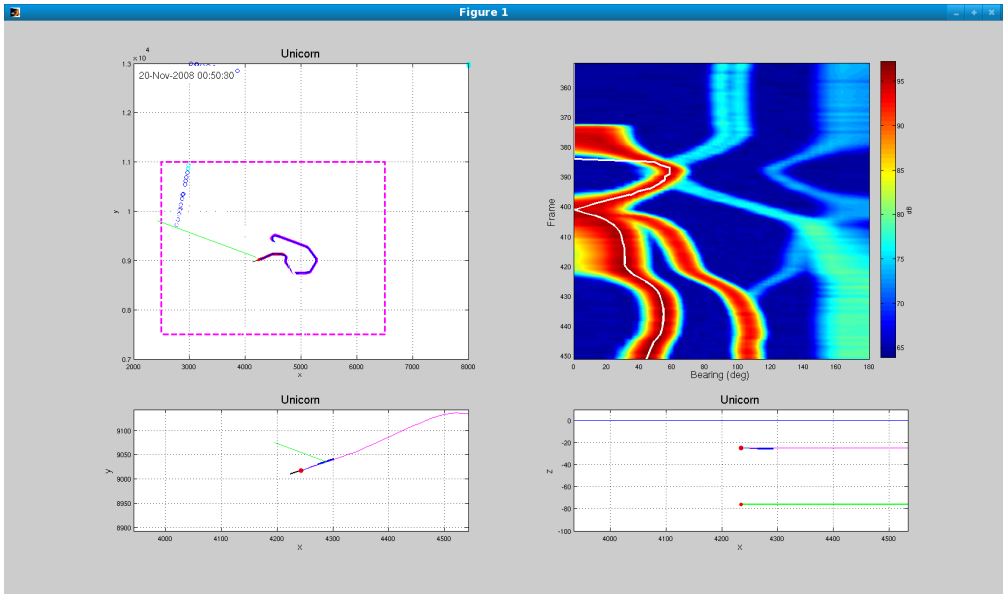

MOOS-IvP

Undersea Autonomous Network Simulator User's Guide

H. Schmidt, M. R. Benjamin, A. Balasuriya, K. Cockrell and R. Lum

Laboratory for Autonomous marine Sensing
Department Mechanical Engineering
Massachusetts Institute of Technology, Cambridge MA

December 8, 2008

Abstract

This paper provides a brief User's Guide for the MIT MOOS-IvP Undersea Autonomous Network Simulation environment.

Contents

Part I
Background and Introduction

1 Overview

1.1 Purpose and Scope of this Document

The purpose of this document is to provide a catalog style overview of modules used for creating a high-fidelity MOOS-IvP simulation environment for networks of autonomous underwater vehicles and surface craft, and to provide aan uintroductory guide to operating the simulator. The scope of discussion includes, for each module, a brief description of the module function, authorship, source for download, rough measure of complexity, and module dependencies. Further, for use by developers of onboard processing modules, for example, the description includes a detailed listing of MOOS variables published by or subscribed to by each simulator module.

1.2 Sonar AUV MOOS Community

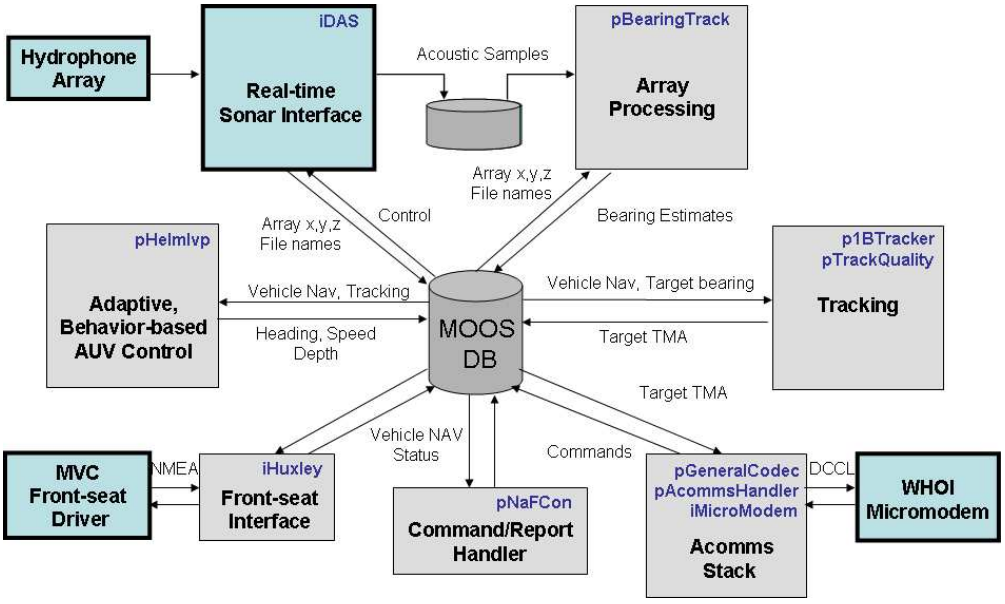

Figure 1: MIT MOOS-IvP Sonar-AUV Community: Principal MOOS-IvP modules shaded in grey, incorporating the IvP-Helm, the handlers af commands and reports communicated with field control, and onboard signal processing. The three principal interfaces to the 'outside world' are shaded in blue: 1. The main vehicle computer (MVC) containing the 'front-seat driver' which handles the lower level AUV navigation and control. 2. The WHOI Micromodem which handles all sub-surface network communication. 3. The sonar, consisting of a receiving hydrophone array, for active systems a source, and a control and data acquisition system (iDAS).

The MIT Laboratory for Autonomous Marine Sensing (LAMS) operates two Bluefin21 AUVs for research into adaptive and collaborative, autonomous acoustic sensing in the ocean. The vehicles, called Unicorn, and Caribou or Macrura depending on the configuration, can be carrying a

variety of customised acoustic source/receiver payloads. In configurations for seabed object detection, classification and localization the two vehicles each have a 16-element fixed hydrophone array mounted in the front section, and an active source for insionifying the seabed. For passive and multistatic acoustic research below 1 kHz, Unicorn is integrated with a 32-element towed hydrophone array.

In either case, the vehicles are operating with MOOS-IvP handling the higher-level intelligent autonomy, implemented on a computer stack in the payload section, integrated with the acoustic source control and the data acquisition system. The MOOS-IvP configuration is shown schematically in Fig. 1. The MOOS-IvP Autonomy System is connected to the main vehicle computer using the 'back-seat driver' paradigm, consistent with the proposed ASTM F41 standard for AUVControl. The MOOS interface to the front-seat driver is platform dependent, with several existing in the source tree, for example `iHuxley` for BF21 and `iOEX` for the Ocean Explorer. Other connections from the payload include the WHOI Micromodem, which forms the backbone of the undersea commmunication network. The modem communication is handled by the MOOS communication stack, consisting of a generic message coder-decoder `pGeneralCodec`, and the `pAcommsHandler`process handling message queueing to and from the modem driver module `iMicroModem`. The main purpose of the sonar payload is obviously to operate the acoustic sources and record and process data from the hydrophone array. The interface is provided by a dedicated data acquisition MOOS module `iDAS` (Digital Acquisition System), which interacts with the signal conditioning for each array element. `iDAS` subscribes for control parameters and commands from the MOOSDB, and publishes array element locations and a filename, every time a new snippet of data is available. The high-bandwidth nature of the raw acoustic data makes it inconvenient to publish these directly in the MOOSDB. Instead they are written to a file, which also serves as permanent recording, and only the file location and name is published, for other processes to access, such as the signal processing module, which for acoustic arrays will typically derive a bearing estimate for a source or a scatterer and publish the result in the MOOSDB for other processing such as tracking.

In addition to these three main connections, there is also a process `iCTD` controlling and receiving data from the CTD sensor which is standard for the vehicle. Also, some of the payloads have their own GPS antenna for accurate timing, and here the MOOS interfacing is handled by the `iGPS` process.

