MACSimJX - MACSim with JADE extension, Pack A

18th January 2011, version 1.7

Author: C. R. Robinson

MACSimJX is based on MACSim (Multi-Agent Control for Simulink) developed by Dr. P. Mendham. It provides an extension enabling an interface between Simulink and JADE for modelling hardware run by software agents.

This Readme file provides instructions on how to setup MACSimJX and run an example of two signals being added and subtracted by agents, te new data being returned to Simulink.

Version History

Created September 2008.
Updated August 2009.
Updated January 2011 - Amended documentation and files for use with Matlab versions 2007, 2008 and 2009.
Last tested with JBuilder 2008 and MATLAB 7.9.0 (R2009b) with Simulink 7.4.

Requirements

In order to run MACSim with the JADE extension (MACSimJX) successfully, several assumptions are made.

You have pre-installed:

JDK	- Java Development Kit 32bit (Not JRE!)
MATLAB	
Simulink	
JADE	-version 3.6.1 is temporarily required, available at www.agentcontrol.co.uk
Borland C++	- a free compiling tool by Borland (bcc32).

(MACSim does not currently support compiling with VisualStudio C++, porting the code to make it compatible is in progress. Visual Studio is a free Microsoft tool for compiling C++ code.)

Section 0: Quick Overview

Running an example with MACSim takes place in three steps, provided you have set things up as detailed in the following sections. These steps also applies to your own models and agent task forces.

The steps are:
1. Set JADE main container running.
2. Run MACSimJX and select the Agent Task Force.
3. Open and run your model in Simulink.

Section 1: Installation

1. Move the contents of this zip folder to C:\macsimjx

2. Add MACSimJX path to MATLAB:
   In MATLAB select "File" and "Set Path" from drop down list. Add to this list the location: C:\macsimjx\MyTools\MACSim\client\bin

3. Ensure required software is installed correctly, particularly the c-compiler (see details for installing the required software below before continuing to Step 4).

4. Continue to Section 2.

Installing the required additional software

<u>JDK</u>:
To check if the jdk has been set up, typing 'javac' at command prompt provides some feedback about usage.

The system variable ‘Path’ should contain:
C:\Program Files\Java\jdk1.6.0_18\bin; D:\MATLAB701\bin\win32;
(among other things and depending on where you installed the software)

<u>JADE</u>:
In System properties, create a system variable called CLASSPATH with properties:
C:\JADE\lib\iiop.jar;C:\JADE\lib\http.jar;C:\JADE\lib\jadeTools.jar;C:\JADE\lib\jade.jar;

<u>Borland C++ Compiler</u>
The Borland compiler bcc32 may still be used for MATLAB versions (tested up to 2009), even though Borland is no longer officially supported by MATLAB. The installation steps and disk space required are significantly less than a Visual Studio installation. If you are not using MATLAB 2009, you will likely need to recompile the macsim s-function. To do this refer to Section 4: Version Support.

Visual Studio 2008 C++ Express Edition:

Not supported at present.

Section 2: Running MACSim and a straight-forward example

1. Open model:
   In MATLAB navigate to "C:\macsimjx\SimModel\" and open example.mdl.
   Before running the model, proceed with the next steps.

2. Running JADE:
   Set a main Jade environment container running:
   At command prompt enter "java jade.Boot -gui" (without quotes).
3. Running the MACSim jar file:
   Open a new command prompt, while the other is still running:

   Type: `cd C:\macsimjx`
   Type: `java -jar macsimjx.jar`

   This opens a GUI that allows you to choose the group of agents you wish to use (designated as the agent task force). The gui also provides the means of incorporating your own agent task force.
   Choose the ATF you wish to use:
   Simple agent example should be already selected.
   If you add customised ATFs, you can use the GUI to change the default selection.

4. Press continue. The following output should appear on the command prompt:

```
Num.: 1
Name: ssa
Path: example.SimpleSummationAgent
---
Num.: 2
Name: sda
Path: example.SimpleDifferenceAgent
---
MACSimInputNum is: 2
MACSimOutputNum is: 2
MACSimSampleStep is: 100
Setting up ssa Agent number 1
Setting up sda Agent number 2
End of agent creation
End of setup.
Setting up AgentServer
Create system pipes
```

```
MACSim Server: Running system "AgentServer"
Press enter to end service....
```

5. Example simulation:
   Return to the Simulink window with the model ‘Simple.mdl’ and start simulation.

6. On completion of simulation run, go the second command prompt to exit MACSim server by hitting the 'enter' key.

7. Return to Simulink and view scope data.

Section 3: Creating and running your own Agent Task Force

Adapting the example agents.
It is advised that you use the java files of the agents from the simple example as a template for generating your own agents. Open the folder ‘C:\macsimjx\ATFs\src\example’ and examine the java files for the two agents The points of interest. The important aspects that require your attention before developing your agents are discussed in the following paragraphs.

