**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 2](#_Toc52240311)

[Infographic: 2](#_Toc52240312)

[NetSim - Simulink Interfacing 2](#_Toc52240313)

[Output/Metrics specific to this example 3](#_Toc52240314)

[Modifications done to NetSim Source codes: 3](#_Toc52240315)

[Sections of source code modified 3](#_Toc52240316)

[Procedure to setup MATLAB for this example 3](#_Toc52240317)

[Procedure to setup NetSim for this example 4](#_Toc52240318)

[Running the NetSim scenario with UAV Based Mobility 5](#_Toc52240319)

[Code modifications in NetSim 6](#_Toc52240320)

[MATLAB Code: OpSimulink.m 11](#_Toc52240321)

# 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** |
|  | |
| **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**:

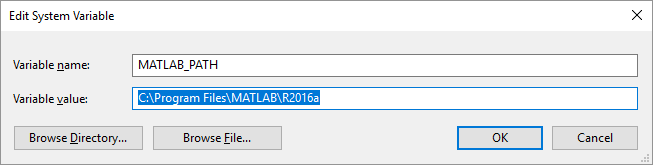
* Mobility.c,
* Mobility.h,
* Added - UAVBasedMobility.c
* Mobility.vcxproj (Project file)

# 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.

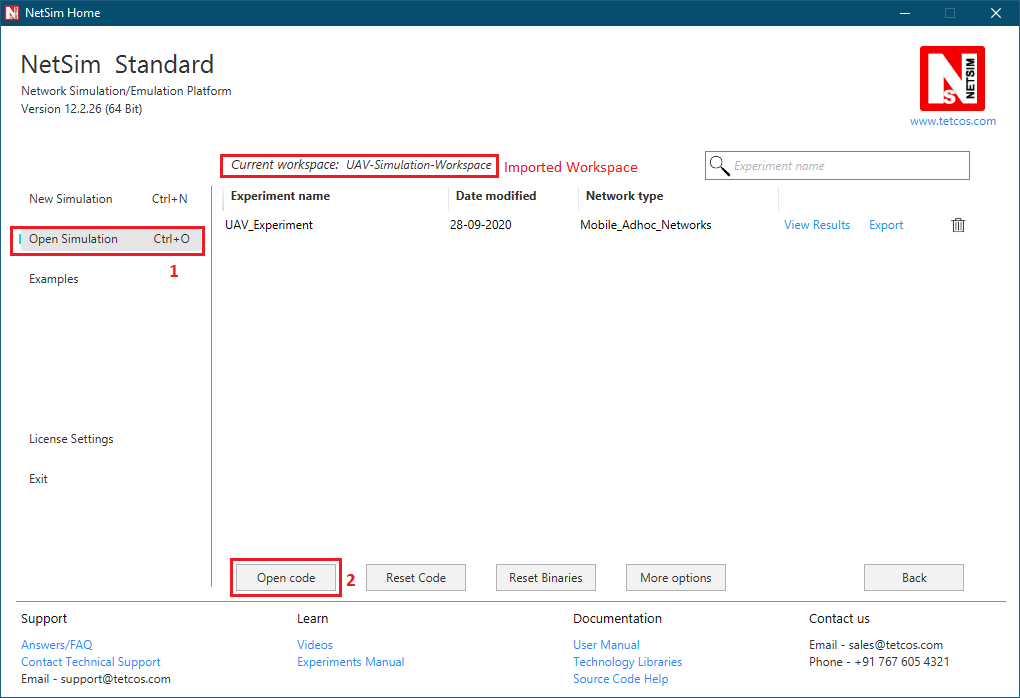
# Procedure to setup MATLAB for this example

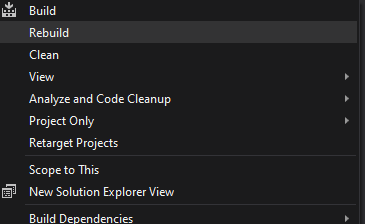
* Create a system environment variable with name as "**MATLAB\_PATH**" and value as "**<path of MATLAB Root directory>**"



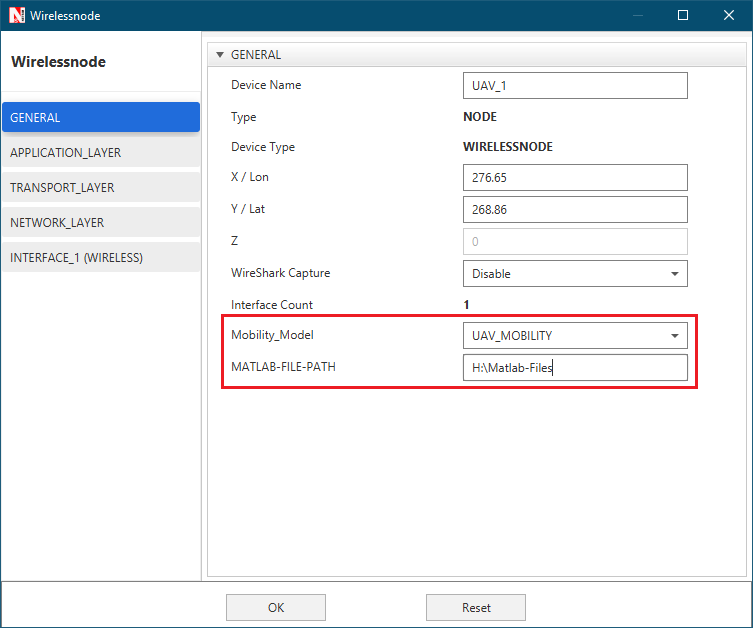
**Note**: The procedure to create an environment variable and the path of the MATLAB root directory will vary based on the Windows OS version and the MATLAB software version respectively.