Part II
MIT Distributed Network Simulation Environment

2 MIT Undersea Autonomous Network Simulator

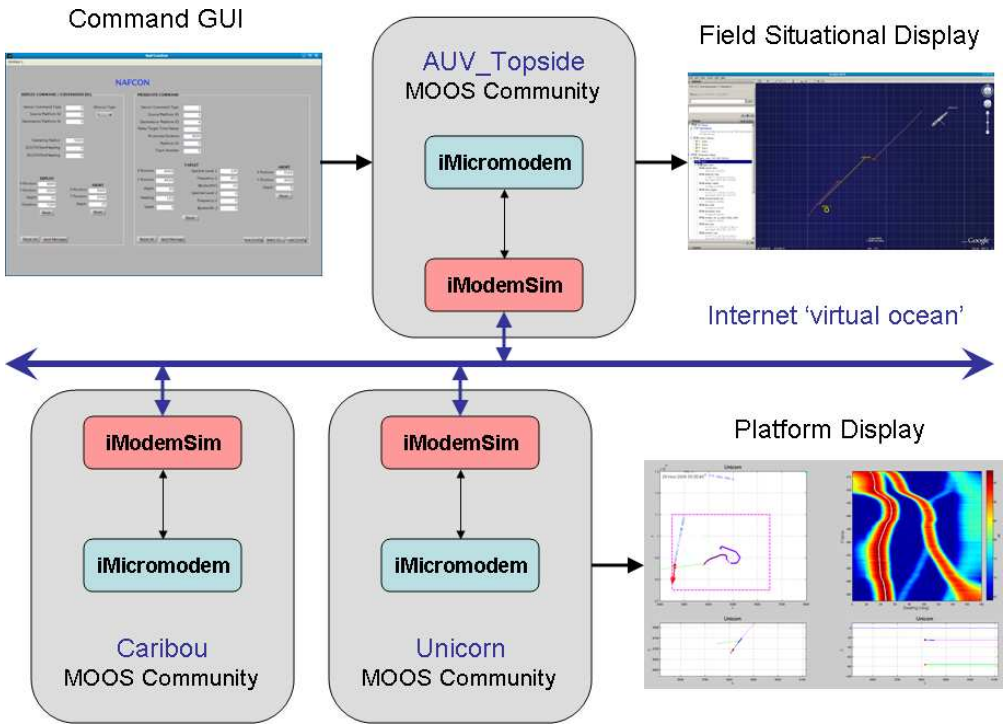

Figure 2: Architecture of MOOS-IvP Undersea Autonomous Network Simulator. The topside MOOS Community, identical to that applied in the field, is connected to all MOOS communities on the local network, including any number of AUV and ASC communities, communicating using the WHOI Micromodem through the `iMicroModem` MOOS utility.

The cornerstone of the MIT multi-node undersea network simulator is the MOOS utility `iModemSim`, which simulates an undersea micromodem network used in the field as shown in Fig. **??** by transparently connecting all MOOS-communities on the local area network, or the same computer, running this process. The utility incorporates physically realistic message transmission uncertainty, intermittency and latency, as well as message collision, and therefore provides a high-fidelity 'virtual ocean' environment for realistic simulation of an undersea network. The network simulation architecture is shown in Fig. 2.

The topside command and control MOOS community `AUV_Topside` is operated unchanged from that used in field deployments, except for the physical serial port to the WHOI micromodem gateway being replaced by a virtual serial port to the `iModemSim` process. Thus, the topside command and control GUIs and the situational and nodal status displays are transparently representing an

actual operational environment.

Similarly, the MOOS Communities for an arbitrary number of underwater vehicles, or surface craft are running on one or more desktop or laptop computers - or on the actual vehicle payload - connected to the 'virtual ocean' by `iModemSim`. Each 'virtual vehicle' are commanded from the topside using CCL modem commands with realistic throughput statistics, and will transmit status reports via the virtual modems to the network when polled to do so. The architecture allows for all virtual nodes to share environmental and situational information. Thus, the environment allows for fully realistic simulation of adaptive and collaborative autonomy by the network assets.

In addition to the field-level display tools applied by the topside community, the simulator incorporates several graphical tools for real-time situational display of nodal information, including results of processed sensor data, such as the inboard processing of acoustic data collected by hydrophone arrays on sonar AUVs, as described in the following.

2.1 Sonar-AUV Simulator

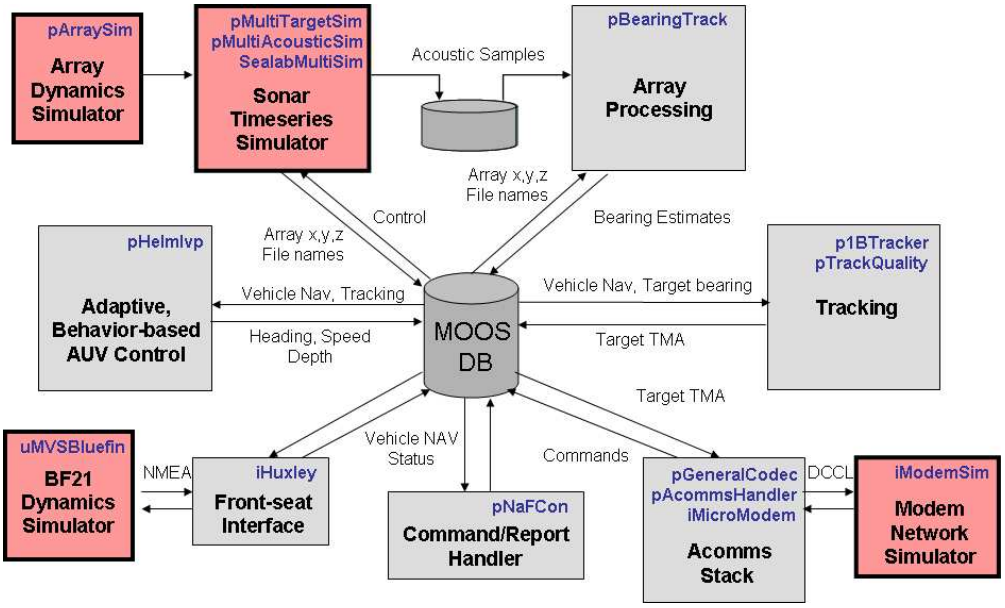

Figure 3: MOOS-IvP Sonar-AUV Simulation Environment: The simulation environment is identical to the AUV payload MOOS environment, except for the 'outside world' interfaces, which are replaced by a set of simulation tools which use the same interface definitions as the actual vehicle hard- and software, making it transparent to the MOOS=-IvP environment whether it is operating in the vehicle or a stand-alone computer. .

The power of the MOOS-IvP simulation environment is that all processes can be operated unchanged, as long as the hardware and software handling the interface between actuators and sensors and the MOOSDB are replaced by dedicated simulation modules publishing and subscribing to the same MOOS variables as the on-board interfaces. Thus, a full simulation capability has been established developing a set of payload connection simulators, which can replace the actual hardware and software transparently to all other MOOS processes connected to the MOOSDB, as illustrated schematically in Fig. 3, where the blue-shaded inboard systems in Fig. 1 are replaced by their red-shaded simulator equivalents. Also, the simulation modules may be used in any combination

with the real system components. Thus, the CTD simulator has been used on a vehicle during scientific missions with a broken CTD unit, due to the fact that the BF21 front-seat driver requires CTD data for operating. Also, the target simulator has been applied in the field in cases of a malfunctioning acoustic array.

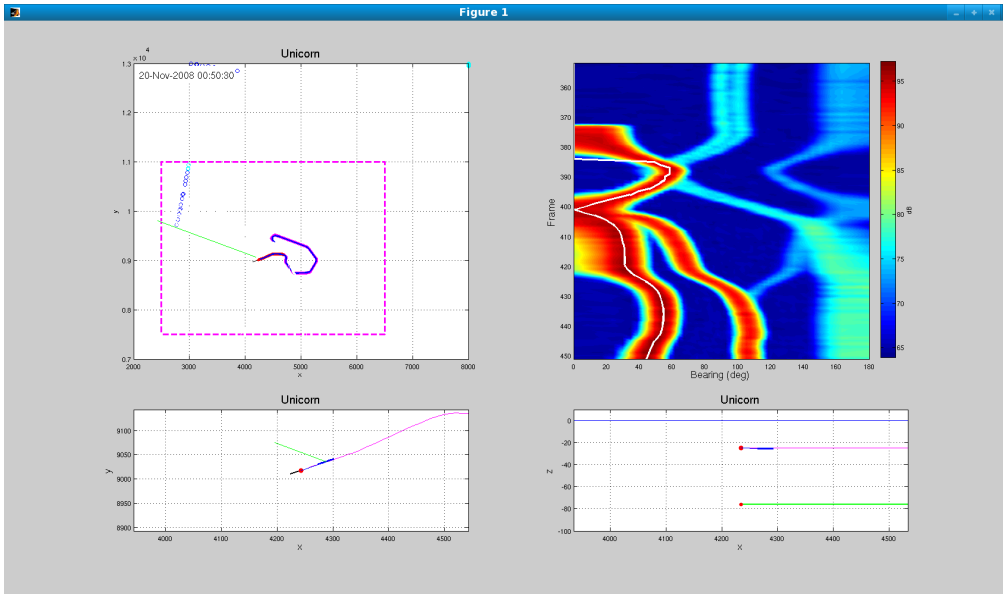

Figure 4: `small_uVis.m:` Real-time display of acoustic sensing mission. The upper left frame shows the auv/array and target history for a sensing mission simulation. The two lower frames show zooms of the auv/array geometry in the horizontal and vertical, while the upper right shows the computed Beam-Time Record (BTR) achieved from the simulated time series, with the real-time stabilized bearing-track estimate indicated by the white curve..

In addition to the system simulator modules, the source tree incorporates several graphical tools for use with the simulator, in addition to the standard topside displays described elsewhere. Thus, the MATLAB script `small_uVis.m` generates a real-time display of auv/array dynamics, and real-time acoustic processing results, as shown in Fig. 4.

Another MATLAB module, `SourceSim.m` activates a GUI, shown in Fig. 5 for adding new acoustic targets to be simulated by `pMultiTargetSim`. Each new target is identified by an initial position, speed and heading, and its acoustic properties, and added to the list of active targets maintained by `pMultiTargetSim`, when activated with 'Apply'..

