane
            lane4(i,:)=[];
            i=i-1;                          %decrement id whenever node leaves the lane
        end
    elseif(id==size(lane3,1) && ((lane4(i,1)-val)>10))     %nearest node is at the end of other lane
        lane3=vertcat(lane3(1:end,:),lane4(i,:));   %move the vehicle to nxt lane
        lane4(i,:)=[];
        i=i-1;
    else
        if(((lane4(i,1)-val)>10)&&((lane3(id+1,1)-lane4(i,1))>10))   %check if no vehicle within +/- 10 range
            lane3=vertcat(lane3(1:id,:),lane4(i,:),lane3(id+1:end,:));   %move the vehicle to nxt lane
            lane4(i,:)=[];
            i=i-1;
        end
    end
    i=i+1;                                %increment id to run the loop
    else
        i=i+1;                            %increment id if no changes are required
    end
end

%connectivity of target node
%check for target id in each lane
if(any(lane1(:,4)==target))
    xpos=0;           %used in no of neighbors calculation
    lanex=lane1;      %using same array for code optimisation
elseif(any(lane2(:,4)==target))
   xpos=3;             %used in no of neighbors calculation
   lanex=lane2;
elseif(any(lane3(:,4)==target))
   xpos=6;             %used in no of neighbors calculation
   lanex=lane3;
elseif(any(lane4(:,4)==target))
   xpos=9;             %used in no of neighbors calculation
   lanex=lane4;
else
    disp('target not found')
end
id=find(lanex(:,4)==target);    %location of target node
ypos=lanex(id,1);               %ycoord of target node

%no of neighbors on each lane irrespective of what lane target is on
ids1=lane1(find(sqrt((lane1(:,1)-ypos).^2+(0-xpos).^2)<=range),4);
ids2=lane2(find(sqrt((lane2(:,1)-ypos).^2+(3-xpos).^2)<=range),4);
ids3=lane3(find(sqrt((lane3(:,1)-ypos).^2+(6-xpos).^2)<=range),4);
ids4=lane4(find(sqrt((lane4(:,1)-ypos).^2+(9-xpos).^2)<=range),4);

%total neighbors
neigh=[ids1;ids2;ids3;ids4]';
neigh(find(neigh==target))=[];    %as per the formula used target is also a neighbor... so deleting it
num_neigh=num_neigh+length(neigh);    %total no of neighbors

if(t==0 && time==0)             %start of each simulation
    set3=[];
    f=0;
    g=[];
    k=[];
    same3=[];
    time_cnt=0;
    flag=0;
    set_30=[];
end

%part b
```

```
                set3=union(neigh,set3);       %this array has all neighbors every iteration
                if(length(intersect(set3,neigh))>=3)    %see if it has same 3 neighbors
                    time_cnt=time_cnt+0.1;
                    flag=1;
                elseif(length(intersect(set3,neigh))<3)
                    if(flag==1)                          %check flag to avoid appending zeroes to array
                        same3=[same3,time_cnt];          % mean of this array is taken at end of each sim
                        time_cnt=0;
                        flag=0;
                    end
                end

                %part c   30 secs is selected
                f=f+0.1;                     %this variable becomes 30 secs after 300 iterations
                set_30=union(neigh,set_30) ;     %set_30 has list of neighbors for 30 secs
                l=length(intersect(set_30,neigh));      %l has no of same neighbors every 100msec
                if(f<=30)
                    if(l>=1)
                        k=[k,l];              %after 30 secs mean of k is appended to g array
                    end
                else
                    f=0;
                    g=[g,mean(k)];            %g has mean neighbors for different 30 secs
                    k=[];
                    set_30=[];
                end
            end
        end
        if(time_cnt~=0)
            same3=[same3,time_cnt];             %at end of each sim, if any value in time_cnt
        end
        sim_conn=sim_conn+round(num_neigh/(minutes*60*10));     %average calculated once every 10 mins i.,e per each sim
        tot_same3=tot_same3+mean(same3);
        tot_same=tot_same+mean(g);
        set3=[];
    end