The following lines need to be altered to reflect which of the inports of the Simulink block this agent is to look at and which ports the results will be fed to.

```
int[] inputPortsOfInterest = {1};
int[] outputPorts = {1};
```

The following line can be altered to indicate the types of services that this agent offers. It is useful should an agent wish to send data to specific types of agents.

```
String[] services = {"arithmetic", "Agent"};
```

Under the line “Agent particular aspects begin here” you first extract the signals from Simulink contained in the array inputPortsOfInterest. If you have any calculations you wish the agent to perform on the signals before using information from other agents, do it here.

Under the line “Set information in dataStructure to be passed to other agents”, you only need to think about altering the first two lines. The array dataToShare is used to pass data to the other agents. The second array, called elementToChange, is used to indicate which ports the results of this agent are destined to be shown in Simulink.

Under the first instance of “Agent specific task inside this block”, the first two lines extract the data received from another agent, the rest of this block is up to the programmer to incorporate the information as desired.

Under the second instance of "Agent specific task inside this block", the agent can complete any calculations required now that all agents have communicated their information. The last two lines return the results of this agent, for the current time-step of Simulink, back to the specified port of the MACSim block in Simulink.

Incorporating your agent java files.
MACSimJX references the ATFs folder when looking for the agent class files. Thus the quickest way to set up your ATF is to do the following:

Create two new folders in the ATFs directory, one for the agent source code and one for the compiled classes, for example:

```
C:\macsimjx\ATFs\src\yournewATFpackage
C:\macsimjx\ATFs\classes\yournewATFpackage
```

Place the files for your java agent task force in 'src\yournewATFpackage'

Compile the java files.
N.B. Ensure your java files have 'package yournewATFpackage;' at the top of each java file.
Compiling can be done from command prompt and linking to the necessary jade and macsim libraries. Enter the following, changing the path to suit the name of your new folders:

```
'cd C:\macsimjx\ATFs\src\yournewATFpackage'
'javac -d C:\macsimjx\ATFs\classes\yournewATFpackage -classpath
..\..\..\lib\jade.jar;..\..\..\lib\macsim.jar;..\..\..\macsimjx.jar *.java'
```

This compiles the java files and places them in the classes folder. The lib folder contains all of JADE's libraries, so the above command may need to be extended if you are using methods from some of the other .jar files. Ensure when adding the .jar files above to separate using a semicolon and have no spaces.

The next step is to run your ATF. This follows the same procedure as described in Section 2. The difference is that you will open your own Simulink model and when the GUI opens the first time, you need to add your ATF with details for each agent to the list.

Further information can be found in the thesis associated with MACSimJX.

Section 4: Version Support

Borland Compiler (5.5)

Given the foundation of MACSim was developed in C++ using the Borland compiler, it is important to keep backward compatibility. In newer versions of MATLAB support for the Borland compiler has been dropped by Mathworks, however the Borland compiler can still be used.

Ensure you have followed the embarcado configuration steps for bcc32
(set your Path environment variable to include Bin location
e.g. C:\Borland\BCC55\Bin;)
Restart computer for this to take effect. Type bcc32 at command prompt to check it is installed (general help information should be printed).

For MATLAB 2006:
You have downloaded the wrong zip file, return to agentcontrol.co.uk and download the older version zip file.

For MATLAB 2009:
The files currently existing in this version of MACSim have already been compiled with MATLAB 2009, so you should be able to skip this part, however, if there are problems, follow the next instructions.

For MATLAB 2007, 2008, or above…

There are some MACSim files that need to be recompiled to be compatible for specific versions of MATLAB. Unless some kind soul has uploaded these compiled files and provided a link in the forums for their particular version, the following steps are necessary.

Now prepare Matlab, copy the file called mexopts.bat from
C:\macsimjx\MyTools\Legacy to:

C:\Documents and Settings\<User>\Application Data\MathWorks\MATLAB\<ver>

If a file already exists, replace it (or rename it), the file mexopts.bat is regenerated each time the "mex -setup" command is used in MATLAB. Now right-click and choose edit file. In this file, check the paths MATLAB, BORLAND and PERL in the general section are appropriate, or change where necessary.