A complete list of the simulation modules are given in Tables 1 and 2. A more detailed description is given in the following chapter.

2.2 Running a Simulation Session

When running a simulation session it is recommended to assign a desktop to each AUV or ASC, and two for the topside: one for the situational display, one for the command GUIs. The first step is to prepare the mission files for use with the simulator.

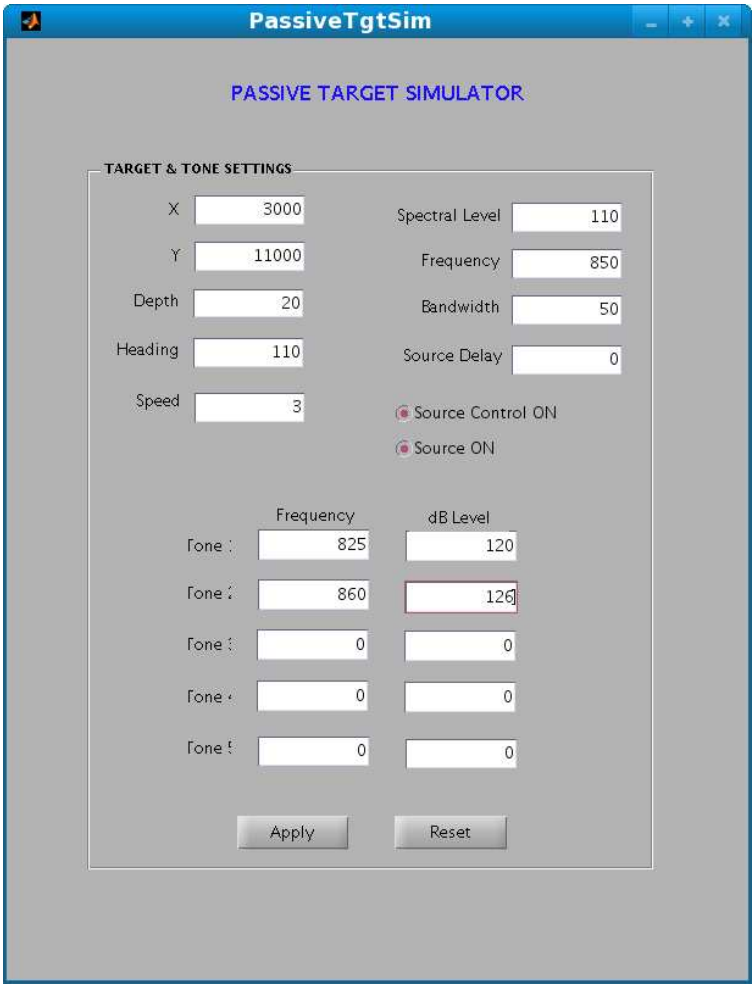

Figure 5: `PassiveTgtSim.m`: GUI for activating acoustic sources to be dynamically simulated by `pMultiTargetSim`.

#	Module Name	Module Description	Author	Size
1	**uMVSBluefin**	Dynamic simulator for Bluefin21 AUV	Battle	
2	**pArraySim**	Towed array simulator used together with uMVSBluefin for coupled dynamics simulation. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Schmidt	4,803
3	**iModemSim**	Simules a modem network. Communicates with iModemSim in all MOOS communities on local network. Handles propagation latency and transmission loss, and collisions. *Libraries: anrp_util, MOOS, MOOSGen, MOOSUtility*	Patrikalakis	1,474 **24,655**
4	**pTargetSim**	Single target simulator used in conjunction with the low-fidelity target bearing simulator **pBearingSim** *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Cockrell	
5	**pMultiTargetSim**	Dynamically simulates an arbitrary number of targets for the multi-source acoustic simulators. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Schmidt	
6	**pBearingSim**	Low-fidelity, single target bearing estimator replacing the onboard acoustic array processing module **pBearingTrack**. Highly efficient and useful for multi-auv collaborative mission simulations, and for onboard bearing simulation in case of hydrophone array malfunctioning.. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Eickstedt	
7	**pMultiAcousticSim**	Medium-fidelity acoustic simulator generating timeseries received on array simulated by **pArraySim**, incorporating ambient noise and signals from targets simulated by **pMultiTargetSim**. Uses simple white-noise model, and target signals simulated as plane waves, but with cylindrical spreading loss and bottom loss included. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Cockrell Schmidt	
8	**pGPSSim**	Simulates received GPS information when vehicle is surfaced. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Schmidt	
9	**uCTDSim2**	Simulates CTD data by interpolating in HOPS-generated virtual ocean or real ocean database. Can be used in field with malfunctioning CTD unit. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Lum	
10	**uBathy**	Simulates bathymetry data by interpolating in bathymetry table. Used for simulating altimeter data. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Lum	
Total unique lines of code Total aggregate lines of code				

Table 1: MIT AUV simulation environment. C++ simulation Modules.

#	Module Name	Module Description	Author	Size
1	**ArraySim.m**	Matlab version of array dynamics simulator	Dumortier	
2	**SealabMultiSim.m**	High–fidelity acoustic simulator generating timeseries received on array simulated by **pArraySim**, incorporating ambient noise and signals from targets simulated by **pMultiTargetSim**. Uses SEALAB synthetic sonar environment for envuironmental acoustic modeling. *Libraries:*	Schmidt	
3	**PassiveTgtSim.m**	GUI for adding acoustic sources simulated by **pMultiTarget-Sim**. The source numbering is sequential and transparent to the user. *Libraries:*	Dumortier Petillo	
4	**small_uVis.m**	Graphical display of AUV mission, including array dynamics, and processed acoustic data. Applied on each simulated platform. May also be used in conjunction with **uPlayback** for replay of actual at-sea missions. *Libraries:*	Schmidt	
Total unique lines of code Total aggregate lines of code				

Table 2: MIT AUV simulation environment - MATLAB modules.

2.2.1 pAntler Simulation MOOS Block

The adaptation of a mission file to the simulation environment is straightforward, involving only a few changes and additions. Thus, apart for defining platform dependent file paths etc., the actual real-time mission file is to be modified in the `pAntler` block, simply adding the start-up of the relevant simulation utilities, and in the configuration block for the back-seat driver interface. The following shows a typical `pAntler` configuration block for an acoustic sensing mission, with two groups of simulation modules added. The first concerns the basic vehicle sensor and activator operation, including the auv dynamics simulator, the CTD and GPS sensor simulations, etc. The second group are payload sensing simulators, involving the dynamics of the array, and the simulation of the relevant environmental acoustics. In the example given here, the efficient medium-fidelity acoustic simulator is used. In cases where the MATLAB-based high-fidelity simulator is applied, the `SealabMultiSim.m` is executed after the pAntler start-up.

Listing 1 - An example of `pAntler` *configuration block for acoustic sensing simulation.*

```
 0 //------------------------------------------------------------------
 1 // Antler configuration  block
 2  ProcessConfig = ANTLER
 3  {
 4     MSBetweenLaunches = 500
 5 //Core MOOS processes
 6     Run = MOOSDB            @ NewConsole = false
 7     Run = pEchoVar          @ NewConsole = true
 8     Run = pHuxley           @ NewConsole = true
 9 // AUV Simulator processes
10     Run = uMVS_Bluefin      @ NewConsole = false
11     Run = pMarinePID        @ NewConsole = true
12     Run = iModemSim         @ NewConsole = true
```

```
13     Run = pGPSSim           @ NewConsole = true
14     Run = uCtdSim2          @ NewConsole = true
15     Run = uBathy            @ NewConsole = true
16 // Array Dynamics and Acoustic simulators
17     Run = pMultiTargetSim   @ NewConsole = true
18     Run = pArraySim         @ NewConsole = true
19     Run = pMultiAcousticSim @ NewConsole = true
20 // Communication Processes
21     Run = iMicroModem       @ NewConsole = true
22     Run = pAcommsHandler    @ NewConsole = true
23     Run = pGeneralCodec     @ NewConsole = true
24     Run = pNaFCon   @ NewConsole = true
25     Run = pTransponderAIS   @ NewConsole = true
26 // Payload Processes
27     Run = iDAS              @ NewConsole = true
28     Run = pAEL              @ NewConsole = true
29     Run = pSearch           @ NewConsole = true
30     Run = pBearingTrack     @ NewConsole = true
31     Run = p1HTracker        @ NewConsole = true
32     Run = pTrackQuality     @ NewConsole = true
33 // Autonomous Control
34     Run = pClusterPriority  @ NewConsole = true
35     Run = pHelmIvP          @ NewConsole = true
36 // Logging
37     Run = pLogger           @ NewConsole = false
38 // Initial DEPLOY
39     Run = pMessageSim       @ NewConsole = true
40 // Watchdog
41     Run = uProcessWatch     @ NewConsole = true
42 // Display and Monitoring
43     Run = uMS               @ NewConsole = false
44 }
```

2.2.2 pHuxley Simulation MOOS Block

In addition to the dedicated simulation modules, the back-seat driver interface, for the Bluefin vehicles `pHuxley` must be configured for simulation, allowing it to interact with the vehicle dynamics simulator, rather than the actual vehicle 'front-seat driver'.