    %after every sim,the values are appended for plotting
    v2v_conn=round(sim_conn/runs);          %average calculated for each traffice density i.e., evry 5 sim
    x_parta=[x_parta,v2v_conn];
    avg_same3=round(tot_same3/runs);
    x_partb=[x_partb,avg_same3];         %total is in secs
    avg_same=round(tot_same/runs);
    x_partc=[x_partc,avg_same];

    sim_conn=0;                     %assigning temp variables back to 0 at end of each sim
    tot_same3=0;
    tot_same=0;

end


figure(1);
plot(traffic_density,x_parta)
title('average v2v connectivity')
ylabel('v2v connectivity') % x-axis label
xlabel('traffic density') % y-axis label
legend('comm range 50 m','Location','northwest')
legend('comm range 50 m','Location','northwest')

figure(2);
plot(traffic_density,x_partb)
title('average duration of same 3 neighbors')
ylabel('average duration in secs') % x-axis label
xlabel('traffic density') % y-axis label


figure(3);
plot(traffic_density,x_partc)
title('average number of same neighbors for 30 sec')
ylabel('average number') % x-axis label
xlabel('traffic density') % y-axis label
legend('comm range 50 m','Location','northwest')

end
```

```
% Functions

% Initializing the nodes
%intialize function takes in traffic density, 5000 metres and lane no and
%returns uniformly allocated lane
function lane = initialize(m,ytotal,lane_no)
global Smax;
global Smin;
global amax;
global amin;
global veh_id;
lane=zeros(m,4);
lane(:,1) = ytotal*rand(m,1);
lane=sortrows(lane);
lane(:,2)=Smin + (Smax - Smin)*rand(m,1);    %100 in a lane have uniformly distributed speed
lane(:,3)=amin + (amax - amin)*rand(m,1);
for i=1:m
    lane(i,4)=(lane_no-1)*m + i;
end
veh_id=lane_no*m;                   %veh_id has total no of nodes after intialization
end

% Function to insert node at entry ramps
%function insertnode takes random speed,accelartion,ramp and lane num as
%arguments and inserts a node on that lane
function l=insertnode(s,a,lane,ramp)
    global veh_id
    entry_ramps = [1510,2510,4010];
    Y=entry_ramps(ramp);
    X = lane(:,1)<Y;                    %select nodes less than entry ramp coords
    [Val,Id]=max(lane(X));              %select the nearest one of the above
    if(isempty(Id)==1)
        l=vertcat([Y,s,a,veh_id],lane(1:end,:));
    elseif(Id==size(lane,1))
        l=vertcat(lane(1:end,:),[Y,s,a,veh_id]);
    else
        l=vertcat(lane(1:Id,:),[Y,s,a,veh_id],lane(Id+1:end,:));       %add our node just after the nearest node
    end
end

% Function to remove node from exit ramp
%function deletenode takes random ramp and lane num as
%arguments and sees if there a node; if yes deletes it
function lane=deletenode(lane,ramp)
exit_ramps = [1500,2500,4000];
    global target
    Y=exit_ramps(ramp);
    x=lane(:,1)==Y;
     if(any(x))                  %select nodes less than exit ramp coords
        id=find(x);             %select the nearest one of the above
        if(lane(id,4)~=target)      %see that target node is not deleted
            lane(id,:)=[];
        end
     end
end

% Function for border effect
function lane = border(lane)
global ytotal;
%position update
lane(:,1)=lane(:,1)+lane(:,2)*(0.1)+(0.5*lane(:,3)*0.01);  %s=vt+1/2at2
x = lane(:,1)>5000;            %select nodes which crossed 5km
if(any(x))                     %if any nodes >5km
    ids=find(x);                %find ids on lane to delete them and store their ids before deleting
    temp=lane(ids,:);           %store their speed,acceleration and ids
    lane(ids,:)=[];
    for i=1:size(ids,1)
        npos=ytotal*rand(1);           %new random position
        y = lane(:,1)<npos;            %select nodes on nxt lane less than new generated position
        [val,id]=max(lane(y));
        if(isempty(id)==1)                    %if needs to be appended at start of lane
            lane=vertcat([npos,temp(i,2:end)],lane(1:end,:));    %insert into lane randomly
        elseif(id==size(lane,1))               %if needs to be appended at end of lane
            lane=vertcat(lane(1:end,:),[npos,temp(i,2:end)]);    %insert into lane randomly
        else
            lane=vertcat(lane(1:id,:),[npos,temp(i,2:end)],lane(id+1:end,:));  %insert into lane randomly
```

