NetSim - MATLAB Interfacing for UAV/Drone/Flying Adhoc, network simulations.

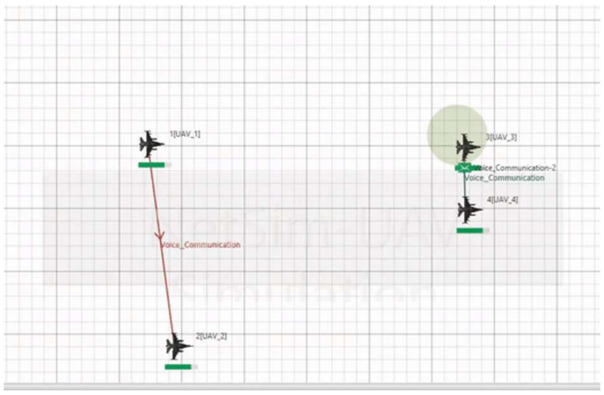
Applicable Versions	**NetSim Pro**	**NetSim Standard**
Applicable Releases	**v12.2**	
MATLAB Version	**R2019b or above**	
MATLAB Toolbox	MATLAB, Simulink, Robotics and System Toolbox	
FE Project	Robotics System Toolbox UAV Library	
Visual Studio	Community Edition	

Contents

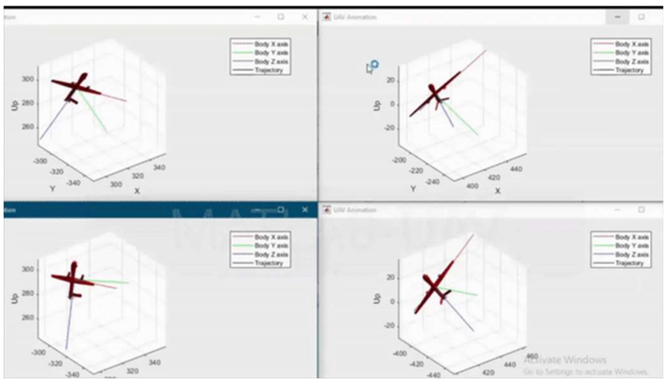
Objective

In this article, we are going to explain how users can simulate UAV device in NetSim by interfacing with MATLAB's Simulink.

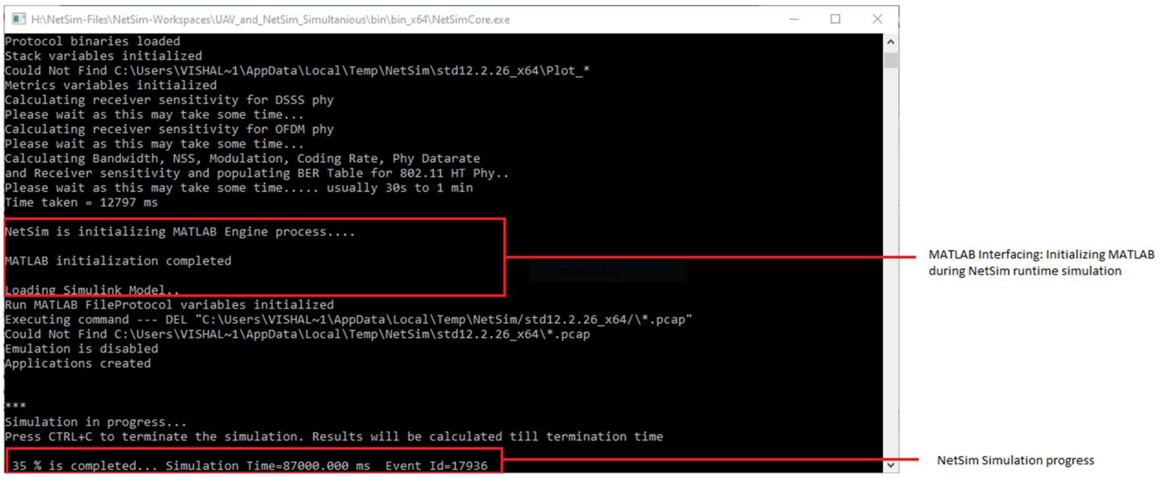
Infographic:



2D- Animation in NetSim



3D Animation in MATLAB



```
H:\NetSim-Files\NetSim-Workspaces\UAV_and_NetSim_Simultanious\bin\bin_x64\NetSimCore.exe
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\VISHAL~1\AppData\Local\Temp\NetSim\std12.2.26_x64\Plot_*
Metrics variables initialized
Calculating receiver sensitivity for DSSS phy
Please wait as this may take some time...
Calculating receiver sensitivity for OFDM phy
Please wait as this may take some time...
Calculating Bandwidth, NSS, Modulation, Coding Rate, Phy Datarate
and Receiver sensitivity and populating BER Table for 802.11 HT Phy..
Please wait as this may take some time..... usually 30s to 1 min
Time taken = 12797 ms

NetSim is initializing MATLAB Engine process....

MATLAB initialization completed

Loading Simulink Model..
Run MATLAB FileProtocol variables initialized
Executing command --- DEL "C:\Users\VISHAL~1\AppData\Local\Temp\NetSim/std12.2.26_x64/\*.pcap"
Could Not Find C:\Users\VISHAL~1\AppData\Local\Temp\NetSim\std12.2.26_x64\*.pcap
Emulation is disabled
Applications created


***
Simulation in progress...
Press CTRL+C to terminate the simulation. Results will be calculated till termination time

 35 % is completed... Simulation Time=87000.000 ms  Event Id=17936
```

MATLAB Interfacing: Initializing MATLAB during NetSim runtime simulation
NetSim Simulation progress

NetSim Simulation Window

NetSim - Simulink Interfacing

Upon interfacing NetSim with MATLAB the following tasks are performed during simulation start:

- MATLAB Engine process is initialized
- MATLAB Desktop window is loaded
- SIMULINK Model is loaded

Upon simulating a network created in NetSim the following tasks are performed periodically:

- SIMULINK Model is simulated
- SIMULINK Model is paused
- NetSim reads the data generated by SIMULINK from MATLAB workspace
- Appends the readings to the packet payload as packets are formed

During the Simulation, the SIMULINK Model is started and paused several times for NetSim and SIMULINK simulations to run synchronously. The readings obtained from SIMULINK are read from

MATLAB workspace and appended to the payload of the packets generated in NetSim. In this example, readings are taken every one second and appended to the packet payload.

Output/Metrics specific to this example

- NetSIm Animation- Mobility of the devices configured in NetSim is given as input from MATLAB

Modifications done to NetSim Source codes:

Project: Mobility
Files:

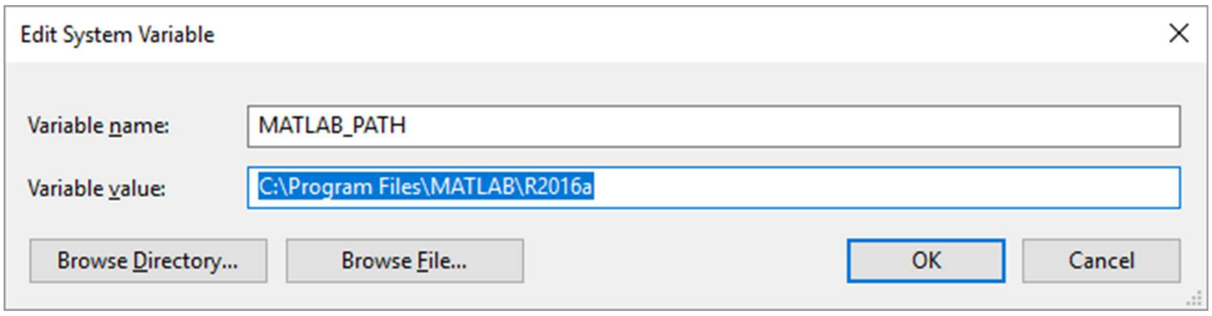
- Mobility.c,
- Mobility.h,
- Added - UAVBasedMobility.c
- Mobility.vcxproj (Project file)

Sections of source code modified

- Mobility.c
  - fn_NetSim_Mobility_Init(): call to init_uav() function
  - fn_NetSim_Mobility_Run(): Call to uav_run() function
  - fn_NetSim_Mobility_Finish(): Call to finish_uav() function
- Mobility.h
  - MATLAB Engine variable - Used to initiate and interact with MATLAB Engine process
- Mobility.vcxproj - This is a Visual Studio project file used to load and manage the source codes related to the Mobility in NetSim
  - path to MATLAB application
  - path to MATLAB include directory
  - path to MATLAB lib directory
  - information related to dependent MATLAB library files
- UAVBasedMobility.c
  - init_uav(): Initializes MATLAB, Loads SIMULINK Model, starts and pauses SIMULINK simulation , and initializes the UAV devices in MATLAB to start simulation along with NetSim's simulation.
  - uav_run(): Starts NetSim and MATLAB simulation simultaneously and gets the co-ordinates from MATLAB workspace for every step size set in NetSim.
  - uavcorr(): Function to get co-ordinates from MATLAB.