Listing 2 - An example of `pHuxley` *configuration block for AUV mission simulation.*

```
 0 //------------------------------------------------------------
 1 ProcessConfig = pHuxley
 2 {
 3   AppTick   = 4
 4   CommsTick = 4
 5   VarNamePrefix = UNICORN
 6 // Huxley IP address & port
 7   HuxleyHost = localhost
 8   HuxleyPort = 29500
 9   HuxleySim = true
10   Verbose = true
11 }
```

2.2.3 iModemSim Setup

The first step is to set up a set of virtual serial ports, one for each 'virtual modem' for the 'virtual ocean' modem simulator `iModemSim`. This has to be done as *superuser*:

```
> su
```

```
Password: xxxxxxxx
> serial_loopback /dev/ttyLOOPA1 /dev/ttyLOOPA2 &
> serial_loopback /dev/ttyLOOPB1 /dev/ttyLOOPB2 &
> serial_loopback /dev/ttyLOOPC1 /dev/ttyLOOPC2 &
..
..
```

2.2.4 AUV Mission Launch

The next step is launching the AUV mission, as specified in in the mission file, here *Unicorn_DURIP_sim.moos*, including a simulation `pAntler` block as described above:

```
> cd ~/moos-ivp-local/missions/AUVs/Unicorn_sim
> pAntler  Unicorn_DURIP_sim.moos &
```

To start the node display, use

```
> cd ~/moos-ivp-local/src/matlab/Macrura_uVis/
> matlab -nojvm < pianosa_9000.m
```

Note that this is the version using port 9000, and the operation box used in GLINT'08 at Pianosa Island, Italy. For other AUVs simulated on the same computer, a different port must be used, and the operation box obviously will be changed.

2.2.5 Topside Launch

The topside situational display and communication infrastructure, a MOOS community AUV_Topside, Port 9123, is launched using the commands:

```
> cd ~/moos-ivp-local/missions/AUV_Topside/
> pAntler AUV_Topside_base.moos &
> pAntler AUV_Topside_CommsSim.moos &
```

This will bring up the situational display using `pShipSideViewer`, with an example shown in Fig. 6.

The topside command and control GUI, shown in Fig. 7 is typically launched on a separate desktop using the commands:

```
> cd ~/moos-ivp-local/src/matlab/NaFConSimTop
> matlab
  >> NaFConSim
```

The GUI will publish the commands to the topside MOOSDB in the topside MOOS community AUV_Topside on Port 9123, and will be broadcast to the virtual network by the modem communication stack.

The final piece of the simulator is to launch the GUI activating acoustic sources for simulating targets and interferers. This is performed using the commands:

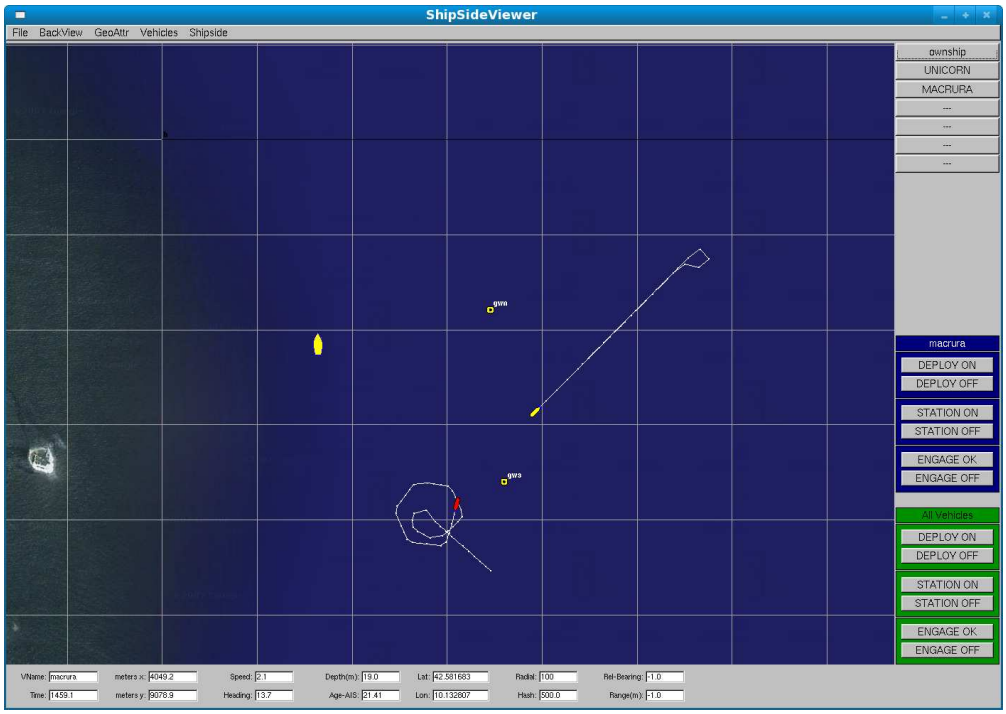

Figure 6: pShipSideViewer situational display

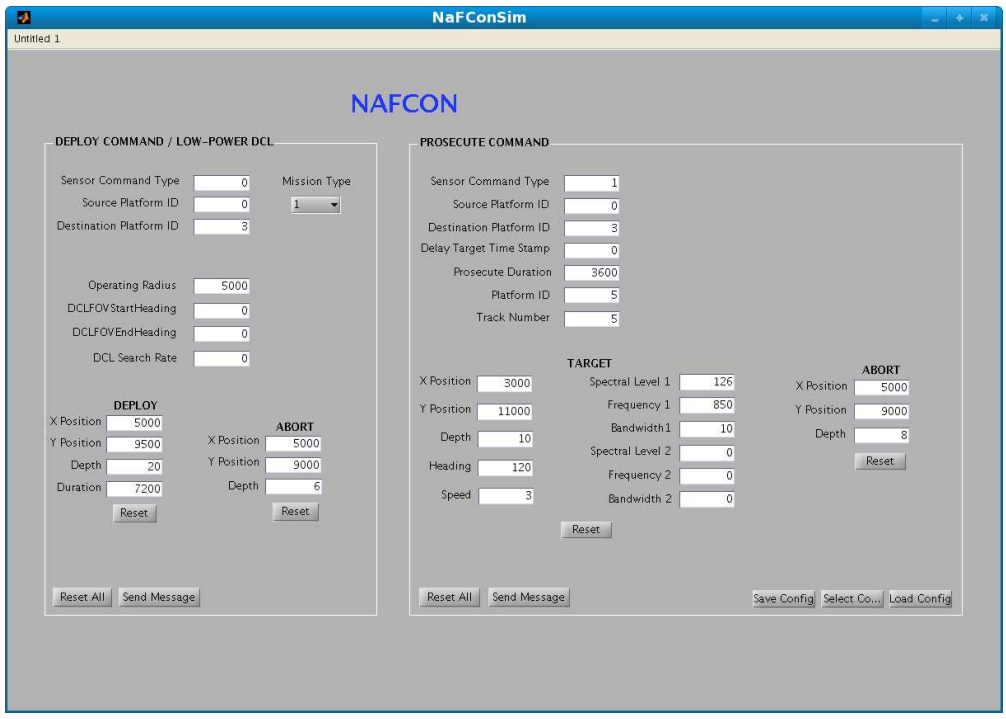

Figure 7: Network and Field Control (NaFCon) GUI for issuing Deploy and Prosecute commands to network nodes.

```
> cd ~/moos-ivp-local/src/matlab/PassiveTgtSim/
> matlab
  >> PassiveTgtSim
```

bringing up the source simulator GUI shown in Fig. 5. The `PassiveTgtSim` process connects to the topside `MOOS` community, and the target parameters are broadcast to all nodes in the virtual network via a special modem message.

3 Sonar AUV Simulation Modules and Utilities

3.1 pTargetSim

3.1.1 Brief Overview

This MOOS module simulates a single target moving along a straight-line path. It was the precursor to `pMultiTargetSim` and solely is used together with the low-fidelity target bearing simulator `pBearingsSim`. Thus, it cannot be used with any of the acoustic timeseries simulators, where `pMultiTargetSim` should be used.

3.1.2 Parameters for the pTargetSim Configuration Block

The following configuration parameters are defined for `pTargetSim`.

`rangeCrit:` Critical range beyond which target will be ignored.

Below is an example configuration block for the `pTargetSim`.

Listing 3.1 - An example `pTargetSim` *configuration block.*

```
 1  LatOrigin  = 42.3584
 2  LongOrigin = -71.08745
 3
 4 // ------------ pTargetSim Configuration block ----------
 5 ProcessConfig = pTargetSim
 6 {
 7   AppTick = 4
 8   CommsTick = 4
 9
10   rangeCrit     = 5000 // maximum source range in meters
11 }
```

The `LatOrigin` and `LonOrigin` parameters on lines 1-2 are not specific to `pTargetSim`, but are specified at the global level in a MOOS file. They are needed to allow the conversion between local x-y coordinates and latitude-longitude coordinates. They are mandatory parameters and `pTargetSim` will refuse to launch if they are lacking - but they are mandatory for other common MOOS applications as well.