```
        end
    end
end
end

% Function for Freeway Mobility Model
function lane=freeway(lane, sdep, Ds)
lane(:,2)=lane(:,2)+(-1+2*rand(1))*lane(:,3);       %velocity dependance on its previous value
len=size(lane,2);
for i=1:len-1
    if(lane(i+1)-lane(i)<=Ds)                   %if within safety distance
        if(lane(i+1,2)>lane(i,2))                %check velocities of trailing and leading
            lane(i,2)=min(lane(i+1,2),sdep);    %if car within Ds, see that speed is less than its prev
        end
    end
end
end
```

2. V2V Code

```
function V2V()

    clc;
    clear all;
    workspace
    grid on;
    title('V2V Connectivity')
    xlabel('X')
    ylabel('Y')
    axis([0 1000 0 1000]);
    uiwait(msgbox('Click on two points for Road 1.', 'modal'));
    [x1, y1] = ginput(2);
    line([x1(1), x1(2)], [y1(1), y1(2)])
    slope1 = (y1(2)-y1(1))/(x1(2)-x1(1))
    uiwait(msgbox('Click on two points for Road 2.', 'modal'));
    [x2, y2] = ginput(2);
    line([x2(1), x2(2)], [y2(1), y2(2)])
    p1 = polyfit(x1, y1, 1);
    p2 = polyfit(x2, y2, 1);
    x_intersect = fzero(@(x) polyval(p1-p2,x),3);
    y_intersect = polyval(p1,x_intersect);

    hold on;
    a1 = [x1(1), y1(1)];
    b1 = [x1(2), y1(2)];
    % straight line function from a1 to b1
    func = @(xa)a1(2) + (a1(2)-b1(2))/(a1(1)-b1(1))*(xa-a1(1));
    % determine the x values
    speed = randi([10 20], 1, 1);
    no_of_points = floor(abs(((x1(1) - x1(2))/speed)));
    % divide the line based on random speed
    xa = linspace(a1(1),b1(1),no_of_points);
    % determine the y values
    ya = func(xa);

    a2 = [x2(1), y2(1)];
    b2 = [x2(2), y2(2)];
    % straight line function from a2 to b2
    func = @(xb)a2(2) + (a2(2)-b2(2))/(a2(1)-b2(1))*(xb-a2(1));
    % determine the x values
    speed = randi([10 20], 1, 1);
    no_of_points = floor(abs(((x2(1) - x2(2))/speed)));
    % divide the line based on random speed
    xb = linspace(a2(1),b2(1),no_of_points);
    % determine the y values
    yb = func(xb);

    a3 = [x2(2), y2(2)];
    b3 = [x2(1), y2(1)];
    % straight line function from a to b
    func = @(xc)a3(2) + (a3(2)-b3(2))/(a3(1)-b3(1))*(xc-a3(1));
    % determine the x values
    speed = randi([10 20], 1, 1);
    no_of_points = floor(abs(((x2(1) - x2(2))/speed)));
    xc = linspace(a3(1),b3(1),no_of_points);
    % determine the y values
```

```
    yc = func(xc);

    % get a handle to a plot graphics object
    hPlot1 = plot(NaN,NaN, 'square');
    hPlot2 = plot(NaN,NaN, 'square');
    hPlot3 = plot(NaN,NaN, 'square');
    % iterate through each point on line

    for k=1:min(min(length(xa),length(xb)),min(length(xa),length(xc)))
        % update the plot graphics object with the next position
        if (k <= length(xa))
            set(hPlot1,'XData',xa(k),'YData',ya(k));
        end
        if (k <= length(xb))
            set(hPlot2,'XData',xb(k),'YData',yb(k));
        end
        if (k <= length(xc))
            set(hPlot3,'XData',xc(k),'YData',yc(k));
        end
        ab = pdist ([xa(k), ya(k);xb(k), yb(k)],'euclidean');
        bc = pdist ([xb(k), yb(k);xc(k), yc(k)],'euclidean');
        ac = pdist ([xa(k), ya(k);xc(k), yc(k)],'euclidean');
        if(ab <=100)
            one = plot ([xa(k), xb(k)], [ya(k), yb(k)], '--g');
        end
        if(bc <=100)
            two = plot ([xb(k), xc(k)], [yb(k), yc(k)], '--g');
        end
        if(ac <=100)
            three = plot ([xa(k), xc(k)], [ya(k), yc(k)], '--g');
        end
        % pause for 0.3 second
        pause(0.3);
        if exist('one', 'var')
            delete(one)
        end
        if exist('two', 'var')
            delete(two)
        end
        if exist('three', 'var')
            delete(three)
        end
    end
end
```