Next, copy the folder 'borland' from 'C:\macsimjx\MyTools\Legacy' to 'C:\MATLAB\<ver>\extern\lib\win32', if it does not already exist.

Finally copy link_borland_mex.pl from the Legacy folder to 'C:\MATLAB\<ver>\bin\win32', if it does not already exist.

Often it may also be necessary to recompile some the .lib files used by the Borland compiler if these differ in version from the Matlab .dll files. In particular the libmx.lib (contained in C:\MATLAB\<ver>\extern\lib\win32\borland) may need to

be regenerated from the libmx.dll of Matlab (contained in C:\MATLAB\<ver>\bin\win32).
Copy the libmx.dll from the win32 directory and place it in the C:\macsimjx\MyTools\MACSim\client\bin folder.

While in this location, open createlib.bat (e.g. right click->edit) and check the perl.exe location is correct (e.g. C:\Program Files\MATLAB\R2009b\sys\perl\win32\bin\perl.exe)

At a command prompt:

```
Type:   cd C:\macsimjx\MyTools\MACSim\client\bin
Type:   createlib libmx
```

Using windows explorer, navigate to C:\MATLAB\R2009b\extern\lib\win32\borland\bc54 and change 'libmx.lib' to 'libmx.lib.old' (**not libmex.lib**!!).
Copy the newly created libmx.lib from (e.g.):
    C:\macsimjx\MyTools\MACSim\client\bin
To(e.g.):
    C:\Program Files\MATLAB\R2009b\extern\lib\win32\borland\bc54

Open 'My Computer' and browse to:
C:\macsimjx\MyTools\MACSim\jserver\bin
Right-click on the file 'make.bat' and click edit.
Scroll to the line: 'set JAVA_INC=' and replace the address here with the location of your installed jdk include folder.
Do the same for the next line, but with the additional extension '\win32' (the line begins 'set JAVA_WIN32_INC=')
Save file.

From windows command prompt (with bcc32):

```
Type:   cd C:\macsimjx\MyTools\MACSim\jserver\bin
Type:   make
```

The output should look like:

```
    Compiling Java files
    Generating Native header files
    Compiling C++ files
    Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
    ..\src.cpp\Server.cpp:
    ..\src.cpp\ConfigServer.cpp:
    ..\src.cpp\SimServer.cpp:
    ..\src.cpp\ServerUtil.cpp:
    Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
    Creating JAR file
    Done.
```

Finally, to recompile MACSim for the S-Block in MATLAB command prompt:
Type: cd C:\macsimjx\MyTools\MACSim\client\bin
Type: make

Should all the above have been done successfully, you will have a working macsim.dll for your version of Matlab.

Refer back to Section 2 and continue.

Visual Studio 2008 C++ Express Edition

Not supported at present. There are still missing stages in the following section for using Visual Studio, and are included for my own benefit. **Do not follow these steps unless you wish to assist with porting the code.**

The installation requirements are slightly different depending on your version of MATLAB:

For MATLAB 2006:
Use of Visual Studio is unadvised because it is not likely to be detected by MATLAB 2006.

For MATLAB 2009:
When I manage to compile the files, one should be able to skip to the next section assuming installation is fine.

For MATLAB 2007, 2008, 2009, or above…

There are some MACSim files that need to be recompiled to be compatible for specific versions of MATLAB. Unless some kind soul has uploaded these compiled files and provided a link in the forums for their particular version, the following steps are necessary.

Note: It may be sensible to switch the order of installation below, i.e. install MS SDK first and then Visual Studio afterwards.

Download vcsetup.exe from:
http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=f3fbb04e-92c2-4701-b4ba-92e26e408569

When installing you do not require the options for Silverlight or the SQL database.

Open MATLAB and visual studio should now be automatically detected. If it is not refer to the troubleshooting section.

Visual Studio must now be updated for 62 bit processing (since Matlab is migrating to 64 bit processing). To do this requires downloading and installing the MS SDK: Windows SDK for Windows Server 2008 and .NET Framework 3.5
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e6e1c3df-a74f-4207-8586-711ebe331cdc&displaylang=en

When installing the SDK:
Select the tools option, i.e. 524Mb, to avoid a 1.3Gb installation (Deselect Samples and Documentation).
Ensure that "Developer Tools"->"Visual C++ Compilers" is selected to get the "x64 Compilers and Tools". This item has the Feature Description "Install the Visual C++ 9.0 Compilers. These compilers allow you to target x86, x64, IA64 processor architectures." Allow installation to complete.