3.1.3 MOOS variables subscribed to by pTargetSim:

.

MOOS variable	Type	Description	Published by
TARGET_CONTROL	$	"ON": Sources controlled by PassiveTgtSim.m, preample = T_ "OFF": Sources controlled by Prosecute message, preample = TARGET_	PassiveTgtSim.m
T_STATE TARGET_STATE	$	Comma-separated list of source parameters: x,y,depth,heading,speed,freq,bw,spl,delay,utc,numpsa	PassiveTgtSim.m pNaFCon
NAV_X	D	Ownship UTM x-coordinate	pHuxley
NAV_Y	D	Ownship UTM y-coordinate	pHuxley
DEPLOY_STATE	$	Deploy state	pNaFCon
DEPLOY_MISSION	$	Deploy mission type	pNaFCon
PROSECUTE_STATE	$	Prosecute state	pNaFCon
PROSECUTE_MISSION	$	Prosecute mission type	pNaFCon

3.1.4 MOOS variables published:by pTargetSim

.

MOOS variable	Type	Description	Format
TRACK_CONTROL	$	Target on/off flag. Set to ON in prosecute state and if deploy mission is 0.	"ON" / "OFF"
TARGET_RESET	$	Set to TRUE when new TARGET_STATE received	"TRUE" / "FALSE"
SIM_TARGET_XPOS	D	Current UTM x-coordinate of target.	2134.5
SIM_TARGET_YPOS	D	Current UTM y-coordinate of target.	5431.3

3.2 pBearingsSim

3.2.1 Brief Overview

This MOOS module provides a bearing estimate with gaussian noise for a single target simulated by `pTargetSim`. This is a low-fidelity bearing estimator which does not perform any sprocessing of array data, but simply computes the current bearing to the target and publishes the bearing state variable in the same format as the on-board signal processing modules. On the other hand, it is highly efficient and therefore well suited for developing adaptive and collaborative behaviors not sensitive to realistic environmental acoustic uncertainty.

3.2.2 Parameters for the pBearingsSim Configuration Block

The following configuration parameters are defined for `pBearingsSim`.

`detection_range:` Detection range for target in meters. If target is outside the tracking state is set to `TRACKING = NO_TRACK`. When targets enter the detection range the state is changed to `TRACKING = AMBIGUOUS`, and when the left-right ambiguity is resolved, to `TRACKING = TRACKING`.

`sigma:` Standard deviation of bearing estimate. used to make the bearing estimate more realisic in regard to array processing uncertainty.

Below is an example configuration block for the `pBearingsSim`.

Listing 3.2 - An example `pBearingsSim` *configuration block.*

```
 1  LatOrigin  = 42.3584
 2  LongOrigin = -71.08745
 3
 4 // ------------ pBearingsSim Configuration block ----------
 5 ProcessConfig = pBearingsSim
 6 {
 7   AppTick = 2
 8   CommsTick = 5
 9
10   detection_range = 1500 // Detection range in meters
11   sigma           = 1.0  // Bearing standard deviation in degrees
12 }
```

The `LatOrigin` and `LonOrigin` parameters on lines 1-2 are not specific to `pBearingsSim`, but are specified at the global level in a MOOS file. They are needed to allow the conversion between local x-y coordinates and latitude-longitude coordinates. They are mandatory parameters and `pBearingsSim` will refuse to launch if they are lacking - but they are mandatory for other common MOOS applications as well.

3.2.3 MOOS variables subscribed to by pBearingsSim:

.

MOOS variable	Type	Description	Published by
VEHICLE_ID	D	Node number for ownship.	pNaFCon
TRACK_CONTROL	$	Target on/off flag. Set to ON in prosecute state and if deploy mission is 0.	“ON” / “OFF”
TARGET_RESET	$	Set to TRUE when new TARGET_STATE received	“TRUE” / “FALSE”
SIM_TARGET_XPOS	D	Current UTM x-coordinate of target.	2134.5
SIM_TARGET_YPOS	D	Current UTM y-coordinate of target.	5431.3
NAV_X	D	Ownship UTM x-coordinate	pHuxley
NAV_Y	D	Ownship UTM y-coordinate	pHuxley
NAV_HEADING	D	Ownship true heading in degrees	pHuxley

3.2.4 MOOS variables published by pBearingsSim:

.

MOOS variable	Type	Description	Format
TRACKING	$	Tracking state. “NO_TRACK”: No detection yet “AMBIGUOUS”: Detection, but ambiguous bearing “TRACKING”: Un-ambiguous bearing tracking	
“CLASSIFYING”: Classifying sub-state			
BEARING_STAT	$	Comma-separated bearing state variables: vehID,state,bearing,x,y,beamno,sigma,utc	3,2,241,1000,2000,0,0.5,122..
IN_RANGE	$	Reset flag for geo trackers	“FALSE”
CLOSE_RANGE	$	Reset flag for geo trackers	“FALSE”

The principal output of `pBearingsSim` is the MOOS variable BEARING_STAT, which containes the current bearing state variables:

State variable	Description
vehID	Ownship VEHICLE_ID, e.g. 3 for Unicorn
state	Tracking state: 0: no detect, 1: ambiguous bearing; 2: Unambiguous bearing track.
bearing	Absolute, true bearing to target in degrees.
x	Current UTM x-coordinate of ownship
y	Current UTM y-coordinate of ownship
beamno	Beam bumber. Set to 0.
sigma	Bearing uncertainty
utc	UTC time.

3.3 pArraySim

3.3.1 Brief Overview

This MOOS process models the dynamics of a multi-sectored towed array in response to the motion of the tow-point. In addition to computing and publishing in the MOOSDB the hydrophone positions, it also publishes the tension on the tow-point, a parameter which is then subscribed to by the model for the vehicle dynamics. This connectivity results in a high-fidelity modeling of the coupled platform/array system.

3.3.2 Parameters for the pArraySim Configuration Block

The cable configuration and hydrophone separations are defined in the configuration (.moos) file. The following configuration parameters are defined for `pArraySim`.

`cond_check`: Flag for checking the condition number for Jacobian computed for the array dynamics updates. Should be set to 1 to avoid unstable solutions. If condition number exceeds 10^6 the array will be re-initialized to steady-state solution with current AUV heading. This can happen in cases of sharp turns and depth changes. Remedied by increasing AppTick.

`start_speed`: Minimum speed for which array dynamics model will be initiated.

`creat_dir_frame`: Flag identifying whether to write the frame number variables `VSA_DIR` and `VSA_FRAME` to the frame file. If set to zero this step will be done by the acoustic simulation module, which is the normal situation.

`write_out_files`: Flag identifying whether to write the non-acoustic data to the file identified by `VSA_DIR` and `VSA_FRAME`. Under normal circumstances it should be set to 0, since the acoustic simulators perform this step.

`filter_length`: Number of time samples used to achieve an average value of the towing speed and the cable tension, which provides a stable cable solution.

`array_type`: Array type. 1: NUWC VSA array, 2: MIT DURIP array

`number_sectors`: Number of array sections, each of which has homogeneous properties as defined in the `SECTOR_#` parameters.

`sector_#`: Comma-separated list of 7 properties characterizing array sector number #.

- Nsub Number of subdevisions applied in the numerical model for this sector.
- Len length of sector in meters.
- Wgt Weight of cable in N/m.
- Dia Diameter of cable in m.
- Td Tangential drag coefficient.(along cable)
- Nd Normal drag coefficient (perpendicular to cable)

E Elastic modulus for array section in N/m^2

`tow_body`: Flag indicating if towbody exists (1) or not (0).

`towbody_parameters`: Comma-separated list of 3 properties the tow body

Wgt Weight of tow body in N

Dcoef Drag coefficient for towbody

Area Cross-sectional area of towbody.

`array_section_number`: cable section containing acoustic array

`first_phone_offset`: Off set in meters of the first hydrophone in array section.

`phone_spacing_groups`: Number of array element spacing groups. For group number # the number of spacings is specified by the variable `NUM_SPACINGS_GROUP_#`, and the spacing by `SPACING_GROUP_#`

`spacing_group_#`: Array element spacing for group number #.

`num_spacings_group_#`: Number of hydrophone spacing intervals for group number #. Note tha the total number of spacings in the array must equal the total number of array elemnts minus one.

Below is an example configuration block for `pArraySim`, here configured for the MIT DURIP towed array.