Procedure to setup MATLAB for this example

- Create a system environment variable with name as "**MATLAB_PATH**" and value as "**<path of MATLAB Root directory>**"


Edit System Variable

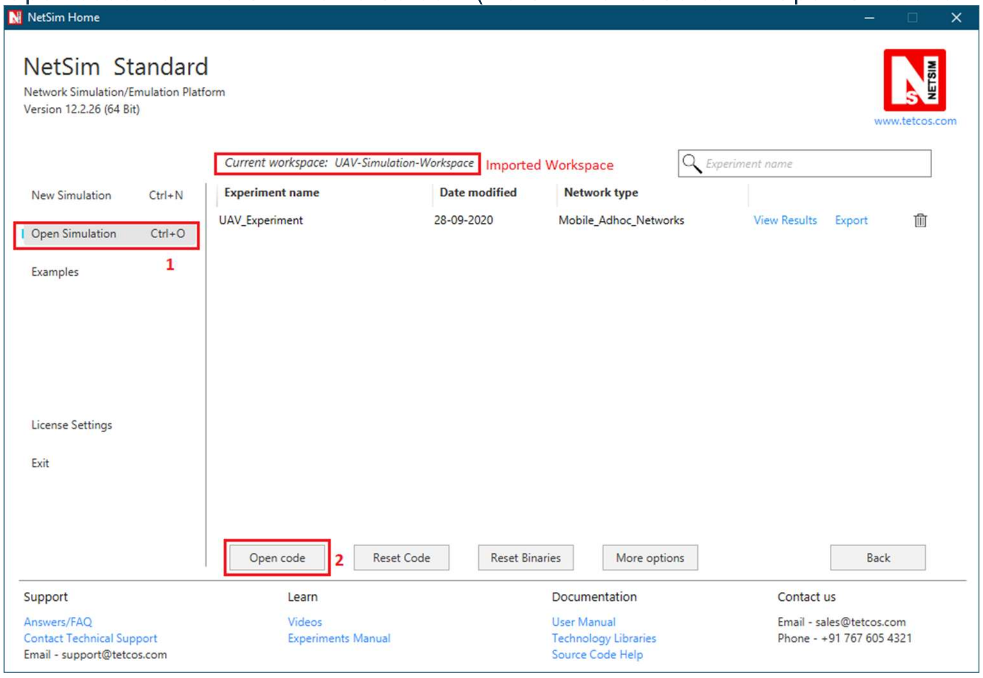
Variable name: MATLAB_PATH

Variable value: C:\Program Files\MATLAB\R2016a

Note: The procedure to create an environment variable and the path of the MATLAB root directory will vary based on the Windows OS version and the MATLAB software version respectively.

Procedure to setup NetSim for this example

- Download the attachment containing both NetSim Workplace and MATLAB FIles
- Import the workspace to NetSim. *<Article on how to import workspace in NetSim>*
- Place MATLAB files in the desired location of your wish. Make sure that the location is both readable and writeable
- Replace GenerlProperty.xml file present in NetSim-Files folder in the attachment in <NetSim-Installation-Directory>/Docs/XML folder.
- Open code from NetSim Home Screen. (NetSim Home Screen > Open Simulation > Open code)


NetSim Home

NetSim Standard

Network Simulation/Emulation Platform

Version 12.2.26 (64 Bit)

Current workspace: UAV-Simulation-Workspace

Experiment name	Date modified	Network type
UAV_Experiment	28-09-2020	Mobile_Adhoc_Networks