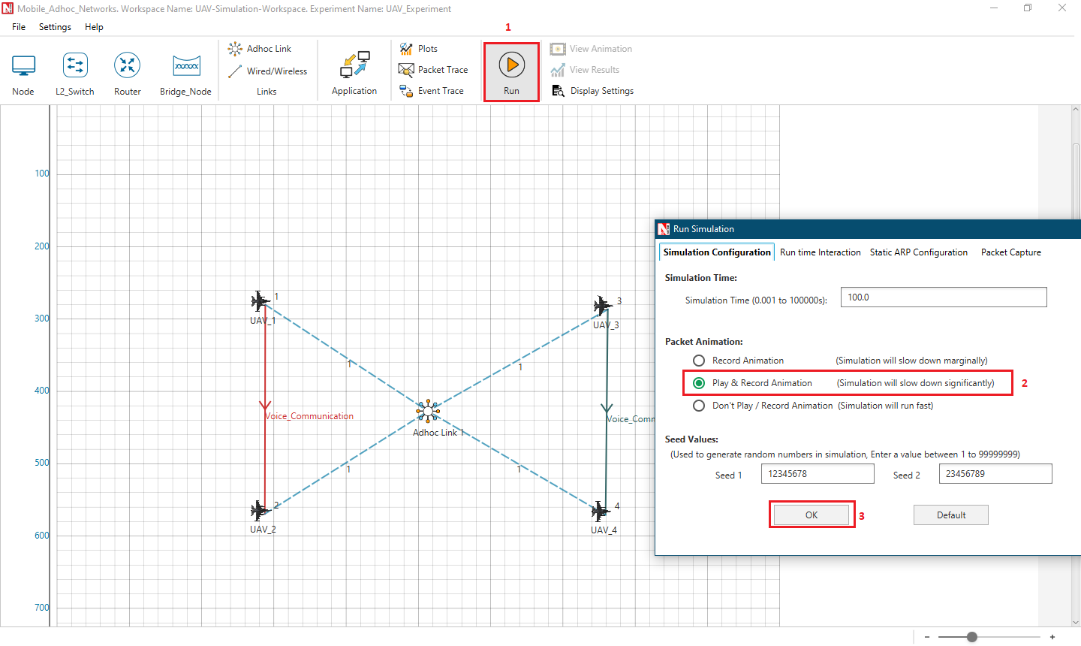
# Procedure to setup NetSim for this example

* 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)
* Rebuild the code

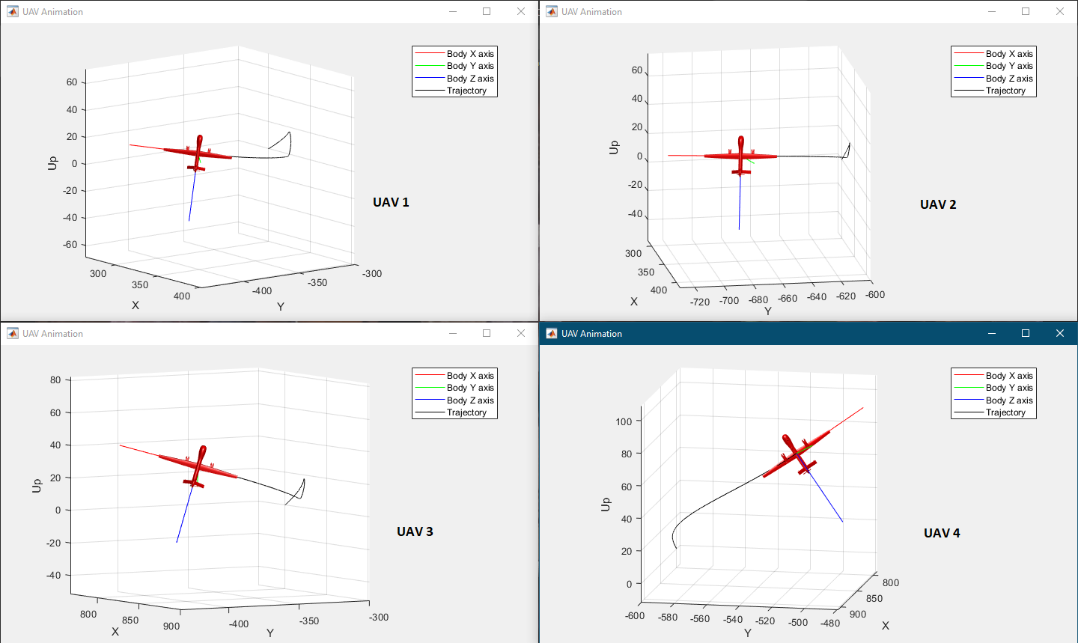


# Running the NetSim scenario with UAV Based Mobility

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



* 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')