To verify that you have the needed components, check that the install location of MS SDK contains the "amd64" version of the C/C++ compiler "cl.exe", typically found at: C:\Program Files\Microsoft Visual Studio 9.0\VC\bin\x86_amd64 .

Rerun VS2008 and choose the repair option – Go to Control Panel->Select VS2008 and change/remove -> In setup wizard select Next -> Repair. This will update VS2008 with the relevant 64 bit compiler cl.exe

Configure VS2008 to run at command prompt by executing (double clicking):
C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat
(This adds environment variables)

Now configure MATLAB to use visual studio.
At the MATLAB command prompt,

type: mex -setup
and choose Visual Studio.

(If a non-default path is used to set up compiler in Matlab, following mex -setup choose -> no -> number for visual studio -> no -> yes -> provide path)

Edit the file mexopts.bat that is located at:
C:\Documents and Settings\User\Application Data\MathWorks\MATLAB\<ver>\
(where User is your windows account name, and ver is your MATLAB version)

Create a backup copy first if you wish.
In this file locate the line "set MATLAB=%MATLAB%", insert a new line after this and type:
set SIMCLASSLIB=C:\macsimjx\MyTools\MACSim\lib\SimClassLib
Find the line beginning set include and add ";%SIMCLASSLIB%\code\include", e.g.:
set
INCLUDE=%VCINSTALLDIR%\ATLMFC\INCLUDE;%VCINSTALLDIR%\INCLUDE;%LINKERDIR%\include;%SIMCLASSLIB%\code\include;%INCLUDE%

Find the line beginning set lib and add ";%SIMCLASSLIB%\code\bin\", e.g.:
set
LIB=%VCINSTALLDIR%\ATLMFC\LIB;%VCINSTALLDIR%\LIB;%LINKERDI
R%\lib;%VSINSTALLDIR%\SDK\v2.0\lib;%SIMCLASSLIB%\code\bin\;%MATL
AB%\extern\lib\win32;%LIB%

Find the line beginning "set LINKFLAGS" insert a new line after this and type:
set LINKFLAGS = %LINKFLAGS% sim.lib

The MACSim server library needs to be rebuilt, these files are:
macsim.cpp
server.cpp
thread.cpp

and

matrix.cpp
matrix_operator.cpp
matrix_access.cpp

which need to be converted to .lib files (since lib files are compiler
dependant and the ones provided are compiled with Borland)

Sidenote 1
In the file matrix.h changed iostream.h to iostream.

Open file at C:\macsimjx\MyTools\MACSim\server\bin\vs2008makefile.bat to and
check if call
"C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat" is the
correct location on your drive (change if necessary). Then execute it (double click) –
this will rebuild the library files for your particular (Visual Studio) compiler.

Sidenote 2
To run vs2008 compiler from normal cmd, the following batch file can be used:
C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat
this configures by default a x86, 32-bit compiler (args can be provided
for 64 bit amd64 etc.)

Open 'My Computer' and browse to:
C:\macsimjx\MyTools\MACSim\jserver\bin
Right-click on the file 'make.bat' and click edit.
Scroll to the line: 'set JAVA_INC=' and replace the address here with the
location of your installed jdk include folder.
Do the same for the next line, but with the additional extension '\win32'
(the line begins 'set JAVA_WIN32_INC=')
Save file.

Sidenote 3
The above file also requires the bcc32 call changed to instead call visual studio, not sure how this should be implemented yet.

From windows command prompt (with visual studio):

type: cd C:\macsimjx\MyTools\MACSim\jserver\bin
type: !!!(Not yet finished, the make file uses Borland, which needs to be changed to instead call visual studio.

Sidenote 4
In a similar manner as described for the Borland compiler, if it is found some other libraries need to be recompiled with visual studio, similar steps may be followed using the createlib.bat file. In such a case this file needs to be edited to comment the Borland section and uncomment the visual studio section.

Finally, to recompile MACSim for the S-Block, in MATLAB:
Type: cd C:\macsimjx\MyTools\MACSim\client\bin
Type: make

Section 5: Troubleshooting

This section addresses some of the common problems people have encountered. A forum is provided at www.agentcontrol.co.uk for more specific questions.