Listing 3.3 - An example `pArraySim` *configuration block (MIT DURIP towed array).*

```
 0 //--------- pArraySim Configuration block -----------------
 1 ProcessConfig = pArraySim
 2 {
 3   AppTick =   2
 4   CommsTick = 5
 5
 6   cond_check = 1
 7   start_speed  =   0.5
 8   create_dir_frame = 0
 9   write_out_files  = 0
10   filter_length   = 40
11   array_type = 2
12   number_sectors = 3
13 //          Nsub, Len       Wgt        dia       td     nd,     E
14   sector_1 = 20,   20.0,    .0000,     .0095,   .001,   0.1,  1.6e9
15   sector_2 = 30,   36.0,    .0000,     .035,    .0024,  0.5,  1.6e9
16   sector_3 = 20,   20.0,    .0000,     .0095,   .001,   0.1,  1.6e9
17   tow_body = 0
18 //                    Wgt    Dcoef    Area
19   towbody_parameters = 0.0,   0.0,     0.0
20   array_section_number = 2
21   first_phone_offset   = 3.0
22   phone_spacing_groups = 3
23   spacing_group_1      = 1.5
24   num_spacings_group_1 = 5
25   spacing_group_2      = 0.75
26   num_spacings_group_2 = 22
27   spacing_group_3      = 1.5
28   num_spacings_group_3 = 4
29 }
```

3.3.3 MOOS variables subscribed to by pArraySim:

.

MOOS variable	Type	Description	Published by
VSA_DIR	$	Directory containing files with acoustic and non-acoustic array data	iVSA
VSA_FRAME	D	Frame number for acoustic and non-acoustic data files in VSA_DIR	iVSA
VEHICLE_ID	D	Node number for ownship. Used for selecting dynamic parameters	pNaFCon
TOW_POS_X	D	UTM x-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Y	D	UTM y-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Z	D	UTM z-coordinate of array cable tow point.	uMVS_Bluefin
TOW_VEL_X	D	x-velocity in m/s of array cable tow point.	uMVS_Bluefin
TOW_VEL_Y	D	y-velocity in m/s of array cable tow point.	uMVS_Bluefin
TOW_VEL_Z	D	z-velocity in m/s of array cable tow point	uMVS_Bluefin

3.3.4 MOOS variables published by pArraySim:

.

MOOS variable	Type	Description	Format
CABLE_TENSION	D	Tension of tow cable in N at tow point	
ARRAY_X	$	x-position of hydrophones in m relative to tow-point	17.23,18.33, ...
ARRAY_Y	$	y-position of hydrophones in m relative to tow-point	-6.13,-6.33, ...
ARRAY_Z	$	z-position of hydrophones in m relative to tow-point	0.01,0.02, ...
ARRAY_PITCH	$	pitch in degrees of array elements	0.01,0.00, ...
ARRAY_HEADING	$	true heading in degrees of array elements	301.01,300.99, ...
CABLE_SHAPE_X	$	x-position of cable elements in m relative to tow-point	0.43,0.84, ...
CABLE_SHAPE_Y	$	y-position of cable elements in m relative to tow-point	-0.10,-0.21, ...
CABLE_SHAPE_Z	$	z-position of cable elements in m relative to tow-point	0.00,0.00, ...
'auv'_AEL_HEADING	D	average true heading in degrees of acoustic array section. 'auv' is defined by VEHICLE_ID, e.g UNICORN.	
'auv'_AEL_PITCH	D	average pitch in radians of acoustic array section	

3.4 pMultiTargetSim

3.4.1 Brief Overview

This MOOS module simulates an arbitrary number of acoustic sources moving along straight-line paths. The sources are activated one at a time using the MATLAB GUI `PassiveTgtSim.m`. `pMultiTargetSim` will then keep a record of all sources within a critical distance from the vehicle, specified in the MOOS-block, and generate the MOOS variables needed by the acoustic timeseries simulators `pMultiAcousticSim` and `SealabMultiSim.m`. Note that the initiation of targets may be performed using the `PassiveTgtSim.m` GUI, or through an issue Prosecute command, dependent on the value of the MOOS variable TARGET_CONTROL. In general `PassiveTgtSim.m` will be used, but the control by the Prosecute command is useful for distributed virtual experiments, where missions may be commanded from a control center in another part of the world. Work is in progress on centralizing `PassiveTgtSim.m`, using a dedicated, broadcast modem message to activate sources simultaneously on all vehicles involved in such virtual experiments, which will make the Prosecute 'hack' to become obsolete.

3.4.2 Parameters for the pMultiTargetSim Configuration Block

The following configuration parameters are defined for `pMultiTargetSim`.

`rangeCrit:` Critical range beyond which target will be ignored.

Below is an example configuration block for the `pMultiTargetSim`.

Listing 3.4 - An example `pMultiTargetSim` *configuration block.*

```
 1  LatOrigin  = 42.3584
 2  LongOrigin = -71.08745
 3
 4 // ------------ pMultiTargetSim Configuration block ----------
 5 ProcessConfig = pMultiTargetSim
 6 {
 7   AppTick = 4
 8   CommsTick = 4
 9
10   rangeCrit     = 5000 // maximum source range in meters
11 }
```

The `LatOrigin` and `LonOrigin` parameters on lines 1-2 are not specific to `pMultiTargetSim`, but are specified at the global level in a MOOS file. They are needed to allow the conversion between local x-y coordinates and latitude-longitude coordinates. They are mandatory parameters and `pMultiTargetSim` will refuse to launch if they are lacking - but they are mandatory for other common MOOS applications as well.

3.4.3 MOOS variables subscribed to by pMultiTargetSim:

.

MOOS variable	Type	Description	Published by
TARGET_CONTROL	$	"ON": Sources controlled by PassiveTgtSim.m, preample = T_ "OFF": Sources controlled by Prosecute message, preample = TARGET_	PassiveTgtSim.m
T_STATE TARGET_STATE	$	Comma-separated list of source parameters: x,y,depth,heading,speed,freq,bw,spl,delay, tn1_frew,tn1_dbl,...,tn5_freq,tn5_dbl,time,number	PassiveTgtSim.m pGeneralCodec pNaFCon

3.4.4 MOOS variables published by pMultiTargetSim:

.

MOOS variable	Type	Description	Format
TARGET_SIM_LIST	$	Comma-separated list of source numbers within critical distance	1,2,3,6
TGT_#_SIM_STATE	$	Comma-separated list of source parameters for source number # #,on/off,utc,x,y,speed,heading,depth, spl,freq,bw,delay,tn1_freq,tn2_dbl,..	2,1,122...,3000,2100,..,0,850,120

3.5 pMultiAcousticSim

3.5.1 Brief Overview

This medium-fidelity acoustic timeseries simulator uses a plane wave model of the acoustic propagation from the source to the array, but incorporates cylindrical spreading and bottom loss. Highly efficient when high-fidelity acoustics is not needed.

3.5.2 Parameters for the pMultiAcousticSim Configuration Block

The following configuration parameters are defined for `pMultiAcousticSim`.

`water_depth`: Local water depth used for defining transition from spherical to cylindrical sprading loss.

`water_c`: Sound speed in water (m/s).

`bottom_loss`: Estimate of average transmission loss due to bottom interaction. Specified in dB/km.

`noise_level`: Noise level in dB.

`array_type`: Array type, defining file format. Options are DURIP, VSA, SINGLE, or DOUBLE.

`num_array_elements`: Number of hydrophone elements in array.

`sampling_frequency`: Sampling frequency for synthesized time series in Hz.

`number_samples`: Number of samples per file.

`hydrophone_gain`: Conversion factor to dB rel. 1 Pa

`base_dir_name`: Path for directory to contain acoustic and non-acoustic data files.

`acoustic_filename_prefix`: Prefix for acoustic data file names

`acoustic_filename_suffix`: Extension for acoustic data file names

`nonacoustic_filename_prefix`: Prefix for non-acoustic (sensor geometries) data file names

`nonacoustic_filename_suffix`: Extension for non-acoustic (sensor geometries) data file names

Below is an example configuration block for the `pMultiAcousticSim`.

Listing 3.5 - An example `pMultiAcousticSim` *configuration block.*

```
1 // ------------ pMultiAcousticSim Configuration block ----------
2 ProcessConfig = pMultiAcousticSim
3 {
4   AppTick =   12
5   CommsTick = 5
6 // Source Parameters
7 // Environmental Parameters
```

```
 8    water_depth = 100             // used for spherical spreading (m)
 9    water_c = 1500                // Speed of sound in water (m/s)
10    bottom_loss = 5.0             // Bottom loss estimate dB/km
11    noise_level = 60              // dB re 1 uPa
12 // Array and sampling parameters
13    array_type = DURIP            // DURIP or VSA
14    num_array_elements = 32       // number of elements in the array
15    sampling_frequency = 4000     // Hz
16    num_samples = 8000            // number of samples per file
17    hydrophone_gain = 1000000000  // conversion from Pascals to float
18 // Output file location and name
19 // absolute path including initial and final forward slash.
20    base_dir_name = /home/henrik/DURIP/
21 // The output filename will be:
22 // [filename_prefix][frame_number][filename_suffix]
23    acoustic_filename_prefix = ACO
24    acoustic_filename_suffix = .DAT
25    nonacoustic_filename_prefix = NAS
26    nonacoustic_filename_suffix = .DAT
27 }
```

3.5.3 MOOS variables subscribed to by pMultiAcousticSim:

.