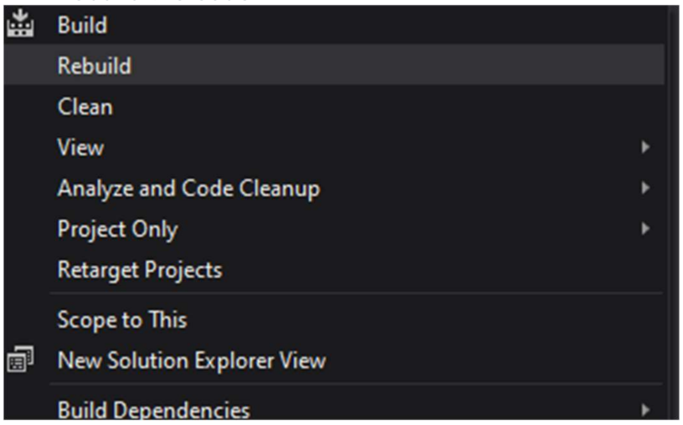
Support: Answers/FAQ, Contact Technical Support, Email - support@tetcos.com

Learn: Videos, Experiments Manual

Documentation: User Manual, Technology Libraries, Source Code Help

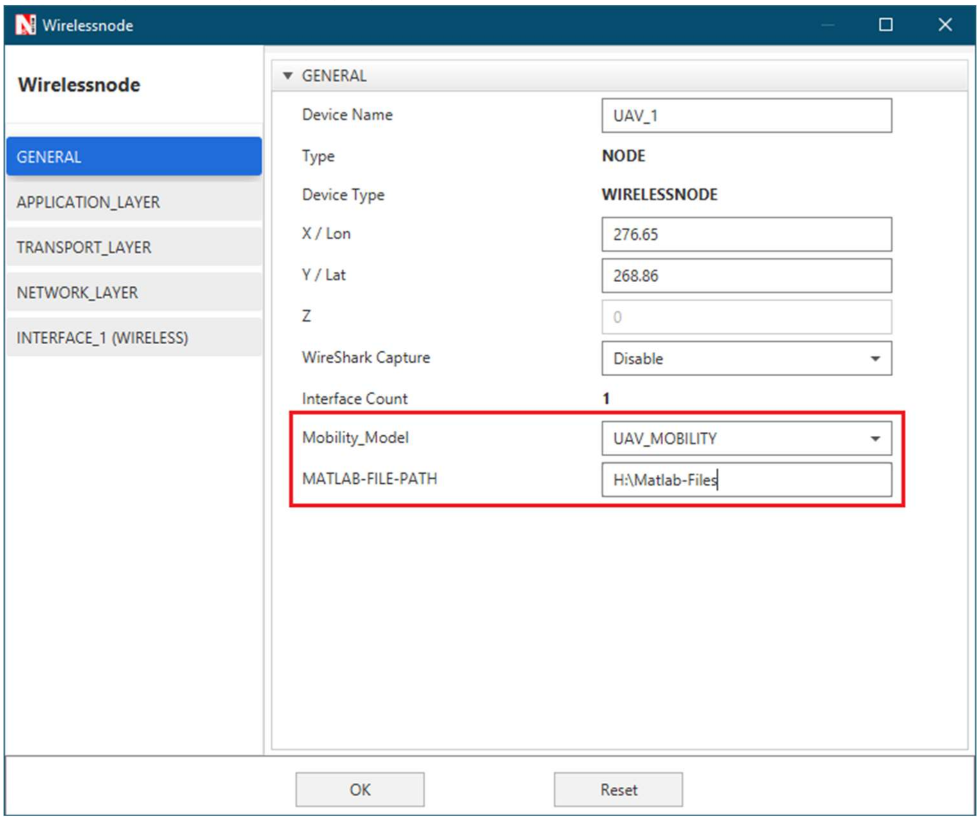
Contact us: Email - sales@tetcos.com, Phone - +91 767 605 4321

- Rebuild the code



Running the NetSim scenario with UAV Based Mobility

- Before opening Example make sure that you have replaced the XML file in NetSim Installation directory mentioned above and then Open Example saved in the workspace.
- Make sure that all devices are UAV Mobility enabled and update path to MATLAB files that you downloaded with the attachement



