# MOOS Meets Matlab — `iMatlab`

Paul Newman

March 17, 2009

## 1 Introduction

Not everyone wants to program in C++. Many folks are happy using Matlab as their research tool. Whilst not advising the use of Matlab to control a real vehicle, it seemed useful to build an application that allows Matlab to join a MOOS community – if only for listening in and rendering sensor data. The project `iMatlab` allows that to happen. In essence it "mex"s footnoteCompiles into a binary form Matlab can call directly. up some central MOOS code so it can be called from inside Matlab. The `CMake` build system supported by current releases of MOOS will build the project for the Linux or Windows version of Matlab.

### 1.1 Configuration

`iMatlab` allows Matlab programmers to access some of the benefits of MOOS. It allows connection to the MOOSDB and access to local serial ports. Configuration for the most part is done via a *.moos file which is either the default iMatlab.moos found locally or at a location specified at initialisation. Figure 1 shows a typical configuration block for `iMatlab` .

Most of the fields are understandable by reading the MOOS documentation. The application-specific fields are:

**MOOSComms** : "true" or "false" – do you want to connect to a community?

**SerialComms** : "true" or "false" – do you want to use serial ports ?

**SUBSCRIBE** : *VariableName @ Period* – one entry for each variable you want to subscribe to and the maximum update rate you are interested in. You can have many SUBSCRIBE lines.

Importantly, always call `iMatlab('init')` first — an error message is printed if you forget. By default iMatlab looks to read configuration data from iMatlab.moos. Alternatively you can use `iMatlab('init','CONFIG_FILE','XYZ.moos')` to read from the file "XYZ.moos". You can specify a process name other than

```
ProcessConfig = iMatlab
{
    AppTick       = 10
    CommsTick     = 10
    Port          = COM6
    BaudRate      = 4800
    Verbose       = false
    Streaming     = false
    MOOSComms = true
    SerialComms = false
    SERIAL_TIMEOUT = 10.0
    SUBSCRIBE = DB_TIME @ 0.0
}
```

Figure 1: A typical configuration block for `iMatlab`

the default "iMatlab" by passing the `MOOSName` parameter at initialisation: `iMatlab('init','MOOSNAME','MyName',.....)` .

## 1.2 Usage

If `MOOSComms` is "true" in the configuration file, then MOOS Comms functionality is enabled.

### 1.2.1 Publishing

To send data use the following syntax `iMatlab(ḾOOS_MAIL_TX;VARNAME,VARVAL)` e.g.
`iMatlab('MOOS_MAIL_TX','A_DATA_VALUE',10)` or
`iMatlab('MOOS_MAIL_TX','MY_NAME','PMN')`

### 1.2.2 Receiving Notifications

To receive data use the syntax `D = iMatlab('MOOS_MAIL_RX')` . This will return a structure array describing the data that has arrived (because of a subscription) since the last `'MOOS_MAIL_RX'` call . Each element of $D$ will be a structure with the following fields given in Table 1.

### 1.2.3 Registering for Notifications

This is done either through the configuration file using `SUBSCRIBE=...` or by calling `iMatlab('MOOS_REGISTER', VarName, MinTime)` For example, calling `iMatlab('MOOS_REGISTER', 'DESIRED_RUDDER', 0.0)` will subscribe to *every* change in `'DESIRED_RUDDER'` while calling `iMatlab('MOOS_REGISTER','DESIRED_RUDDER',0.2)` will subscribe in a way that means we'll only be told about changes in 'DESIRED_RUDDER' every 0.2 seconds.

| Field | type | Description |
|---|---|---|
| KEY | string | The name of the variable |
| TYPE | string | The type of the variable ('DBL'/'STR') |
| TIME | double | The time the data was valid |
| STR | string | The string value of the data if TYPE=='STR' |
| DBL | double | The double value of the data if TYPE=='DBL' |
| SRC | string | The name of the process that issued the data |
| ORIGINATING_COMMUNITY | string | The name of the community which SRC belongs to |

Table 1: The contents of a MOOS mail structure in iMatlab

## 1.3 Serial Ports

If "SerialComms = true" in the configuration file then serial port functionality is enabled.

### 1.3.1 Sending Data

Call `iMatlab('SERIAL_TX',Data)` where `Data` is a string or a vector of type uint8.

### 1.3.2 Reading Data

To receive data on a serial port call `D = iMatlab('SERIAL_RX',Data)` . If "Streaming=false" in the configuration file then the function will block until a timeout occurs or a telegram is received (ASCII, carriage return terminated only in the release). If `''Streaming = true''` the function returns immediately with a structure array containing all the telegrams received since the last call. Each element of $D$ is a structure described by Table 2.

| Field | Description |
|---|---|
| STR | The data Rx'd |
| TIME | The time it was received |

Table 2: The contents of the data structure pertaining to received serial data in `iMatlab`

## 1.4 Other Functionality

### 1.4.1 Pausing

Calling `iMatlab('MOOS_PAUSE',T)` suspends the calling thread (Matlab itself in this case) for T seconds. This is pretty useful for a non-busy wait in contrast to the CPU loading when calling Matlab's own `pause` function.