MOOS variable	Type	Description	Published by
TOW_POS_X	D	UTM x-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Y	D	UTM y-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Z	D	UTM z-coordinate of array cable tow point.	uMVS_Bluefin
ARRAY_X	$	x-position of hydrophones in m relative to tow-point	pArraySim
ARRAY_Y	$	y-position of hydrophones in m relative to tow-point	pArraySim
ARRAY_Z	$	z-position of hydrophones in m relative to tow-point	pArraySim
ARRAY_PITCH	$	pitch in radians of array elements	pArraySim
ARRAY_HEADING	$	true heading in degrees of array elements	pArraySim
TARGET_SIM_LIST	$	Comma-separated list of source numbers within critical distance	pMultiTargetSim
TGT_#_SIM_STATE	$	Comma-separated list of source parameters for source number # #,on/off,utc,x,y,speed,heading,depth,spl,freq,bw	pMultiTargetSim

3.5.4 MOOS variables published by pMultiAcousticSim:

.

MOOS variable	Type	Description	Format
TARGET_XPOS	D	UTM x position for current target for plotting by `small_uVis.m`	
TARGET_YPOS	D	UTM x position for current target for plotting by `small_uVis.m`	

3.5.5 pMultiAcousticSim Details

pMultiAcousticSim models acoustic sources that have a flat spectrum in addition to sinusoidal tones. pMultiAcousticSim takes into account spreading loss (spherical to the waveguide depth, cylindrical thereafter), as well as bottom loss. The waves impinging upon the hydrophones are cylindrical waves originating from the sources' respective locations. Because the modeled waves are cylindrical, the depth of each hydrophone is irrelevant.

pMultiAcousticSim has the following shortcomings:

- Some approximations are made to simplify the math. These approximations will make the transmission loss inaccurate for distances less than one waveguide depth.
- The synthesized time series is discontinuous across files.
- For files generated for the VSA, the time stamp in the acoustic file is zero.
- pMultiAcousticSim will write a file if it has been longer than `number_samples` divided by `sampling_frequency` since the last file. It will not try to make up for "lost time" because this is not desirable in all cases (i.e. when the hydrophone or source position(s) stop being published to the MOOSDB, and then start again). If the processing chain relies on a file being published at very precise intervals, pMultiAcousticSim will have to be modified.

3.6 pGPSSim

3.6.1 Brief Overview

This module is used for simulating GPS fixes during surfacing. Used e.g. in connection with BHV_PeriodicSurface behavior.

3.6.2 Parameters for the pGPSSim Configuration Block

The following configuration parameters are defined for `pMultiTargetSim`.

`gps_depth`: Depth above which GPS is assumed to be active.

`gps_interval`: Time between GPS fixes in seconds

`min_gps_time`: Minimum time spent on surface for GPS fixes in seconds

Below is an example configuration block for the `pMultiTargetSim`.

Listing 3.6 - An example `pMultiTargetSim` *configuration block.*

```
1 ProcessConfig = pGPSSim
2 {
3    AppTick = 1
4    CommsTick = 1
5     gps_depth = 0.5        // Depth at which GPS becomes active
6     gps_interval = 2       // Interval between gps fixes in seconds
7     min_gps_time = 10      // Minimum surface time for achieving GPS
8 }
```

3.6.3 MOOS variables subscribed to by pGPSSim:

.

MOOS variable	Type	Description	Published by
NAV_X	D	UTM x-coordinate of vehicle.	pHuxley
NAV_Y	D	UTM y-coordinate of vehicle.	pHuxley
NAV_DEPTH	D	Vehicle depth in meter.	pHuxley

3.6.4 MOOS variables published by pGPSSimpin:

.

MOOS variable	Type	Description	Format
SECS_TO_DIVE	D	Time in seconds remaining before diving	
GPS_UPDATE_RECEIVED$		GPS 'measurement'	Timestamp=122.., Latitude=42.12345, Longitude=10.98765

3.7 uCtdSim2

3.7.1 Brief Overview

This module is used for simulating CTD data using HOPS-generated virtual ocean or real ocean database.

3.7.2 Parameters for the uCtdSim2 Configuration Block

The following configuration parameters are defined for `uCtdSim2`.

`Mods_Bin_File:` Name of binary file containing CTD database.

Below is an example configuration block for the `uCtdSim2`.

Listing 3.7 - An example `uCtdSim2` *configuration block.*

```
1 ProcessConfig = uCtdSim2
2 {
3    AppTick   = 5
4    CommsTick = 5
5
6    Mods_Bin_File = ../../../data/may07_ctdsim2.bin
7 }
```

3.7.3 MOOS variables subscribed to by uCtdSim2:

.

MOOS variable	Type	Description	Published by
NAV_LONG	D	Longitude of vehicle in degrees.minutes.	pHuxley
NAV_LAT	D	Latitude of vehicle in degrees.minutes.	pHuxley
NAV_DEPTH	D	Depth of vehicle in meter.	pHuxley

3.7.4 MOOS variables published by uCtdSim2:

.

MOOS variable	Type	Description	Format
CTD1	$	Message concatenating published and subscribed fields	
CTD_SOUND_VELOCITY	D	Sound velocity in m/s given logitude, latitude and depth of vehicle	
CTD_TEMPERATURE	D	Temperature in degree celsius given logitude, latitude and depth of vehicle	
CTD_SALINITY	D	Salinity in PSU given logitude, latitude and depth of vehicle	

3.8 uBathy

3.8.1 Brief Overview

This module is used for simulating bathymetry data using bathymetry table.

3.8.2 Parameters for the uBathy Configuration Block

The following configuration parameters are defined for `uBathy`.

`Bathy_Bin_File:` Name of binary file containing bathymetry table.

Below is an example configuration block for the `uBathy`.

Listing 3.8 - An example `uBathy` *configuration block.*

```
1 ProcessConfig = uBathy
2 {
3    AppTick   = 5
4    CommsTick = 5
5
6    Bathy_Bin_File = /home/raylum/project-plusnet/data/monterey.xyz.bin
7 }
```

3.8.3 MOOS variables subscribed to by uBathy:

.

MOOS variable	Type	Description	Published by
NAV_X	D	x position of vehicle in meter.	pHuxley
NAV_Y	D	y position of vehicle in meter.	pHuxley
NAV_Z	D	z position of vehicle in meter.	pHuxley

3.8.4 MOOS variables published by uBathy:

.

MOOS variable	Type	Description	Format
BATHY	$	Message concatenating published and subscribed fields	
BATHY_Z	D	Depth of water column given x and y positions of vehicle	

3.9 Arraysim.m

3.9.1 Brief Overview

This is the MATLAB version of the array dynamics simulator. The functionality of the two are identical, and they use identical MOOSDB interface definitions.

3.9.2 Configuration Files

The array information is defined in the configuration file config.txt: For description of the significance of each parameter, see the description for the equivalents for `pArraySim`

Listing 3.9 - An example `pArraySim` *configuration block (MIT DURIP towed array).*

```
1  moos_file       Array.moos
2  moos_name       ArraySim
3  cond_check 1
4  start_speed     0.5
5  create_dir_frame 0
6  write_out_files  0
7  filter_length   40
```

and a cable definition file, which for the MIT DURIP array is cable_durip.dat. Again the significance of the parameters is identical to the `pArraySim` equivalents.

```
0  Parameters for DURIP array  with 3 sections (tow cable+acoustic section+drogue)
1   3
2  20  20.0     .0000     .0095   .001     0.2     1.6e9
3  30  36.0     .0000     .035    .0024    0.3     1.6e9
4  20   20.0     .0000     .0095   .001     0.2     1.6e9
5  0.0 0.0 0.0
```

The remaining parameters, such as those defining hydrophone spacing etc. are specified internally, based on the MOOS variable VEHICLE_ID, which for DURIP should be 3 (Unicorn). This in contrast to the C++ version `pArraySim` which is entirely generic, with all variables specified in the configuration file.

3.9.3 MOOS variables subscribed to:

.

MOOS variable	Type	Description	Published by
VSA_DIR	$	Directory containing files with acoustic and non-acoustic array data	iVSA
VSA_FRAME	D	Frame number for acoustic and non-acoustic data files in VSA_DIR	iVSA
VEHICLE_ID	D	Node number for ownship. Used for selecting dynamic parameters	pNaFCon
TOW_POS_X	D	UTM x-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Y	D	UTM y-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Z	D	UTM z-coordinate of array cable tow point.	uMVS_Bluefin
TOW_VEL_X	D	x-velocity in m/s of array cable tow point.	uMVS_Bluefin
TOW_VEL_Y	D	y-velocity in m/s of array cable tow point.	uMVS_Bluefin
TOW_VEL_Z	D	z-velocity in m/s of array cable tow point	uMVS_Bluefin

3.9.4 MOOS variables published:

.