Question: **Why do I receive warning messages when I open my model which contains a large number of inlined parameters in Simulink 6.2 (R14SP2)?**
Problem Description:

My model contains a large number of parameters defined in the MATLAB workspace. I then select the option to inline the parameters for this model. However when I then reopen the model with the parameters still defined in the MATLAB workspace I receive a series of the following warning messages:

```
Warning: InlineModel_Error.mdl, line 1049: block_diagram does not have a parameter named 'ime'.
> In general\private\openmdl at 13
In open at 141
In uiopen at 181
```

Also most of the parameters names with the warning messages do not match with any of my model's parameters names.

Solution:

(according to mathworks.com support)

This bug has been fixed in Release 14 Service Pack 3 (R14SP3). For previous product releases, read below for any possible workarounds:

We have verified that there is a limitation in Simulink 6.2 (R14SP2) and below in the number of tuneable parameters that a block diagram can hold in its MDL file. To work around this issue, define your model's parameters as Simulink. Parameters objects. You can find more information on how to define your parameters as Simulink. Parameters objects in your Simulink documentation using the following URL:

http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/slref/simulink.parameter.shtml

Question: **I get "Agent AgentServer died without properly being terminated".**

Answer: This is likely to be related either to setting your paths incorrectly (ensure folders are in locations indicated above), or not compiling the MACSim library files correctly.

Question: **How do I solve "Error in S-function 'example/S-Function': S-Function 'MACSim' does not exist".**

Answer: This is most likely to occur either if you have not set the path correctly in MATLAB or have not executed the ‘make’ command to generate the appropriate macsim S-function for your version of MATLAB.
(see information above concerning Set path or refer to Section 4).

Question: When **I type ‘make’ I get many errors, including:**

Error E2209 ..\src.cpp\../include/macsim_internal_Server.h 2: Unable to open inc
lude file 'jni.h'

Answer: This is likely generated because the file ‘make.bat’ has not had the paths set correctly for your jdk. See relevant section above to fix this problem.

Question: **Visual Studio is not automatically detected by MATLAB.**

Answer: This may occur if a non-default path is used to set up Visual Studio. In MATLAB select "File" and "Set Path". Add to this “C:\VisualStudio2008\” (or your installed location, without quotes).
Then at the MATLAB command line, enter “mex -setup” and choose -> no -> select number for visual studio -> no -> yes -> provide path.

.

Question: **When trying to run make I get**
This is mex, Copyright 1984-2007 The MathWorks, Inc.
D:\PROGRA~1\MATLAB\R2009B\BIN\MEX.PL: Error: '..\src\macsim.cpp' not found.
??? Error using ==> mex at 221
Unable to complete successfully.
Error in ==> make at 2
mex -v ..\src\macsim.cpp ..\src\client.cpp "C:\macsimjx\MyTools\MACSim\lib\SimClassLib\code\bin\sim.lib" -largeArrays

Answer: You may have not updated the mexopts.bat file correctly.

Question: **My compiler is not detected when I type mex –setup.**

Answer 1: If you installed your compiler in a non-default location, you may need to provide the setup routine with the actual location of the compiler.

Answer 2: The compiler is not supported by Matlab. This does not indicate that a compiler will not work with Matlab, after some tweaking. Read the version support section for some tips.

Question: **I get the following message when running the simulation: Warning: Inconsistent sample times. Sample time ([0, 0]) of signal driving input port 1 of 'example/S-Function' differs from the expected sample time ([0.01, 0]) at this input port.**

Answer: This is a misplaced warning from Simulink – it has no effect on the simulation.

Question: **How do I resolve Error: Unresolved external 'functionx' e.g. '_mxCreateDoubleMatrix_700'.**

Answer: This indicates a disparity between the version of an associated .lib file and the MATLAB dynamic implementation (.dll). In other words a function is found in one and not the other. To fix this, the .lib file needs to be recompiled using the associated compiler (.lib files are compiler dependant) from the associated .lib file contained in Matlab (C:\MATLAB\R2009b\bin\win32). In the example provided, the function _mxCreateDoubleMatrix_700 is contained in the file libmx.dll, but is not in the libmx.lib contained in the Borland folder. Thus the libmx.lib file needs to be regenerated, using the steps provided the version support section.

Question**: I copied mex –setup to MATLAB and I get the error:**
D:\PROGRA~1\MATLAB\R2009B\BIN\MEX.PL: Error: '–setup' not found.

??? Error using ==> mex at 221
Unable to complete successfully.

Answer: Try typing the command in directly – the hyphens are not handled correctly.

Question: **JADE gui has stopped loading when using switch (jade.Boot -gui).**

Answer: Check your environmental paths.