Wirelessnode

Wirelessnode

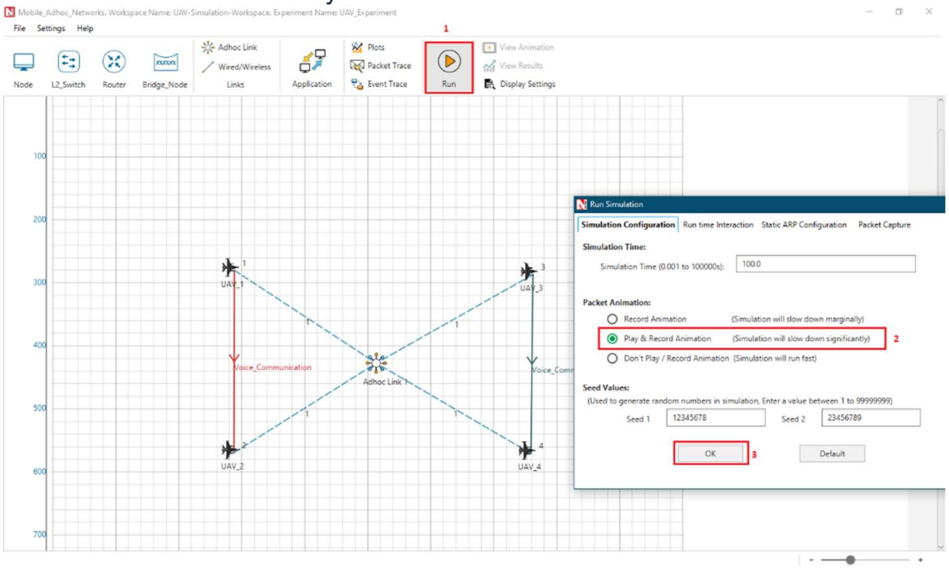
GENERAL | APPLICATION_LAYER | TRANSPORT_LAYER | NETWORK_LAYER | INTERFACE_1 (WIRELESS)

GENERAL

Device Name	UAV_1
Type	NODE
Device Type	WIRELESSNODE
X / Lon	276.65
Y / Lat	268.86
Z	0
WireShark Capture	Disable
Interface Count	1
Mobility_Model	UAV_MOBILITY
MATLAB-FILE-PATH	H:\Matlab-Files

OK Reset

- Run the simulation with Play and record animation in order to have both Animations simultaneous



Mobile_Adhoc_Networks. Workspace Name: UAV-Simulation-Workspace. Experiment Name: UAV_Experiment

Run Simulation

Simulation Configuration | Run time Interaction | Static ARP Configuration | Packet Capture

Simulation Time:

Simulation Time (0.001 to 100000s): 100.0

Packet Animation:

- ○ Record Animation (Simulation will slow down marginally)
- [✓ ● Play & Record Animation (Simulation will slow down significantly)]
- ○ Don't Play / Record Animation (Simulation will run fast)

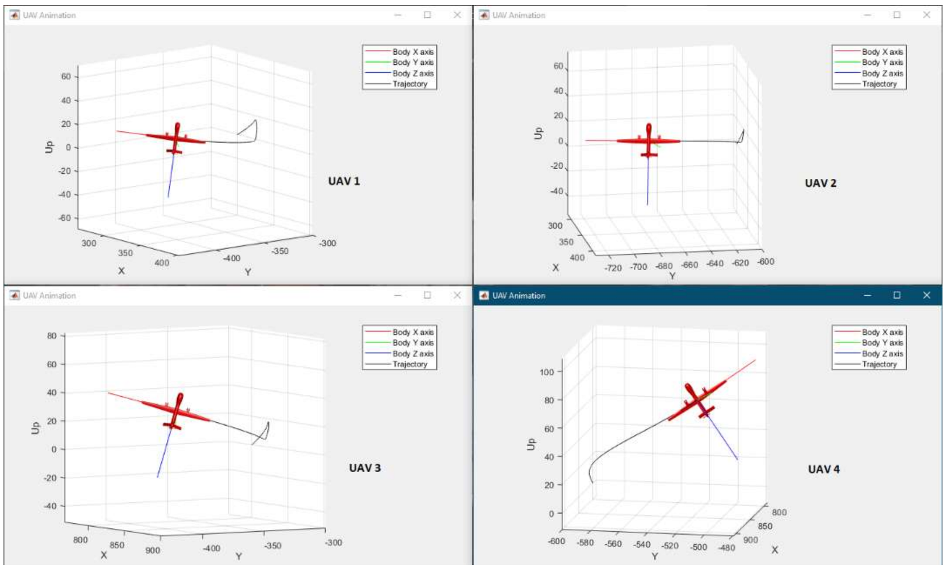
Seed Values:

(Used to generate random numbers in simulation, Enter a value between 1 to 99999999)

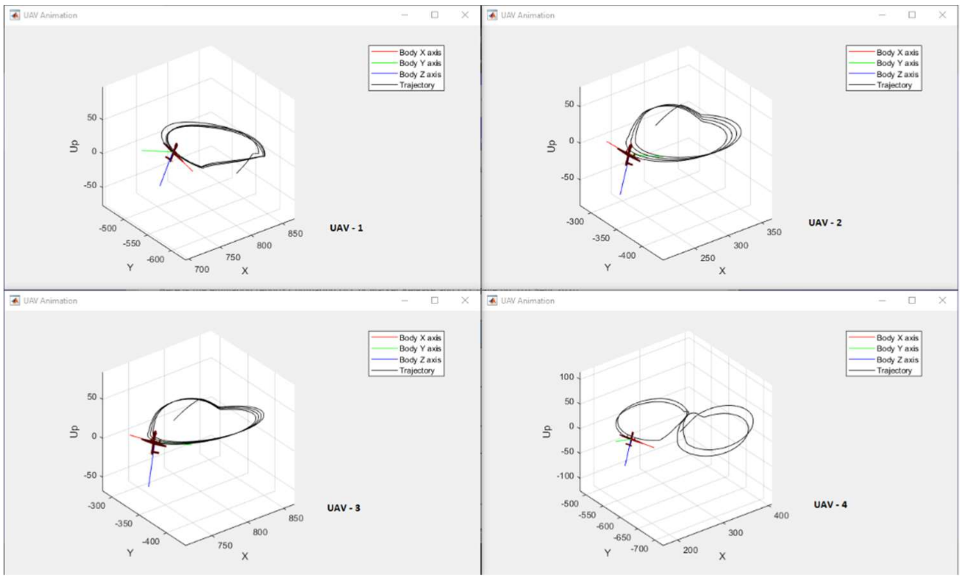
Seed 1: 12345678 Seed 2: 23456789

OK Default

- MATLAB Animation



End of Simulation:



Code modifications in NetSim

A new file UAVBasedMobility.c is added to the **Mobility** Project, which contains the following source code:

```
#include "main.h"

#include "Mobility.h"

#include "Animation.h"

double* uavcorr(char* id);

char* matlabloc; //Used to get MATLAB files path



/*init_uav() function is to initialize MATLAB Engine and start UAVs*/
```

```
void init_uav()

{

		char wp[100];

		char buf[100] = "cd ";

		strcpy(wp, matlabloc);

		strcat(buf, wp);

		static int nouav = 1;			// Declared as static, since we want it to be declared and
changed only once

		if (nouav)							// This will run only at the 1st time

		{

				//MATLAB/SIMULINK INTERFACING

				fprintf(stderr, "\nNetSim is initializing MATLAB Engine process....\n");

				if (!(ep = engOpen(NULL))) {

						MessageBox((HWND)NULL, (LPCWSTR)"Can't start MATLAB
engine",(LPCWSTR)"MATLAB_Interface.c", MB_OK);

						fprintf(stderr, "\nMATLAB Initialization Failed\nPress any key to
proceed without MATLAB...\n");

						_getch();

				}

				else

				{

						engEvalString(ep, "desktop");

						engEvalString(ep, buf); //Update user-path

						fprintf(stderr, "\nMATLAB initialization completed\n");

						fprintf(stderr, "\nLoading Simulink Model..");

						engEvalString(ep, "opSimulink");// Loads all UAV in MATLAB

				}

				//MATLAB/SIMULINK INTERFACING

				nouav = 0;

		}

}
```

```
/*This function is used to receive co-ordinates from MATLAB during run time*/

void uav_run()

{

        MOBILITY_VAR* pstruMobilityVar = (MOBILITY_VAR*)NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruMobVar;      //Define Mobility variable

        double dPresentTime = pstruMobilityVar->dLastTime;



        memcpy(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruCurrentPosition,

                NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruNextPosition,

                sizeof * NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruCurrentPosition);



        if (pstruMobilityVar->dLastTime + pstruMobilityVar->dPauseTime * 1000000 < pstruEventDetails->dEventTime + 1000000)            //Everytime Mobility being called

        {

                double* coordinates;                // Pointer for array of X and Y coordinates

                coordinates = uavcorr(pstruEventDetails->nDeviceId);            //Get coordinates from matlab

                if (coordinates != NULL)

                {

                        NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruNextPosition->X = coordinates[0]; // Update the coordinates in Network stack

                        NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruNextPosition->Y = coordinates[1];

                        NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]->pstruDeviceMobility->pstruNextPosition->Z = coordinates[2];

                        free(coordinates);                                // Free memory of pointer

                }



                //store the last time
```

```
				pstruMobilityVar->dLastTime = pstruEventDetails->dEventTime + 100;
				// Update Last time since we want to match timings with MATLAB

		}

		//update the device position

		memcpy(NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]-
>pstruDevicePosition,

				NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]-
>pstruDeviceMobility->pstruCurrentPosition,

				sizeof * NETWORK->ppstruDeviceList[pstruEventDetails->nDeviceId - 1]-
>pstruDevicePosition);



		mobility_pass_position_to_animation(pstruEventDetails->nDeviceId,

				pstruEventDetails->dEventTime,

				DEVICE_POSITION(pstruEventDetails->nDeviceId));



		//Add event for next point

		pstruEventDetails->dEventTime += (1* SECOND);

		fnpAddEvent(pstruEventDetails);

		pstruEventDetails->dEventTime -= (1 * SECOND);

}


double* uavcorr(int id)

{

		double* coordinates;

		char buf[100];

		mxArray* xmat = NULL;

		mxArray* ymat = NULL;

		mxArray* zmat = NULL;

		double* xcor = NULL;

		double* ycor = NULL;

		double* zcor = NULL;

		if (id == 1)
```

```
                engEvalString(ep, "set_param('UAV1','SimulationCommand','pause');");
else if (id == 2)
                engEvalString(ep, "set_param('UAV2','SimulationCommand','pause');");
else if (id == 3)
                engEvalString(ep, "set_param('UAV3','SimulationCommand','pause');");
else
                engEvalString(ep, "set_param('UAV4','SimulationCommand','pause');");

engEvalString(ep, "[xa,c]=size(North)");
engEvalString(ep, "x_out = North(xa, :)");
engEvalString(ep, "[ya, c] = size(East)");
engEvalString(ep, "y_out = East(ya, :)");
engEvalString(ep, "[za, c] = size(Height)");
engEvalString(ep, "z_out = Height(za, :)");
xmat = engGetVariable(ep, "x_out");
ymat = engGetVariable(ep, "y_out");
zmat = engGetVariable(ep, "z_out");
xcor = mxGetPr(xmat);
ycor = mxGetPr(ymat);
zcor = mxGetPr(zmat);
coordinates = (double*)malloc(3 * sizeof * coordinates);
coordinates[0] = xcor[0];
coordinates[1] = ycor[0];
coordinates[2] = zcor[0];

if (id == 1)
                engEvalString(ep, "set_param('UAV1','SimulationCommand','continue');");
else if (id == 2)
                engEvalString(ep, "set_param('UAV2','SimulationCommand','continue');");
else if (id == 3)
```

```
                    engEvalString(ep, "set_param('UAV3','SimulationCommand','continue');");
          else
                    engEvalString(ep, "set_param('UAV4','SimulationCommand','continue');");
          return (coordinates);
}
```

Generic

MATLAB Code: OpSimulink.m

```
% This function is used to initialize and start Simulink model called in NetSim- init_uav() function.


% Initiating UAV 1
model='UAV1';
load_system(model);
set_param(model,'SimulationCommand','start')
set_param(model,'SimulationCommand','pause')


% Initiating UAV 2
model1='UAV2';
load_system(model1);
set_param(model1,'SimulationCommand','start')
set_param(model1,'SimulationCommand','pause')


% Initiating UAV 3
model2='UAV3';
load_system(model2);
set_param(model2,'SimulationCommand','start')
set_param(model2,'SimulationCommand','pause')


% Initiating UAV 4
model3='UAV4';
```

```
load_system(model3);

set_param(model3,'SimulationCommand','start')

set_param(model3,'SimulationCommand','pause')
```