MOOS variable	Type	Description	Format
CABLE_TENSION	D	Tension of tow cable in N at tow point	
ARRAY_X	$	x-position of hydrophones in m relative to tow-point	17.23,18.33, ...
ARRAY_Y	$	y-position of hydrophones in m relative to tow-point	-6.13,-6.33, ...
ARRAY_Z	$	z-position of hydrophones in m relative to tow-point	0.01,0.02, ...
ARRAY_PITCH	$	pitch in radians of array elements	0.01,0.00, ...
ARRAY_HEADING	$	true heading in degrees of array elements	301.01,300.99, ...
CABLE_SHAPE_X	$	x-position of cable elements in m relative to tow-point	0.43,0.84, ...
CABLE_SHAPE_Y	$	y-position of cable elements in m relative to tow-point	-0.10,-0.21, ...
CABLE_SHAPE_Z	$	z-position of cable elements in m relative to tow-point	0.00,0.00, ...
'auv'_AEL_HEADING	D	average true heading in degrees of acoustic array section. The preample 'auv' is defined by VEHICLE_ID, e.g UNICORN.	
'auv'_AEL_PITCH	D	average pitch in radians of acoustic array section	

3.10 SealabMultiSim.m

3.10.1 Brief Overview

This is the MATLAB-based interface between the MOODDB and the Sealab sonor simulation environment. Sealab is a commercial product (VASA Associates Inc.), which uses state of the art lecacy propagation models and a native 3-D coupled mode model to produce high-fideklity element-level timeseries simulation for arbitrary 3D arrays and source configurations in a complex ocean model. It incorporates ocean wave-guide acoustic effects including mode coupling, doppler spread etc. It is sufficiently efficient for producing high-fidelity simuklated data in real time on a single platform.

3.10.2 Configuration MOOS-block

```
//.moos for SealabMultiTarget
ProcessConfig = SealabMultiTarget
{
    AppTick     = 10
    CommsTick   = 10
    Port        = COM6
    BaudRate    = 4800
    Verbose     = true
    Streaming   = false
    MOOSComms   = true          // Publish output to MOOSDB
    SerialComms = false         // Send output over serial line.
    SERIAL_TIMEOUT = 10.0
    SUBSCRIBE = VEHICLE_ID @ 0
    SUBSCRIBE = DB_TIME @ 0
    SUBSCRIBE = NAV_SPEED @ 0
    SUBSCRIBE = TOW_POS_X @ 0
    SUBSCRIBE = TOW_POS_Y @ 0
    SUBSCRIBE = TOW_POS_Z @ 0
    SUBSCRIBE = TOW_VEL_X @ 0
    SUBSCRIBE = TOW_VEL_Y @ 0
    SUBSCRIBE = TOW_VEL_Z @ 0
    SUBSCRIBE = ARRAY_X @ 0
    SUBSCRIBE = ARRAY_Y @ 0
    SUBSCRIBE = ARRAY_Z @ 0
    SUBSCRIBE = ARRAY_PITCH @ 0
    SUBSCRIBE = ARRAY_HEADING @ 0
    SUBSCRIBE = TARGET_SIM_LIST @ 0
    SUBSCRIBE = VSA_DIR @ 0
    SUBSCRIBE = VSA_FRAME @ 0
}
```

3.10.3 MOOS variables subscribed to:

.

MOOS variable	Type	Description	Published by
VEHICLE_ID	D	Node number for ownship. Used for selecting dynamic parameters	pNaFCon
TOW_POS_X	D	UTM x-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Y	D	UTM y-coordinate of array cable tow point.	uMVS_Bluefin
TOW_POS_Z	D	UTM z-coordinate of array cable tow point.	uMVS_Bluefin
TOW_VEL_X	D	x-velocity of array cable tow point. For doppler.	uMVS_Bluefin
TOW_VEL_Y	D	y-velocity of array cable tow point. For doppler.	uMVS_Bluefin
TOW_VEL_Z	D	z-velocity of array cable tow point. For doppler.	uMVS_Bluefin
ARRAY_X	$	x-position of hydrophones in m relative to tow-point	pArraySim
ARRAY_Y	$	y-position of hydrophones in m relative to tow-point	pArraySim
ARRAY_Z	$	z-position of hydrophones in m relative to tow-point	pArraySim
ARRAY_PITCH	$	pitch in radians of array elements	pArraySim
ARRAY_HEADING	$	true heading in degrees of array elements	pArraySim
VSA_DIR	$	Directory containing files with acoustic and non-acoustic array data	iVSA
VSA_FRAME	D	Frame number for acoustic and non-acoustic data files in VSA_DIR	iVSA
TARGET_SIM_LIST	$	Comma-separated list of source numbers within critical distance	pMultiTargetSim
TGT_#_SIM_STATE	$	Comma-separated list of source parameters for source number # #,on/off, utc, x, y, speed, heading, depth, spl, freq, bw	pMultiTargetSim

3.10.4 MOOS variables published:

.

MOOS variable	Type	Description	Format
TARGET_XPOS	D	UTM x position for current target for plotting by `small_uVis.m`	
TARGET_YPOS	D	UTM x position for current target for plotting by `small_uVis.m`	

3.11 PassiveTgtSim.m

3.11.1 Brief Overview

This MATLAB module uses the GUI shown in Fig. 5 for activating acoustic sources for the simulation environment. Thus, it is used for interactively activating targets to be simulated by the target dynamics simulator `pMultiTargetSim`.

3.11.2 Usage

The dynamic target simulator as well as the various-level fidelity acoustic simulators are always excuted on each sensor node. However, the activation of the acoustic target and interferrer sourecs may occur either locally in a simulated node `MOOS` community, or centrally, e.g in the topside moos community, The latter is important for syncronizing sources simulated on multiple nodes for testing collaborative tracking processing and behaviors. The centralized target control is made possible by defining the interface between the GUI and the target dynamics simulators compatibleusing a format which is compatible ith the generic messages Codec `pGeneralCodec`, such that the targets may be activated on all platforms in a synchronized manner via a madem transmission echoing the initial target status to all nodes. In addition to be used in entirely virtual experiments, this architecture allows for activating acoustic target simulators on actul, submerged nodes, e.g. in cases where equipment failure has made the sensing arrays unusable, thus allowing testing of collaborativ tracking behaviors even in such cases.

The echoing of the target initialization to all network nodes is activated by including the `XML` file `nafcon_targetsim.xml` in the configuration for `pGeneralCodec`, as described in Appendix **??**.

Note that the `MOOS` variable `TGT_STATE_OUT` is assumed to contain the target initialization message on the local node, while the `MOOS` variable receiving the message on the receiving nodes is `TGT_STATE_IN`. Thus, it will be necessary to echo these variables from and into the standard `MOOS` variable `T_STATE` published by `PassiveTgtSim`, using `pEchoVar`. In addition, the `MOOS` variable `TARGET_CONTROL` is echoed to all other platforms to activate the source control through `PassiveTgtSim`. Thus, it is important to at least once broadcast a target initialization before issuing a *Prosecute Command*, which woulk otherize activate the nodal source simulators.

3.11.3 Configuration MOOS-block for `PassiveTgtSim`

Listing 3.10 - An example `PassiveTgtSim` *configuration block.*

```
ProcessConfig = PassiveTgtSim
{
       AppTick     = 10
       CommsTick   = 10
       Port     = COM6
       BaudRate    = 4800
       Verbose     = false
       Streaming   = false
       MOOSComms = true           // true: use MOOSDB.
       SerialComms = false        // true: use serial communication
       SERIAL_TIMEOUT = 10.0
       SUBSCRIBE = DB_TIME @ 0
}
```

3.11.4 MOOS variables subscribed to:

.

MOOS variable	Type	Description	Published by
DB_TIME	D	MOOS time	MOOSDB

3.11.5 MOOS variables published:

.

MOOS variable	Type	Description	Format
TARGET_CONTROL	$	"ON": Sources controlled by PassiveTgtSim.m, preample = T_ "OFF": Sources controlled by Prosecute message, preample = TARGET_ . The control flag will be echoed to all platforms when used together with `pGeneralCodec`	
T_SOURCE	$	"ON": Source on "OFF": Source off	
T_STATE	$	Comma-separated list of source parameters with identifiers: position,heading,speed,freq,bw,spl,delay,tones,utc,tgt_num	tgt_x=2000, tgt_y=3000, tgt_depth=15, tgt_hdg=90, tgt_speed=4, tgt_freq=850, tgt_bw=100, tgt_spl=135, tgt_delay=0, tn1_freq=800, tn1_dbl=120, ... tn5_freq=0, tn5_dbl=0, tgt_utc=122345.., tgt_num=2

Notes: The parameters listed in the MOOS variable `T_STATE` are

`tgt_x`: UTM x-coordinate of acoustic source starting point

`tgt_y`: UTM y-coordinate of acoustic source starting point

`tgt_depth`: Depth of acoustic source

`tgt_hdg`: Heading of acoustic source in degrees

`tgt_speed`: Speed of acoustic source in m/s.

`tgt_freq`: Center frequency of broadband acoustic source

`tgt_bw`: Bandwidth in Hz of broadband acoustic source

`tgt_spl`: Spectral level of broadband source

`tgt_delay`: Delay in seconds of source transmission

`tn#_freq`: Frequency in Hz of tone number #. GUI allows for up to 5 tones which will be superimposed to broadband source signature.

`tn#_dbl`: Source level in dB of tone number #. GUI allows for up to 5 tones which will be superimposed to broadband source signature.

`tgt_utc`: UTC time of source starting location

`tgt_num`: ID number assigned to acoustic source. Will be automatically incremented for each new source activation.